Lab 2 – ECE311 Introduction to Linear Control Systems

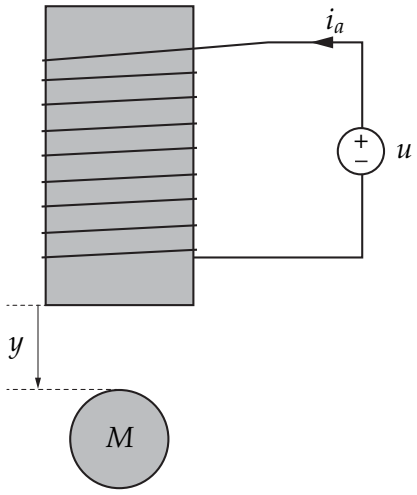
Control of a Magnetically Levitated Ball

Main concepts of this lab

- How to linearize nonlinear control systems in Matlab/Simulink
- How to check stability of LTI systems in Matlab
- Design of a lead controller for magnetic levitation of a steel ball

1 Introduction

In this lab you will learn how to define and linearize nonlinear control systems in Matlab/Simulink, and how to check the stability of the linearization. For this, we will investigate a basic magnetic levitation problem, whereby the current in an electromagnet is controlled so as to levitate a steel ball at a fixed distance from the bottom face of the electromagnet. The setup is depicted in the figure below.



The system can be modelled as an RL circuit representing the electromagnet winding, and a mass M subject to gravity and the force imparted by the electromagnet. The control input u is the voltage applied at the electromagnet terminals, while the output y is the distance of the ball from the bottom face of the

electromagnet. Denoting by R_a and L_a the resistance and inductance of the winding, the circuit equation is

$$L_a \frac{di_a}{dt} + R_a i_a = u. \tag{1}$$

The force imparted by the electromagnet on the steel ball, directed upward, is[1] $F_m = k_m i_a^2 / y^2$, where k_m is a positive constant. Newton's equation for the steel ball is

$$M\ddot{y} = -F_m + Mg = -k_m \frac{i_a^2}{y^2} + Mg, \tag{2}$$

where g is the acceleration due to gravity. The numerical values of various constants in the model are found in the table below. The model above is only valid if $y > 0$, i.e., if the steel ball is placed *below* the electromagnet.

Parameter	Description	Numerical value
L_a	Winding inductance	0.05 H
R_a	Winding resistance	3 ohms
M	Mass of the steel ball	0.1 Kg
k_m	Coefficient of electromagnetic force	0.1 N m^2/A^2
g	Acceleration due to gravity	9.81 m/sec^2

We define the state vector $x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^\top$, with $x_1 = y$, $x_2 = \dot{y}$, and $x_3 = i_a$, and from equations (1) and (2) we obtain the nonlinear control system

$$\begin{aligned} \dot{x} = f(x,u) &= \begin{bmatrix} x_2 \\ -\frac{k_m}{M}\frac{x_3^2}{x_1^2} + g \\ -\frac{R_a}{L_a}x_3 + \frac{1}{L_a}u \end{bmatrix} \\ y &= x_1. \end{aligned} \tag{3}$$

The control problem is to levitate the ball at a distance $\bar{y}$ metres ($\bar{y} > 0$) to the bottom face of the electromagnet. Our approach will be to linearize system (3) at the equilibrium corresponding to $\bar{y}$, then design a controller stabilizing the linearization, and finally apply this controller to the nonlinear model (3).

Note: This lab was developed using Matlab R2020b. If you use an earlier version of Matlab there is a very small chance that some of the instructions here might not work for your version, in which case you should update your software.

Throughout the lab, you will be guided through a number of steps which will require you to write Matlab code or draw Simulink diagrams. You will write your code in a Matlab script called `labx.m`, where `x` is the lab number. Your Simulink diagram will be saved as `labx.slx`. If there are multiple Simulink files, save them as `labx_1.slx,labx_2.slx`, and so on. You will submit this code as a group. Your code should provide certain outputs (figures, numbers, and the like). A request for output is highlighted with a shaded area of text, such as the following.

[1] This expression of the force F_m only makes sense if $y > 0$. If $y < 0$, F_m would be pointing downward, and we would need a negative sign in front of F_m.

Output 1. Print the poles of the transfer function.

Parts of the text containing directions for the writing of your Matlab code will be highlighted in a different colour, such as this:

Convert the state space model into a transfer function model, then find the poles.

2 Submission guidelines and mark breakdown

Marks are assigned to groups. Members of each group get identical marks, unless special circumstances occur. The group lab mark will be made up of three components.

Matlab code	6 pts
Presentation video	2 pts
Lab report	2 pts
Total	**10 pts**

Matlab code. Your code should be clean and readable, and it should contain abundant commentary, so that the instructors may follow your development and check its correctness. This component of the mark will be based on the correctness of the code and its readability, and it will be assigned as follows:

0 out of 6	the code is absent
1 out of 6	the code is largely incomplete
2 out of 6	the code is largely complete, but there are parts missing and the outputs are incorrect
3 out of 6	the code is complete, but it does not produce correct outputs
4 out of 6	the code is complete, it produces correct outputs, but there is no commentary in the code, and/or the code is poorly organized
5 out of 6	the code is complete, correct, and contains some but insufficient commentary, and/or its organization is below par
6 out of 6	the code is complete, correct, and the commentary and organzation are adequate so that it is easy to read

Presentation video. Once you've completed your lab code, you will submit a 5-6 minute video presentation of the code as a group. Each group member will present a different portion of the code. The objective of the presentation is to show that you understand what you did and why you did it. It's important that each group member contributes equally to the video. In the video, include any observations about the outputs you produced, any insight you derived from the lab steps.

0 out of 2	group does not submit a presentation, or the presentation is unintelligible
1 out of 2	the presentation is somewhat intelligible but does not display adequate understanding of the code and/or the lab document, or the group members give somewhat disconnected presentations
2 out of 2	the presentation is intelligible and displays an adequate understanding of the code and the lab document. The group members contribute equally to the presentation and their contributions are well-connected

Lab report. Write a concise report containing the requested outputs, but don't just print out a bunch of numbers and figures. Add some flesh to the outpus that are requested within the lab document so that one can follow the logical development of the lab. Aim for a style like this:

> **Output 1.** Below is a plot of the output signal that was obtained with this controller.
> (...)
> We observe that the output signal converges to a steady-state value of 25 rad/sec (...)
> **Output 2.** Tuning of the controller gain gave the following result, see the figure below illustrating the step response of the system. We observe that there is marked improvement in the transient performance of the control system, and indeed (...)

Do not screenshot Matlab figures. Rather, save them as jpeg files (or other formats) and include them in the report in the appropriate order with clear titles and axes labels.

When the lab document asks you to comment on something, strive to provide meaningful, insightful commentary. This portion of the mark will be assigned as follows:

0 out of 2	the report is either absent, or a simple printout of the Matlab outputs without commentary
1 out of 2	the report is incomplete and/or the commentary is inadequate
2 out of 2	the report is complete and the commentary is adequate

Late submissions. *All submissions are due at 8PM* of the day indicated on the course website. We *do not* accept late submissions under any circumstances. For more details, see the *late submission policy* on the course website.

3 Theoretical linearization

In this section, you will be guided through the steps required to determine the equilibrium of system (3) that corresponds to having $y = \bar{y}$. Later, you will verify your theoretical work in Matlab/Simulink.

Perform this theoretical work which you will include in the report as per the instructions below.

- By imposing $f(x, u) = 0$, with f given in (3), find the *unique* equilibrium $\bar{x} = \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \end{bmatrix}^\top$ of (3) and the *unique* control input $\bar{u}$ satisfying these two properties:

  (i) $\bar{x}_1 = \bar{y}$;

(ii) $\bar{x}_3 > 0$.

Your expressions of $\bar{x}$ and $\bar{u}$ will be parametrized by $\bar{y}$. Later we will choose a numerical value for this quantity. Also, they will be parametrized by the physical constants M, L_a, R_a, g, k_m, which for now you need to leave as symbolic parameters in the expressions.

- Define error variables $\tilde{x} = x - \bar{x}$, $\tilde{u} = u - \bar{u}$, and $\tilde{y} = y - \bar{x}_1$, and find the linearization of (3) at the equilibrium $\bar{x}$ with control $\bar{u}$ found above. The matrices A, B, C, D of the linearization will depend on $\bar{y}$ and the physical constants M, L_a, R_a, g, k_m.

Output 1.
- Write in your report the equilibrium $\bar{x} \in \mathbb{R}^3$ and the control $\bar{u}$ that you determined above. These quantities should contain $\bar{y}$, and M, L_a, R_a, g, k_m as parameters.
- Write the linearization of (3) at $\bar{x}$ with control $\bar{u}$, defining all error variables.
- Double-check carefully your work. Verify that $f(\bar{x}, \bar{u}) = 0$ and that the various partial derivatives were computed correctly.

4 Numerical linearization and stability assessment

In this section you will create a Simulink diagram representing the model (3), and you will then numerically linearize it using Matlab. You will compare the numerical linearization matrices to the theoretical ones that you computed by hand in Section 3. You will also test the stability of the linearized model.

Simulink blocks (inside the Library Browser)

`Ports & Subystems` → `Subsystem`	creates a block containing a subsystem
`Sources` → `In1`	input port
`Sinks` → `Terminator`	used to terminate output signals
`Sinks` → `Out1`	output port
`Continuous` → `Integrator`	integrates input signal
`User-defined functions` → `Interpreted MATLAB Fcn`	computes values using a Matlab function

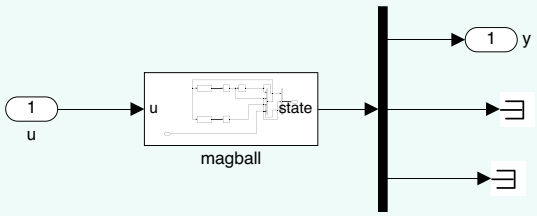
Matlab commands

`[A,B,C,D]=linmod('mod',xbar,ubar)`	Computes the numerical linearization of model `mod` at the equilibrium `xbar` with control `ubar`

`norm(M)`	Computes the norm of a matrix `M` (or also a vector)
`eig(M)`	Computes the eigenvalues of a square matrix `M`

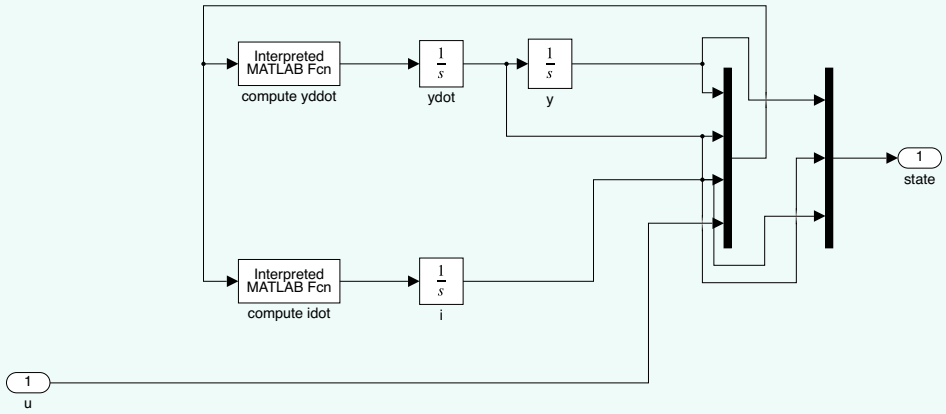
Open Simulink and the Library Browser. Create a blank diagram called `lab2_1.slx`.

- Using the blocks listed above, draw a Simulink diagram like the one depicted below, where `magball` is a block containing a subsystem that you will define in a moment. Label all quantities as in the figure, and in particular label the subsystem block `magball`.



- Open the input port `u` on the left end side of the block diagram above. Click on `Signal Attributes` and in the box labelled `Port dimensions` write `1` in place of the default `-1`.

- Open the `magball` subsystem block, and in it draw the diagram depicted below, labelling various quantities as in the figure.



There are two `Interpreted MATLAB Fcn` blocks in the figure. These blocks take in input the four-dimensional vector $\begin{bmatrix} x_1 & x_2 & x_3 & u \end{bmatrix}^\top$, and compute, respectively, $\ddot{y}$ (upper block) and di_a/dt (lower block) using the expressions for di_a/dt and $\ddot{y}$ found in (1) and (2). Open these blocks and enter the expressions for $\ddot{y}$ and di_a/dt. In doing so, note the Matlab convention that the input vector to the Function block is called `u`, so if you need to refer to, say, x_3, you will write `u(3)`, and the control input u will be `u(4)` according to our ordering. You will soon define the physical constants M, L_a, R_a, g, k_m in a script. Inside the `Interpreted MATLAB Fcn` blocks, refer to them simply as `M`, `La`, `Ra`, `g`, `km`.

The output of the upper block representing $\ddot{y}$ is integrated twice to produce the states $x_1 = y$ and $x_2 = \dot{y}$, while the output of the lower block is integrated once to produce the state $x_3 = i_a$. The three states so defined, together with the input u coming from the input port are collected in a vector by the `Mux` block. This vector is fed back to the `Interpreted MATLAB Fcn` blocks. This subsystem with input u will output the state x, formed using a second `Mux` block.

- Create a Matlab script named `lab2.m`. Begin by declaring variables `M, La, Ra, g, km` with numerical values given in the Table of Section 1.
- Declare a variable `ybar` representing the equilibrium value of y, and set this variable equal to 0.1 metres. Then, declare a 3×1 vector `xbar` and a scalar `ubar` containing the expressions for $\bar{x}$ and $\bar{u}$ that you found in Section 3.

  Note: By setting $\bar{y} = 0.1$, we are declaring that we are interested in stabilizing the ball at a distance of 10 cm from the magnet face.
- Using the command `linmod` calling the Simulink model `lab2_1.slx`, find the matrices A, B, C, D of the numerical linearization at `xbar,ubar`.
- Declare matrices `A1,B1,C1,D1` containing the expressions for the linearization matrices that you found in Section 3, using the parameters `M, La, Ra, g, km` just declared. Using the command `norm(A-A1)`, compute the error between theoretical and numerically derived matrices. Repeat for `B,B1`.

  Note: The command `norm` computes the norm of a matrix, which roughly speaking is a measure of how large the entries of the matrix are. The norm of a matrix is zero if and only if the matrix is zero.
- Using the matrices `A1,B1,C1,D1`, and using the technique your learned in lab 1, find the transfer function of the linearized model and extract the list of its poles. Name this transfer function `G`.
- Using the command `eig`, compute the eigenvalues of `A1`.

Output 2.

- Print the numerically derived matrices `A,B` and their theoretically derived counterparts, `A1,B1`.
- Print the errors described above, and comment on the accuracy of the numerical approximation performed by Matlab/Simulink.
- Print the eigenvalues of `A1` and the poles of `G`. Note that the poles of the transfer function coincide with the eigenvalues of the matrix `A1`. Is the linearized model internally stable? Is it BIBO stable? Comment on how your findings about stability conform with physical intuition.

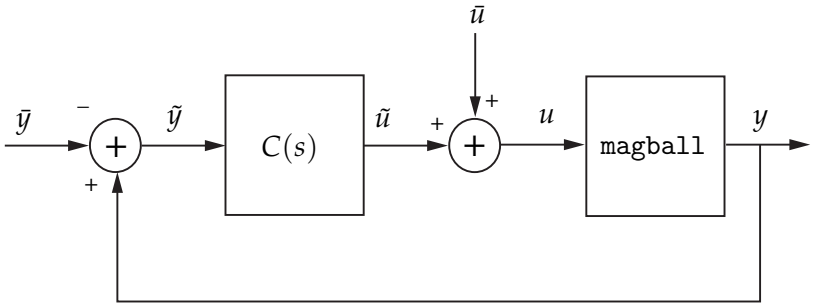
5 Feedback control of the magnetic levitation system

You have computed the transfer function of the linearized model of the magnetic levitation system. Note that the input of the transfer function is $\tilde{u} = u - \bar{u}$, and the output is $\tilde{y} = y - \bar{y}$. In this section you'll design a so-called lead controller to stabilize the linearized plant, and you'll test the controller on the nonlinear Simulink model `magball` that you've developed in Section 4. More precisely, our objective is to make $y(t)$ converge to $\bar{y}$, or equivalently make $\tilde{y}(t)$ converge to zero. We are therefore asking the error output $\tilde{y}$ to track a zero reference signal.

Before we begin, we need to understand how to implement a linearization-based controller, given that the input of this latter is $\tilde{y}$, and the output is $\tilde{u}$. From the definition, we have

$$u = \bar{u} + \tilde{u}, \ \tilde{y} = y - \bar{y},$$

so the block diagram of our controller looks like this.



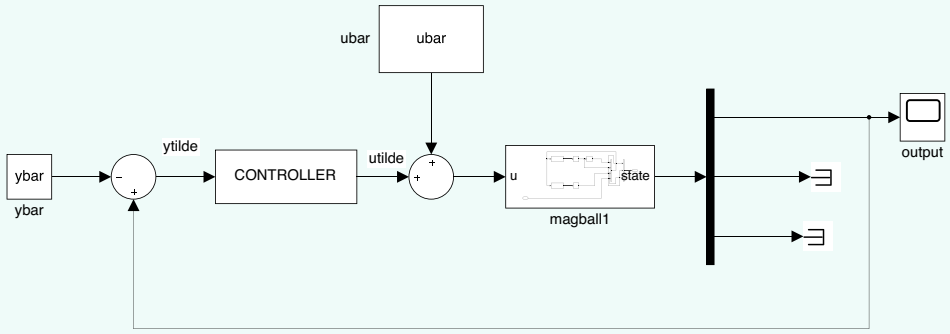
Note that the input and output of the nonlinear model `magball` are u and y. The diagram uses y and $\bar{y}$ to compute $\tilde{y}$, which is then fed to the controller. The controller computes $\tilde{u}$, to which we add the bias $\bar{u}$ and get u, which is then fed to the nonlinear control system model `magball`.

Simulink blocks (inside the Library Browser)

`Control System Toolbox` → `LTI System` Block containing LTI system object

Create a blank diagram called `lab2_2.slx`.

- Cut and paste the subsystem `magball` from `lab2_1.slx` in the new diagram, and draw the feedback loop depicted below.



- Open the `Model settings` and under the `Solver` menu, set both relative and absolute tolerances to 10^{-10}. Make sure that the solver is `variable-step`, and set the `stop time` to 10 seconds.
- The blocks labelled `ybar`,`ubar` are constants that call variables defined previously in the workspace. Pay particular attention to the ordering of signs of the summing block, which you can achieve opening the summing block and using the string `|-+`.
- The `LTI System` block[2] labelled `CONTROLLER` is an LTI object, and `CONTROLLER` is the name of the object that will soon be defined in the workspace.

In Section 4 you should have determined that the linearized model is unstable (both internally and from the input-output point of view). It turns out that the stabilization problem for this system is much harder than the speed control problem for the DC motor that you investigated in lab 1. For this reason, we will suggest a specific controller structure, and you'll simply tune a parameter to achieve stability.

The controller we propose is called a *lead controller*, and has the form

$$C(s) = K\frac{s+z}{s+p}, \tag{4}$$

where $K, z, p > 0$ are design parameters such that $z < p$.

Return to your Matlab script `lab2.m` and do the following.

- Create an LTI system object named `CONTROLLER` representing the transfer function (4). Choose parameters `z=10`, `p=100`, `K=70` for this transfer function.

  Note: Do not name the controller `C`, as this symbol was already used earlier to define the output matrix of the linearization.
- The poles of the closed-loop system transfer function (using the linearized plant), are the roots of $1 - C(s)G(s)$. Note the minus sign, due to definition of $\tilde{y}$ which led us to flipping the signs in the summing block of the feedback loop. Using the tools you learned in lab 1, determine the zeroes of `1-CONTROLLER*G` and verify that the system is unstable.
- Repeat the above operation with increasing values of K and find a value of K for which the closed-loop system is BIBO stable (try increasing K by increments of 10).

  Optional: If you want to try a more advanced tool, look at the Matlab help for the command `rlocus` (root locus). The root locus of a transfer function $T(s)$ is the plot in the complex plane of the roots of $1 + KT(s)$ as K ranges from 0 to ∞. In our case, we are interested in the roots of `1-CONTROLLER*G`, and hence in the root locus of `-CONTROLLER*G`.

[2]This block requires the Control Systems Toolbox. If you don't have this toolbox installed, you can use a transfer function block calling numerator and denominator arrays in the workspace, as you did in lab 1.

- Now you'll test your controller on the nonlinear control system. Return to the Simulink diagram `lab2_2.slx`. Edit the `magball` block and, in it, edit the integrator block which outputs y. Set the initial condition of this block to 0.15 metres. You are initializing the ball at 15 cm from the magnet face, or 5 cm away from the desired distance. Run the diagram and check whether the output converges to 0.1 metres. If it doesn't, tune the parameter K.
- Having found a value of K for which your controller work, you will now roughly determine how large is the range of initial ball positions $y(0)$ for which the controller works. Without changing K, change the initial condition of the integrator block above to find rough lower and upper limits of $y(0)$ for which the controller manages to stabilize the ball.

Output 3.
- Print the value of K you found above. Produce a figure of the output $y(t)$ when $y(0) = 0.15$ m.
- Describe the procedure you followed in finding K, and any observations you made along the way.
- Print the range of initial conditions $y(0)$ for which your controller succeeds in stabilizing the ball, and comment on why your controller does not work for initial conditions that are far from the equilibrium.