Lab 2 – ECE557F Linear Control Theory

Position tracking for the cart system

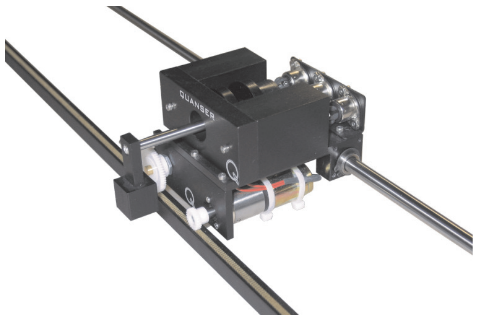
Manfredi Maggiore
Department of Electrical and Computer Engineering
University of Toronto

Main concepts of this lab

- Designing state feedback controllers using a first principles procedure for eigenvalue assignment
- Designing a state estimator (an observer) using the first principles procedure
- Applying these ideas to a cart system, both in simulation and experimentally

1 Introduction

In this lab we consider the cart system depicted below (in later labs we will add a pendulum on the cart), with the objective of making the cart position track a square wave reference signal, thereby making the cart move back and forth between two points on the track. You might have used this setup in a previous control course (e.g., ECE311 or ECE356).



In this setup, a DC motor drives a gear that engages the track. The torque of the DC motor produces a force along the track propelling the cart. By controlling the motor voltage by means of a suitable algorithm, we are able to control the cart position.

We model the system as a mass sliding on a flat plane with viscous friction (i.e., a friction force whose magnitude is proportional to the cart speed). Denoting by z the position of the cart (with the convention that z is zero when the cart is in the middle of the track), by F the force applied to the cart by the motor, by M the mass of the cart, and by B the viscous friction coefficient, the mathematical model of the setup in the figure above is

$$M\ddot{z} = F - B\dot{z}. \tag{1}$$

We need to model the actuator dynamics of the DC motor, responsible for producing the applied force F. For that, we will ignore the transient dynamics of the motor armature and simply assume that

$$F = \alpha_1 u - K_e \dot{z}.$$

In other words, the force F imparted by the motor on the cart is proportional to the motor voltage u (with multiplicative factor α_1), and there is a friction-like term due to the motor's back emf and proportional to the cart speed $\dot{z}$. Substituting the expression for F in (1) and letting $\alpha_2 = B + K_e$, we arrive at the model

$$M\ddot{z} = \alpha_1 u - \alpha_2 \dot{z}.$$

Defining the state vector $x = \begin{bmatrix} z \\ \dot{z} \end{bmatrix}$ and recognizing that the model above is LTI, we arrive at

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & -\alpha_2/M \end{bmatrix} x + \begin{bmatrix} 0 \\ \alpha_1/M \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x. \end{aligned} \tag{2}$$

In the above, y is the measurement output, i.e., the information that we get from sensors. In this case, we measure the cart position, z, via an optical encoder. Throughout the preparation, we will use the following theoretical values for physical parameters appearing in (2):

α_1	1.7265 N/V
α_2	10.5 Ns/m
M	0.8211 Kg

2 Organization of this lab

This lab has the following components:

- A preparation (Section 4) **to be completed and submitted in advance of the lab.** Each lab group will submit one preparation.
- Lab activities to be performed on the day of your lab session (Section 5) comprised of a Simulink-based simulation of an output feedback controller (Section 5.1) and physical experimentation (Section 5.2).
- A report **to be submitted within about one week of the lab**. Each group will submit one report.

Throughout the lab, you will be guided through a number of steps which will require you to write Matlab code. These parts will be highlighted in a different colour and will look like this:

Create a function `pendulum.m`. The first line of the function will be

```
function xdot = pendulum(t,x,parameters)
```

Your final report should provide outputs (quantities, figures) and discussions. Requests for such outputs to be included in the report look like this:

Output 1.

- Display the eigenvalues of the matrix A.
- Present the graph of the state signal $x(t)$.
- Discuss how the tuning of controller gains affects the qualitative properties of $x(t)$.

Even though requests for output might be found in the preparation, these requests should be addressed only in the final report and need not be addressed in the preparation.

3 Submission guidelines and mark breakdown

Marks are assigned to groups. Members of each group get identical marks, unless special circumstances occur (for instance, a group member misses the lab). The group lab mark will be made up of three components.

Matlab/Simulink code	4 pts
Lab performance	4 pts
Lab report	2 pts
Total	**10 pts**

Matlab/Simulink code. *Submit your Matlab and (if applicable) Simulink 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.

Lab performance. You must show to the lab TA the main steps of your preparation and your lab experiment. Your grade for this part will be based on your understanding of the steps you are executing and the successful conclusion of the lab experiment.

Lab report. *Submit a lab report within about one week of your lab session (see Quercus for due date).* Your lab report **must be typed**. Write a concise report containing the requested outputs, but don't just print out a collection of matrices and figures. Add some flesh to the outputs that are requested within the lab document so that one can follow the logical development of the lab. When the lab document asks you to comment on something, strive to provide meaningful, insightful commentary.

Report guidelines

- Your report must include an introduction giving a brief description of the lab purpose, a block diagram of the closed loop system showing the plant, the observer, and the controller. Label the inputs, outputs, and state of each block, and include the state equations describing their behavior.
- Write a section of the report for each of the outputs requested in this document.
- Figures/plots must have titles and axes labels, as well as captions (e.g., Figure 1) that are referenced in the text.
- Figures/plots must be of adequate image resolution and visibility. Screenshots are fine as long as they are clear.
- Please try to use the equation editor in Microsoft Word or use LaTeX to write equations so that they are clear.
- It is preferred to format MATLAB results in-line seamlessly with the rest of the report with appropriate truncation of decimal points.

4 Preparation

In this preparation you will use first principles to design a so-called *output feedback controller* making the cart position converge to a desired piecewise constant signal, $z_d(t)$ (a square wave). The output feedback controller you'll design is the combination of a *state feedback controller* and a state estimator called an *observer* that estimates the state signal $x(t)$ from the output signal $y(t)$ and the control signal $u(t)$. In class, you will learn about these designs in great detail. The goal here is to expose you to the main ideas before they are introduced more rigorously in class.

4.1 Submission guidelines for preparation

Before your lab session (see Quercus for precise due date), please submit your Matlab script `lab2.m` and the Simulink diagram `lab2_part1`. You *do not* need to submit a report at this stage. You *do not* need to address the request for output (you will address it in your final report).

4.2 State feedback stabilization

As a stepping stone towards the design of an output feedback controller, we first consider the case when the whole state signal $x(t)$ is available for feedback control. Also, we assume at first for simplicity that the desired cart position is $z_d = 0$, i.e., that we want the cart to converge to the middle of the track[1]. We'll correspondingly want the cart speed $\dot{z}$ to also converge to zero. Later, we'll easily extend the controller to the case $z_d \neq 0$. The goal is then to design a state feedback controller making the state signal $x(t)$ converge to zero from any initial condition. This control goal is called *state feedback stabilization*.

A state feedback stabilizer is a linear function of the state

$$u = Kx = \begin{bmatrix} K_1 & K_2 \end{bmatrix} x = K_1 z + K_2 \dot{z},$$

where the row vector $K \in \mathbb{R}^{1\times 2}$ is to be designed so as to ensure that the closed-loop system obtained by substituting this linear function into the state equation (2),

$$\dot{x} = (A + BK)x,$$

is asymptotically stable. In other words, we need to find K such that the eigenvalues of $A + BK$ are at some desired location in $\mathbb{C}^-$.

The matrices A and B in (2) have this special form

$$A = \begin{bmatrix} 0 & 1 \\ 0 & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ b_2 \end{bmatrix}.$$

[1]More precisely, the zero position of the cart is the initial position when the Arduino code is run.

Let's look at the matrix $A + BK$ for a generic row vector $K = [K_1 \;\; K_2]$:

$$\begin{aligned} A + BK &= \begin{bmatrix} 0 & 1 \\ 0 & a_{22} \end{bmatrix} + \begin{bmatrix} 0 \\ b_2 \end{bmatrix} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \\ b_2 K_1 & a_{22} + b_2 K_2 \end{bmatrix}. \end{aligned}$$

The characteristic polynomial of the matrix above is

$$\det(A + BK - \lambda I_2) = \lambda^2 - (a_{22} + b_2 K_2)\lambda - b_2 K_1. \tag{3}$$

Using the gains K_1 and K_2 we can arbitrarily assign the coefficients of the terms of degree 1 and 0. As we show in a moment, this implies that we can *arbitrarily assign the eigenvalues of the closed-loop system*. We will learn in the course that this property means that the plant is *controllable*. Note that not all plants are controllable so it is not always possible to arbitrarily assign the eigenvalues of the closed-loop system.

Say we want the eigenvalues of $A + BK$ to be at $p_1, p_2 \in \mathbb{C}^-$, then we want the characteristic polynomial of this matrix to be equal to

$$(\lambda - p_1) \cdot (\lambda - p_2) = \lambda^2 - (p_1 + p_2)\lambda + p_1 p_2. \tag{4}$$

Comparing (3) and (4) and solving for K_1, K_2, we get

$$\begin{aligned} K_1 &= -\frac{p_1 p_2}{b_2} \\ K_2 &= \frac{-a_{22} + p_1 + p_2}{b_2}. \end{aligned} \tag{5}$$

Matlab commands

`eig(A)`	Returns the eigenvalues of matrix A.

The script `lab2.m` and the function `state_feedback_design` Create a Matlab script titled `lab2.m`.

- In the script, define matrices A, B, C as in (2), with numerical values for the physical parameters given in the table presented at the end of Section 1.
- At the end of the script `lab2.m` (not as a separate file), write a function `[K1 K2] = state_feedback_design(A,B,p)`

  This function takes as input the matrices A and B of system (2) and a row vector p containing desired eigenvalues, and it outputs gains K_1, K_2 for the state feedback controller.
- In the script `lab2.m`, test your function by finding values of K_1, K_2 assigning the eigenvalues of $A + BK$ to be at $\{-5, -5\}$.

- Using the gains K_1, K_2 you've just obtained, test the eigenvalues of $A + BK$ and check they are at $\{-5, -5\}$.

  These matrices will be used by the Simulink diagram you'll construct in the next section.

The state feedback design is almost complete. What is left to do is remove the assumption that $z_d = 0$. This can be done quite easily by *shifting* the state coordinates. Suppose now that z_d is a nonzero constant value (in a moment, $z_d(t)$ will become a square wave). Define a desired equilibrium state $x_d = \begin{bmatrix} z_d \\ 0 \end{bmatrix}$ and the error state $e = x - x_d$. Since x_d is a constant vector (independent of t), we have

$$\dot{e} = \dot{x} = Ax + Bu.$$

Expressing x in terms of e as $x = x_d + e$, we rewrite

$$\dot{e} = Ae + Ax_d + Bu.$$

Noticing that $Ax_d = 0$, we arrive at

$$\dot{e} = Ae + Bu.$$

The plant matrices after the coordinate shift are the same as before. In place of the feedback $u = Kx$, we use $u = Ke$ which gives the closed-loop system

$$\dot{e} = (A + BK)e,$$

for which we know that $e(t)$ converges to zero from any initial condition. Since $e(t) \to 0$, $x(t)$ converges to x_d, as desired. In conclusion, having designed K to assign the eigenvalues of $A + BK$, the state feedback stabilizer of the equilibrium x_d is $u = K(x - x_d)$.

Lastly, we replace the constant z_d by a square wave signal $z_d(t)$, and letting $x_d(t) = [z_d(t) \;\; 0]^\top$, we have the controller $u = K(x - x_d(t))$. This effectively means that our controller periodically alternates between the stabilization of two equilibria, corresponding to the high and low values of the square wave $z_d(t)$. In conclusion, the state feedback controller is given by

$$u = K(x - x_d(t)) = K \begin{bmatrix} z - z_d(t) \\ \dot{z} \end{bmatrix}. \tag{6}$$

In the next section, you'll simulate the effects of this controller using Simulink.

4.3 Simulink simulation of state feedback controller

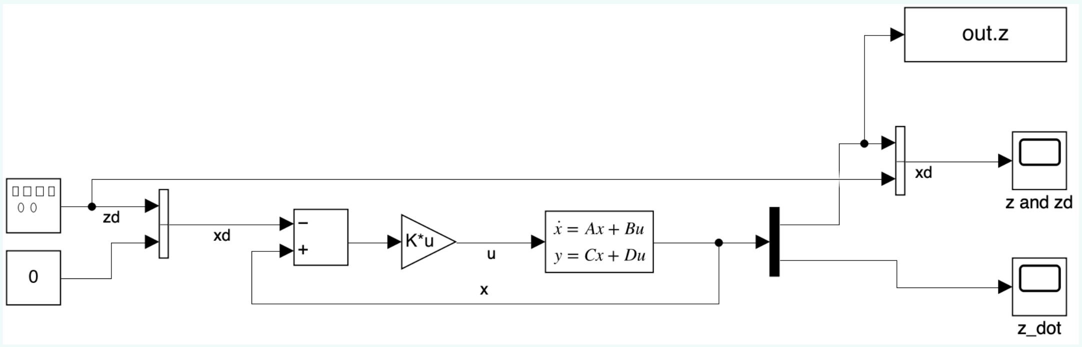
In this section you'll use Simulink simulate the state feedback stabilizer (6). You'll need the following Simulink blocks.

Simulink blocks (inside the Library Browser)

`Continuous` → `State-space`	define a state space LTI object
`Sources` → `Signal generator`	generate a variety of signals
`Sources` → `Constant`	output a desired number or vector or matrix
`Signal Routing` → `Vector concatenate`	concatenate two vectors
`Signal Routing` → `Demux`	extract the components of a vector
`Math Operations` → `Gain`	scalar, vector, or matrix gain
`Math Operations` → `Add`	add two numbers with desired addition signs
`Sinks` → `Scope`	plot signal vs time
`Sinks` → `To Workspace`	save a simulation signal to the workspace

The Simulink diagram `lab2_part1`

- Create a blank Simulink diagram called `lab2_part1`.
- 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 30 seconds.
- Using the blocks described above, create a diagram that looks like this.



- In the diagram, ensure the following:
  - The signal generator should be set to generate a square wave of amplitude 0.03 metres and frequency 0.1 Hz.
  - The state space block should refer to matrices A, B defined in the main file `lab2.m` and available in the workspace.
  - Since right now we are measuring the entire state, the matrix C is the 2×2 identity matrix, while $D = [0; 0]$.
  - The `Add` block should have signs $-, +$ as indicated on the figure.
  - The gain vector K is in the workspace and was defined in the main Matlab file `lab2.m`. Make sure to select the option where the block performs a matrix multiplication (rather than the default element-wise multiplication).

- Run your diagram and verify that $z(t)$ tracks the square wave $z_d(t)$, while $\dot{z}(t)$ converges to zero between two consecutive values in the square wave $z_d(t)$.
- Using either the `Cursor Measurement Tool` of the `Scope` block or saving the signal $z(t)$ in the workspace, measure the settling time T_{s_1} of $z(t)$ between two consecutive jumps.

  Recall that the settling time is the time it takes for the magnitude of the error between the output response to a step and its steady-state value to reach and stay below 2% of the step size. In our setup, consider the time $\bar{t} > 0$ representing an intermediate step in the square wave from high to low. (Discard the initial step). At $\bar{t}$ the reference signal z_d jumps from $+0.03$ to -0.03. Assuming that $z(t)$ is in steady-state before this jump, the error at $\bar{t}^+$ is $z(\bar{t}^+) - z_d(\bar{t}^+) \approx 0.03 - (-0.03) = 0.06$. Thus T_{s_1} is the time at which $|z(\bar{t} + T_{s_1}) - (-0.03)| \leq 0.02 \cdot 0.06 = 1.2 \cdot 10^{-3}$.

The request for output below (and the ones that follow) should be addressed only in the final report and need not be addressed in the submission of the preparation.

Output 1.

- Provide the gain K you've found.
- Provide plots of $z(t)$ and $z_d(t)$, and of $\dot{z}(t)$ coming out of the simulations. To save the Simulink scope, select `File` $\rightarrow$ `Print to Figure`, then save the figure in the format of your choice. **Do not take screenshots or you will be penalized.**
- Provide the numerical value of T_{s_1} that you've found.
- Now rerun the code assigning eigenvalues at $\{-2, -2\}$. Estimate again the settling time, call it T_{s_2}. If you have estimated the settling times correctly, you should find that their ratio is approximately $T_{s_1}/T_{s_2} \approx 0.4$. Explain why we expect this ratio between the settling times.

4.4 State estimation and output feedback control

We now need to estimate the state of the plant using information in the measurement output $y(t)$ and the control signal $u(t)$ both of which are available to us. A device that can do the job is called an *observer*. Consider an LTI system (we assume $D = 0$ and SISO for simplicity)

$$\dot{x} = Ax + Bu$$
$$y = Cx.$$

An observer for the system above is an LTI system whose state $\hat{x}$ is an estimate of x and whose structure is this:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}). \tag{7}$$

The right-hand side of this equation looks like the equation of the plant plus a correction term, $L(y - C\hat{x})$, involving the *prediction error* between the measured output y and the predicted output $C\hat{x}$. This error is scaled by a column vector L whose role will be clear in a moment. Defining the *estimation error* $\tilde{x} = x - \hat{x}$, we have

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= (Ax + Bu) - (A\hat{x} + Bu + L(y - C\hat{x})) \\ &= A(x - \hat{x}) - L(Cx - C\hat{x}) \\ &= (A - LC)(x - \hat{x}) \\ &= (A - LC)\tilde{x}.\end{aligned}$$

Thus the estimation error is the solution of the LTI system

$$\dot{\tilde{x}} = (A - LC)\tilde{x}.$$

We see here the role of the design vector L: if we choose it appropriately, we might be able to assign the eigenvalues of the matrix $A - LC$ in $\mathbb{C}^-$, and that would guarantee that the estimation error converges to zero exponentially.

The theory behind this idea will be explored in class. In this lab, we'll just apply the idea to our particular matrices A, B, C in (2) which, we recall, have this structure:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ b_2 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

We follow the same basic idea explored in Section 4.2 and let L be a generic vector $L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}$. Then,

$$A - LC = \begin{bmatrix} -L_1 & 1 \\ -L_2 & a_{22} \end{bmatrix},$$

whose characteristic polynomial is

$$\lambda^2 + (L_1 - a_{22})\lambda + L_2 - L_1 a_{22}.$$

A before, if we want to assign eigenvalues at $\{p_1, p_2\} \subset \mathbb{C}^-$, we'll want this desired characteristic polynomial

$$\lambda^2 - (p_1 + p_2)\lambda + p_1 p_2.$$

Equating coefficients and solving for L_1, L_2 we get

$$\begin{aligned} L_1 &= a_{22} - (p_1 + p_2) \\ L_2 &= L_1 a_{22} + p_1 p_2. \end{aligned}$$

As in the state feedback stabilization problem of Section 4.2, we see that we are able to arbitrarily assign the eigenvalues of the matrix $A - LC$ governing the evolution of the state estimation error. We will learn in the course that this property means that the plant is *observable*. Not all plants are observable so it is not always possible to arbitrarily assign the eigenvalues of $A - LC$.

The function `observer_design`

- Write a function `[L1,L2]=observer_design(A,p)`

  This function takes as input the matrix A of system (2) and a row vector p containing desired eigenvalues, and it outputs observer gains L_1, L_2. The function is to be written at the end of the main file `lab2.m`, not as a separate file.

- In the script `lab2.m`, test your function by finding values of L_1, L_2 assigning the eigenvalues of $A - LC$ to be at $\{-10, -10\}$.

- Using the gains L_1, L_2 that you've just obtained, test the eigenvalues of $A - LC$ and check they are at $\{-10, -10\}$.

We've thus ensured that, for arbitrary initial conditions, the state signal $\hat{x}(t)$ of the observer (7) will exponentially converge to the state signal $x(t)$, and the rate of convergence can be rendered as fast as desired by designing a gain vector L assigning the eigenvalues of $A - LC$ to be far on the left of $\mathbb{C}^-$.

With a state estimate at hand, we can replace our state feedback controller $u = K(x - x_d)$ of Section 4.2 by the *output feedback controller* $u = K(\hat{x} - x_d)$. The overall controller is this

$$\begin{aligned}\dot{\hat{x}} &= A\hat{x} + Bu + L(y - C\hat{x})\\ u &= K(\hat{x} - x_d).\end{aligned}$$

Substituting the expression for u from the second identity into the differential equation we arrive at the final form of the output feedback stabilizer:

$$\begin{aligned}\dot{\hat{x}} &= (A + BK - LC)\hat{x} + Ly - BKx_d\\ u &= K(\hat{x} - x_d(t)).\end{aligned} \tag{8}$$

Recall that $x_d(t)$ is the vector $x_d(t) = [z_d(t) \;\; 0]^\top$, where $z_d(t)$ is the square wave reference signal. Notice that both y and x_d are *inputs* in the state equation above.

The final output feedback controller is an LTI system with state $\hat{x}$, inputs (y, x_d) and output u. In other words, the controller takes as input the reference state signal $x_d(t)$ and the measurement output signal $y(t)$, and it outputs the control signal $u(t)$ to be fed to the plant. The controller has the form

$$\begin{aligned}\dot{\hat{x}} &= A_{\text{ctrl}}\hat{x} + B_{\text{ctrl}}\begin{bmatrix} y \\ x_d \end{bmatrix}\\ u &= C_{\text{ctrl}}\hat{x} + D_{\text{ctrl}}\begin{bmatrix} y \\ x_d \end{bmatrix},\end{aligned} \tag{9}$$

where

$$\begin{aligned}A_{\text{ctrl}} &= A + BK - LC && \in \mathbb{R}^{2\times 2}\\ B_{\text{ctrl}} &= \begin{bmatrix} L & -BK \end{bmatrix} && \in \mathbb{R}^{2\times 3}\\ C_{\text{ctrl}} &= K && \in \mathbb{R}^{1\times 2}\\ D_{\text{ctrl}} &= \begin{bmatrix} 0 & -K \end{bmatrix} && \in \mathbb{R}^{1\times 3}.\end{aligned}$$

The function `output_feedback_controller`

- Write a function

  `[Actrl,Bctrl,Cctrl,Dctrl]=output_feedback_controller(A,B,C,p_feedback,p_observer)`

  This function takes as input the matrices A, B, C of system (2) and row vectors `p_feedback` and `p_observer` containing desired eigenvalues for $A + BK$ and $A - LC$, respectively, and it outputs controller matrices $(A_{\text{ctrl}}, B_{\text{ctrl}}, C_{\text{ctrl}}, D_{\text{ctrl}})$. The function is to be written at the end of the main file `lab2.m` and **it must call the functions** `state_feedback_design` **and** `observer_design`.

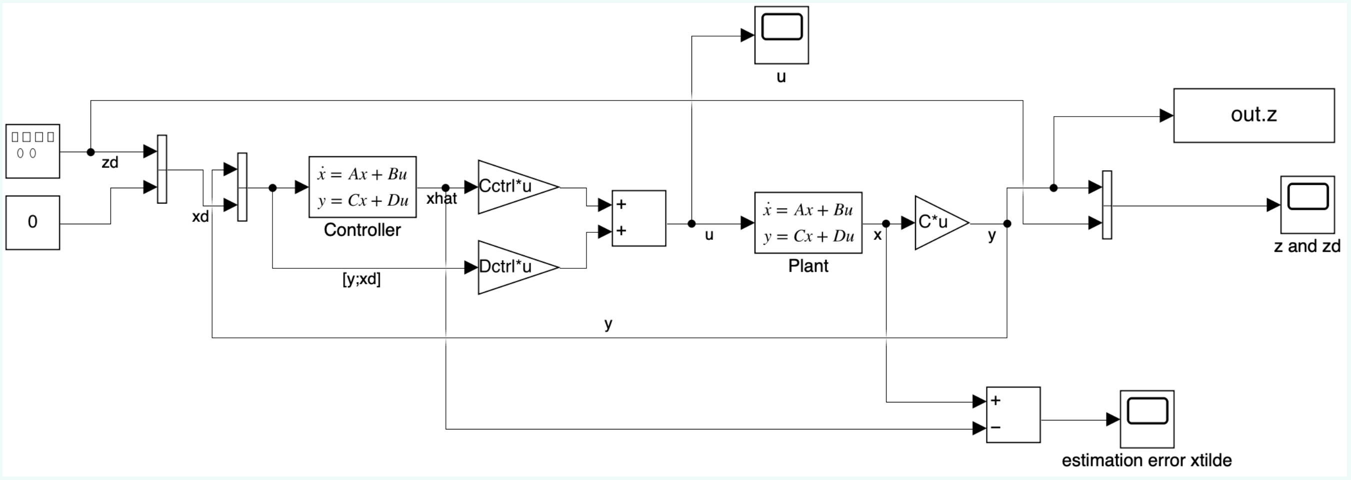
- Using this new function, find the matrices of an output feedback controller with `p_feedback` $= [-2 \quad -2]$ and `p_observer` $= [-20 \quad -20]$.

5 Lab activity

5.1 Simulink simulation of output feedback controller

The Simulink diagram `lab2_part2`

- Create a blank Simulink diagram called `lab2_part2`.
- 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 30 seconds.
- Create a diagram that looks like this.



- In the diagram, ensure the following:
  - The signal generator is set in the same manner as we did in the diagram `lab2_part1`.

- Note how the input of the controller block is the concatenation of two signals: the plant output $y(t)$ first, then the reference state $x_d(t)$.
- Since we want to access the state estimate $\hat{x}$, the matrices used in the controller block are `(Actrl,Bctrl,eye(2),zeros(2,3))`. This ensures that the output of the controller is the state signal $\hat{x}(t)$, which is used in the diagram to compute and plot the estimation error $\tilde{x}$ at the bottom right-hand side.
- Note how the controller state $\hat{x}$ is fed to a gain block C_{ctrl}. Make sure that this gain block is set to perform a matrix multiplication (rather than the default element-wise multiplication).
- Note further how the input of the controller, the vector $[y \;\; x_d]^\top$, is fed to the gain matrix D_{ctrl}, and the output is summed to $C_{\text{ctrl}}\hat{x}$ to form the control input u, just as in (9). Make sure that the gain block D_{ctrl} is set to perform a matrix multiplication.
- Similarly for the plant model, since we need to access the state signal $x(t)$ we use matrices `(A,B,eye(2),zeros(2,1))`, then feed the state signal to the gain block C (once again set to perform a matrix multiplication) to get the output signal $y(t)$.
- Run the diagram and verify that reference tracking is achieved. Determine from the scopes the peak value of the voltage magnitude $|u(t)|$ and estimate the settling time of $z(t)$ in the same manner as you did earlier with the diagram `lab2_part1`.

Output 2.

- Produce the gains K and L that you've found.
- Produce the matrices $(A_{\text{ctrl}}, B_{\text{ctrl}}, C_{\text{ctrl}}, D_{\text{ctrl}})$.
- Produce plots from the three Simulink scopes, **making sure to avoid taking screenshots**.
- Repeat the simulation by changing the eigenvalues of $A + BK$ to `p_feedback` $= [-5 \;\; -5]$. Discuss the effects that these changes have on the tracking error $z - z_d(t)$ as well as the peak value of the motor voltage magnitude $|u(t)|$. In this discussion, ignore the first period of the square wave.
- Revert to `p_feedback` $= [-2 \;\; -2]$, and now change the eigenvalues of $A - LC$, setting `p_observer` $= [-10 \;\; -10]$. Discuss the effects that these changes have on the state estimation error $x - \hat{x}(t)$.

5.2 Physical experimentation

Preliminary steps and safety precautions:

- **Make sure to remove the weight off the cart.** Ask your TA to show you how to do it.
- Please ensure that the cart is positioned at the middle of the track.

- Do not modify the Arduino code outside of those areas outlined in the lab manual.
- Do not change the frequency and amplitude of the square wave reference signal.
- When running the experiment ensure that one student from the group is firmly holding the track.
- If your model goes unstable, lift the motor gear off the track and either detach the power cable of the Arduino from the computer side or upload an empty Arduino file, as you did in lab 1.

Next, you'll edit the Arduino code to insert your output feedback controller.

The file `lab2.ino`

- Locate the provided Arduino file `lab2.ino`.
- In the Arduino Software, make sure the correct `Board` and `Port` are selected under the Tools tab.
- The structure of the Arduino code is analogous to `lab1.ino` studied in Lab 1.
- In Section 1 of the Arduino code, define the system matrices as matrix variables. These matrices $(A_{\mathrm{ctrl}}, B_{\mathrm{ctrl}}, C_{\mathrm{ctrl}}, D_{\mathrm{ctrl}})$ are determined by your Matlab script `lab2.m`.
- In Section 5, compute the plant output (the cart position z) and store it in the global variable `y`. The cart encoder count is stored in the variable `encoderPos`. To convert this reading into the plant output, you need to multiply `encoderPos` by the parameter `K_encoder`.
- Still in Section 5, compute the control input. You may add additional variable definitions/declarations to Sections 1 and 5 if needed.
  - To begin, you need to write code computing the derivative $\dot{\hat{x}}$ according to (9). In order to compute $\dot{\hat{x}}$ via (9), you need to use the vector $\hat{x}$ stored in memory in the variable `x_hat` and the input vector $[y \;\; x_d]^\top$, where y is the sensor output that you stored in the global variable `y`, and $x_d = [z_d \;\; 0]^\top$, where the reference signal $z_d(t)$ is stored in the global variable `z_d`.
  - Below your code for the computation of $\dot{\hat{x}}$, the code provided to you performs the state update step. To understand the meaning of this update, recall that the controller in (9) runs in continuous-time and needs to be discretized. The discretization provided updates the vector $\hat{x}$ like this

$$(\hat{x})_{\mathrm{new}} = (\hat{x})_{\mathrm{old}} + T_{\mathrm{sample}} \cdot \dot{\hat{x}}.$$

    This approximation of the solution of the ODE is called the *Euler discretization*.
  - Below the Euler update just described, compute the control input u using the matrices C_{ctrl} and D_{ctrl}, the updated state $\hat{x}$, and the input vector $[y \;\; x_d]^\top$.
- Position the cart in the middle of the track, upload the Arduino Code to the Arduino Mega Board with the upload button (or press Ctrl+u).

The file `lab2_plot.m`

- Locate the provided Matlab file `lab2_plot.m`.
- Change the value stored in the variable `port` on line 10, which is set to "COM3", to the serial port that is being used by the Arduino. Set `stop_time` to the total time you'd like the experiment to be plotted for. Run the m-file, which will start reading values from the serial port given by `port`. The data received from Arduino is recorded and MATLAB plots $z_d(t)$ and $y(t)$ versus time t automatically.

At this point you need to tune the controller to achieve the *best performance you can.* Performance in the context of this lab is measured by two metrics:

(a) The steady-state tracking error at the end of each half-period of the square wave reference signal (i.e., when the signal switches from high to low or vice versa). You will notice noise in the position signal $z(t)$. When determining the tracking error, consider the *average* of the steady-state signal.

(b) The settling time T_s associated with the cart position, as defined in Section 4.3. You need to adapt the definition of T_s to the parameters of the square wave being used in the experiment. Since you will be dealing with a noisy position signal $z(t)$, your determination of T_s will be approximate.

Tuning the controller means performing these steps:

1. Tune the controller gains via the script `lab2.m`.
2. Simulate the controller via the Simulink diagram `lab2_part2` to determine its theoretical performance.
3. Update the controller in Section 5 of the Arduino code `lab2.ino`.
4. Run the controller via the Matlab script `lab2_plot.m` and compare the experimental data to the simulation results.

Output 3.

- Explain the trial and error procedure that you followed to get the best performance on the experimental setup, and provide the parameters `p_feedback,p_observer` that gave the best results.
- Provide an experimental plot of the output signal $y(t)$ and the reference signal $z_d(t)$, as well as the settling time. Discuss the results: whether they are satisfactory, whether they conform to your expectations, and so on.

- Provide a simulation plot of the output signal $y(t)$ and the reference signal $z_d(t)$ for the same set of controller parameters used in your experiments. Discuss the differences between simulation and experiment.