Lab 4 – ECE557F Linear Control Theory

Square wave tracking for the cart-pendulum robot

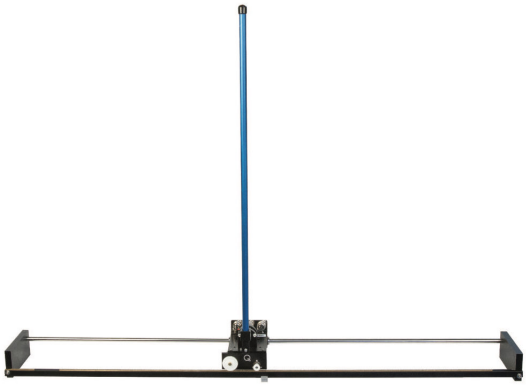
Manfredi Maggiore
Department of Electrical and Computer Engineering
University of Toronto

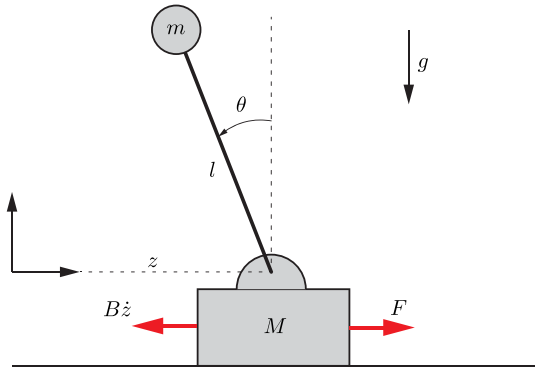
Main concepts of this lab

- Designing output feedback controllers using eigenvalue assignment and LQR
- Applying these ideas to a cart-pendulum system, both in simulation and experimentally
- Designing an output feedback integral controller and testing it experimentally

1 Introduction

In this lab we return to the cart-pendulum robot depicted below, this time designing an *output feedback controller* to make the cart move back and forth between two points on the track (i.e., make the cart track a square wave signal) while keeping the pendulum close to the vertical upright configuration.





As a reminder, in lab 3 you have carried out the following tasks:

- You have written code to automatically generate a Matlab function `cartpend.m` calculating the right-hand side of the nonlinear cart-pendulum differential equation. You then called this function from Simulink to simulate the nonlinear model.

- Using eigenvalue assignment and LQR, you have designed state feedback controllers to asymptotically stabilize the equilibrium $x = [0 \;\; 0 \;\; 0 \;\; 0]^\top$, corresponding to the cart being in the middle of the track[1] and the pendulum being upright.
- You compared the performance of various controllers both using the linearized and the nonlinear model.

Objectives of this lab: In this lab, you will perform the following tasks.

- You'll modify your state feedback controllers from lab 3 to make the cart track a square wave while keeping the pendulum close to the upright configuration.
- You'll design an observer for the linearized cart-pendulum model, using which you will arrive at an *output feedback controller* for square wave tracking.
- Using Simulink, you'll simulate the output feedback controller using the nonlinear cart-pendulum model.
- You'll implement your output feedback controller on the Arduino and perform experiments on the physical cart-pendulum robot.
- You will also design and experimentally test an output feedback *integral* controller, comparing its performance to that of the previous controller.

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) during which you will test two different controllers for square wave tracking: a standard output feedback controller and an output feedback controller with integral action.
- 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)
```

[1] Or, more precisely, the cart being at the starting position when the Arduino code was run

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-text.
- Figures/plots must have adequate image resolution and visibility. Screenshots are fine as long as they are clear.
- Please try to use LaTeX or the equation editor in Microsoft Word to write equations so that they are clear.
- It is preferred to format MATLAB results inline seamlessly with the rest of the report with appropriate truncation of decimal points.

4 Preparation

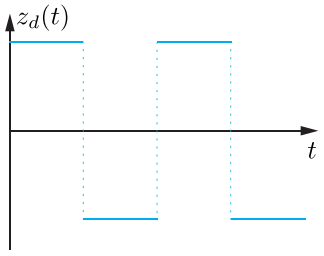
The objective of this preparation is modify your state feedback controllers from lab 3 to make the cart track a square wave signal, while keeping the pendulum close to the upright configuration and to design an observer for the linearized model. You'll simulate your output feedback controller using Simulink.

4.1 Submission guidelines for preparation

Before your lab session (see Quercus for due date), please submit your Matlab script `lab4.m` and the Simulink diagram `lab4_prep`. 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 Square wave tracking: Theory and Matlab design

Recall from lab 3 that we use the state $x = \begin{bmatrix} z & \dot{z} & \theta & \dot{\theta} \end{bmatrix}^\top$ for the cart-pendulum model. In order to design the output feedback controller, we first design a *state feedback controller* solving the problem, and then replace the state signal by the estimate coming from an observer. The theoretical underpinnings behind this design procedure are presented in lectures and in the textbook.



Consider a square wave reference signal $z_d(t)$ with zero average, as depicted in the figure. Since the signal $z_d(t)$ is *piecewise constant* the control specification discussed above can be cast as the problem of asymptotically stabilizing an equilibrium $x_d = \begin{bmatrix} z_d & 0 & 0 & 0 \end{bmatrix}^\top$ where the sign of z_d is periodically switched. In lab 3, you designed a controller to asymptotically stabilize the origin $x_d = 0_{4\times 1}$ so now we need to modify your controller to stabilize $x_d = \begin{bmatrix} z_d & 0 & 0 & 0 \end{bmatrix}^\top$ instead.

As in lab 3, we linearize the cart-pendulum model at the origin

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= Cx, \end{aligned} \tag{1}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

In order to make $x(t) \to x_d$, we follow the idea presented in lab 2 and shift the state coordinates by defining the error state $e = x - x_d$. Since x_d is constant (for now!), we have

$$\begin{aligned} \dot{e} &= \dot{x} \\ &= Ax + Bu \\ &= Ae + Ax_d + Bu \\ &= Ae + Bu, \end{aligned}$$

where the last identity is due to the fact that[2] $Ax_d = 0$. We've already verified in lab 3 that (A, B) is controllable. If K is a stabilizing gain, the controller $u = Ke$ will make $e(t) \to 0$ and thus $x(t) \to x_d$, as desired. In original state coordinates, replacing the desired equilibrium x_d by the *piecewise constant* reference signal $x_d(t)$, the controller is given by

$$u = K(x - x_d(t)). \tag{2}$$

Now we replace the state signal $x(t)$ by an estimate $\hat{x}(t)$ coming from an observer. Assuming that observability holds (you'll check that in a moment), an observer for (1) is

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y(t) - C\hat{x}),$$

where $L \in \mathbb{R}^{4\times 2}$ is chosen to guarantee asymptotic stability of the matrix $A - LC$. Using $\hat{x}$ in place of x in the controller (2) we obtain the output feedback controller

$$u = K(\hat{x} - x_d(t)).$$

Substituting this expression for u in the observer we get a concise expression for the controller:

$$\begin{aligned} \dot{\hat{x}} &= (A + BK - LC)\hat{x} + \begin{bmatrix} L & -BK \end{bmatrix} \begin{bmatrix} y(t) \\ x_d(t) \end{bmatrix} \\ u &= K\hat{x} - Kx_d(t). \end{aligned} \tag{3}$$

In conclusion, the output feedback controller is an LTI system with *six inputs* in the vector $\begin{bmatrix} y & x_d \end{bmatrix}^\top$ and one output, u. The system matrices of the controller are

$$\begin{aligned} A_{\text{ctrl}} &= A + BK - LC && \in \mathbb{R}^{4\times 4} \\ B_{\text{ctrl}} &= \begin{bmatrix} L & -BK \end{bmatrix} && \in \mathbb{R}^{4\times 6} \\ C_{\text{ctrl}} &= K && \in \mathbb{R}^{1\times 4} \\ D_{\text{ctrl}} &= \begin{bmatrix} 0_{1\times 2} & -K \end{bmatrix} && \in \mathbb{R}^{1\times 6}. \end{aligned} \tag{4}$$

MATLAB COMMANDS

`obsv(A,C)`	Observability matrix of system (A, B, C, D).

Create a blank script `lab4.m`.

- From your lab 3 script `lab3.m`, copy the initial part of the code that defines numerical variables `M,m,l,g,alpha1,alpha2`, saves the function `cartpend.m`, and generates matrices (A, B) of the linearization of the cart-pendulum model at the origin.

[2] Recall that x_d is a vector whose only nonzero component is the first one. The first column of A is zero because the differential equation $\dot{x} = f(x, u)$ of the cart-pendulum model is independent of z and so the first column of the Jacobian $\partial f/\partial x$ is zero. Thus $Ax_d = 0$.

- Define the output matrix `C` as above.
- Using the commands `rank` and `obsv` check that the linearization is observable.
- Using the command `place`, design a gain $K \in \mathbb{R}^{1\times 4}$ such that the eigenvalues of $A + BK$ are approximately all at -5. Recall that Matlab assigns the eigenvalues of $A - BK$ so you need to flip the sign of K you get from the command `place`.
- Design a gain $L \in \mathbb{R}^{4\times 2}$ such that the eigenvalues of $A - LC$ are approximately all at -10. Here you will need to use duality, see the textbook and the lecture notes.
- Using the gains K and L just designed, create matrices $(A_{\text{ctrl}}, B_{\text{ctrl}}, C_{\text{ctrl}}, D_{\text{ctrl}})$ as in (4).
- Define an initial condition vector `x0=[0; 0; pi/24; 0]` and the voltage saturation level `Ulim=5.875`.

4.3 Simulation comparison of state and output feedback controllers

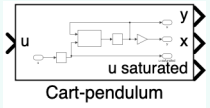
Now you will simulate the cart-pendulum robot with state feedback controller (2) and its output feedback counterpart (3). You'll compare state and output feedback controllers.

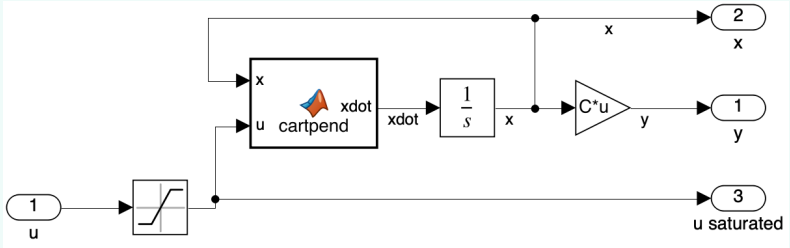
Simulink blocks (inside the Library Browser)

`Ports & Subsystems` → `Subsystem` create a subsystem block

The Simulink diagram `lab4_prep`

- Create a blank Simulink diagram called `lab4_prep`.
- 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`.
- Create a blank subsystem block (see below, left-hand side) which we'll call `Cart-pendulum`. Inside it, draw the diagram depicted below on the right-hand side.

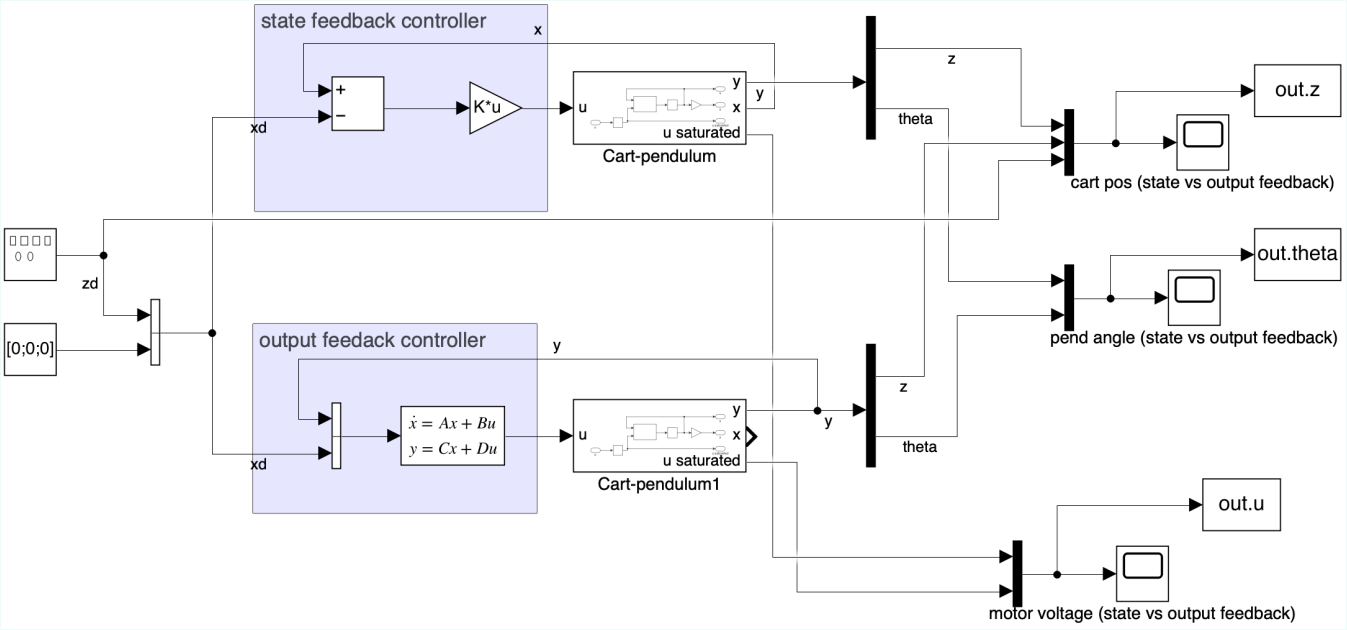




- Inside the subsystem, ensure the following:

- Many of the blocks aside from the input source labelled `u` and the output sinks labelled `x,y,u saturated` come from your previous diagram `lab3_nonlinear`. Cut and paste them from that diagram to the subsystem block. Double-check that various parameters in the blocks are the same as before: the initial condition of the integrator should be `x0`. The saturation levels should be $\pm$`Ulim`.
- The output gain block should refer to the matrix `C` and should perform a matrix multiplication, as opposed to an element-wise multiplication.
- Label the input and three outputs exactly as indicated in the figure.

- Return to the main Simulink diagram `lab4_prep`. Using two copies of the subsystem block you've just created, create a diagram that looks like this.



- In the diagram, note the following:

  - The signal generator block should be set to generate a square wave reference signal $z_d(t)$ with amplitude 0.1 m (10 cm) and frequency 0.1 Hz.
  - The reference signal is fed to two control systems in parallel. The upper system uses the state feedback controller (2), while the lower system uses the output feedback controller (3).
  - The reference signal $z_d(t)$ is padded with three zeroes via a `Vector Concatenate` block. This gives the state reference signal $x_d(t)$.
  - In the upper path, $x_d(t)$ is subtracted from x and the result is sent to a gain block K implementing the state feedback controller (2). As usual, make sure that the gain block performs a matrix multiplication.

  - In the lower path, the plant output signal $y(t)$ is concatenated with $x_d(t)$ to form the input for the output feedback controller (3). This controller is implemented via a state space LTI block with matrices $(A_{\text{ctrl}}, B_{\text{ctrl}}, C_{\text{ctrl}}, D_{\text{ctrl}})$ defined in your script.
  - The outputs of the two cart-pendulum systems are demuxified and sent to scopes, together with the saturated control input signals.
  - There are three `To Workspace` blocks. Label them the same way as the figure. You'll need the labels in your Matlab script later.

- Run the diagram for 30 seconds with the values of x_0, K, L defined by your script `lab4.m`. You should observe that after a transient both controllers perform acceptably, and their output signals come to coincide. You'll make further consideration during your lab sessions.

Now you'll tune the parameters of your output feedback controller so as to meet certain performance specifications.

MATLAB COMMANDS	
`out=sim('diagram.slx',T)`	Runs the Simulink diagram `diagram.slx` for `T` seconds and returns outputs in a structure `out`

Return to your script `lab4.m`.

- Using the command `sim`, run your Simulink diagram `lab4_prep` for 30 seconds and save the outputs in a structure `out`.
- Using the plotting code suggested in Section 4.4 of the lab 4 document, create a figure containing the signals $(z(t), \theta(t), u(t))$ using both the state and output feedback controllers.
- Now change `L` to assign the eigenvalues of $A - LC$ at around -40. Rerun the simulation and plot again the signals $(z(t), \theta(t), u(t))$. Comparing the two figures you've created, determine the effects of moving the eigenvalues of $A - LC$ from -10 to -40.
- Now tune your feedback gain K, without changing L (L should be such that the eigenvalues of $A - LC$ are approximately equal to -40). For the tuning, using both eigenvalue assignment and LQR, you need to obtain an output feedback controller meeting two specifications:
  (a) The settling time T_s for the cart displacement $z(t)$ is less than 1 second.
  (b) The control signal $u(t)$ should not display excessive saturation, i.e., saturation for too long time. We deliberately leave this specification vague. The point is that your controller should not be too "aggressive".

For the calculation of T_s, look at the signal $z(t)$ over the time interval 10 to 15 seconds corresponding to the second period of the square wave reference signal. In this interval, $z(t)$ starts at approximately 0.1 m and transitions to a new steady-state value of -0.1 m. Thus, T_s is the time it takes the error $|z(t) - (-0.1)|$ to reach and stay below 2% of the total step variation, 0.2 m, i.e., such that $|z(t) + 0.1| < 0.02 \cdot 0.2$.

Output 1.

- Provide one figure overlaying the signals $(z(t), \theta(t), u(t))$ corresponding to state and output feedback controllers when the eigenvalues of $A + BK$ are at -5 and those of $A - LC$ are at -10.
- Discuss the differences between state feedback and output feedback solutions in the figures above.
- Provide a second figure when the eigenvalues of $A - LC$ are moved to -40, and discuss the changes you observe. Print T_s and T_{sat} for this latter simulation.
- Discuss how you tuned K to meet the specifications. Provide the parameters for the tuning (either the eigenvalues or the Q, R matrices) and provide T_s for your output feedback controller.

5 Lab activity

The objectives of this lab activity are:

- To implement the output feedback controller (3) in Arduino and run experiments to test its real-world performance.
- To add integral action to your output feedback controller and verify experimentally the performance improvement that this modification yields.

5.1 Physical experimentation of output feedback controller

Preliminary steps and safety precautions:

- 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 `lab4.ino`

- Locate the provided Arduino file `lab4.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 $(A_{\text{ctrl}}, B_{\text{ctrl}}, C_{\text{ctrl}}, D_{\text{ctrl}})$ are determined by your Matlab script `lab4.m`.
- In Section 5, compute the plant output (the cart position z and pendulum angle θ) and update the global variable `y`. The cart encoder count is stored in the variable `encoderPos`. The pendulum encoder count is stored in `encoderPos_P`. To convert this reading into the plant output, you need to multiply `encoderPos` by the parameter `K_encoder` and multiply `encoderPos_P` by `K_encoder_P`.
- Still in Section 5, declare the vector $x_d(t) = [z_d(t) \;\; 0 \;\; 0 \;\; 0]^\top$. The square wave signal $z_d(t)$ is stored in the global variable `z_d`.
- Still in Section 5, compute the control input.
  - To begin, you need to write code computing the derivative $\dot{\hat{x}}$ according to (3). In order to compute $\dot{\hat{x}}$ via (3), 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 and x_d are the variables that you defined previously.
  - Below the Euler update step, 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 `lab4_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.

Running the Matlab script `lab4_plot.m` will now run the experiment. 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).

(b) The settling time T_s 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.

Tuning the controller means performing these steps:

1. Tune the controller gains via the script `lab4.m`. You can tune both gains K and L but *be careful about moving the observer eigenvalues to the left of -50 as this could cause noise amplification that would be transmitted to the DC motor and could damage it.*
2. Simulate the controller via the Simulink diagram `lab4_prep` to determine its theoretical performance.
3. Update the controller in Section 5 of the Arduino code `lab4.ino`.
4. Run the controller via the Matlab script `lab4_plot.m` and compare the experimental data to the simulation results.

Output 2.

- Explain the trial and error procedure that you followed to get the best performance on the experimental setup, and provide the parameters of the controllers (either the eigenvalues or the Q, R matrices) that gave the best results.
- Provide an experimental plot of the signals $(z(t), \theta(t), u(t))$, 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 signals $(z(t), \theta(t), u(t))$ 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.

5.2 Output feedback control with integral action

Return to the model of the cart-pendulum robot linearized at the origin,

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

with the objective of making $z(t) \to z_d$, where z_d is assumed to be constant at first, and later is replaced by the usual square wave reference signal.

The idea behind integral control is to augment the plant with an integrator driven by the tracking error $z_d - z$:

$$\dot{x} = Ax + Bu$$
$$\dot{\xi} = z_d - C_1 x,$$

where $C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$ is the vector extracting z from x. We have an augmented plant with state $X = (x, \xi)$ and dynamics

$$\dot{X} = \underbrace{\begin{bmatrix} A & 0_{4\times 1} \\ -C_1 & 0 \end{bmatrix}}_{\bar{A}} X + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\bar{B}} u + \begin{bmatrix} 0_{4\times 1} \\ 1 \end{bmatrix} z_d. \tag{5}$$

We check whether the pair $(\bar{A}, \bar{B})$ is controllable (it is), and design a state feedback controller $u = \bar{K}X$ assigning the eigenvalues of $\bar{A} + \bar{B}\bar{K}$. Partitioning $\bar{K}$ as $\bar{K} = [K_x \;\; K_\xi]$, with K_x representing the first four entries of $\bar{K}$ and K_ξ representing the last entry, the state feedback controller is

$$u = K_x x + K_\xi \xi.$$

The closed-loop system is a stable LTI system with a bounded input. Stable LTI systems are BIBO stable for any input and output matrices, so we know that $X(t)$ is bounded. Moreover, since the input is constant, $X(t)$ will asymptotically converge to a constant, i.e., to an equilibrium. At equilibrium, $\dot{\xi}$ must be zero, so

$$\dot{\xi} = z_d - z = 0,$$

and asymptotic tracking is achieved.

As usual, in the controller above we replace the constant z_d by a *piecewise constant* square wave $z_d(t)$, and we replace the state signal $x(t)$ by an estimate coming from the observer that you've already designed earlier. We thus arrive at the *output feedback integral controller*

$$\begin{aligned} \dot{\xi} &= z_d - z \\ \dot{\hat{x}} &= A\hat{x} + Bu + L(y - C\hat{x}) \\ u &= K_x \hat{x} + K_\xi \xi. \end{aligned} \tag{6}$$

This controller has three inputs, the measurement output $y \in \mathbb{R}^2$ and the tracking error $z_d(t) - z$, and one output, u.

Now you'll design this new controller in Matlab.

Return to your script `lab4.m`

- Define matrices `Abar`,`Bbar` as in (5).
- Check that the pair `(Abar,Bbar)` is controllable.

- Design a gain `Kbar` using LQR with matrices

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad R = 0.01.$$

  Double-check that the eigenvalues of `Abar + Bbar*Kbar` are in $\mathbb{C}^-$. (Recall that you need to flip the sign of `Kbar` coming out of the Matlab command `lqr`).

- Extract from `Kbar` the first four components in a vector `Kx` and the last component in a vector `Kxi`.

Next, we return to Arduino.

Duplicate your file `lab4.ino` and save it as `lab4_integral.ino`.

- In Section 5 of the code, implement the output feedback integral controller (6). Note that the controller has now *five* states, $\xi \in \mathbb{R}$ and $\hat{x} \in \mathbb{R}^4$.
- 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).
- Tune the integral controller by changing the matrices Q and R of the LQR controller and possibly the observer gain L until you achieve the *best performance you can* according to the two metrics described in the previous section. Concerning the observer gain L, *be careful about moving the observer eigenvalues to the left of -50 as this could cause noise amplification that would be transmitted to the DC motor and could damage it.*

Output 3.

- Explain the trial and error procedure that you followed to get the best performance on the experimental setup, and provide the parameters of the controllers (Q, R matrices) that gave the best results.
- Provide an experimental plot of the signals $(z(t), \theta(t), u(t))$, as well as the settling time. Discuss the results: whether they are satisfactory, whether they conform to your expectations, and so on.
- Compare your experimental results with the experimental results that you obtained in the previous section. Does the output feedback integral controller perform better than standard output feedback controller?