CONTROL SYSTEMS LABORATORY
ECE410F
LAB3 : Ball and Beam Experiment

1 Purpose

The purpose of this experiment is to familiarize you with modelling nonlinear systems using SIMULINK, performing linearization, designing and evaluating different control laws based on pole placement and optimal control using MATLAB.

2 Introduction

The ball and beam experiment consists of a ball placed on a grooved beam such that the ball is allowed to roll with one degree of freedom along the length of the beam. A rotating shaft is attached to the beam at its midpoint. This shaft allows the beam to tilt by an angle θ in order to roll the ball along the beam. A gearbox attaches the shaft to a motor which creates the torque required to tilt the beam. When current u is applied, the motor rotates the shaft and tilts the beam from its horizontal position and the ball rolls down the beam due to gravity. The following figure illustrates the physical system.

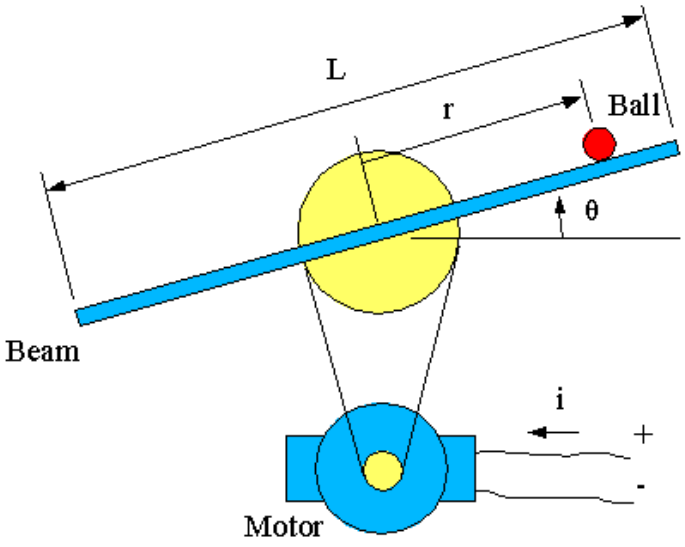

Figure 1: Diagram of the ball and beam system

The system sensors only provide the position of the ball r and the angle of the beam θ.

2.1 Nonlinear Dynamics

The differential equations governing the dynamics of the system are

$$\ddot{\theta} = \frac{K_m u - mgr\cos\theta}{J_R} \tag{1}$$

$$\ddot{r} = \frac{-mg\sin\theta - F_r\dot{r}}{\frac{J_B}{R^2} + m} \tag{2}$$

where

K_m	Motor and gearbox constant
m	Mass of ball
g	Gravitational constant
J_R	Inertia of beam
F_R	Coefficient of friction for the ball
J_B	Inertia of ball
R	Radius of ball
u	Control input current

2.2 Control Objectives

The control design objectives in this experiment are

(a) Stabilize the nonlinear system in the neighbourhood of an equilibrium point.

(b) Control the position of the ball to track an input reference r_o.

(c) Ensure the ball does not fall off the beam for a given step input.

(d) Ensure the beam does not slope more than 25 degrees from the horizontal for a given step input, i.e. $\|\theta\| < 25°$. This guarantees the system does not venture too far away from the operating point or linear region.

(e) Design the controller such that the ball settles to its desired position along the beam within 4 seconds for a given step input, i.e. settling time < 4 seconds.

2.3 Parameter Values

Table 1 gives the parameter values to be used for this experiment.

Parameter	Value	Units
K_m	7.35	$\frac{Nm}{A}$
m	0.050	kg
g	9.8	$\frac{m}{s^2}$
J_R	0.049	kgm^2
F_R	0.07	$\frac{kg}{s}$
J_B	8e-6	kgm^2
R	0.02	m
L	1	m

Table 1: Physical parameters for the ball and beam system

3 Preparation

All preparation calculations should be done manually without the assistance of a computer.

(a) Since this is a course about *linear* control systems the first step is to linearize the nonlinear dynamics (1)-(2) about an operating point $x_o = (\dot{r}_o, r_o, \dot{\theta}_o, \theta_o)$. Define the state variable $x = [\ \dot{r}\ r\ \dot{\theta}\ \theta\ ]^T$. First write a nonlinear output equation of the form $y = g(x)$ where $g : \mathbb{R}^4 \to \mathbb{R}^2$, based on the placement of the system sensors. Recall that the sensors provide the position of the ball r and the angle of the beam θ. Next write the equations (1)-(2) in the state space form

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

You will linearize this model about a given operation point. First, determine $(\dot{r_o}, \dot{\theta}_o, \theta_o)$ and also u_o as a function of r_o assuming that $r = r_o$ at equilibrium. Next, compute the linearized model

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u} \tag{3}$$
$$\tilde{y} = C\tilde{x}\,, \tag{4}$$

where $\tilde{x} = x - x_o$, $\tilde{y} = y - y_o$, and $\tilde{u} = u - u_o$. Do this calculation assuming all parameter values are unknown, i.e. symbolically.

(b) Using numeric values for all the parameters, check if the linearized system (3)-(4) is controllable and observable.

(c) Design a state feedback controller $\tilde{u} = K\tilde{x}$ for (3)-(4), assuming all states of $\tilde{x}$ are available for measurement. Do this using numeric values for all parameters (i.e. not symbolically). Place the poles of the closed-loop system at -2, -4, -10 and -20.

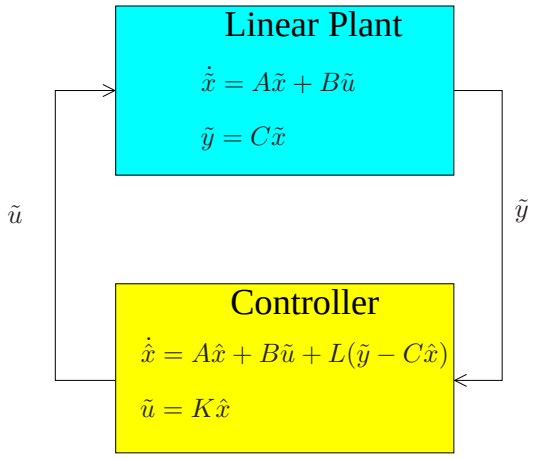

Figure 2: Block Diagram of closed-loop linearized system

(d) Assuming that we are using an observer with a gain matrix L, determine the closed-loop state-space model of the system shown in Figure 2, assuming that the combined state of the closed-loop system is $[\ \tilde{x}^T\ \hat{x}^T\ ]^T$ and the output is $\tilde{y}$. Do this symbolically in terms of the variables A, B, C, K and L. Also, compute symbolically the transfer function of the "compensator" (observer+controller) from its input $\tilde{y}$ to its output $\tilde{u}$. Note that since the design of the observer gain L is tedious, you will design the observer later in Section 4 with the aid of MATLAB.

(e) Study and familiarize yourself with the MATLAB commands described in the attached Appendix.

(f) Study your notes on optimal control as you will be designing an optimal controller in parts (d) and (e) of Section 4.

4 Experiment

Show your work by providing all scripts and SIMULINK diagrams used to perform the experimental procedure. Make sure plots have labels, titles and legends. Hints are provided at the end of each part indicating which MATLAB commands to use. ***Work in groups of two.***

(a) Check that the system is controllable and observable using MATLAB. Compute the gains of the state-feedback controller using MATLAB. Compare the results you obtain with MATLAB with the results you computed in the lab preparation. Design an observer by placing two poles at -40 and the other two poles at -50. Check that the closed-loop poles are placed correctly by finding the eigenvalues of $A - BK$ and $A - LC$.

Using your state-space equations for observer and controller, convert the state-space model of the compensator to a transfer function with input $\tilde{y}$ and output $\tilde{u}$. Determine the poles and zeros of the transfer function. What are the open-loop poles of the plant? [MATLAB commands: `place`, `acker`, `eig`, `ctrb`, `obsv`, `rank`, `ss`, `tf`, `ss2tf`, `zpk`]

(b) The linearized model (3)-(4) is special for several reasons:

1. The system matrices A, B, and C are independent of the choice of operating point x_o.
2. The system satisfies the property $Ax_o + Bu_o = 0$.
3. Letting $y_o := g(x_o)$, the system satisfies the property that $y_o = Cx_o$.

Note that these properties do not hold for general nonlinear systems.

Thus, starting from (3) we have

$$\dot{x} = \dot{\tilde{x}} = A(x - x_o) + B(u - u_o) = Ax + Bu\,.$$

Similarly one can show that the $y = Cx$. This means we can express the linearized system directly in terms of the original variables x, y, and u. A modified block diagram showing this representation of the linearized model is in Figure 3.

Using the arrangement shown in Figure 3, simulate the closed-loop system using your design for controller and observer from part (a). Modify the SIMULINK model file *ballandbeamlinear.mdl* provided, by filling in the controller block with your designed controller. Then apply a step reference input u_o corresponding to going from $r_o = 0$cm to $r_o = 40$cm. If necessary, tune the state-feedback pole locations such that the design specifications are met. Perform the simulation twice. The first time set the initial conditions of the plant to $(\dot{r}_o, r_o, \dot{\theta}_o, \theta_o) = (0, 0, 0, 0)$ and the observer initial conditions to $(\dot{\hat{r}}_o, \hat{r}_o, \dot{\hat{\theta}}_o, \hat{\theta}_o) = (0, 0, 0, 0)$. The second time set the initial conditions of the plant to $(\dot{r}_o, r_o, \dot{\theta}_o, \theta_o) = (0, 0, 0, 0)$ and the observer initial conditions to $(\dot{\hat{r}}_o, \hat{r}_o, \dot{\hat{\theta}}_o, \hat{\theta}_o) = (0, 0.2, 0, 0.4)$. Plot the step reference signal alongside the output r from both simulations. Next, plot the step reference signal alongside the output θ from both simulations. Do the two simulations differ? Why or why not? You should have two plots for this part. [MATLAB commands: `plot, legend, xlabel, ylabel, title, hold on, sim`]

(c) Repeat part(b) except now use the nonlinear model of the ball and beam system. The setup is shown in Figure 4.

Modify the SIMULINK model file *ballandbeamnonlinear.mdl* by filling in the controller block with your designed controller and apply the same step input. If necessary re-tune your pole locations if the design specifications are not met. Obtain the two plots for the two different initial conditions as was done in

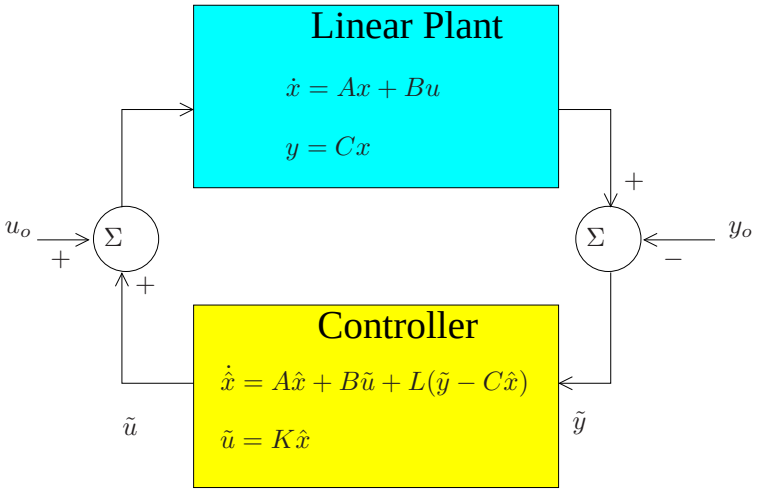

Figure 3: Alternate block diagram of closed-loop linearized system

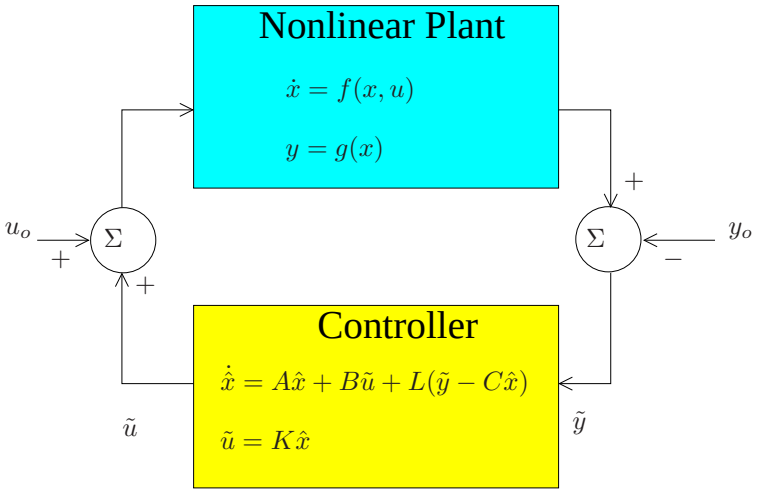

Figure 4: Block diagram of closed-loop nonlinear system

part (b). Next, for your third graph, plot the nonlinear outputs r and θ and the linear outputs r and θ from part (b), on the same axis for when the initial conditions of the plant and observer are both zero. How do the results compare to that of part (b)? You should have three plots for this part. [MATLAB commands: `plot, legend, xlabel, ylabel, title, hold on, sim`]

(d) Design an optimal controller using MATLAB such that along with your observer, the control objectives are satisfied for the linear model. For your optimal control weighting matrices, initially start with $R = 1$ and

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

When designing your optimal controller, you may have to reposition your observer poles making sure that they are roughly twice the real part of the optimal state-feedback poles. Use the same step

reference input as in part (b) and (c) and set the initial conditions of the plant and observer to zero. Plot the step reference signal alongside output r and θ. What is your final Q matrix? What is your final R? Where are your optimal controller poles located? How do the results compare to part (b) where you used regular pole placement? You should have one plot for this part. [MATLAB commands: `lqr, acker, eig, place, diag, plot, legend, xlabel, ylabel, title, hold on, sim`]

(e) Run your optimal controller on the nonlinear model of the system. Are the control objectives satisfied? Compare the results of the optimal controller for part (d) and part(e) by plotting output r and θ on the same figure. You should have one plot for this part. [MATLAB commands: `lqr, acker, eig, place, diag, plot, legend, xlabel, ylabel, title, hold on, sim`]

Show your results for Parts (a)-(e), including plots, to the Lab TA before leaving the lab.

5 Appendix - MATLAB Commands

The following is a list of MATLAB commands which may be useful for completing the lab.

- `acker` - Same functionality as the command `place`. e.g. `K = acker(A,B,[-1 -2])`
- `clear all` - Clears the workspace of all variables
- `close all` - Closes all figures
- `ctrb` - Calculates the controllability matrix. e.g. `Qc = ctrb(A,B)`
- `diag` - Creates a diagonal matrix with diagonal elements taken from a vector. e.g. `Q = diag([1 2 3 4])`
- `eig` - Determines the eigenvalues of a matrix. e.g. `poles = eig(A)`
- `figure` - Opens a new blank figure for plotting
- `grid on` - Draws a grid on a plot
- `help` - Access to help on MATLAB commands. e.g. `help place`
- `hold on` - Holds the current figure such that subsequent plots can be drawn on the same figure without erasing the previosuly drawn plots.
- `legend` - Creates a legend for a plot. e.g. `legend('Reference Step', 'Ball Position')`
- `load` - Loads mat files into the workspace. e.g. `load datafile`
- `lqr` - Computes the optimal state-feedback matrix K for a given cost function. e.g. `K = lqr(A, B, Q, R)`
- `obsv` - Calculates the observability matrix. e.g. `Qo = obsv(A,C)`
- `place` - Performs pole placement to determine the state-feedback matrix. e.g. `K = place(A,B,[-1 -2])`
- `plot` - Creates a plot on a figure. e.g. `plot(time, output, 'b')`
- `rlocus` - Draws a root locus of a system. e.g. `rlocus(sys)`
- `save` - Saves a variable or the entire workspace to a mat file. e.g. `save datafile variable`
- `sim` - Runs a SIMULINK model file. e.g. `sim('ballandbeamlinear', 10)`

- `ss` - Creates a state-space system. e.g. `sysss = ss(A,B,C,D)`
- `step` - Simulates a step response for a system. e.g. `[Y,T] = 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)`
- `whos` - Displays the variables currently present in the workspace.
- `xlabel/ylabel` - Creates a label for the axis of a plot. e.g. `xlabel('Time (seconds)')`
- `zpk` - Creates zero-pole-gain models or converts to zero-pole-gain format. e.g. `syszpk = zpk(sometf)`