EE222 Ball and Beam Project: Phase 1

GitHub Repository

Soomi Lee, Arvind Kruthiventy, Emily Lukas

4 April 2025

1 Introduction

In the first phase of this project, we designed and simulated controllers for the nonlinear ball-and-beam system. The primary objective is to track a time-varying reference trajectory of the ball's position along the beam, while minimizing energy consumption and ensuring system safety based on the real physical constraints. The system is actuated via a servo motor that adjusts the beam angle, which then affects the ball's position through gravitational and inertial effects. The system dynamics are defined by a set of nonlinear equations, where the state vector consists of x_1 (ball position), x_2 (ball velocity), x_3 (beam angle), and x_4 (beam angular velocity).

Two control approaches were explored. The first employs feedback linearization with LQR, where the control law is derived by canceling the system's nonlinearities through an analytical inversion of the dynamics. This yields a linearized system that is then regulated using LQR. The second method adopts a trajectory linearization strategy: the system is linearized around the current state and reference trajectory, and a local Jacobian is computed to design an LQR controller. Both controllers are implemented in MATLAB using `studentControllerInterface.m`, just with a flag to switch between the two strategies. Additionally, a Luenberger observer was implemented to estimate unmeasured states (x_2 and x_4) from the available sensor measurements (ball position z and beam angle θ).

The project is motivated by the need to achieve low tracking error, reduced energy consumption, and strict adherence to safety constraints (e.g., ensuring that the ball does not reach the beam end and the servo angle remains within operational limits).

2 Project Context and Modeling

The ball-and-beam system is a classic example used in nonlinear control courses (like this great course) to illustrate advanced control techniques. The dynamics are derived from the kinematics of the ball and beam, considering gravitational, centrifugal, and rolling resistance effects. The system model was given to us by:

$$
\begin{aligned}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= \frac{5g}{7}\frac{r_g}{L}\sin x_3 - \frac{5}{7}\left(L - x_1\right)\left(\frac{r_g}{L}\right)^2 x_4^2 \cos^2 x_3, \\
\dot{x}_3 &= x_4, \\
\dot{x}_4 &= -\frac{x_4}{\tau} + \frac{K}{\tau}u,
\end{aligned}
$$

where u is the servo voltage, r_g is the servo arm length, L is the beam length, g is the gravitational constant, K is the motor constant, and τ is the motor time constant. Note that only x_1 (ball position) and x_3 (beam angle)

are measured. The given performance metrics include the tracking error,

$$J_{\text{tracking}} = \frac{1}{T}\int_0^T (z(t) - z_r(t))^2 dt,$$

the energy consumption,

$$J_{\text{energy}} = \frac{1}{T}\int_0^T u(t)^2 dt,$$

and a binary safety indicator J_{safety} that penalizes trajectories where the ball or beam exceed safe limits and is visualized in the simulations in red.

3 Methods

3.1 Theoretical Background and Implementation

3.1.1 Feedback Linearization

Feedback linearization is an advanced control technique used to cancel the nonlinearities inherent in a system, making it linear and thus controllable by linear control methods. Consider a nonlinear system:

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

with output

$$y = h(x). \tag{2}$$

The technique involves differentiating y until the control u explicitly appears. If u appears after r derivatives, then:

$$y^{(r)} = L_f^r h(x) + L_g L_f^{r-1} h(x)\, u, \tag{3}$$

where $L_f h(x)$ and $L_g h(x)$ denote the Lie derivatives of $h(x)$ along f and g.

A virtual control input v is defined as:

$$v = y_{\text{ref}}^{(r)} + K_e e, \tag{4}$$

with the tracking error $e = y - y_{\text{ref}}$ and gain K_e. The actual control input is then given by:

$$u = \frac{v - L_f^r h(x)}{L_g L_f^{r-1} h(x)}. \tag{5}$$

In our implementation, we compute a series of Lie derivatives (via functions `lie1_func`, `lie2_func`, `lie3_func`, and `lie4_func`) up to the fourth derivative. We ensured the nonlinear cancellation by functions $\phi(x)$ and $\psi(x)$ such that:

$$u = \frac{v - \phi(x)}{\psi(x)}.$$

This formulation analytically cancels the nonlinear dynamics, leaving a linear relationship between the virtual input v and the output. In our controller, the derivatives are computed manually and the resulting control u is stored in the function `control_func`.

3.2 Linearization about a Trajectory

For our second controller, we linearized the model around the reference trajectory. This can be easily accomplished by taking the Jacobian of the system matrices A, B about the reference trajectory $r(t), v(t)$. The Jacobian of the system is presented below:

$$J_A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ C_1 & 0 & C_2 & C_3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1/\tau \end{bmatrix}$$

$$C_1 = ((5/7)x_1)((r_g/L)^2)(x_4^2)cos(x_3)^2 \tag{6}$$

$$C_2 = (5gr_g/(7L))cos(x_3) + (5/7)((L/2) - x1)((r_g/L)^2)x_4^2 sin(2x_3) \tag{7}$$

$$C_3 = (10/7)((L/2) - x_1)((r_g/L)^2)x_4 cos(x_3)^2 \tag{8}$$

$$J_B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -k/\tau \end{bmatrix}$$

The Jacobians, J_A, and J_B are evaluated at the reference state $r(t)$ and control input $v(t)$ to get the linearized system and then discretized through the use of the zero-hold relation for state space systems. This discretization of the system is passed to the LQR for optimal state feedback.

3.2.1 Linear Quadratic Regulator (LQR)

For a linear system of the form:

$$\dot{x} = Ax + Bu, \tag{9}$$

the LQR method minimizes the cost function:

$$J = \int_0^\infty \left(x^\top Q x + u^\top R u\right) dt, \tag{10}$$

where $Q \geq 0$ and $R > 0$ are weighting matrices. The solution involves solving the Algebraic Riccati Equation:

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0, \tag{11}$$

from which the optimal gain is computed as:

$$K = R^{-1}B^\top P. \tag{12}$$

The control law is then:

$$u = -Kx. \tag{13}$$

Our implementation employs a custom ARE solver (`ARESolve`) to compute K in discrete time for compatibility with Simulink, since we had some issues with our original method of using the symbolic math toolbox. To find the optimal gain, we first iterate backwards repeatedly repeatedly to find the steady state performance index, P_{ss}, for a large number of steps. Equation 14 presents the relationship between $P(k+1)$ and $P(k)$ which can be leveraged to repeatedly compute $P(k)$ until convergence.

$$P(k) = A^T P(k+1)A + Q - A^T P(k+1)B(R + B^T P(k+1)B)B^T P(k+1)A \tag{14}$$

After P has converged, the optimal state feedback gain can be readily computed.

3.2.2 Luenberger Observer

Due to limited sensor measurements (only ball position and beam angle), we implemented a Luenberger observer to estimate the full state x. For the system:

$$\dot{x} = Ax + Bu, \quad y = Cx, \tag{15}$$

the observer is designed as:

$$\dot{\hat{x}} = A\hat{x} + Bu + L\left(y - C\hat{x}\right), \tag{16}$$

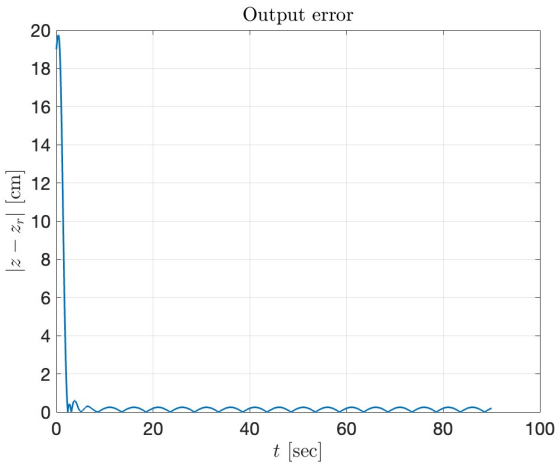
where $\hat{x}$ is the estimated state and L is chosen to ensure rapid convergence of the estimation error.

4 Results

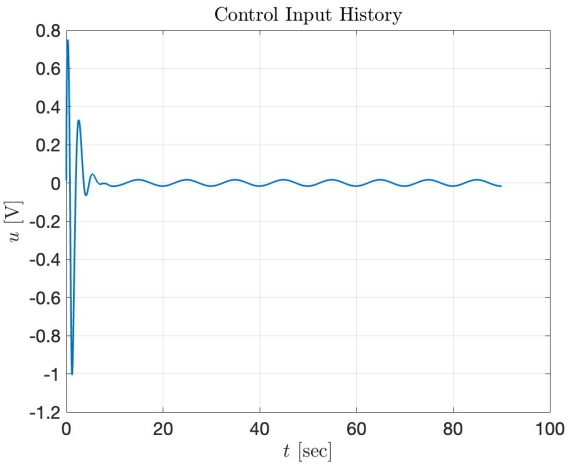
Table 1 summarizes the performance metrics for both controllers with sine wave and square wave inputs. Figures 1 and 2 show the plots for the sine wave input, while Figures 3 and 4 display the corresponding results for the square wave input. The videos of the animation are linked in the Github repo.

Metric	**Controller 1 (Sine)**	**Controller 2 (Sine)**	**Controller 1 (Square)**	**Controller 2 (Square)**
Average Tracking Error	0.0005	0.0004	0.0020	0.0014
Average Energy Consumption	0.0091	0.0309	0.0316	0.1062
Safety Constraint Violation	0	0	0	0
Tracking Cost	0.94	0.67	3.53	2.51
Energy Cost	0.05	0.15	0.16	0.53
Safety Cost	0.00	0.00	0.00	0.00
Total Score	0.98	0.82	3.68	3.04

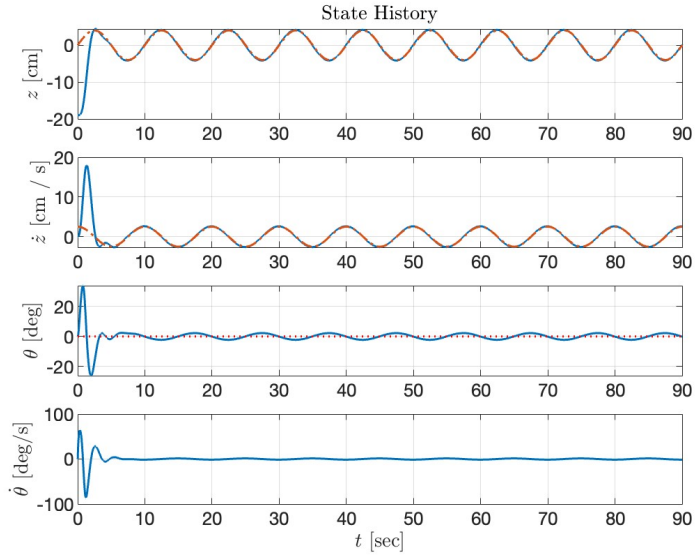
Table 1: Performance metrics for the two controllers with sine and square wave inputs.



(a) *Tracking error for Controller 1.*

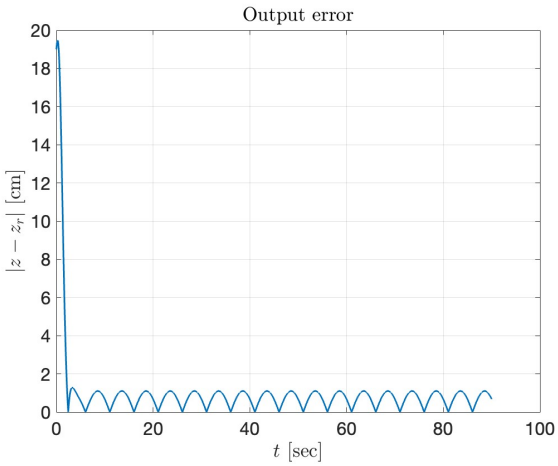


(b) *Control input history for Controller 1.*

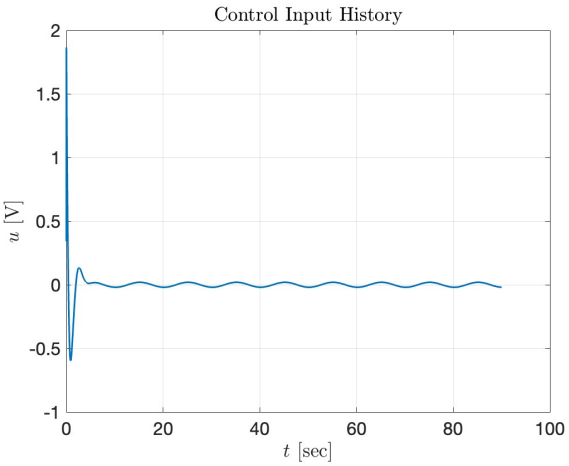


(c) *State history for Controller 1.*

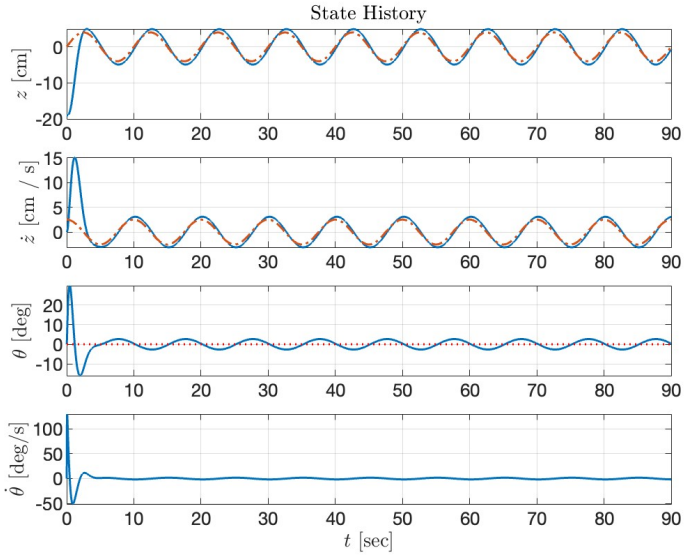
Figure 1: Performance plots for Controller 1 with sine wave input.



(a) *Tracking error for Controller 2.*

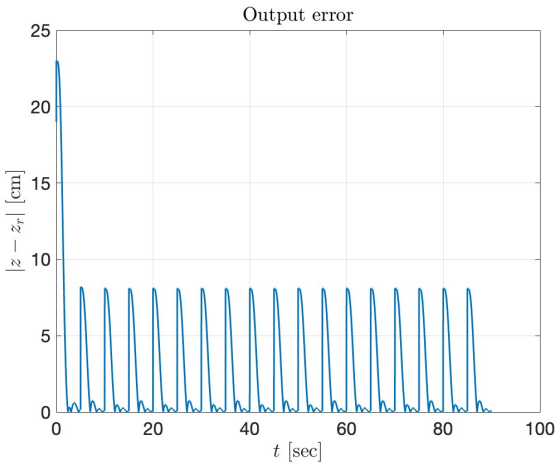


(b) *Control signal behavior for Controller 2.*

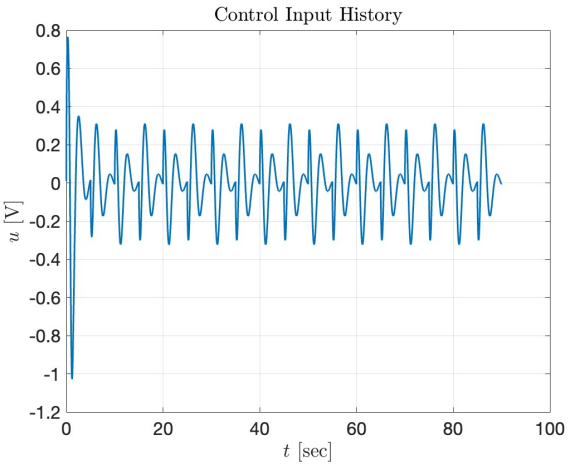


(c) *State history for Controller 2.*

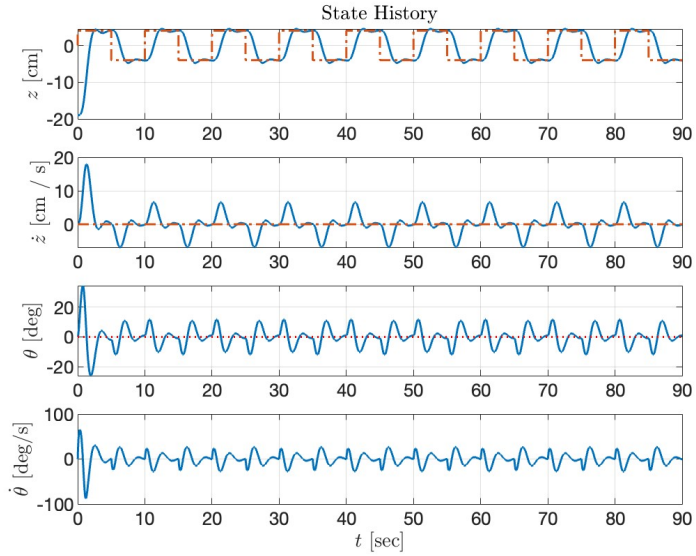
Figure 2: Performance plots for Controller 2 with sine wave input.



(a) *Tracking error for Controller 1.*

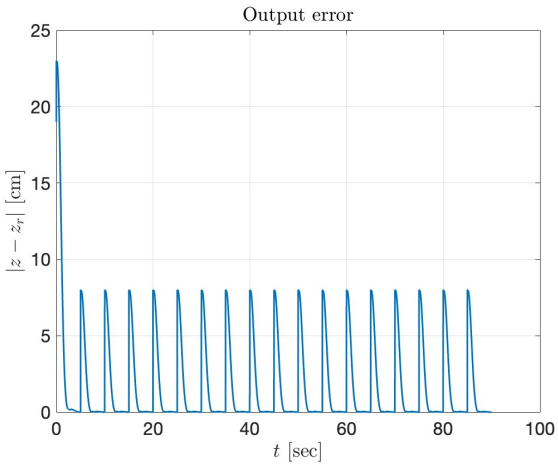


(b) *Control input history for Controller 1.*

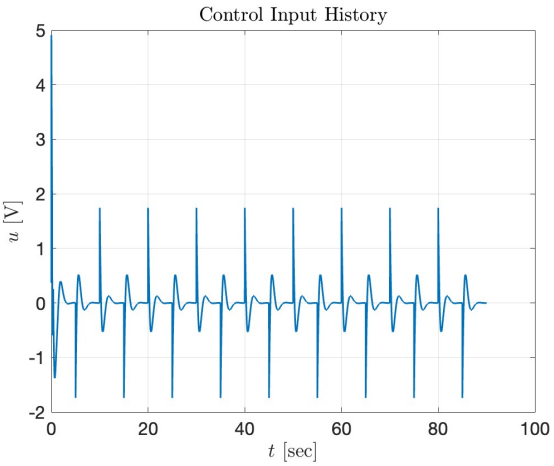


(c) *State history for Controller 1*

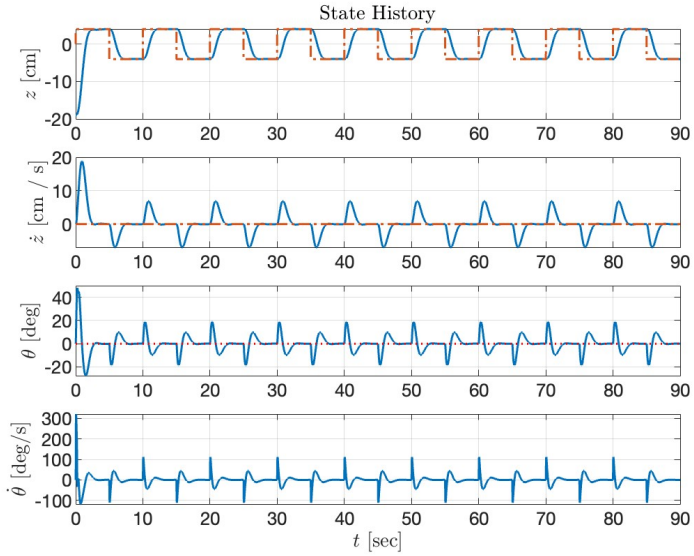
Figure 3: Performance plots for Controller 1 with square wave input.



(a) *Tracking error for Controller 2.*



(b) *Control signal behavior for Controller 2.*



(c) *State history for Controller 2.*

Figure 4: Performance plots for Controller 2 with square wave input.

5 Discussion

The experimental results indicate distinct trade-offs between the two control strategies. Controller 1, based on feedback linearization with LQR, achieves low energy consumption and satisfactory tracking performance by canceling the nonlinearities and linearizing the system. In contrast, Controller 2 achieves marginally better tracking accuracy at the cost of higher energy consumption. Both controllers adhere to safety constraints, ensuring the ball remains within the operational region and the servo stays within its angular limits. Both of the controllers perform better for the sine wave than the square wave, likely due to the absence of corners.

Key challenges included ensuring compatibility between MATLAB and Simulink environments. Our initial design, which utilized symbolic computation and MATLAB's built-in LQR function, had to be modified to replace symbolic expressions with numerical functions and to implement a custom ARE solver. Discretization of the continuous-time dynamics further posed challenges in maintaining numerical stability while preserving the fidelity of the model. These modifications were essential for ensuring that the controller would interface correctly with the real hardware.

6 References

We primarily used the class notes and the directions of this project in our submission.