Nonlinear Control of Ball and Beam System

***Abstract*—We use the ball and beam system as a platform to apply the control algorithms from class to real-world hardware. This system exhibits interesting mathematical properties which theoretically invalidates the use of algorithms like input-output feedback linearization. Nonetheless, we demonstrate that input-output linearization achieves considerable performance through the use of a clever modeling approximation. We achieve asymptotic tracking in simulation, and we achieve a performance score of 2.87 on the Quanser hardware platform. Our implementation can be found here https://github.com/ian-chuang/EE222-Nonlinear-Systems-Ball-and-Beam-Project/tree/main.**

I. Introduction

The ball and beam system is a classic example of a nonlinear system, where the nonlinearities arise from the connection between the beam and the rotating servo motor.

Many algorithms have been proposed towards the control of nonlinear systems. The Jacobian Linearization approach involves locally approximating the system dynamics by a linear system and and deriving a control through state feedback. The Backstepping algorithm [1] repeatedly uses the principle of virtual control to arrive at single stabilizing control and simultaneously constructs a certificate Lyapunov function. Model Predictive Control (MPC) [2] casts the tracking objective as an optimization problem and uses optimization techniques to solve for control. Recently, there has been a lot of excitement around Deep Reinforcement Learning [3], where neural network controllers are used to find approximate solutions to optimal control problems. Input-output Linearization [4] offers a control paradigm that does not involve any system approximation. The ball and beam system is an interesting platform to study control design because it does not exhibit a well-defined relative degree. For this reason, Input-output Linearization control may not directly be applied.

Observer design for nonlinear systems is an interesting problem in its own right. The Kalman Filter [5] is an optimal filtering algorithm for linear systems in the presence of Gaussian process and observation noise. The Extended Kalman Filter (EKF) [6] for nonlinear systems builds off the Kalman Filter algorithm by using Jacobian linearization to arrive at local linear approximations for the system dynamics and observation models. The Luenberger Observer is an exponentially-converging state observer for linear systems. Unlike the Kalman Filter, random noise is not modeled in this framework. Similar to the EKF, the Extended Luenberger algorithm [7] for nonlinear systems uses Jacobian linearization to arrive at local linear approximations for the system dynamics and observation model and locally applies the Luenberger Observer algorithm.

In this lab, we set out to control the ball's position along the beam using the ball's position and servo angle as observations and the motor voltage as input. Section II outlines the derivation of the ball and beam state-space dynamical equations. In Section III, we discuss our implementation of observer and control designs. In Sections IV and V, we showcase our performance in simulation and on the Quanser hardware platform, respectively. Finally, in Section VI, we summarize our results and outline possible directions for improvement.

II. System Modeling

In this section, we detail the physical derivation of the ball and beam dynamical system. The hardware includes a metallic ball which rolls along the length of a beam. The beam's angle is modulated by a servo motor. We define the state of the system as

$$x = \begin{bmatrix} z, \dot{z}, \theta, \dot{\theta} \end{bmatrix}^\top$$

where z is the offset of the ball from the center of the beam, and θ is the angle of the arm of the servo motor with respect to the horizontal axis.

Let m be the mass of the ball, L is the beam length, r_b is the ball radius, α is the angle that the beam makes with the horizontal, and $J_b = \frac{2}{5} m r_b^2$ is the moment of inertia of the ball. The net force exerted on the ball is described by

$$m\ddot{z} = mg \sin\alpha - m\left(\frac{L}{2} - z\right)\dot{\alpha}^2 - \frac{J_b}{r_b^2}\ddot{z}, \tag{1}$$

which is a sum of gravitational, centrifugal, and resistance forces. We approximate $\alpha \approx \sin\alpha$ and $L\sin\alpha \approx r_g \sin\theta$, where r_g is the servo arm length. Using this, we equate

$$\sin\alpha = \frac{r_g}{L}\sin\theta, \quad \dot{\alpha} = \frac{r_g}{L}\cos\theta\,\dot{\theta}. \tag{2}$$

Lastly, we use the Quanser motor dynamics

$$\tau\ddot{\theta} = -\dot{\theta} + KV \tag{3}$$

where V is the voltage applied to the motor, τ is the delay constant, and K is the motor constant.

We join the expressions above in a concise state-space form,

$$\dot{x}_1 = x_2 \tag{4}$$

$$\dot{x}_2 = \frac{5gr_g}{7L}\sin x_3 - \frac{5}{7}\left(\frac{L}{2} - x_1\right)\left(\frac{r_g}{L}\right)^2 x_4^2 \cos^2 x_3 \tag{5}$$

$$\dot{x}_3 = x_4 \tag{6}$$

$$\dot{x}_4 = -\frac{x_4}{\tau} + \frac{K}{\tau}u \tag{7}$$

where $u = V$ is defined as the control input to the system.

The relevant model parameters are given below:

r_g	$0.0254\,\mathrm{m}$
L	$0.4255\,\mathrm{m}$
g	$9.81\,\mathrm{m/s^2}$
K	$1.5\,\mathrm{rad/sV}$
τ	$0.025\,\mathrm{s}$

III. Observer and Controller Design

A. Observer Design - Extended Kalman Filter (EKF)

As our observer we implemented the Extended Kalman Filter. The EKF model[6] is as follows

$$x_{k+1} = f(x_k, u_k, t) + w_k,$$
$$z_{k+1} = h(x_k, t) + v_k,$$

where x_k is the state, f is a nonlinear function, u_k is the control input, w_k is a zero-mean multivariate gaussian with a covariance (Q_k) we select, z_k is the measurements we are provided, h is a nonlinear function of the states which in our case is simply results in the first and third variables of the state vector, and finally v_k is another zero-mean multivariate gaussian with a covariance (R_k) we select. A complication we run into here is that the EKF in this form is a discrete-time algorithm, while our model for the system is a continuous-time model. To rectify this we use Euler's method to discretize f as follows (note that f_{old} refers to the f from the original model of the system):

$$f := x_{k+1} = x_k + f_{old}(x_k, u_k)\Delta t.$$

The EKF algorithm has two stages, the first being the predict stage and the second being the update stage. During the prediction stage, the state is estimated using the previous control input and the previous state estimate. Then during the update stage, we use a number of calculations to predict an updated state estimate using the current measurements that we receive from the system. Below, we provide the equations for each stage:

Note that $\hat{x}_{n|m}$ represents the estimate of x at time n given observations up to and including time $m \leq n$.

Predict

$$\text{Predicted state estimate} \quad \hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1})$$
$$\text{Predicted covariance estimate} \quad P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{k-1}$$

Update

$$\text{Innovation or measurement residual} \quad \tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$$
$$\text{Innovation (residual) covariance} \quad S_k = H_k P_{k|k-1} H_k^T + R_k$$
$$\text{Near-optimal Kalman gain} \quad K_k = P_{k|k-1} H_k^T S_k^{-1}$$
$$\text{Updated state estimate} \quad \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$
$$\text{Updated covariance estimate} \quad P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

where the state transition and observation matrices are defined as the following Jacobians:

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_k} \quad \text{and} \quad H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}}.$$

B. Control Method 1 (Feedback Linearization with LQR)

We follow the solution in [8] and implement an approximate feedback linearization based controller. The idea behind feedback linearization is to utilize a nonlinear change of coordinates such that we arrive at a linear system between the control and output signals.

The standard feedback linearization design is as follows: starting with the system output $y = h(x)$, repeatedly take the time derivative until the control signal u appears. Define the number of differentiation steps, r, to be the system's relative degree. We have

$$\begin{aligned}
\dot{y} &= L_f h(x), \\
\ddot{y} &= L_f^2 h(x), \\
&\vdots \\
y^{(n)} &= L_f^n h(x) + L_g L_f^{n-1} h(x) u.
\end{aligned}$$

From this, we perform the nonlinear change of variables change of variables transformation:

$$\begin{aligned}
\zeta_1 &= y = h(x), \\
\zeta_2 &= \dot{y} = L_f h(x), \\
&\vdots
\end{aligned}$$

where the choice of $u = \frac{-L_f^n h(x) + v}{L_g L_f^{n-1} h(x)}$ results in the linear system

$$\begin{aligned}
\dot{\zeta}_1 &= \zeta_2, \\
\dot{\zeta}_2 &= \zeta_3, \\
&\vdots \\
\dot{\zeta}_n &= v.
\end{aligned}$$

Finally, we use LQR to optimally place the poles of the linear system through the state-feedback control $v = -Kz$. However, in the case of the ball and beam dynamics, the relative degree is not well-defined. After following the procedure

for feedback linearization for the ball and beam dynamics, $L_g L_f^{n-1} h(x)$ is zero when the motor angular velocity or ball position are zero, making feedback linearization inapplicable. However, in the case of the ball and beam, the dynamics can be approximated by removing terms that include the input since these terms follow conditions explained in [8]. With the input terms removed the output can be further differentiated, allowing for a well-defined relative degree of 4.

C. Control Method 2 (Time-Varying LQR)

The feedback linearization approach is nice because it ensures the system controlled by the LQR is in fact linear, so all the mathematical guarantees that the LQR gives are accurate. However, the linearized dynamics are different from the true dynamics, so our notion of optimality is incorrect. To correct this, we implemented an LQR directly on the nonlinear dynamics using a linearization about the target state. As we are only given the desired output and two derivatives, not the full state, we had to solve for the linearization point.

$$\begin{bmatrix} \dot{x}_d \\ \ddot{x}_d \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} f(x_d, \dot{x}_d, \underbrace{\theta, \dot{\theta}, u}_{\text{unknown}}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{8}$$

This seems underconstrained because there are three variables and two equations, but this is not the case; u does not affect $\dot{x}$ or $\ddot{x}$. We therefore leave it set to zero to minimize energy cost. Once we had solved the above equation for solutions $\theta^\star$, and $\dot{\theta}^\star$, we could simply linearize the system about the computed target state, then use that linearization to construct an LQR. Since the controller's score is based on a quadratic cost with given weights, we simply set our weight matrices to match. We used the following state and control costs.

$$Q = \operatorname{diag}\left(\begin{bmatrix} 1800 & 0 & 0 & 0 \end{bmatrix}\right) \tag{9}$$

$$R = \begin{bmatrix} 5 \end{bmatrix} \tag{10}$$

The control was then given by

$$u = -R^{-1} B^\top P (x - \bar{x}) \tag{11}$$

Where $\bar{x} = \begin{bmatrix} x_d & \dot{x}_d & \theta^\star & \dot{\theta}^\star \end{bmatrix}^\top$ and P is the solution to the continuous-time algebraic Riccati equation:

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0 \tag{12}$$

While the nonlinearity means that the control is not actually cost-minimizing, it should be much better than feedback linearization, which does not attempt to minimize our actual cost.

D. Safety Enforcement

We found that for all tested trajectories, the safety constraints were not violated. Therefore, we find it sufficient to have a properly functioning controller and make the assumption that the reference trajectory is feasible under the safety constraints. An additional safety enforcement policy like a control barrier function would simply add computation without changing the controller's score. Since our goal is to minimize the score, we have excluded any explicit safety enforcement policies, instead relying on the controller to remain safe.

We did have an exceedingly simple proportional gain to correct the system if the motor angle θ ever exceeded $\pm\frac{3\pi}{8}$, but this was primarily for the actual safety of the test setup, and ensuring that even if the controller was significantly off during testing, the hardware would not be harmed.

IV. Simulation Results

In the ball-and-beam simulation, we evaluate the performance of our EKF observer in two configurations: combined with our FL + LQR controller and with our LQR controller. We test both configurations on two reference tracking tasks: a sine wave and a square wave.

Let A be the amplitude and ω the angular frequency. The reference sine wave is:

$$p_{\text{ref}} = A \cdot \sin(\omega t)$$

And the reference square wave is:

$$p_{\text{ref}} = A \cdot \operatorname{sign}(\sin(\omega t))$$

For both reference signals, we set $A = 0.04$ and $\omega = \frac{2\pi}{10}$ for our simulation experiments and run the evaluation for a total of 100 seconds. Parameters for the EKF obsever, FBL + LQR controller, LQR controller are listed in Table II

The simulation results and plots of the EKF with FL + LQR and the EKF with LQR are presented in Table I and Figure 1.

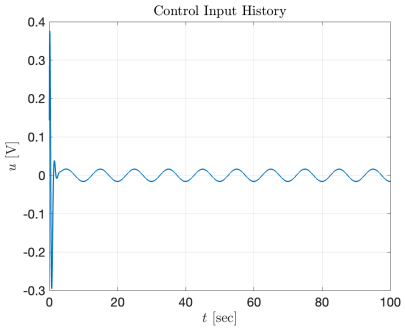
TABLE I: Simulation Performance

Reference	Controller	Tracking Cost	Energy Cost	Total Score
Sine Wave	LQR	0.53	0.01	0.54
	FL + LQR	0.00	0.00	0.01
Square Wave	LQR	3.48	0.03	3.51
	FL + LQR	2.03	0.29	2.32

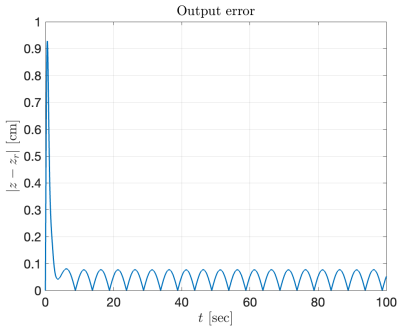
V. Hardware Experiment Results

For the hardware experiment, we evaluate the performance of EKF with FL + LQR and EKF with LQR. Both configurations are tested on an unknown reference trajectory, and the evaluation runs for a total of 20 seconds. The parameters for the EKF observer, FL + LQR controller, and LQR controller are provided in Table II.

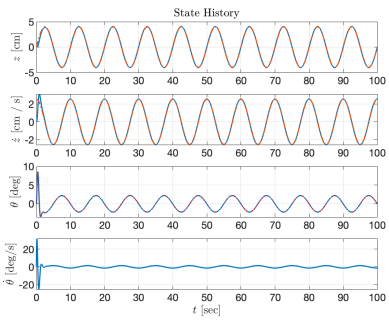
The hardware results, along with plots comparing the EKF with FL + LQR and the EKF with LQR, are presented in Table III and Figure 2.



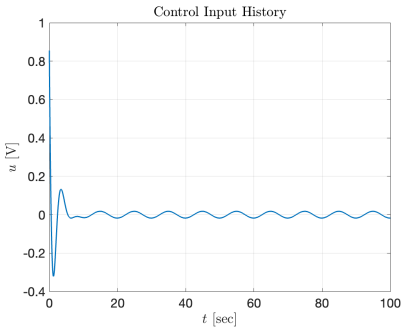
(a) FL+LQR (Sine): Control Input



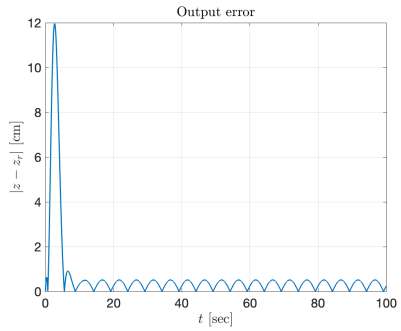
(b) FL+LQR (Sine): Tracking Error



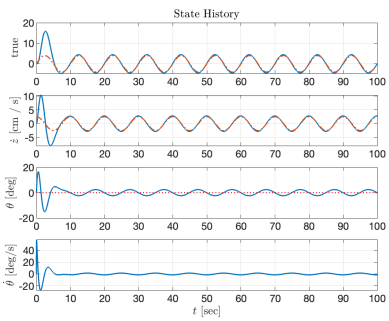
(c) FL+LQR (Sine): State vs Ref



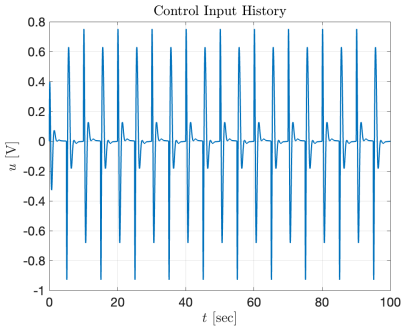
(d) LQR (Sine): Control Input



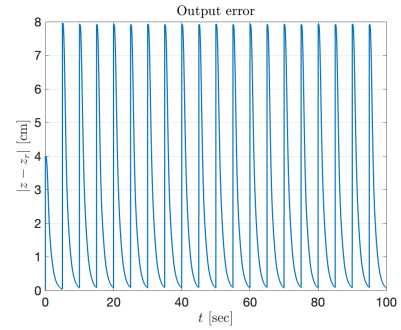
(e) LQR (Sine): Tracking Error



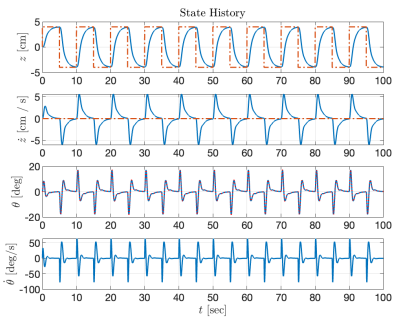
(f) LQR (Sine): State vs Ref



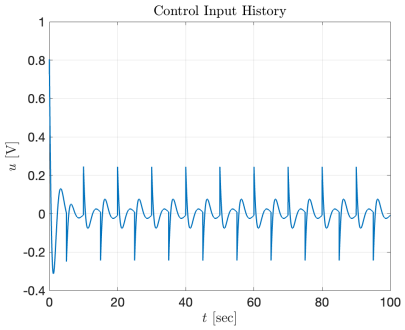
(g) FL+LQR (Square): Control Input



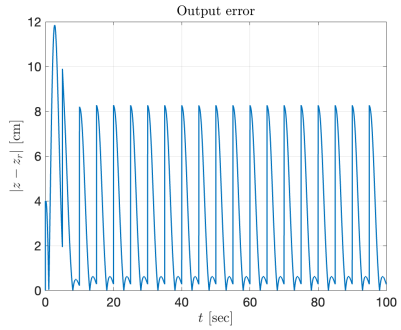
(h) FL+LQR (Square): Tracking Error



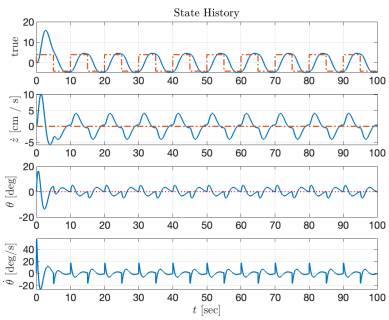
(i) FL+LQR (Square): State vs Ref



(j) LQR (Square): Control Input



(k) LQR (Square): Tracking Error

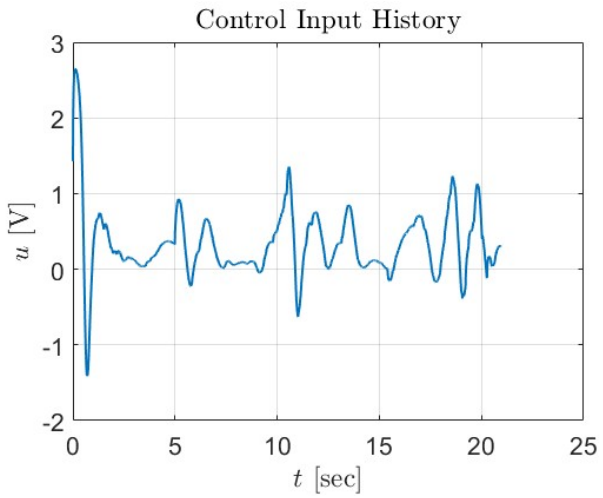


(l) LQR (Square): State vs Ref

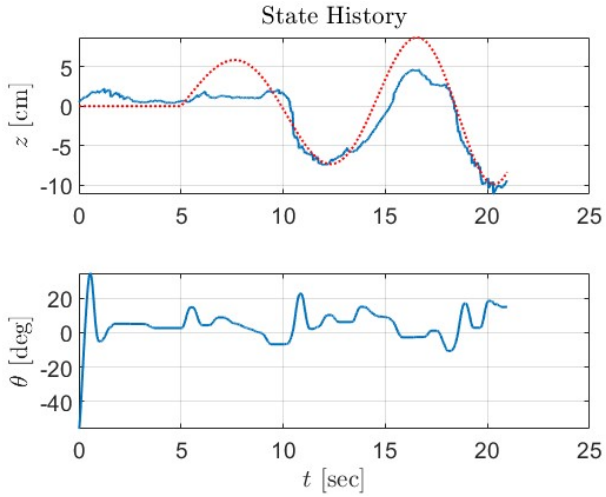
Fig. 1: Simulation results of tracking sine and square waves using FL+LQR and LQR controllers with EKF observer.

VI. Conclusion

In simulation, we found that the FL + LQR controller outperformed the standard LQR controller, achieving both lower tracking cost and lower energy cost for reference sine and square wave trajectories. However, we suspect a minor bug in our LQR implementation, as the simulation plots show a large initial tracking error spike at the start of each rollout. In hardware experiments, FL + LQR also performed better overall. While this may partly be due to the suspected issue with the LQR implementation, we believe FL + LQR is still better for the real system. Although FL + LQR relies on an



(a) Control Input History



(b) State vs Reference Trajectory

Fig. 2: Real hardware results for FL + LQR controller with EKF.

TABLE II: EKF, FBL + LQR, and LQR parameters for both simulation and real experiments.

Description	**Value**
EKF: Initial state covariance P	$\begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$
EKF: Process noise covariance Q	$\begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$
EKF: Measurement noise covariance R	$\begin{bmatrix} 0.003 & 0 \\ 0 & 0.003 \end{bmatrix}$
FBL + LQR: State cost matrix Q	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
FBL + LQR: Control cost scalar R	0.0001
LQR: State cost matrix Q	$\begin{bmatrix} 1800 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
LQR: Control cost matrix R	5

TABLE III: Hardware Performance

Reference	**Controller**	**Tracking Cost**	**Energy Cost**	**Total Score**
Evaluation Trajectory	LQR	5.29	1.53	6.82
	FL + LQR	1.05	1.82	**2.87**

approximate model as described in [8], this approximation is more accurate than the Jacobian Linearization in the LQR approach.

One major challenge we encountered was model/plant mismatch, where controller performance was significantly better in simulation than on real hardware. This discrepancy was unexpected and introduced several complications, particularly in implementing an effective state observer. Initially, we developed two observers—an extended Luenberger observer and a custom observer from Homework 6—but neither performed well in conjunction with the FL + LQR controller. It is unclear whether this was due to implementation errors or difficulties in tuning the observer parameters. Only after switching to an Extended Kalman Filter (EKF) did the FL + LQR controller begin to exhibit reasonable performance on hardware. This led us to conclude that having a well-tuned, accurate observer is essential for controller effectiveness. Even with the EKF, we faced challenges in tuning, such as needing to use significantly larger gains for FL + LQR on real hardware compared to simulation, which resulted in higher control effort and thus increased energy cost. Given our limited experience with using EKFs and the fact that we only had one lab session to tune it, we believe that with more time and proper tuning, even better results could be achieved.

Another significant issue stemming from the model/plant mismatch was the effect of gravity on the physical ball-and-beam system. Specifically, the gravitational force caused the ball and beam to exert a downward torque on the servo motor, which was not accounted for in our system dynamics. This led to a consistent offset or bias, where the ball tended to favor the side of the beam that naturally tilted downward due to gravity. Due to time constraints, we were unable to address this issue thoroughly, so we applied an additional voltage input to partially compensate for the sagging. With more time, we would consider explicitly incorporating gravitational effects into the system dynamics or adding an integral term to the

controller to mitigate the bias.

If we were to repeat this lab, there are several things we would do differently. A major issue was that our code did not initially compile in Simulink, and we were unaware that Simulink compatibility needed to be tested during the simulation phase of the project. As a result, we lost two lab periods modifying the code to work with Simulink. Additionally, we didn't make efficient use of our time outside the lab; too much of our lab sessions were spent coding and debugging rather than testing and tuning. Another organizational issue was placing all of our observers and controllers into a single file, which made the codebase difficult to manage. Using separate branches or files for different combinations would have resulted in much cleaner and more manageable code.

Overall, although we ran into many issues this lab was a valuable experience on getting hands-on experience developing controllers for real systems.

REFERENCES

[1] S. Vaidyanathan and A. T. Azar, "Chapter 1 - an introduction to backstepping control," in *Backstepping Control of Nonlinear Dynamical Systems*, ser. Advances in Nonlinear Dynamics and Chaos (ANDC), S. Vaidyanathan and A. T. Azar, Eds. Academic Press, 2021, pp. 1–32. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128175828000088

[2] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109889900022

[3] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064–5078, 2024.

[4] A. J. Krener, *Feedback Linearization*. New York, NY: Springer New York, 1999, pp. 66–98. [Online]. Available: https://doi.org/10.1007/978-1-4612-1416-8_3

[5] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[6] W. contributors, "Extended kalman filter — Wikipedia, the free encyclopedia," 2024, [Online; accessed 3-April-2024]. [Online]. Available: https://en.wikipedia.org/wiki/Extended_Kalman_filter

[7] M. Zeitz, "The extended luenberger observer for nonlinear systems," *Systems Control Letters*, vol. 9, no. 2, pp. 149–156, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167691187900211

[8] J. Hauser, S. S. Sastry, and P. Kokotovic, "Nonlinear control via approximate input-output linearization: the ball and beam example," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 392–398, March 1992.