Control of a Nonlinear System: Ball and Beam

Leo Huang*
Electrical Engineering
University of California, Berkeley
Berkeley, USA
klhftco@berkeley.edu

Dylan Lee*
Mechanical Engineering
University of California, Berkeley
Berkeley, USA
dlee1009@berkeley.edu

Jinsheng Li*
Physics
University of California, Berkeley
Berkeley, USA
ljs233233@berkeley.edu

Miranda Xie*
Electrical Engineering
University of California, Berkeley
Berkeley, USA
mirandax@berkeley.edu

***Abstract*—In this report, we address the control of the classic nonlinear ball-and-beam dynamical system by developing two control schemes: a PID-Wrapped LQR controller and an approximate Feedback Linearization controller, each wrapped inside a control barrier function to guarantee safety. The controllers are given state estimations using an Extended Kalman Filter. The controllers are tested in simulation before conducting hardware experiments. The results of both the simulation and hardware show some differences in behavior between the controllers, with the PID-LQR controller performing somewhat better in both cases. Furthermore, comparing the simulation and hardware results highlighted modeling inaccuracies and how important they can be in designing controllers.**

Our full code is uploaded to this Github repository: https://github.com/ljs-233233/EE222-Ball-and-Beam.

I. Introduction

We developed a controller for a Ball and Beam dynamical system. This setup consists of a track along which a metal ball can roll freely. One side of the beam is attached to a lever arm coupled to the load gear of a rotary servo unit. By controlling the servo's position, the beam angle is adjusted to give the ball a desired acceleration.

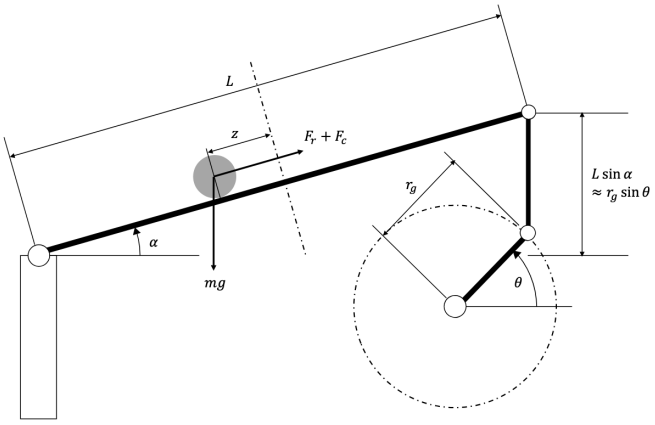

Fig. 1: The Ball and Beam Setup.

*Listed in alphabetical order.

The controller was evaluated based on three criteria: how accurately it tracks a given reference trajectory, its energy consumption, and any violations of physical or safety constraints. Each category incurs a penalty, and the overarching goal is to minimize the total score.

In Section II, we discuss the mathematical model of the system, in Section III, we present the design of our observer and controllers, in Section IV, we present the simulation results, and in Section V, we present the results on hardware.

II. System Modeling

An idealized version of the ball and beam dynamics can be derived using geometry and Newton's laws. By ignoring friction and assuming that α remains small, the resulting dynamics in state space are given by the nonlinear Equation (1), $\dot{\mathbf{x}} = F(\mathbf{x}, u)$ where $\mathbf{x} \in \mathbb{R}^4$ and $u \in \mathbb{R}$. The numerical values of system parameters are given in Table I.

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{5g}{7}\frac{r_g}{L}\sin x_3 - \frac{5}{7}\left(\frac{L}{2} - x_1\right)\left(\frac{r_g}{L}\right)^2 x_4^2\cos^2 x_3 \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= -\frac{x_4}{\tau} + \frac{K}{\tau}u
\end{aligned} \tag{1}$$

We can Euler-discretize the system as follows:

$$\begin{aligned}
x_1[k+1] &= x_1[k] + x_2[k]dt \\
x_2[k+1] &= x_2[k] + \left[\frac{5g}{7}\frac{r_g}{L}\sin x_3[k]\right. \\
&\quad \left. -\frac{5}{7}\left(\frac{L}{2} - x_1[k]\right)\left(\frac{r_g}{L}\right)^2 x_4[k]^2\cos^2 x_3[k]\right]dt \\
x_3[k+1] &= x_3[k] + x_4[k]dt \\
x_4[k+1] &= x_4[k] - \left(\frac{x_4[k]}{\tau} + \frac{K}{\tau}u[k]\right)dt
\end{aligned} \tag{2}$$

Where dt is the time step. We define this system as $\mathbf{x}[k+1] = \tilde{F}(\mathbf{x}[k], u)$.

TABLE I: Ball and Beam System Parameters

Parameter	Variable(s)	Value
Ball Position	x_1, z	
Ball Velocity	x_2, $\dot{z}$	
Beam Angle	x_3, θ	
Beam Angular Velocity	x_4, $\dot{\theta}$	
Motor Voltage	u, V	
Gravitational Acceleration	g	9.81 m/s^2
Servo Arm Length	r_g	0.0254 m
Beam Length	L	0.4255 m
Motor Constant	K	1.5 rad/sV
Delay Constant	τ	0.025 s

We aim to design a controller that computes the servo motor voltage V to drive the ball along a desired time-varying trajectory over a period of $T = 20$ s. The primary objectives are to minimize the tracking error and the motor's energy consumption, while ensuring safe operation. In particular, the ball must remain within the physical limits of the beam. The metrics used to quantify system performance are the Tracking Error:

$$J_{\text{tracking}} = \frac{1}{T}\int_0^T (x_1(t) - z_{\text{ref}}(t))^2 \, dt \tag{3}$$

Energy Consumption:

$$J_{\text{energy}} = \frac{1}{T}\int_0^T u^2(t) \, dt \tag{4}$$

and Safety Constraint Violations:

$$J_{\text{safety}} = \begin{cases} 1, & \mathbf{x}(t) \text{ is unsafe} \\ 0, & \text{else} \end{cases} \tag{5}$$

Where $\mathbf{x}(t)$ is unsafe when:

$$\begin{cases} |z(t)| \geq 0.19 \text{ m}, & \text{collision imminent} \\ |\theta(t)| \geq 60^\circ, & \text{angle outside operation range} \end{cases} \tag{6}$$

The goal is to minimize the total score achieved by the controller:

$$J_{\text{total}} = w_t J_{\text{tracking}} + w_e J_{\text{energy}} + w_s J_{\text{safety}} \tag{7}$$

Where w_t, w_e, and w_s are the weight coefficients assigned to each cost.

III. Observer and Controller Design

A. Observer: Extended Kalman Filter

The observer we designed is an Extended Kalman Filter (EKF), selected for two primary reasons. First, we anticipated the sensors having at least some degree of noise, and the EKF, by construction, aims to filter out measurement noise. Second, we bear in mind that the system dynamics in Section II will almost certainly be inaccurate, as friction, delay, and the potential for large angles was ignored. The EKF also permits some degree of modeling uncertainty, weighing the assumed model and measurements according to its tuning.

We begin by simply assuming additive process noise in the model:

$$\begin{aligned} \tilde{x} &= x + v \\ \Rightarrow \tilde{x}[k+1] &= F(\tilde{x}[k], u[k]) + v[k] \end{aligned} \tag{8}$$

where $v = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}^\top$ is to be tuned. We also define our (noisy) measurements:

$$\tilde{h}(\tilde{x}, w) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tag{9}$$

where the measurement noise w is to be measured via calibration or tuned.

We can then use the standard EKF design as discussed in [1] to obtain the estimated state $\hat{x}[k]$ and variance $P[k]$ at any time step k. The initialization of the state is $\hat{x}[0] = \begin{bmatrix} x_1(0) & 0 & x_3(0) & 0 \end{bmatrix}^\top$ since we have access to the initial sensor measurements. The initialization of the variance is arbitrary, and we use $P(0) = 0.1\, I_{4x4}$. At each time step, we perform first a prediction/process update, obtaining

$$\begin{aligned} \hat{x}_p[k] &= F\left(\hat{x}[k-1], u[k-1]\right) \\ P_p[k] &= AP[k-1]A^\top + J\Sigma_{vv}J^\top \end{aligned} \tag{10}$$

where the subscript p indicates a predicted quantity, $A = \frac{\partial F}{\partial \tilde{x}}\big|_{\tilde{x}=\hat{x}[k-1], u=u[k-1], v=0}$, $J = \frac{\partial \tilde{h}}{\partial \tilde{x}}\big|_{\tilde{x}=\hat{x}[k-1], w=0}$, and Σ_{vv} is the covariance matrix of the process noise v. Note all these matrices may be time varying (indeed A is), but those which are not (in this case L and Σ_{vv} by design) can be pre-computed to improve runtime. If we then take measurements from the sensors, we can perform a measurement update, obtaining

$$\begin{aligned} K &= P_p[k]H^\top \left(HP_p[k]H^\top + M\Sigma_{ww}M^\top\right)^{-1} \\ \hat{x}[k] &= \hat{x}_p[k] + K\left(y[k] - \tilde{h}(\hat{x}_p, 0)\right) \\ P[k] &= (I - KH)\, P_p[k]\, (I - KH)^\top \\ &\qquad + KM\Sigma_{ww}M^\top K^\top \end{aligned} \tag{11}$$

where $H = \frac{\partial h}{\partial \tilde{x}}\big|_{\tilde{x}=\hat{x}_p[k], w=0}$, $M = \frac{\partial \tilde{h}}{\partial w}\big|_{\tilde{x}=\hat{x}_p[k], w=0}$, and Σ_{ww} is the covariance matrix of the measurement noise w. Again, these matrices may all be time varying. Note that the noise covariance matrices Σ_{vv} and Σ_{ww} are tuning parameters for the observer. For a full list of the matrices used in the observer, see Appendix A.

B. Controller 1: PID-Wrapped LQR

The first controller we decided to implement was a PID-wrapped LQR controller. This decision was a result of some preliminary simulation testing with a pure PID controller and a pure LQR controller. We observed that the PID controller tracked the reference trajectory $r_t(t)$ considerably better than the LQR controller but used significantly more energy. This qualitatively makes sense as the PID controller simply does not penalize energy usage, while LQR balances a quadratic cost for tracking and energy. We therefore decided to combine the two controllers.

Specifically, we noticed that the x_3, x_4 subsystem is in fact linear, and that the energy cost we are evaluated with is quadratic. These facts lend the x_3, x_4 subsystem to be controlled well by an LQR controller, so we implemented an

LQR which determines its reference based on a cascaded PID wrapper. Therefore, the exact structure of the controller is

$$\begin{aligned} x_{2,d} &= \dot{r}_t - k_{p,1}e_1 - k_{i,1}\int_{t_0}^{t} e_1\,\mathrm{d}\tau - k_{d,1}\dot{e}_1 \\ x_{3,d} &= \frac{7L}{5gr_g}\left(\ddot{r}_t - k_{p,2}e_2 - k_{i,2}\int_{t_0}^{t} e_2\,\mathrm{d}\tau - k_{d,2}\dot{e}_2\right) \\ x_{4,d} &= -k_{p,3}e_3 - k_{i,3}\int_{t_0}^{t} e_3\,\mathrm{d}\tau - k_{d,3}\dot{e}_3 \\ u &= -G\begin{bmatrix} e_3 \\ e_4 \end{bmatrix} \end{aligned} \tag{12}$$

where $e_1 = x_1 - r_t$, $e_j = x_j - x_{j,d}$ for $j = \{2,3,4\}$, the gains $k_{\star,i}$ are to be tuned and G is the standard LQR gain obtained by solving the Discrete Algebraic Riccati Equation (DARE). Note that Simulink does not support MATLAB's built-in `lqr()` or `dlqr()` functions, so a fast DARE solver was explicitly implemented, with reference to [2].

C. Controller 2: Approximate Feedback Linearization

There are several reasons we chose to implement feedback linearization as one of our controllers. First, we note that the system is an affine system, which suggests that feedback linearization may be possible. Furthermore, we note that the reference trajectory is composed of either sinusoidal or square waves. The sinusoidal references are smooth and infinitely differentiable. Meanwhile, the square references are piecewise constant, and are differentiable except at a finite set of points; we therefore set all desired derivatives to zero at those points, making the derivatives smooth and infinitely differentiable. By taking up to the fourth-order derivatives of the reference trajectory, we may be able to, by selecting the output as

$$y = h(x) = x_1 \tag{13}$$

and have a relative degree of $r = n = 4$ without any zero dynamics. In reality, an analysis of the output derivatives shows that the relative degree is not well defined. However, following feedback linearization techniques in [3] produces an approximate system with a well-defined relative degree of 4.

We apply the standard feedback linearization change of variables and control law:

$$\begin{aligned} \xi_i &= y^{(i-1)} \\ u &= \frac{1}{L_g L_f^3 h}\left(-L_f^4 h - \sum_{i=1}^{4} k_i \xi_i\right) \end{aligned} \tag{14}$$

where $L_\star(\cdot)$ is the Lie derivative of $(\cdot)$ in the direction of $\star$. Hauser's work [3] presents two methods to approximate the system such that a well-defined relative degree is obtained. In both methods, we find $L_g L_f^3 h = \frac{5gr_g K \cos x_3}{7L\tau}$. $L_f^4 h$ using the "worse" approximation described by [3] is equal to $-\frac{5gr_g}{7L\tau}x_4 \cos x_3 + x_4^2 \sin x_3$, while using the "better" approximation yields an expression with considerably more terms. We select the gains k_i such that the matrix

$$A_k = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_1 & -k_2 & -k_3 & -k_4 \end{bmatrix} \tag{15}$$

is Hurwitz, and the dynamics via approximation is controllable, allowing the gains k_i to be arbitrarily tuned.

D. Safety: Control Barrier Function

To guarantee that the ball remains safely on the beam, we augment both controllers with a safety filter based on a Control Barrier Function. The approach formally ensures that the system state remains within a predefined safe set. In our design following [4], we define a safe set for the ball's position using the barrier function

$$B(x) = p_{\text{ball}}^2 - (0.175)^2,$$

where p_{ball} is the ball's position. This choice reflects our observation that the ball trajectories remain within 15 cm, and thus we enforce a conservative safety margin by setting the safe limit to 17.5 cm. The ball is safe as long as $B(x) \leq 0$.

The CBF condition is formulated to ensure that the safe set is forward invariant. Specifically, we require that

$$L_f^2 B(x) + L_g^2 B(x)\, u + 2\lambda\, \dot{B}(x) + \lambda^2\, B(x) \leq 0,$$

where $\lambda > 0$ (set as $\lambda_{\text{cbf}} = 10$ in our implementation) determines the aggressiveness of the safety enforcement.

Our controller first computes a nominal control input. If the nominal control input does not satisfy the CBF condition, we project it onto the set of safe controls by solving the scalar inequality [4]

$$L_g^2 B(x)\, u + \left(L_f^2 B(x) + 2\lambda\, \dot{B}(x) + \lambda^2\, B(x)\right) \leq 0.$$

This safety filter guarantees that, regardless of the nominal command, the system will not drive the ball into an unsafe region (i.e., $B(x) > 0$). We test the safety of our controllers in simulation by setting the reference trajectory to be consistently unsafe ($z = 0.19$ cm). The controller drives the ball very close to the desired position but does not ever violate the safety constraints. For an illustration of the controller safety, see the media in the repository linked in the abstract.

IV. Simulation Results

A list of the parameter values used in simulation testing is given in Table II.

The simulation results with each of the controllers is shown in Table III. Note that the performance is based on a sinusoidal reference with amplitude 7.5 cm and period 8 seconds, or a square reference with amplitude 5 cm and period 8 seconds. The resulting trajectories are shown in Figure 2. In both simulation and hardware performance, we did not experience any safety constraint violations, hence omitting the "Safety Cost" category in all scoring tables.

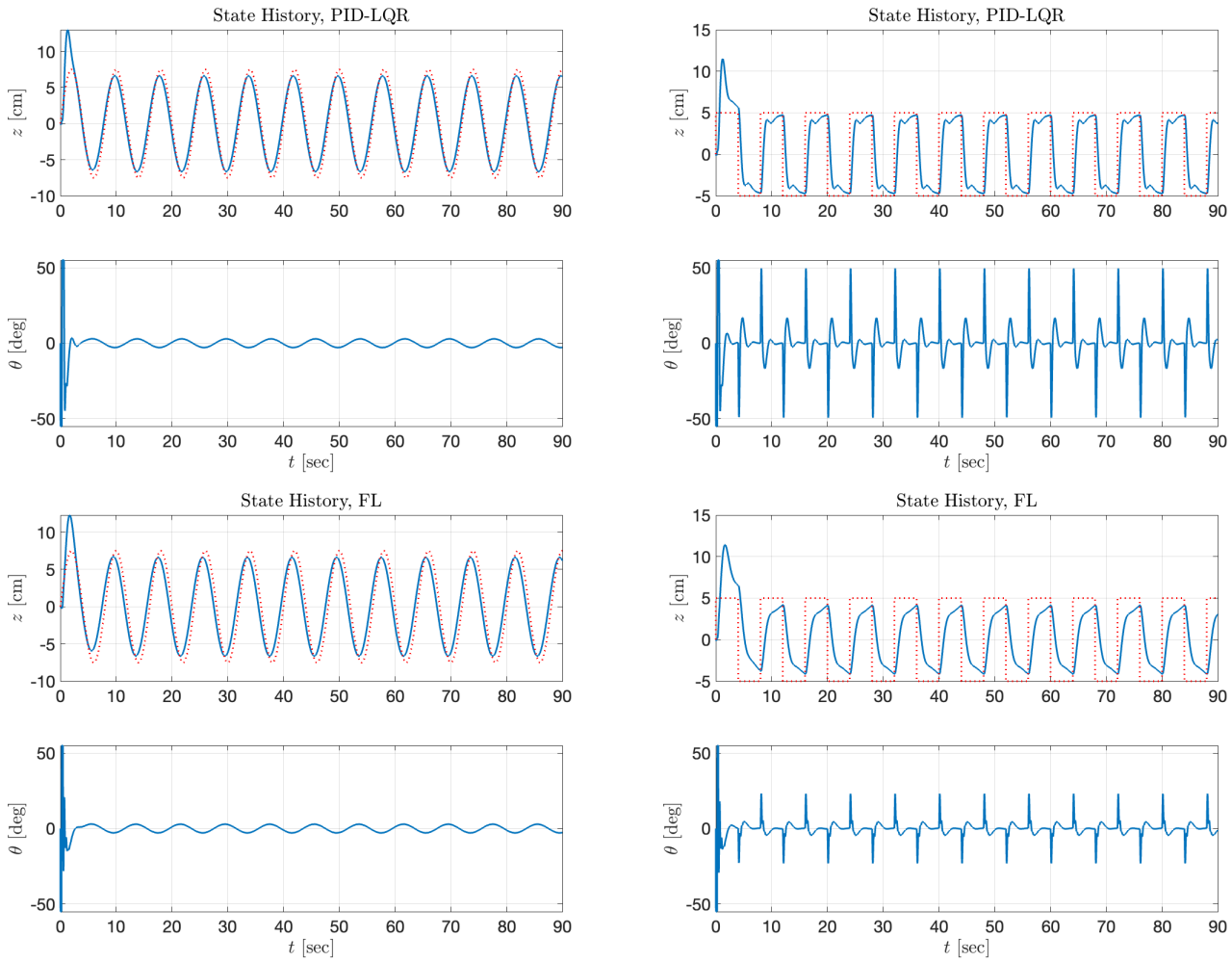

Fig. 2: Simulated trajectories using either PID-LQR (top) or FL (bottom) with a sinusoidal (left) or square (right) reference.

TABLE II: Simulation Parameters

Param.	Description	Value
$\hat{x}[0]$	EKF initial state	$\begin{bmatrix}0 & 0 & -55\pi/180 & 0\end{bmatrix}^\top$
$P[0]$	EKF initial covariance	$0.1 I_{4\times4}$
Σ_{vv}	EKF process noise	$0.01 I_{4\times4}$
Σ_{ww}	EKF measurement noise	$0.01 I_{4\times4}$
$k_{p,1}$	(PID-LQR) pos. (x_1) P gain	1.2
$k_{p,2}$	(PID-LQR) vel. (x_2) P gain	3.0
$k_{p,3}$	(PID-LQR) angle (x_3) P gain	1.0
...	(PID-LQR) other gains	0
Q	(PID-LQR) Q matrix	$\mathrm{diag}\,(1000, 100)$
R	(PID-LQR) R matrix	1
p_i	(FL) pole placements	$\{-1.12, -2.7, -6.0, -32.5\}$
λ	(CBF) "aggressiveness"	10

As can be seen in both the trajectories and Table III, the PID-LQR controller seems to perform better overall. Even though it uses proportionally more energy, the significant improvement in tracking cost is enough to render PID-LQR superior to FL in simulation. Note additionally that the "worse" and "better" feedback linearization approximations presented in [3] do not appear to yield noticeably different costs.

TABLE III: Simulation Performance

Reference	Controller	Tracking Cost	Energy Cost	Total Score
Sine Wave	PID-LQR	0.30	0.39	**0.69**
	FL	0.73	0.63	1.35
Square Wave	PID-LQR	1.91	0.80	**2.71**
	FL	3.08	0.72	3.80

V. Hardware Experiment Results

We then attempted to implement our observer and controllers on hardware. As expected, we discovered that various parameters required re-tuning. In summary,

1) We adjusted the measurement covariance Σ_{ww} based on our observations of the actual measurements. In fact, we found that measurements tended to be noisier closer to the servo end of the beam, so we used a sigmoid for the

magnitude of the noise dependent on the ball position. Specifically, we used:

$$\begin{aligned}&\Sigma_{ww}(\hat{x}_1[k-1])\\&\quad= 0.01 + 0.04\left(1+\exp\left(30(\hat{x}_1[k-1]+0.1)\right)\right)^{-1}\\&\quad:= w(\hat{x}_1[k-1])\end{aligned} \tag{16}$$

2) We observed that when the ball is stationary, the ball had to overcome very large static friction to start rolling; when the ball is in motion, rolling friction causes the ball to have less acceleration than predicted by the ideal model. Since the FL controller is model-dependent, we redefined the dynamics to incorporate static and rolling friction. Specifically, we modified the acceleration term in 1. When the ball is stationary, we have:

$$\begin{aligned}\dot{x}_{2,\text{ideal}} &= \frac{5g}{7}\frac{r_g}{L}\sin x_3 - \frac{5}{7}\left(\frac{L}{2}-x_1\right)\left(\frac{r_g}{L}\right)^2 x_4^2\cos^2 x_3\\ \dot{x}_2 &= \operatorname{sgn}(\dot{x}_{2,\text{ideal}})\cdot R\Big(|\dot{x}_{2,\text{ideal}}| - \underbrace{\mu_s\frac{5g}{7}\frac{r_g}{L}\cos x_3}_{\text{Static Friction}}\Big)\end{aligned} \tag{17}$$

When the ball is rolling, the acceleration is:

$$\dot{x}_2 = \dot{x}_{2,\text{ideal}} - \underbrace{\operatorname{sgn}(x_2)\mu_r\frac{5g}{7}\frac{r_g}{L}\cos x_3}_{\text{Rolling Friction}} \tag{18}$$

The coefficients of static and rolling friction, μ_s and μ_r, were tuned during hardware testing. Note that when the ball is stationary, the magnitude of acceleration is lower-bounded by zero by means of a unit ramp function $R(x) = \max\{0, x\}$, such that friction should never move the ball in the opposite direction of its natural acceleration. We recalculated the relevant Lie derivatives after this modification and ensured the relative degree was well defined (it is so long as $\mu_s, \mu_r < \frac{1}{\sqrt{2}}$).

3) We were given the freedom to arbitrarily select an initial position, but the beam angle had to be initialized at $-55°$, causing the ball to initially accelerate in the negative direction. Thus, we released the ball around 0.05 m uphill to borrow some gravitational potential and lower the initial input energy while quickly converging to the reference trajectory around the center of the beam.
4) We modified the "aggressiveness" of controllers (i.e. pole placements, gain magnitudes, etc.).
5) After noticing a large proportion of our incurred cost consisted of energy usage, we saturated the amplitude of allowable control input by V_{sat}. Since the beam angle is initialized at $-55°$, the controller tends to apply an unnecessarily large control input at the beginning. We therefore defined V_{sat} to increase linearly up to some maximum value tuned for each controller individually during hardware testing.

The saturation ramp-ups we adopted are:

$$\begin{aligned}V_{\text{sat,PID-LQR}} &= \min\{0.5+10t, 3\}\\ V_{\text{sat,FL}} &= \min\{0.5+2.5t, 3\}\end{aligned} \tag{19}$$

Where t is the time since the beginning of motion.

A list of the parameter values used in the hardware experiment is given in Table IV.

TABLE IV: Hardware Parameters

Param.	Description	Value
$\hat{x}[0]$	EKF initial state	$\begin{bmatrix}0 & 0 & -55° & 0\end{bmatrix}^\top$
$P[0]$	EKF initial covariance	$0.1\ I_{4\times4}$
Σ_{vv}	EKF process noise	$0.01\ I_{4\times4}$
Σ_{ww}	EKF measurement noise	$w\left(\hat{x}_1[k-1]\right)$ in (16)
$k_{p,1}$	(PID-LQR) pos. (x_1) P gain	1.0
$k_{i,1}$	(PID-LQR) pos. (x_1) I gain	0.2
$k_{d,1}$	(PID-LQR) pos. (x_1) D gain	0.1
$k_{p,2}$	(PID-LQR) vel. (x_2) P gain	3.0
$k_{i,2}$	(PID-LQR) vel. (x_2) I gain	1.0
$k_{p,3}$	(PID-LQR) angle (x_3) P gain	1.0
...	(PID-LQR) other gains	0
Q	(PID-LQR) Q matrix	$\operatorname{diag}(1000, 100)$
R	(PID-LQR) R matrix	81
p_i	(FL) pole placements	$\{-1.5, -2.7, -6.0, -32.5\}$
μ_s	(FL) stationary friction coefficient	0.15
μ_r	(FL) rolling friction coefficient	0.1
V_{sat}	(PID-LQR) control saturation	0.5 - 3 V in (19)
V_{sat}	(FL) control saturation	0.5 - 3 V in (19)
λ	(CBF) "aggressiveness"	10

The resulting performance using each of the two controllers is presented in Table V. As can be seen in the table, the PID-wrapped LQR controller performs noticeably better than the approximate FL controller, achieving a lower cost in both tracking and energy costs.

TABLE V: Hardware Performance

Reference	Controller	Tracking Cost	Energy Cost	Total Score
Sine Wave	PID-LQR	0.81	1.15	**1.96**
	FL	1.44	1.22	2.65
Square Wave	PID-LQR	4.05	1.95	**6.00**
	FL	5.25	1.84	7.08
Evaluation Trajectory	PID-LQR	0.24	1.08	**1.36**
	FL	0.65	1.51	2.16

For the PID-LQR controller, we present plots of the states and controls over the evaluation trajectory in Figure 3.

VI. Conclusion

In general, we observe that the trends seen in simulation for the controllers we designed are qualitatively similar to those seen on hardware. Specifically, we see that the PID-LQR controller generally tracks better than the FL controller for a given reference trajectory. Meanwhile, the energy costs are more comparable between the two controllers; it seems for sinusoidal references, the PID-LQR manages to use less energy than the FL controller, but for aggressively changing square waves, the PID-LQR controller expends more energy to achieve tracking. This is even despite visible "jerks" in the servo angle when using the FL controller on hardware (likely due to our switching logic to account for friction dynamics).

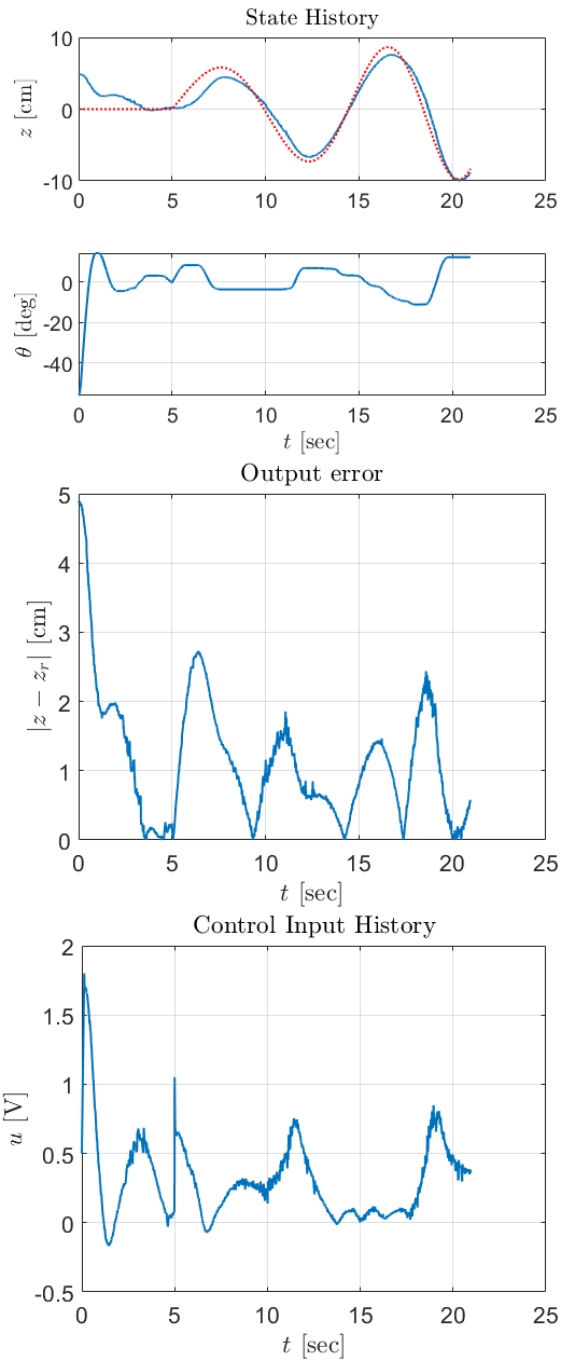

Fig. 3: States, absolute tracking error, and control inputs on hardware using PID-LQR controller on evaluation trajectory.

While the qualitative trends may be similar, there are notable model/plant mismatches. Firstly, we note that the actual costs (both tracking and energy) are significantly higher in hardware than in simulation. This is consistent with the fact that the model cannot capture all relevant dynamics of the actual plant (most notably friction or delays). In fact, the unmodified FL controller works quite well in simulation, but essentially fails to move the ball on hardware, and required a modification in the dynamical model to achieve the scores listed in Table V. Additionally, a non-negligible amount of tuning was required when switching from simulation to hardware, most notably R increasing from 1 to 81, as well as nonzero integral and differential gains for the PID controller.

More interestingly, the actual behavior of the controllers and the resulting trajectories differ somewhat for each controller. For example, in simulation, the PID-LQR controller seemed to achieve better convergence to stationary references (Fig. 2), but on hardware, the FL controller converges to a stationary reference significantly better than the PID-LQR controller. This further supports that models do not perfectly reflect the dynamics of the plant. We also see great potential in the friction dynamics model for designing a more accurate FL controller, especially if direct measurements of μ_s and μ_r were taken to better capture the true non-ideal behavior.

As for hardware testing, although we were not too surprised by the difference between real and simulation, there were a few lessons we learned. One key takeaway was to expect that two different hardware setups would perform non-identically and to ideally work on the same station. If we were to start this lab again, we would also be more systematic about sweeping parameters for all design choices to arrive at a better understanding of the actual performances of each controller.

APPENDIX

A. EKF Matrices

Matrices used in EKF observer as defined in Section III-A.

$$A = \begin{bmatrix} 1 & dt & 0 & 0 \\ \alpha\, dt & 1 & \beta\, dt & \gamma\, dt \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 - \frac{dt}{\tau} \end{bmatrix}, \qquad J = I_{4x4}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad M = I_{2x2}$$

with

$$\alpha = \frac{5}{7}\left(\frac{r_g}{L}\right)^2 x_4^2 \cos^2 x_3^2$$

$$\beta = \frac{5g}{7 r_g L}\cos x_3 + \frac{5}{7}\left(\frac{L}{2} - x_1\right)\left(\frac{r_g}{L}\right)^2 x_4^2 \sin(2x_3)$$

$$\gamma = -\frac{10}{7}\left(\frac{L}{2} - x_1\right)\left(\frac{r_g}{L}\right)^2 x_4 \cos^2 x_3$$

REFERENCES

[1] M. Mueller, "ME C231B lecture notes," University of California, Berkeley, 2024.

[2] E. K. W. Chu, H. Y. Fan, W. W. Lin, and C. S. Wang, "Structure-preserving algorithms for periodic discrete-time algebraic Ricatti equations," *International Journal of Control*, vol. 77, no. 8, pp. 767–788, 2003.

[3] J. Hauser, 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, 1992.

[4] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," *European Control Conference*, vol. 18, pp. 3420–3431, 2019.