Nonlinear Control of Ball and Beam System

***Abstract*—This work presents the design and implementation of a nonlinear state estimation and control strategy for the classical ball-and-beam system. We develop a continuous-time extended Kalman filter to reconstruct the full four-dimensional state from noisy ball-position and beam-angle measurements on the system hardware. The controllers explored in this work are (1) Feedback Linearization (FL) with a linear–quadratic regulator (LQR) and (2) discrete LQR (DLQR) with integral control to reduce the steady state error, both of which are synthesized on the linearized dynamics to guarantee asymptotic tracking of reference trajectories while penalizing control effort. Simulation and hardware experiments demonstrate tracking accuracy and robustness to model–plant mismatch and system noise. In both simulation and real-world experiments, DLQR is superior to feedback linearization and this gap is more apparent in real-world testing – which will be explored furthered in the results section. Code is found here, with branch main containing Phase I and branch master containing Phase II.**

I. Introduction

The ball-and-beam apparatus is a benchmark nonlinear control testbed, comprising a rigid beam actuated by a rotary servo and a free-rolling ball constrained to move along the beam. Controlling the ball to follow a prescribed trajectory presents significant challenges due to the underactuated, highly nonlinear, and inherently unstable nature of the system. In particular, the coupling between the beam angle and ball motion induces centrifugal effects and sinusoids in the equations of motion, while sensor limitations restrict feedback to only the ball position and beam angle. This project addresses these challenges through the synthesis of two estimator–controller architectures. The remainder of the paper is organized as follows: Section II derives the continuous-time nonlinear model and its state-space linearization; Section III details the observer and controller methods 1 and 2; Section IV summarizes the simulation results; Section V presents hardware validation; and Section VI concludes with discussion, challenges, and future work.

II. System Modeling

We derive the dynamics by combining the ball kinematics on an inclined beam, its rolling resistance, and the servo motor behavior. A free-body diagram (Figure 1) shows forces acting on the ball along the beam axis: gravitational component $mg\sin\alpha$, centrifugal force $F_c = m(L/2 - z)\dot{\alpha}^2$, and rolling inertia force $F_r = J_b/r_b^2, \ddot{z}$, with relevant system parameters: z is the ball displacement from beam center, α is beam tilt,

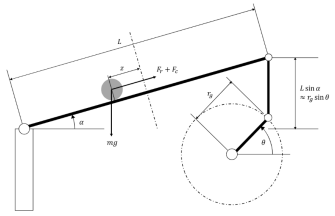

Figure 1: Ball and beam setup

Fig. 1. Free-body diagram of the ball–and–beam system.

m the mass, L beam length, r_b ball radius, $J_b = \frac{2}{5}mr_b^2$ its moment of inertia, gravity g, motor time constant τ, and motor constant K.

Applying Newton's second law along the beam axis:

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

Since $J_b/r_b^2 = 2m/5$, (1) simplifies to:

$$\ddot{z} = \frac{5}{7}g\sin\alpha - \frac{5}{7}(\tfrac{L}{2} - z)\dot{\alpha}^2. \tag{2}$$

The servo arm of length r_g imparts tilt α to the beam through the relation

$$r_g\sin\theta = L\sin\alpha. \tag{3}$$

For small angles, $\sin\alpha \approx \alpha$, giving $\sin\alpha \approx (r_g/L)\sin\theta$ and $\dot{\alpha} \approx (r_g/L)\cos\theta\dot{\theta}$. Substituting into (2) and reintroducing the full nonlinear terms yields the ball acceleration in terms of z and θ as in (5).

The servo motor produces torque $\tau = KV$ proportional to input voltage V, and exhibits first-order dynamics:

$$\ddot{\theta} + \frac{1}{\tau}\dot{\theta} = \frac{K}{\tau}V, \tag{4}$$

with time constant τ. Defining input $u = V$ completes the motor equation in (5).

Combining the above, we then define the state vector $x = [z, \dot{z}, \theta, \dot{\theta}]^\top$ and the full continuous-time nonlinear state

equations are:

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

III. Observer and Controller Design

A. Observer Design

1) EKF: For state estimation we used the extended Kalman filter (EKF), which is a recursive estimator that linearizes a nonlinear system around the current estimate, then applies the standard discrete Kalman-filter time and measurement updates. It is optimal (in the minimum-mean-square-error sense) for systems with Gaussian noise and locally linear dynamics [1].

We chose to use this observer because we only measure ball position and pendulum angle as part of this project, so we need an observer to reconstruct the full four-state vector. The extended Kalman filter (EKF) is a natural choice for this nonlinear plant, because it (a) linearizes online around the current estimate, (b) fuses noisy measurements optimally under Gaussian assumptions, and (c) admits a well-studied discrete implementation suitable for real-time execution on typical hardware. The theoretical basis for the EKF is as follows. We assume the continuous-time nonlinear dynamics

$$\dot{x} = f(x,u), \quad y = Cx + v,$$

with process noise $w \sim \mathcal{N}(0, Q)$ and measurement noise $v \sim \mathcal{N}(0, R)$. Discretizing with sampling interval Δt gives

$$x_{k+1} = f(x_k, u_k) + w_k, \quad y_k = C\,x_k + v_k.$$

Then, we linearize f about $\hat{x}_k$:

$$A_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_k, u_k}$$

and propagate the state and covariance:

$$\hat{x}_{k+1}^- = f(\hat{x}_k, u_k), \tag{6}$$

$$P_{k+1}^- = A_k\,P_k\,A_k^\top + Q. \tag{7}$$

To update the measurement, we compute the Kalman gain and correct: Let

$$S_{k+1} = C\,P_{k+1}^-\,C^\top + R, \qquad M_{k+1} = I - K_{k+1}C.$$

Then the update equations become

$$K_{k+1} = P_{k+1}^-\,C^\top\,S_{k+1}^{-1}, \tag{8}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + K_{k+1}\big(y_{k+1} - C\,\hat{x}_{k+1}^-\big), \tag{9}$$

$$P_{k+1} = M_{k+1}\,P_{k+1}^-\,M_{k+1}^\top \;+\; K_{k+1}\,R\,K_{k+1}^\top. \tag{10}$$

After much trial and error, the covariance update above is implemented in the so-called Joseph form [2] to guarantee symmetry and positive semi-definiteness of P despite finite-precision effects [3]:

$$P_{k+1} = (I - K_{k+1}C)\,P_{k+1}^-\,(I - K_{k+1}C)^\top + K_{k+1}\,R\,K_{k+1}^\top.$$

This explicitly preserves the required properties of the error covariance matrix and is more numerically stable. These steps repeat at each k, providing the best linear unbiased estimate of the state in the presence of noise.

We followed this theoretical formulation to implement the EKF under the Joseph formulation into our code. We used empirical methods to start with values for the covariances, with process noise covariance Q_w set to $\mathrm{diag}(0.01,\ 0.02,\ 0.01,\ 0.02)$ to reflect highest uncertainty in angular-velocity and acceleration dynamics, lower uncertainty in position and angle. Measurement noise covariance R_v was chosen as $\mathrm{diag}(1,\ 0.09)$ based on sensor specs (±1 mm position, ±0.3° angle). Initial estimator covariance $P_0 = \mathrm{diag}(10^{-1}, 10^{-1}, 10^{-1}, 10^{-1})$ encodes a small prior uncertainty around the equilibrium start.

B. Control Method 1: EKF + FL + LQR

For this control method, we used the EKF for state estimation and FL with LQR for control. The theoretical basis for FL and LQR are detailed below, as are the empirical parameter choices we made for the implementation.

1) Feedback Linearization: Feedback linearization is an advanced control technique that cancels system nonlinearities to yield exact linear behavior. Consider a general affine system:

$$\dot{x} = f(x) + g(x)\,u, \tag{11}$$

with output

$$y = h(x). \tag{12}$$

Differentiating the output repeatedly until the input u appears yields:

$$y^{(r)} = L_f^r h(x) + L_g L_f^{r-1} h(x)\,u, \tag{13}$$

where the Lie derivatives along f and g are defined as:

$$L_f h(x) = \frac{\partial h}{\partial x}\,f(x), \tag{14}$$

$$L_g h(x) = \frac{\partial h}{\partial x}\,g(x). \tag{15}$$

The relative degree r is the smallest integer such that:

$$L_g L_f^{r-1} h(x) \neq 0. \tag{16}$$

To enforce desired output tracking behavior, we introduce a virtual input:

$$v = y_{\mathrm{ref}}^{(r)} + K_e\begin{bmatrix} e^{(r-1)} & \cdots & e \end{bmatrix}^\mathsf{T}, \tag{17}$$

where the tracking errors are defined as:

$$e^{(i)} = y^{(i)} - y_{\mathrm{ref}}^{(i)}, \quad i = 0, 1, \ldots, r-1. \tag{18}$$

Substituting into the output derivative, the feedback linearizing control law becomes:

$$u = \frac{v - L_f^r h(x)}{L_g L_f^{r-1} h(x)}. \tag{19}$$

Necessary conditions for exact feedback linearization are that the vector fields $g, [g,f], [f,[g,f]], \ldots$ must be involutive and the zero dynamics (internal dynamics when $y = y_{\text{ref}}$) must be stable.

2) LQR: We combined this technique with that of LQR in order to linearize and stabilize the system. To derive the LQR formulation, consider the continuous-time LTI system

$$\dot{x} = A\,x + B\,u, \quad x(0) = x_0,$$

and the infinite-horizon cost

$$J = \tfrac{1}{2}\int_0^\infty \left(x^\mathsf{T} Q\,x + u^\mathsf{T} R\,u\right)\mathrm{d}t, \quad Q = Q^\mathsf{T} \succeq 0,\ R = R^\mathsf{T} \succ 0.$$

Form the Hamiltonian

$$\mathcal{H} = \tfrac{1}{2}x^\mathsf{T}Qx + \tfrac{1}{2}u^\mathsf{T}Ru + \lambda^\mathsf{T}(Ax + Bu),$$

so stationarity $\partial\mathcal{H}/\partial u = 0$ gives

$$u = -R^{-1}B^\mathsf{T}\lambda.$$

With $\lambda = P\,x$, we obtain the Algebraic Riccati Equation

$$A^\mathsf{T}P + P\,A - P\,B\,R^{-1}B^\mathsf{T}P + Q = 0.$$

and from there, get that the optimal law is

$$u^*(t) = -K\,x(t), \quad K = R^{-1}B^\mathsf{T}P,$$

and the closed-loop $\dot{x} = (A - BK)x$ is Hurwitz. A Lyapunov function $V = x^\mathsf{T}Px$ satisfies $\dot{V} = -x^\mathsf{T}Qx - u^\mathsf{T}Ru \leq 0$, ensuring asymptotic stability.

We followed the LQR theoretical formulation to compute state-feedback gains to use for control. We tuned Q and R repeatedly to try to improve behavior, and settled on

$$Q = \mathrm{diag}(100, 100, 10, 1), \quad R = 0.1$$

by empirically increasing the weights on ball position and velocity to prioritize rapid tracking while keeping beam-angle penalties low to avoid oscillations; the relatively small R allowed aggressive control action without saturating the actuator. We

C. Control Method 2: EKF + DLQR + Integral Control

For the second control method, we used the aforementioned EKF formulation for state estimation and DLQR plus integral control based on 4. Our controller uses two cascaded modules: an EKF for state estimation, predicting the nonlinear dynamics via Euler integration and updating with measurements using the Joseph form [3], and a discrete-time LQR with an added position-error integrator based on [4] since that was where the most error was occurring in ball tracking. We form the augmented state

$$\xi_k \;=\; \begin{bmatrix} \hat{x}_k \\ \displaystyle\sum_{i=0}^{k} (z_i - z_i^r)\,\Delta t \end{bmatrix},$$

discretize via zero-order hold,

$$A_d = e^{A\Delta t}, \quad B_d = \int_0^{\Delta t} e^{A\tau}B\,d\tau,$$

and minimize

$$J \;=\; \sum_{k=0}^{\infty} \left(\xi_k^\top Q_{\text{aug}}\,\xi_k + u_k^\top R\,u_k\right)$$

with state and input weight matrices found empirically as

$$Q_{\text{aug}} = \mathrm{diag}(?,\ ?,\ ?,\ ?), \quad R = ?.$$

These values were iteratively refined through experiments: first adjusting the state weights to try to improve tracking, then increasing R to reduce saturation.

The optimal gain $K = [K_x,\ K_i]$ solves the discrete algebraic Riccati equation, similar to the LQR formulation explained above,

$$P = A_d^\top P A_d - A_d^\top P B_d\,(R + B_d^\top P B_d)^{-1} B_d^\top P A_d + Q, \tag{20}$$

$$K = (R + B_d^\top P B_d)^{-1} B_d^\top P A_d. \tag{21}$$

Finally, we add the integral control to try to reduce the position error [4], while also incorporating torque control to cancel the nonlinearities of the system [5]:

$$u_k = \frac{-\,f_x(x_k, \dot{x}_k) - K_x\,e_k}{g_x(x_k)} - K_i \sum_{i=0}^{k} (z_i - z_i^r)\,\Delta t,$$

where $f_x(x_k, \dot{x}_k)$ represents the nonlinear drift dynamics at the current state x_k and velocity $\dot{x}_k$, and $g_x(x_k)$ denotes the input gain mapping the control input to the system's response. The term K_x is the linear quadratic regulator (LQR) feedback gain, computed to stabilize the closed-loop dynamics around the nominal trajectory, and e_k is the state error vector at time step k. The quantity z_i denotes the measured output (ball position) at time step i, while z_i^r is the corresponding reference value to be tracked. The summation $\sum_{i=0}^{k}(z_i - z_i^r)\,\Delta t$ accumulates the position tracking error over time, providing integral action to eliminate steady-state bias. After trials, to prevent integrator windup and maintain stability 6, we also clamped accumulated error before being multiplied by the integral gain K_i.

D. Safety Measure

For both controllers, we implemented a safety measure to prevent the control from violating the safety constraints. After computing the control input, we check whether the beam angle has exceeded the safety limits of maximum and minimum bounds of ±45°. If the angle has exceeded, the control input is overridden to drive the beam to the allowable range. We define the control input to be the maximum or minimum value of the current control input and the adjusted control input. The adjusted control input is proportional to how far the current beam angle has deviated from the bounds and is scaled by a gain of 10. The scaling is to ensure strong response towards the safety violation, thus ensuring safety of the system.

IV. Simulation Results

The simulation results for the two combinations of controllers and observers in the MATLAB and Simulink simulations are presented in Table 2, and the plots are presented in Figures 2-7.

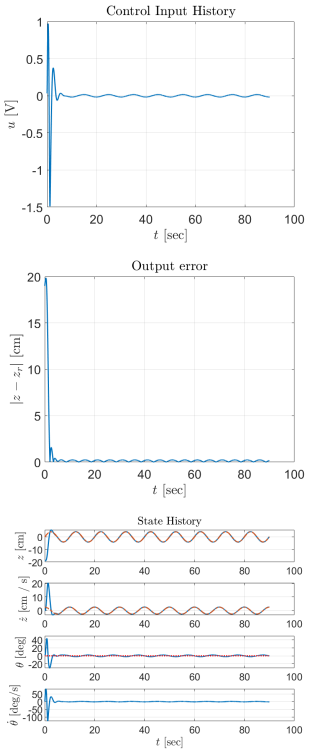

Fig. 2. **(a)** MATLAB – EKF + FL

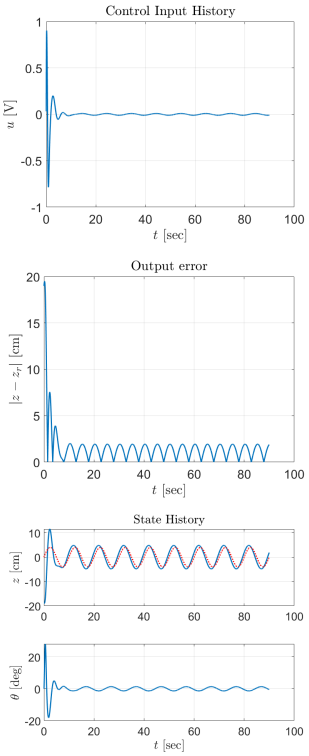

Fig. 3. **(b)** Simulink – EKF + FL

Fig. 4. Comparison of EKF + FL simulation results under sine wave input for MATLAB and Simulink implementations.

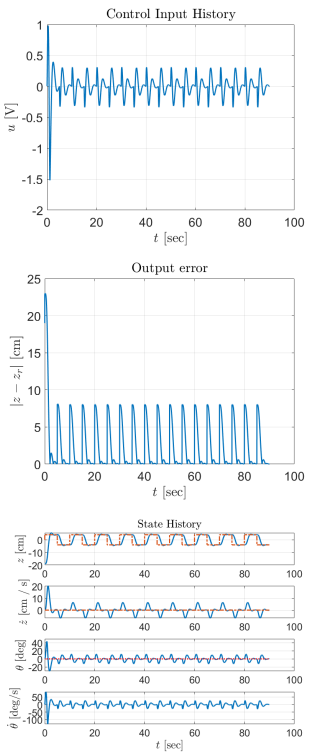

Fig. 5. **(a)** MATLAB – EKF + FL

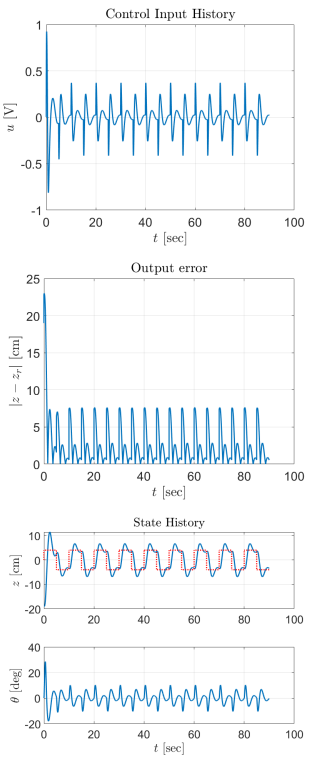

Fig. 6. **(b)** Simulink – EKF + FL

Fig. 7. Comparison of EKF + FL simulation results under sine wave input for MATLAB and Simulink implementations.

TABLE I
DLQR SIMULATION PARAMETERS

Simulation	Q (diag)	R	Ki
Matlab	[123.45, 19.30, 0.32, 0.19]	0.04	0.01
Simulink	[1023.45, 129.30, 7.24, 5.87]	0.001	0

TABLE II
SIMULATION PERFORMANCE: CONTROLLER VARIATIONS ACROSS OBSERVER CONDITIONS

Reference	Simulator	Controller	Tracking Cost	Energy Cost	Score
Sine	Matlab	DLQR	0.79	0.11	0.90
	Matlab	FL/LQR	0.93	0.06	0.99
	Simulink	DLQR	0.84	0.001	0.84
	Simulink	FL/LQR	1.12	0.03	1.15
Square	Matlab	DLQR	2.9	0.49	3.39
	Matlab	FL/LQR	3.49	0.16	3.65
	Simulink	DLQR	2.20	0.01	2.21
	Simulink	FL/LQR	2.93	0.11	3.04

V. HARDWARE EXPERIMENT RESULTS

The hardware results for the two combinations of controllers and observers are presented in Table 2, and the plots are presented in Figures 8-13.

TABLE III
CONTROLLER PARAMETERS FOR DLQR AND FL (SAME PARAMETERS IN SIMULATION AND HARDWARE)

Controller	Parameter	Value
DLQR	Q(diag)	[1000, 100, 0.1, 1e-10]
	R	0.0035
	Ki	1.9
FL	K	[23.4521, 29.3025, 17.2402, 5.8720]

TABLE IV
HARDWARE PERFORMANCE

Reference	Controller	Tracking Cost	Energy Cost	Total Score
Evaluation Trajectory	EKF + DLQR	0.35	2.29	**2.64**
	EKF + FL	0.29	3.07	3.37

VI. CONCLUSION

In simulation, the EKF-DLQR controller has a definitive edge over the EKF-Feedback linearization controller across both trajectory types and both simulation types. For the sine-wave reference trajectory the gap between the two controllers is minute with only a 0.1-0.2 difference, whereas for the square wave the gap is significantly wider, particularly in the Simulink. The output error graphs show a similar pattern but it is clear that once the DLQR controller converges onto the trajectory it has a lower phase delay and consistently lower output error for both sets of simulations and trajectories. These trends match up with the results seen in the hardware

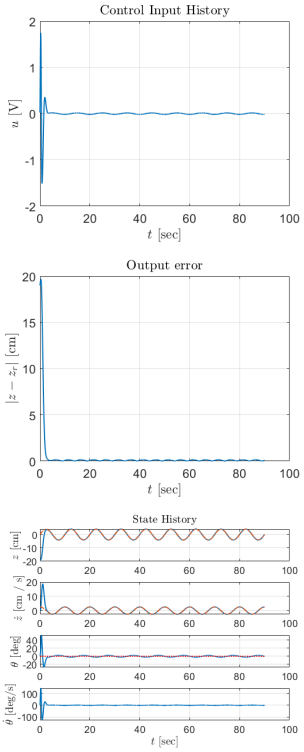

Fig. 8. **(a)** MATLAB – EKF+DLQR

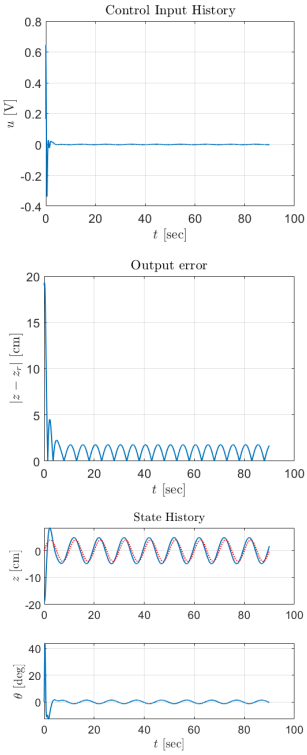

Fig. 9. **(b)** Simulink – EKF + DLQR

Fig. 10. Comparison of EKF + DLQR simulation results under sine wave input for MATLAB and Simulink implementations.

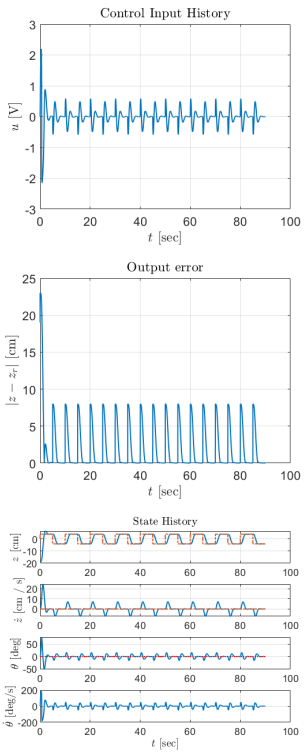

Fig. 11. **(a)** MATLAB – EKF + DLQR

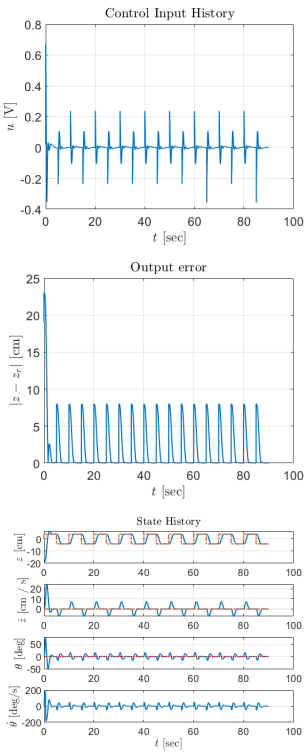

Fig. 12. **(b)** Simulink – EKF + DLQR

Fig. 13. Comparison of EKF + DLQR simulation results under sine wave input for MATLAB and Simulink implementations.

testing where the DLQR has the lower cost and is ultimately the more successful controller; however, in real world testing there is much more noise both in control input history and output error. Another notable difference is that the working parameters for simulation and hardware are very dissimilar for the DLQR; in the simulation, integral action appears to produce no significant benefit, but integral action seems to crucial in the real world testing for effective control.

Although both controllers performed well in simulation hardware testing yielded significantly different initial results: total costs 2 to 4 times higher due to noisier sensors and unmodeled friction dynamics. In simulation, there is no noise involved so even simple observers such as a Luenberger observer for the linearized system can achieve high estimation accuracy. We quickly discovered that the ball position data had relatively high noise which meant that our initial attempt with a simple Luenberger Observer resulted in noisy velocity estimates. These poor velocity estimates combined with the nonlinear disturbance caused by friction forces resulted in the controller frequently oscillating around the reference trajectory with a large amplitude. This behavior was consistent for a wide range of controller gains for the DLQR and feedback-linearization controller.

To address this issue, we focused more on developing a more accurate observer and tested three options: default EKF, EKF with Joseph stabilized form, and a high gain nonlinear observer. All three options with relatively untuned gains

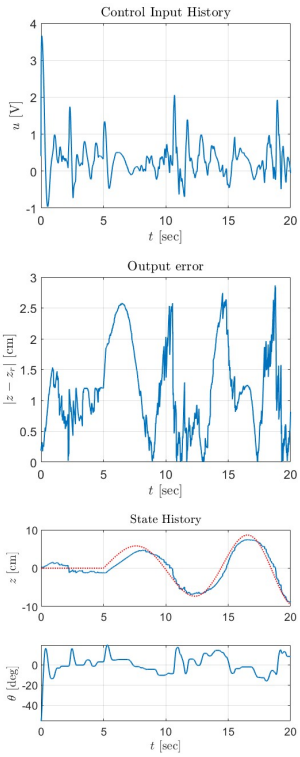

Fig. 14. Hardware - EKF + DLQR.

produced much smoother and consistent velocity estimates but introduced a new issue: stalling. With these observers, the controller would get stuck for extended periods even as the error grew rapidly. The high gain nonlinear observer exhibited the worst stalling so it was disregarded first, and a few hours of tuning was performed for the process and measurement covariance matrices in both EKF versions. With tuned noise covariance matrices, the Joseph stabilized form of EKF minimized this stalling behavior and was used in the final implementation, producing the results seen in the tables/plots above.

The gains for both controllers were re-tuned and the DLQR with the integral action far outperformed the feedback linearization. The DLQR exhibits less "chattering" at the beginning phase of the trajectory with zero reference and closely tracks the growing sinusoidal wave. However, it still shows slight stalling particularly at the first peak of the reference whereas the feedback linearization does not get stuck in the same way; the tradeoff here is that feedback linearization suffers from greater oscillations and steady-state error.

We added integral action in the DLQR to eliminate steady state error, which counteracts the steady state errors from the unmodeled friction forces, removing the need for the potentially high control gains to drive the system to the reference and reducing oscillations. We also employed computed torque control [5] in the DLQR to eliminate nonlinearities: a feedforward nonlinearity compensation term cancels the nonlinearities in the system and linearizes about the trajectory, which allows the controller to track the fairly dynamic and aggressive trajectory. Feedback linearization does not have the integral compensation, meaning that it produces a steady state offset, and the unaccounted nonlinearities from friction greatly degrade the feedback linearization. If we had more time, we would have added the integral control to the Feedback Linearization controller as well to try to reduce the steady state error. The handling of unmodeled friction dynamics appears to be the critical factor that separates the quantitative and qualitative performance of the two controllers on the hardware.

From these hardware experiments, we learned that observer choice and tuning were critical to the final performance of the controller. The test bench sensors had significant noise, particularly for the ball position, and friction which can hinder linear state estimators from smoothly converging. Unexpectedly, these two factors also varied greatly between test benches and in between runs, so choosing an observer that can model these added disturbances/nonlinearities well and tuning for the specific test bench is key. The impact of the observer is difficult to assess in simulation, so we discovered that the first step is settling on a well-performing observer. We found that the Extended Kalman Filter converged to smoother estimates compared to the Luenberger. However, it still required careful tuning for a few hours and the final estimation accuracy was highly dependent on the specific implementation of the covariance and filter gain updates. Tuning the observer allowed us to observe consistent trends and tune the controller gains much quicker. On the controller side, steady-state error compensation was necessary and we found a simple integral term to be an effective method. If we were repeating this lab, we would develop multiple observers – with a focus on ones designed for nonlinear systems – early on and prioritize tuning them on the hardware before modifying the controller. We were surprised by how much results can differ between runs so tuning should be done after determining the general trends. We also discovered that while controller/observer parameters had a massive impact, the implementation was more important to verify early which is what we would focus more on in a repeat attempt. Overall, this lab helped us understand the limitations of simulations even for a relatively simple system, and informed us more about the design process for controllers on real hardware.

REFERENCES

[1] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," University of North Carolina at Chapel Hill, Chapel Hill, NC, Technical Report 95-041, 2001.

[2] R. Zanetti and K. J. DeMars, "Joseph formulation of unscented and quadrature filters with application to consider states," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 6, pp. 1860–1864, 2013. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/1.59935

[3] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, 2nd ed. Providence, RI: AMS Chelsea Publishing, 2005, originally published by John Wiley & Sons, 1968.

[4] D. Ruscio, "Discrete lq optimal control with integral action: A simple controller on incremental form for mimo systems," *Modeling, Identification and Control: A Norwegian Research Bulletin*, vol. 33, 04 2012.

[5] D. Bravo and C. Rengifo, "Comparative analysis between computed torque control, lqr control and pid control for a robotic bicycle," 10 2019.

[6] K. J. Astrom and L. Rundqwist, "Integrator windup and how to avoid it," in *Proceedings of the 1989 American Control Conference*, 1989, pp. 1693–1698, s2CID 36848080.