ME155C Final Project Report
Inverted Pendulum Control: Swing-up, Balance, and Catching

Raaghav Thirumaligai, Cole Giusto, Tien Nguyen

June 8, 2025

Abstract

This project focuses on the identification, modeling, and control of an inverted pendulum on a cart system. Through experimental identification, control design, and closed-loop testing, we implemented robust swing-up, balancing, and catching controllers. System identification methods included logarithmic sine sweeps and parametric identification with MATLAB's tfest. Controller design employed energy-based swing-up methods and robust LQR/LQG balancing strategies. Experimental validations confirm effective performance, achieving stable equilibrium transitions and pendulum catching.

1 Introduction

This project investigates the control of an inverted pendulum on a cart, specifically targeting swing-up, balancing, and catching tasks. The inverted pendulum, a classic control problem, requires precise and robust feedback control strategies due to its inherent instability. Previous literature includes methods involving energy-based swing-up and state-space feedback stabilization. The report is structured with system identification in Section 2, controller design in Section 3, closed-loop results in Section 4, and conclusions in Section 5.

2 System Identification

2.1 Process Description

The experimental setup consists of an asymmetric metal rod mounted to a motor-driven cart, with encoders providing measurements of pendulum angle (θ) and cart position (x). The control input is the motor voltage (u), and the primary controlled outputs are cart position and pendulum angle.

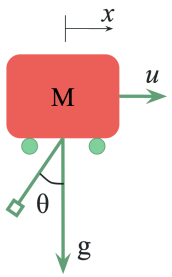


Figure 1: Experimental setup: Cart-pendulum system.

2.2 Identification Methods

2.2.1 Non-parametric

Non-parametric identification utilized a logarithmic sine sweep using the `logspace` of frequencies between$(0.3 - 3$ Hz).

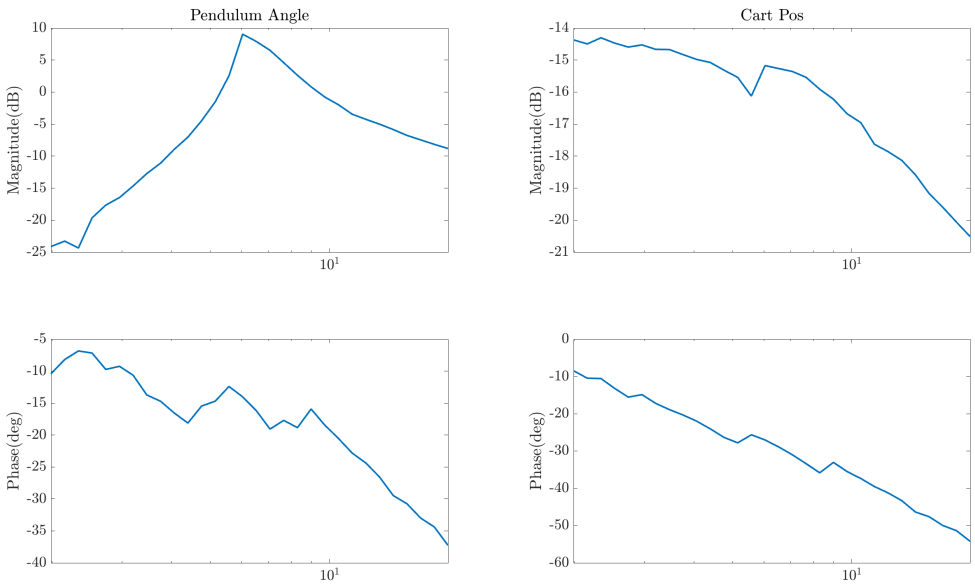


Figure 2: Non-parametric bode plots of the system.

We used this to help inform our parametric identification.

2.2.2 Parametric

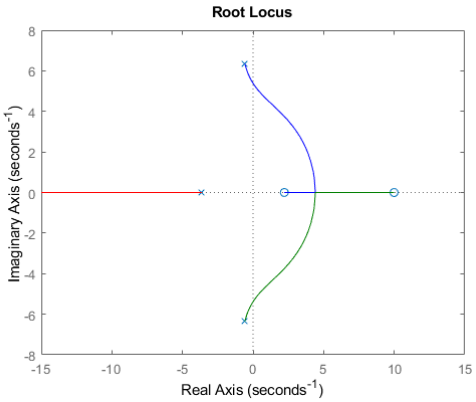
For our parametric identification, we used the same trajectories as for the non-parametric. Parametric identification was done using MATLAB's `tfest` function, performing 30 experimental trials.

We assumed a structural integrator in the transfer function to cart position. As shown in the bode plots, this integrator was actually the dominant element of the model for the cart position, with the other pole being far into the LHP.

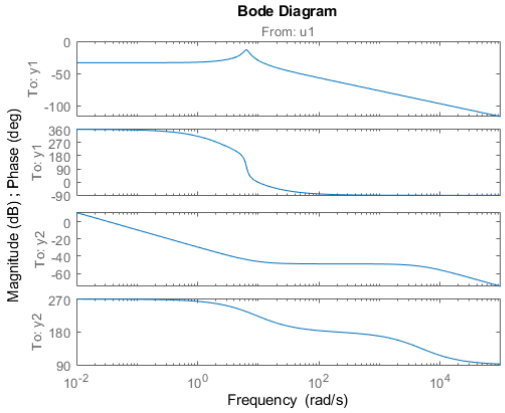
For the transfer function of the pendulum, we found that 3 poles and 2 zeros provided the best fit. The bode plots resulting from this identification can be seen in Figure 3b.

As you can see, there is a peak in the bode plot that corresponds to a resonant frequency in the system.

In order to modify our model for the pointing up case, we simply reflected one pole and both zeros from the pendulum angle transfer function about the imaginary axis. This makes sense, as the dynamics flip from stable to unstable. The transfer functions for this modified system can be seen in Figure 4b.

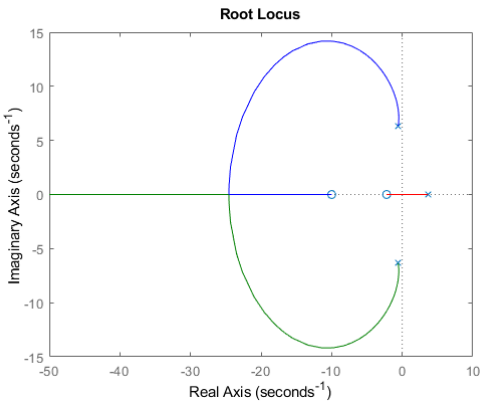


(a) Root locus for pendulum in downward position.

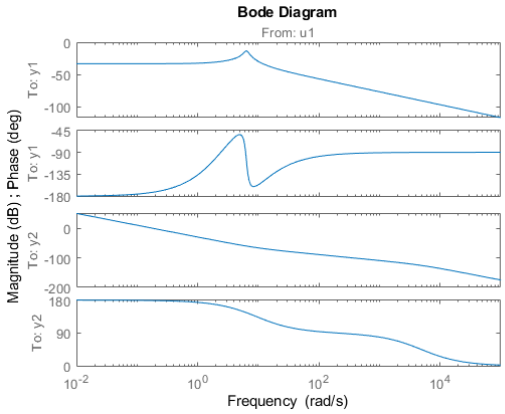


(b) Identified frequency response for cart dynamics and pendulum dynamics downwards.

Figure 3: Root locus plot and Bode of the pendulum dynamics in the downward configuration.



(a) Root locus for pendulum in upward position.



(b) Identified frequency response for cart dynamics and pendulum dynamics upwards.

Figure 4: Root locus plot and Bode of the pendulum dynamics in the upward configuration.

In addition, the root loci of the pendulum transfer functions can be seen in Figures 3a and 4a. Observe the reflected pole and zeros.

3 Controller Design

3.1 Swing-Up Controller

The swing-up controller uses an energy addition method to drive the pendulum toward the upright position. Around every zero crossing of the pendulum angle θ, the controller applies a force to move the cart in the direction opposite to the pendulum's velocity in order to

increase the energy in the pendulum. This continues until the pendulum is able to fully "swing up". Additionally, a simple proportional controller is used to regulate the cart's position to stay near zero meters, preventing excessive cart movement during the swing-up phase. The activation is limited to a range $\theta \in [-30°, 30°]$ to try and reach the inverted state with a small velocity. This slow swing-up allows the balancing controller to take over without having to attenuate too much extra energy.

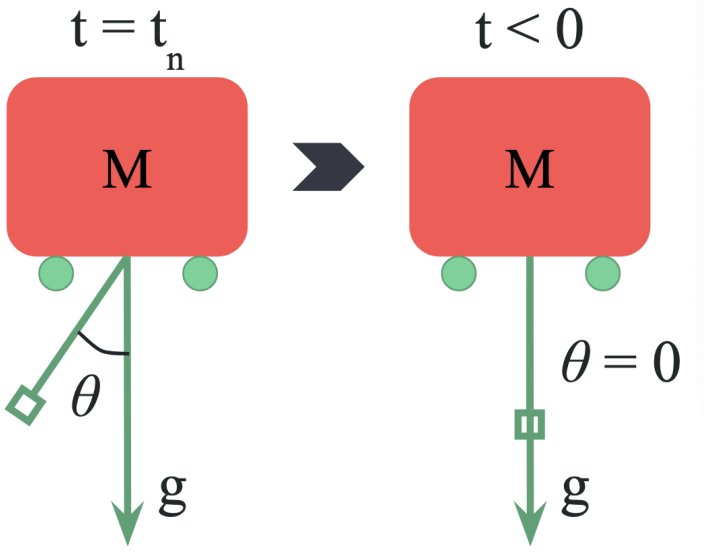


Figure 5: When we reach a region of the state-space in an epsilon bound around zero, this correlates to a downwards pendulum and centered cart with small kinetic energy. This triggers the swing up controller as it is now safe to attempt adding controlled energy back into the pendulum to stabilize.

3.2 Balancing Controller

The balancing controller employs LQR/LQG methods to stabilize the pendulum in the inverted (unstable) equilibrium within a narrow angular region. First, we created an LQR controller to properly penalize the relevant states and try to achieve the best performance. The Q matrix was tuned to aggressively control the pendulum angle but allow the cart to move relatively freely. This allowed the system to use large cart movements to stabilize the equilibrium and stay in the linear regime.

$$Q = \begin{bmatrix} 2000 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

The columns correspond to the states: angular position, angular velocity, cart position, and cart velocity respectively.

For the LQG state estimation to feed into the LQR controller, we used a Kalman filter block

in Simulink. With the default settings, feeding in our output and input signals, the Kalman filter was able to reliably provide an accurate estimate of the state to the controller.

3.3 Catching Controller

Similar to balancing, catching employs LQR/LQG but aims to return the pendulum reliably to the stable downward equilibrium.

To achieve this, we used the weights Q below:

$$\begin{bmatrix} 8000 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

This even more aggressive control of the angular position prevents the pendulum from swinging to more than 90°, which means that the catching controller does not need to wait for it to swing back down.

3.4 State Machine

Control modes transition between swing-up, balancing, and catching using a robust state machine architecture. This state machine was implemented in Simulink using logic gates, comparing the pendulum angle and velocity to determine the current state and enable the correct controller.

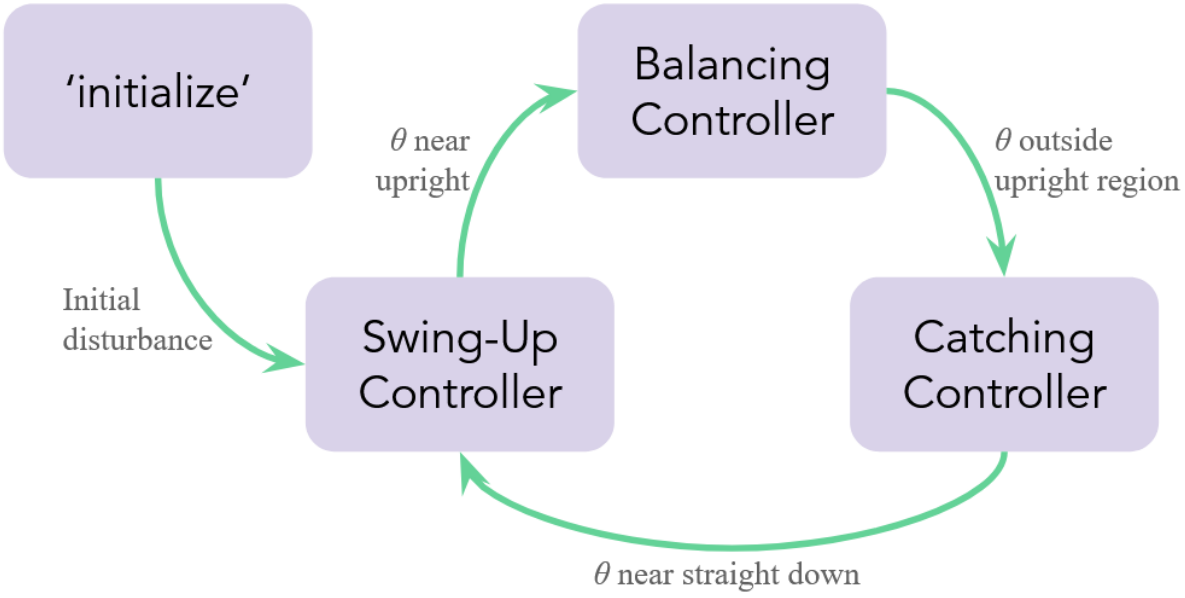


Figure 6: State machine for mode switching between controllers.

This digital logic ensured that our system was able to recover from most disturbances we tried, with the only exceptions being when the momentum of the pendulum caused the cart to "jump" off of the track.

3.5 Full Simulink Block Diagram

These components come together to form the controller seen in Figure 7.

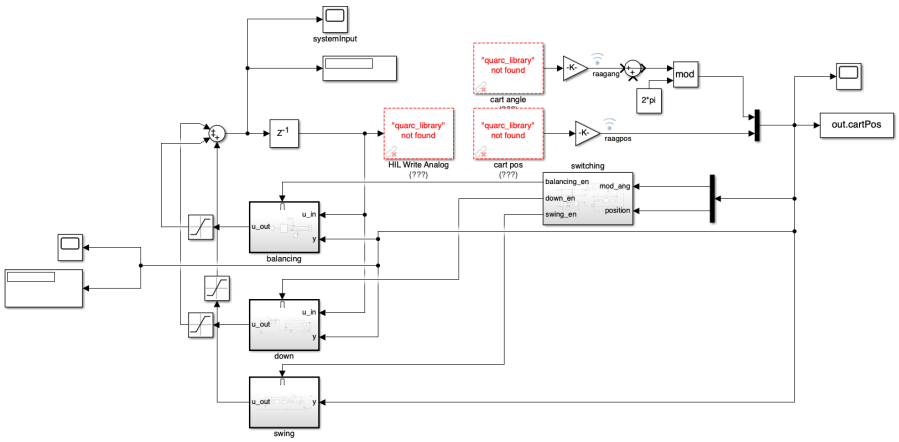


Figure 7: Full controller in Simulink

You can see the switching block on the bottom right that enables the three subsystems to the right.

4 Closed-Loop Performance

Closed-loop experiments validated the controllers' performance. The step response showed quick stabilization and effective handling of mode transitions, with no large steady state oscillations. And, even when the balancing controller failed, the catching and swing-up controllers could bring it back to the upward position within 15s. See the video here.

5 Conclusions and Future Work

The controllers successfully addressed swing-up, balancing, and catching objectives, validating theoretical designs with robust experimental performance. Further work includes enhancing failure logic and addressing steady-state error. We could augment the state machine to better handle when the cart hits the guard rails or when the pendulum swing-up is too fast for the balancing controller to act effectively. Also, when the pendulum made a full rotation, the angular encoder could not measure this perfectly, and the cart would slowly drift to one side during balancing. This steady-state error could be mitigated by adding a small integrator to the controller as a correction term.

We recognize our system ID and closed-loop sections don't include all metrics or comparisons, but we successfully implemented swing-up, balance, and catching on hardware!