Cartpole PID Controller Design Report

Samarth Barve

***Abstract*— In this report, we detail the design and implementation of a PID controller for stabilizing an inverted cartpole system using MATLAB. The controller is designed to maintain the cartpole in an upright vertical position by adjusting the force applied to the cart.**

I. INTRODUCTION

The inverted cartpole problem is a classic control problem that involves stabilizing a pole attached to a cart in an upright position. Our approach to solving this problem involves designing a PID controller to regulate the cart's movement and maintain the pole's balance.

II. APPROACHING THE PROBLEM

Our initial step was to understand the dynamics of the inverted cartpole system. We analyzed the physical principles governing the system and identified the key variables affecting its stability. One crucial aspect we focused on was the relationship between the cart's position, the pole's angle, and the force required to maintain equilibrium.

III. INITIAL ATTEMPT

For our initial attempt, we implemented a basic PID controller with arbitrary values for the proportional, integral, and derivative gains (KP, KD, KI). We used trial and error to adjust these values and observed the system's response to different input signals. However, this approach was largely heuristic, and while it provided some insight, it did not yield optimal results.

IV. FINAL ATTEMPT

After several iterations and experimentation, we employed a systematic approach to fine-tune the PID controller. We conducted sensitivity analysis to understand the impact of each PID gain on the system's behavior. This allowed us to adjust the KP, KD, and KI values within appropriate ranges. We utilized MATLAB's simulation environment to validate our controller's performance and iteratively optimized the gains to achieve optimal stability and responsiveness.

V. LEARNING OUTCOME

Through the process of designing and tuning the PID controller, we gained insights into the importance of each PID gain and its impact on the system's behavior. We learned to balance trade-offs between stability, responsiveness, and overshoot, leading to a more robust and effective controller design. Additionally, we developed a deeper understanding of control system design principles and their practical application in real-world scenarios.

VI. CONTROLLER TRAINING

The PID controller was trained by adjusting the KP, KD, and KI values within certain ranges. We conducted extensive experimentation to identify the optimal combination of gains that resulted in stable and responsive control behavior. Typically, the values of KP, KD, and KI for the inverted cartpole problem fall within the following ranges:

- Proportional Gain (KP): 1 to 10
- Derivative Gain (KD): 0.1 to 1
- Integral Gain (KI): 0.01 to 0.1

We iteratively adjusted these values and evaluated the controller's performance until satisfactory stability and responsiveness were achieved.

A. PID Controller Constants

The PID controller utilizes the following constants for position and angle control:

- **Position Control Constants:**
  - Proportional Gain (Kp_x): 5
  - Integral Gain (Ki_x): 0.2
  - Derivative Gain (Kd_x): 0.04
- **Angle Control Constants:**
  - Proportional Gain (Kp_θ): 4
  - Integral Gain (Ki_θ): 0.3
  - Derivative Gain (Kd_θ): 0.02

B. Implementation Details

- **Persistent Variables:**
  - `integral_error_x`: Persistent variable for integral error in position control.
  - `integral_error_theta`: Persistent variable for integral error in angle control.
- **Initialization:** The persistent variables are initialized if they are empty.
- **Error Calculation:** Errors for position (`error_x`) and angle (`error_theta`) are calculated by comparing the current state with the desired state. Error derivatives (`error_x_dot` and `error_theta_dot`) are also computed based on desired and actual velocities.
- **Integral Error Update:** Integral errors for position and angle are updated using the accumulated errors.
- **Control Input Calculation:** Control inputs (`u_x` and `u_theta`) are computed using PID control formulas for position and angle, respectively.
- **Combining Control Inputs:** Control inputs for position and angle are combined to obtain the final control signal (`u`).

- **Control Input Limitation:** The final control signal is constrained to ensure it remains within the allowable limits (`u_max`).

VII. Controller Design

The force equation governing the inverted cartpole system was derived based on the dynamics of the cart and pole. It involves balancing the gravitational and inertial forces acting on the cart and pole while considering the cart's position and the pole's angle. The PID controller was then designed to regulate this force to maintain the system's stability.

VIII. Conclusion

The implemented PID controller effectively stabilizes the inverted cartpole system by continuously adjusting the force applied to the cart based on the current and desired states. Fine-tuning of the PID gains and further optimization may be required based on the specific dynamics and performance requirements of the system. Additionally, exploring built-in MATLAB functions for control system design and tuning could enhance the controller's performance and efficiency.

IX. Suggestions for Optimization

- **Parameter Tuning:** Experiment with different values of PID gains to achieve better performance and stability.
- **Robust Control Techniques:** Investigate robust control methods to enhance the controller's performance in the presence of disturbances or uncertainties.
- **Model Predictive Control (MPC):** Explore advanced control strategies like MPC for improved trajectory tracking and disturbance rejection.
- **MATLAB Control Toolbox:** Utilize MATLAB's built-in control toolbox functions for system modeling, analysis, and automatic controller design to streamline the development process and improve controller performance.

References

[1] MATLAB documentation for control system design and analysis functions.

X. Additional Documentation

A. Cart-pole balancing

The objective of the assignment was to write a Proportional-Integral-Differential (PID) controller to balance an inverted pendulum on a cart whilst keeping the cart position at the origin. In addition to this it was also required to experiment with a parameter tuning algorithm (other than the manual-tuning process) to obtain best PID gains for the controller. A complex controller was also to be experimented with from a given set of choices.

B. Implementation

0.1 PID Control I have implemented the PID system for the case of balancing the pendulum with the initial state being upright. The implementation uses a combination of two different PID controllers: PID for the pendulum & PID for the cart. The PID system for the pendulum takes into account the angle & the angular-velocity of the pendulum whilst the PID controller for the cart uses the velocity & the distance of the cart from the origin. The proportional gains (for both controllers) are subjected against the positional errors: the distance from origin in the case of the cart and the deviation-angle from its intended balance position (π - radians) for the pendulum. The cumulative value of these positional-errors are used to generate the integral gain for the system. Moreover, the velocity terms for both the cart and the pendulum were utilized as inputs for the differential gain. The pendulum's angle (θ) was passed into the PID control function in its $\cos(\theta)$ and $\sin(\theta)$ values. Hence, according to the position of the pendulum, these values varied in a non-uniform manner. It was required to re-scale them to a range such that the positional error (of the pendulum) calculated using θ reflected the magnitude and the direction of the force to be applied on the cart. Thus, I converted the input-angle to be within a range between (02π) rad based on the $\cos$ and $\sin$ values of the angle in different quarters of the coordinate system. This conversion ensured that a positive-error (indicating positive force) was required when the pendulum was on the right side of the cart and vice-versa when it was on the left. It also ensured that the magnitude of the force was higher for pendulum states in the lower quarters of the plane. The cart's positional error was deduced by subtracting the distance of the cart from the origin (received as an input to the function) from ZERO. Any state to the right of the origin demanded a negative force backwards and vice-versa when it was to the left of the origin. The differential errors were calculated directly by subtracting the the velocities of the two entities (cart and pendulum) from ZERO. However, when using this error term (with the gain parameter) in the controller it was noted that the direction of the pendulum's differential error had to be converted. This was because the pendulum's velocity was calculated positive to the anti-clockwise direction. When the pendulum is having a positive velocity (imagine the pendulum on the top-right quarter) the optimal control on the cart must be positive to project it towards the intended upright position. The differential error, when calculated as a difference between ZERO and a positive velocity gives a negative value. This results in a force pulling the cart further towards the left. This is evaded by inverting the sign of the differential error of the pendulum. Two global variable were maintained to keep track of the cumulative errors of each of the two control entities in the system. This cumulative error along with the learned gain was used to produce the integral gain for the two PID systems. The three (P,I,D) controls for the two entities (the cart and the pendulum) were summed separately to produce the final control parameter per PID

system. Finally, the difference between the control value for the pendulum's PID system and the cart's PID system ($PID_{pendulum} - PID_{cart}$) was output as the optimal control parameter for the system.

C. Learning process & outcomes

I started implementing the controller based on the basic idea behind PID control systems. Initially, I addressed the task as a single PID control problem and implemented the error terms (proportional, integral & differential) as a combination of both: the state variables of the pendulum and the cart. However, this resulted in a very unstable system. Even with numerous combinations of the parameter variables I was not able to observe any reasonable control. Then, upon talking to some of my colleagues I realized that the problem had to be approached as a system of 2 separate PID controllers. This allowed me to come up with a more stable solution. With the splitting of the controllers into two parts, I had to also verify that each of the error terms for the two PIDs were adjusted appropriately.