Lab: Control of an Inverted Pendulum on a Cart

CS3333 Industrial Instrumentation and Control
Dept. of Computer Science and Engineering, University of Moratuwa

Learning Outcomes

This lab aims to provide a gentle and simplified introduction to Control Systems, to teach you the utility of Control Systems in the real world. It will do so through a “toy” example: an Inverted Pendulum on a Cart.

Upon completing this lab, you will be able to:

- Explain what the terms of a PID controller represent (in terms of error);
- Implement a PID controller from scratch in any language you prefer;
- Tune a controller by trial and error;

Tutorial

Please refer to the document titled `cartpole.ipynb` in `PID_Lab.zip` and follow the steps there.

Exercise

You are to tune a PID controller where the environment's physical parameters are slightly modified based on your unique student ID. You will document your tuning process and analyze the controller's behavior.

The primary goal is to **balance the pole vertically** to achieve the following performance criteria for the pole angle (θ) when the simulation starts with a small random intial disturbance (maximum of 0.05 radians in either direction from vertical):

- The pole must remain balanced indefinitely.
- The maximum angle deviation *after the initial disturbance* should be minimized (e.g., aim for less than 10-15% of the initial disturbance, though perfect elimination might be hard).
- The pole angle should return to and stay within ±0.01 radians within 5 seconds.

The secondary goal is to keep the **cart centered horizontally** within the environment.

- The cart must not exceed 0.5 units of displacement from the center position during the simulation.

The following steps will guide your work for this lab:

Step 0: Create a copy of `cartpole.ipynb` with your student ID appended to it called `cartpole_{YOUR_STUDENT_ID}.ipynb`.

Step 1: Enter the **last 2 *numerical* digits** of your Student ID where it says `YOUR_INDEX_NUMBER` in the code. This will change the parameters of the system you are tuning, to be somewhat unique to you.
e.g. If your Student ID is 258011X, please enter 11.

Step 2: Tune a PID controller for this system, by changing the K_P, K_I and K_D values in `cartpole_{YOUR_STUDENT_ID}.ipynb`, while describing the steps you took to find your final K_P, K_I and K_D values. Document all your steps **IN THE JUPYTER NOTEBOOK ITSELF, IN MARKDOWN**.

- Start with potentially zero gains (K_I, K_D = 0) and tune K_P first, then K_D, then K_I, or use another systematic approach (e.g., trial-and-error based on understanding the effect of each gain).
- Explain *why* you made changes. For example: "Increased K_P, response became faster but more oscillatory." -> "Added K_D to dampen oscillations." -> "Small steady deviation observed, added small K_I to correct it."
- Include plots generated by the notebook showing the pole angle (θ), cart position (x), and control action (1 or 0) over time for your **final tuned controller**. You might optionally include plots from intermediate tuning steps that illustrate key improvements or problems.

Step 3: What are the limitations of your PID controller and the CartPole system itself? Consider the following points in your answer:

- The force applied to the cart is limited ±10.0 in Gym environment. How does this limit the controller's ability to correct large/fast disturbances?
- Can the controller keep the cart within typical position limits (e.g., -2.4 to +2.4)? Does it try to apply unrealistic forces if the angle gets too large?
- Is the PID structure itself sufficient? Are there situations (e.g., very large disturbances) where it fundamentally fails?
- What would happen to the system if the integrator term was left to increase indefinitely, as it currently does?

Submission guidelines

Create a `.zip` file of `cartpole_{STUDENT_ID}.ipynb` and the `pyproject.toml` file and upload it to moodle. Kindly ensure that it includes:

- The code with your `YOUR_INDEX_NUMBER` correctly entered and parameters modified.
- Your final K_P, K_I and K_D values.
- Code to run the simulation and generate plots for your final tuned controller.
- The required plots (angle, position, action vs. time) embedded in the notebook.

References

Chapter 14, “Fundamentals of Industrial Instrumentation and Process Control” - William C. Dunn (McGraw-Hill, 2005)

Control Tutorials for MATLAB and Simulink - Inverted Pendulum: System Modeling - https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling