Finding Balance: PID Control for a Self Balancing Robot

Stephanie B. Hernandez
Valencia College · Seneff Honors College · Electrical Engineering

Motivation

PID (Proportional Integral Derivative) controllers are used in a variety of industrial settings. They can regulate variables such as speed, temperature, flow, and pressure. PID can optimize the production, or ease the work load, of industrial processes.

This work seeks to understand PID controllers and their implementation to solve problems through exploration and data extraction in the form of a PID controlled self-balancing robot.

Methods and Materials

Research Background
- Fundamentals of PID control [1][2]
- Inverted Pendulum Model and Bench Scale Model
- PID controlled self-balancing robots
- Codes and Hardware [4][9]

Model Creation
- Microcontroller → Arduino UNO acts as a master and relay between the MPU6050 and H-bridge
- Degrees of Freedom → MPU6050 sensor with gyroscope and accelerometer capabilities needed for proper balancing [6]
- Wheels are controlled by the H-bridge.
- Everything is powered by two 9V batteries
- System is programmed with Arduino’s open-source IDE and libraries [4][9]

Inverted Pendulum on Cart

The inverted pendulum figure above illustrates the basic principles for self-balancing robots which gives the foundation for the programming. In this case, a force $u$ moves the cart right or left, with the goal being for the angle $\theta$ to equal $\pi$. The program will make adjustments according to these basic principles and through the PID portion of the code and its set parameters.

Results

A physical model was developed and is functioning successfully. As depicted in the stills above, the weight distribution still needs to be adjusted in order center the pendulum’s angle and make it equal $0$ like the model in Figure 2. The bot seems to be leaning back, which affects its ability to balance properly. The motors run at different speeds, requiring adjustments within the code to correct the issue. There also seems to be a traction issue when the bot is on different surfaces, further analysis is required for this issue.

Conclusions/Future

- Learned about PID control theory
- Learned to program microcontrollers
- Built a self-balancing robot model
- Successful demonstration
- Quantitative data analysis
- Graphing data from Arduino IDE
- Parametric study - the bot’s balance being affected by surface types - requiring different settings for each type.

References

4. How to Make a Balancing Bot at Home, Dahake, Tarun K., Youtube, 10 June 2018.
7. Sane, Ansel. “How to Interface Arduino and MPU6050 Sensor.” Maker Pro, EETechMedia LLC.

Acknowledgements: I would like to thank Dr. Mayank Tiagi for his support and mentoring throughout this experience, Daniel S. Schmidt for providing his materials as a foundation for my research, and my fellow REU participants for their help and kindness during our stay. Also, a sincere thank you to the CCT for this opportunity.