KCL ROBOTICS

3 Ways to Control a Cartpole

Introduction to Control Systems for Robotics enthusiasts

Workshop 1 - Introduction and PID Control

KCL Robotics Society

2022 (v1)

Introduction: Why?

Why control and why cartpole model?

Motivation

- **Why learn control?**

Because it is **<u>everywhere</u>**, especially in robotics!

- **Who are these workshops for?**

For any aspiring roboticist and anyone curious about the way control systems work.

- **What will we cover?**

We'll cover the 3 currently most popular approaches to control - classic control (PID), optimal control (LQR), and reinforcement learning.

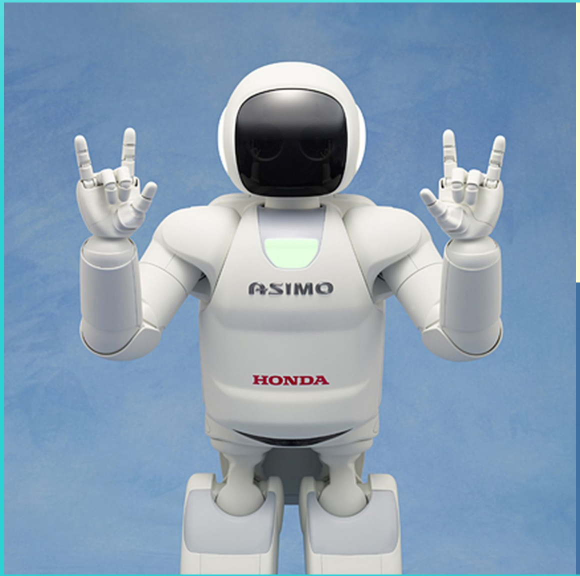
- **Are there any prerequisites for the workshops?**

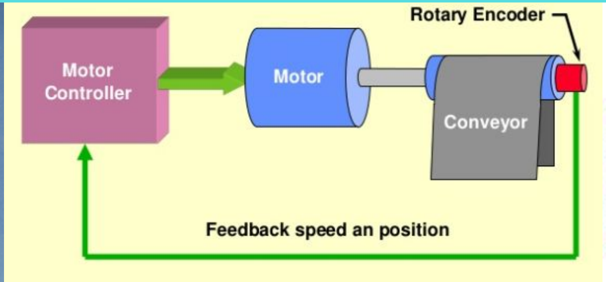
No, we tried to make them approachable to everyone. If you have any issues with the content, please ask!

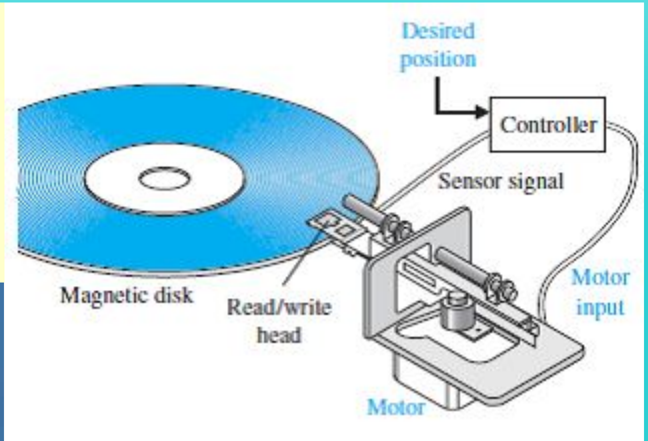
Motivation (II):

- **Where can we find control systems?**

Everywhere!

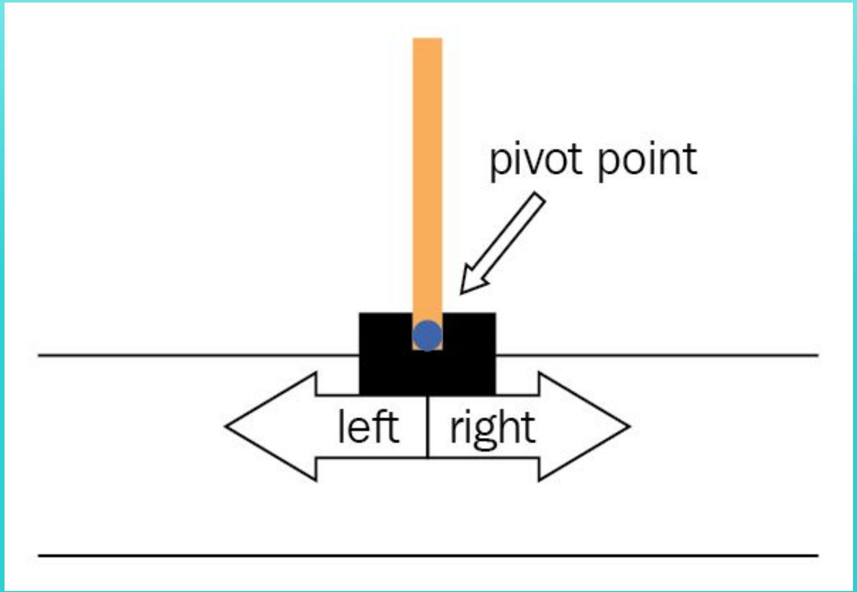






Why a Cartpole Model?

- Cartpole control is of special relevance in robotics: This is how we model balancing and walking of bipedal robots (e.g. ASIMO).
- It is a well-researched problem, there are plenty of resources online which are easy to find and explain everything well.
- It allows us to use Open AI Gym Cartpole environment so you can play around and code your own controllers!

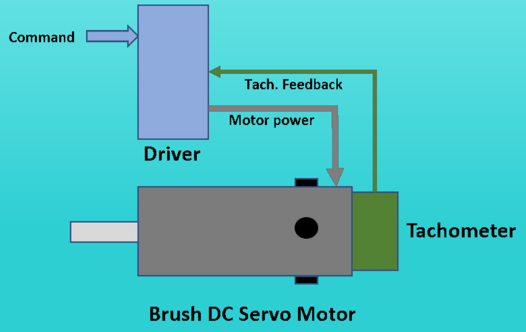


Introduction to PID Control:

How and why PID control works

What is PID Control?

- PID Control stands for proportional, integral, derivative control.
- PID control is **closed loop** control, which means we get **feedback** from the system we are trying to control. Open loop example: toaster.
- We are trying to get a system to the desired goal, which we call a **setpoint,** or reference.
- The difference between the setpoint and our current position is called the **error**, which we are trying to minimise over time.
- Example: Servo motor with an encoder, which has to move 90 degrees clockwise.

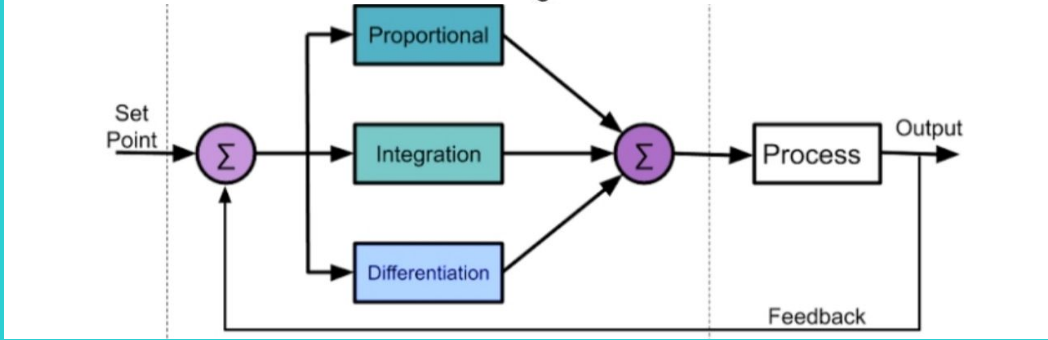


Error equation:
Error(t) = Setpoint - Current position(t)

More formally…

- PID controller consists of 3 terms that each manipulate the error in a different way:
  - Proportional term: The calculated error is just the **current** error.
  - Integral term: The error is the **accumulated** error.
  - Derivative term: The error is the **rate of change** of the error.
- Each of the terms is multiplied by a constant, called **gain**, usually denoted by the letter K.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$



Okay… how do we choose the right gains?

- The easiest way to choose gains is to tune them by hand, but it takes a very long time, and the performance isn’t so great.
- There are some approximate tuning methods we can use (Ziegler-Nichols).
- MATLAB has an amazing tuning tool that is able to design a good set of parameters for the plant you are trying to control.
- However, there is no easy algorithmic way to calculate optimal gains for a PID controller, but it **can be done**.
- If you want to optimally tune a PID controller, you need to know the dynamics of the system you are trying to control!

Today’s task:

- Today’s task is to implement a PID controller that controls the cartpole in the provided Jupyter notebook
- I recommend you use Google Colab, for which you need to sign in with your Google account
- You can access the notebook here: https://github.com/Ptisni/Cartpole-course.git
- If you need any help or have any additional questions, ask any of the instructors.

Happy coding!

Additional Resources:

Classic Control Theory:

- https://www.tutorialspoint.com/control_systems/index.htm
- https://www.youtube.com/watch?v=oBc_BHxw78s&list=PLUMWjy5jgHK1NC52DXXrriwihVrYZKqjk

PID Tuning methods:

- https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method
- https://www.incatools.com/pid-tuning/pid-tuning-methods/