AI Robotics

PID Control

Overview

Robot Model

Sensor Measurements:
$Z_{0:t} = \{z_0, \dots, z_t\}$

Robot State (or pose):
$X_{0:t} = \{X_0, \dots, X_t\}$

sense

act

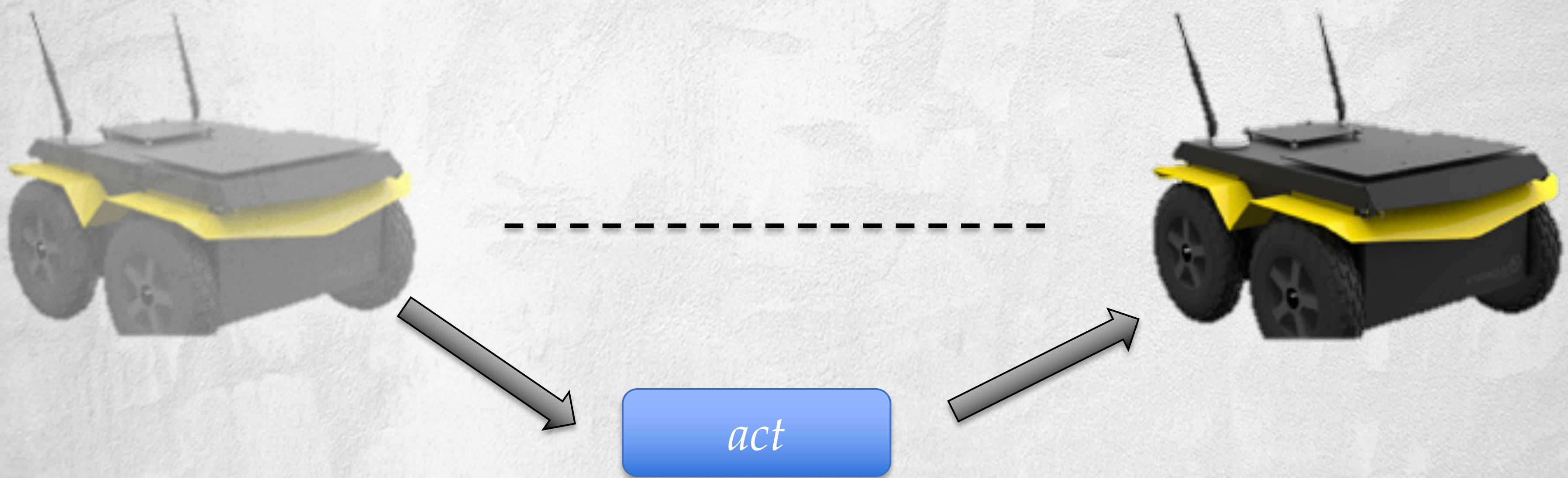
Landmark or Map:
$m = \{m_1, \dots, m_n\}$

Robot Controls:
$u_{0:t} = \{u_0, u_1, \dots, u_t\}$

Control

Robot State (or pose):
$X_{0:t} = \{X_0, \dots, X_t\}$

Control: An input action that causes the robot state to change

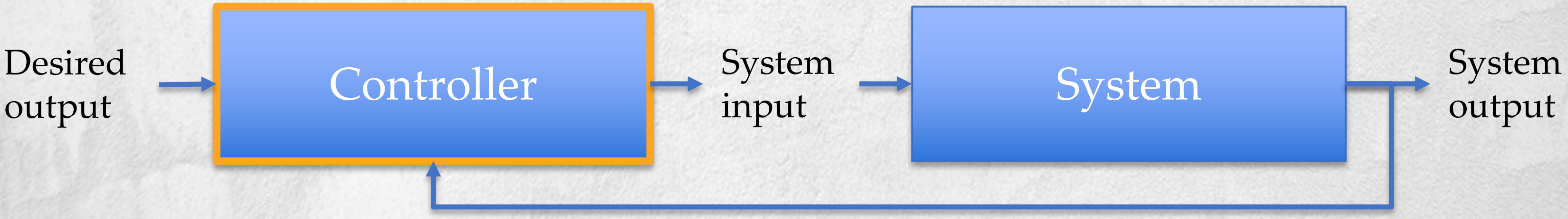


Robot Controls:
$$u_{0:t} = \{u_0, u_1, \dots, u_t\}$$

Control

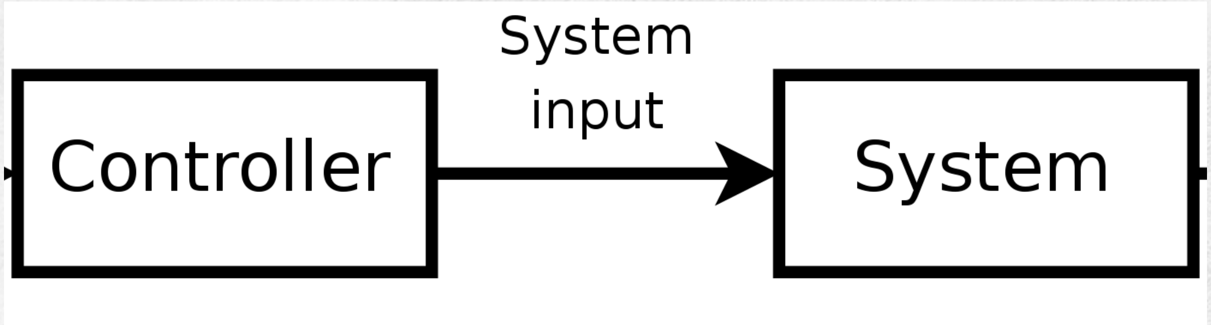
Control

Control is the process of generating the robot system input to change the system output to the desired output

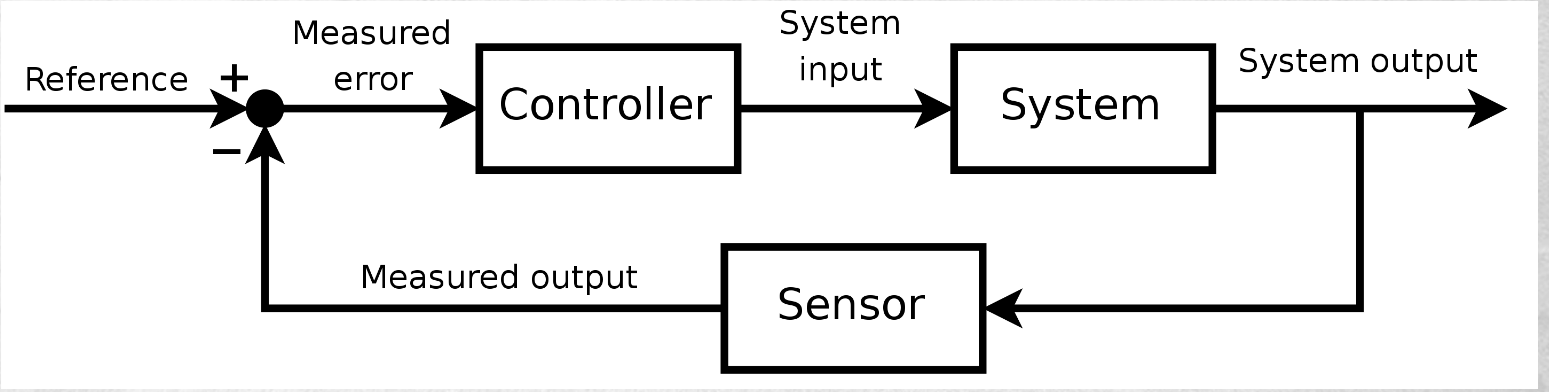


Open Loop vs Closed Loop

Open Loop

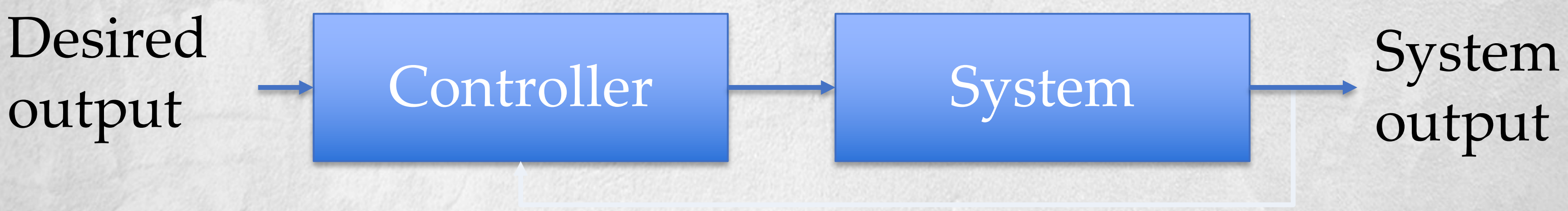


Closed Loop



Open-loop Control

- Open-loop control does NOT use sensors
  - Example
    - Stepper motor
      - Say it has 1 degree / 1 pulse.
      - We want to rotate it by 90 degrees.
      - Generate 90 pulses

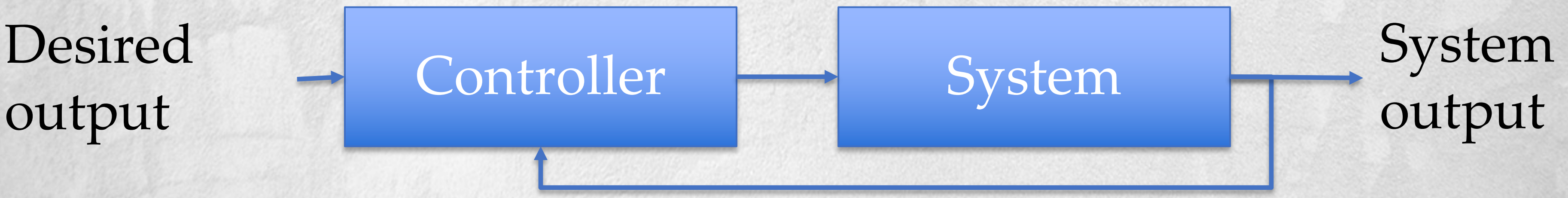


Open-Loop Control: Problems

- What if the system *does not* do what we expect?
  - Motor going out-of-sync
  - Ground conditions
  - Battery drains over time
  - Random disturbances

Closed-loop Control

- Closed-loop control uses feedback from sensors
  - E.g. temperature control, driving a car, …
    - Driving a car to cruise at 50 mph.
    - Current speed 40 mph → push the pedal more to accelerate
    - Current speed 60 mph → brake/release the pedal to decelerate



Closed-loop Control

- Let's define the error e = desired - current.
- Control should minimize |*e*|.

PID Control

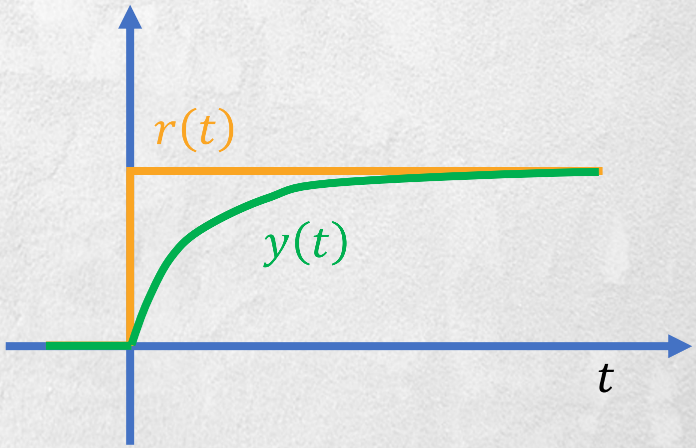
PID Control

- **P**roportional-**I**ntegral-**D**erivative control
  - Control input $= K_P \cdot e + K_I \cdot \int e\, dt + K_D \cdot \frac{de}{dt}$

- **Proportional** term: depends on <u>error</u>
- **Integral** term: depends on <u>sum</u> of errors
- **Derivative** term: depends on <u>rate of change</u> of error

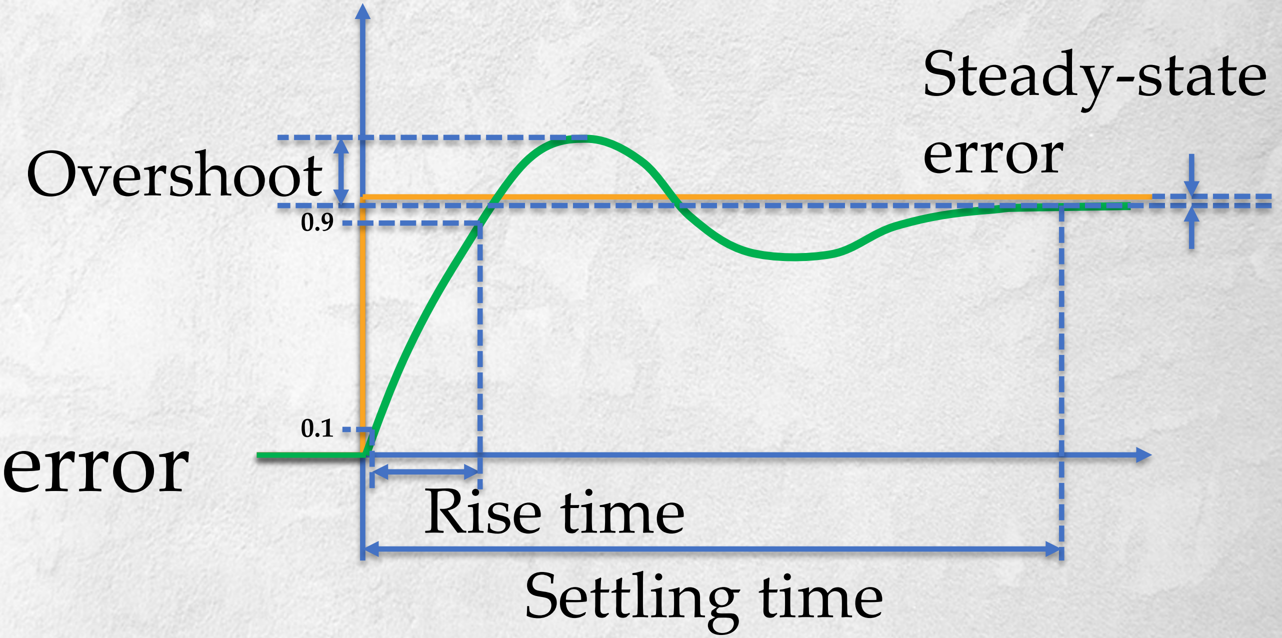
Step Response

- Used to compare the performance of different controllers
- Desired output: $r(t)$
- System output: $y(t)$



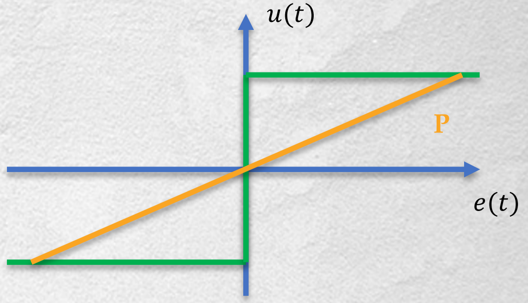
Performance Measures

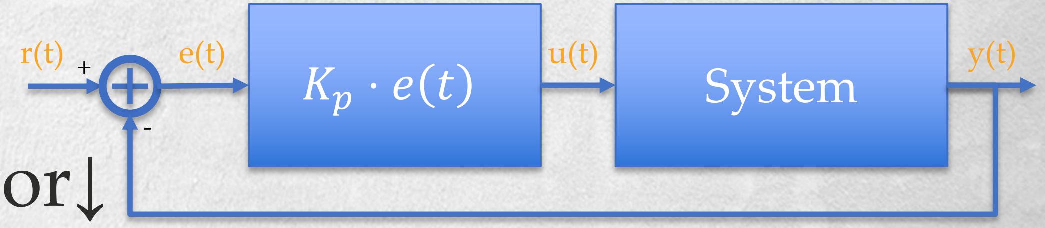
- Rise time
- Overshoot
- Settling time
- Steady-state error



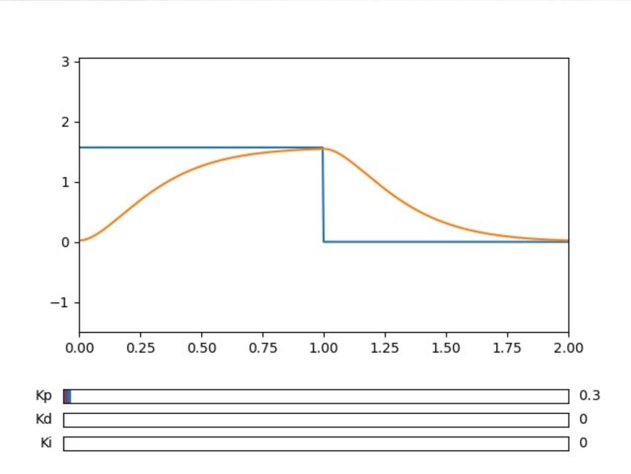
Proportional (P) controller

- Controller input: proportional to error
  - $u(t) = K_p \cdot e(t)$
- Effect of increasing K_p
  - Rise time ↓
  - Overshoot ↑
  - Settling time ~
  - Steady-state error↓

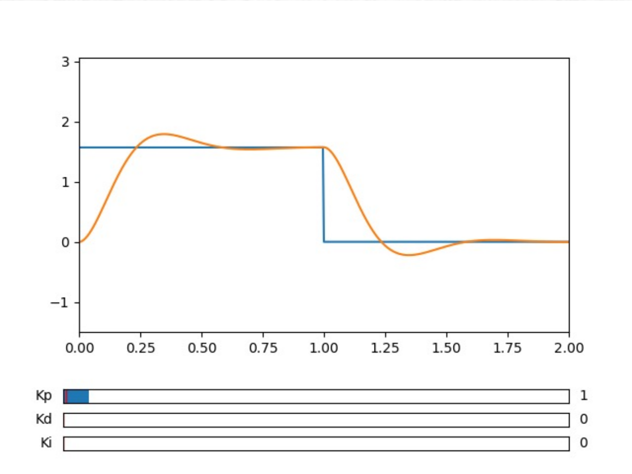




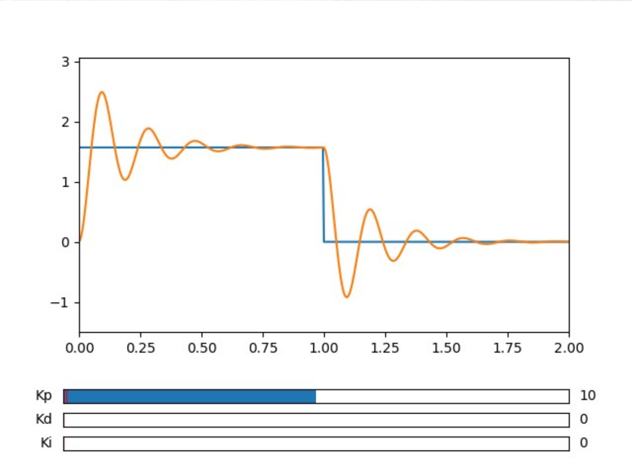
Proportional (P) controller



$K_p = 0.3$

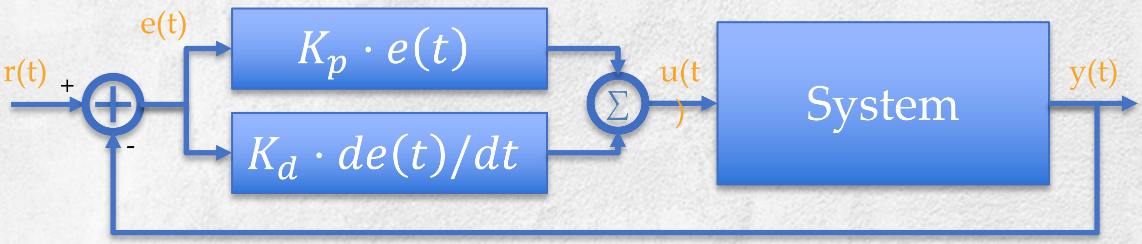


$K_p = 1.0$



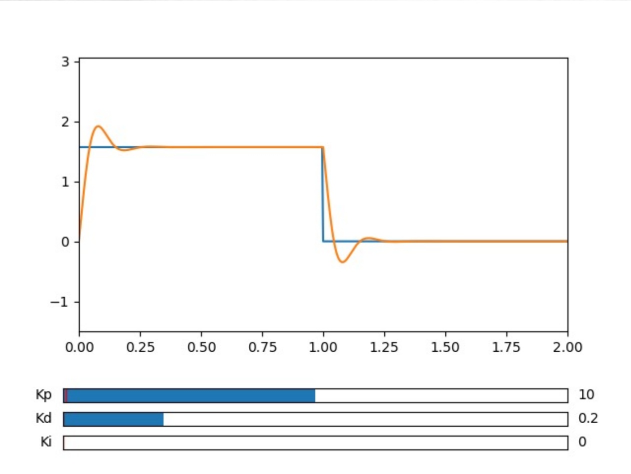
$K_p = 10$

Proportional-Derivative (PD) Controller

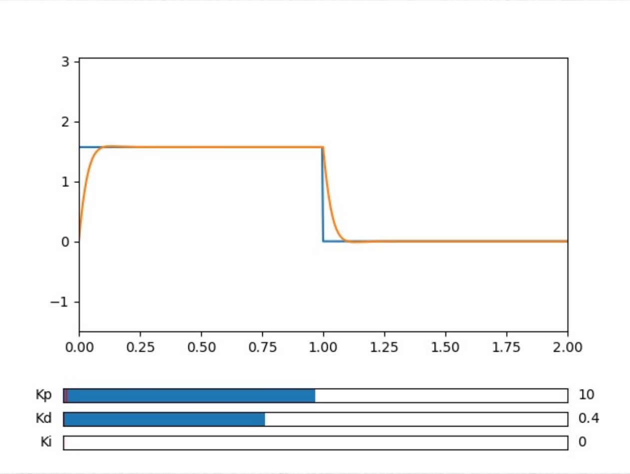


- Effect of increasing K_d
  - Rise time ~
  - Overshoot ↓
  - Settling time ↓
  - Steady-state error =

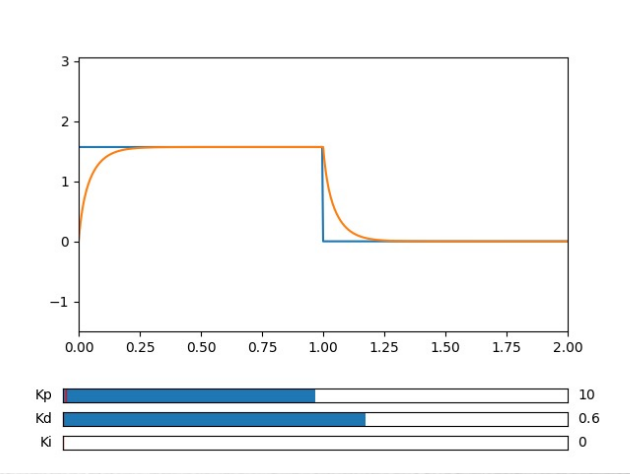
Proportional-Derivative (PD) Controller



$K_p = 10, K_d = 0.2$



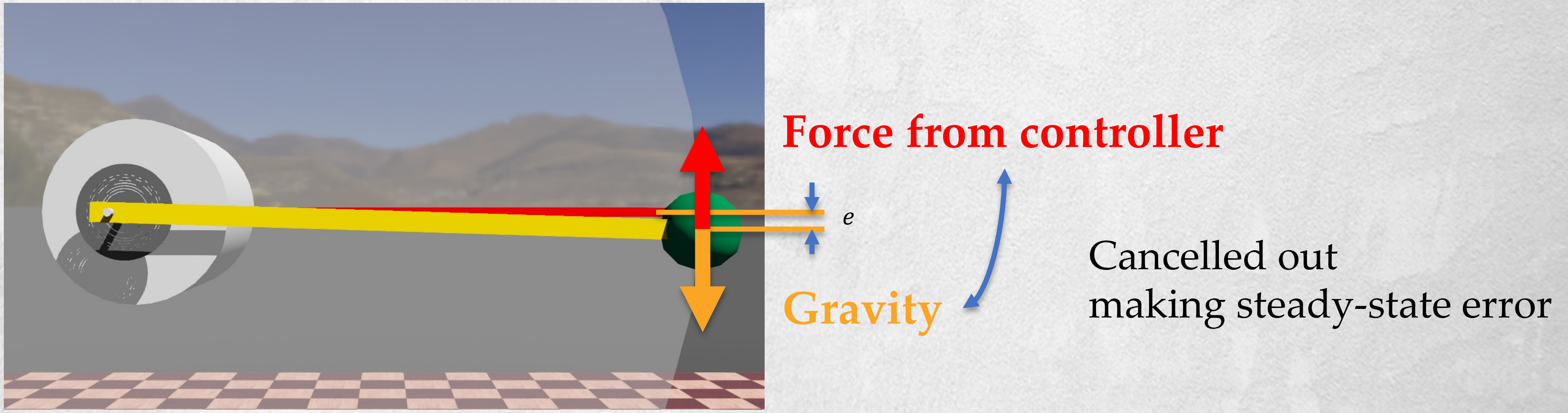
$K_p = 10, K_d = 0.4$



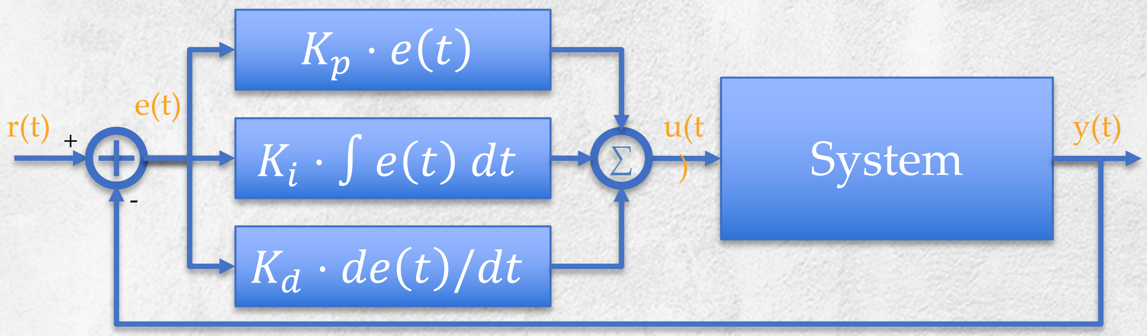
$K_p = 10, K_d = 0.6$

Proportional Integral Derivative (PID) Controller

- Friction / gravity / external load
- Steady-state error means that $\frac{de(t)}{dt} = 0$.

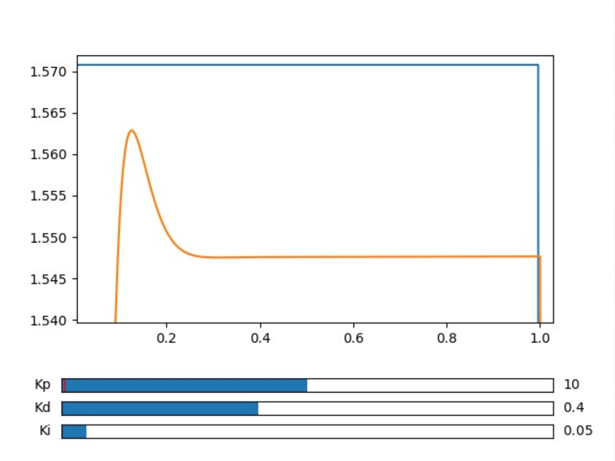


Proportional Integral Derivative (PID) Controller

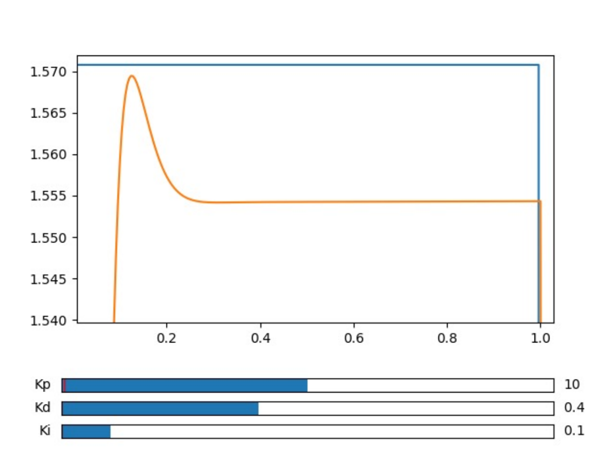


- Effect of increasing K_i
  - Rise time ↓
  - Overshoot ↑
  - Settling time ↑
  - Steady-state error ↓

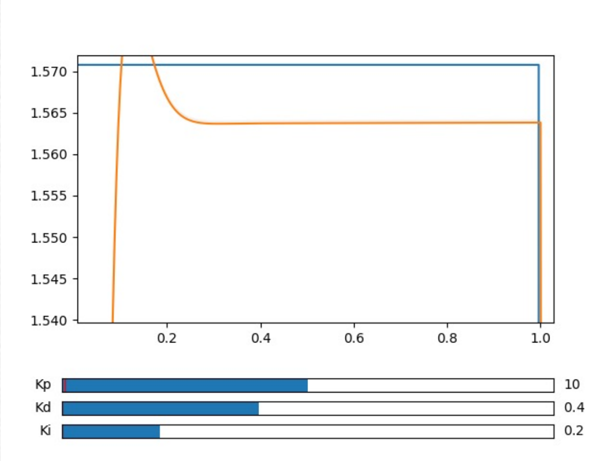
Proportional Integral Derivative (PID) Controller



$K_p = 10, K_d = 0.4, K_i = 0.05$



$K_p = 10, K_d = 0.4, K_i = 0.10$



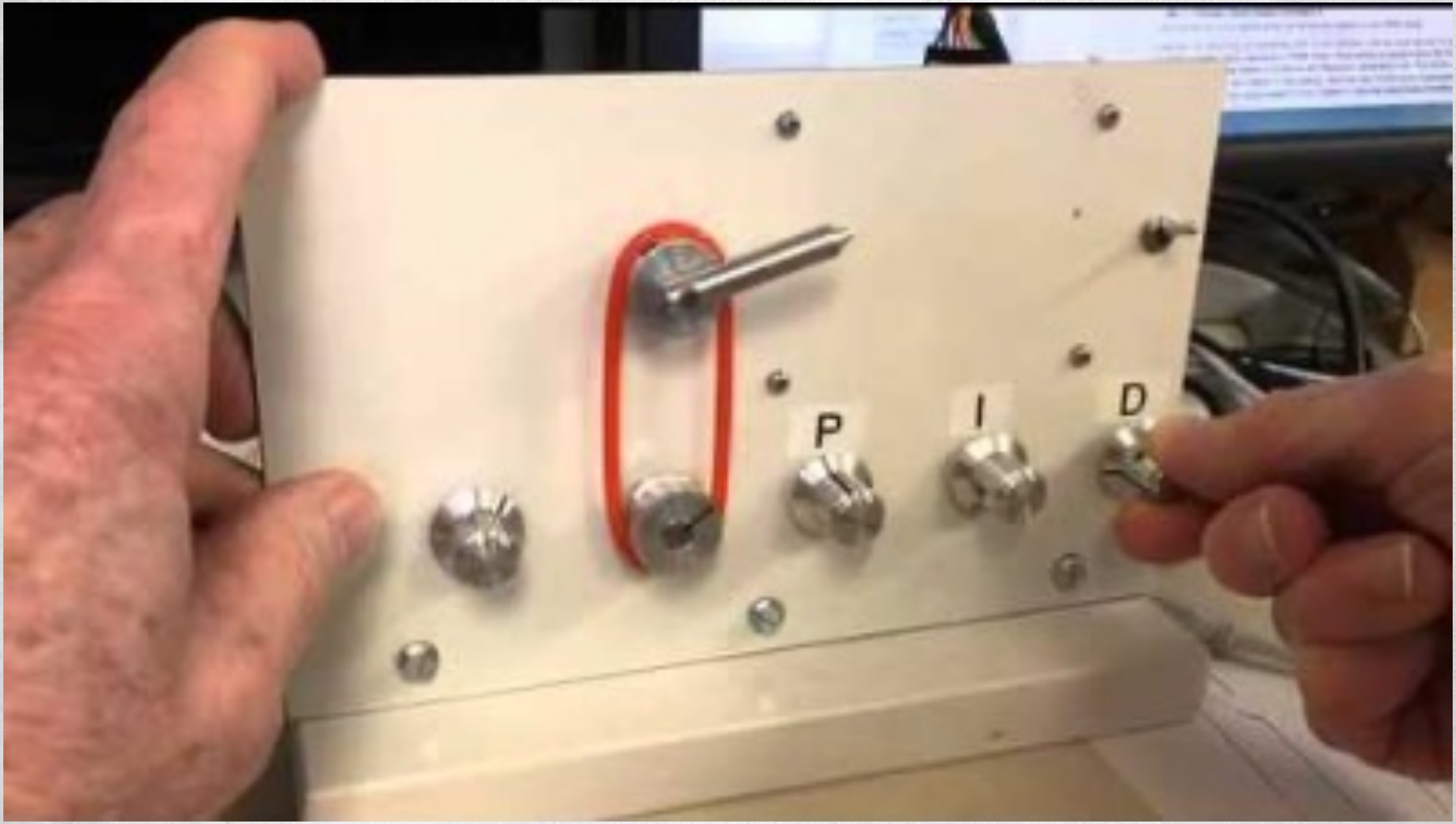
$K_p = 10, K_d = 0.4, K_i = 0.20$

Summary

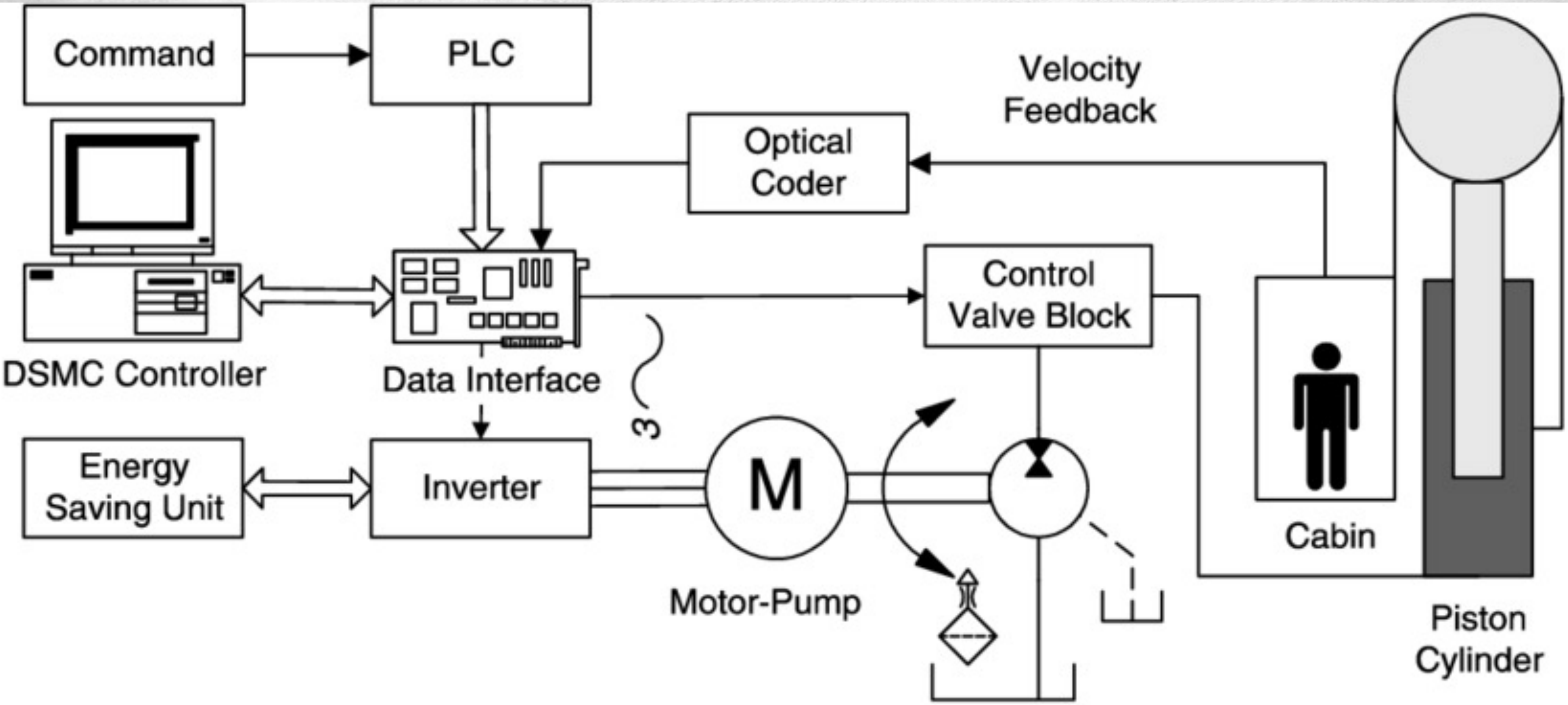
- PID controller
  - $u(t) = K_p \cdot e(t) +$ (**P**roportional)
    $K_i \cdot \int e(t)\, dt +$ (**I**ntegral)
    $K_d \cdot \frac{de(t)}{dt}$ (**D**erivative)
- Parameters
  - K_p: Rise time ↓, overshoot ↑
  - K_d: Overshoot ↓
  - K_i: Steady-state error → 0

Examples

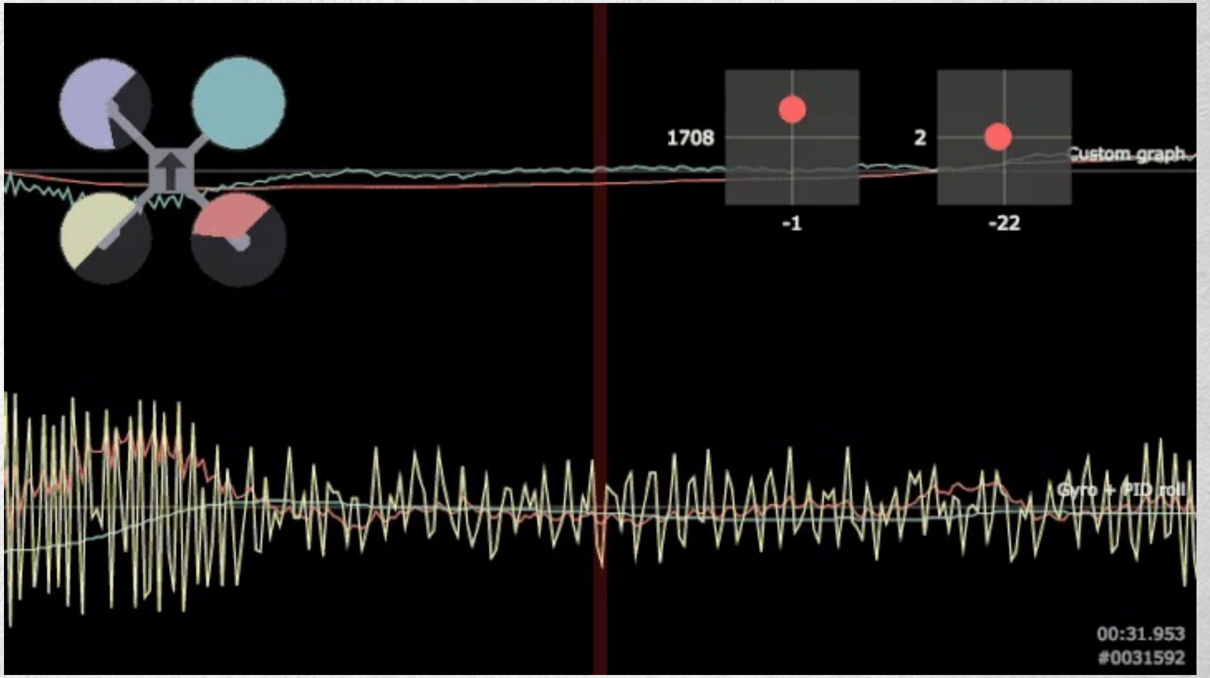
PID Tuning



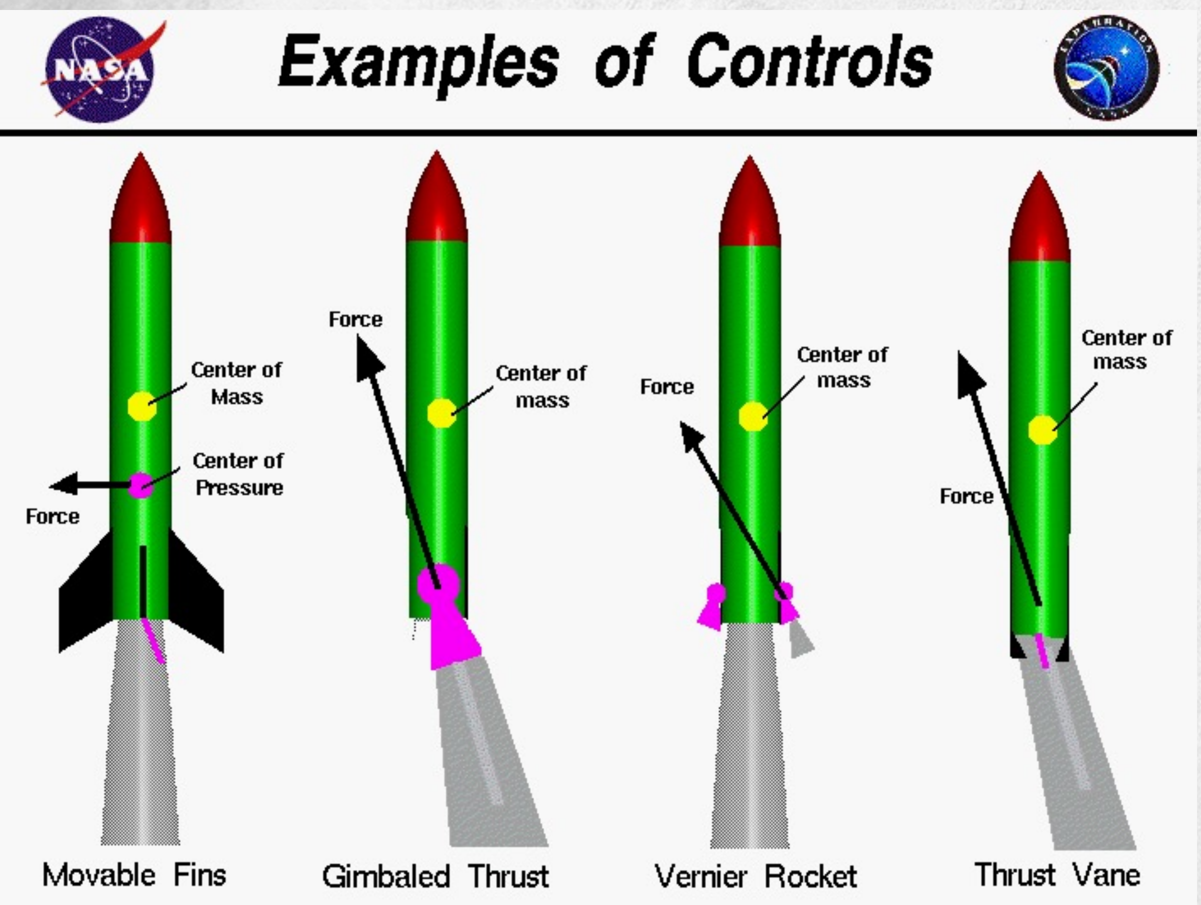
Elevator Control

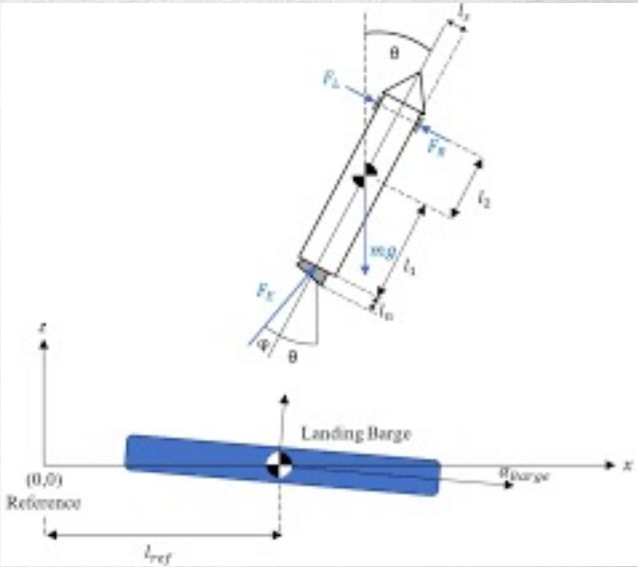


Drone Stabilization



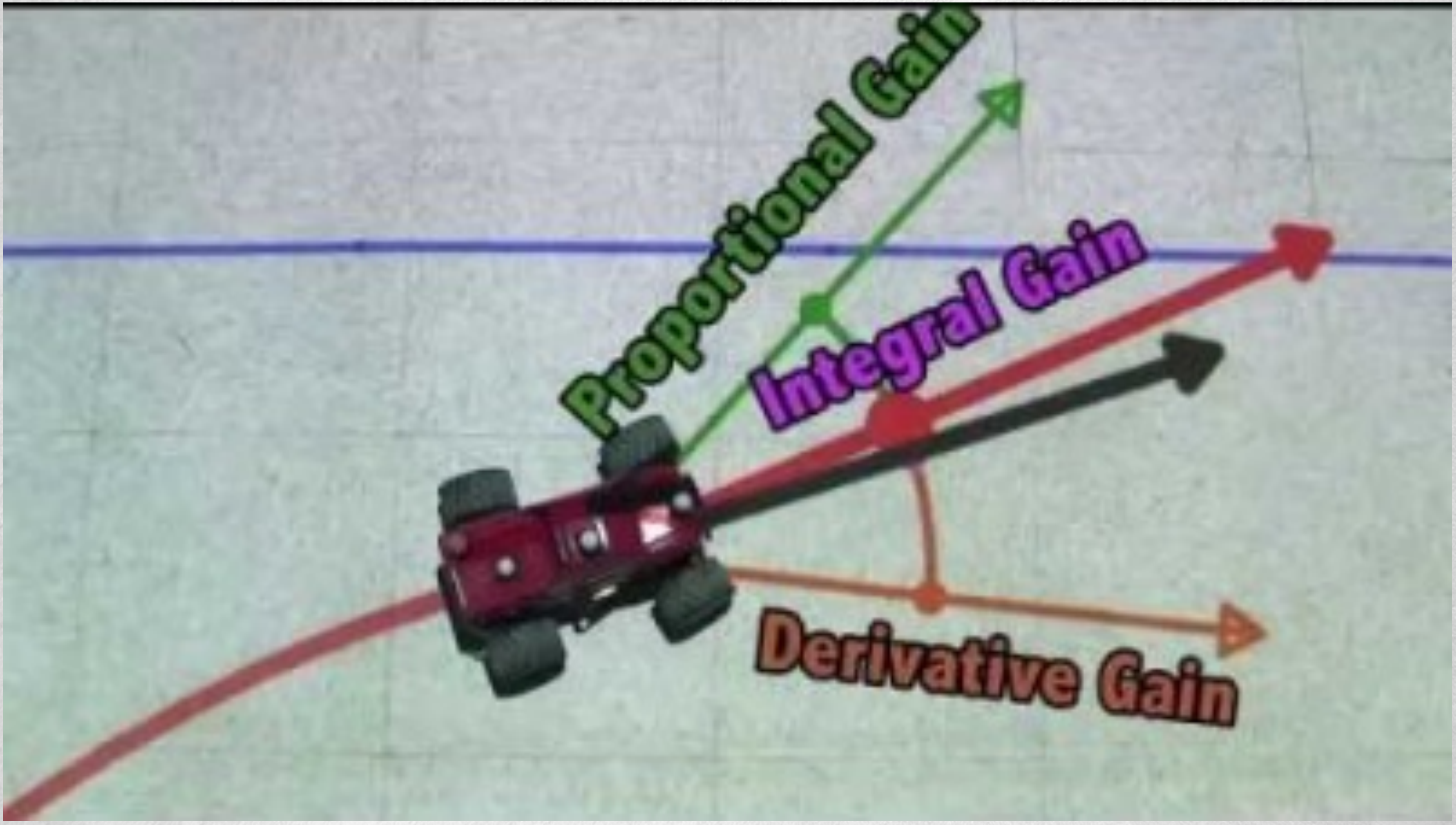
Rocket Control





Rocket Control

Lane Keeping



References

1. https://en.wikipedia.org/wiki/Control_theory
2. https://www.youtube.com/watch?v=qKy98Cbcltw
3. https://youtu.be/HxXKJXRHthk?t=288
4. https://www.youtube.com/watch?v=4Y7zG48uHRo