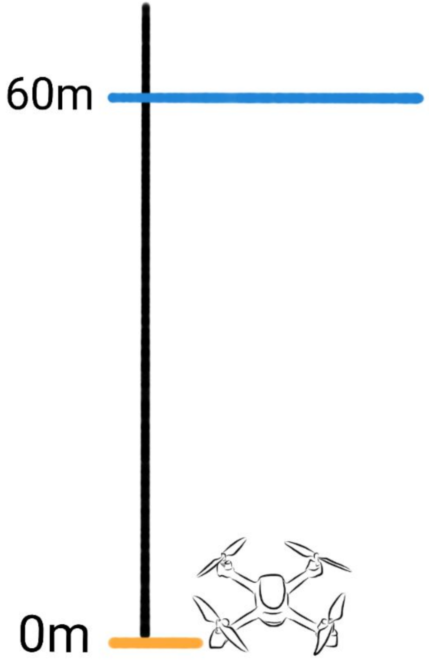
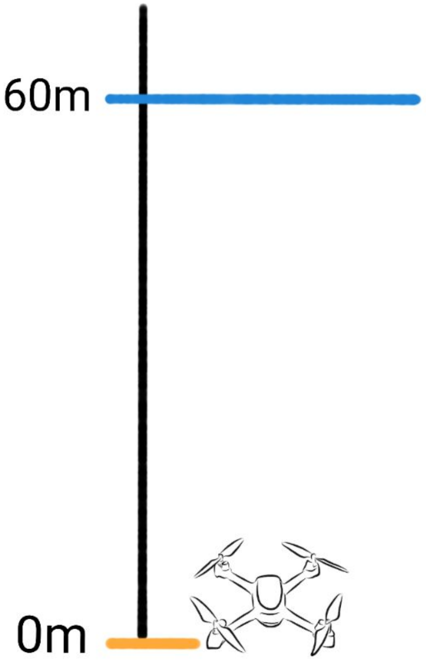
Introduction to Robotics
CSE 461

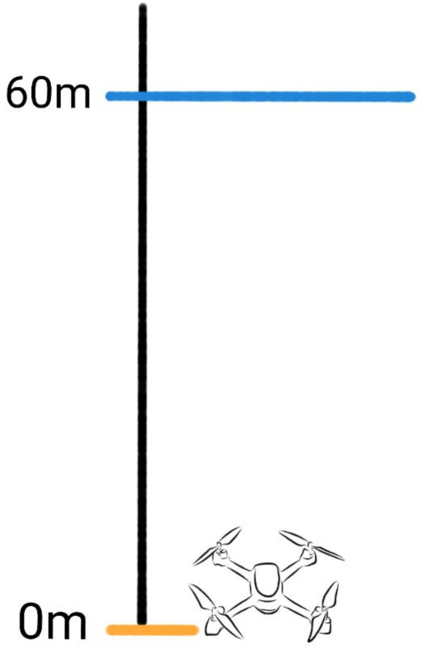
Class Topic: Introduction to Control System Theory (PID)

Riad Ahmed
Lecturer
Brac University


60m
0m

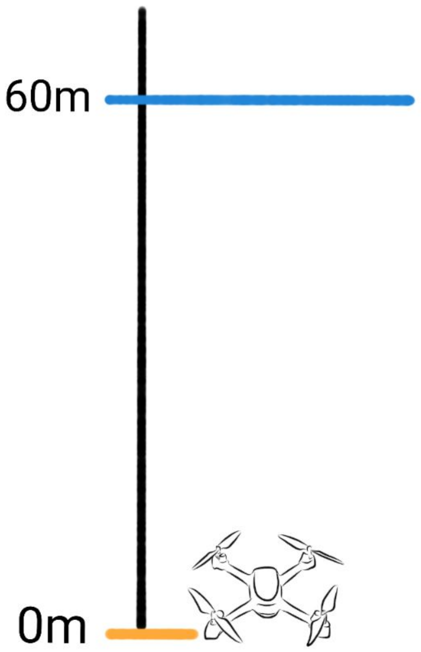


Desired Altitude = 60m
Actual Altitude = 0m



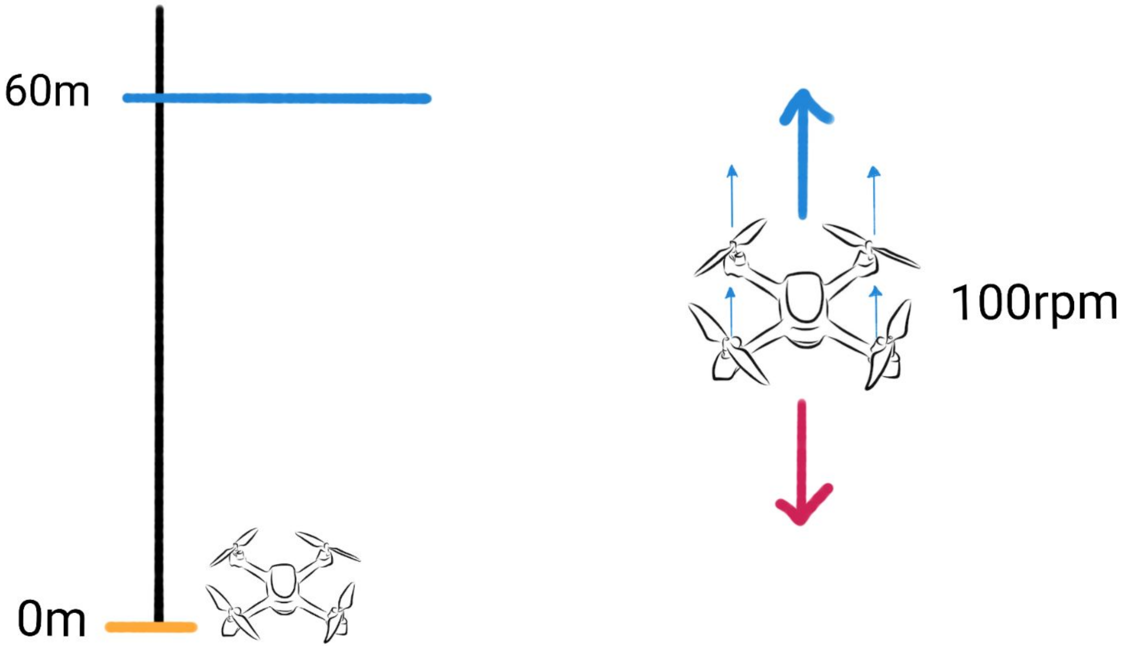
Desired Altitude = 60m
Actual Altitude = 0m

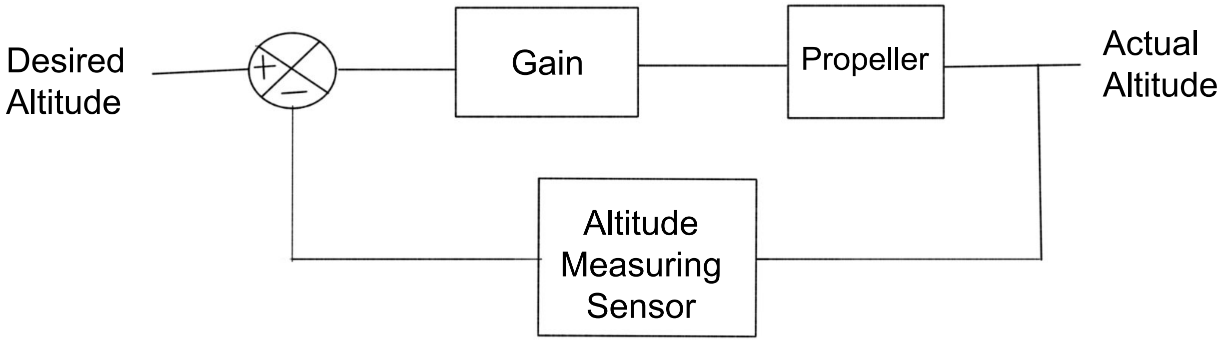
Desired Altitude - Actual Altitude
= 60m - 0m
= 60

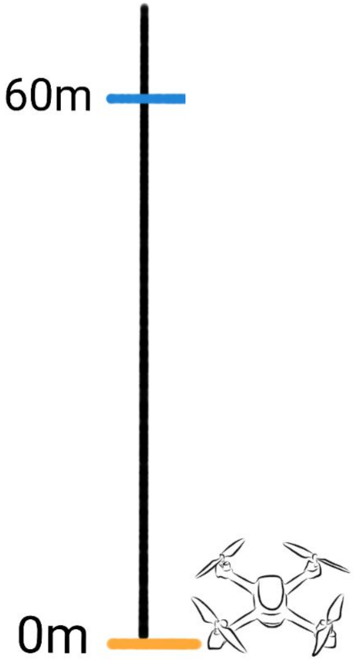


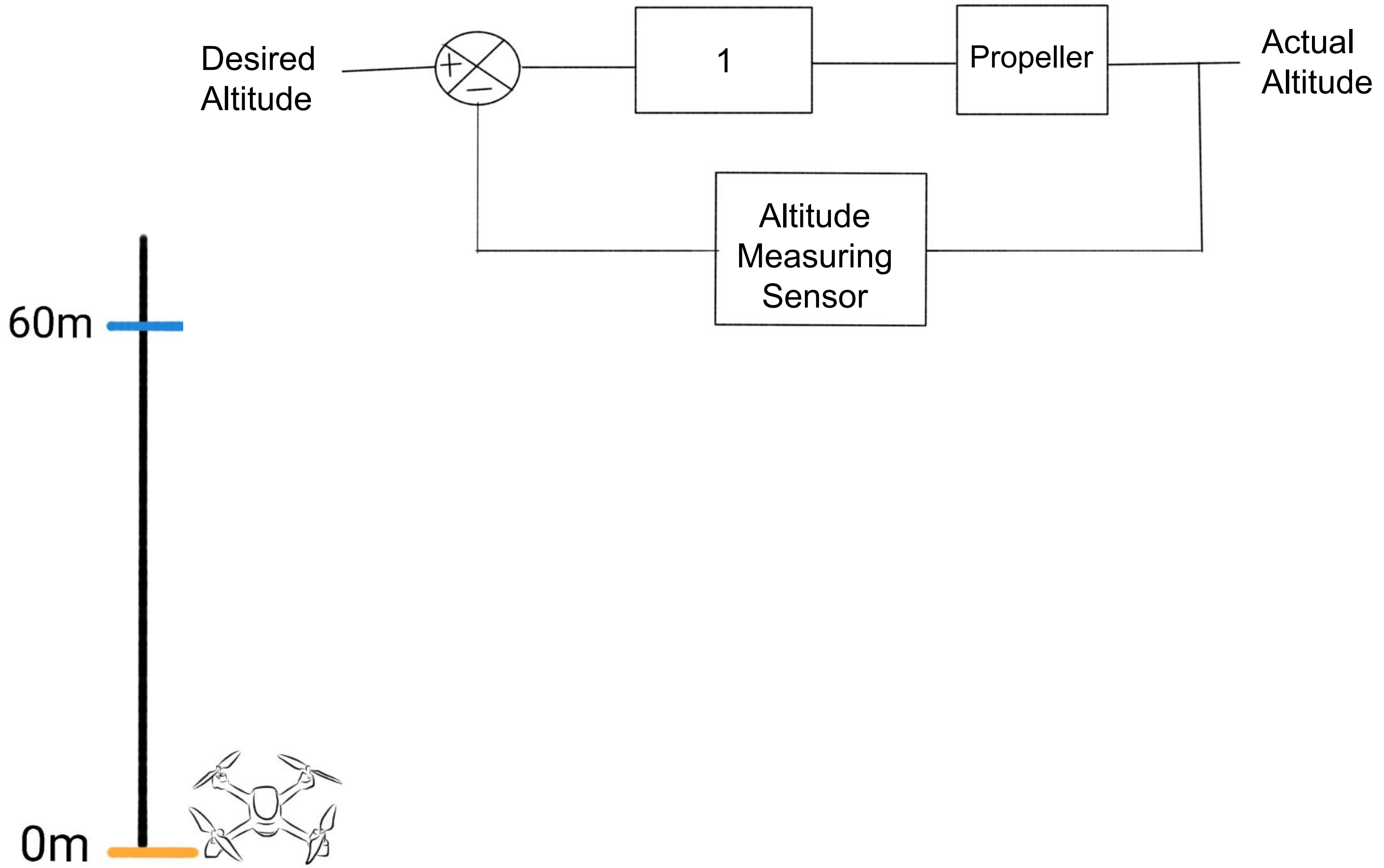
Desired Altitude = 60m
Actual Altitude = 0m

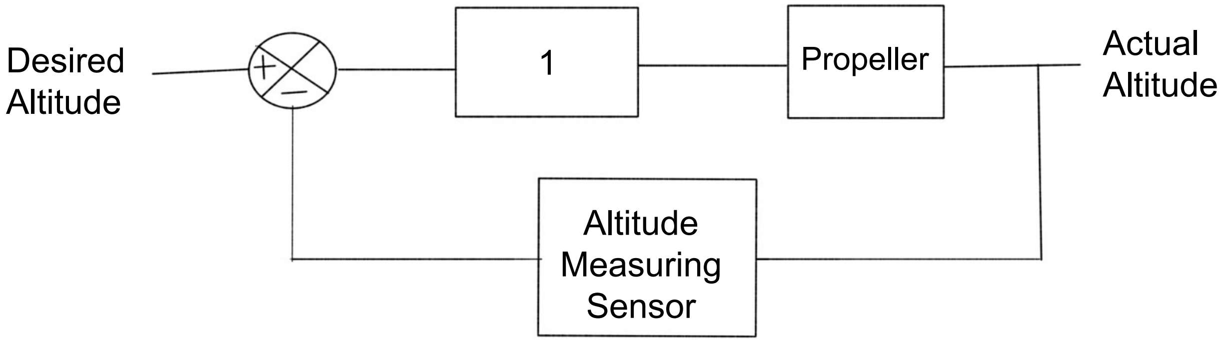
Desired Altitude - Actual Altitude
= 60m - 0m
= 60
= Error

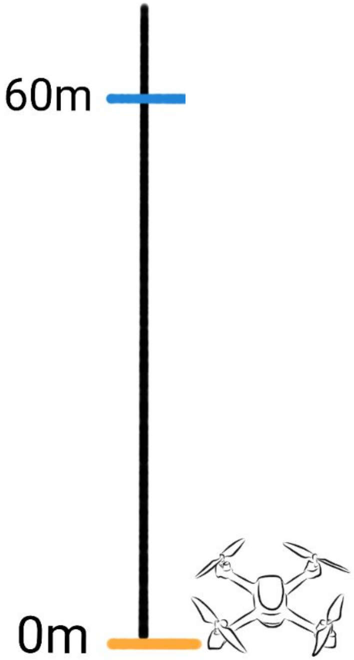

60m
0m
100rpm


Desired Altitude
+
−
Gain
Propeller
Actual Altitude
Altitude Measuring Sensor


60m
0m

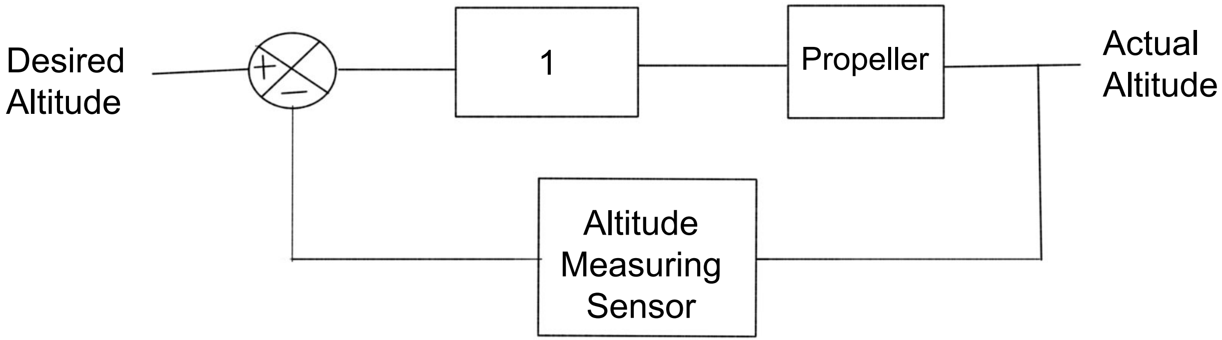

Desired Altitude
+
−
1
Propeller
Actual Altitude
Altitude Measuring Sensor
60m
0m





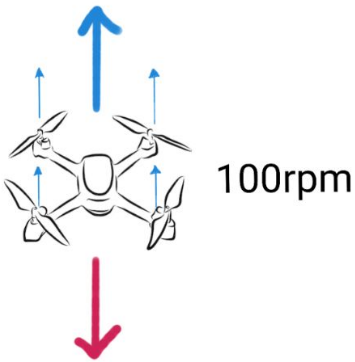
Desired Altitude - Actual Altitude
= 60m - 0m
= 60

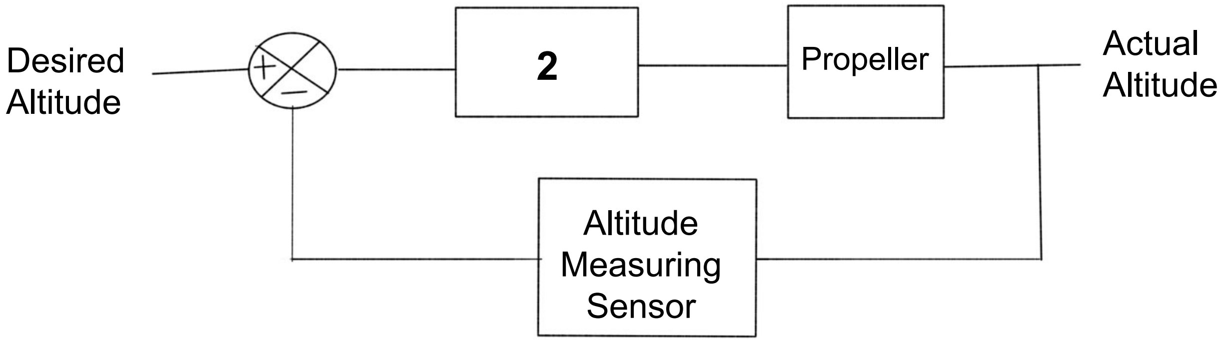
Propeller Speed = 60 * 1 = 60 rpm



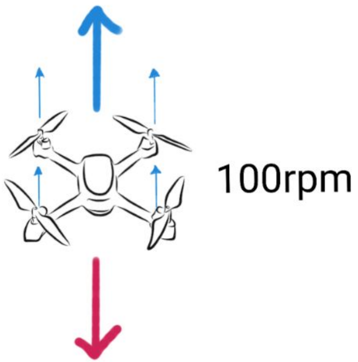
Desired Altitude - Actual Altitude
= 60m - 0m
= 60

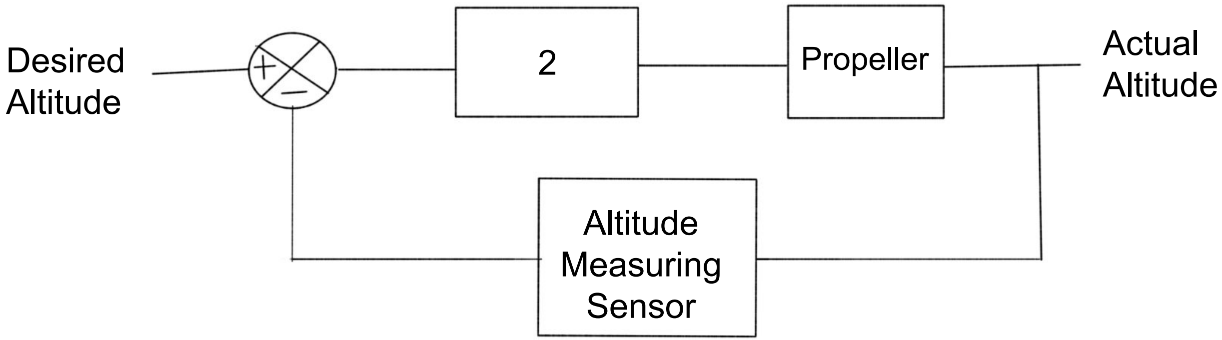
Propeller Speed = 60 * 1 = 60 rpm

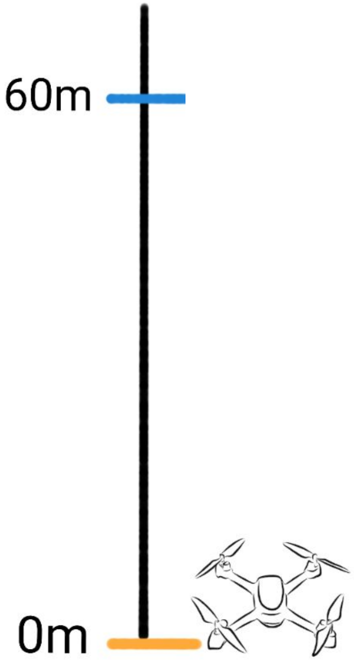



Desired Altitude
+
−
2
Propeller
Actual Altitude
Altitude Measuring Sensor


60m
0m

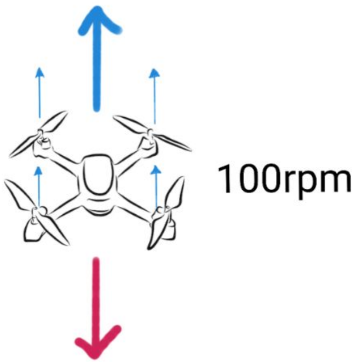

100rpm

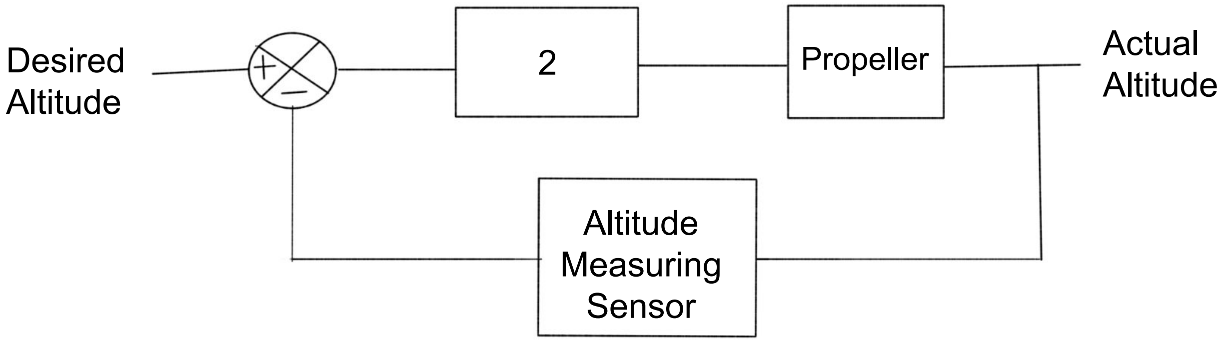


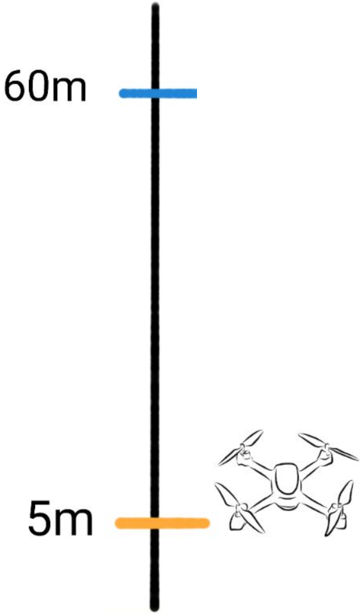


Desired Altitude - Actual Altitude
= 60m - 0m
= 60

Propeller Speed = 60 * 2 = 120 rpm

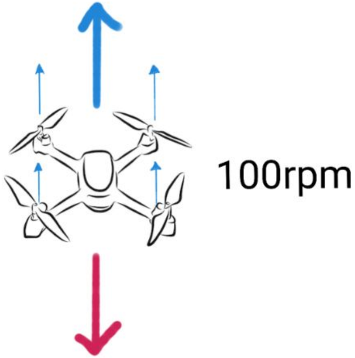


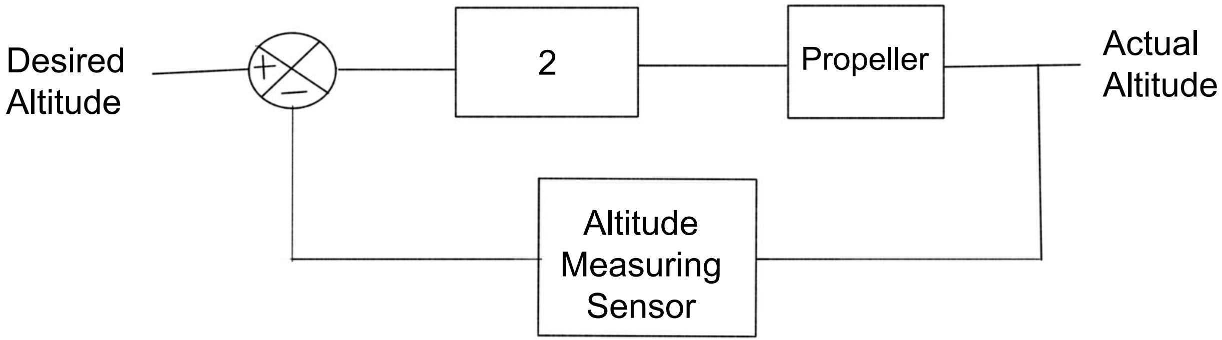


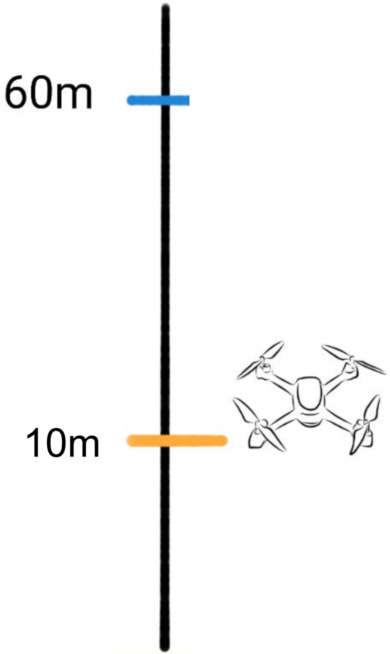


Desired Altitude - Actual Altitude
= 60m - 5m
= 55

Propeller Speed = 55 * 2 = 110 rpm

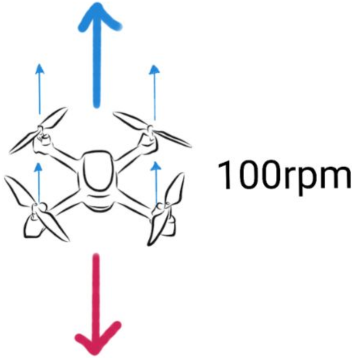


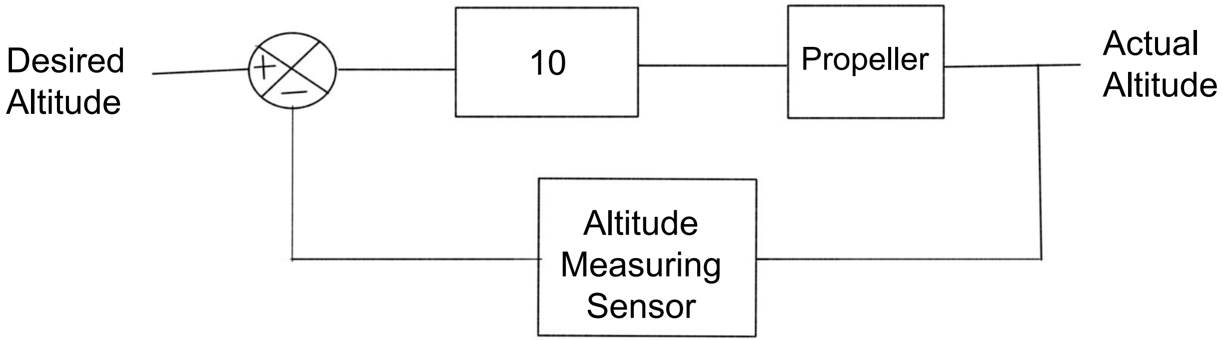


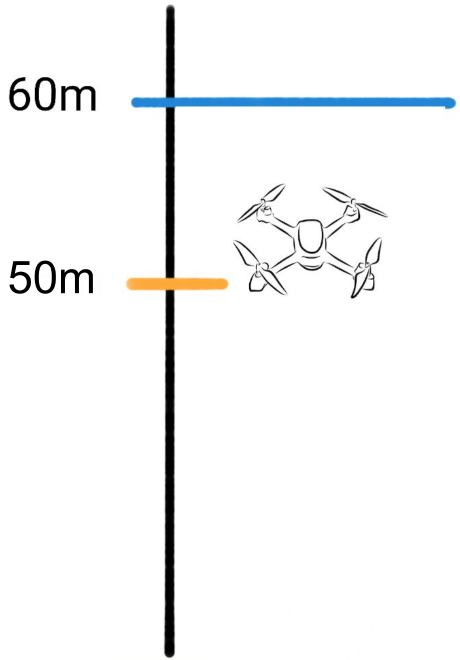


Desired Altitude - Actual Altitude
= 60m - 10m
= 50

Propeller Speed = 50 * 2 = 100 rpm

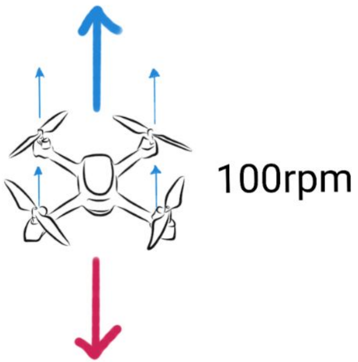


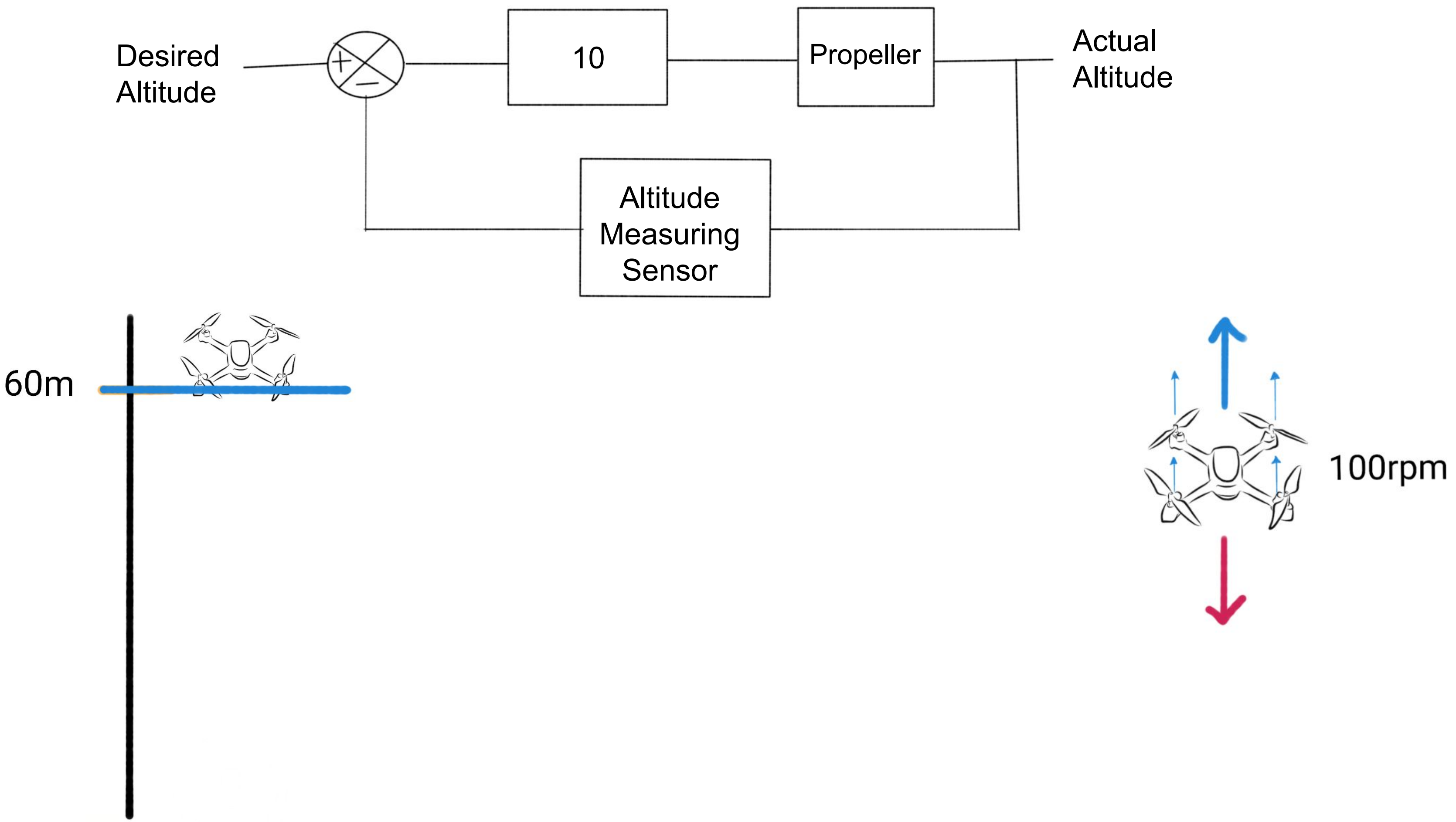


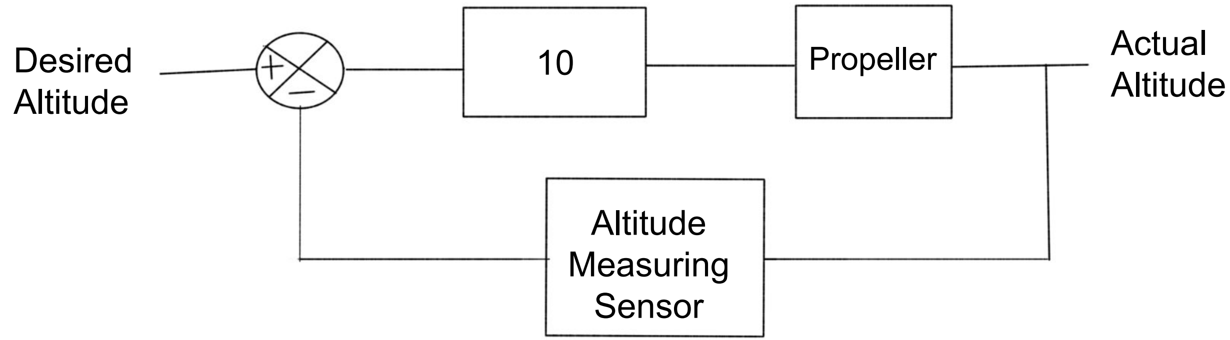


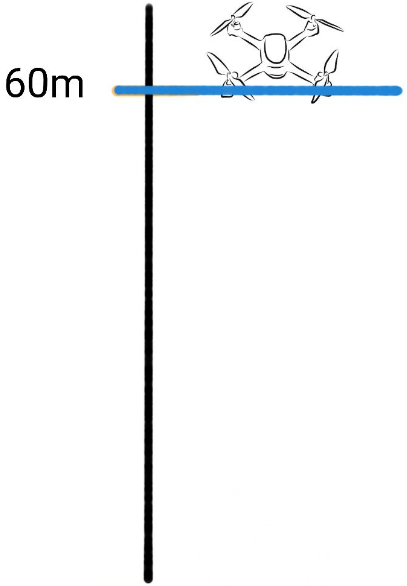
Propeller Speed = 10 * 10 = 100 rpm

Error = 10



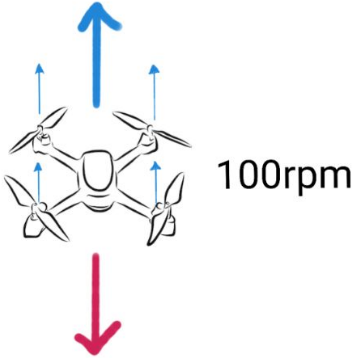

Desired Altitude
+
−
10
Propeller
Actual Altitude
Altitude Measuring Sensor
60m
100rpm

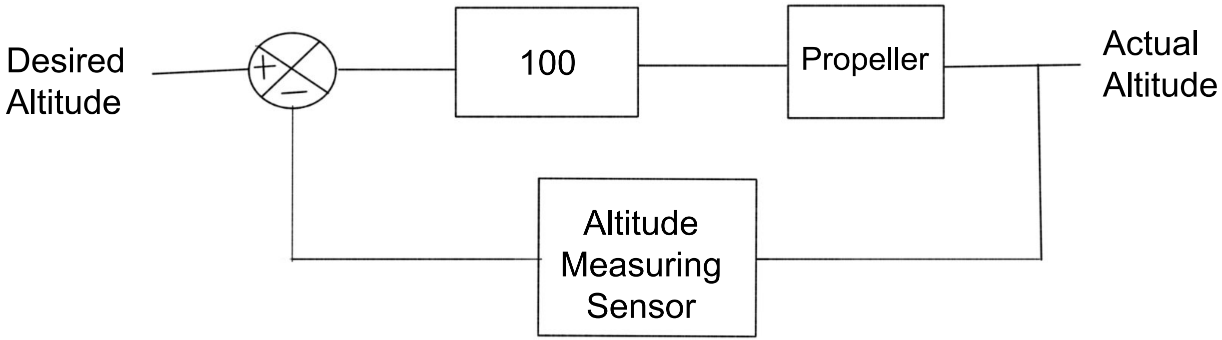


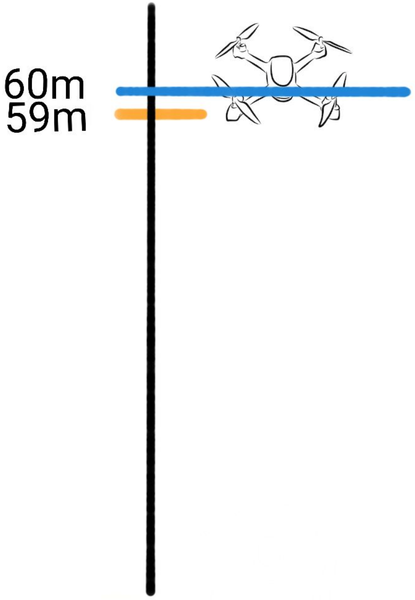


Desired Altitude - Actual Altitude
= 60m - 60m
= 0

Propeller Speed = 0 * 10 = 0 rpm

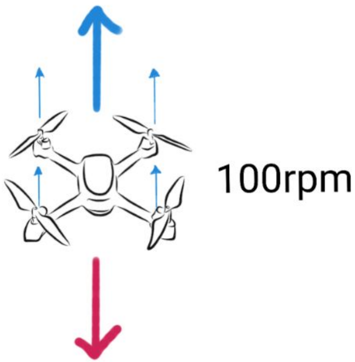


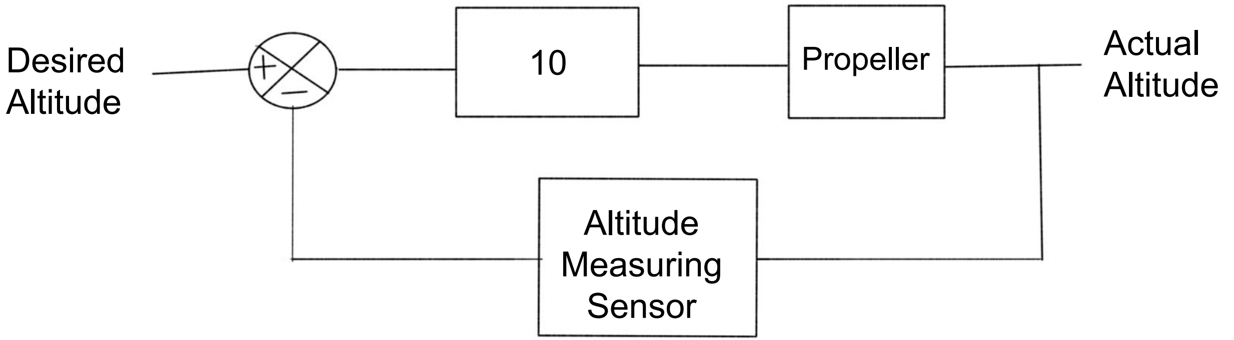


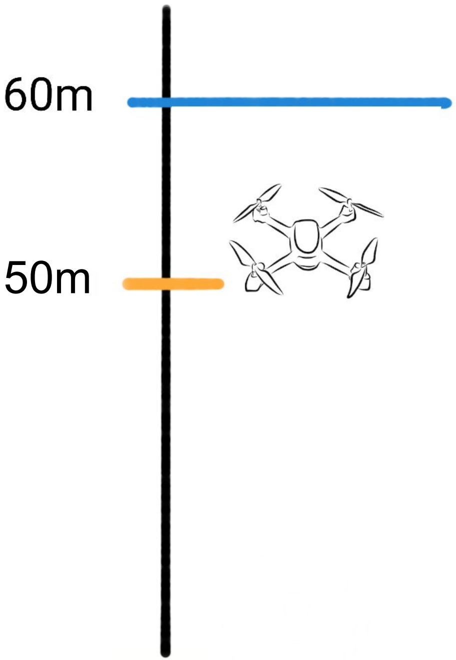


Desired Altitude - Actual Altitude
= 60m - 59m
= 1

Propeller Speed = 1 *100 = 100 rpm

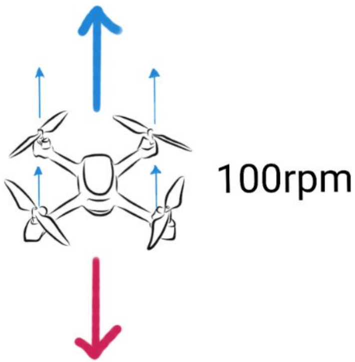




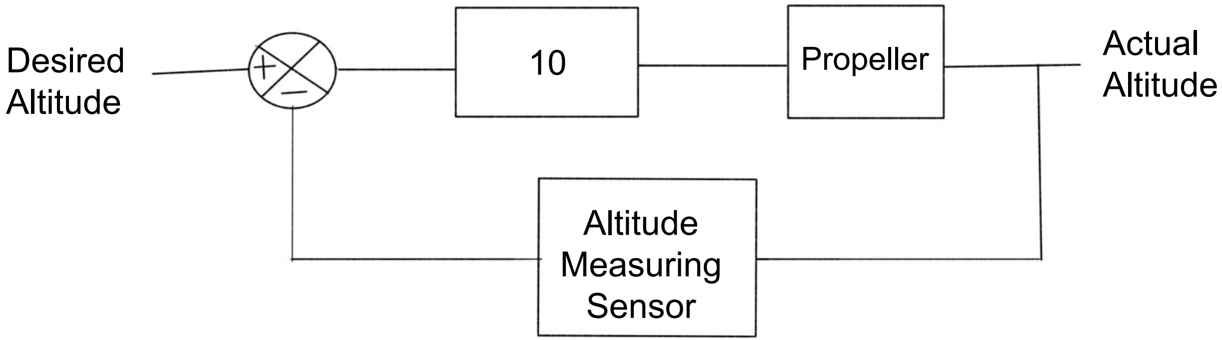


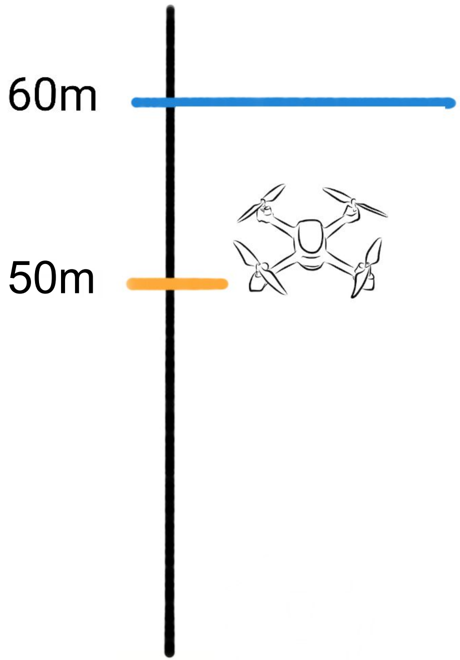
Propeller Speed = 10 * 10 = 100 rpm

Error = 10



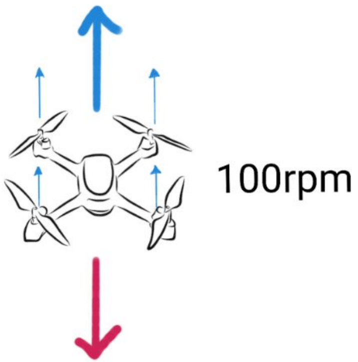
Output = Kp * error





Propeller Speed = 10 * 10 = 100 rpm

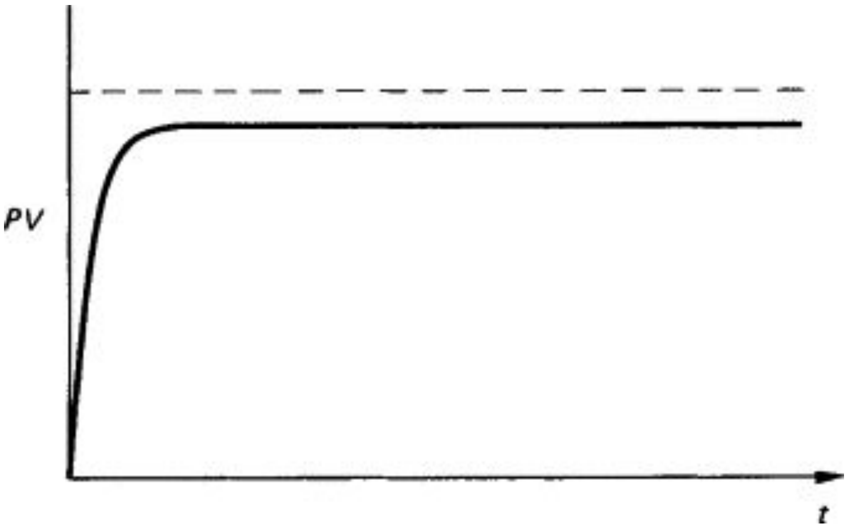
Error = 10

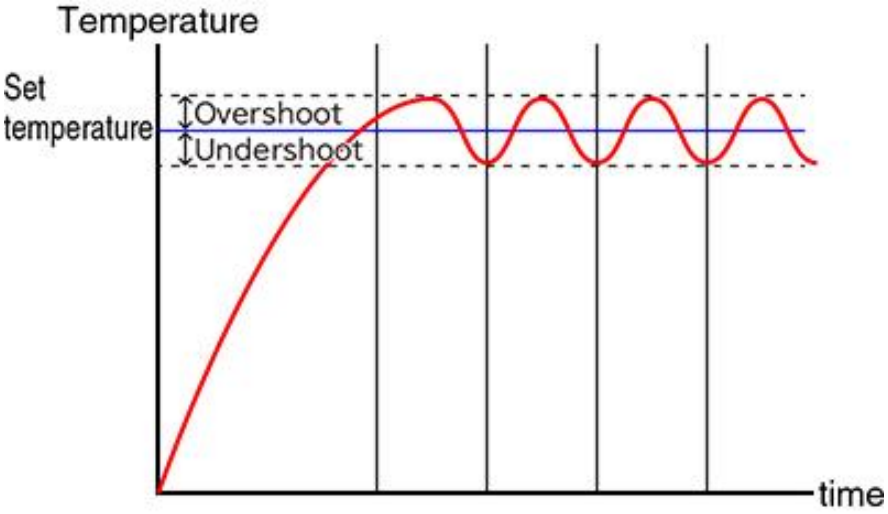


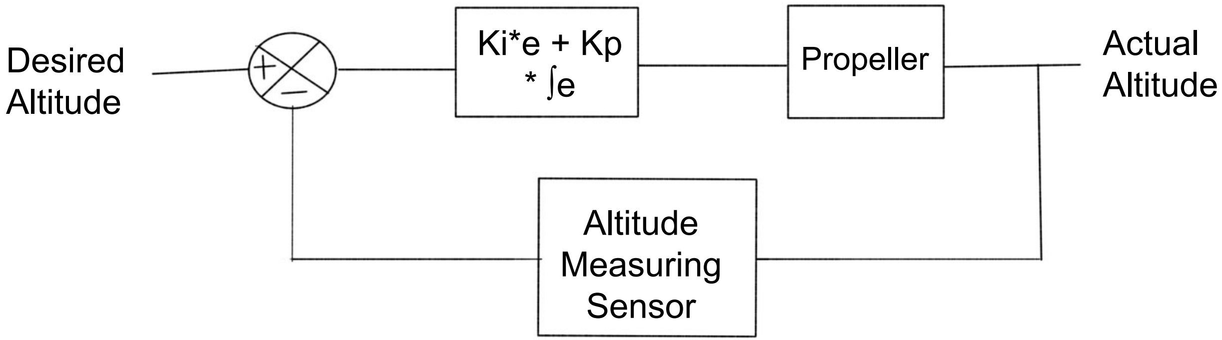
Output = Kp * error

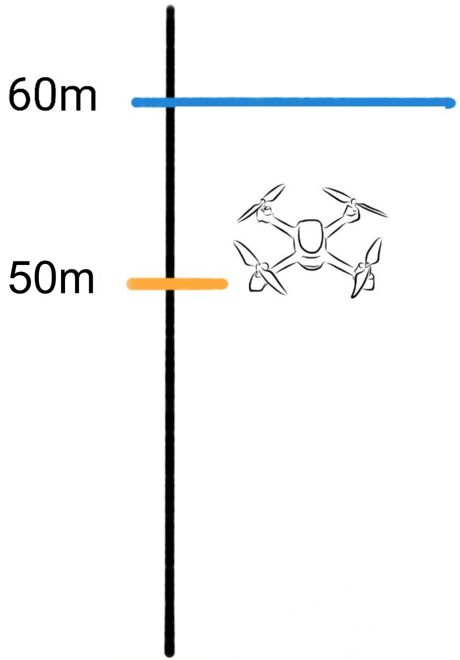
Proportional Control

Proportional Control



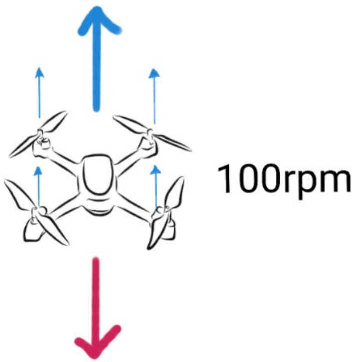


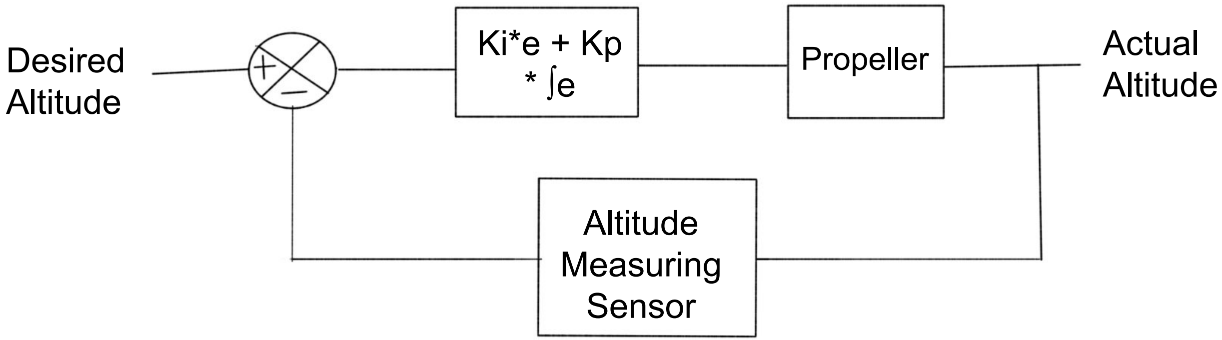


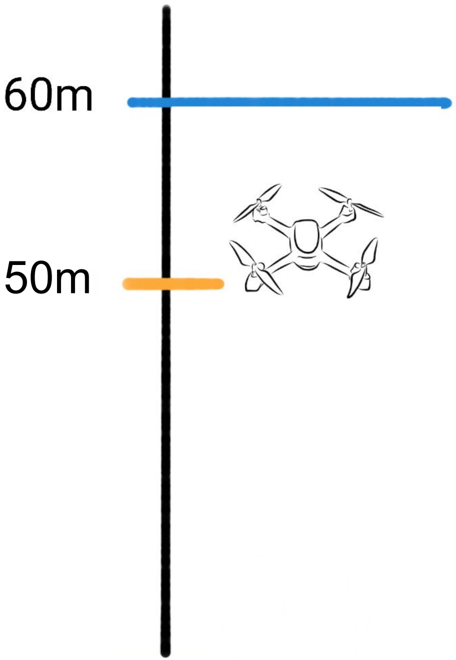


Propeller Speed = 10 * 10 = 100 rpm

Error = 10

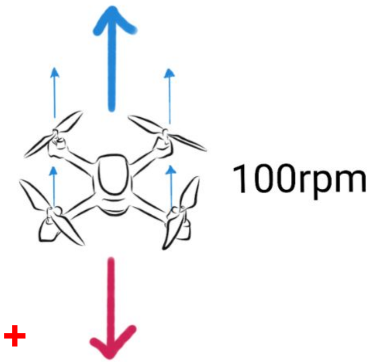




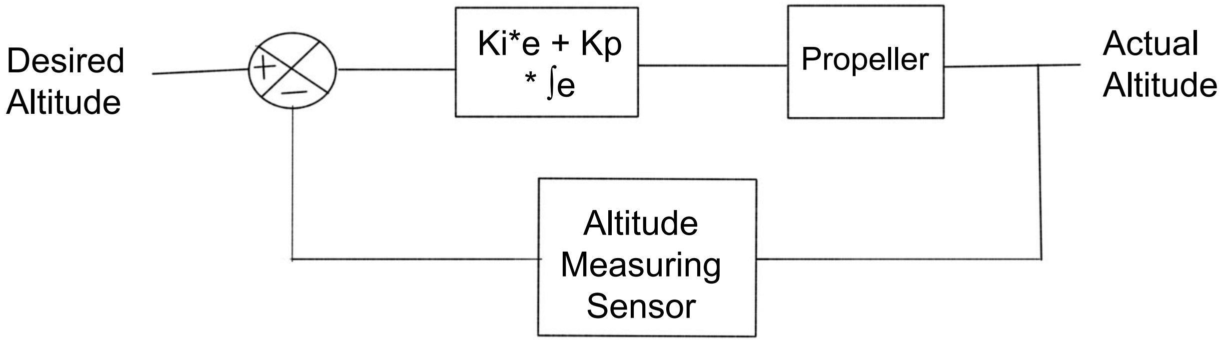


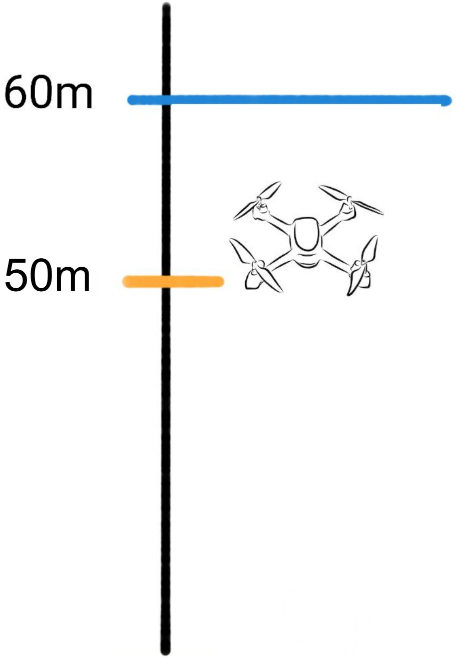
Propeller Speed = 10 * 10 = 100 rpm

Error = 10



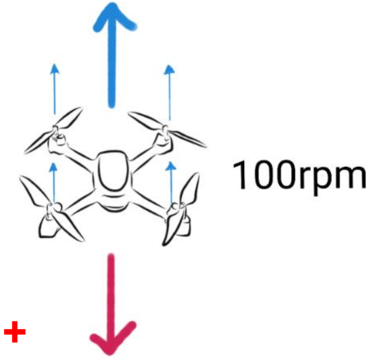
Output = Kp * error + Ki * (error1 + error2 + error3)





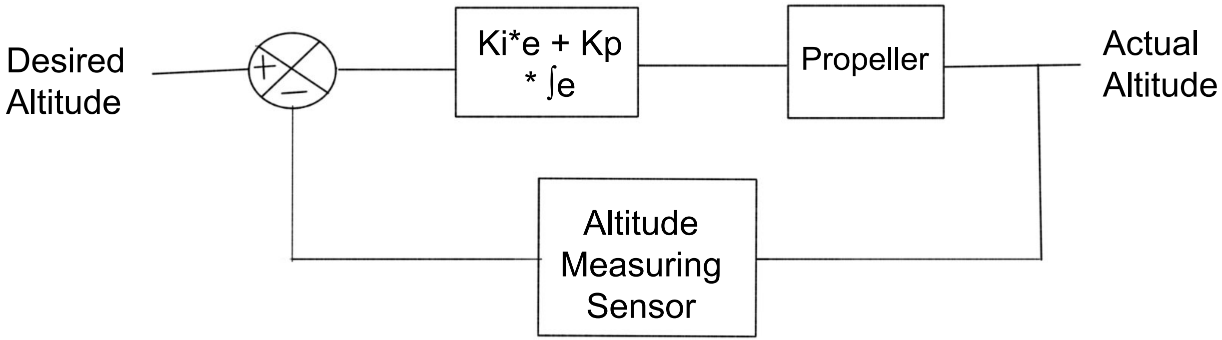
Propeller Speed = 10 * 10 = 100 rpm

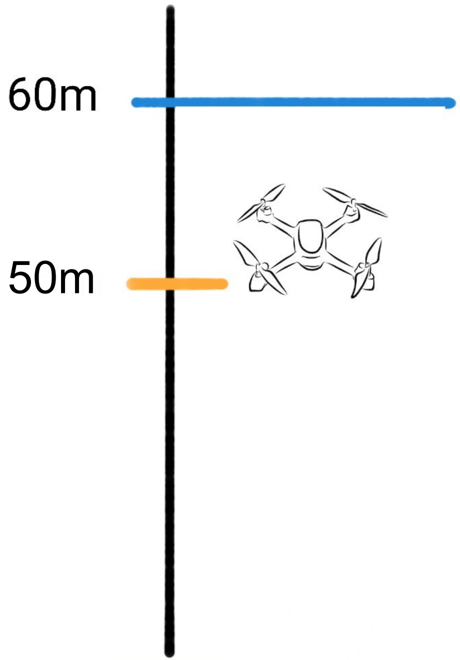
Error = 10

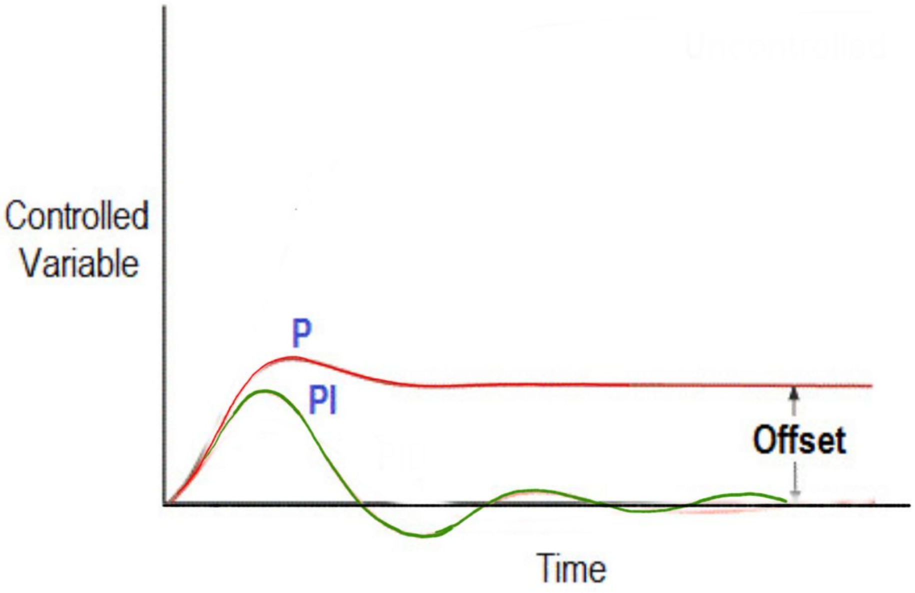


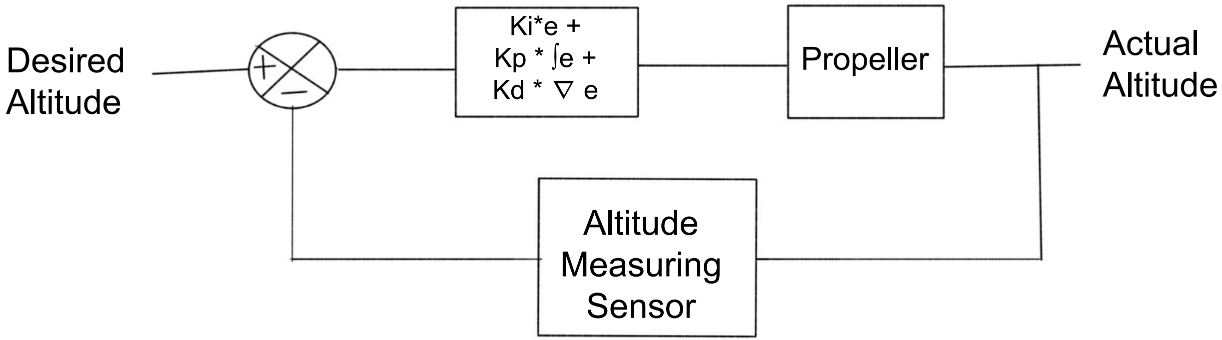
Output = Kp * error + Ki * (error1 + error2 + error3)

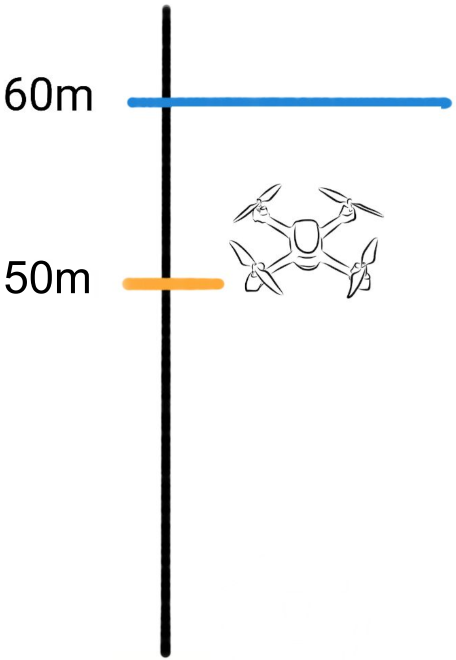
Proportional - Integral Control


Desired Altitude
+
−
Ki*e + Kp * ∫e
Propeller
Actual Altitude
Altitude Measuring Sensor


60m
50m

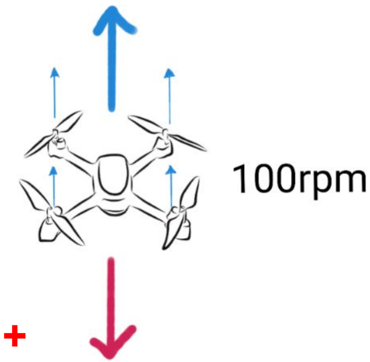

Controlled Variable
P
PI
Offset
Time



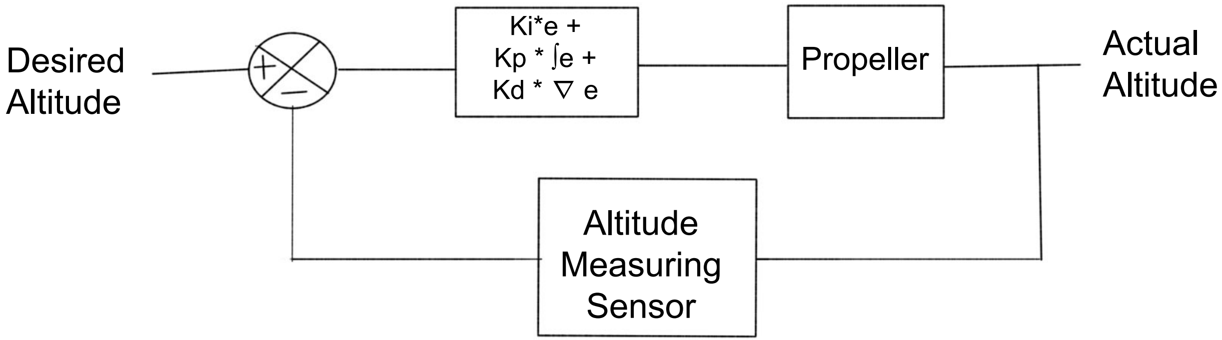


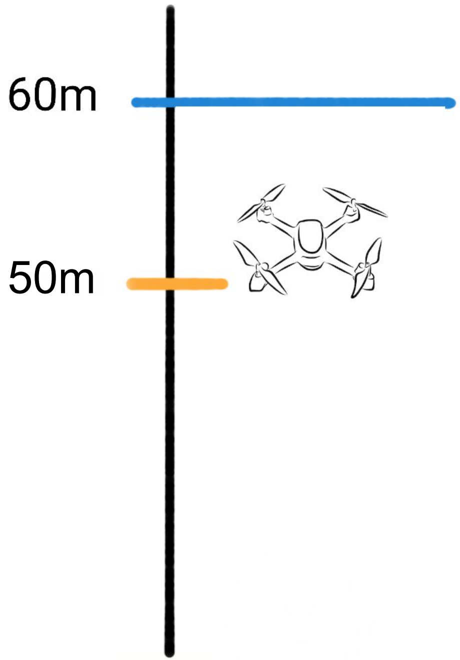
Propeller Speed = 10 * 10 = 100 rpm

Error = 10



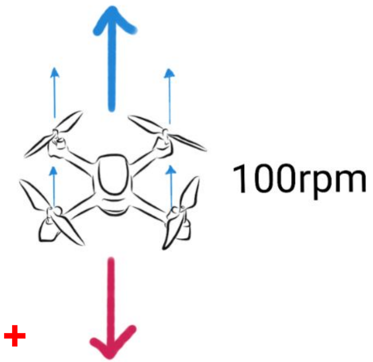
Output = Kp * error + Ki * (error1 + error2 + error3) + Kd * (error3- error 2)





Propeller Speed = 10 * 10 = 100 rpm

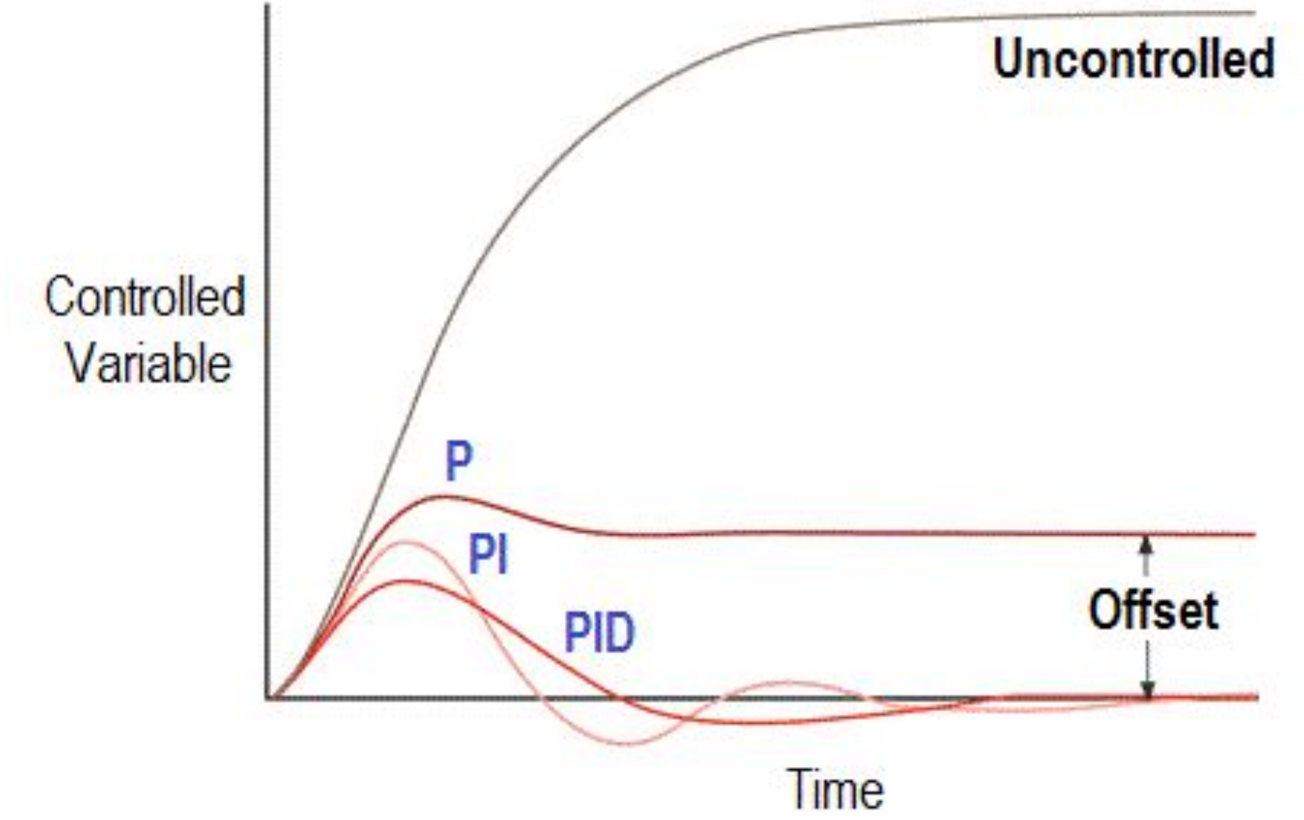
Error = 10



Output = Kp * error + Ki * (error1 + error2 + error3) + Kd * (error3- error 2)

Proportional - Derivative - Integral Control (PID) Control

Proportional - Derivative - Integral Control (PID) Control



PID Controller

$$\text{Proportion al control :} \quad u(t) = K_p e(t)$$

$$\text{Integral control :} \quad u(t) = K_i \int_0^t e(t)dt$$

$$\text{Differenti al control :} \quad u(t) = K_d \frac{d}{dt} e(t)$$

- It produces an output, which is the combination of the outputs of proportional , integral & derivative controllers

$$u(t) \propto e(\mathrm{t}) + \int e(t) + \frac{\mathrm{d}}{\mathrm{d}t} e(t)$$

$$\gg u(t) = K_P e(\mathrm{t}) + K_I \int e(t) + K_D \frac{\mathrm{d}}{\mathrm{d}t} e(t)$$

Effect of Controller Functions

• Proportional Action

Simplest Controller Function, The P term helps to reduce the steady-state error and improve the system's responsiveness. However, too high of a proportional gain can lead to instability and oscillations in the system's response.

• Integral Action

Eliminates steady-state error, The I term helps to improve the system's stability and eliminate any bias in the system. However, too high of an integral gain can lead to overshoot and instability in the system's response.

• Derivative Action (“rate control”)

Effective in transient periods, The D term helps to improve the system's stability and reduce the effects of disturbances in the system. However, too high of a derivative gain can lead to noise amplification and instability in the system's response.

How to get the PID parameter values ?

- If we know the transfer function, analytical methods can be used (e.g., root-locus method) to meet the transient and steady-state specs.
- When the system dynamics are not precisely known, we must resort to experimental approaches.

Ziegler-Nichols Rules for Tuning PID Controller

Using only Proportional control, turn up the gain until the system oscillates without dying down, i.e., is marginally stable. Assume that K and P are the resulting gain and oscillation period, respectively.

Then Use

for P control	for PI control	for PID control
$K_p = 0.5\ K$	$K_p = 0.45\ K$	$K_p = 0.6\ K$
	$K_i = 1.2 / P$	$K_i = 2.0 / P$
		$K_d = P / 8.0$

Ziegler-Nichols Tuning for second or higher order systems

Generating PID parameters

- **Auto-Tuning Algorithms**:
  - Relay-based Auto-Tuning
  - A relay is introduced in the control loop.
  - Observes the system's oscillatory response.
  - Calculates PID parameters based on oscillation characteristics.
- **Trial and Error:** Iteratively adjust parameters based on system response.

Advantages and Applications of PID control

Advantages:

- Simple: easy to understand.
- Effective: Accurate and stable, even in dynamic environments.
- Robust: Can be adopted to different robot systems

Application:

- Arm positioning
- Robot Navigation
- Speed Control
- Balance control

Next Class

Machine Learning