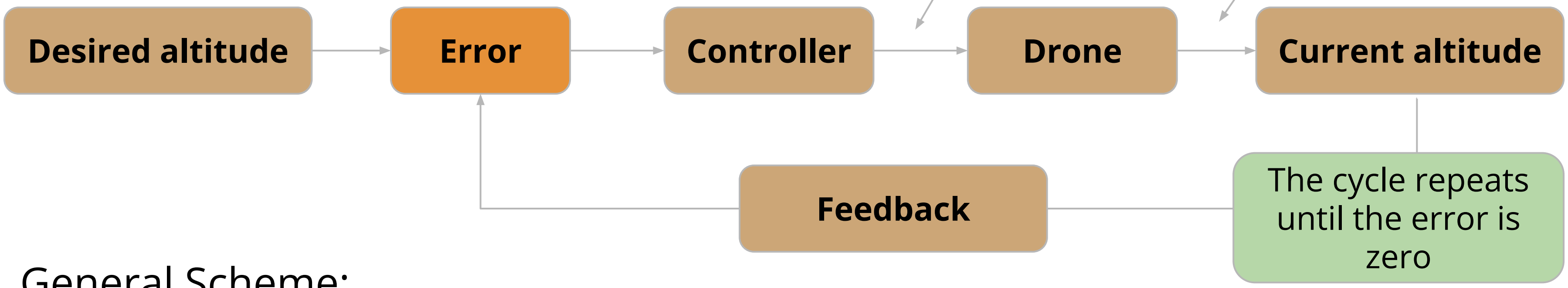
PID control

Table of Contents

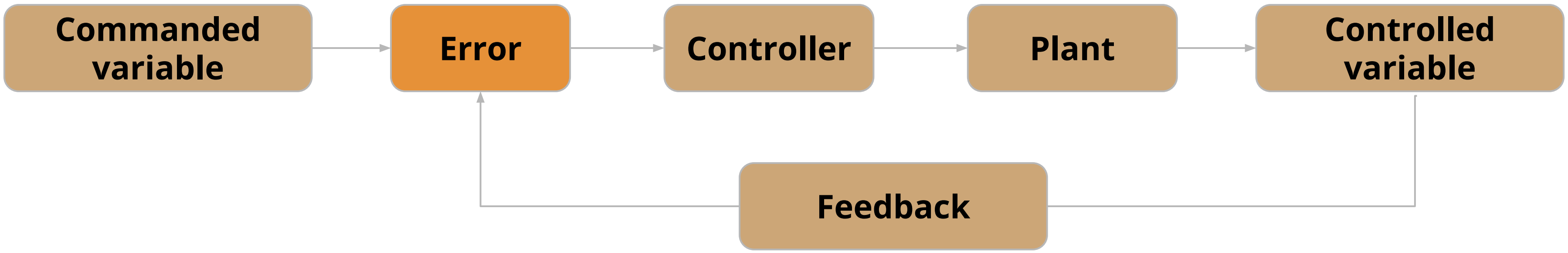
PID controllers

Controller - general idea

Example - controlling altitude of a drone:



General Scheme:



PID Controller - Etymology

If some terms are missing we have “P”, “PD” and so on controllers

Takes into account 3 terms based on the error:

Error

- **P**roportional — What is the error now
- **I**ntegral — What was happening with the error in the past
- **D**erivative — How the error is changing (what will happen in the future)

Best to understand with an example...

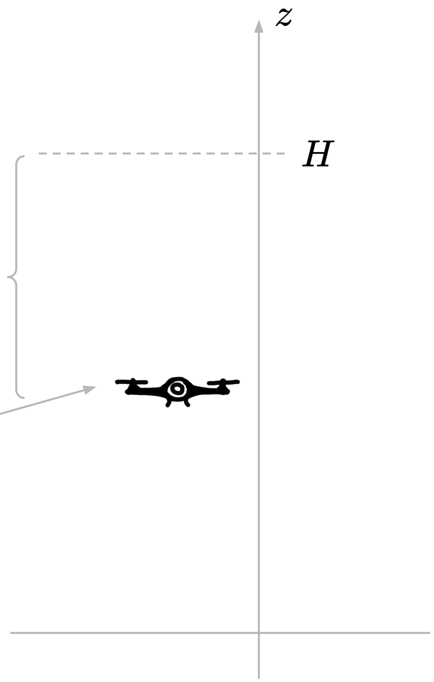
PID Controller - Drone Example

Notation

- current drone altitude: z
- desired altitude: H

signed error:

$$e(z) = z - H$$



We can control the thrust of the propellers...

... they determine the acceleration of the drone

PID Controller - Drone Example

Proportional term:

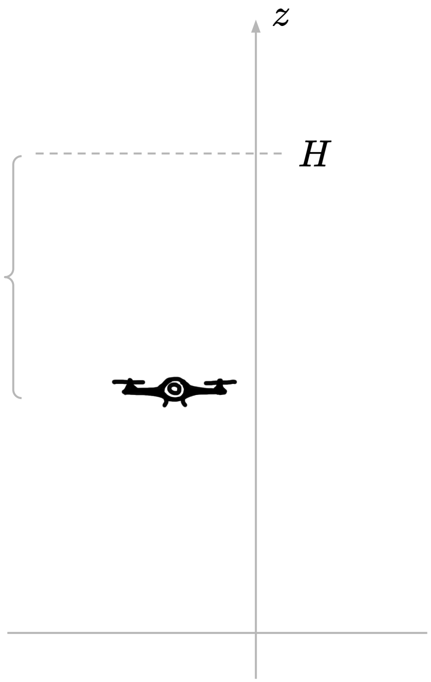
$$F_p = \underbrace{k_{prop}} \cdot e(z)$$

Propellers force

proportional **gain**

signed error:

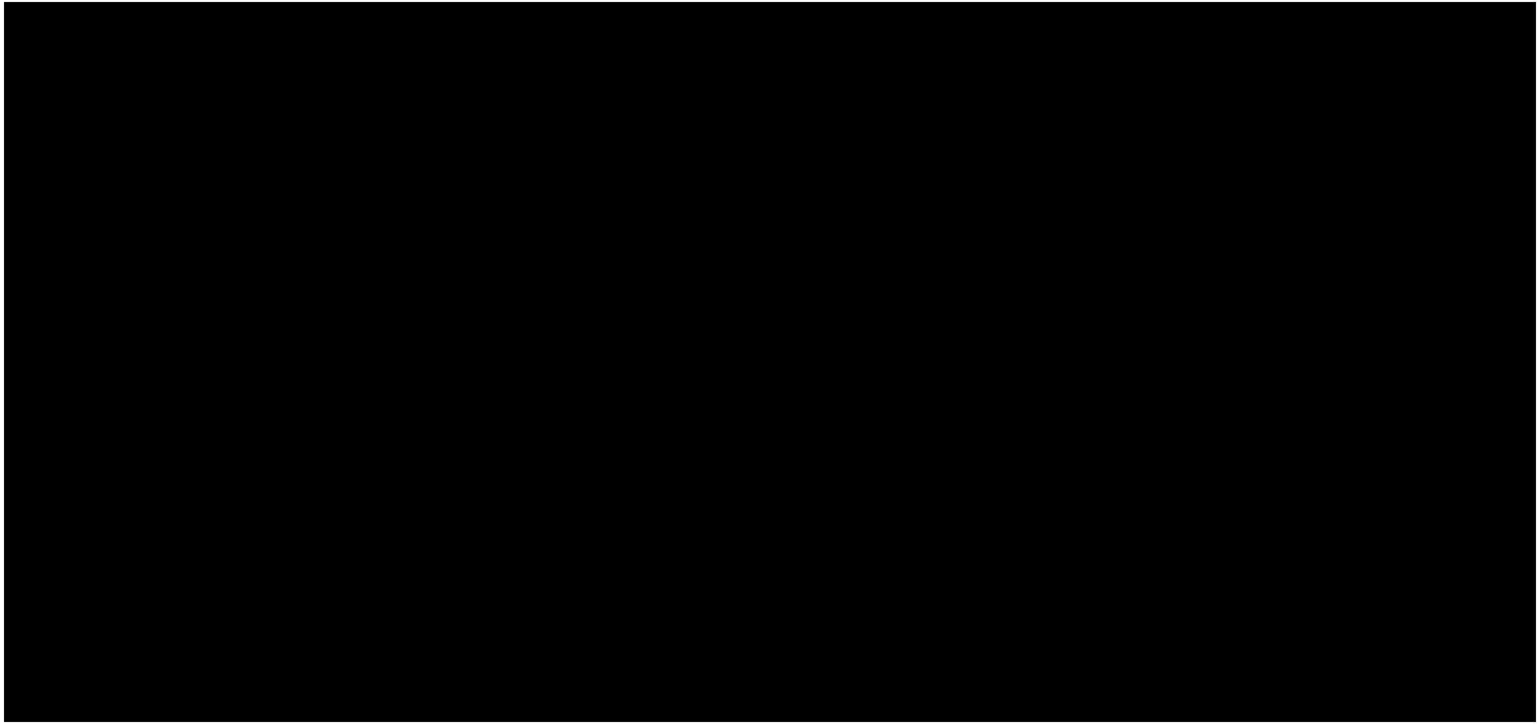
$$e(z) = z - H$$



What do you think will happen?

Let's see a simulation for ***H* = 2**...

Drone Example - PD Controller Simulation

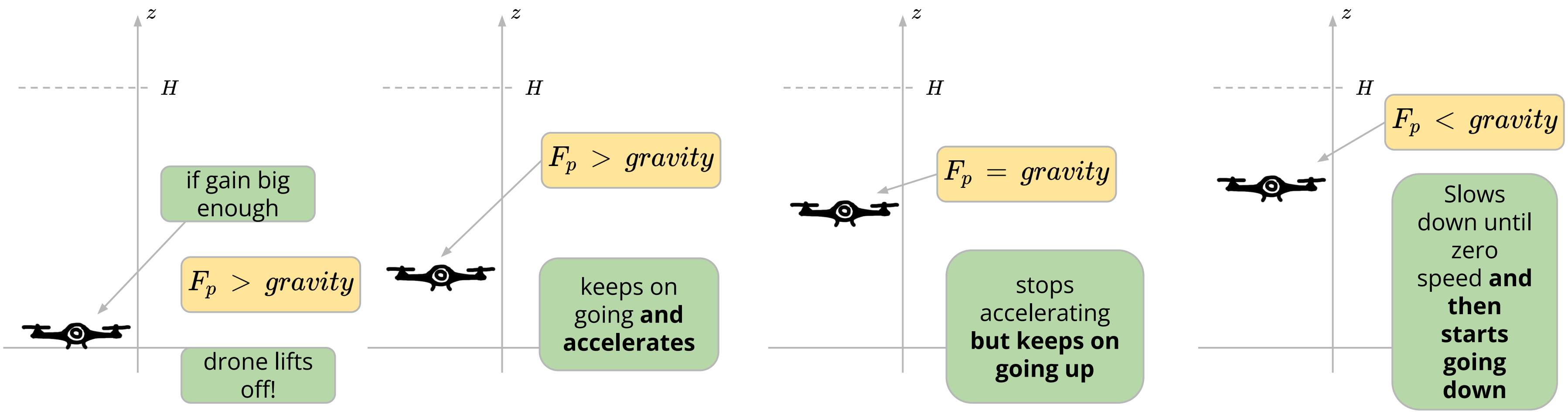


PID Controller - Drone Example

Proportional term:

Will analyse precisely soon - let’s consider few special cases for now

$$F_p = k_{prop} \cdot e(z)$$



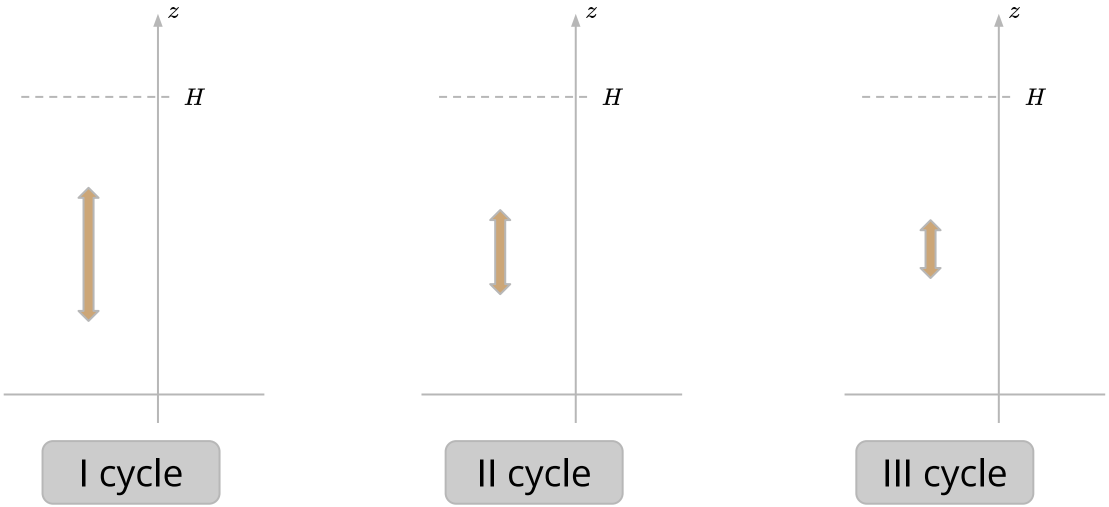
… and then the cycle repeats over and over again

PID Controller - Drone Example

<u>Proportional term:</u>

$$F_p = k_{prop} \cdot e(z)$$

But the amplitudes were actually slowly getting smaller and smaller



This is because we have air resistance...

... which depends on velocity, i.e.: $\dot{z}$

Let's use this to our advantage!

PID Controller - Drone Example

<u>Derivative term:</u>

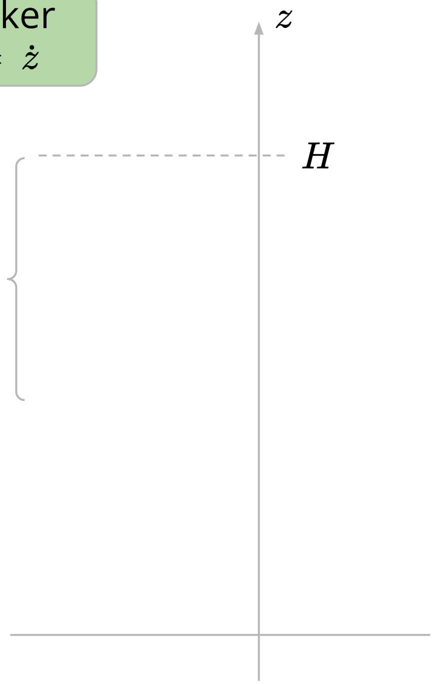
Let's add our own "air resistance" term for quicker convergence, i.e. in our case: $\dot{e}(z) = \dot{z}$

$$F_p = k_{prop} \cdot e(z) + k_{der} \cdot \dot{e}(z)$$

derivative gain

signed error:

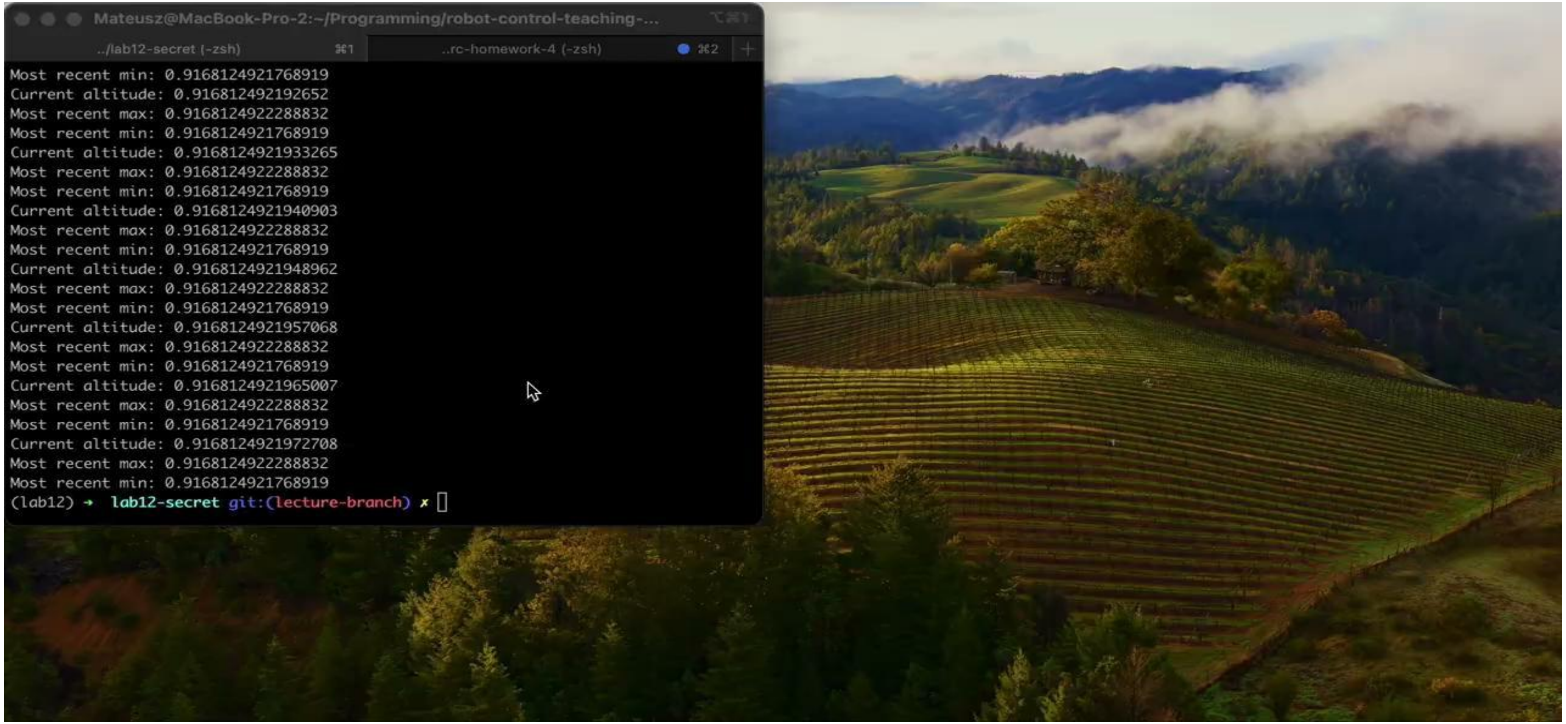
$$e(z) = z - H$$



What do you think will happen?

Let's see a simulation for ***H* = 2**...

Drone Example - PD Controller Simulation



PID Controller - Drone Example

Derivative term:

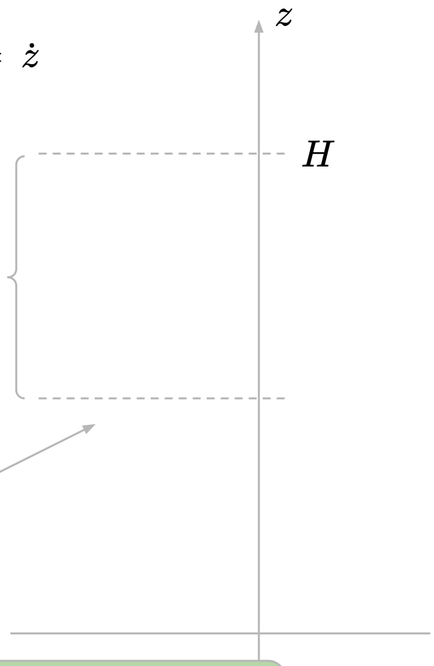
$$F_p = k_{prop} \cdot e(z) + k_{der} \cdot \dot{e}(z)$$

The drone quickly stabilised - great!

$$\dot{e}(z) = \dot{z}$$

signed error:

$$e(z) = z - H$$



But it did not stabilise where we wanted it to...

The drone will stay at the altitude where: $F_p = gravity$

This is not at *H*, because at *H*: $F_p = 0$

PID Controller - Drone Example

<u>Integral term:</u>

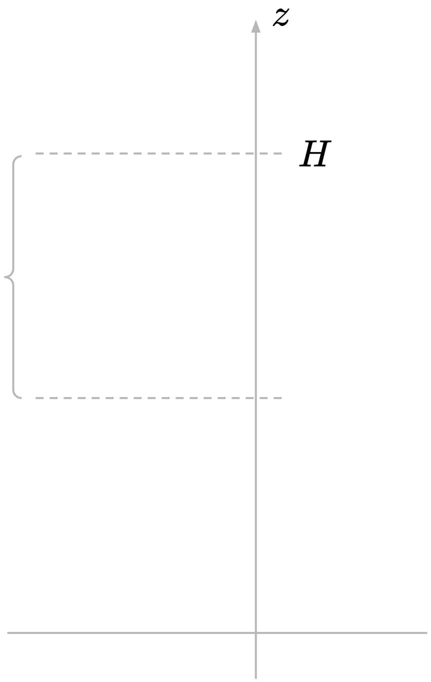
How much error we have accumulated over time

$$F_p = k_{prop} \cdot e(z) + k_{der} \cdot \dot{e}(z) + k_{int} \overbrace{\int e(z)dt}$$

integral gain

signed error:

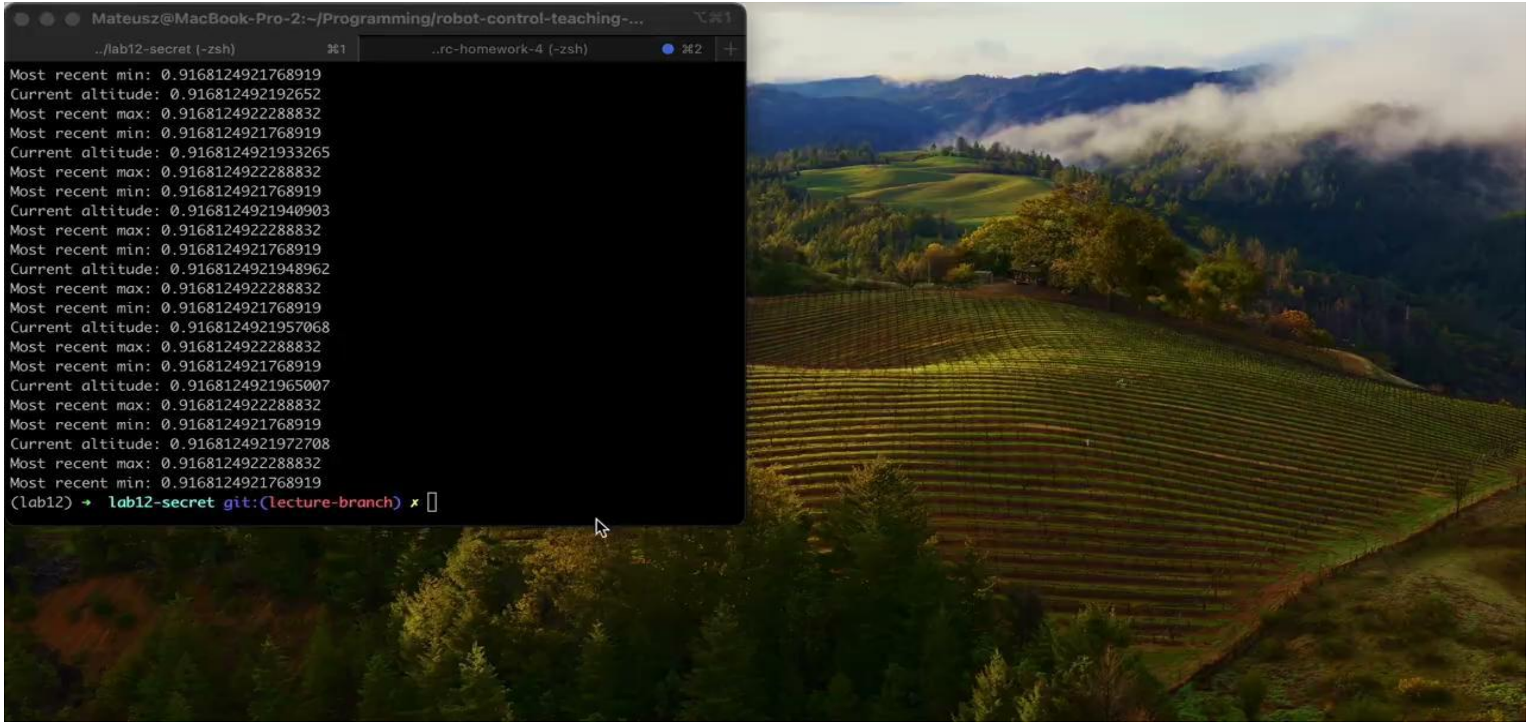
$$e(z) = z - H$$



What do you think will happen?

Let's see a simulation for ***H*** **= 2**...

Drone Example - PID Controller Simulation



PID Controller - Drone Example

Integral term:

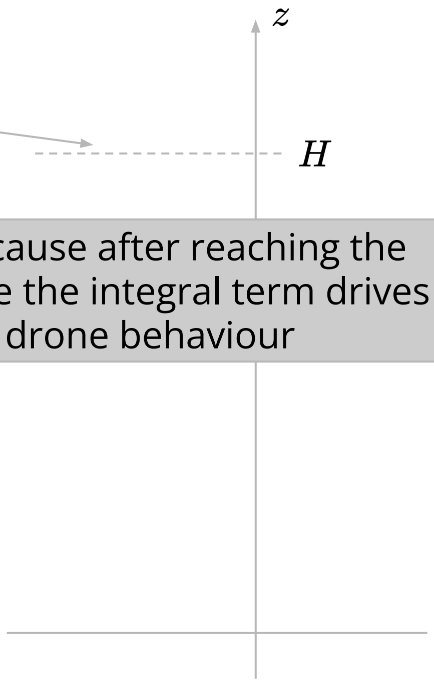
The drone reached the desired altitude and stayed there :)

$$F_p = k_{prop} \cdot e(z) + k_{der} \cdot \dot{e}(z) + k_{int} \int e(z)dt$$

gets smaller

gets bigger with time if error of the same sign

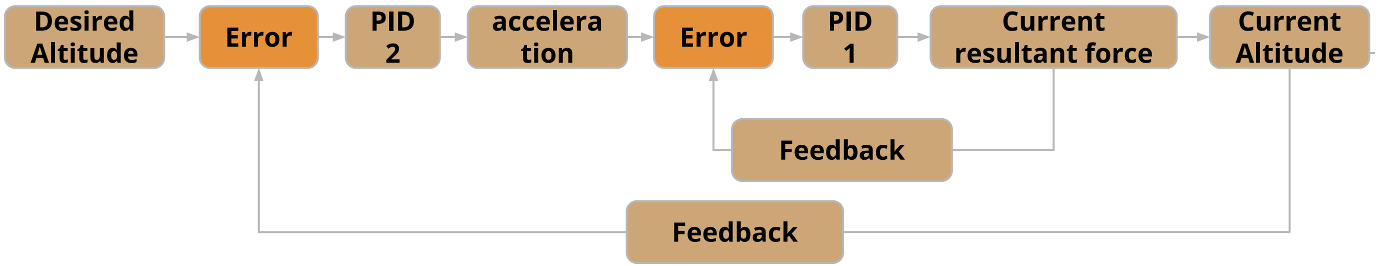
doesn't change much if integral part reacts slowly

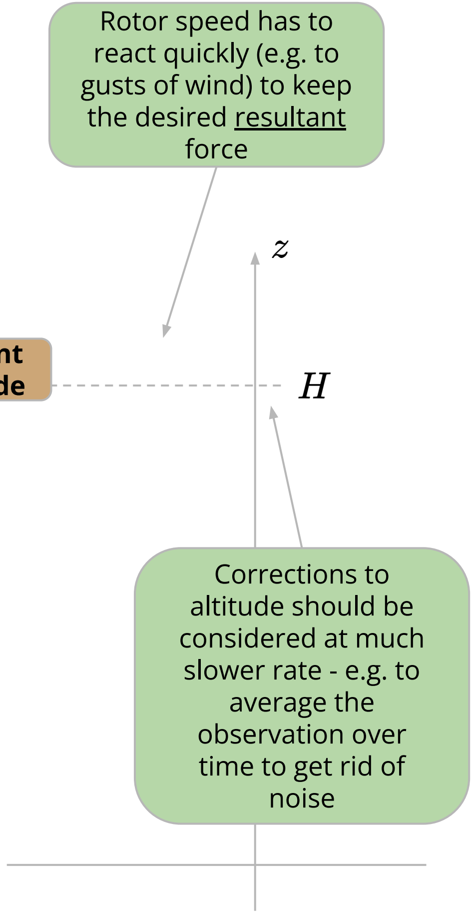


This is because after reaching the steady state the integral term drives the drone behaviour

Multiloop PID Controller

In real life we often have multiple PID controllers interacting with each other:





Why do this?

For example: if one variable controlled at much higher frequency

Transfer Functions

Transfer Function - General Idea

General idea:

started dealing with many interconnected blocks

$\longrightarrow$ **B1** $\longrightarrow$ **B2** $\longrightarrow$

some input: $u(t)$

generates output: $y(t)$

becomes an input for the next block: $\tilde{u}(t)$

generates output: $\tilde{y}(t)$

How to combine these consecutive signals (inputs and outputs)?

Transfer functions - efficient way to do it

Transfer Function - General Idea

Comments:

- Vast topic - if seen for the first time, don’t try to process everything at once
- From perspective of this course: enough to treat it as a black box tool

Combining Signals without Transfer Functions

Block profiles

Every block has some profile...

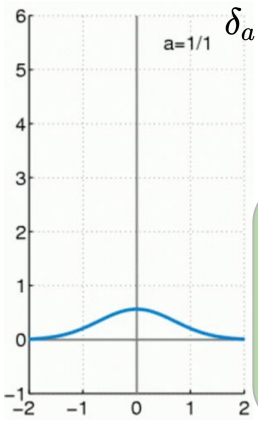
... which determines how the output is generated

$g(t)$

What is this profile?

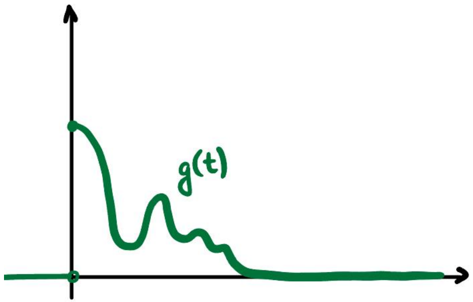
This generates some output:

Imagine single impulse input (modelled with dirac delta function):



$$\delta_a(x) = \frac{1}{|a|\sqrt{\pi}} e^{-\left(\frac{x}{a}\right)^2}$$

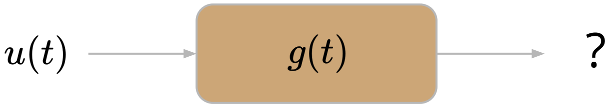
a $\rightarrow$ 0 in the sense of distributions, i.e. integral stays equal to 1



This is the profile!

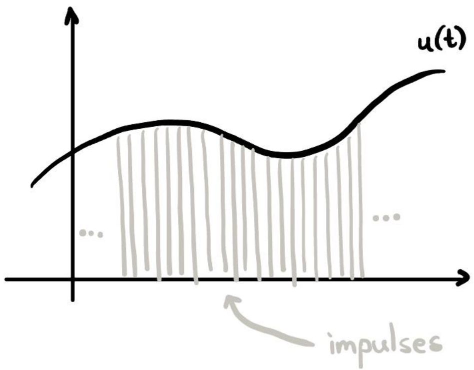
Combining Signals without Transfer Functions

Generating outputs for known profiles



How to determine the output for any signal (not necessarily an impulse)?

Look at the signal as an infinite sum of impulses:

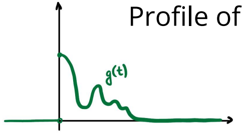


Determine the effect of each impulse and combine

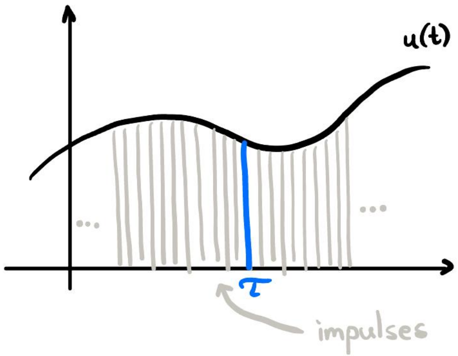
Combining Signals without Transfer Functions

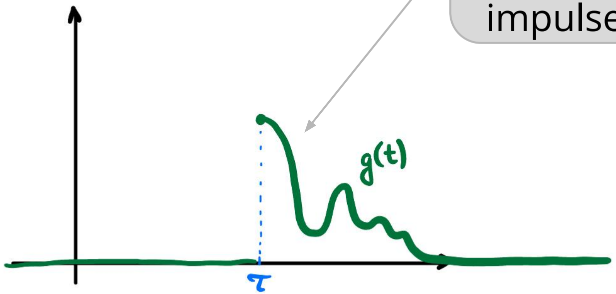
<u>Generating outputs for known profiles</u>

Profile of g



$u(t) \longrightarrow \boxed{g(t)} \longrightarrow ?$





must be scaled according to the corresponding impulse strength

Only impulses at time $\tau < t$

Will have an impact on the output at time t

Combining Signals without Transfer Functions

<u>Generating outputs for known profiles</u>

Sum up over all impulses which can potentially influence time *t*

$$u(t) \longrightarrow \boxed{g(t)} \longrightarrow \int_0^t u(\tau)g(t-\tau)d\tau = u * g$$

So the output at time *t* is generated by impulses at times $\tau < t$

Convolution

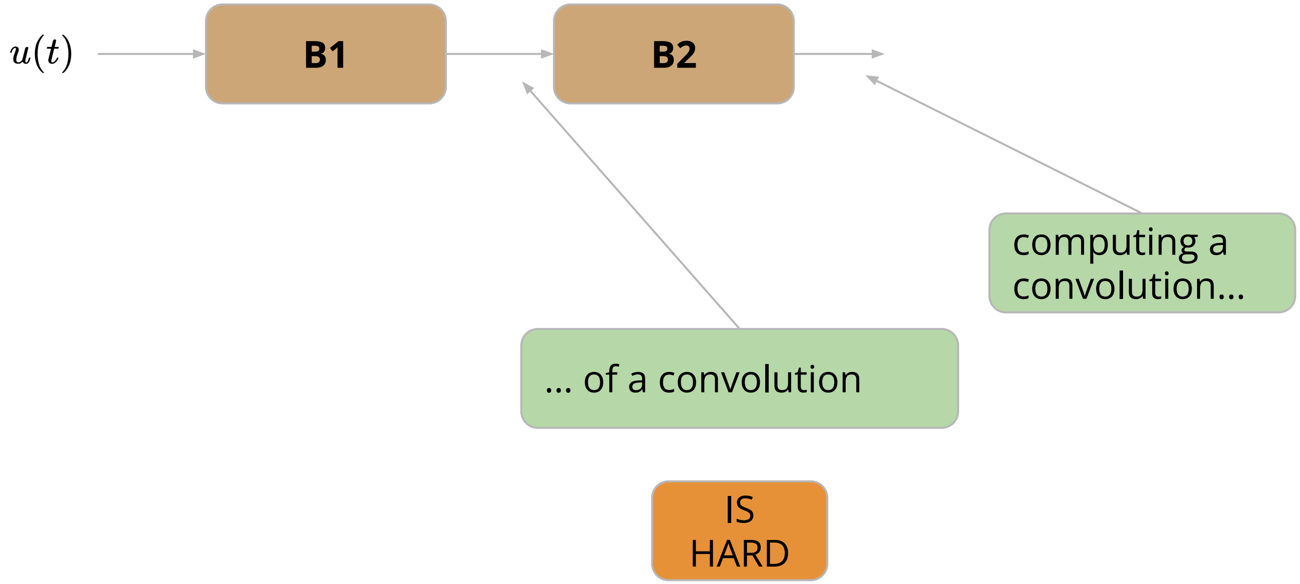
$$u(\tau)g(t-\tau)$$

Impulse amplitude at time τ

Output generated at time *t* (according to profile)

Combining Signals without Transfer Functions

The problem is...



Transfer Functions

Turns out we can use Laplace transform: $\mathcal{L}$

$$u(t) \longrightarrow \boxed{g_1(t)} \longrightarrow \boxed{g_2(t)} \longrightarrow$$

To represent all our functions with (unique!) functions from a different domain $\mathcal{L}u = U$

$$\mathcal{L}u(s) = U(s) = \int_0^\infty u(t)e^{-st}dt$$

The result of convolution in time domain is the multiplication of representatives in *s* domain:

$$\mathcal{L}(u * g) = \mathcal{L}u \cdot \mathcal{L}g$$

AND
MULTIPLICATION
IS EASY

Transfer Functions - PID Example

What will we get in *s* domain

Given an input represented in *s* domain

$U(s)$

Proportional	$k_{prop} \cdot U(s)$
Integral	$\frac{k_{int}}{s} \cdot U(s)$
Derivative	$k_{der} \cdot s \cdot U(s)$

More on these topics

AKA References

More on these topics and references:

- lectures on PID control: https://youtube.com/playlist?list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y&si=VbJdrjhBxXwnmcwL
- transfer functions: https://www.youtube.com/watch?v=RJleGwXorUk
- understanding convolution in the context of signal processing: https://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_ch6.pdf