Lecture 3

IO representations and PID control

Jerome Jouffroy, Professor
jerome@sdu.dk

SDU

Today's lecture

- Transfer functions in the Laplace domain
- Stability and TFs
- Reminder on the basics of PID control

Transfer Functions

Short brush-up on the Laplace transform:

The signal $y(t)$ corresponds to $Y(s)$ in the Laplace domain, with $s \in \mathbb{C}$

For time derivatives, we have

$$\dot{y}(t) \Rightarrow sY(s)$$

$$\ddot{y}(t) \Rightarrow s^2Y(s)$$

Hence, for the ODE $\ddot{y}(t) + a_1\dot{y}(t) + a_0y(t) = b_1\dot{u}(t) + b_0u(t)$

we have $s^2Y(s) + a_1sY(s) + a_0Y(s) = b_1sU(s) + b_0U(s)$

or $(s^2 + a_1s + a_0)Y(s) = (b_1s + b_0)U(s)$

so that we get

$$\boxed{Y(s) = \frac{b_1s + b_0}{s^2 + a_1s + a_0}U(s) = P(s)U(s)}$$

with P(s) Transfer Function

General case

$$\boxed{\frac{Y(s)}{U(s)} = \frac{b_ms^m + \ldots + b_1s + b_0}{s^n + \ldots + a_1s + a_0} = P(s) = \frac{n_p(s)}{d_p(s)}}$$

with $n_p(s) = b_ms^m + \ldots + b_1s + b_0$ $\Rightarrow$ Roots of $n_p(s)$: “zeros”

$d_p(s) = s^n + \ldots + a_1s + a_0$ $\Rightarrow$ Roots of $d_p(s)$: “poles”

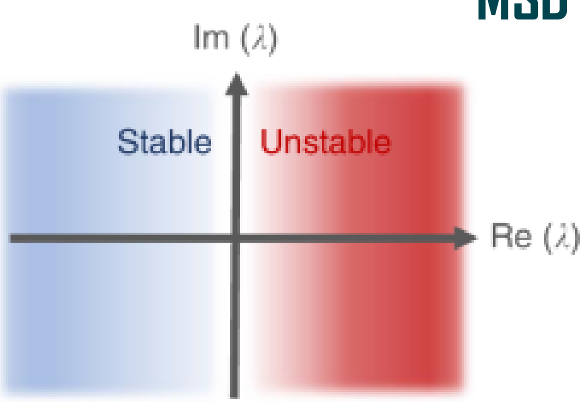
Stability of systems described by transfer functions

<u>Question</u>: What is the interest of the Laplace transform?

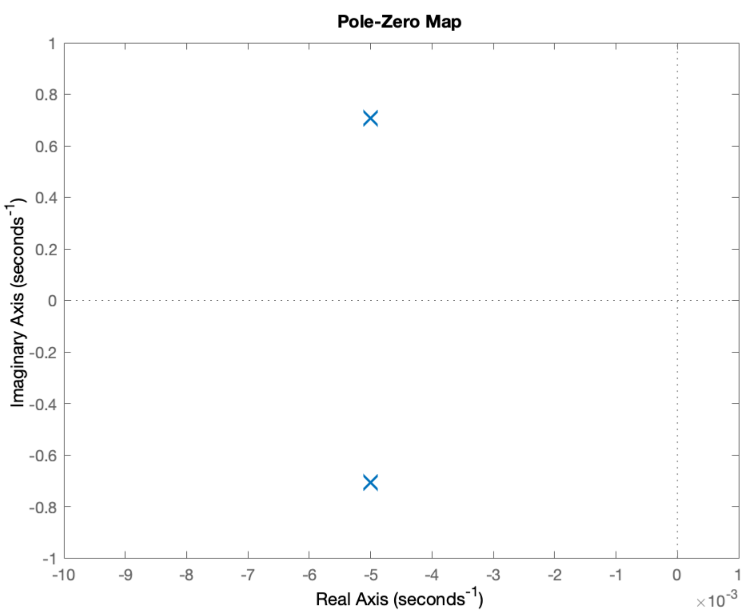
solving an algebraic equation is easier than solving an ODE...

What about stability?

Stability criterion: A system described by Transfer Function $P(s)$ is said to be stable if and only if the real part of each pole (i.e. each root of denominator $d_p(s)$) is strictly negative^2. □



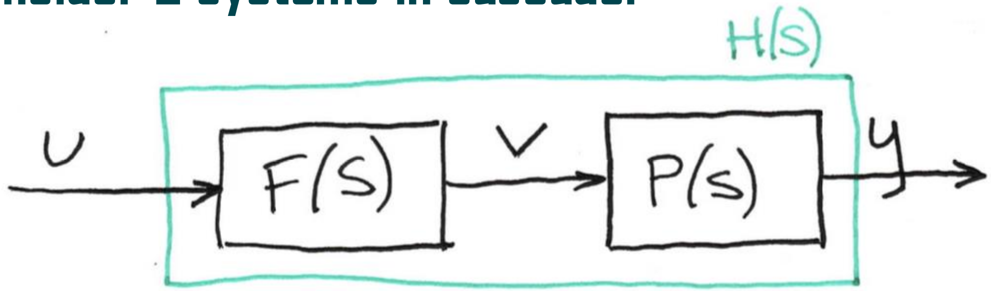
MSD system example:
m=10, d=0.1, k=5



Combining Transfer Functions: cascade

<u>Basic result</u>: combining 2 (or more) TFs gives another TF.

Consider 2 systems in cascade:



TF of first system: $V(s) = F(s)U(s)$

TF of second system: $Y(s) = P(s)V(s)$

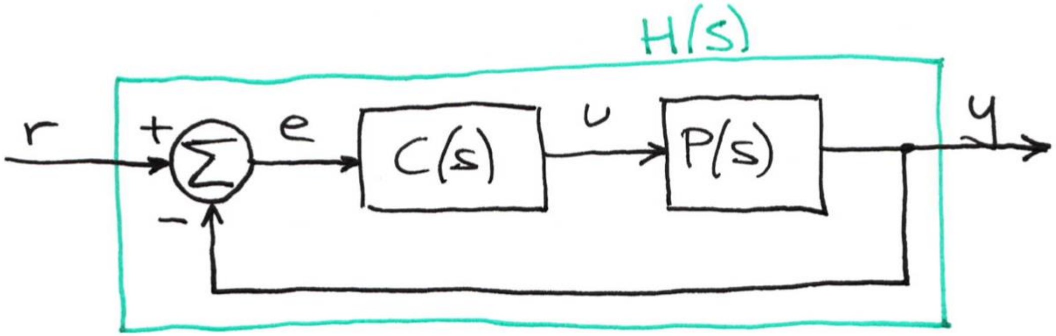
Cascade of the 2 systems together:

$$\boxed{Y(s) = P(s)F(s)U(s)}$$ with $H(s) = P(s)F(s)$

Remark: it can easily be checked that H(s) is stable as long as each (sub)system is itself stable.

Combining TFs: feedback controller with unitary gain (1/2)

Consider a feedback connection with unitary gain:



(typical case: PID controller)

Transfer between y and r?

let $e(t) := r(t) - y(t)$

start with $Y(s) = P(s)C(s)E(s)$

and use $E(s) = R(s) - Y(s)$

to get $Y(s) = P(s)C(s)\left[R(s) - Y(s)\right]$

and isolate Y(s): $\left[1 + P(s)C(s)\right]Y(s) = P(s)C(s)R(s)$ ...

SDU

Combining TFs: feedback controller with unitary gain (2/2)

write again $$[1 + P(s)C(s)]\,Y(s) = P(s)C(s)R(s)$$

to get the TF $$\boxed{H(s) = \frac{Y(s)}{R(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)}}$$

Let's detail that:

$$H(s) = \frac{Y(s)}{R(s)} = \frac{\dfrac{n_p(s)n_c(s)}{d_p(s)d_c(s)}}{1 + \dfrac{n_p(s)n_c(s)}{d_p(s)d_c(s)}} = \frac{\dfrac{n_p(s)n_c(s)}{d_p(s)d_c(s)}}{\dfrac{d_p(s)d_c(s) + n_p(s)n_c(s)}{d_p(s)d_c(s)}}$$

so that we have $$\boxed{H(s) = \frac{n_p(s)n_c(s)}{d_p(s)d_c(s) + n_p(s)n_c(s)}}$$

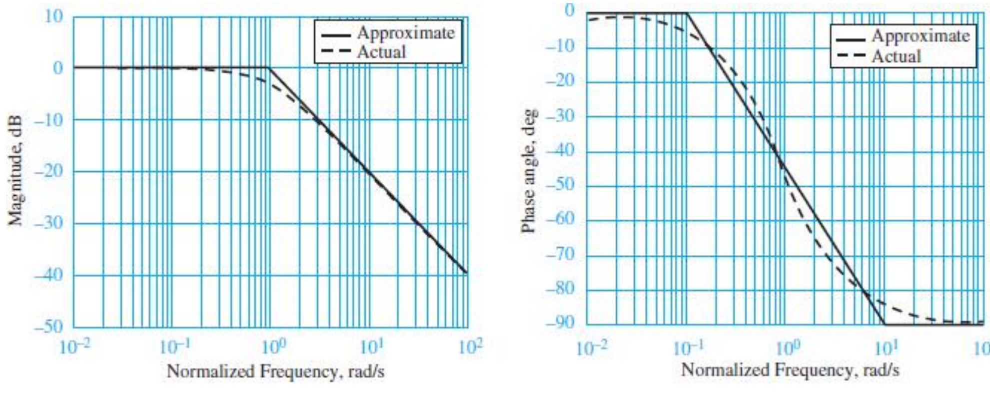
H(s) is stable if the roots of

$$d_p(s)d_c(s) + n_p(s)n_c(s)$$

are all in the left half-plane

$\Rightarrow$ <u>having both P(s) and C(s) stable does not necessarily imply stability of H(s)</u>.

SDU



(a) (b)

Bode plots

For any stable linear system, any sinusoidal input signal

$$u(t) = \sin(\omega t)$$, we have the output signal $$y(t) = A\sin(\omega t + \varphi)$$

where A and φ depend on ω

Magnitude:

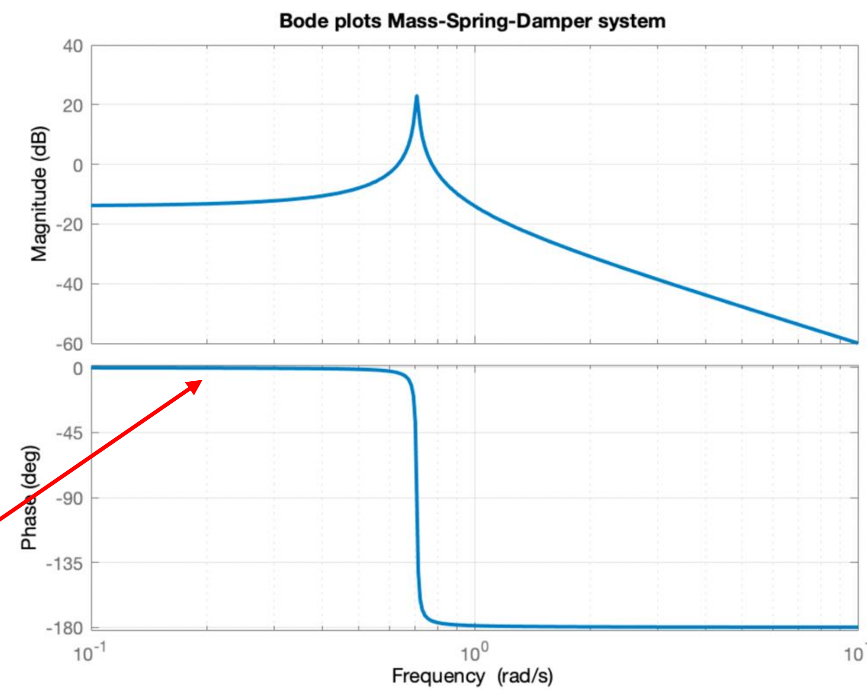
$$A = |P(s = j\omega)| \ , \ \omega \in \mathbb{R}^+$$

Phase:

$$\varphi = arg(P(s = j\omega)) \ , \ \omega \in \mathbb{R}^+$$

Remark: log scale for ω and decibels for gain plot: $20\log_{10}|P(s = j\omega)|$

MSD system example: (m=10, d=0.1, k=5)



SDU

Zero-frequency gain

Simple but fundamental quantity in control engineering: $P(s)$ when $\left| \begin{array}{l} s = 0 \\ \omega = 0 \end{array} \right.$

(also called DC gain)

Links steady-state output to steady-state input
(related to notion of equilibrium)

MSD example again:

Start with $m\ddot{y}(t) + d\dot{y}(t) + ky(t) = u(t)$

"Laplace transform" to $ms^2Y(s) + dsY(s) + kY(s) = U(s)$

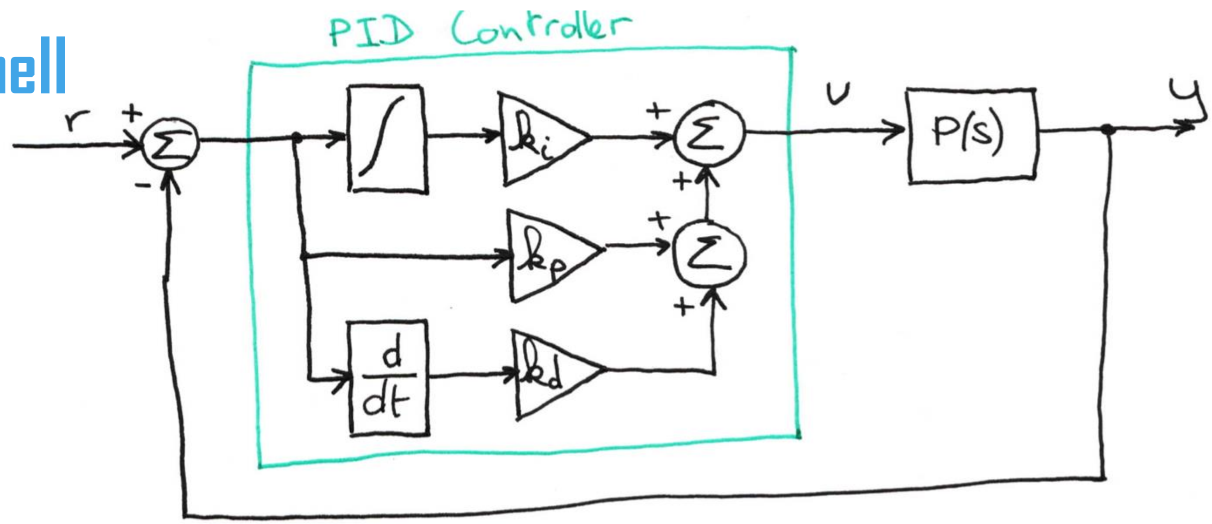
or $\left(ms^2 + ds + k\right)Y(s) = U(s)$

so that we have the TF $$\frac{Y(s)}{U(s)} = P(s) = \frac{1}{ms^2 + ds + k} = \frac{1/m}{s^2 + (d/m)s + k/m}$$

$$P(0) = 1/k.$$

SDU

PID control in a nutshell



Transfer Function: $C(s) = k_p + k_i/s + k_d s$ **(ideal version)**

in the time domain: $u(t) = k_p e(t) + k_d \dot{e}(t) + k_i \int_0^t e(\tau) d\tau$

with $e(t) := r(t) - y(t)$

3 terms:

P: $\approx$ stabilize the system

I: make the output y eventually reach the reference r (0 steady-state error)

D: adds damping in the system to “make the output smoother”

Tuning the PID controller: the Ziegler-Nichols rule

Question: how to tune a PID controller? ➔ many guidelines exist

Zieglier-Nichols tuning "algorithm":

- set all gains to 0

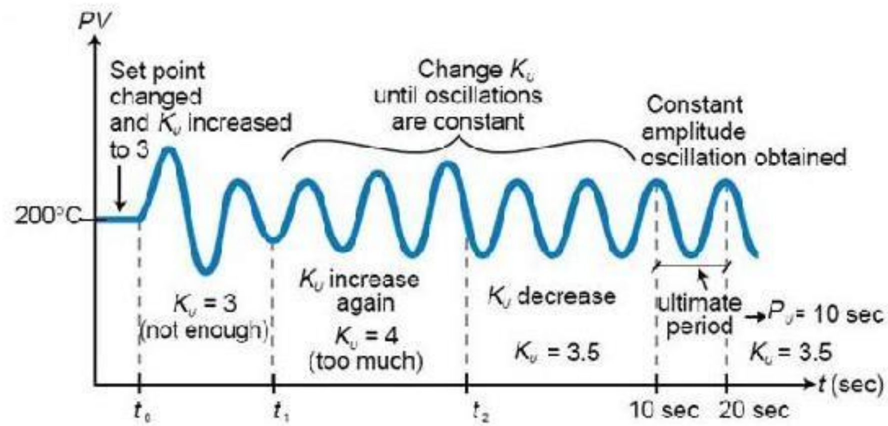
- change k_p until $y(t)$ oscillates

- when it does, note as k_c the current value of k_p

- finally, let $\boxed{k_p = 0.6 k_c,}$

$$\boxed{k_i = \frac{k_p}{0.5 T_c},}$$

$$\boxed{k_d = k_p 0.125 T_c}$$



Remarks:
- many variations of the ZN rule exist
- only guidelines: might continue manual tuning afterwards
- only works for stable systems
- not always possible to make a system oscillate

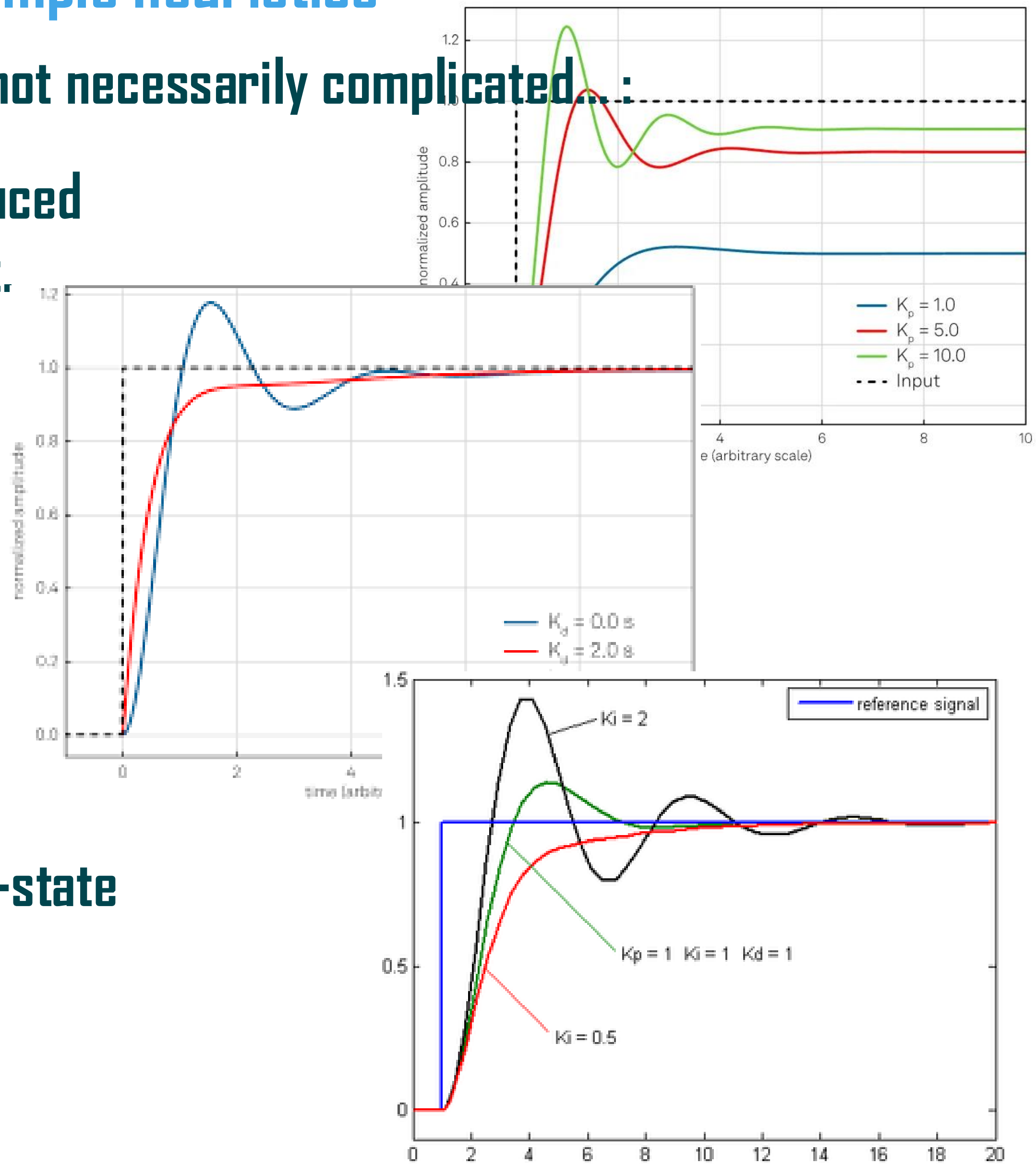
Tuning the PID controller: simple heuristics

… tuning a PID controller is not necessarily complicated… :

- change k_p until you have a reduced steady-state error / stable syst.

- change k_d to reduce the oscillations of the output

- increase k_i to bring the steady-state error to 0



SDU

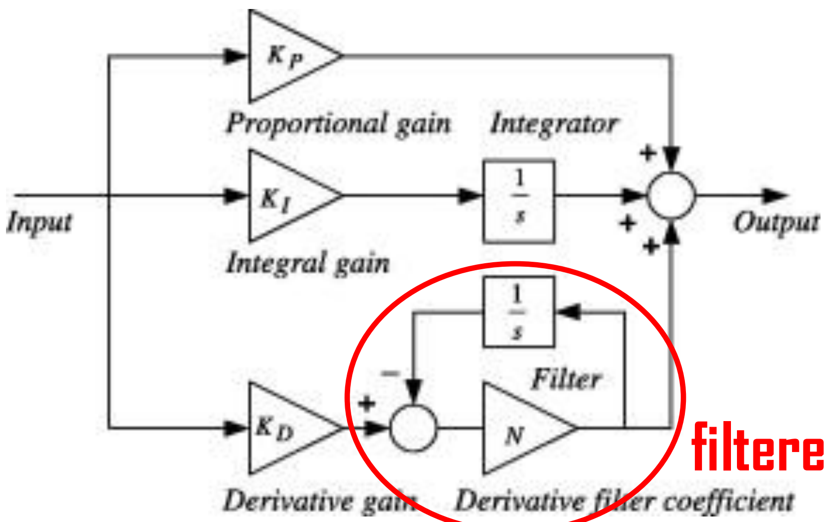
Two implementation tricks: derivative term

<u>Derivative issue</u>: Remember what we said? differentiator = BAD

integrator = GOOD

So what do we do about $k_d \dot{e}(t)$ (will amplify noise + jumps if change in r(t))

Remedy: replace $k_d s E(s)$



filtered derivative

with

N

$$\boxed{k_d s \frac{1/T_f}{s + 1/T_f} E(s)}$$

(filtered derivative)

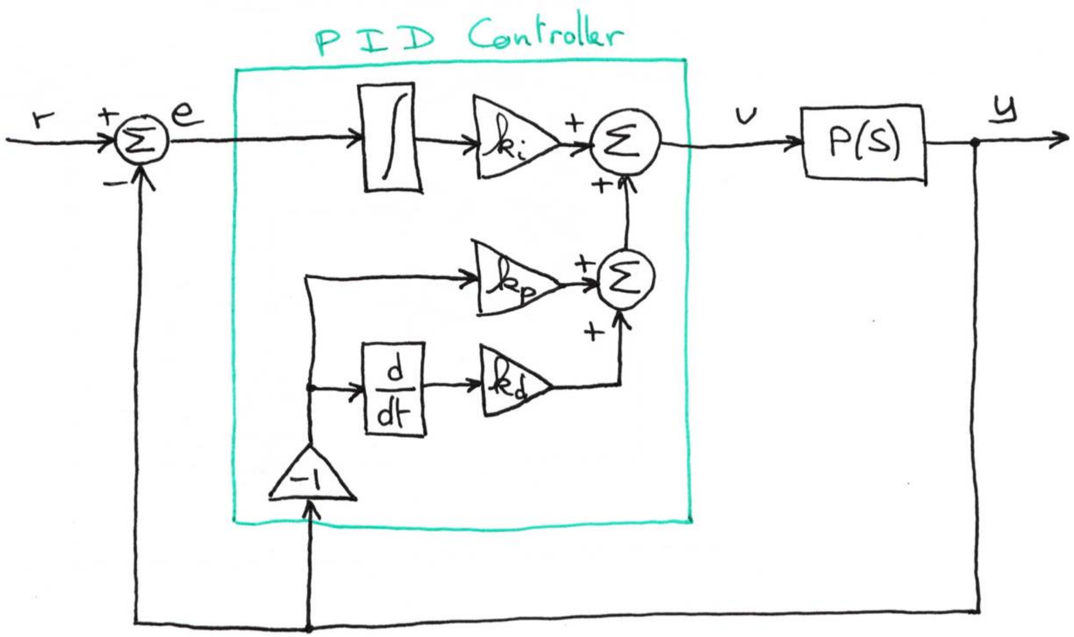
Two implementation tricks: "jumps" in the reference signal

Problem of using the error signal $e(t) := r(t) - y(t)$ in all terms:

P term: a jump/sudden change in r(t) leads to a jump in u(t) ($u(t)=k_p e(t)$) → not good for the actuator

Useful trick: using the error only in the integrator still allows to stabilize a system around the reference:

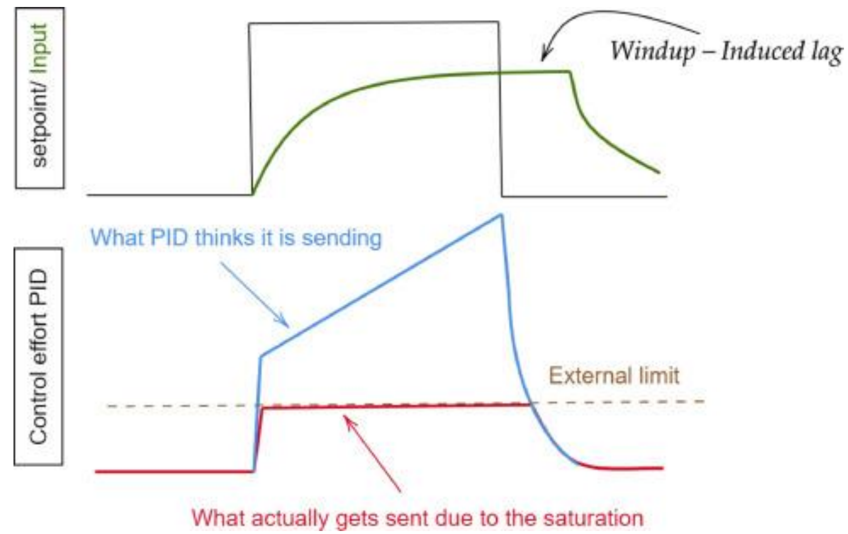
$$u(t) = k_p(-y(t)) + k_d(-\dot{y}(t)) + k_i \int_0^t (r(\tau) - y(\tau))d\tau.$$



There is a lot more to PID control

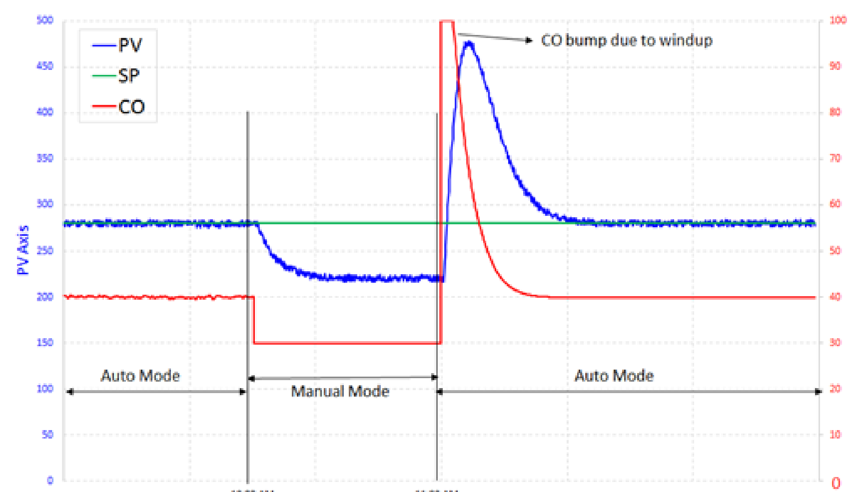
Anti-windup

The windup issue: when what is asked from u(t) is beyond the actuator limits, e(t) cannot be 0, and this makes the integrator value grow bigger and bigger, way too large when r(t) comes back to something feasible…



Bumpless transfer

The switching bump issue: when a controller is set off, keeps integrating the error, and then put back on, thus creating a large overshoot due to integrator accumulation…



SDU