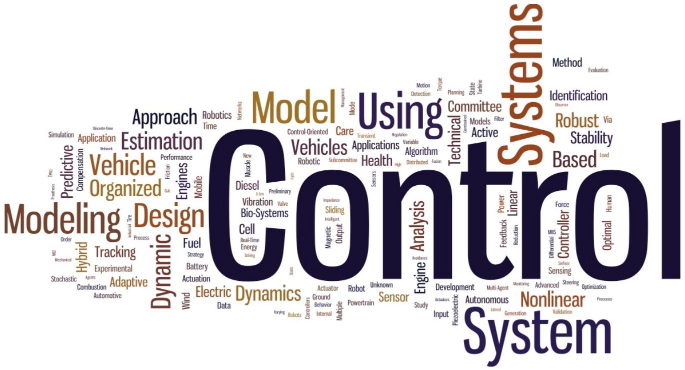


Automatique

PID controllers

Effects of proportional, integral and derivative terms

Hugues GARNIER

hugues.garnier@univ-lorraine.fr

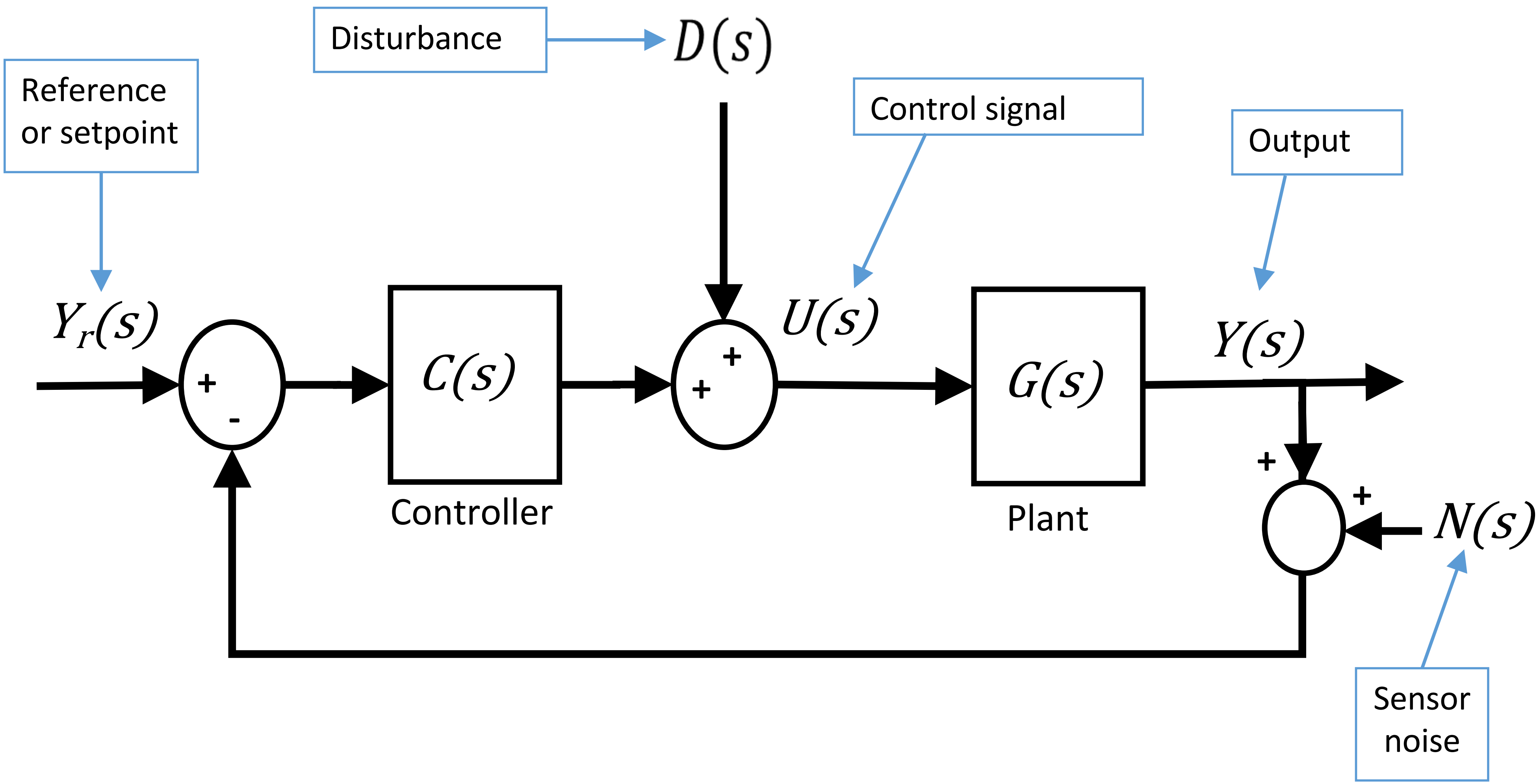
Version du 8 décembre 2024

These slides have been modified from an initial version developed by Quanser

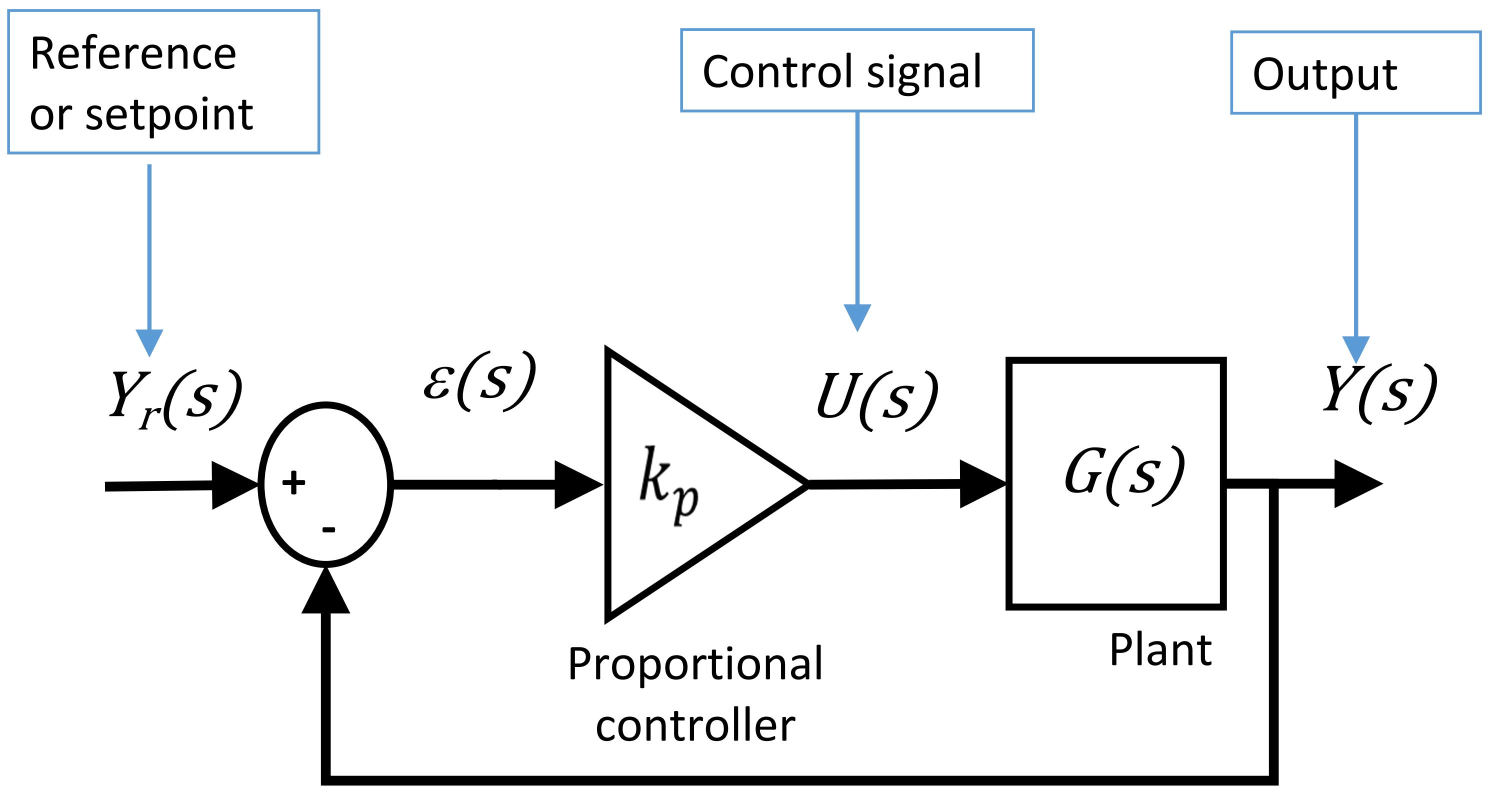
https://www.quanser.com

I sincerely thank Quanser for allowing me to adapt them

General feedback control diagram



Proportional Control



Equation of the P control law and transfer function

In the time domain:

$$u(t) = k_p\left(y_r(t) - y(t)\right)$$

$$u(t) = k_p\,\varepsilon(t)$$

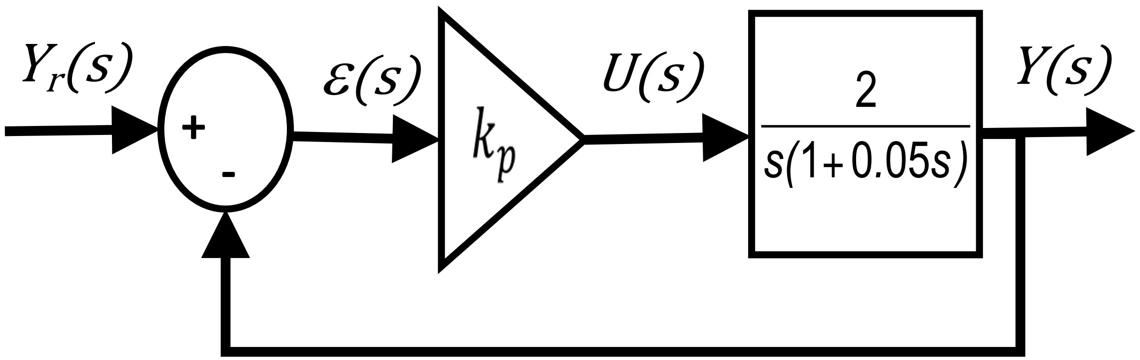
In the Laplace domain:

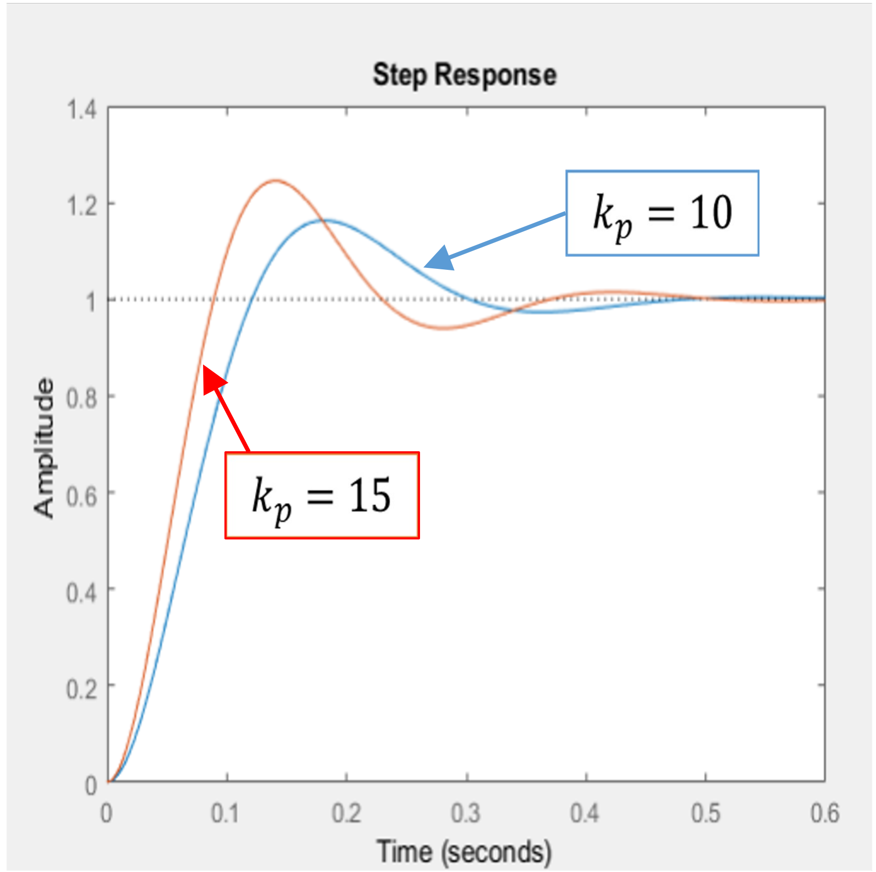
$$U(s) = k_p\left(Y_r(s) - Y(s)\right)$$

$$U(s) = k_p\,\varepsilon(s)$$

$$C(s) = \frac{U(s)}{\varepsilon(s)} = k_p$$

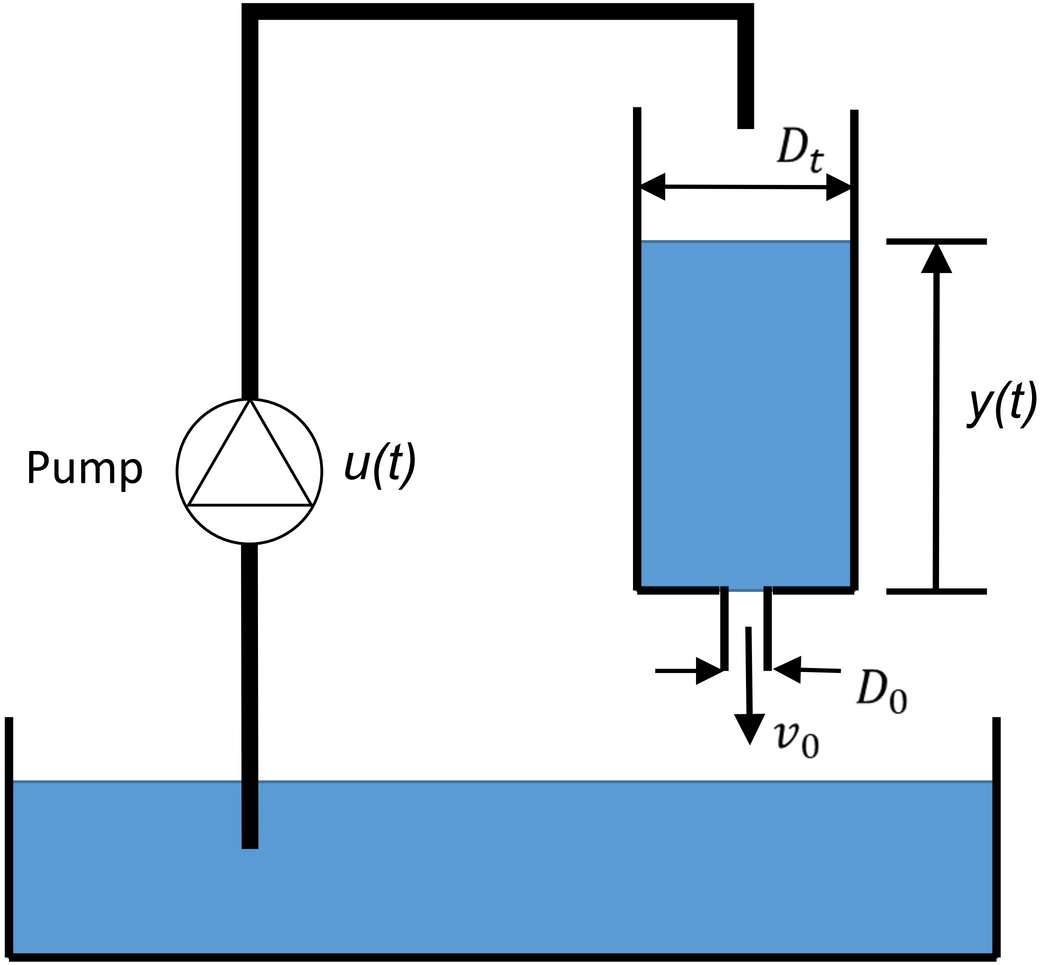
P control: effect of proportional gain





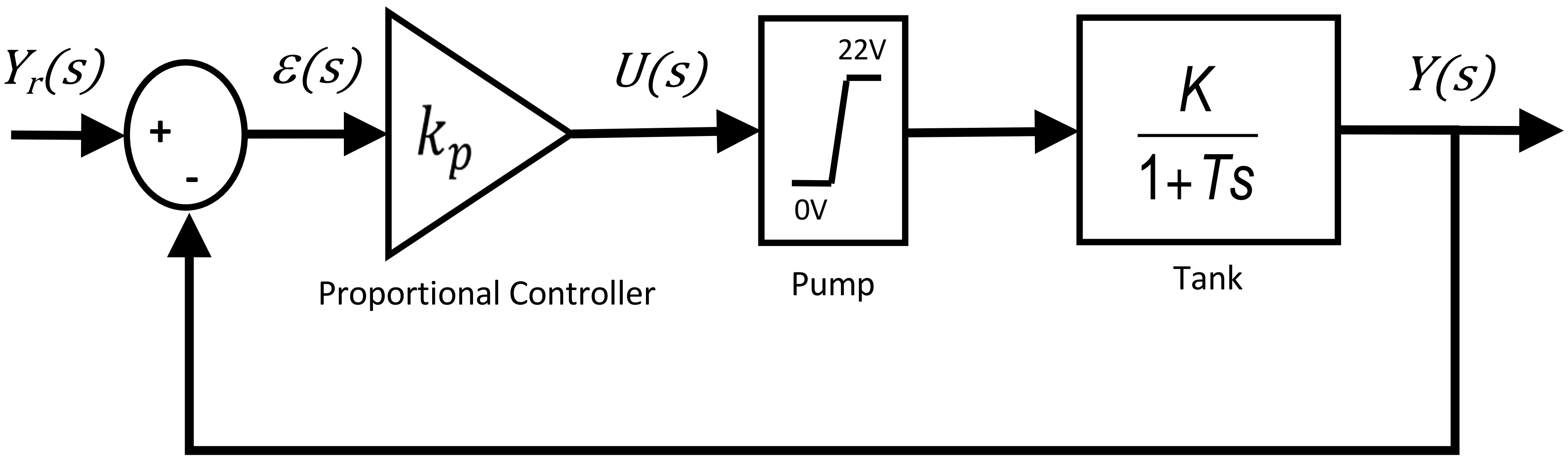
- Increase $\boldsymbol{k_p}$ gradually
- What can be noticed ?
  - Peak time decreases, *i.e.* faster response
  - Overshoot increases

Example 1: water tank level control



- Input: voltage sent to the pump *u(t)*
- Output : water level in the tank *y(t)*

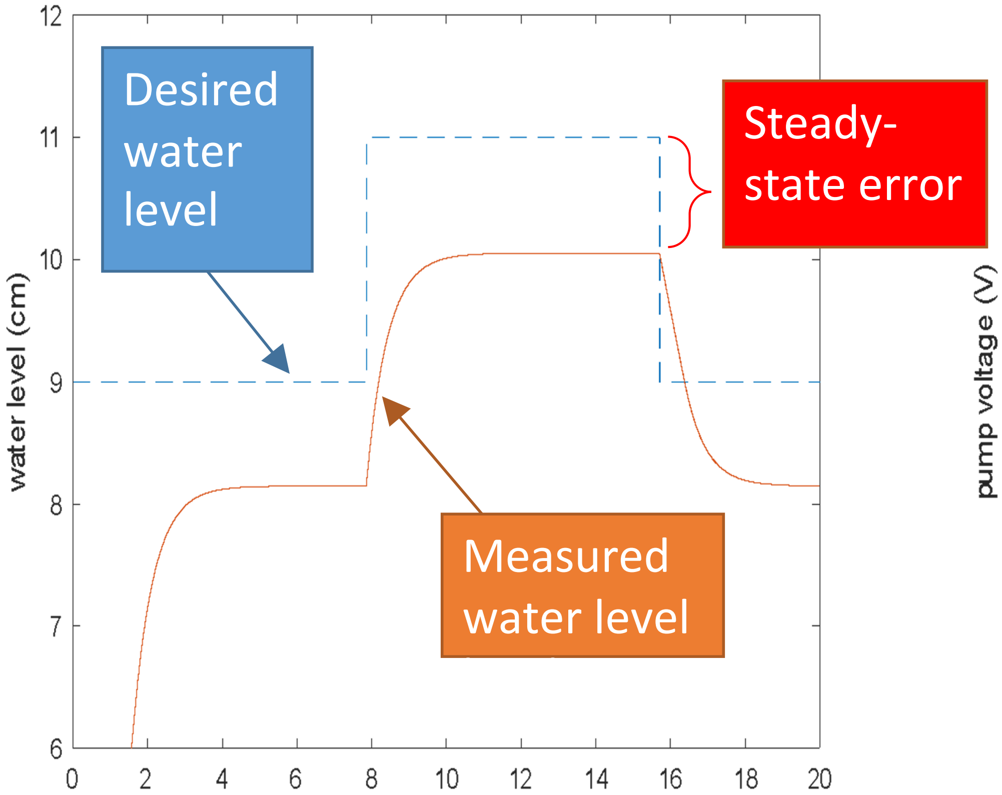
Example 1: water tank level control

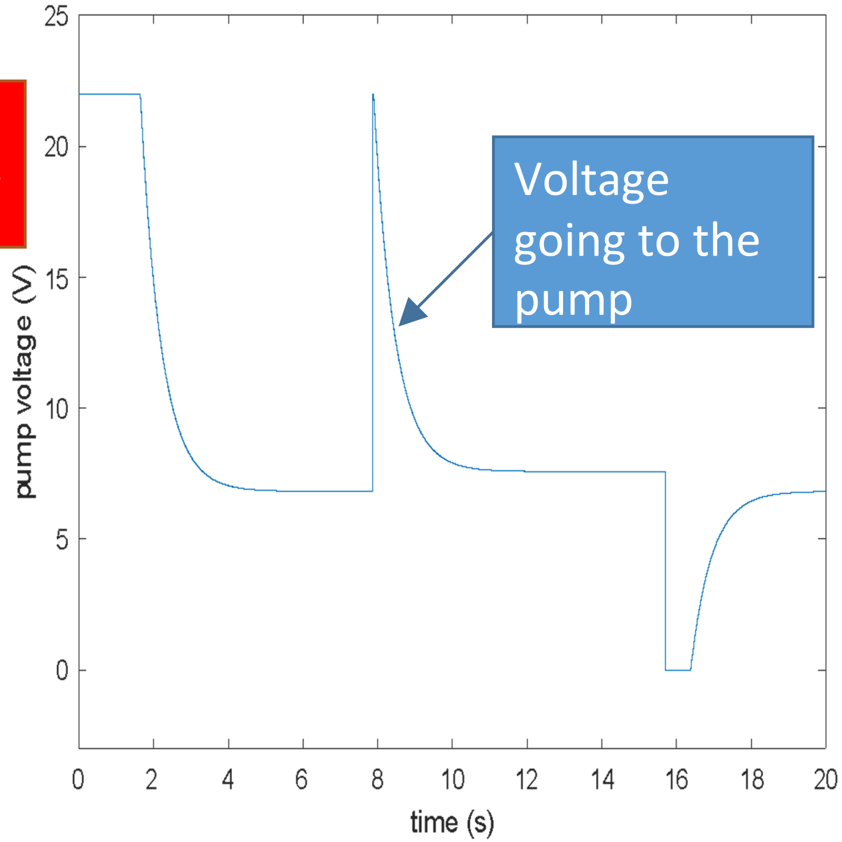


Response: water tank level P control

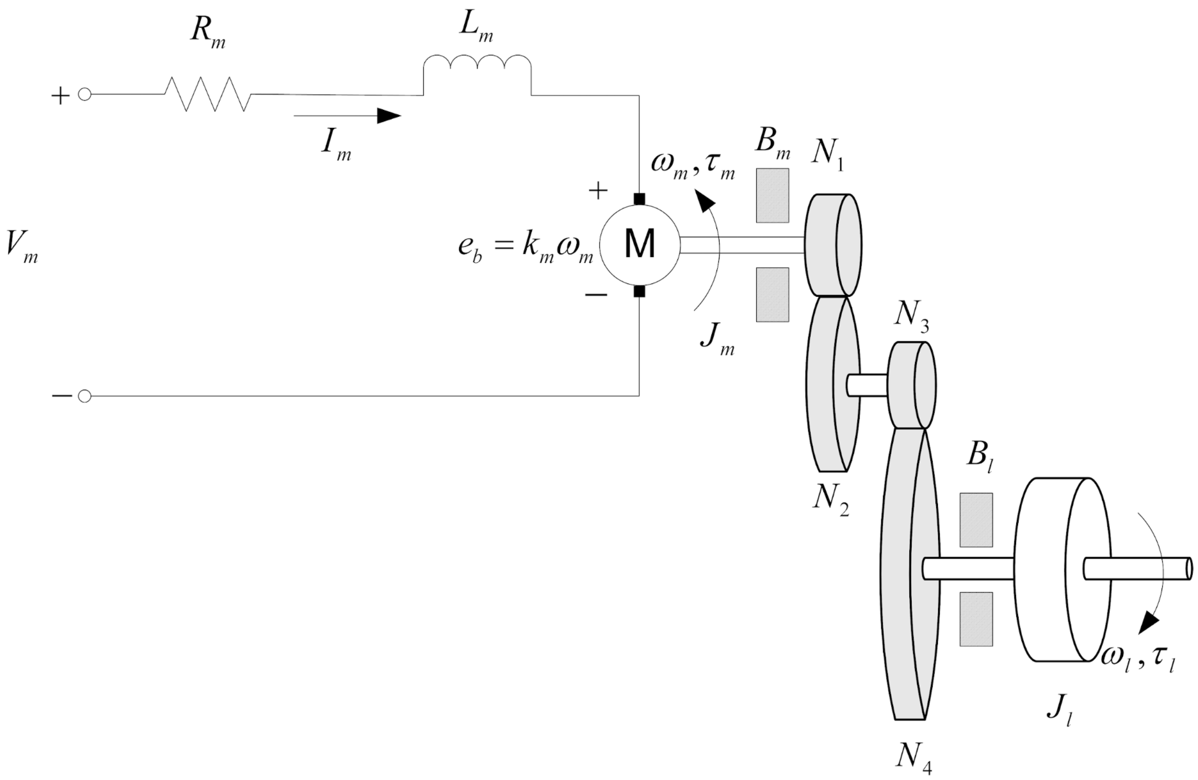
Tank response does not track desired water level well

Control effort, i.e. voltage going to pump, is smooth



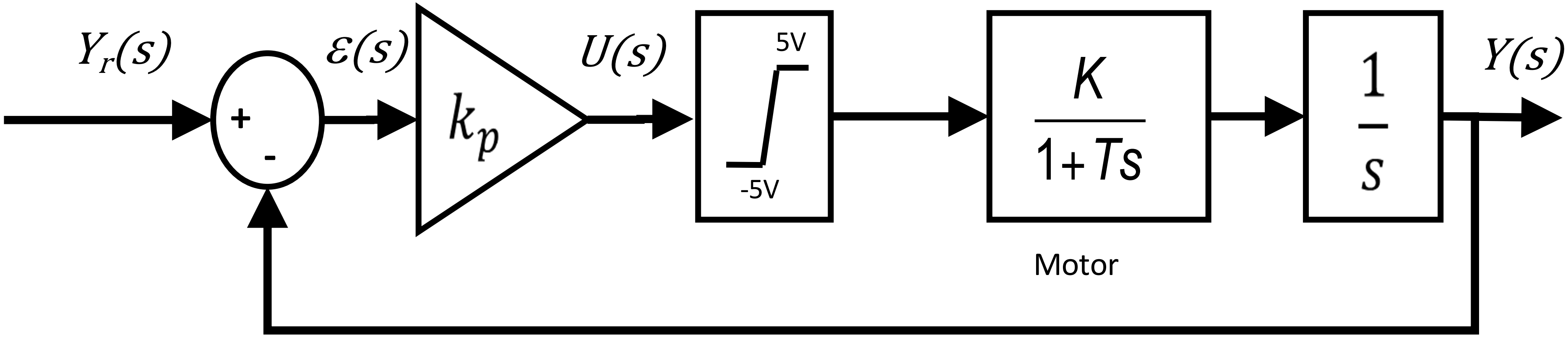


Example 2: P for servo motor position control



- Input: voltage sent to the motor *u(t)*
- Output : angular position *y(t)*

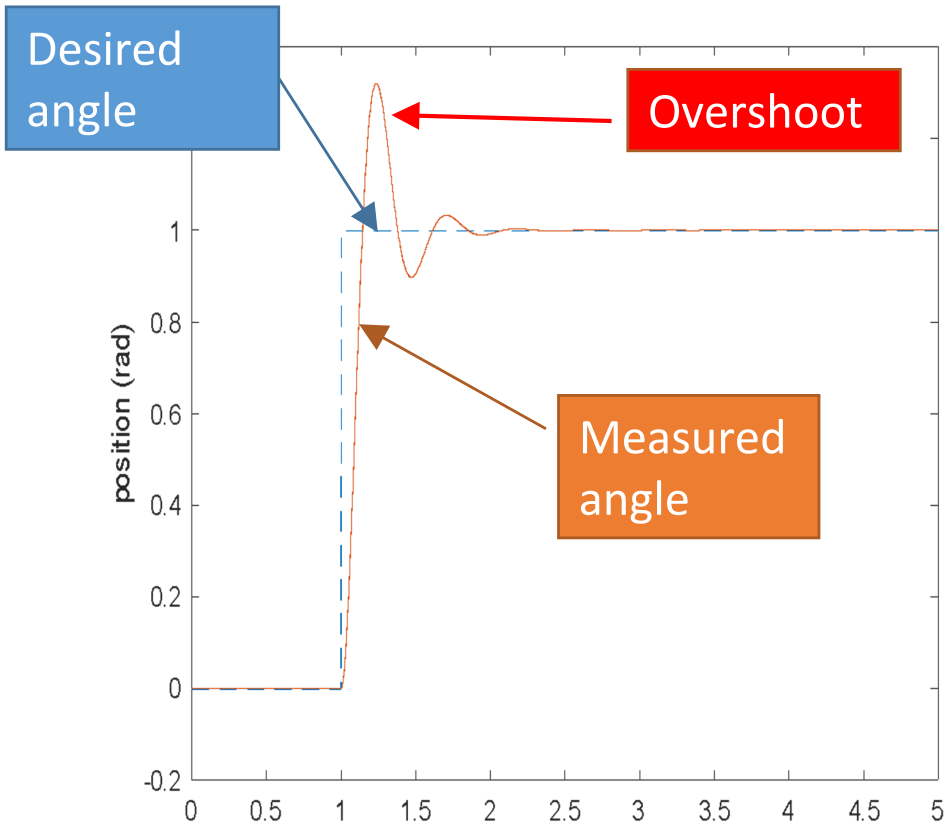
Example 2: P for servo motor position control

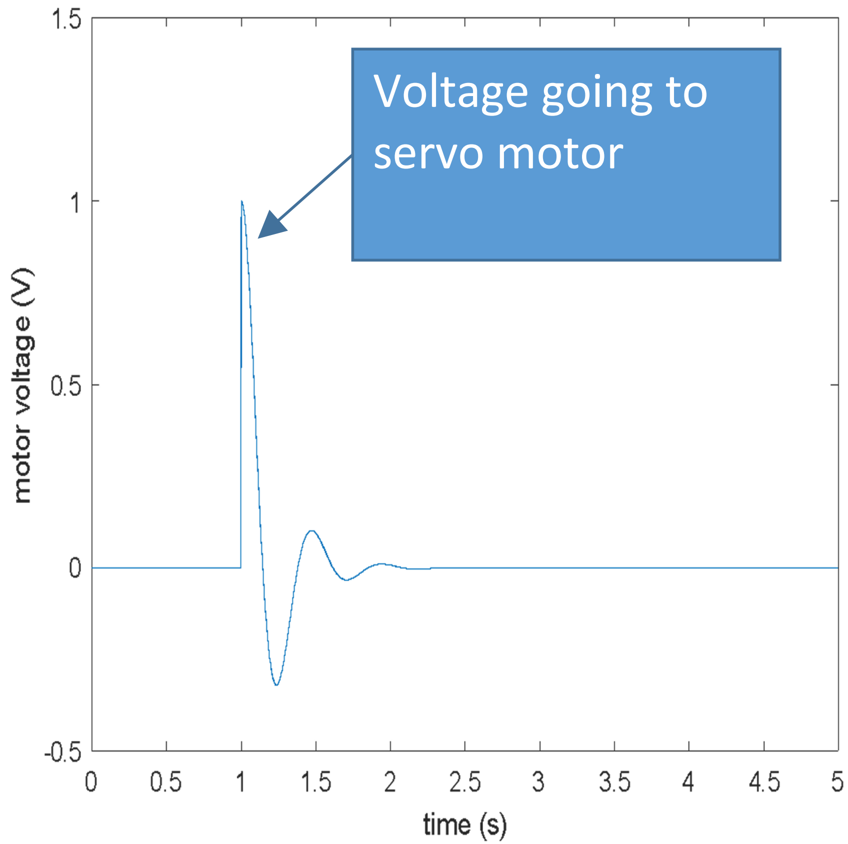


Response: P for servo motor position control

Servo response tracks desired servo angle well, but there is a large overshoot.

Control effort, i.e. voltage going to servo motor, is smooth





P control: take home message

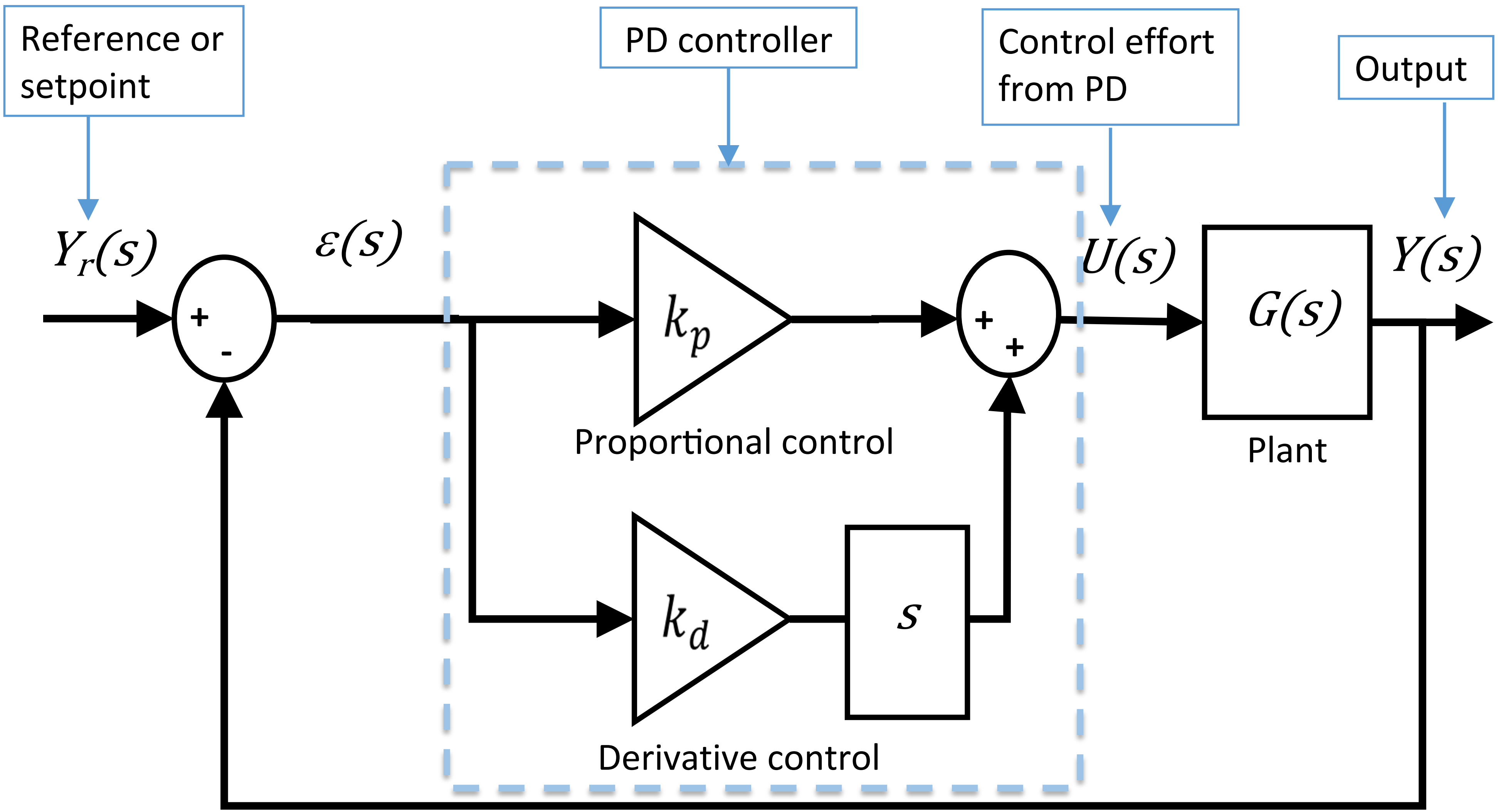
Benefits

- Simple control
- "Good enough" for many systems
  - *e.g.* systems with an integrator in their plant

Drawbacks

- Can result in steady-state error
  - When plant has no integrator
  - When system has friction
- Can results in large overshoot

PD control


Reference or setpoint
PD controller
Control effort from PD
Output
$Y_r(s)$
$\varepsilon(s)$
+
-
k_p
Proportional control
+
+
$U(s)$
$G(s)$
$Y(s)$
Plant
k_d
s
Derivative control

Equation of the PD control law and transfer function

Parallel form

In the time domain:

$$u(t) = k_p\, \varepsilon(t) + k_d \frac{d\varepsilon(t)}{dt}$$

In the Laplace domain:

$$U(s) = \left(k_p + k_d s\right)\varepsilon(s)$$

$$C(s) = k_p + k_d s$$

k_p is the proportional gain
k_d is the derivative gain

$$K_c = k_p$$

$$T_d = \frac{k_d}{k_p}$$

Form used in the industry

In the time domain:

$$u(t) = K_c \left(\varepsilon(t) + T_d \frac{d\varepsilon(t)}{dt}\right)$$

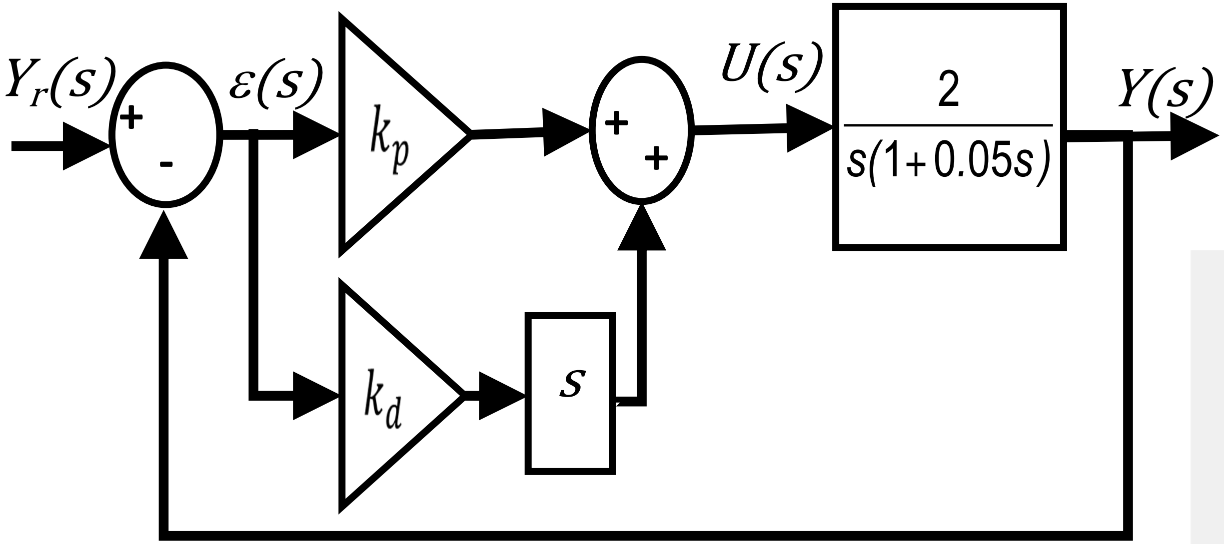
In the Laplace domain:

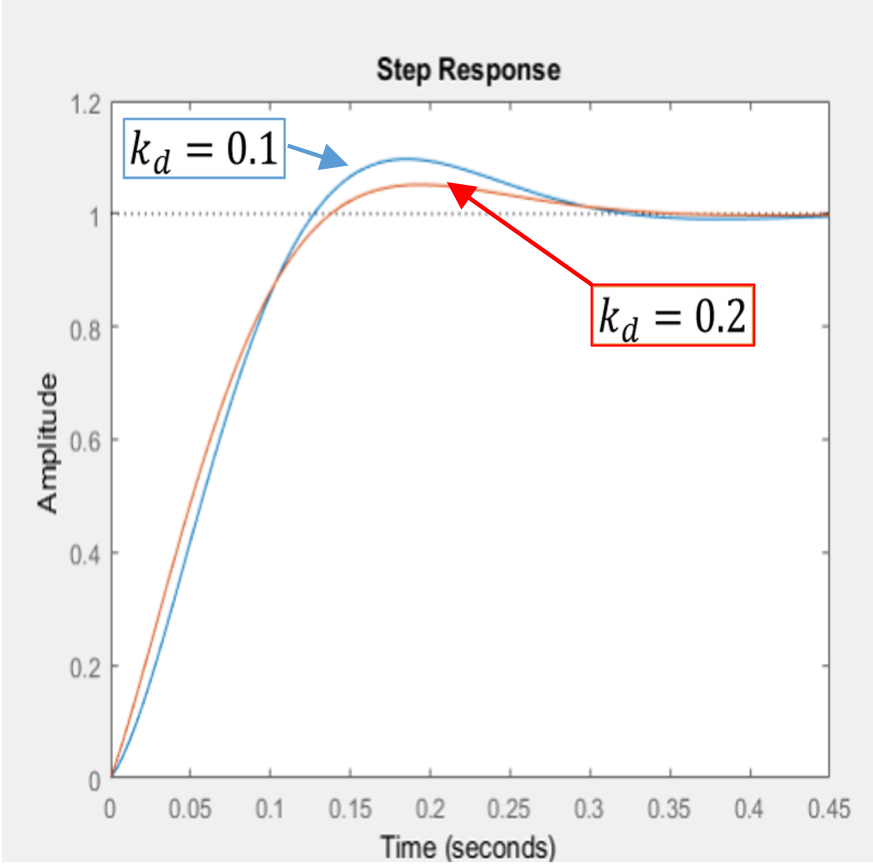
$$U(s) = K_c\left(1 + T_d s\right)\varepsilon(s)$$

$$C(s) = K_c\left(1 + T_d s\right)$$

K_c is the proportional gain
T_d is the derivative time-constant

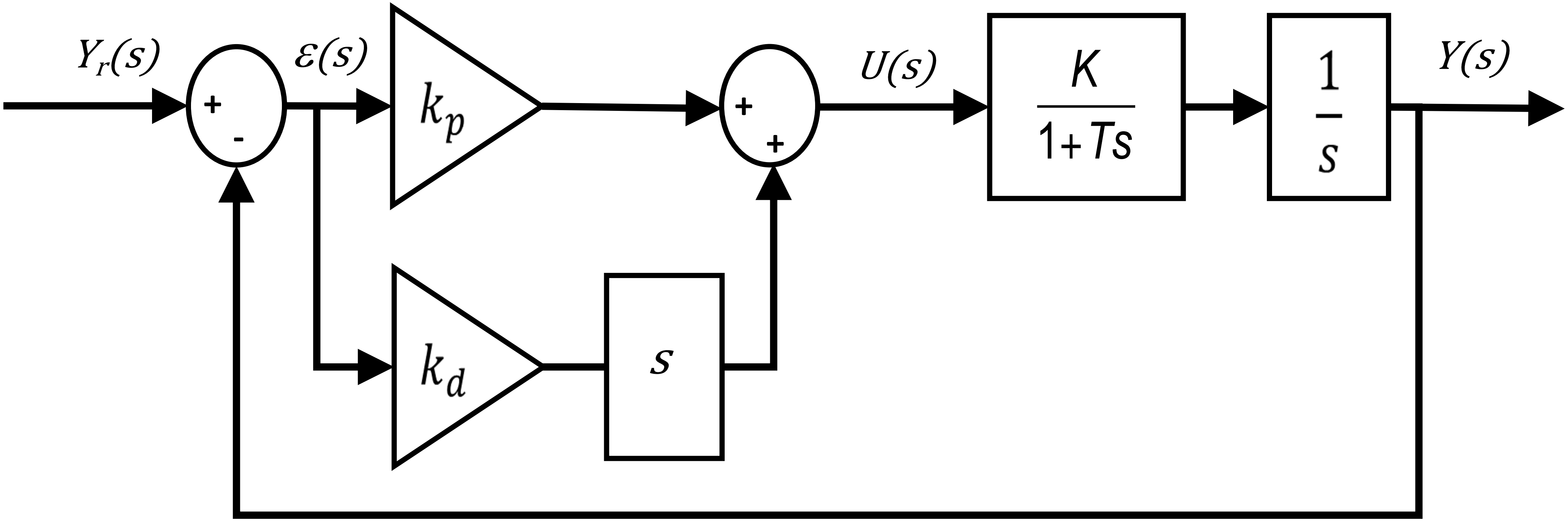
PD control: effect of derivative gain





- Set $k_p = 10$
- Increase k_d gradually. What do you notice?
  - Overshoot decreases
  - Peak time increases, i.e. response slower

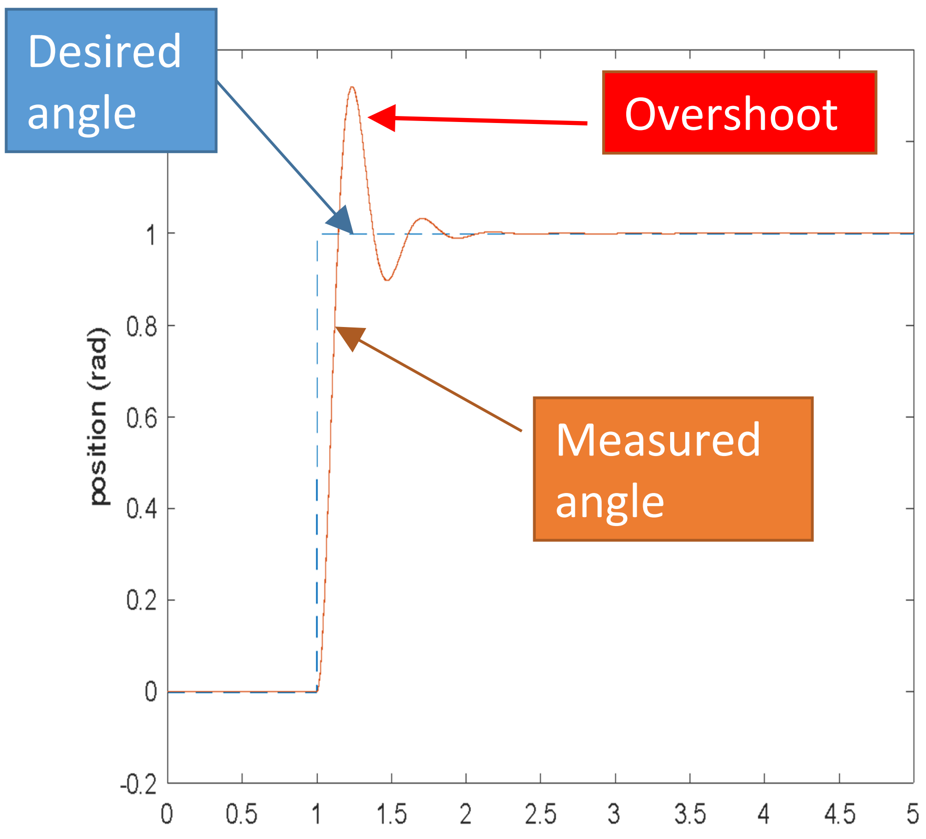
Example: PD for servo motor position control

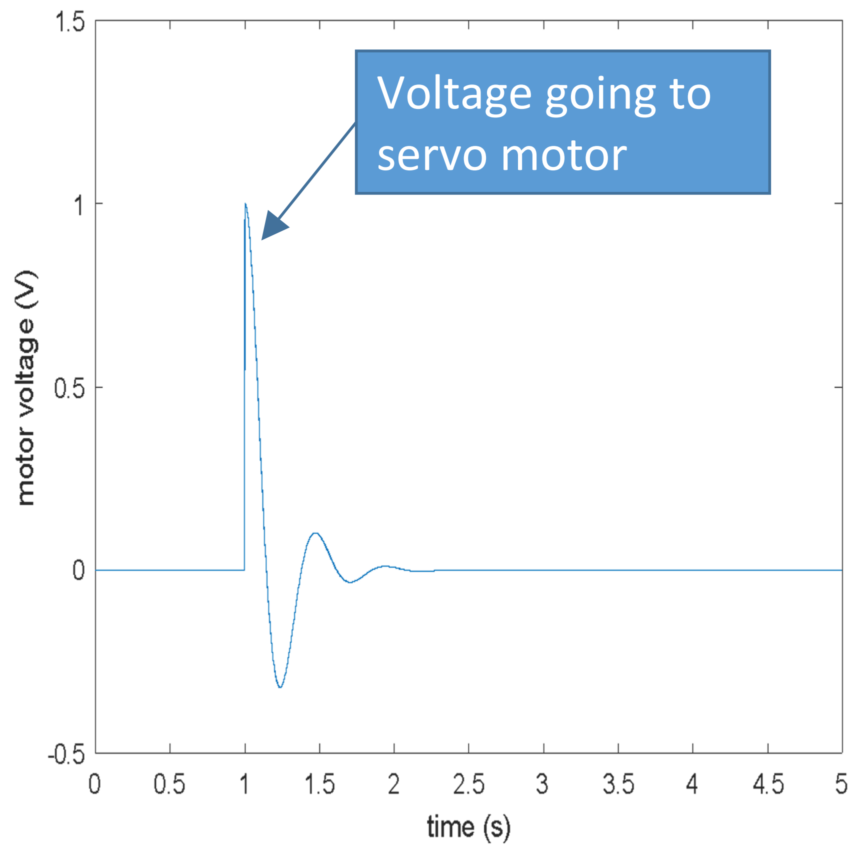


Recall: response with simple P servo control

Servo response tracks desired servo angle well, but there is a large overshoot

Control effort, i.e. voltage going to servo motor, is smooth

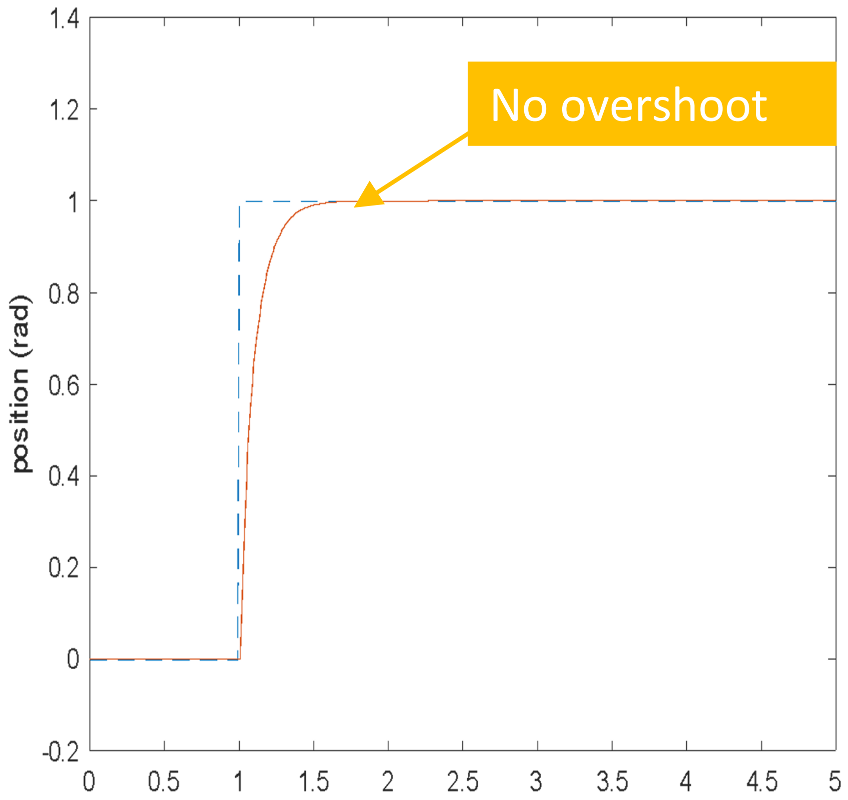


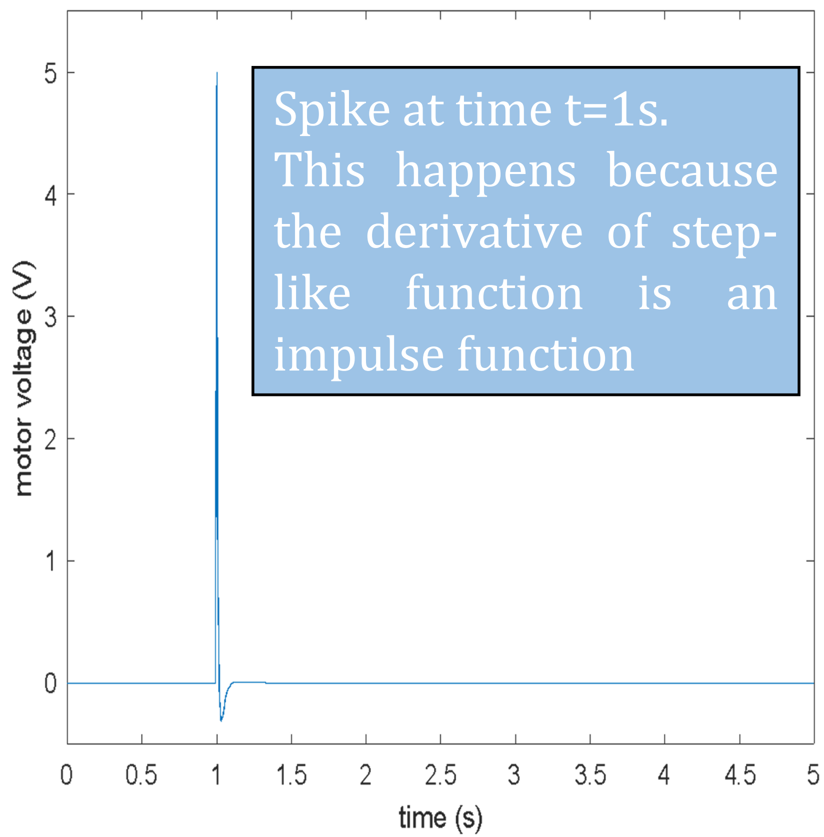


Now response with PD control

Adding derivative control lowers or removes the overshoot, but it slows down the response (i.e. increases peak/rise time)

Control effort, i.e. voltage going to servo, is smooth. But it is saturating the actuator at 5V

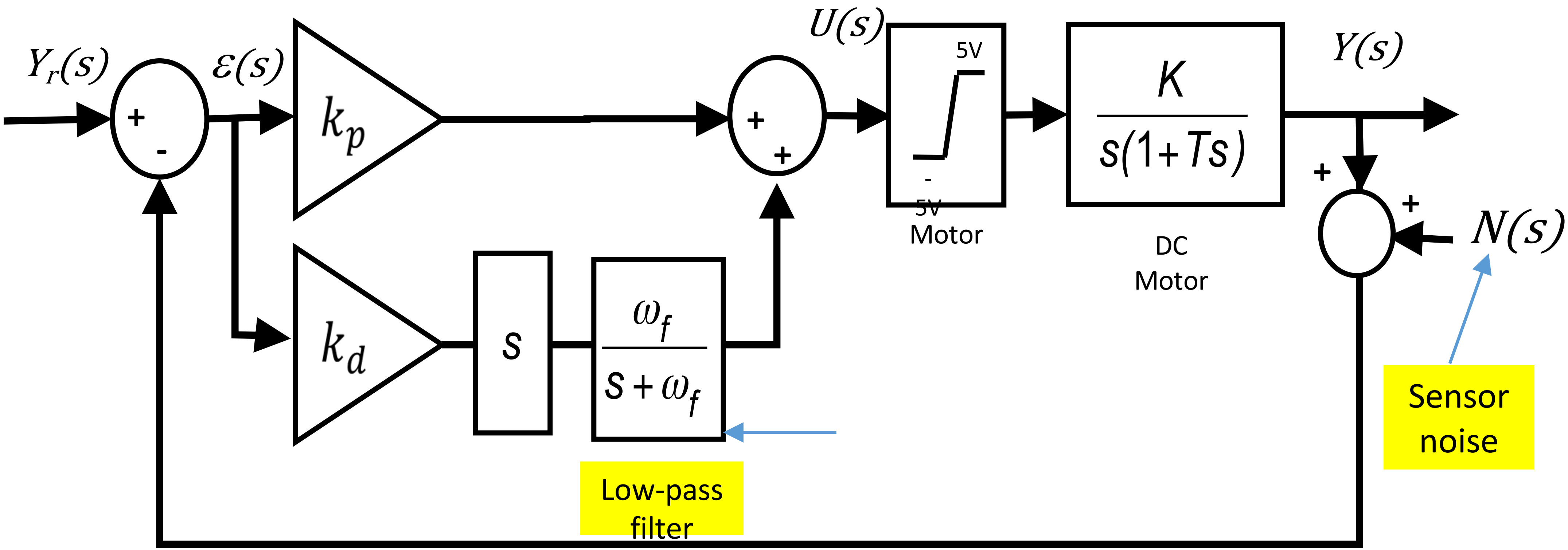




PD with low-pass filter

Goal of the filter: to reduce the effect of sensor noise

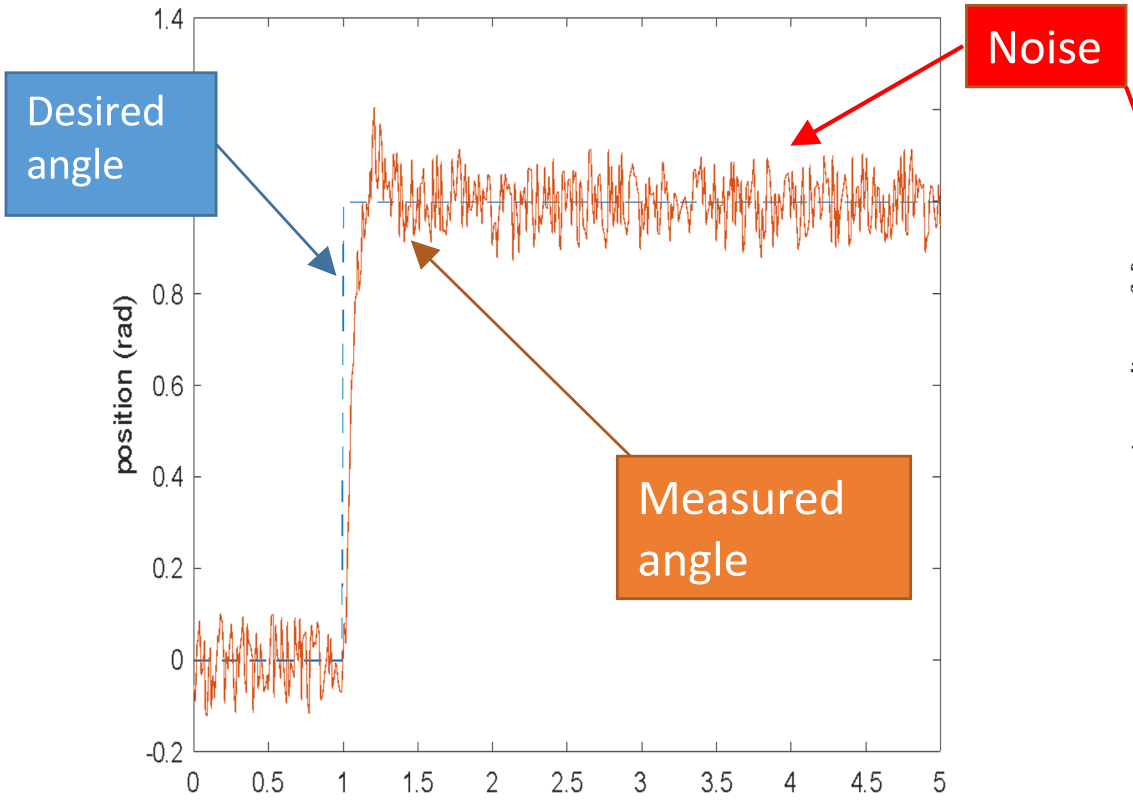
Example: PD with filtering for servo motor position control

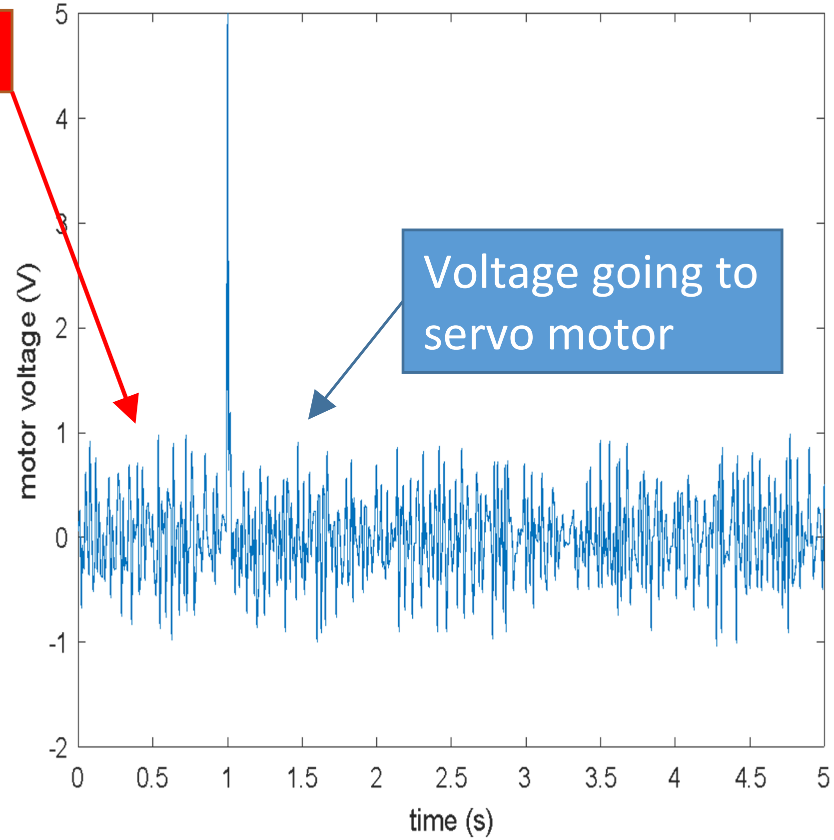


PD control with NO filtering

Servo response tracks desired servo angle well, but there is a noise

Control effort, i.e. voltage going to servo, is noisy

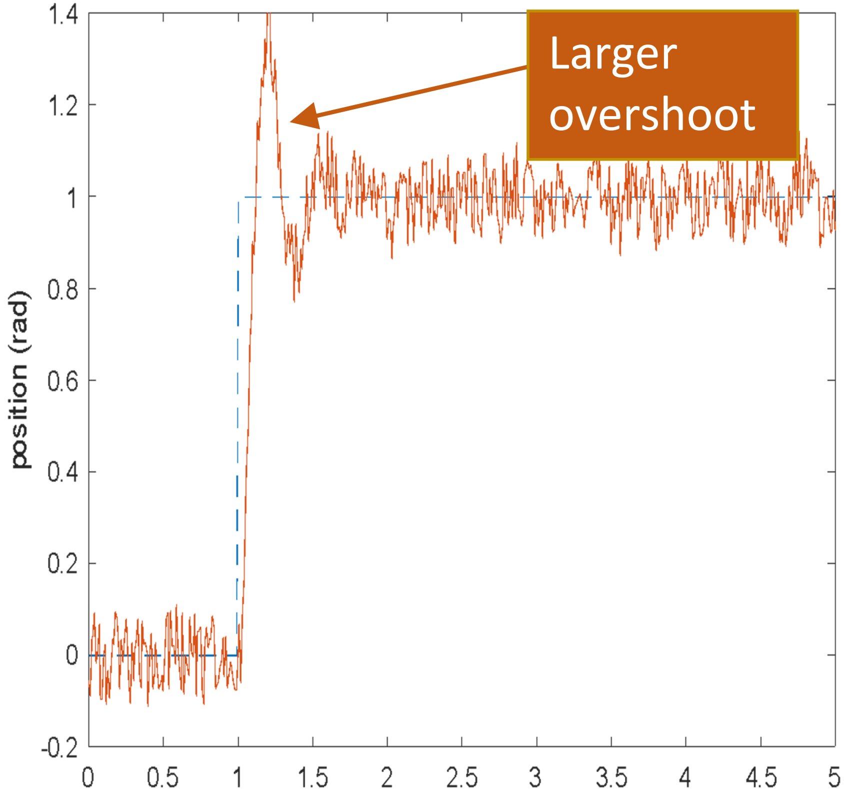




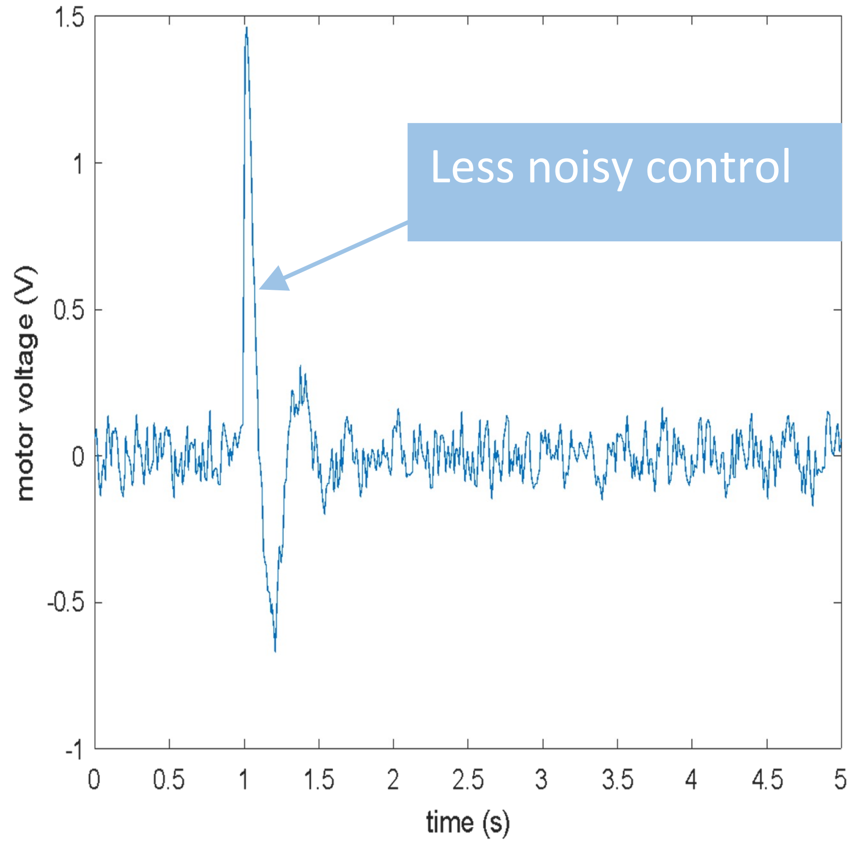
H. Garnier

PD control with low-pass filtering

Adding filtering lowers noise, but it adds overshoot.



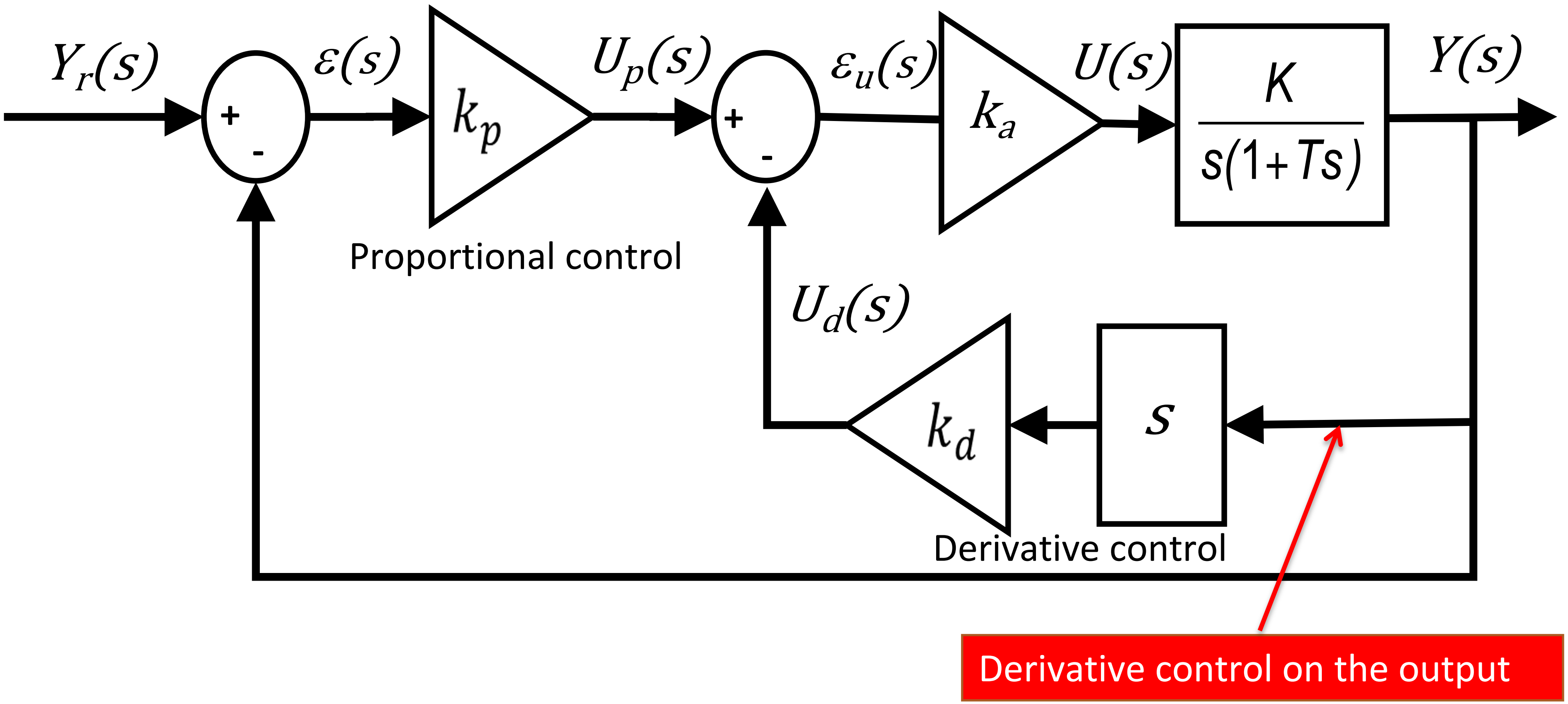
Control effort, i.e. voltage going to servo, is smoother (less noisy).



Variation of PD control

PD control with derivative effect on the output

Goal: to avoid the spike effect after a step change on the setpoint



PD control: take home message

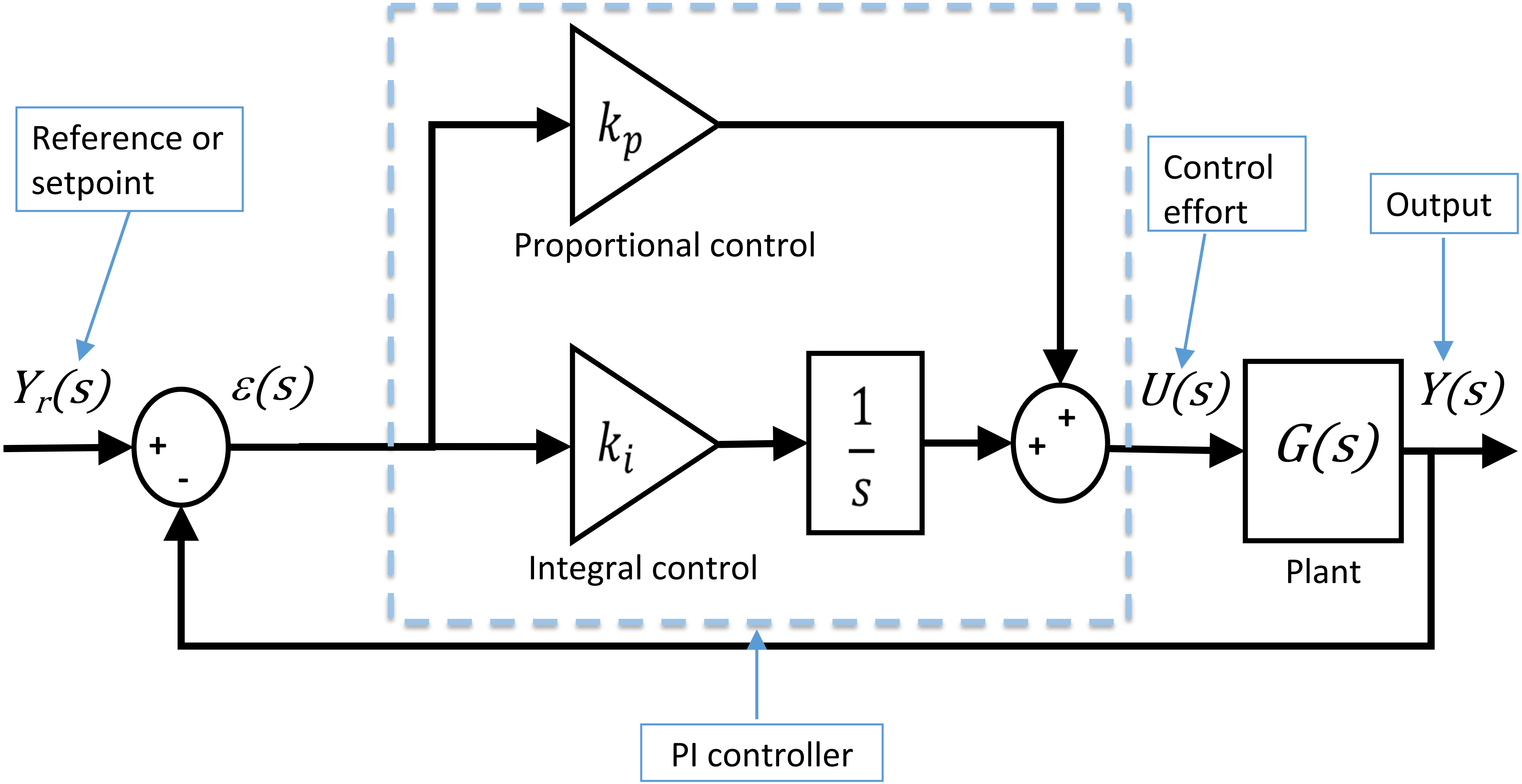
Benefits

- Removes overshoot
- Derivative plus low-pass filter can mitigate noisy output measurement effects
- Derivative control on the output to reduce the spike response after a setpoint step change

Drawbacks

- Can make output and control input noisy
  - *e.g.* due to sensor noise used in feedback
- Filtering slows down response and may result in overshoot

PI control



Reference or setpoint
$Y_r(s)$
+
-
$\varepsilon(s)$
k_p
Proportional control
k_i
Integral control
$\frac{1}{s}$
+
+
Control effort
$U(s)$
$G(s)$
Plant
Output
$Y(s)$
PI controller

Equation of the PI control law and transfer function

Parallel form

In the time domain:

$$u(t) = k_p\,\varepsilon(t) + k_i \int_0^t \varepsilon(\tau)d\tau$$

In the Laplace domain:

$$U(s) = \left(k_p + \frac{k_i}{s}\right)\varepsilon(s)$$

$$C(s) = k_p + \frac{k_i}{s}$$

k_p is the proportional gain
k_i is the integral gain

$$K_c = k_p$$

$$T_i = \frac{k_p}{k_i}$$

Form used in the industry

In the time domain:

$$u(t) = K_c\left(\varepsilon(t) + \frac{1}{T_i}\int_o^t \varepsilon(\tau)d\tau\right)$$

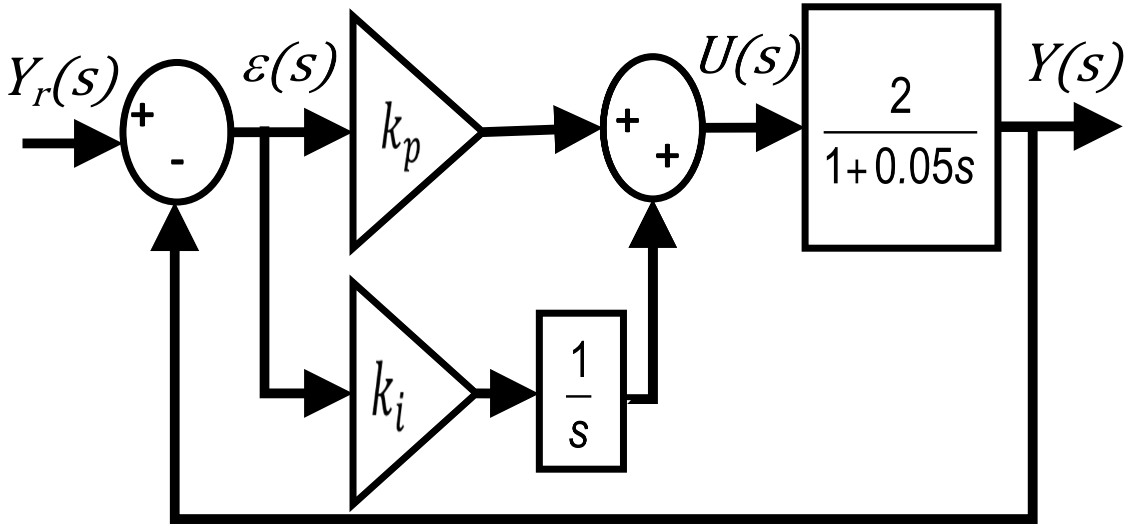
In the Laplace domain:

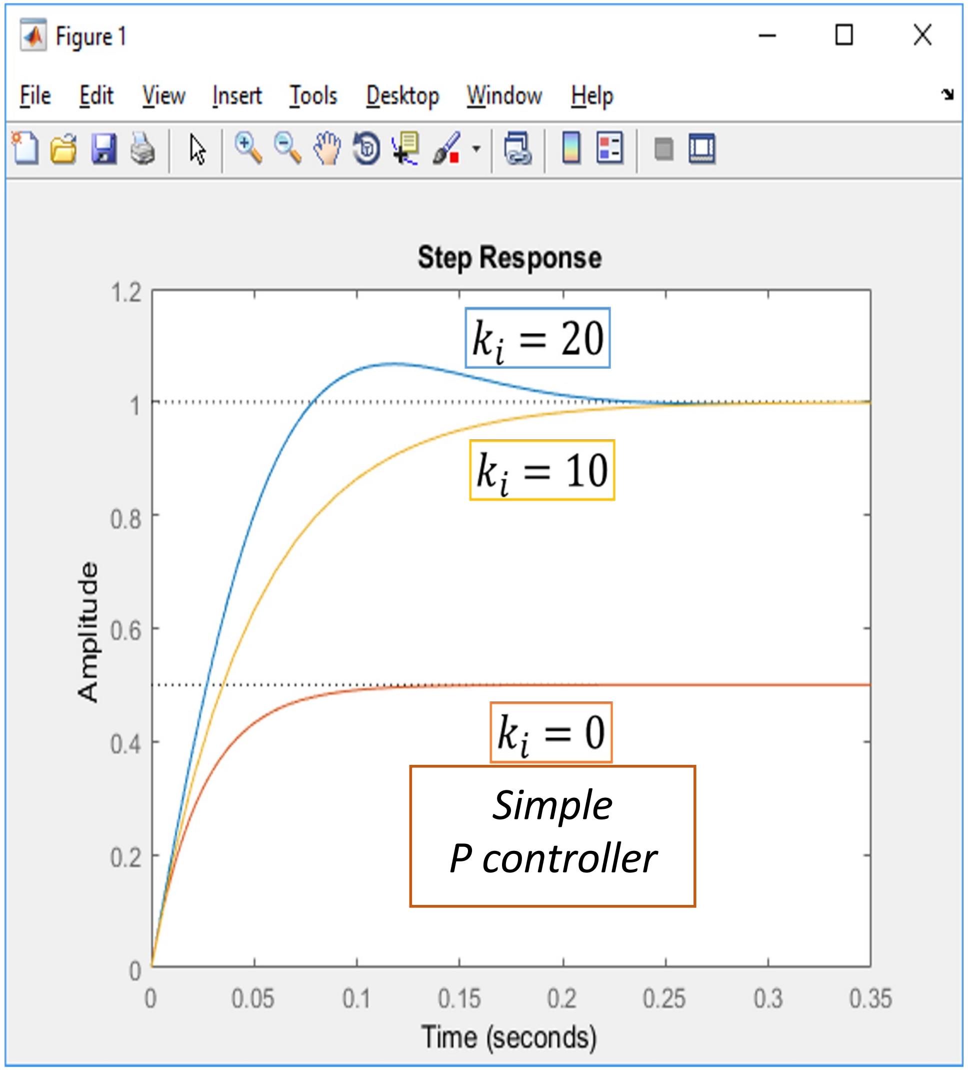
$$U(s) = K_c\left(1 + \frac{1}{T_i s}\right)\varepsilon(s)$$

$$C(s) = K_c\left(1 + \frac{1}{T_i s}\right)$$

K_c is the proportional gain
T_i is the integral time-constant

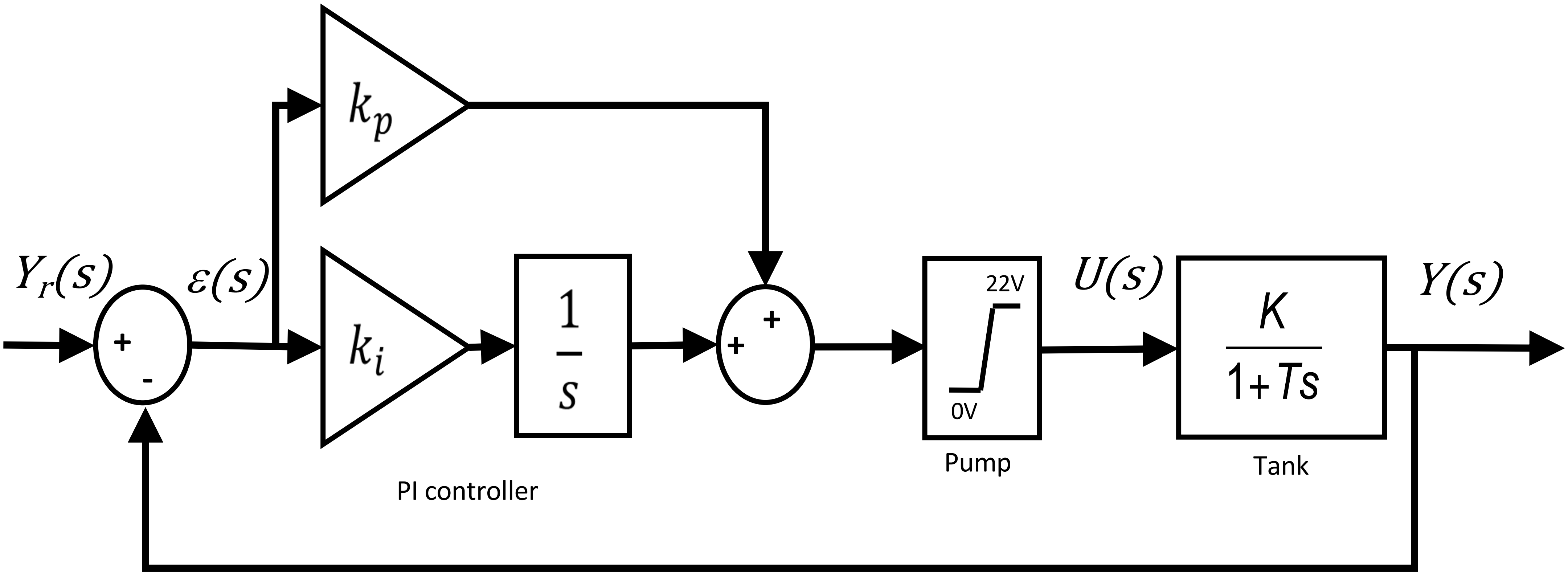
PI control: effect of integral gain





- Set $k_p = 0.5$
- Increase k_i gradually. What do you notice?
  - Steady-state error **decreases**
  - Response becomes **faster**, i.e. peak time decreases
  - Overshoot increases, i.e. response slows a bit

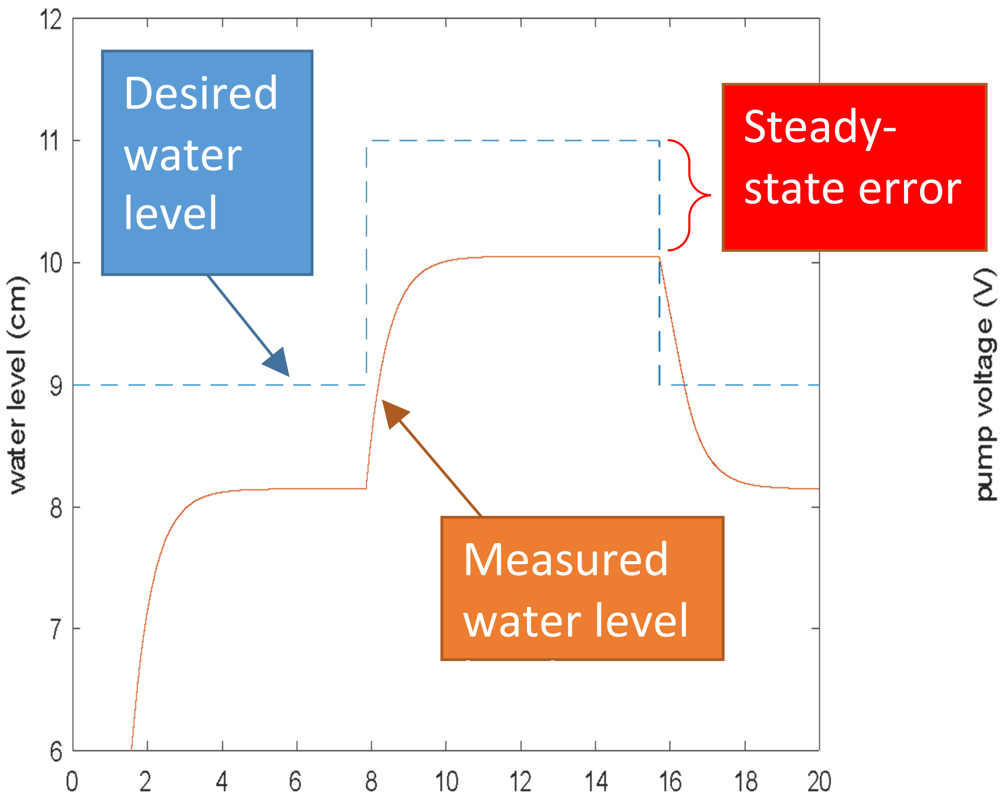
Example: tank level process control

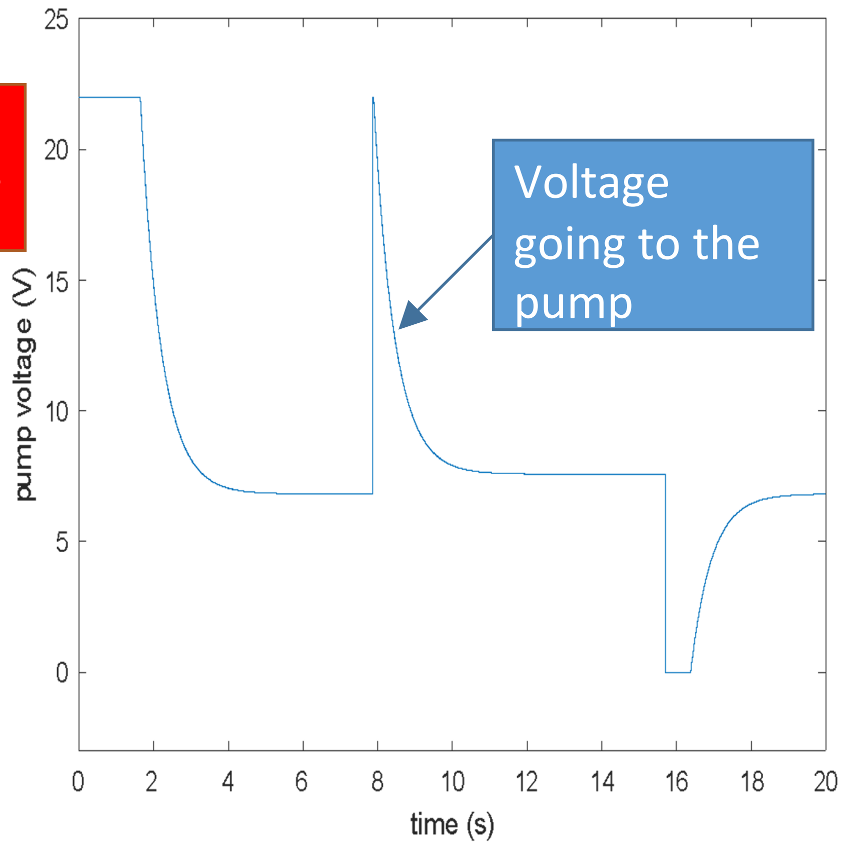


Recall the water tank level P control

Tank response does not track desired water level well

Control effort, i.e. voltage going to pump, is smooth

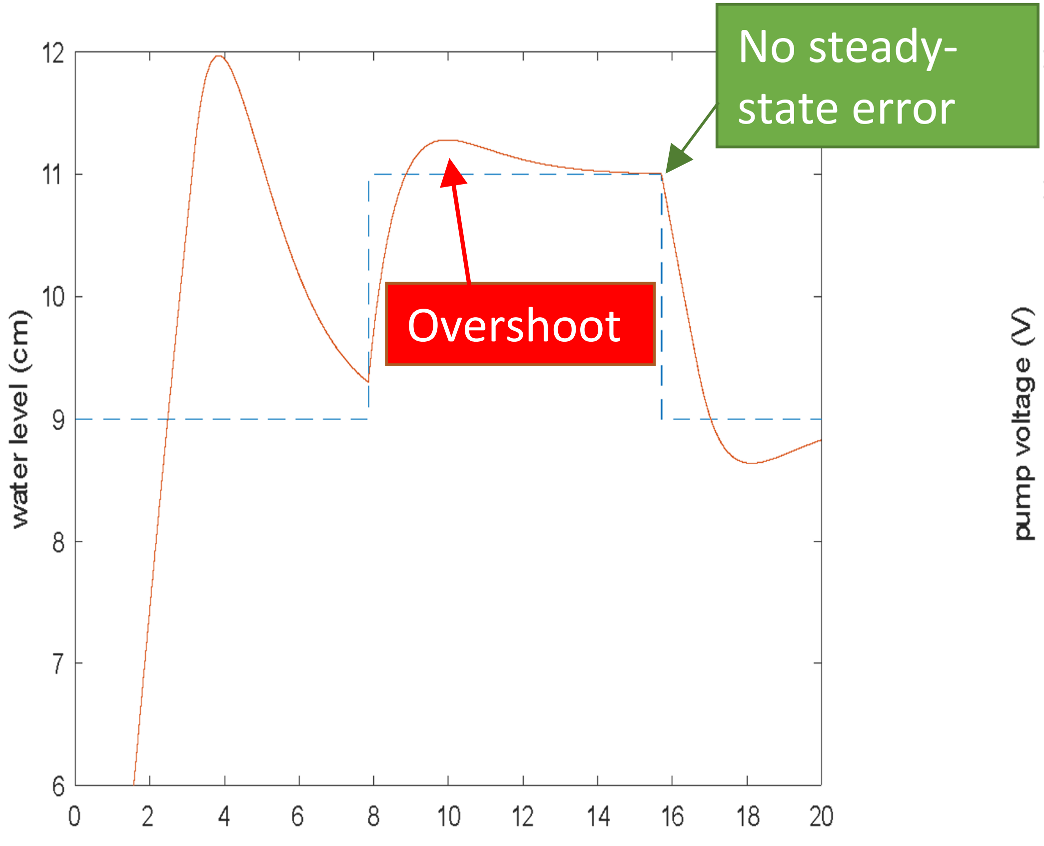


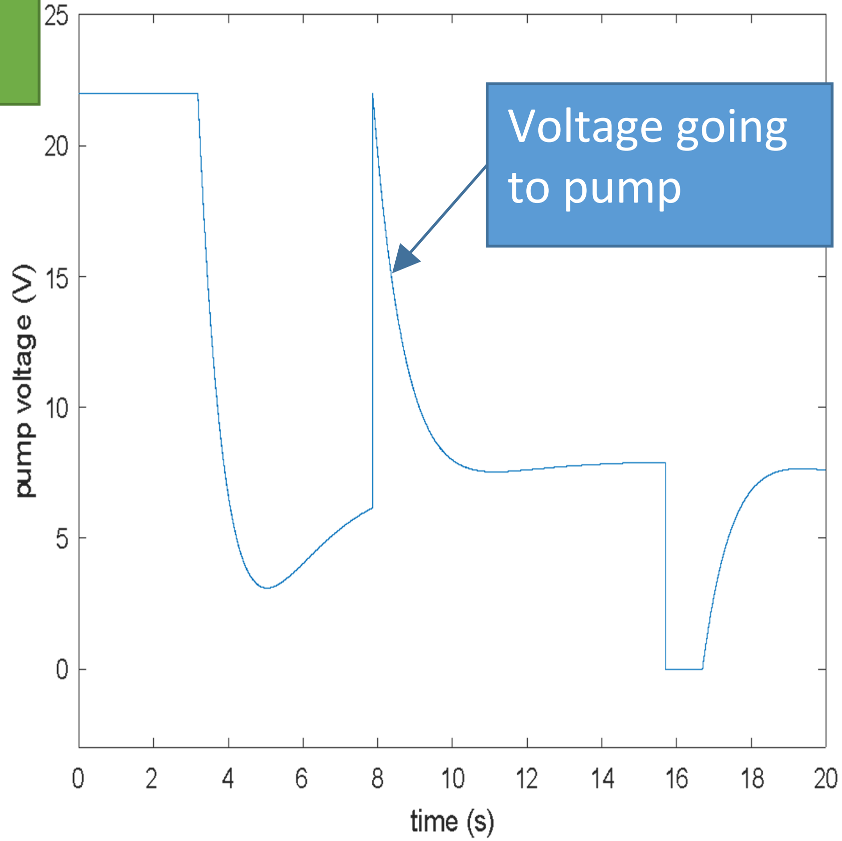


Water tank level PI control response

Tank response tracks desired water level well, but large overshoot

Control effort, i.e. voltage going to pump, is smooth but saturates actuator





PI control: take home message

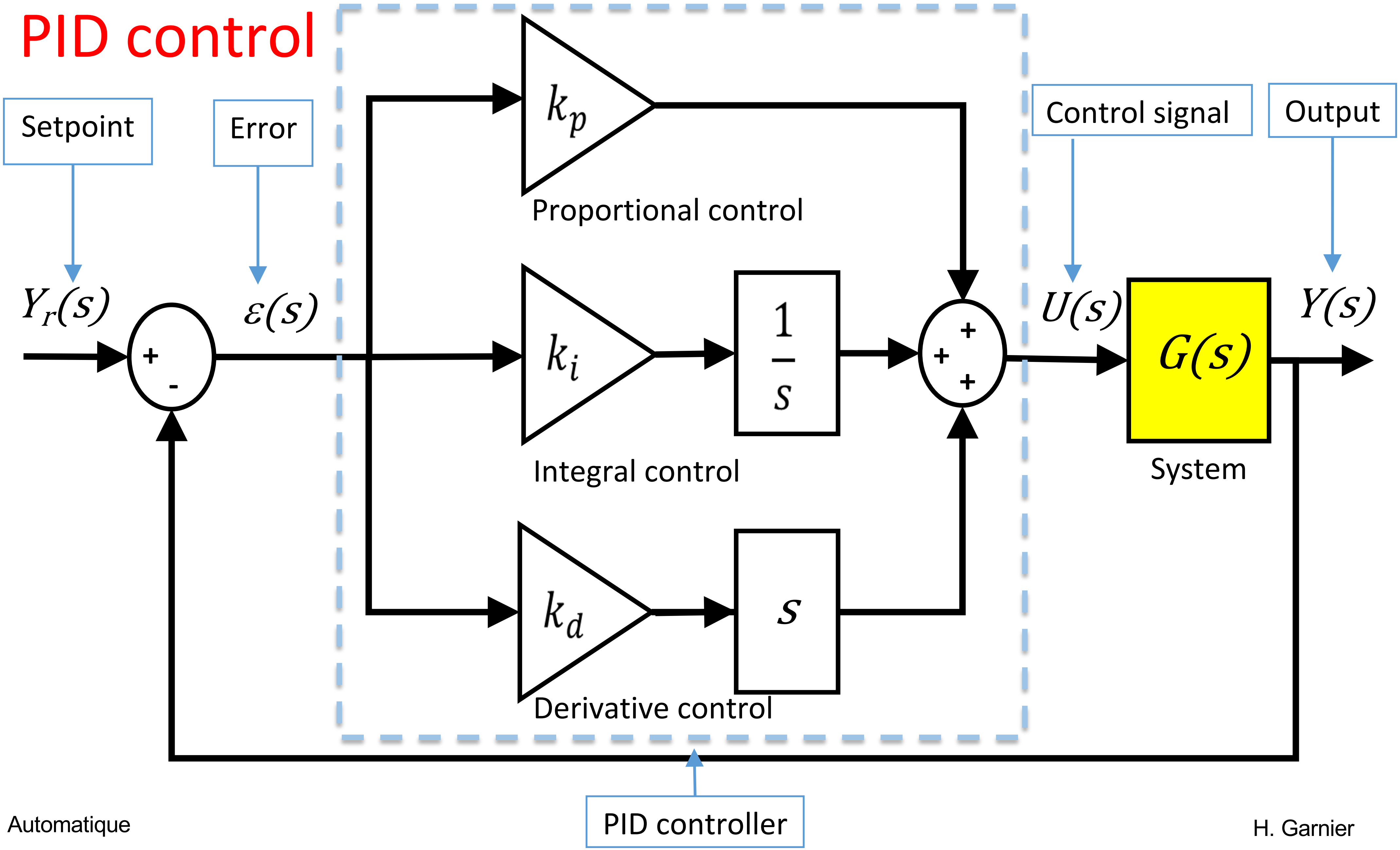
Benefits

- Removes steady-state error
- Can reject disturbances

Drawbacks

- Can lead to large overshoot in response when control signal becomes saturated, *i.e.* "integral windup"
- Improperly designed integral gain can lead to instabilities

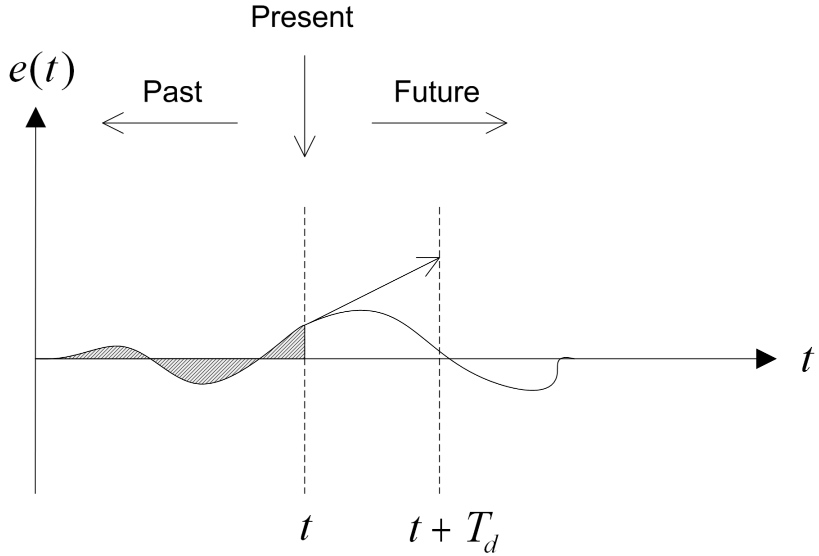
PID control


Setpoint
Error
Control signal
Output
$Y_r(s)$
+
-
$\varepsilon(s)$
k_p
Proportional control
k_i
$\frac{1}{s}$
Integral control
k_d
s
Derivative control
+
+
+
$U(s)$
$G(s)$
System
$Y(s)$
PID controller

PID terms

To make corrective effects:

- k_p based on present error
- k_i depends on past error
- k_d prediction of future error



present past future

$$u(t) = k_p\ \varepsilon(t) + k_i \int_0^t \varepsilon(\tau)d\tau + k_d\ \frac{\varepsilon(t)}{dt}$$

POLYTECH NANCY

UNIVERSITÉ DE LORRAINE

Equation of the PID control law and transfer function

Parallel form

In the time domain:

$$u(t) = k_p\,\varepsilon(t) + k_i \int_0^t \varepsilon(\tau)d\tau + k_d \frac{\varepsilon(t)}{dt}$$

In the Laplace domain:

$$U(s) = \left(k_p + \frac{k_i}{s} + k_d s\right)\varepsilon(s)$$

$$C(s) = k_p + \frac{k_i}{s} + k_d s$$

k_p is the proportional gain
k_i is the integral gain
k_d is the derivative gain

$$K_c = k_p$$

$$T_i = \frac{k_p}{k_i}$$

$$T_d = \frac{k_d}{k_p}$$

Form used in the industry

In the time domain:

$$u(t) = K_c\left(\varepsilon(t) + \frac{1}{T_i}\int_o^t \varepsilon(\tau)d\tau + T_d \frac{\varepsilon(t)}{dt}\right)$$

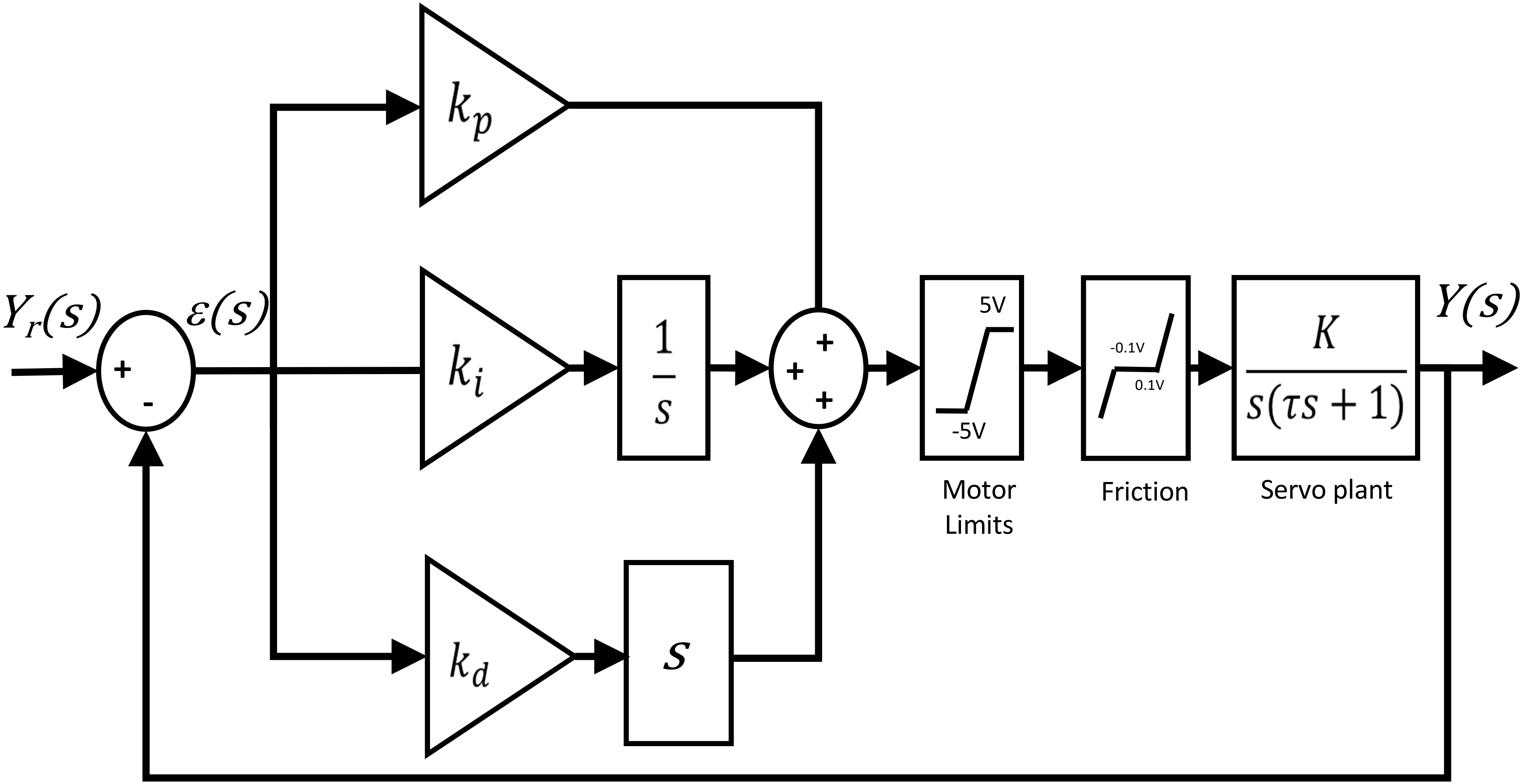
In the Laplace domain:

$$U(s) = K_c\left(1 + \frac{1}{T_i s} + T_d s\right)\varepsilon(s)$$

$$C(s) = K_c\left(1 + \frac{1}{T_i s} + T_d s\right)$$

K_c is the proportional gain
T_i is the integral time-constant
T_d is the derivative time-constant

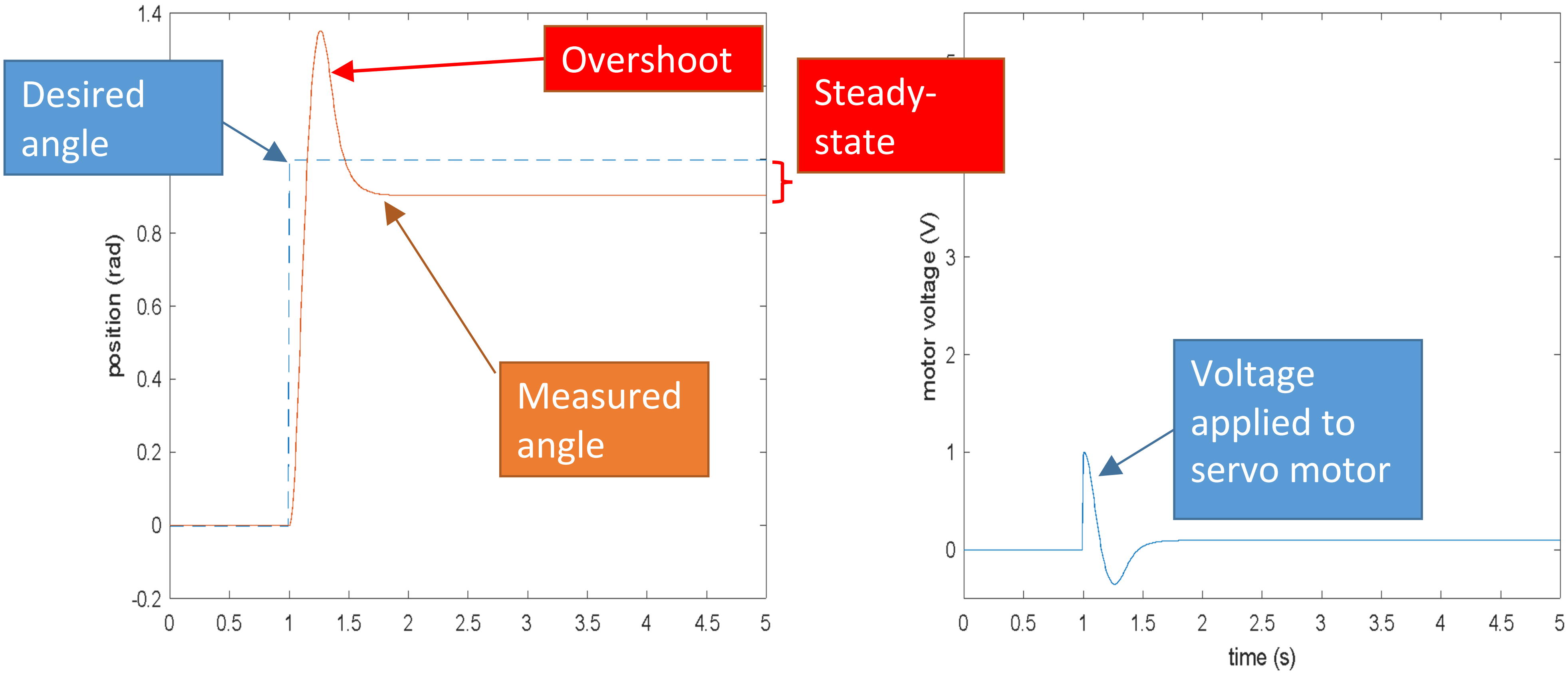
Example: PID for servo motor position control



Response with simple P control

Servo response tracks desired servo angle well, but there is a large overshoot and steady-state error (due to friction).

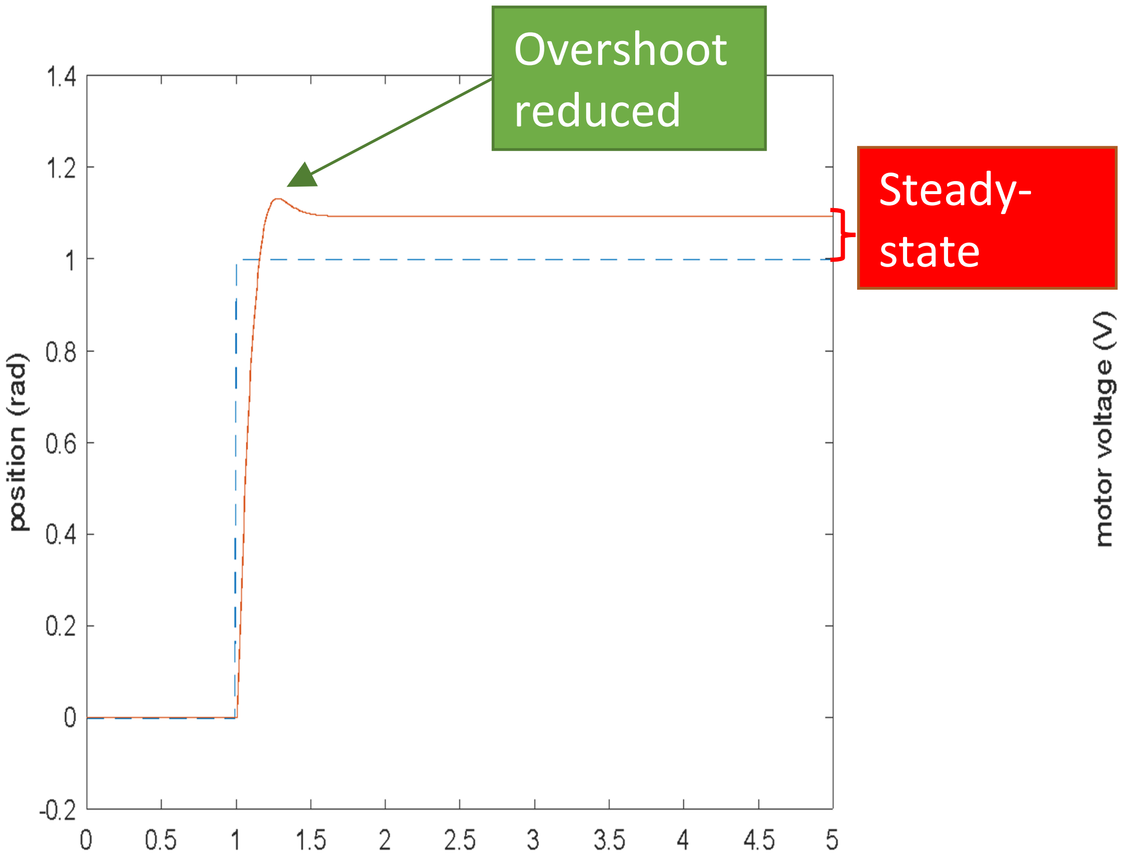
Control effort, i.e. voltage going to servo, is smooth

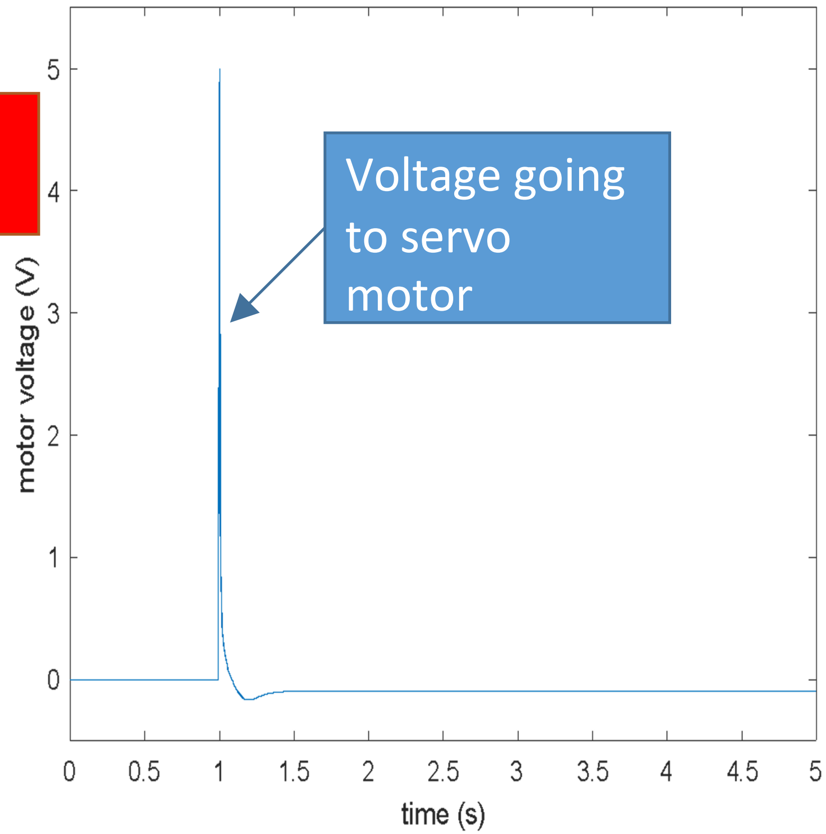


Response with PD control

Servo response tracks desired servo angle well, overshoot is reduced, but there is a steady-state error.

Control effort, i.e. voltage going to servo, is smooth. But it is saturating the actuator at 5V.

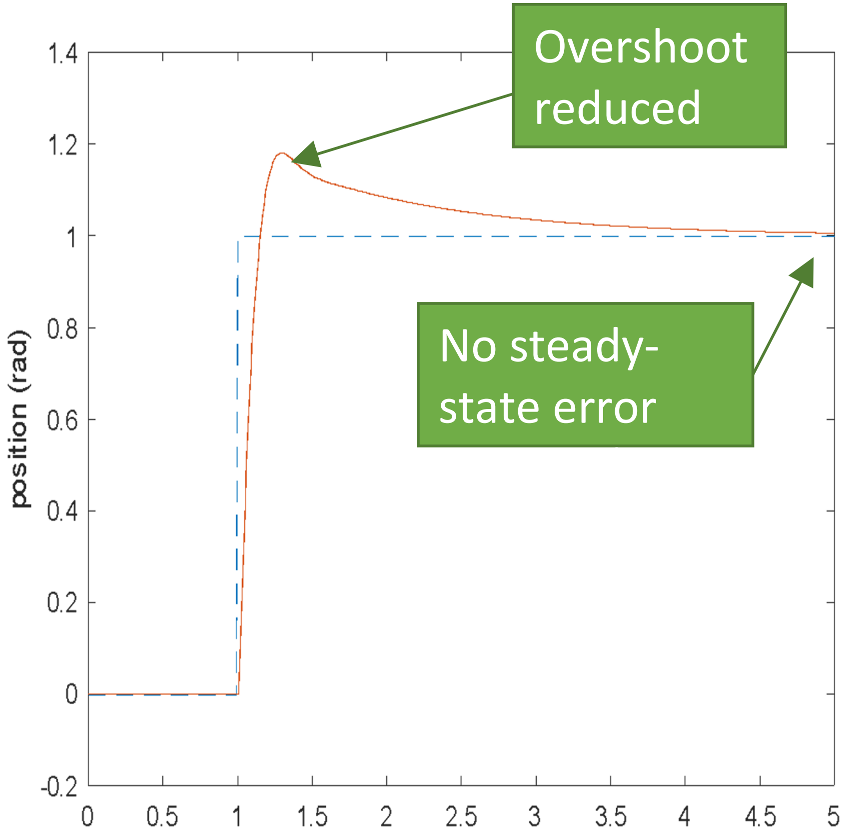


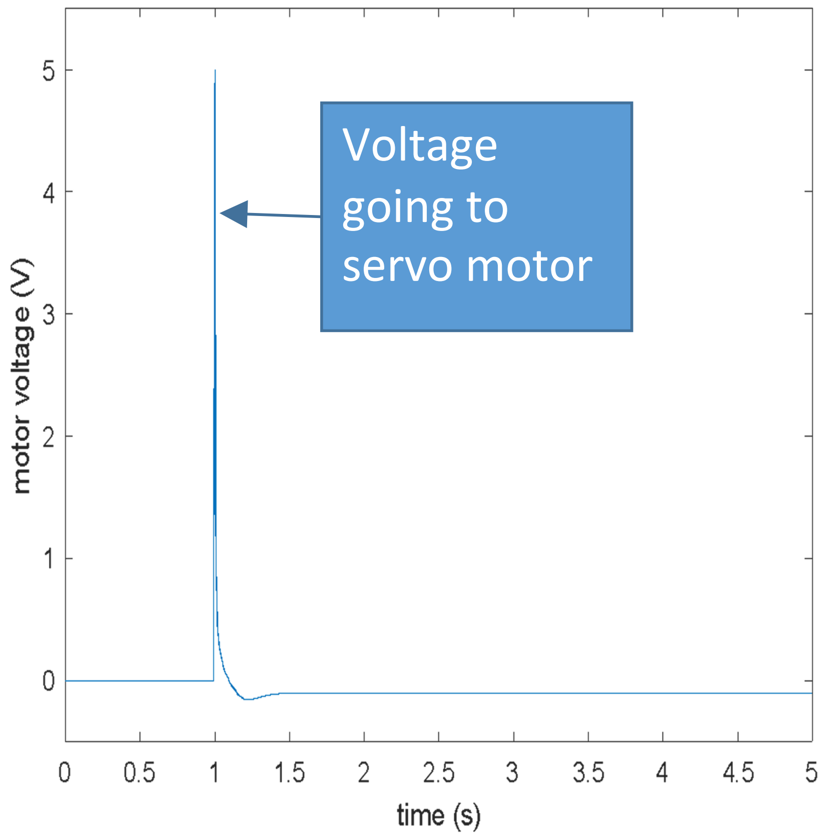


Response with PID control

Servo response tracks desired servo angle well, overshoot is reduced, and steady-state error has been removed.

Control effort, i.e. voltage going to servo, is smooth. But it is saturating the actuator at 5V.





Practical form of PID controller used in the industry

$$C(s) = K_c\left(1 + \frac{1}{T_i s} + T_d s\right) = K_c\left(\frac{T_i T_d s^2 + T_i s + 1}{T_i s}\right)$$

Some controllers, in particular old PID version have a proportional band (*PB*) setting instead of a controller gain K_c

$$PB = \frac{100}{K_c} \qquad C(s) = \frac{100}{PB}\left(1 + \frac{1}{T_i s} + T_d s\right)$$

where *PB* is the proportional mode and is defined as, “the percentage error that results in a 100% change in controller output”

The derivative term causes the gain to increase without bound as frequency goes up. Practical PID controllers limit this high frequency gain with a first-order low pass filter. A practical PID controller form is then the following

$$C(s) = K_c\left(\frac{T_i T_d s^2 + T_i s + 1}{T_i s}\right)\left(\frac{\omega_f}{s + \omega_f}\right) \qquad \frac{2}{T_d} \le \omega_f \le \frac{10}{T_d}$$

Summary: what do PID terms do?

Advantage

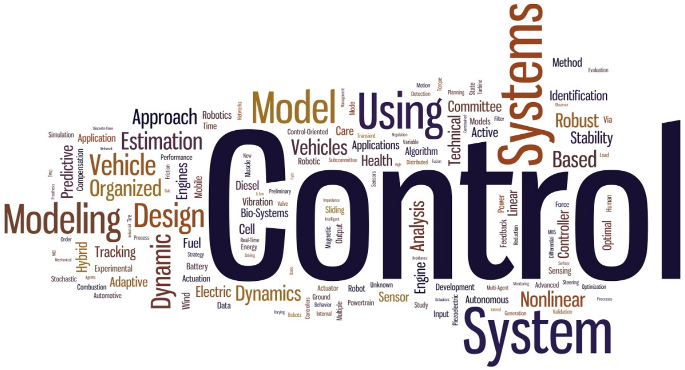
- Proportional (P)
  - Speeds up response
- Derivative (D)
  - Decreases overshoot
- Integral (I)
  - Cancels steady-state error
  - Rejects disturbance

Disadvantage

- Proportional (P)
  - Increases overshoot
- Derivative (D)
  - Slows down response
- Integral (I)
  - Slows down response

POLYTECH NANCY

UNIVERSITÉ DE LORRAINE



Automatique

Practical aspects of PID controllers

Anti-windup strategy

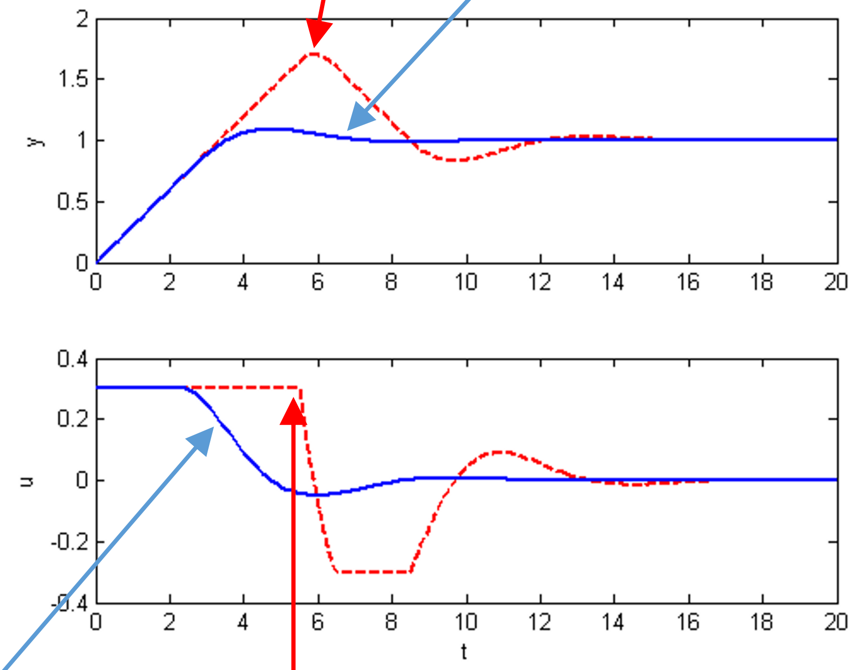
Hugues GARNIER

hugues.garnier@univ-lorraine.fr

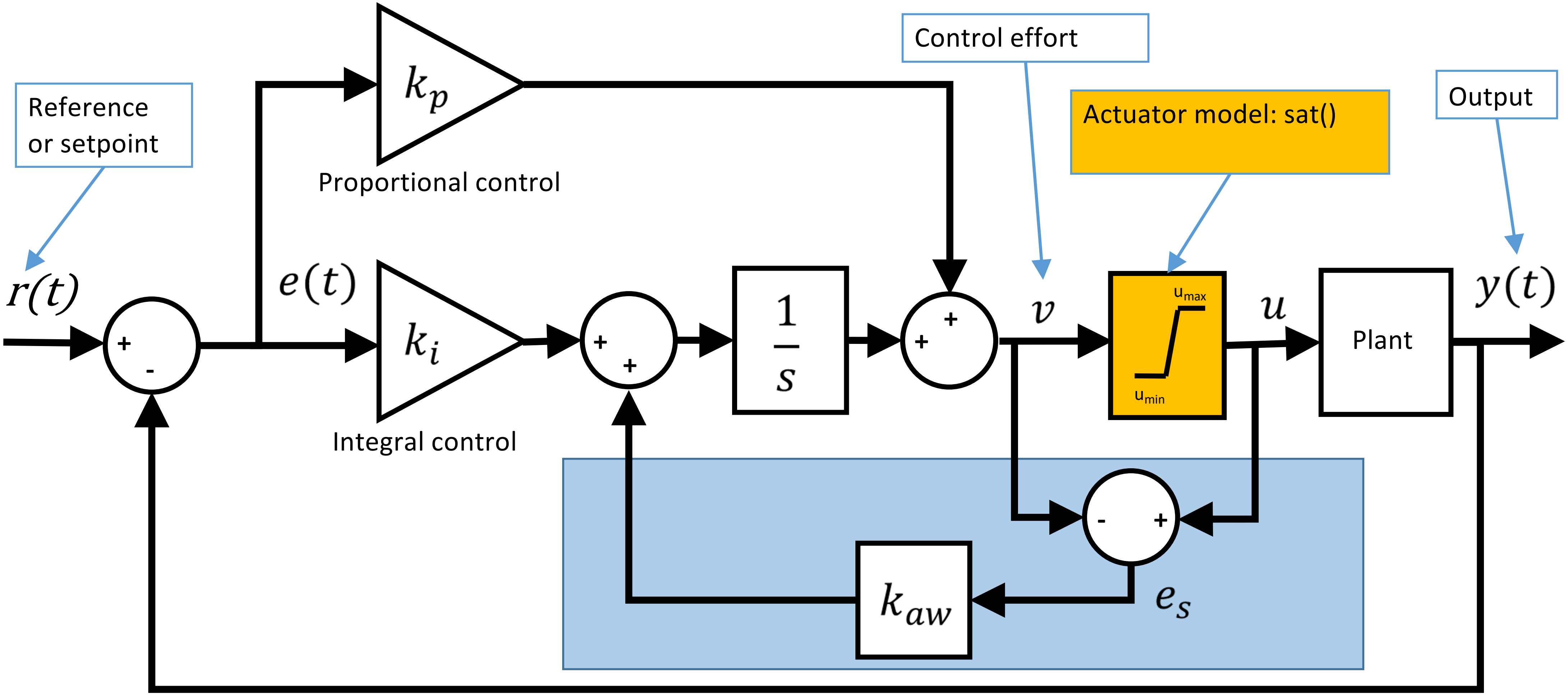
Version du 28 novembre 2024

Integral Windup Effect

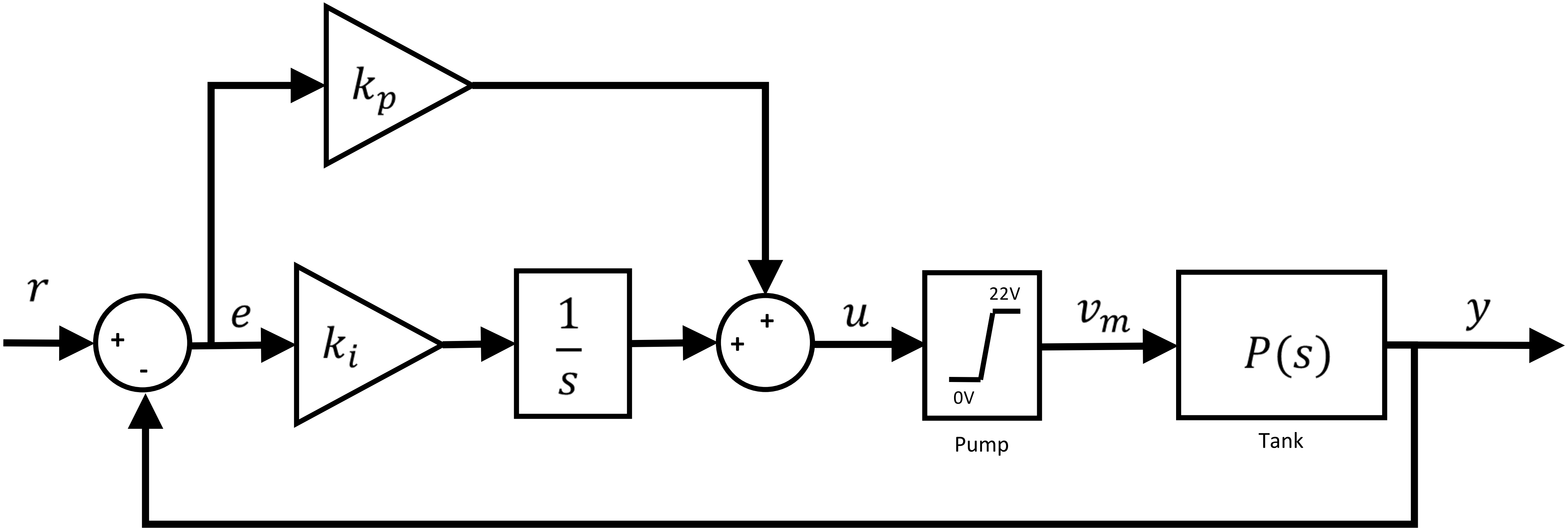
- When the control signal becomes saturated, the integrator keep outputting a control signal to compensate for the error
  - Output reaches setpoint at t = 4
  - … but integrator has stored so much energy, it overshoots
  - Control signal goes down only at t = 6



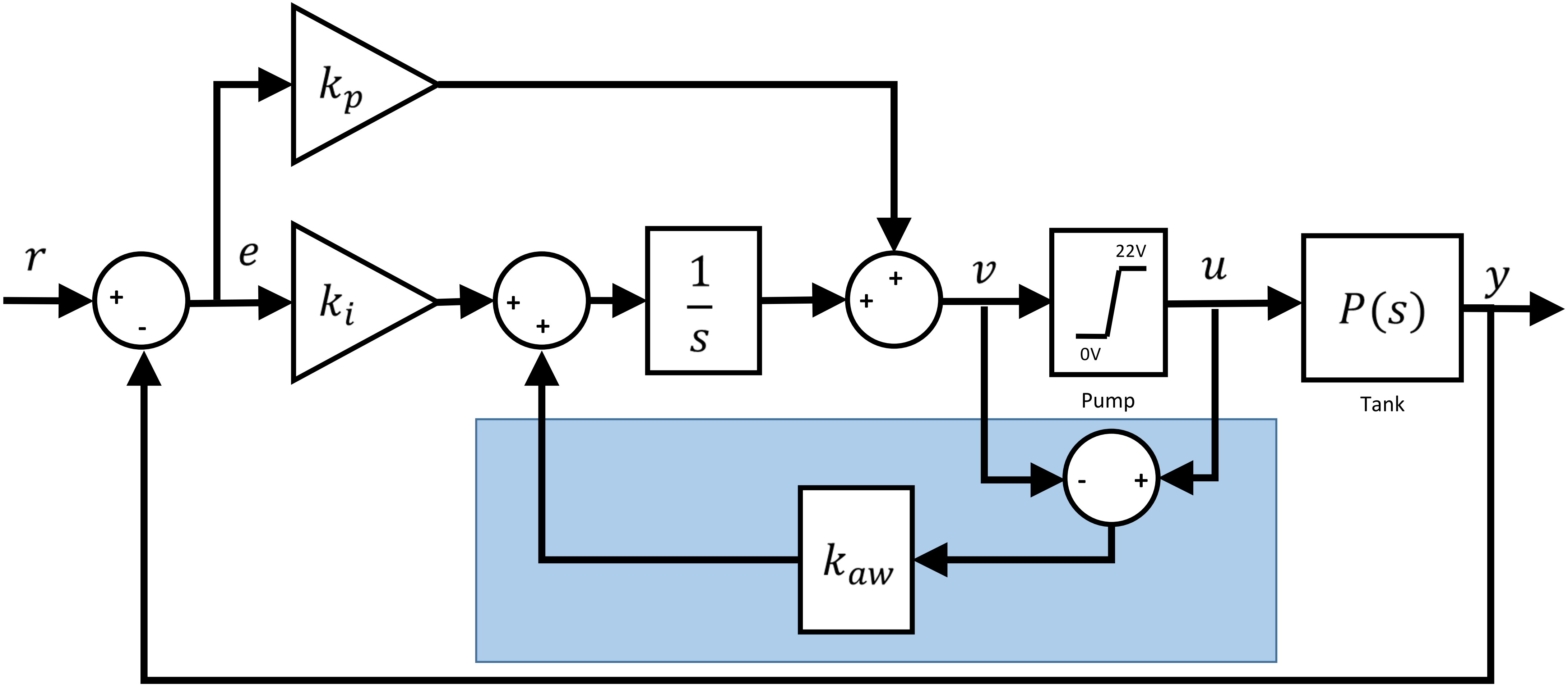
PI Feedback Loop with Integral Anti-Windup



Tank Level Process Control – without Anti-Windup



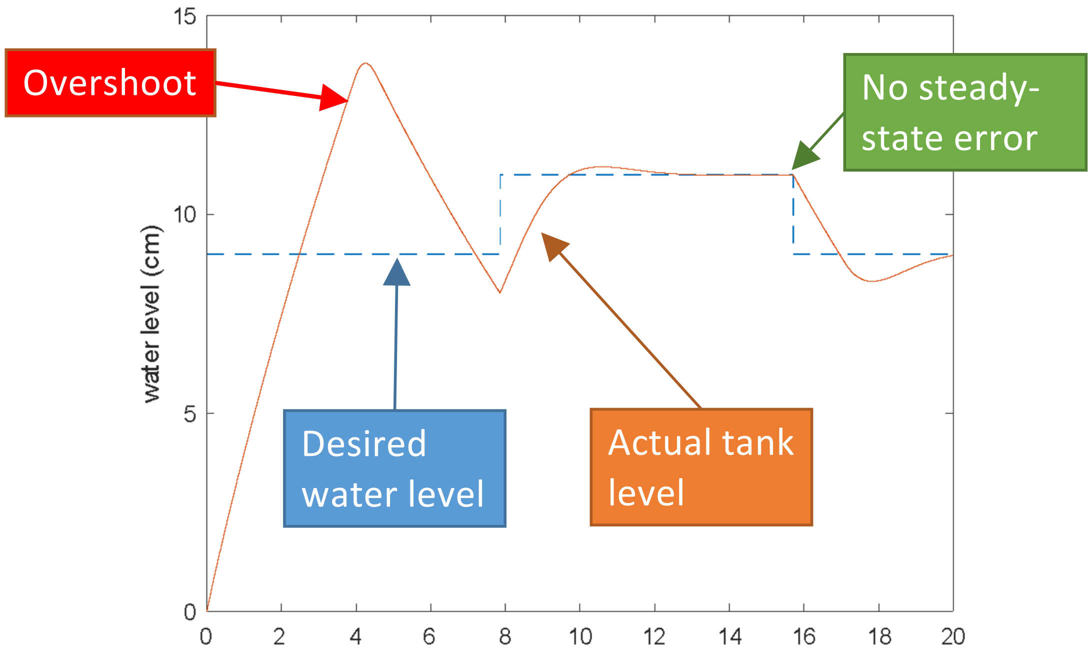
With Integrator Anti-Windup

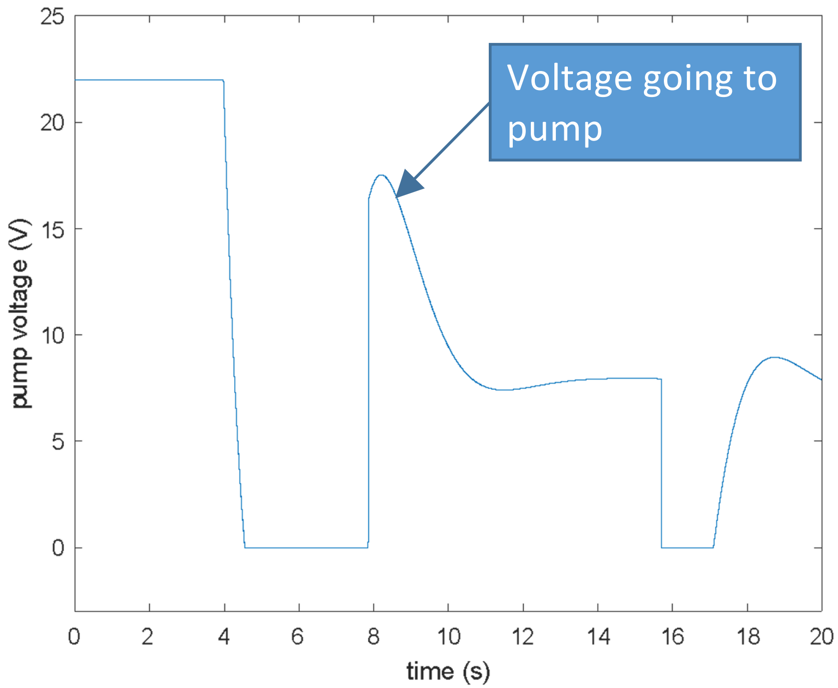


Recall: PI Control Response (without anti-windup)

Tank response tracks desired water level well, but large overshoot

Control effort, i.e. voltage going to pump, is smooth but saturates actuator

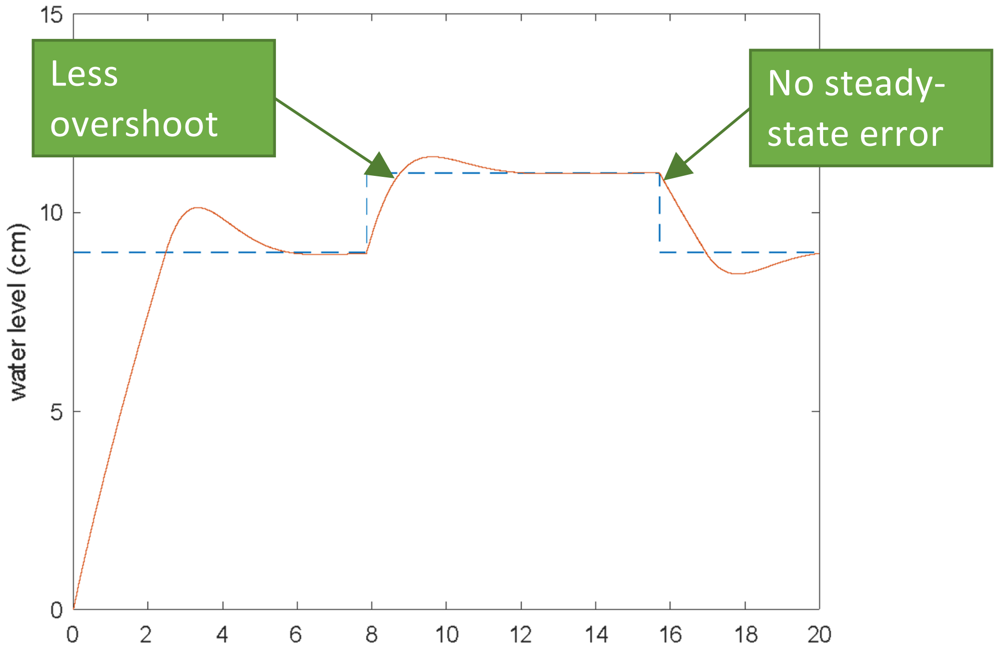


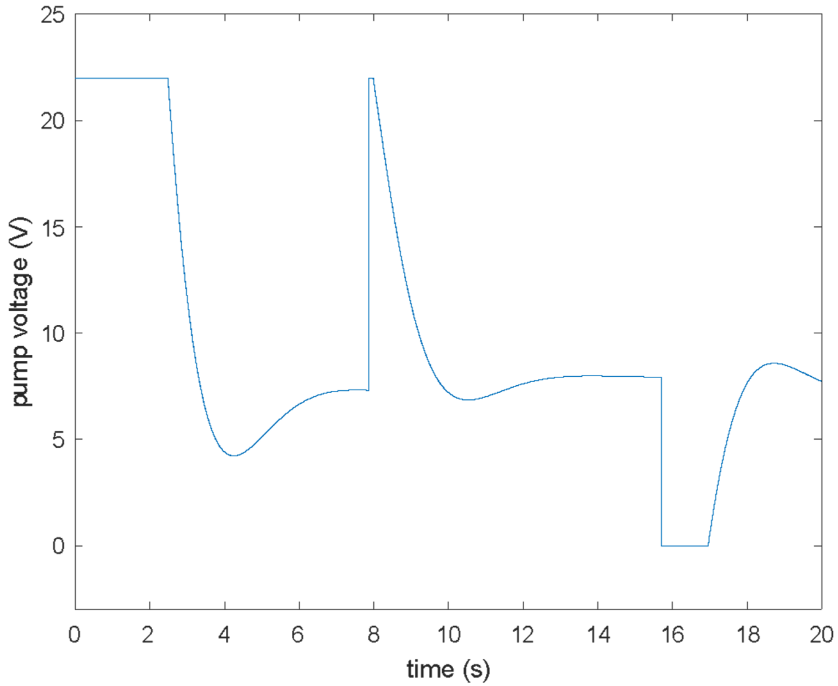


PI + Anti-Windup Control Response

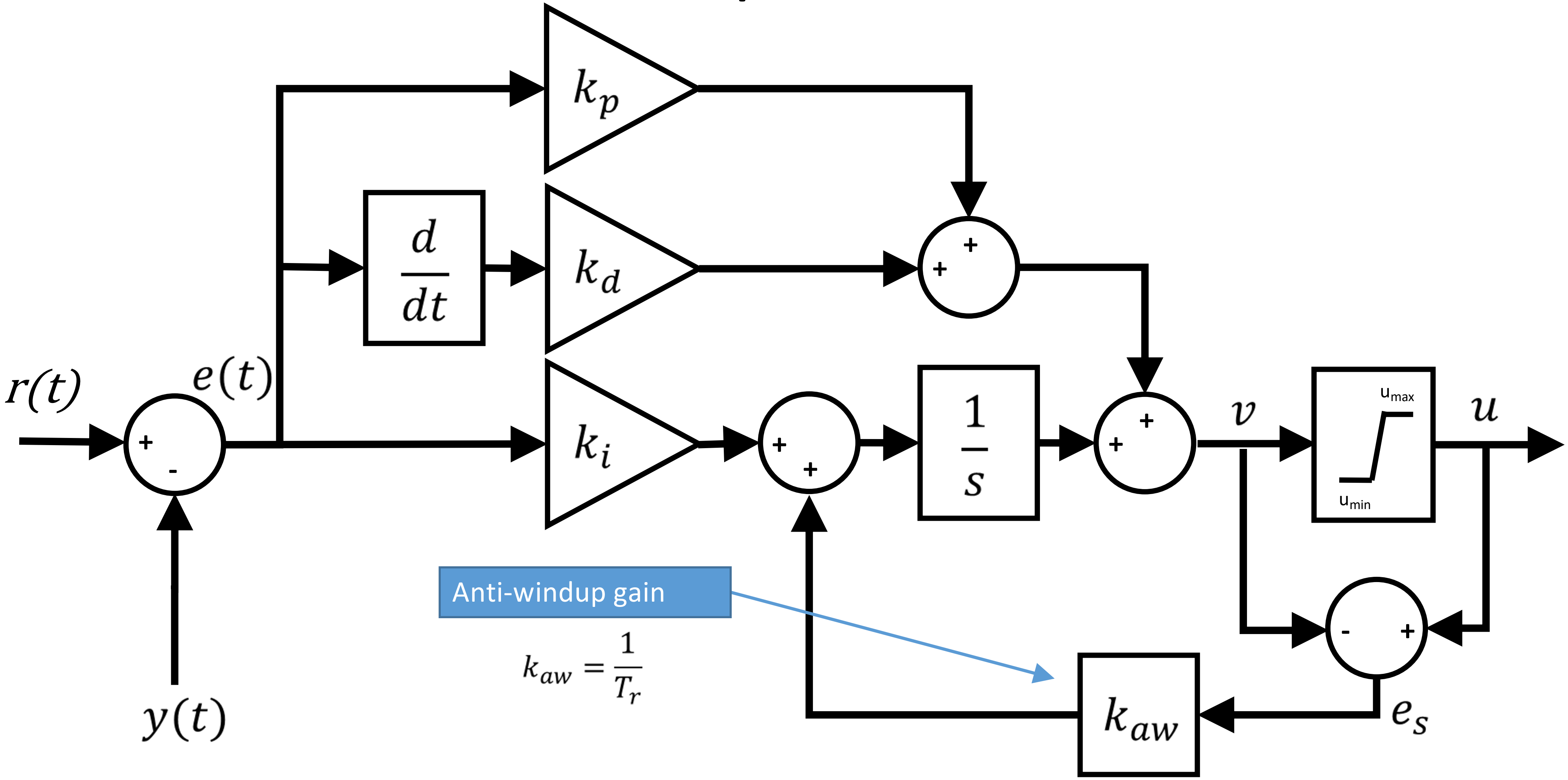
Tank response tracks desired water level well and now has less overshoot

Control effort, i.e. voltage going to pump, is smooth. It still saturates the actuator, but less then before





PID with Anti-Windup



Anti-Windup Gain Design

Anti-windup gain is commonly defined with respect to a **reset time** T_r

$$k_{aw} = \frac{1}{T_r}$$

- Short reset time (large gain) integral reset more quickly
- Long reset time (small gain) integral is reset more slowly
- **Caution**: Setting T_r too small can lead to undesirable reset when measurement noise is present

Anti-Windup Gain Design

Reasonable compromise is to select a reset time that is a fraction of the integral and derivative time.

$$T_r = \sqrt{T_i T_d}$$

Or with respect to the anti-windup gain

$$k_{aw} = 1/T_r = \frac{1}{\sqrt{T_i T_d}}$$

Parameterized PID equation (commonly used in industry):

$$u(t) = k\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right)$$

where k is the proportional gain, T_i is the integral time, and T_d is the derivative time

In the time-domain:

$$u(t) = k_p e(t) + k_i\int_0^t e(\tau)d\tau + k_d\frac{de(t)}{dt}$$

Takeaways – PID with Anti-Windup

Benefits

- Removes steady-state error like PI
- Can reject disturbances like PI
- Removes undesirable overshoot
- More robust

Drawbacks

- Actuator is still saturated, but this helps
- More complicated to implement