Aalto University
School of Electrical
Engineering

ELEC-E8101: Digital and Optimal Control

Lecture 8

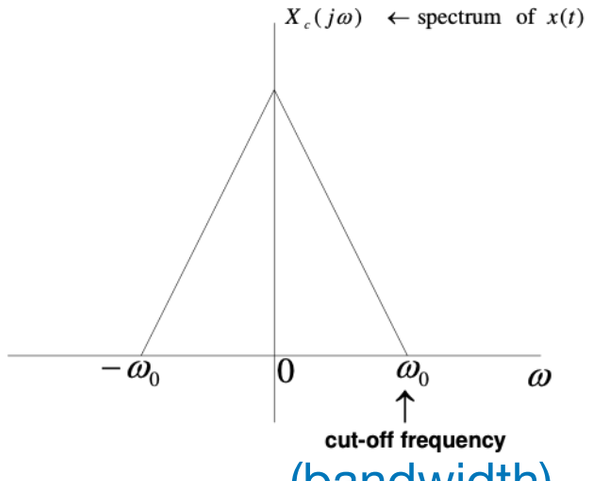
Discrete PID Controllers

Ville Kyrki
Office 2570, Maarintie 8 Building
Email: ville.kyrki@aalto.fi

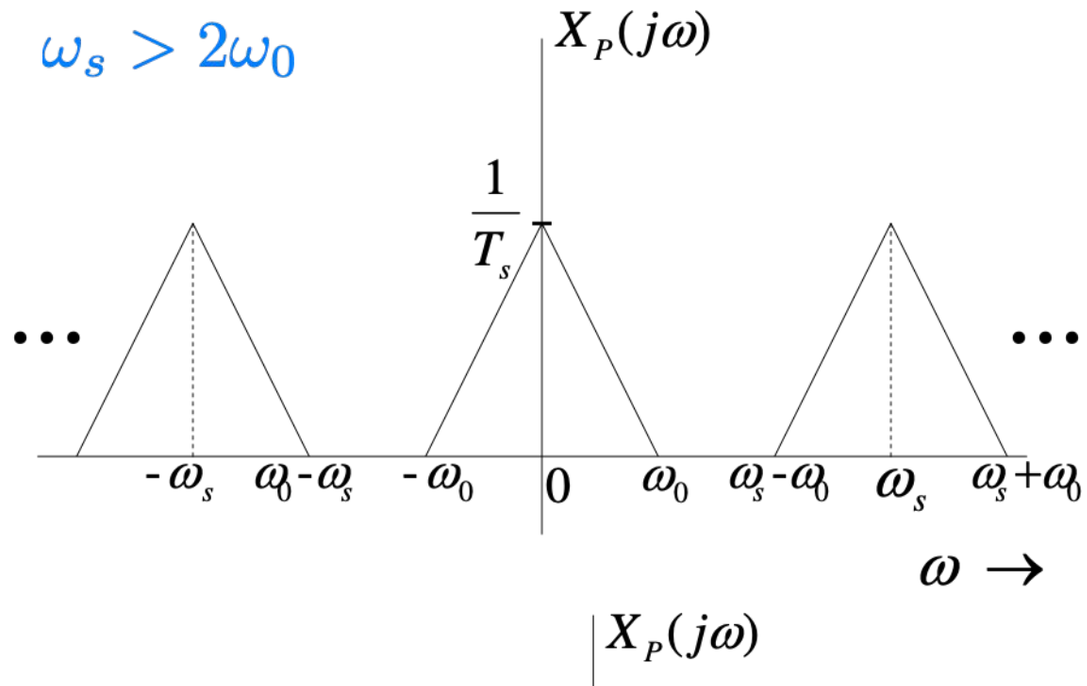
Slides based on ELEC-E8101 material by Themistoklis Charalambous

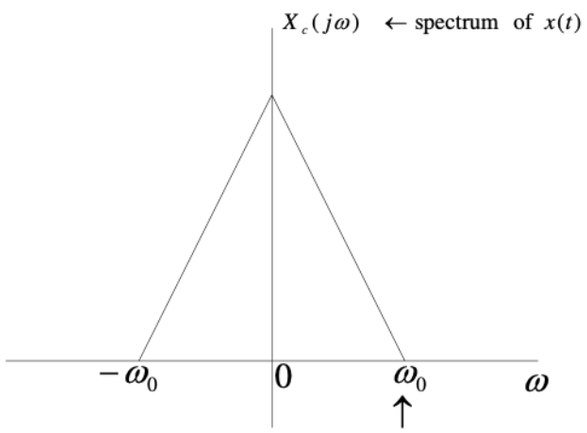
In the previous lecture…

- **What is the sampling frequency so that one can reconstruct the signal**

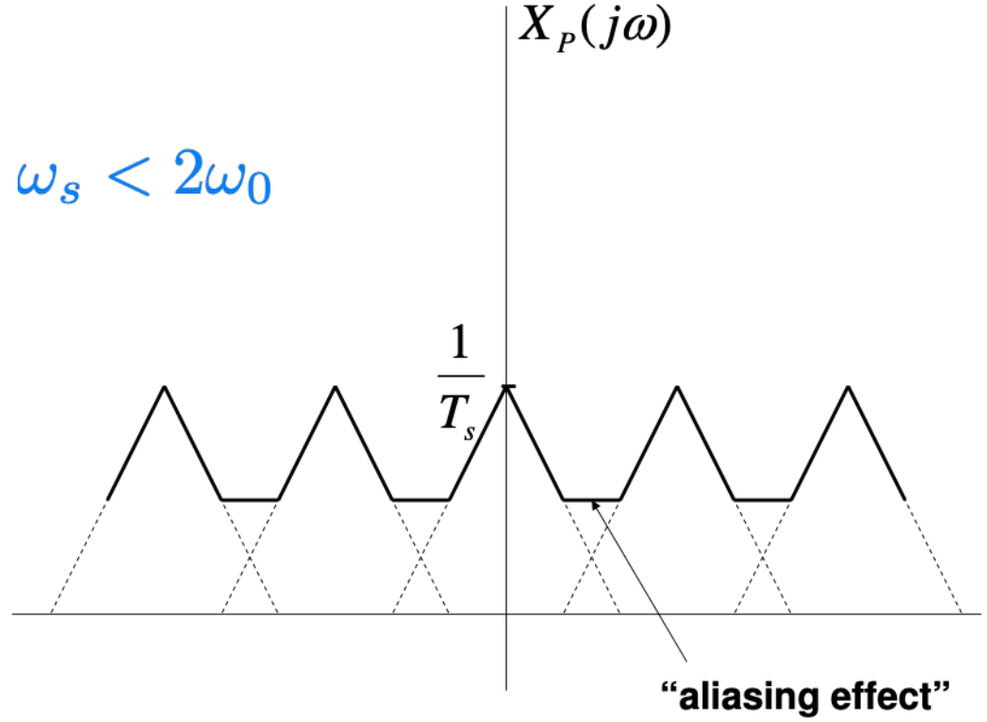


(bandwidth)





(bandwidth)



In the previous lecture…

- **Considered the options of discretization in control systems**

  - Relies on the use of numerical methods for solving differential equations describing the given system and for converting them to difference equations

  ✓ Backward difference method

  ✓ Forward difference method

  ✓ Approximation of derivative and integral

  ✓ Bilinear (or Tustin) method

  - Match the response of continuous-time systems to specific inputs (e.g., impulse, step and ramp functions) to those of discrete-time systems for the same inputs

  ✓ Impulse-invariance method

  ✓ Step-invariance method

Today

We will talk about

- Feedback in discrete setting (recap)
- PID control in continuous time (recap)
- PID in discrete time
- Integral windup and anti-windup methods

Learning outcomes

By the end of *this* lecture, you should be able to:

- Design practical PID controllers for applications
- Design anti-windup schemes to avoid wind-up

PID-Controllers

- Proportional-Integral-Derivative (PID) control is the standard for industrial control with **over 90% of industrial control systems** using PID control

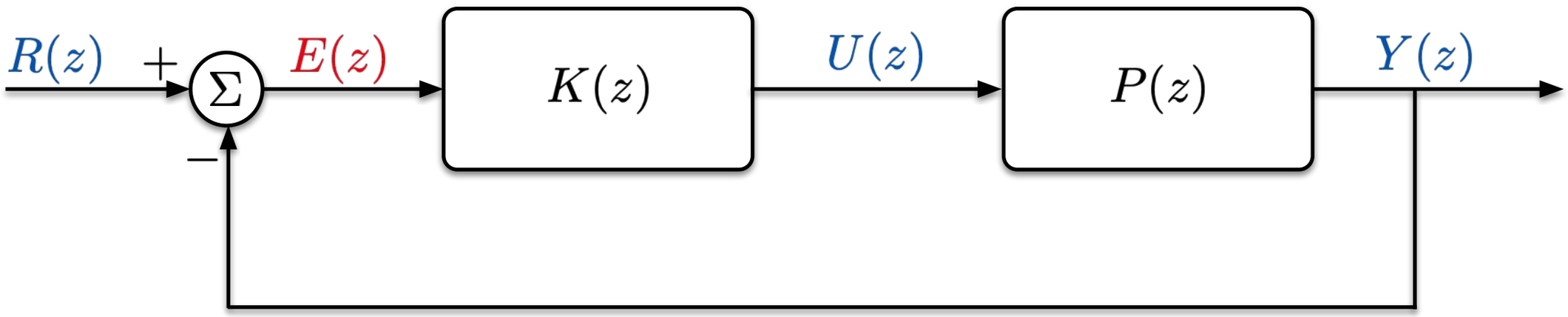
- The ubiquitous nature of PID control stems from:
  - its simple structure
  - the distinct effect of each of the three PID terms
  - its established use in industry
  - engineers' preference to improve existing methods before adopting new

- The first theoretical analysis and practical application was in the field of automatic steering systems for ships - early 1920s onwards.

- Then, used for **automatic process control in manufacturing industry**, where it was widely implemented in pneumatic and electronic controllers.

Feedback control in a discrete setting

- Let us examine the following block diagram of control system:



- We have

$$Y(z) = P(z)U(z)$$

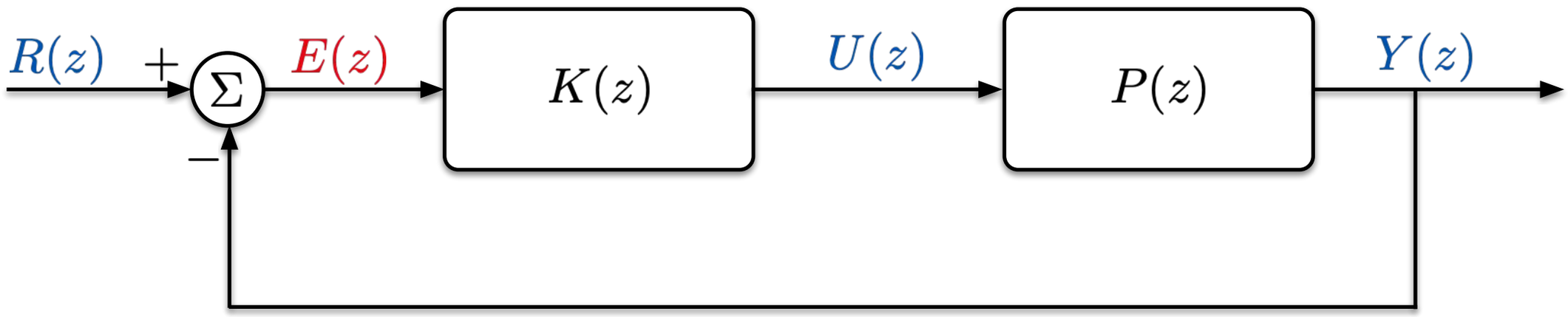
$$U(z) = K(z)E(z)$$

$$E(z) = R(z) - Y(z)$$

- Therefore,

$$Y(z) = P(z)K(z)E(z) = P(z)K(z)(R(z) - Y(z))$$

Feedback control in a discrete setting



- Solving for $Y(z)$

$$Y(z) = P(z)K(z)E(z) = P(z)K(z)(R(z) - Y(z))$$

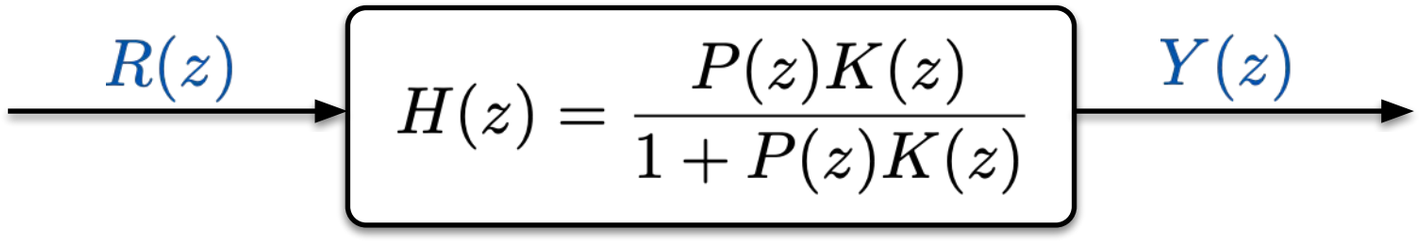
$$(1 + P(z)K(z))Y(z) = P(z)K(z)R(z) \Rightarrow Y(z) = \underbrace{\frac{P(z)K(z)}{1 + P(z)K(z)}}_{\triangleq H(z)} R(z)$$

$$\Rightarrow \boxed{H(z) = \frac{Y(z)}{R(z)} = \frac{P(z)K(z)}{1 + P(z)K(z)}}$$

(closed-loop transfer function from reference/input to output)

Feedback control in a discrete setting

- The block diagram of the control system can be simplified as



- What about the error $E(z)$?

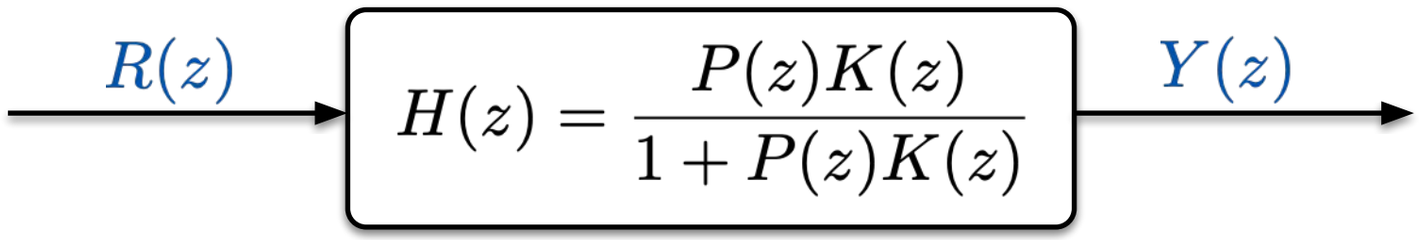
$$Y(z) = P(z)U(z)$$

$$U(z) = K(z)E(z)$$

$$E(z) = R(z) - Y(z)$$

Feedback control in a discrete setting

- The block diagram of the control system can be simplified as



- Similarly, for the error $E(z)$

$$E(z) = R(z) - Y(z) = R(z) - \frac{P(z)K(z)}{1+P(z)K(z)}R(z)$$

$$= \left(1 - \frac{P(z)K(z)}{1+P(z)K(z)}\right)R(z) = \frac{1}{1+P(z)K(z)}R(z)$$

- The problem becomes how to choose an appropriate *K(z)* such that

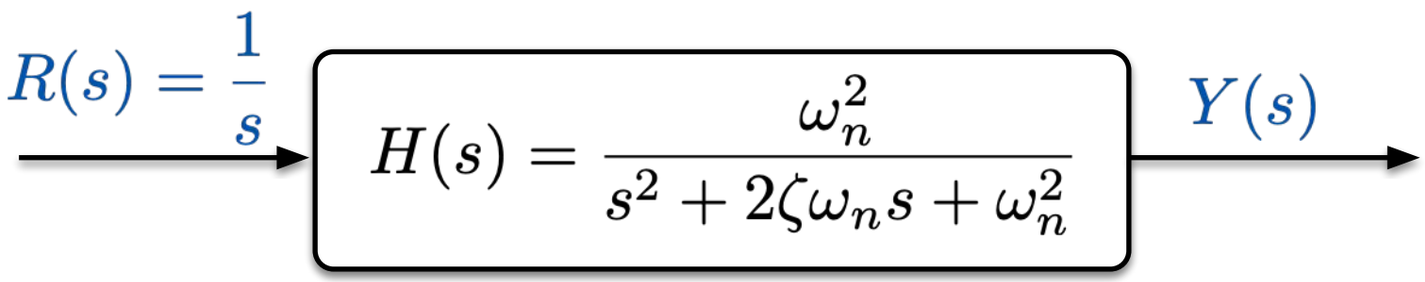
  ➡ *H(z)* will yield desired properties

  ➡ the resulting error function *e[k]* goes to zero as quick and as smooth as possible - equivalent to output *y[k]* tracking the given reference *r[k]*

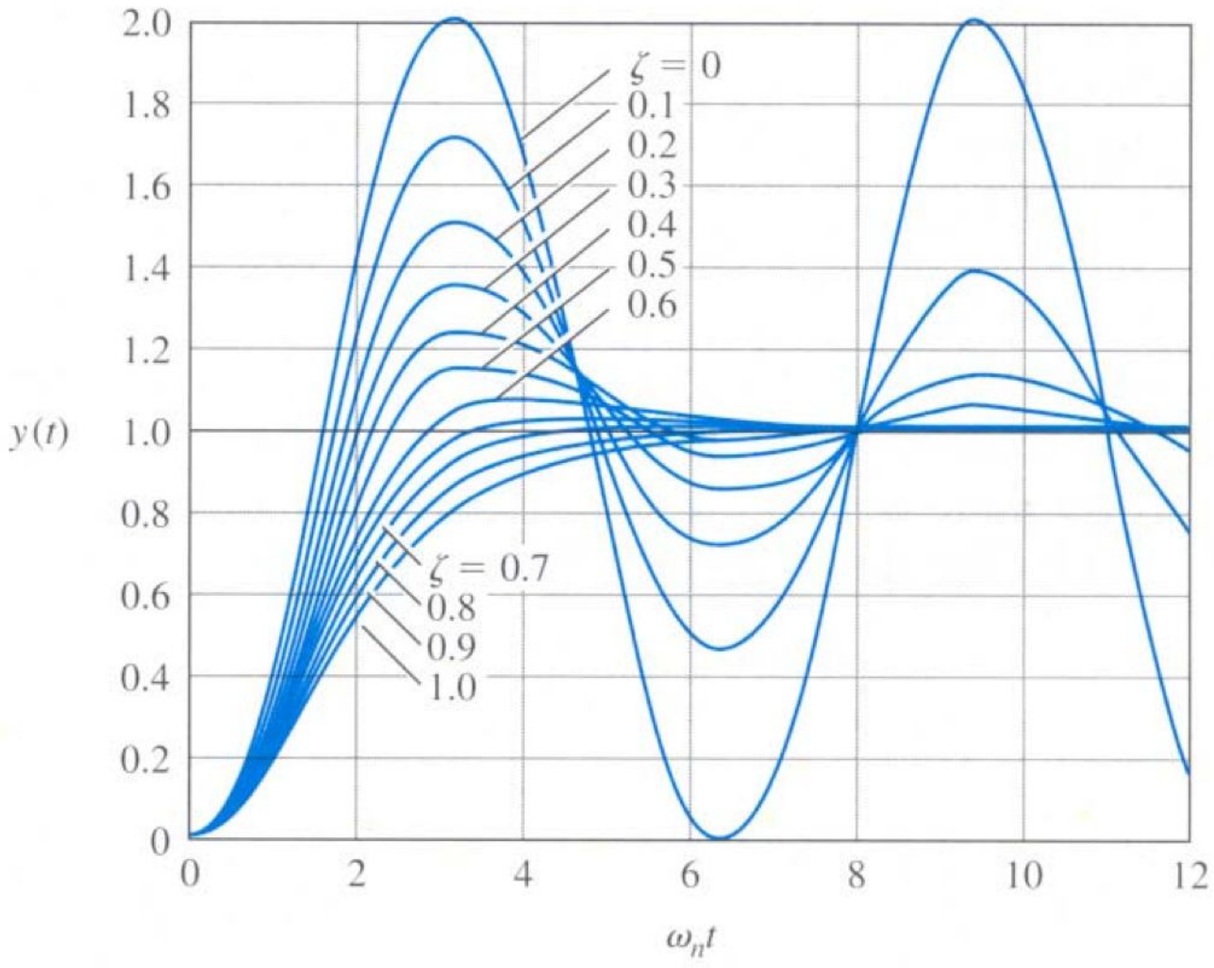
Behavior of continuous 2nd order systems

with unit step input

- Consider the following block diagram with a standard 2nd order system



- The behavior of the system is as follows:

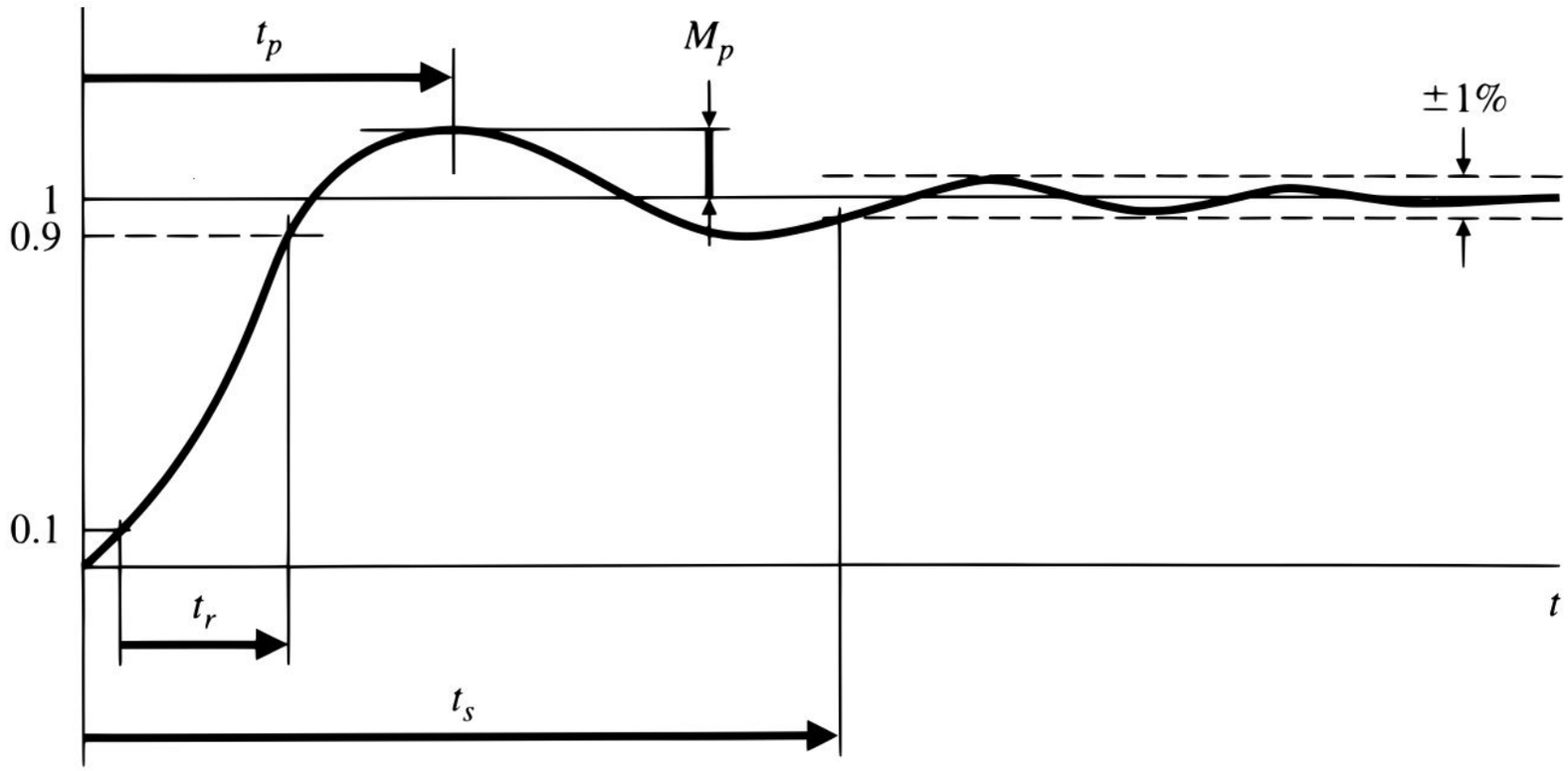


The behavior of the system is fully characterized by:

- ζ the damping factor
- ω_n the natural frequency

Time domain design specification

- Typical specifications for the step response (continuous-time domain):

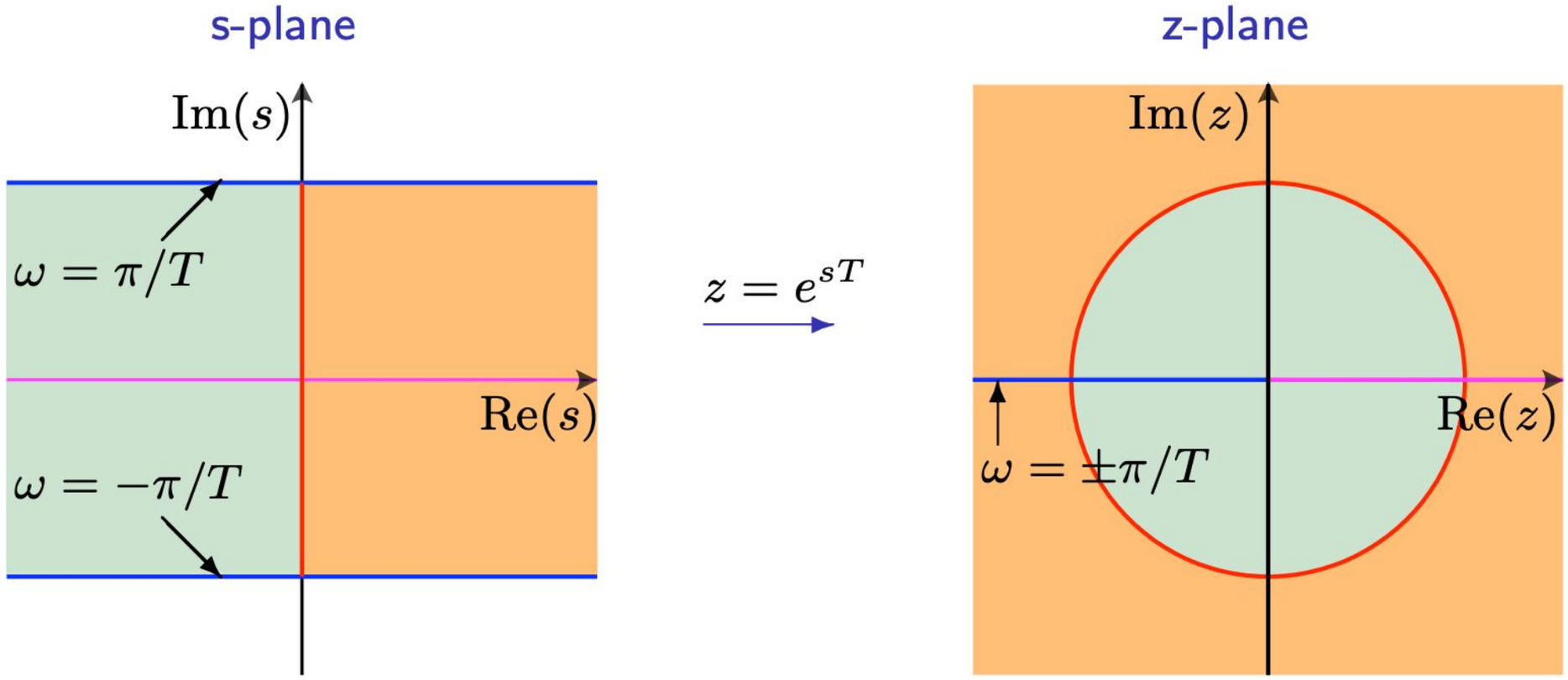


- Steady-state accuracy e_{ss}
- Rise time (10% — 90%) $t_r = 1.8/\omega_n$
- Peak overshoot $M_p \approx e^{-\pi\zeta/\sqrt{1-\zeta^2}}$ or $\zeta \geq 0.6(1 - M_p)$
- Settling time (1%) $t_s = 4.6/(\zeta\omega_n)$

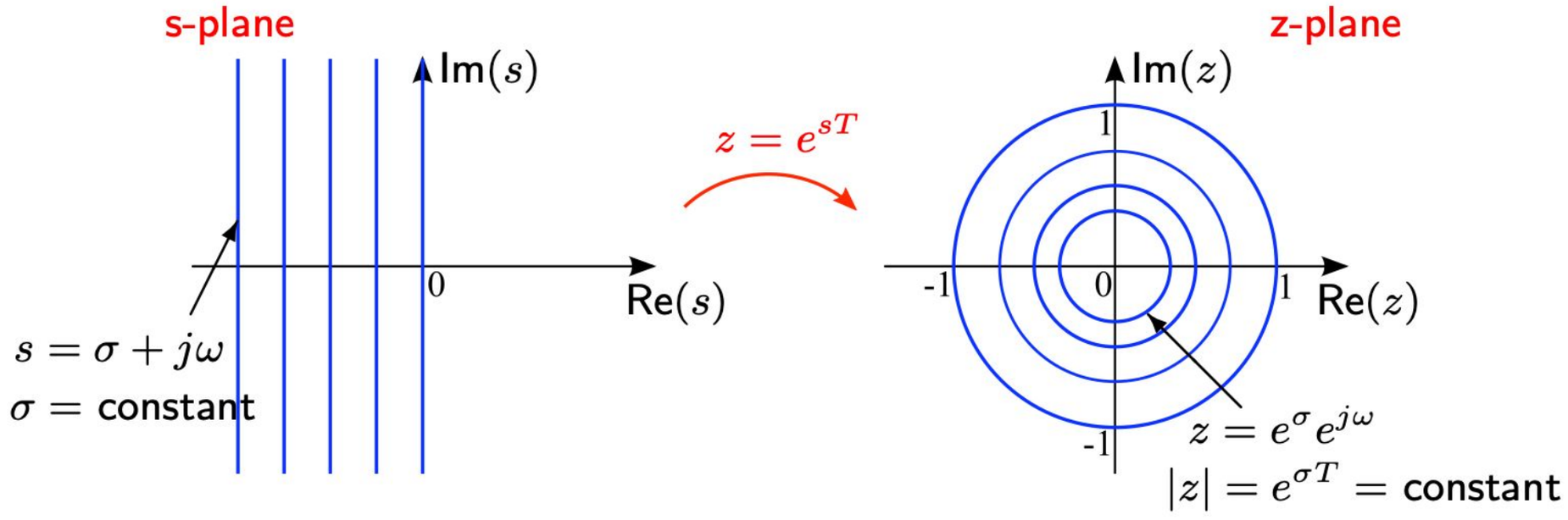
For 2nd order systems

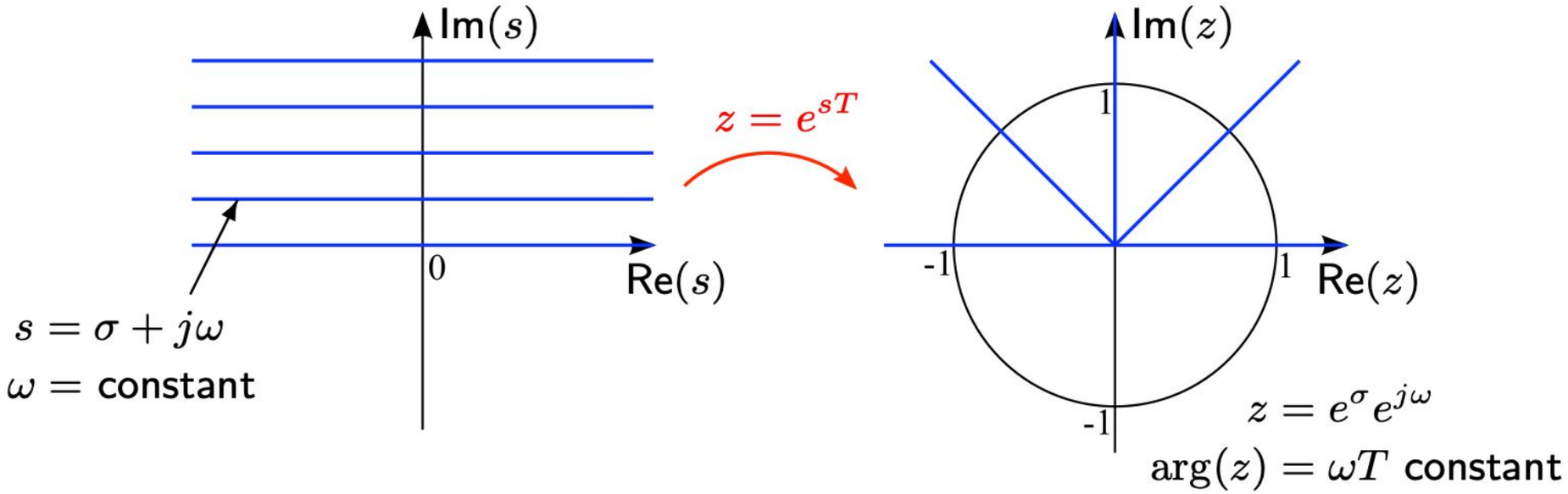
The mapping from the s-plane to the z-plane

- Locus of $s = \sigma + j\omega$ under the mapping $z = e^{sT}$



The mapping from the s-plane to the z-plane





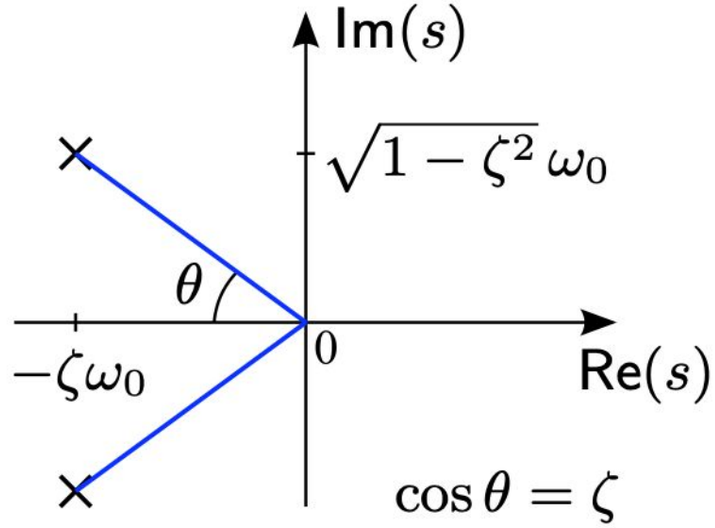
The mapping from the s-plane to the z-plane

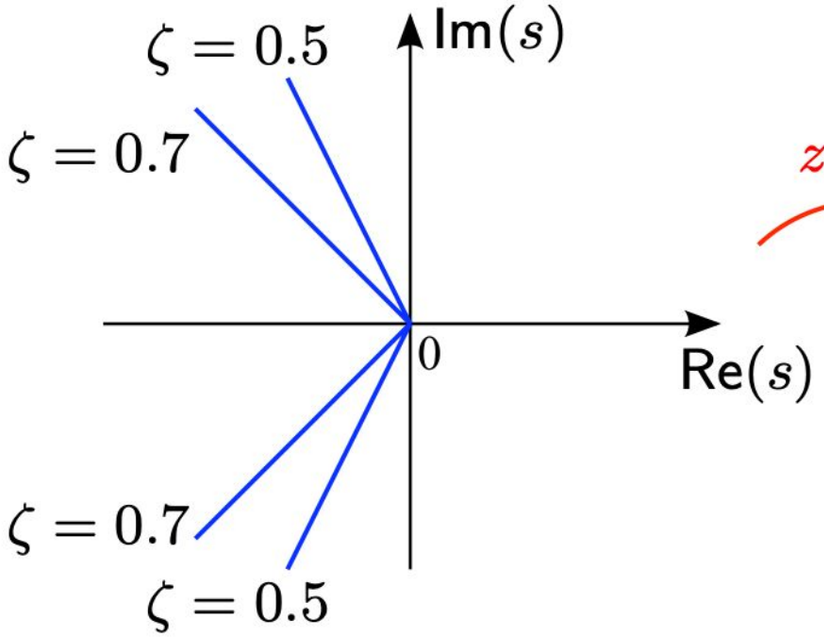
Pole locations for constant damping ratio $\zeta < 1$

$$s^2 + \zeta\omega_0 s + \omega_0^2 = 0$$

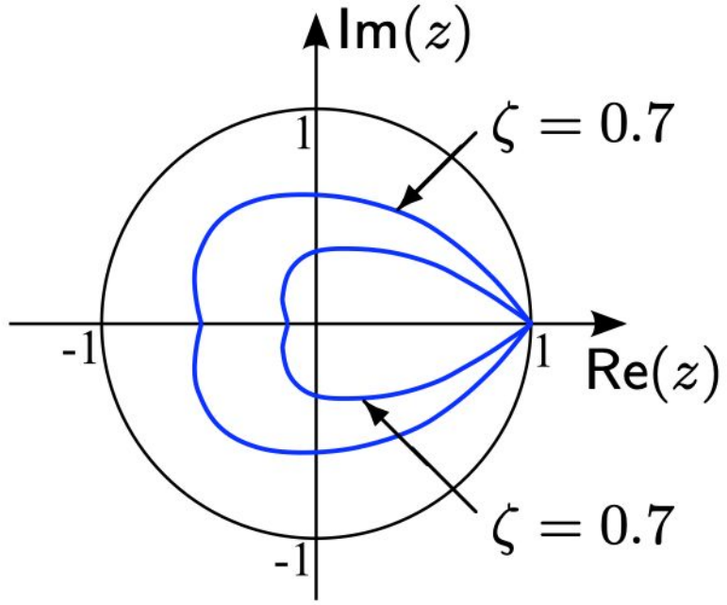
$$\Downarrow$$

$$s = -\zeta\omega_0 \pm j\sqrt{1-\zeta^2}\,\omega_0$$





$z = e^{sT}$

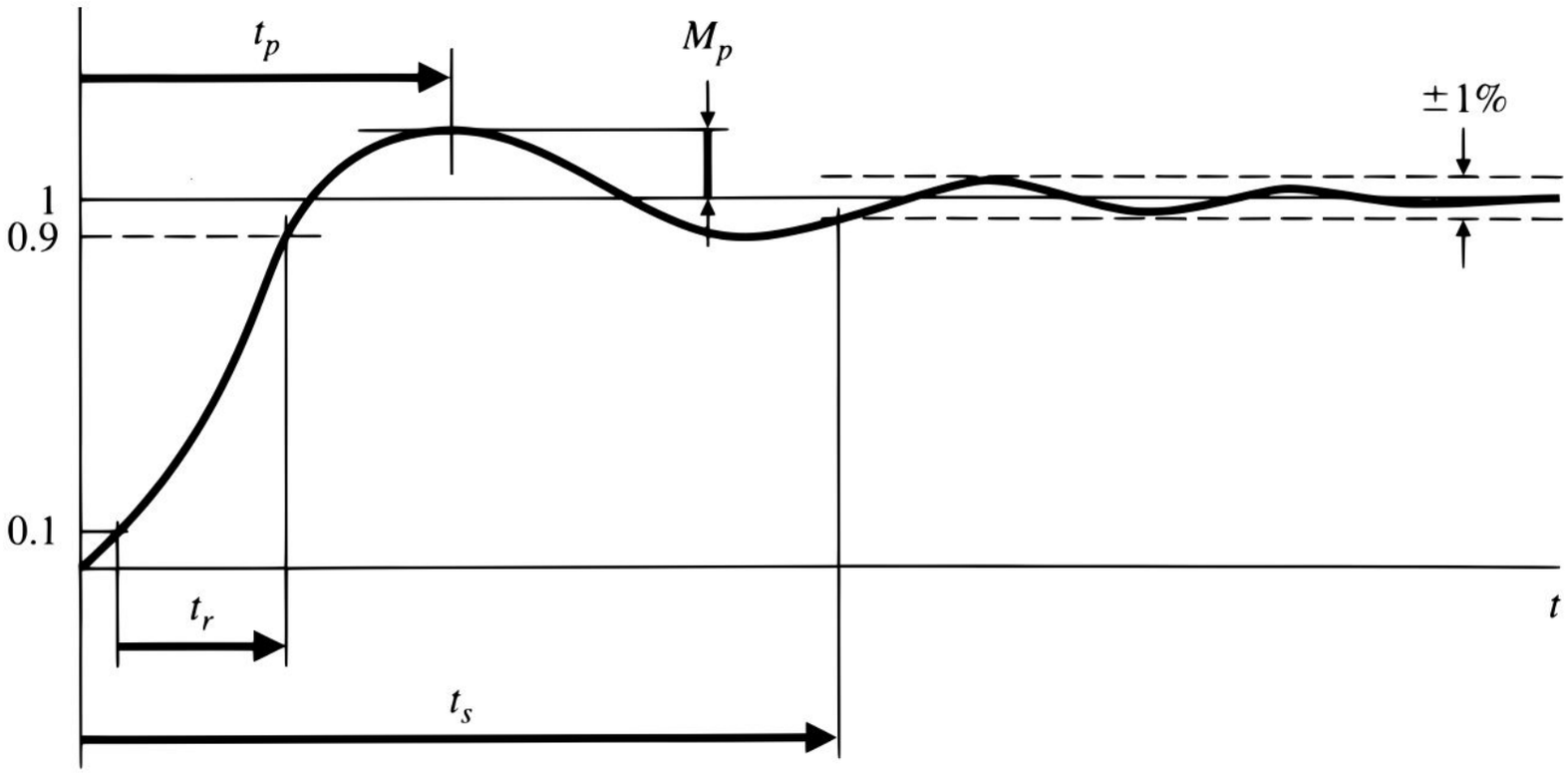


$s = -\zeta\omega_0 + j\sqrt{1-\zeta^2}\,\omega_0$: ζ = constant

$$z = e^{-\zeta\omega_0 T} e^{-j\sqrt{1-\zeta^2}\omega_0 T}$$

Time domain design specification

- Typical specifications for the step response (discrete-time domain):



- Steady-state accuracy $e_{ss} = \lim_{z \to 1}(z-1)E(z)$
- Rise time (10% — 90%) $t_r = 1.8/\omega_n$
- Peak overshoot $M_p \approx e^{-\pi\zeta/\sqrt{1-\zeta^2}}$ or $\zeta \geq 0.6(1 - M_p)$
- Settling time (1%) radius of poles: $|z| < 0.01^{T_s/t_s}$

Example

- A continuous system with transfer function

$$G(s) = \frac{1}{s(10s+1)}$$

is controlled by a discrete control system with ZOH. The closed-loop system is required to have:

- Step response overshoot: $M_p < 16\%$
- Step response settling time (1%): $t_s < 10s$
- Steady-state error to unit ramp: $e_{ss} < 1$

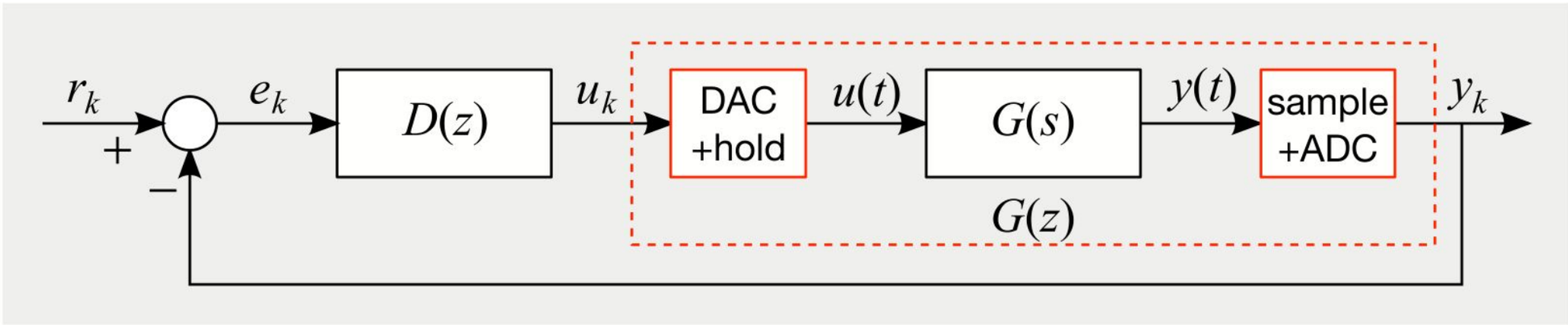
Check these specifications if the sampling time is $T_s = 1$and the controller is

$$u[k] = -0.5u[k-1] + 13(e[k] - 0.88e[k-1])$$

Example

Solution:

a) We first find the pulse transfer function of $G(s)$ with the ZOH:



$$G(z) = \frac{z-1}{z}\mathcal{Z}\left\{\frac{G(s)}{s}\right\}$$

$$= \frac{z-1}{z}\mathcal{Z}\left\{\frac{0.1}{s^2(s+0.1)}\right\}$$ (check it at home!)

$$= \frac{z-1}{z}\frac{z\left((0.1-1+e^{-0.1})z+(0.1-1-e^{-0.1})\right)}{0.1(z-e^{-0.1})(z-1)^2}$$

$$= \frac{0.0484(z+0.9672)}{(z-1)(z-0.9048)}$$

Example

$$u[k] = -0.5u[k-1] + 13(e[k] - 0.88e[k-1])$$

b) Find the controller transfer function

$$D(z) = \frac{U(z)}{E(z)} = 13\frac{1 - 0.88z^{-1}}{1 + 0.5z^{-1}} = 13\frac{z - 0.88}{z + 0.5}$$

c) Check the steady-state error for a unit ramp. We already showed that

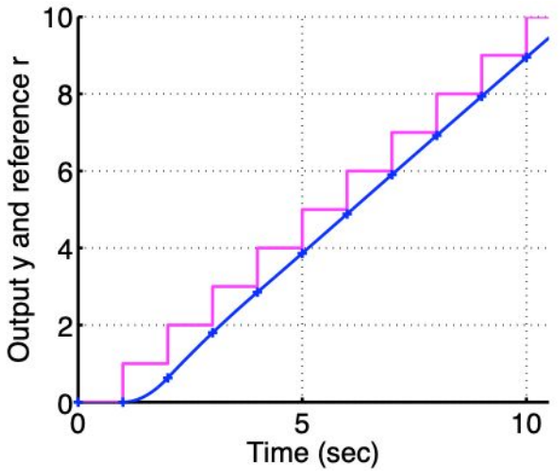
$$E(z) = \frac{1}{1 + P(z)K(z)}R(z)$$

where the input is: $R(z) = \frac{Tz}{(z-1)^2} = \frac{z}{(z-1)^2}$ Therefore,

$$e_{ss} = \lim_{k\to\infty} = \lim_{z\to 1}(z-1)E(z)$$

$$= \lim_{z\to 1}\left\{(z-1)\frac{z}{(z-1)^2}\frac{1}{1 + D(z)G(z)}\right\} = \ldots = 0.96$$

$$\Rightarrow e_{ss} < 1 \quad \text{(as required)}$$



Example

d) Step response:

$$M_p < 16\% \Rightarrow \zeta > 0.5$$

$$t_s < 10s \Rightarrow |z| < 0.01^{1/10} = 0.63$$

The closed loop poles are the roots of $1 + D(z)G(z) = 0$

$$1 + \underbrace{13\frac{z - 0.88}{z + 0.5}}_{D(z)} \underbrace{\frac{0.0484(z + 0.9672)}{(z - 1)(z - 0.9048)}}_{G(z)} = 0$$

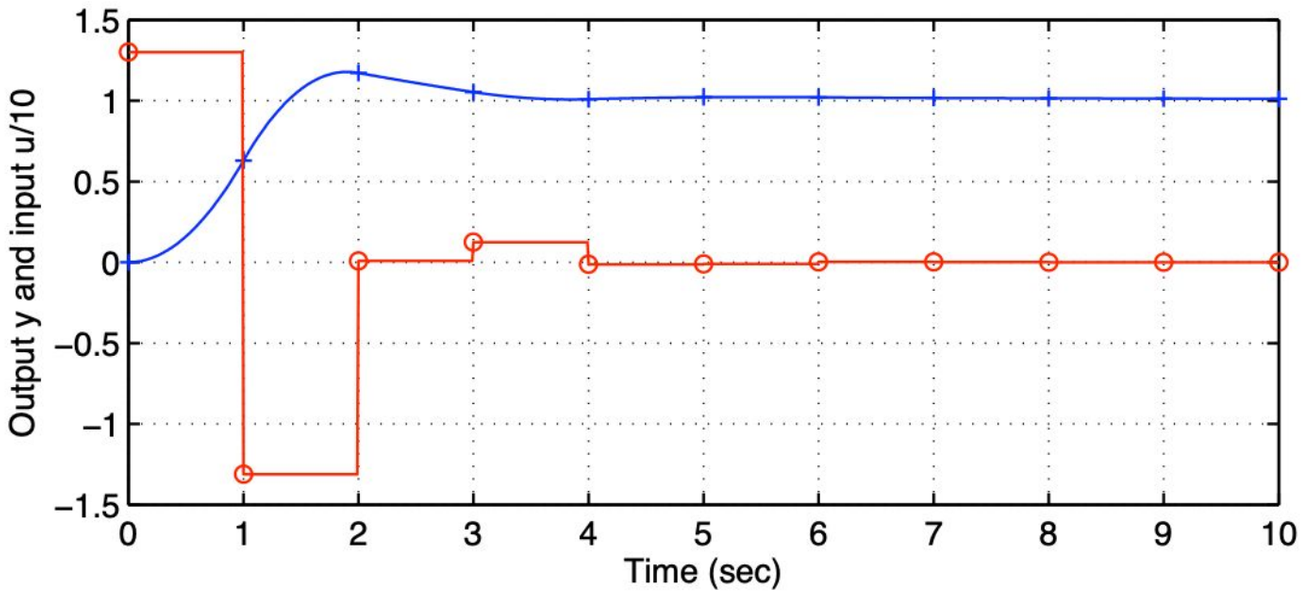
$$\Rightarrow z = 0.88,\ -0.050 \pm j0.304$$

But the pole at $z = 0.88$ is canceled by controller's zero and, therefore,

$$z = -0.050 \pm j0.304 = re^{\pm j\theta} \Rightarrow \begin{cases} r = 0.31,\ \theta = 1.73 \\ \zeta = 0.56 \end{cases}$$

$$z = e^{-\zeta\omega_0 T} e^{-j\sqrt{1-\zeta^2}\omega_0 T}$$

Example

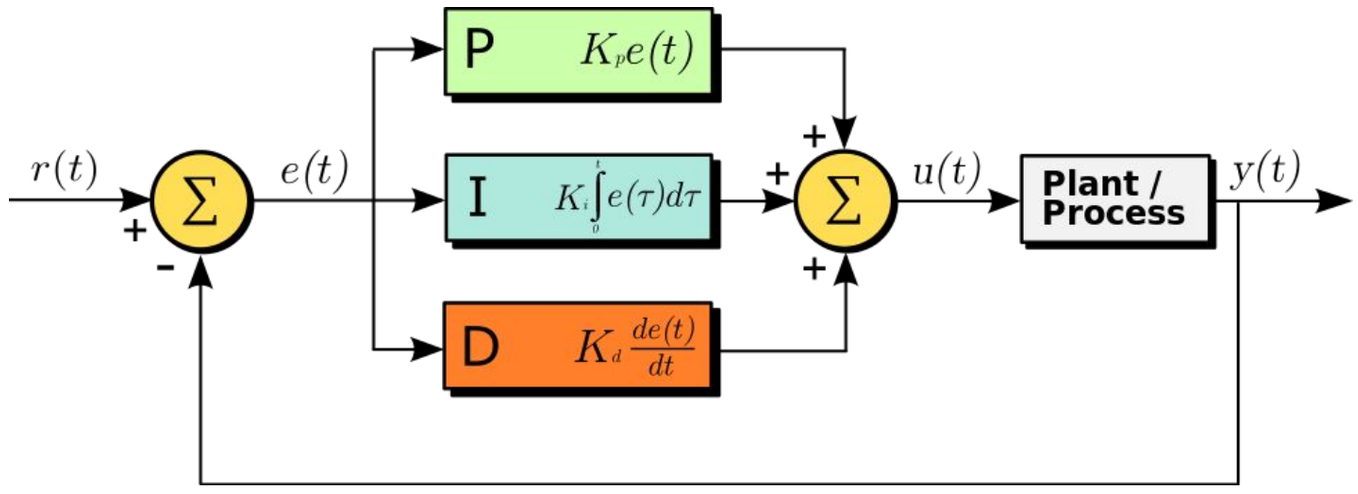


all specs satisfied!

Or by simulation!

Continuous-time PID-Controllers

- **P**: amplifies the error by K_p.
- **I**: eliminate the residual error by adding a control effect due to the historic cumulative value of the error
- **D**: reduce the effect of the error by exerting a control based on the rate of error change



- The continuous-time PID controller (in time domain) is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

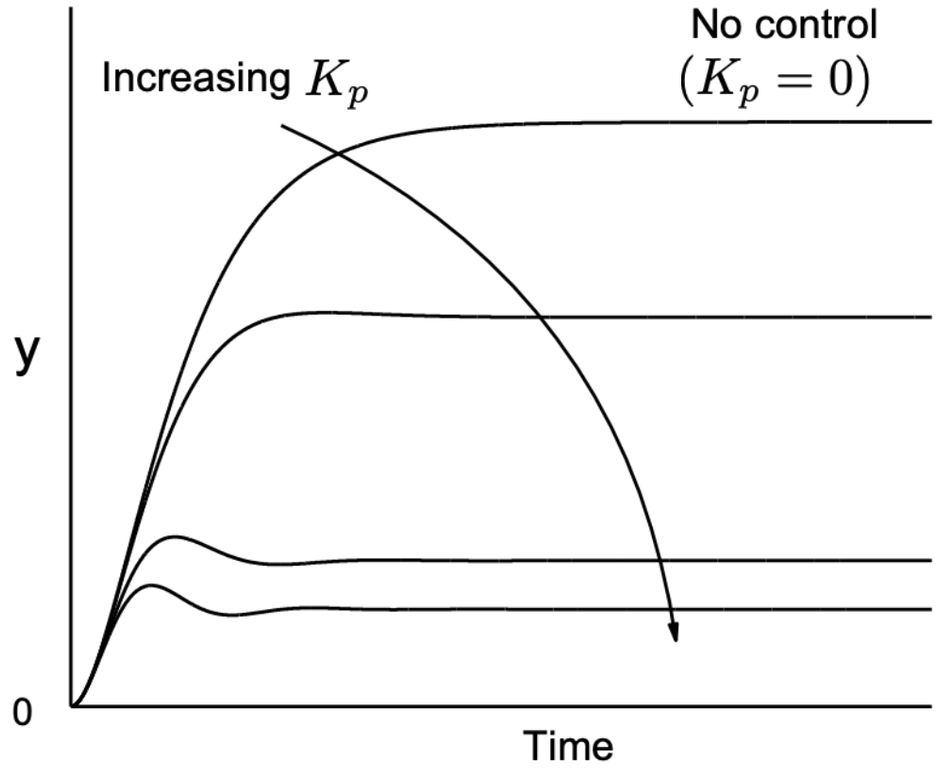
- Taking Laplace transforms:

$$\bar{u}(s) = \left(K_p + \frac{K_i}{s} + K_d s \right) \bar{e}(s)$$

P-Controller

- The obvious method - proportional control

- This method fails if, for instance, the error corresponds to more than a single task or the system changes; hence, for the same error, different gains are needed.

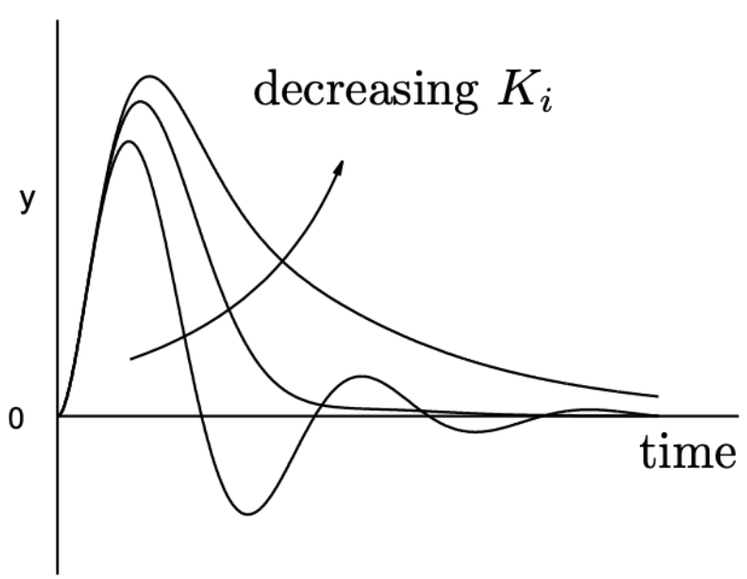


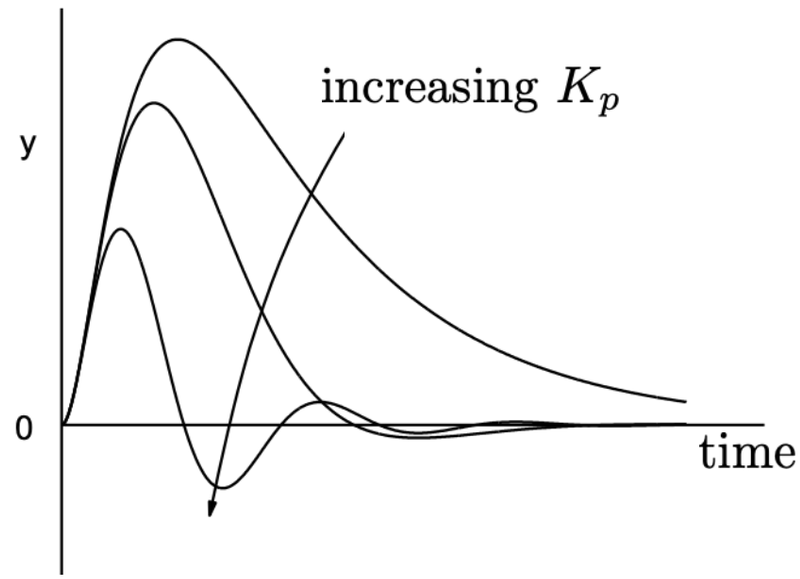
- That's where the integral and derivative terms play their part.

I-Controller

- An **integral** term increases action in relation not only to the error, but also the time for which it has persisted. So, if applied control action is not enough to bring the error to zero, this control action will be increased as time passes.

- A pure "**I**" controller could bring the error to zero, however, it would be both slow reacting at the start, brutal, and slow to end, prompting overshoot and oscillations.



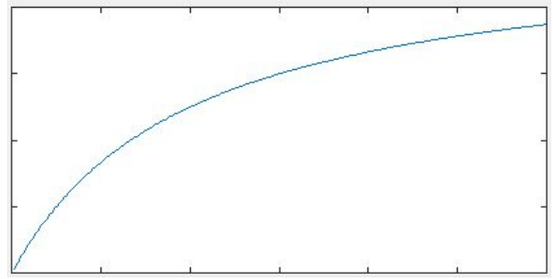


- Alternative formulation: change the error in small persistent steps - over time the steps accumulate and add up dependent on past errors;
  this is the discrete-time equivalent to integration.

D-Controller

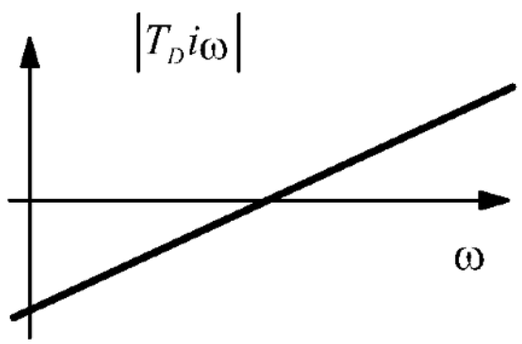
- Aims at flattening the error trajectory into a horizontal line, damping the control applied, and so reduces overshoot

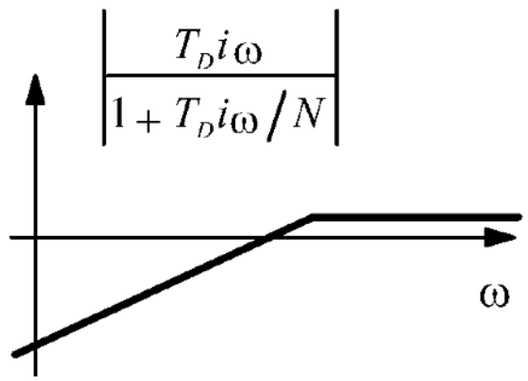
- Ideal derivative control cannot (and must not) be realized in a PID-controller. Practical systems always contain high frequency disturbances (e.g., white noise), which are amplified by derivative control.



- Because of that a lag term is usually added to the derivation.

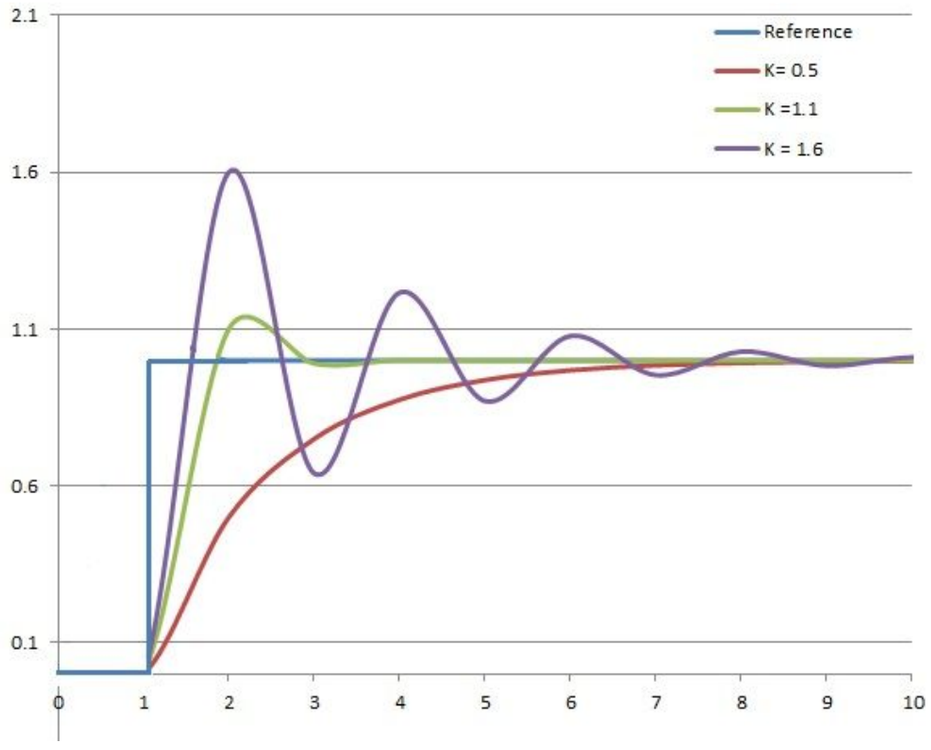
$$K_d s \rightsquigarrow \frac{K_d s}{1 + K_d s / N}$$



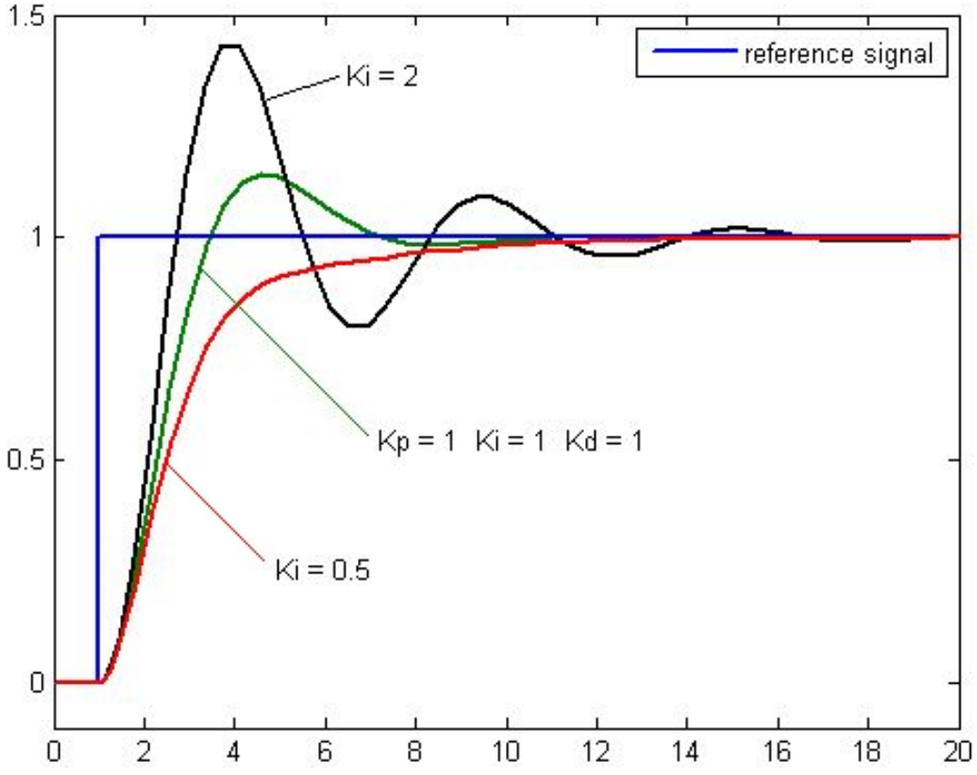


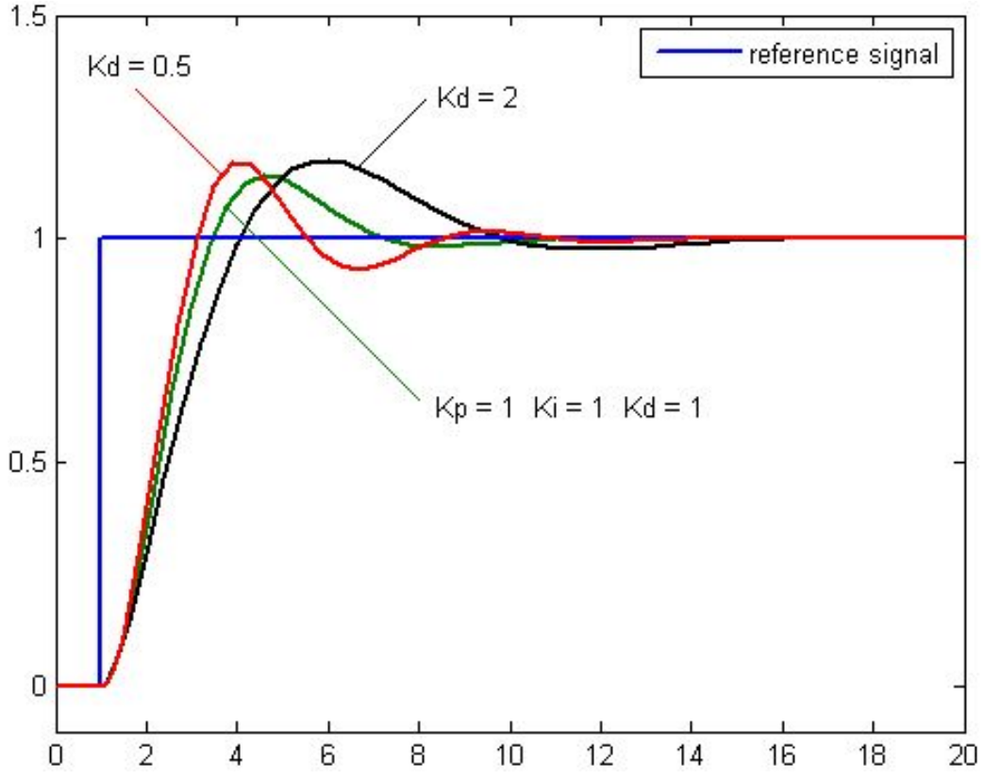
- Other practical modification is to *derivate only the output* (not the reference, not the error signal)

PID-Controller



- **Top-left:** P-controller effect (ID-controllers kept constant)
- **Bottom-left:** I-controller effect (PD-controllers kept constant)
- **Bottom-right:** D-controller effect (PI-controllers kept constant)





From continuous- to discrete-time PID controllers

- Simple discretization:

$$\begin{cases} P(t) = K_p e(t) \\ I(t) = K_i \int_{-\infty}^{t} e(\tau) d\tau \\ D(t) = K_d \frac{de(t)}{dt} \end{cases} \Rightarrow \begin{cases} P(kh) = K_p e(kh) \\ I(kh) = K_i \sum_{n=-\infty}^{k-1} e(nh)h = K_i h \sum_{n=-\infty}^{k-1} e(nh) \\ D(kh) = K_d \frac{e(kh) - e(kh-h)}{h} = \frac{K_d}{h} \Delta e(kh) \end{cases}$$

- Therefore:

$$u(kh) = K_p e(kh) + K_i h \sum_{n=-\infty}^{k-1} e(nh) + \frac{K_d}{h} \Delta e(kh)$$

- Taking the z-transform (**why?**):

$$U(z) = \underbrace{\left(K_p + \frac{K_i h}{z-1} + \frac{K_d}{h} \frac{z-1}{z} \right)}_{H_{\text{PID}}} E(z)$$

From continuous- to discrete-time PID controllers

- Note: the above PID-controller is not the only interpretation of a discrete PID-controller. For example, if **backward integration** is used in the integral part, the formula below follows:

$$H_{\mathrm{PID}} = G_{\mathrm{PID}}(s)\Big|_{s=\frac{z-1}{zh}} = K_p + \frac{K_i h z}{z-1} + \frac{K_d}{h}\frac{z-1}{z}$$

- The discretization of a practical PID-controller is as straightforward:

$$\begin{cases} P_m(s) = K_p(Y_{\mathrm{ref}} - Y(s)) \\ I(s) = \dfrac{K_i}{s}(Y_{\mathrm{ref}} - Y(s)) \\ D_m(s) = -\frac{K_d s}{1+K_d s/N} Y(s) \end{cases} \Rightarrow \begin{cases} P_m(z) = K_p(Y_{\mathrm{ref}} - Y(z)) \\ I(z) = \dfrac{K_i h}{z-1}(Y_{\mathrm{ref}} - Y(z)) \\ D_m(z) = D_m(s)\Big|_{s=\frac{z-1}{zh}} = -\frac{K_d \frac{z-1}{zh}}{1+\frac{K_d}{N}\frac{z-1}{zh}} Y(z) \end{cases}$$

Tuning PID controllers

- The structure of the used discrete PID algorithm must always be told together with the tuning parameters K_p, K_i, K_d (and h).

- Controller design is based on heuristic design methods for selecting the controller parameters.

- The principal design goal is stability: The system is stable when the closed loop poles are on the left-half of s-plane or inside the unit circle in z-plane

- Secondary criteria are, for example, rise, overshoot, settling time, and steady state error. These can be analyzed graphically from impulse, step and ramp responses of the close loop system

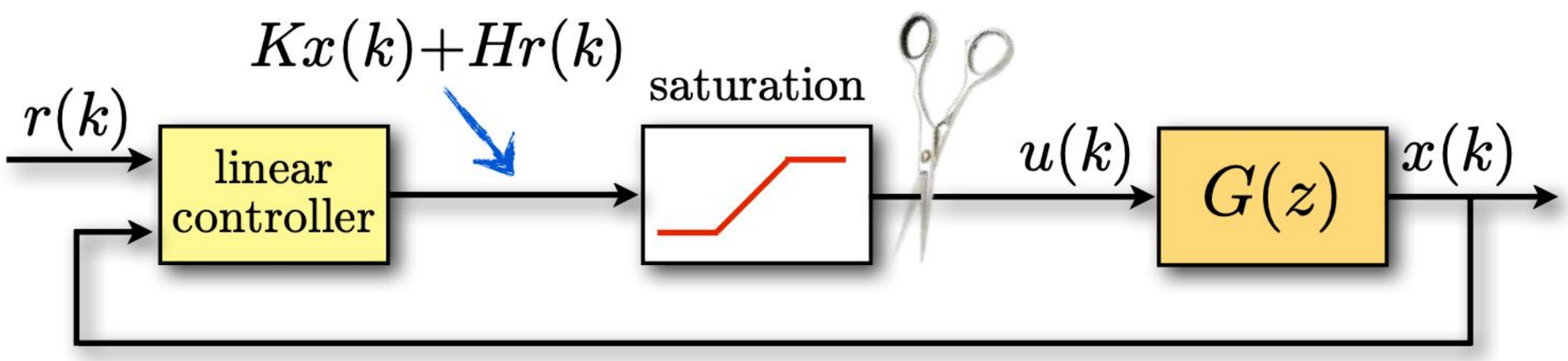
Effects of *increasing* a parameter independently

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if small

Disturbances considered next week

Actuator saturation

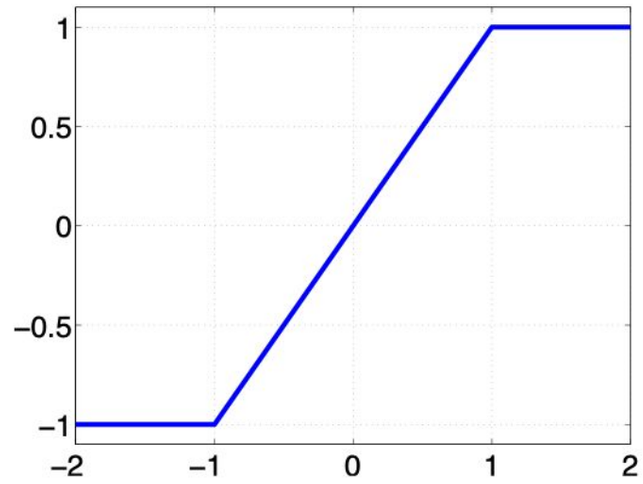
- Most control systems are designed based on linear theory
- A linear controller is simple to implement and performance is good, as long as dynamics remain close to linear
- Nonlinear effects require care, such as actuator saturation (always present)
- Saturation phenomena, if neglected in the design phase, can lead to closed-loop instability, especially if the process is open-loop unstable.
  Main reason: the control loop gets broken if saturation is not taken into account by the controller



Saturation function

- Saturation can be defined as the static nonlinearity

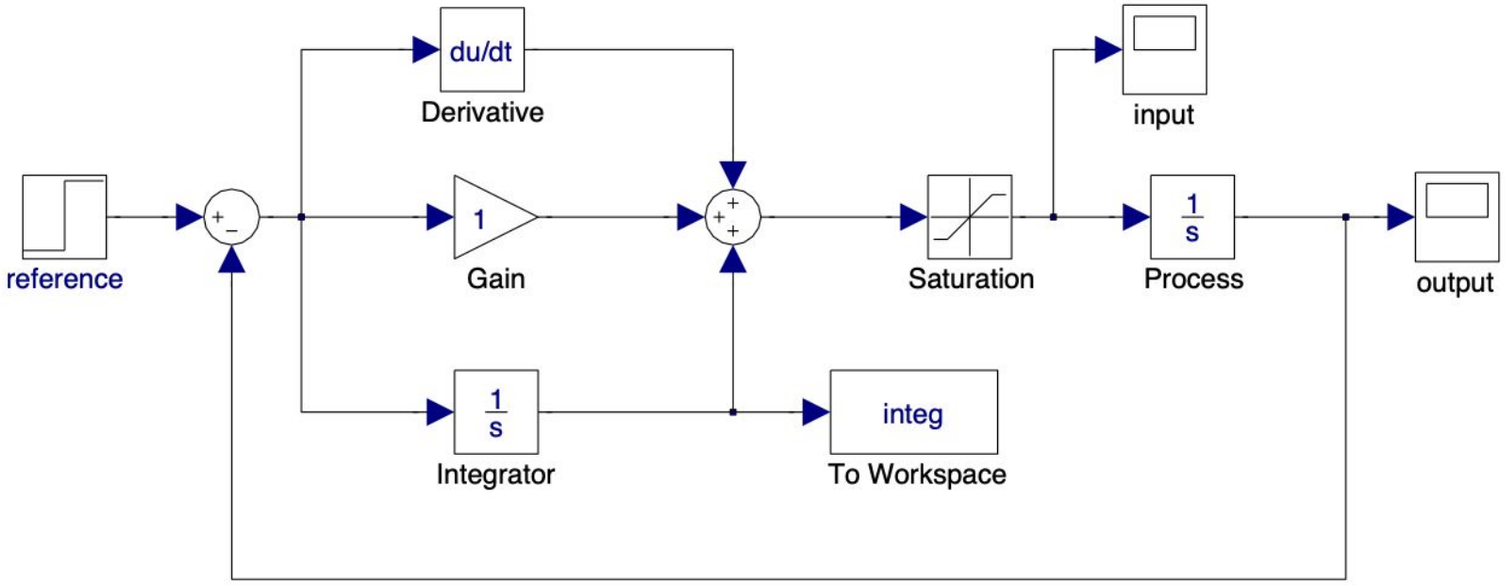
$$sat(u) = \begin{cases} u_{\min}, & \text{if } u < u_{\min} \\ u, & \text{if } u_{\min} \leq u \leq u_{\max} \\ u_{\max}, & \text{if } u > u_{\max} \end{cases}$$

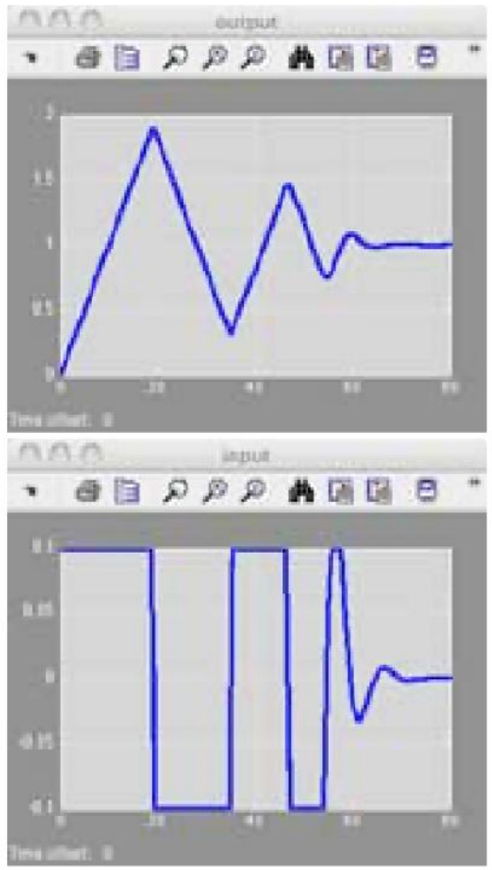


- u_{min} and u_{max} are the minimum and maximum allowed actuation signals (for example, maximum current for an electric motor)

- If u is a vector with $m>1$ components, the saturation function is defined as the saturation of all its components

The wind-up problem

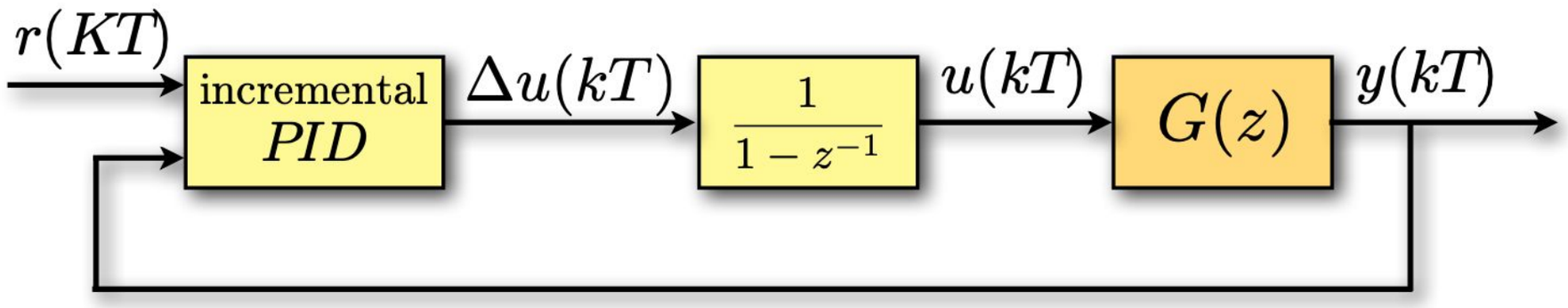




very simple process (=an integrator)
controlled by a PID controller ($K_P = K_I = K_D = 1$)
under saturation $-0.1 \leq u \leq 0.1$

- The output takes a long time to go steady-state
- The reason is the "wind-up" of the integrator contained in the PID controller, which keeps integrating the tracking error even if the input is saturating
- Anti-windup schemes avoid such a wind-up effect

Anti-windup #1: Incremental algorithm

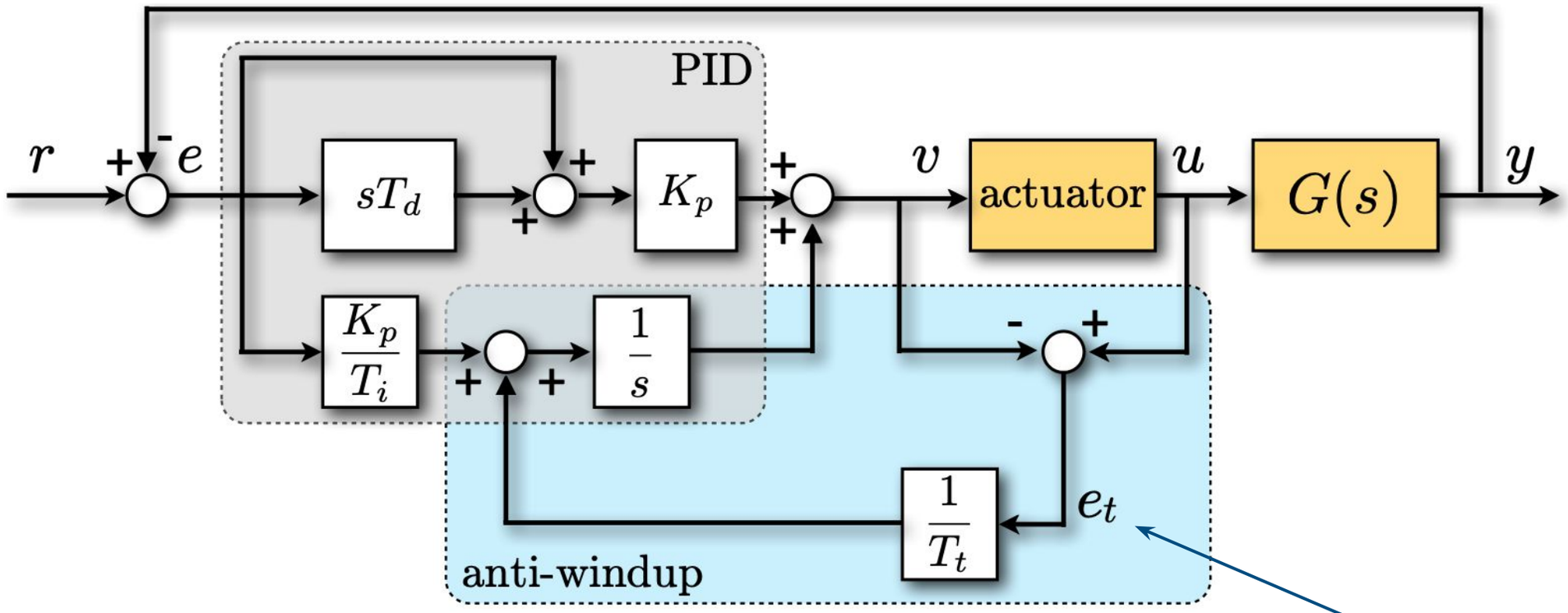


- It only applies to PID control laws implemented in incremental form

$$u[k+1] = u[k] + \Delta u[k]$$

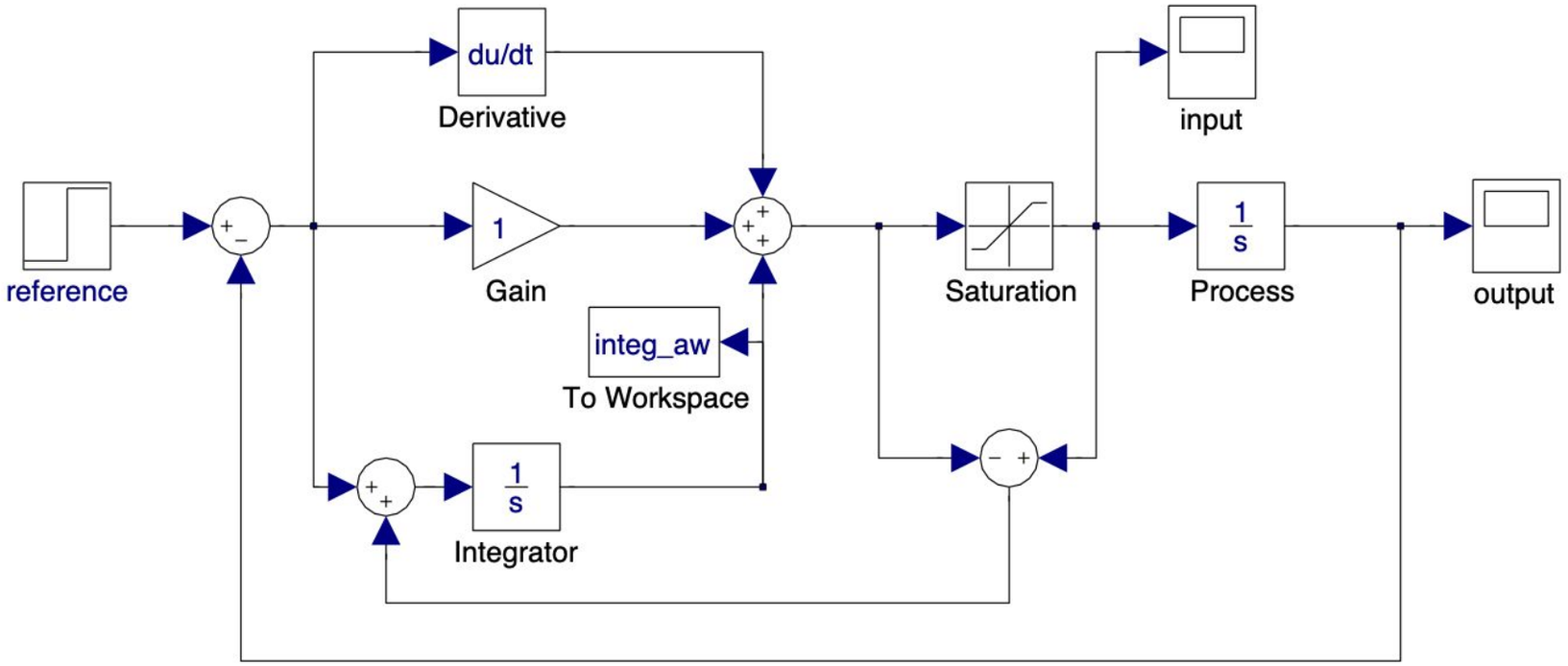
- Integration is stopped if adding a new $\Delta u[k]$ causes a violation of the saturation bound

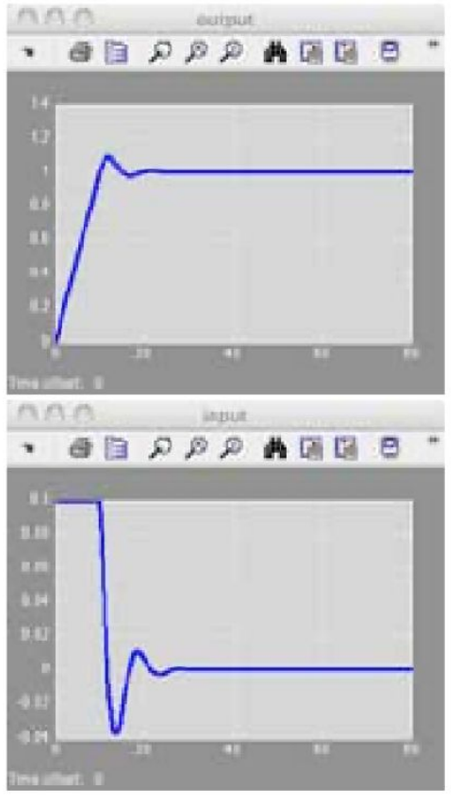
Anti-windup #2: Back-calculation



- Anti-windup scheme has no effect when the actuator is not saturating, $e_{t=0}$
- The time constant T_t determines how quickly the integrator of the PID controller is reset (negative feedback to integrator)
- If the actual output $u(t)$ of the actuator in not measurable, we can use a mathematical model of the actuator, e.g., $e_t(t) = v(t) - sat(v(t))$

Anti-windup #2: Example



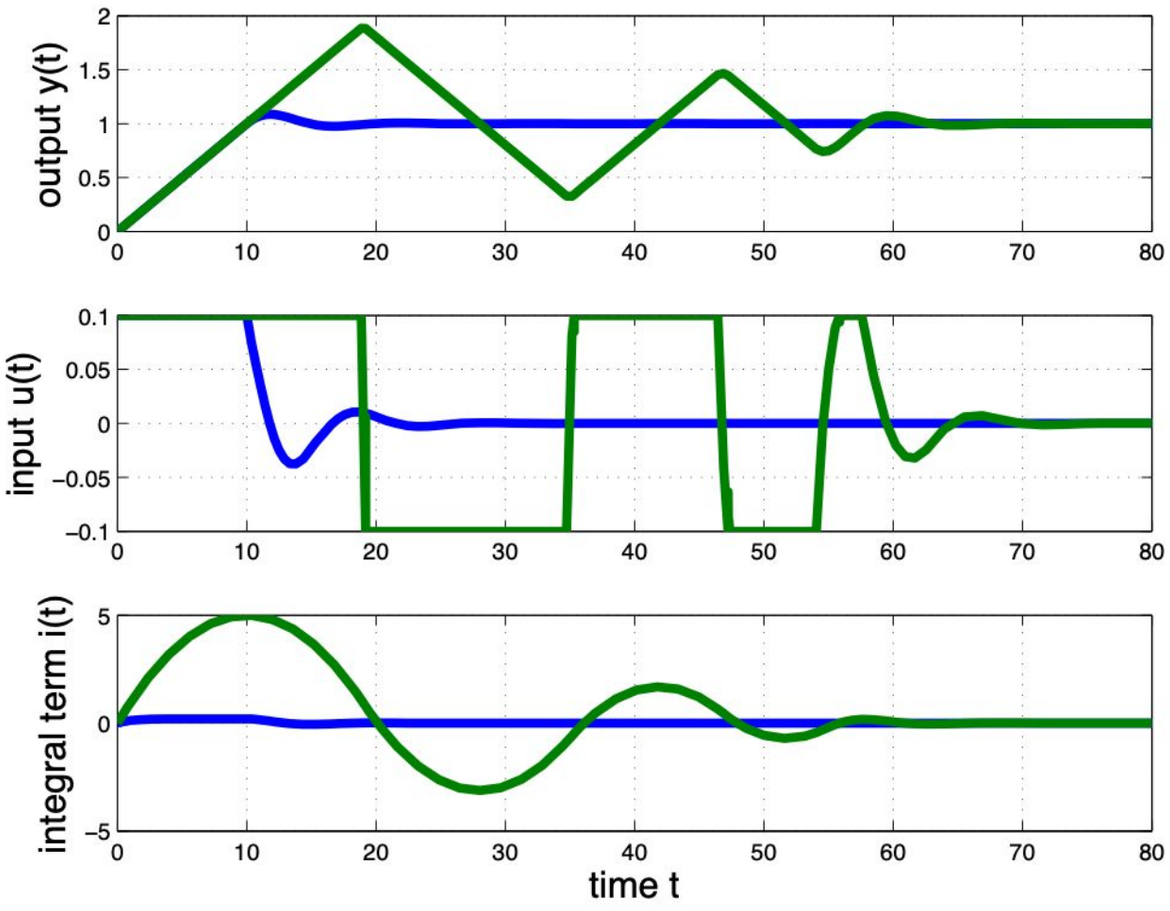


anti-windup scheme for a PID controller ($K_P = K_I = K_D = 1$, $T_t = 1$)

- Integrator windup is avoided thanks to back-calculation
- Back-calculation only requires tuning one parameter, the time constant T_t. But it only applies to PID control.

Benefits of the anti-windup scheme

- Let's look at the difference between having and not having an anti-windup scheme



- Note that in case of wind up we have:
  - Large output oscillations
  - Longer time to reach steady-state
  - Peaks of control signal

PID limitations

While being widely used, PID controllers have some limitations

- PID are feedback controllers with constant gains. Thus, their behavior is reactive and cannot vary across situations (e.g. different process conditions).
  - Gain scheduling (changing gains) can be used.

- They do not explicitly incorporate process knowledge, which, if known, can greatly increase controller performance.
  - Simple feedforward can already improve tracking performance.

- PID is based on a linear process model.
  - Linearization in current or intended operating point.

Learning outcomes

By the end of *this* lecture, you should be able to:

- Design practical PID controllers for applications
- Design anti-windup schemes to avoid wind-up