Systems Engineering/Process Control L2

- Process models
- Step-response models
- The PID controller

Reading: *Systems Engineering and Process Control:* 2.1–2.5

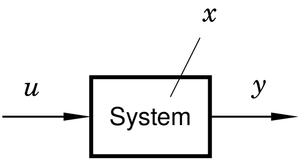
Process models

We will primarily work with processes that are described by

- **continuous** (as opposed to discrete – FX),
- **linear** (as opposed to nonlinear – F3, F5),
- **time invariant** (as opposed to time varying),
- **dynamic** (as opposed to static)

systems

Static vs dynamic systems



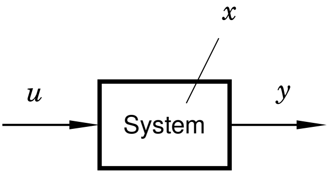
Static system: $$y(t) = f\big(u(t)\big)$$

- Output y right now depends only on input u right now
- New equilibrium is found instantaneously after input changes

Dynamic system: $$y(t) = f\big(u_{[0,\,t]},\; x(0)\big)$$

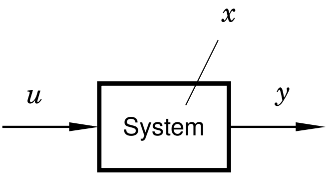
- Output $y(t)$ depends on all old inputs $u_{[0,\,t]}$ and the system initial state $x(0)$
- For (stable) dynamical systems, there is a lag before a new equilibrium is reached after an input change

Static or dynamic system?



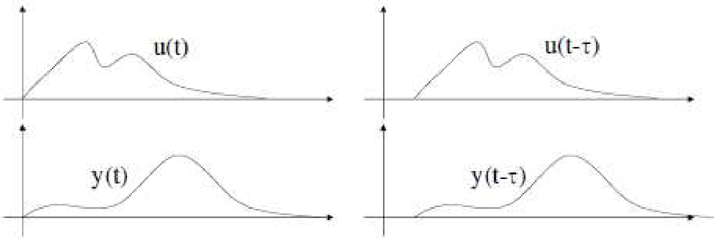
System	*Input* (u)	*Output* (y)	S/D
Shower	Temperature knob	Water temperature	D
Lamp	Light switch	Light	S
Lamp	Dimmer	Light	S
Water tank	Inflow and outflow	Water level	D
Cruise control	Throttle	Speed	D

Time invariant vs time varying systems



Time invariant system: The system dynamics does not change over time

Input delayed by τ time units $\Rightarrow$ output delayed by τ time units:



Examples of time invariant/varying systems

Time varying systems:

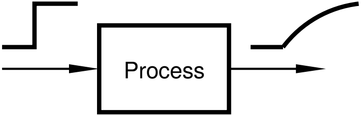
- Lamp with switch and timer: Different response depending on time
- Rockets: Decreasing fuel amount $\Rightarrow$ system dynamics change

Time invariant systems:

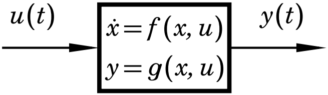
- Lamp with switch without timer
- Water tank with inflows and outflows
- Cruise control in the car

Process models used in course

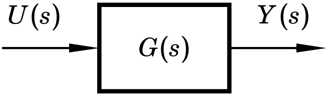
Step-response model (L2)



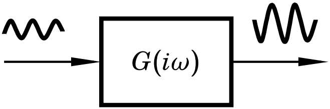
State-space model (L3)



Transfer function (L4)

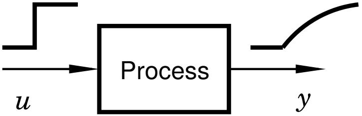


Frequency-response function (L8)



Step-response experiment

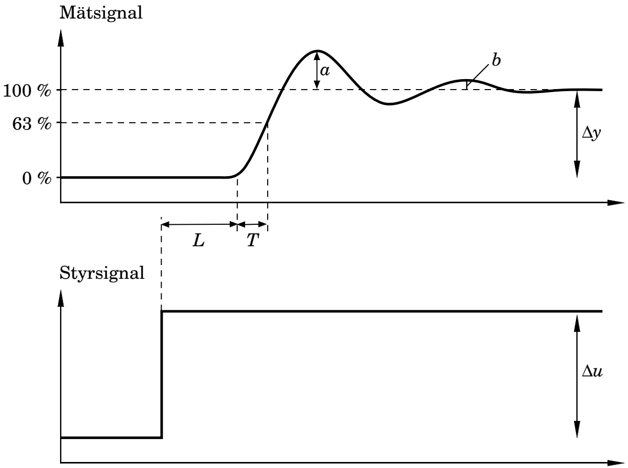
A simple method to learn the process dynamics



- Wait until process is in equilibrium
- Change input u with a step of size Δu
- Record and analyze output y

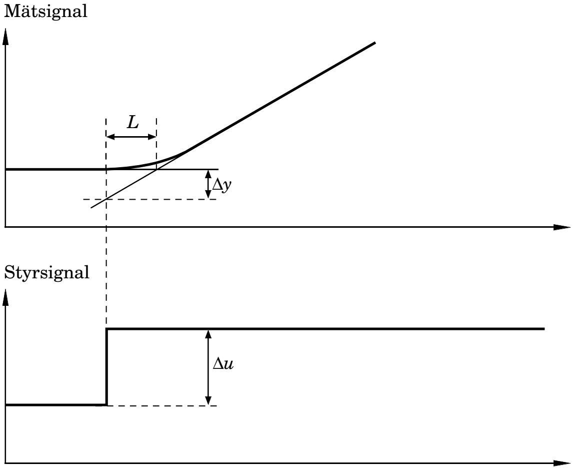
(We assume here **one** input and **one** output)

Step-response example



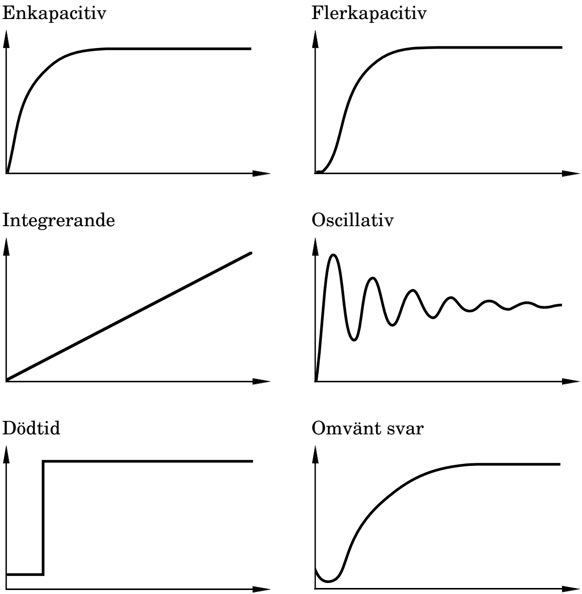
- Dead time = L
- Time constant = T
- Static gain = $K_p = \Delta y / \Delta u$
- Overshoot = $a / \Delta y$
- Damping = $1 - b/a$

Step-response for integrating process

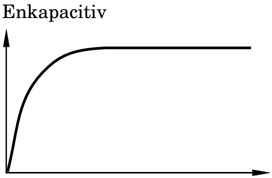


- Dead time = L
- Velocity gain = $K_v = \Delta y / (\Delta u \cdot L)$

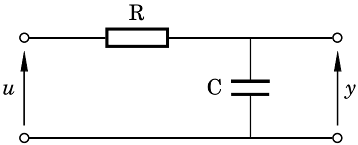
Step-response for some different process types

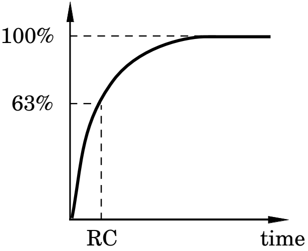


Single-capacitive processes



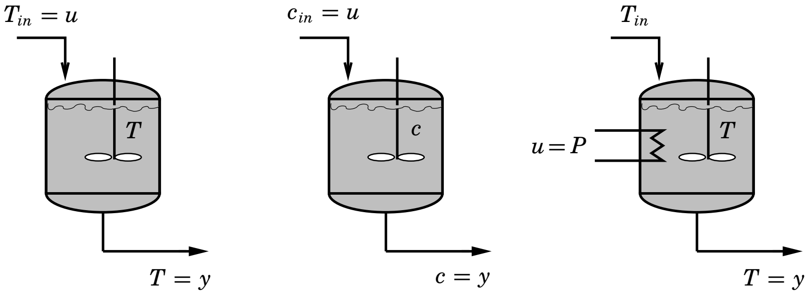
Example: RC circuit



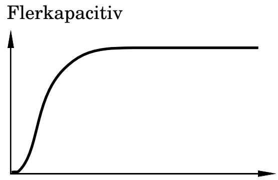


Single-capacitive processes

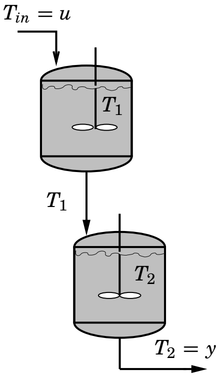
Example: Continuously stirred tank (CST) with constant flow



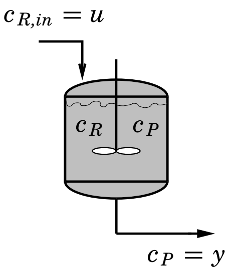
Multi-capacitive processes



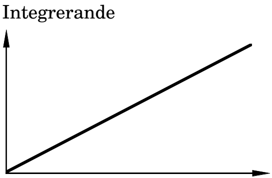
Example:



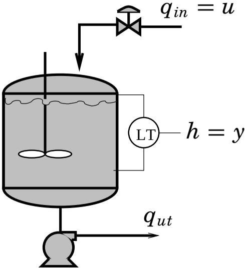
CSTR, $R \to P$

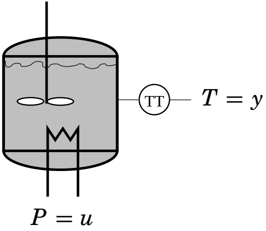


Integrating processes

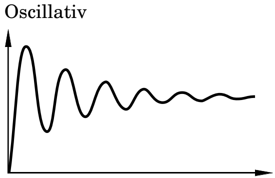


Example:

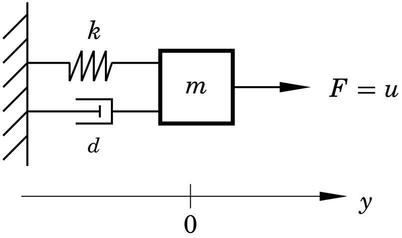




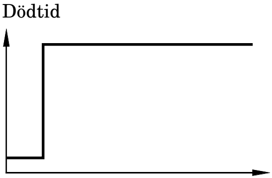
Oscillatory processes



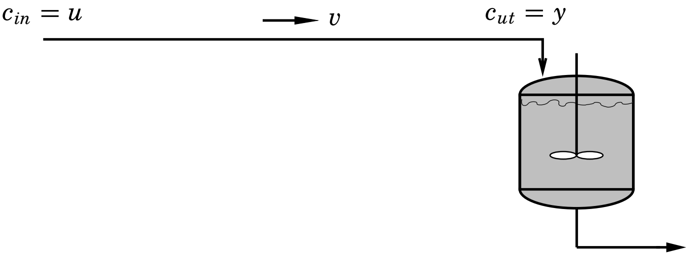
Example: Mechanical system with little damping



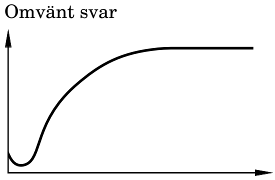
Dead time processes



Example:



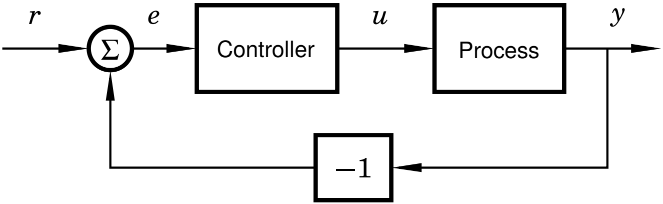
Inverse response processes



Examples:

- Parallel parking with car
  - Input: steering wheel angle
  - Measurement: (smallest) distance from *front* wheel to curb
- Bus turn
  - Input: steering wheel angle
  - Measurement: (smallest) distance from *back of bus* to curb

The standard feedback loop

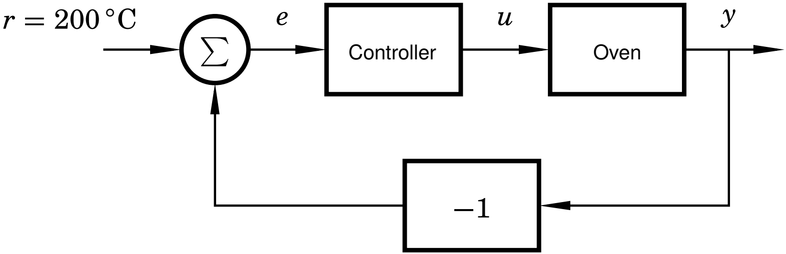


- Objective: measurement signal y should follow setpoint (reference) r
- Controller computes input u from control error $e = r - y$

Simple feedback controllers

- On/off-controller
  - The simplest feedback controller
- PID-controller
  - The most common controller in industry
  - P = proportional
  - I = integral
  - D = derivative

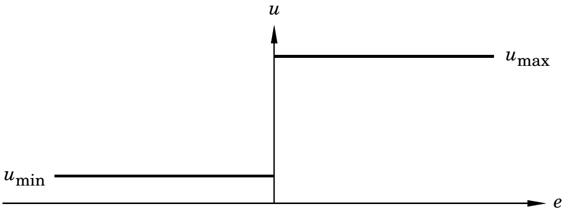
Example: Oven



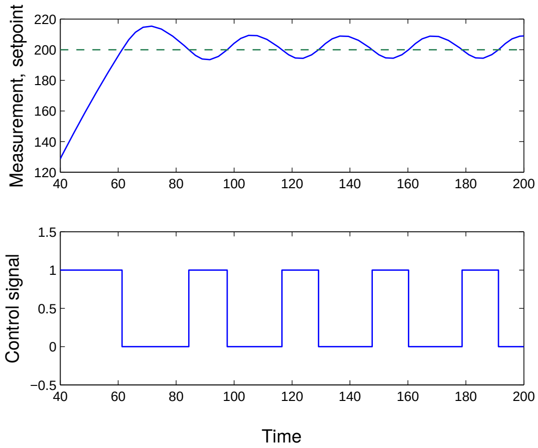
- y = measured temperature (output/measurement signal)
- r = desired temperature (setpoint/reference)
- u = heating effect $(0 \leq u \leq 1)$ (control signal/input)

On/off-control

$$u(t) = \begin{cases} u_{\max}, & \text{if } e(t) > 0 \ (i.e., y(t) < r(t)) \\ u_{\min}, & \text{if } e(t) < 0 \ (i.e., y(t) > r(t)) \end{cases}$$



Simulation of oven with on/off-control



Drawbacks with on/off-control

- Oscillations
- Wear on actuators
- Works only for processes with:
  - simple dynamics
  - low performance requirements

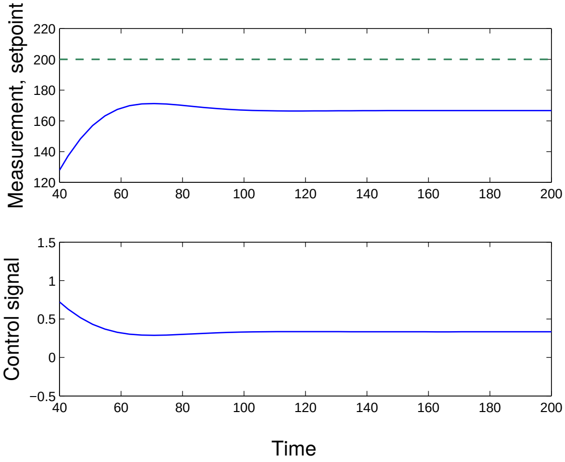
P-control

- Use proportional (to control error) control:

$$u(t) = u_0 + Ke(t)$$

- $K =$ proportional gain
- *(Simplest control structure except on/off)*

Simulation of oven with P-control ($u_0 = 0$)



- Stationary control error (at stationarity $y(t) \neq r(t)$)

Mini problem

Approximately what K-value is used in previous slide?

Stationary error with P-control

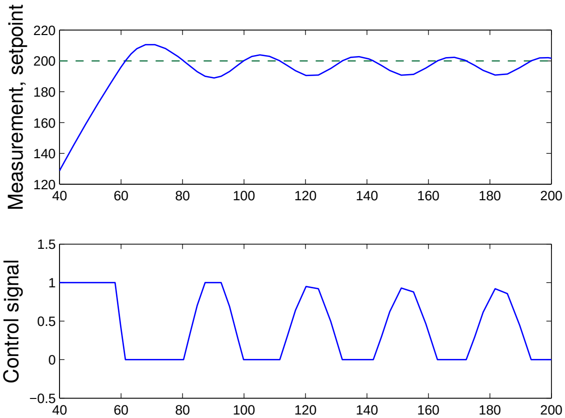
The stationary error when using a P controller is:

$$e = \frac{u - u_0}{K}$$

Two ways to eliminate stationary error (i.e., get $e = 0$):

- Let $K \to \infty$
- Select u_0 such that $e = 0$ in stationarity (difficult to find such u_0)

Simulation of P-control with increased K



- Faster control but more oscillations

PI-control

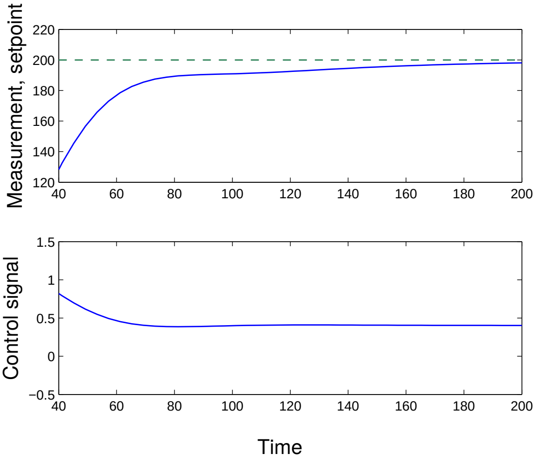
- Are there other ways to remove stationary errors?
- Update u_0 automatically: Replace the constant term u_0 with integral part:

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

  - T_i = integral time

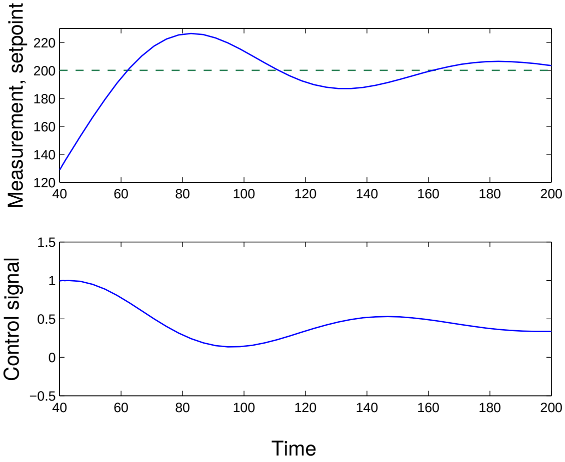
(Note: The PI-controller is a dynamical system in itself!)

Simulation of oven with PI-control



- Control error goes asymptotically towards zero
- Can prove that stationary error is always zero when using PI-control (provided closed loop system is stable)

Simulation of oven with decreased T_i

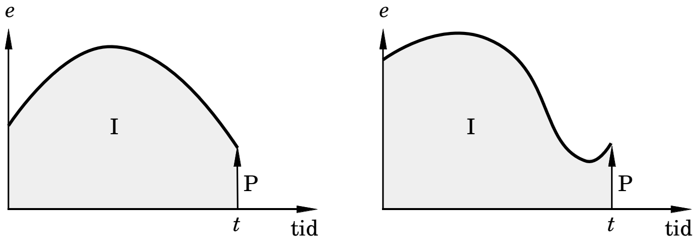


- More integral action
- Faster control but more oscillations

Prediction

A PI-controller does not predict future errors

The same control signal is obtained in both of the following cases:



Want something that can react on predicted future errors

PID-control

This can be achieved by adding a derivative (D) part to the PI controller:

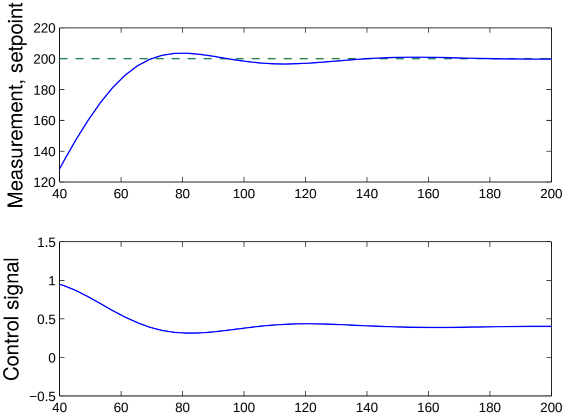
$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right)$$

- T_d = derivative time

The derivative part tries to estimate the error change in T_d time units:

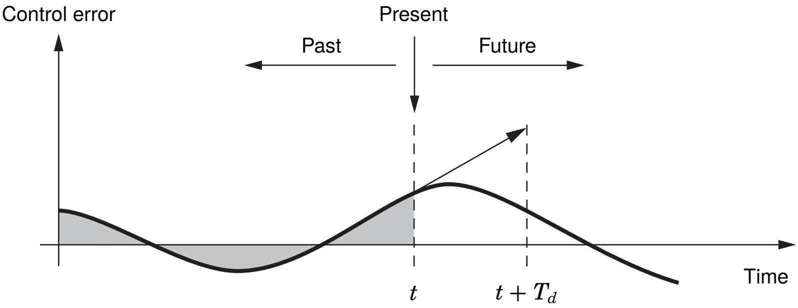
$$e(t+T_d) - e(t) \approx +T_d\frac{de(t)}{dt}$$

Simulation of oven with PID-control



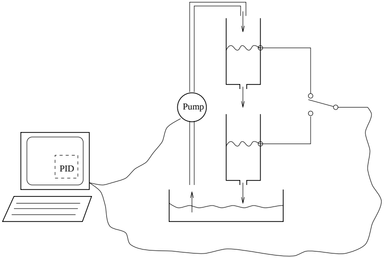
- Fast and well damped response, no stationary error

Summary of PID



The parameters to set: K, T_i, T_d

Laboration 1 – Empirical PID-control



Control of water level in upper/lower tank

- Open-loop and closed-loop control
- Manual and automatic control
- Empirical setting of K, T_i, T_d

Controller type selection

- (On/off-controller)
- P-controller
- PD-controller
- PI-controller
- PID-controller
- I-controller

P-controller

Is good enough in some cases:

- Control of single-capacitive and integrating processes
  - big K gives small stationary error; no problems with stability
- Level control in buffer tanks
  - small K as long as tank is not almost empty or almost full
- As controller in inner loop in cascade control structure (F9)

PD-controller

Suitable in some cases:

- Control of some multi-capacitive processes, e.g., slow temperature processes
- Big K and T_d requires measurements with little noise

PI-controller

The most common choice of controller

- Eliminates stationary errors
- With cautious settings (small K big T_i) it works on all stable processes including dead time processes and processes with inverted response

PID-controller

- Can give improved performance compared to PI-controller, especially for multi-capacitive and integrating-capacitive processes
  - K can be increased and T_i decreased compared to PI-control
- Derivative part is sensitive to measurement noise

I-controller

A pure I-controller is given by

$$u(t) = k_i \int_0^t e(\tau) d\tau$$

- k_i = integral gain

Can be used for static processes or single-capacitive processes to eliminate stationary errors