

TrueTime: Simulation of Networked and Embedded Control Systems

Anton Cervin

Department of Automatic Control
Lund University
Sweden

Dept. of Automatic Control at Lund University

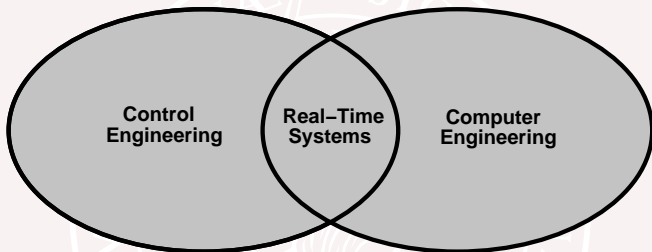


- Founded in 1965 by Karl Johan Åström (IEEE Medal of Honor in 1993)
- Approx. 50 persons

Dept. of Automatic Control at Lund University

- Basic and advanced control education for almost all engineering disciplines at the faculty of engineering (≈ 1000 students/year)
- Research in many areas, including
 - modelling and control of complex systems
 - **real-time systems and control**
 - process control
- Diverse applications:
 - robotics
 - medicine
 - telecommunication
 - automotive
 - windpower
 - ...

Real-time systems and control



- All control systems are real-time systems
- Many real-time systems are control systems

Real-time systems and control

- Control engineers need real-time systems to implement their systems.
- Computer engineers need control theory to build “controllable” systems
- Many interesting research problems in the interface

Embedded real-time control systems

- Limited computer resources
 - Cheap, embedded micro-controllers
 - Communication networks with limited bandwidth
- The computer and the network are shared resources, which must be scheduled
- Delay and jitter from the implementation \Rightarrow control performance degradation

Tentative Schedule

Wednesday:

- 09:30–12:00 Introduction to automatic control
- 13:15–15:00 TrueTime tutorial 1
- 15:15–17:00 Computer exercise 1

Thursday:

- 09:00–12:00 TrueTime tutorial 2
- 13:15–16:00 Computer exercise 2 (miniproject)

Acknowledgments

Some of the content has previously appeared in the ARTIST2 Graduate Course on Embedded Control Systems (held in Valencia 2005, Prague 2006, Lund 2007 and Stockholm 2008)

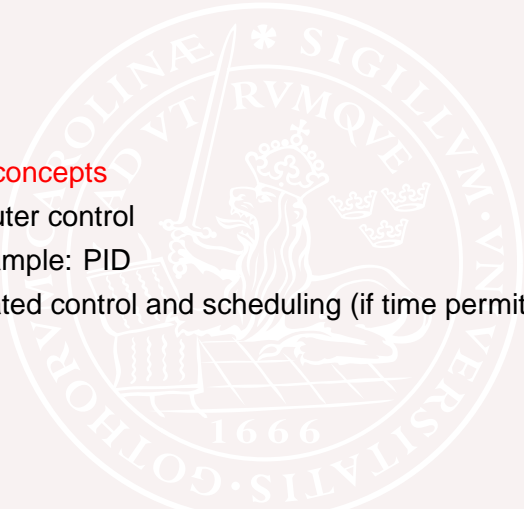
Developed in close collaboration with Karl-Erik Årzén, with important contributions from Dan Henriksson and Martin Ohlin.



Lecture 1

Introduction to Automatic Control

Outline of Lecture

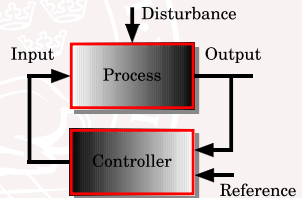
- 
- 1 Basic concepts
 - 2 Computer control
 - 3 An example: PID
 - 4 Integrated control and scheduling (if time permits)

Automatic control

Use of **models** and **feedback**

Activities:

- Modeling
- Analysis
- Simulation
- Control design
- Implementation



Automatic control

Sometimes called “the hidden technology”:

- Widely used
- Very successful
- Seldom talked about, except when disaster strikes!



What control system is (was!) this?

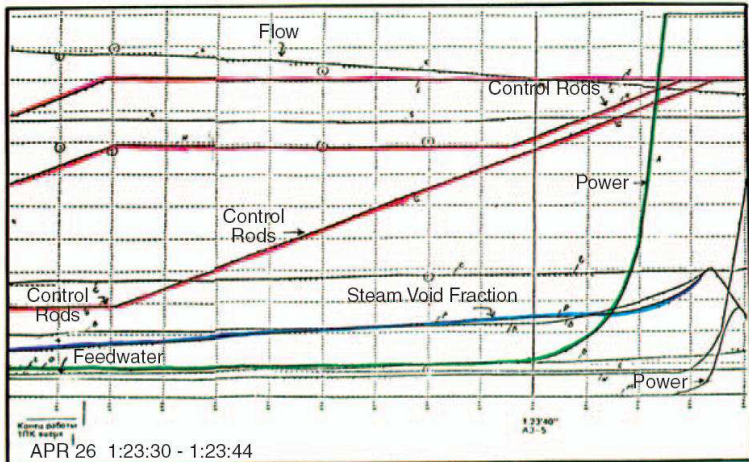
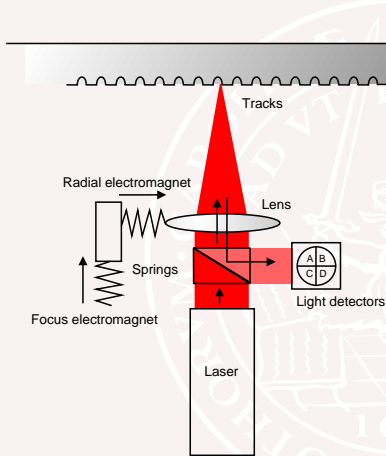




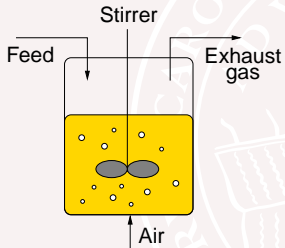
Figure 2. *Chernobyl nuclear power plant shortly after the accident on 26 April 1986.*

Example: track following in a DVD player



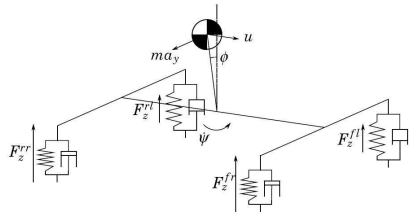
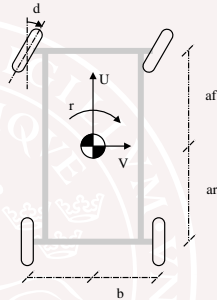
[Bo Lincoln, Automatic Control LTH, 2003]

Example: optimal growth of bacteria



[Lena de Maré, Automatic Control LTH, 2006]

Example: stabilization of vehicle dynamics



[Brad Schofield, Automatic Control LTH, 2007]

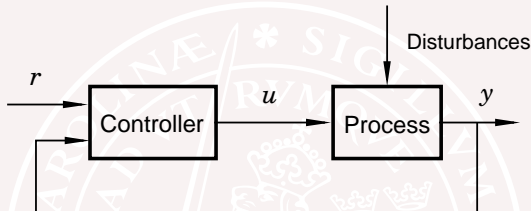
More examples of control

- Control of the economy using the central bank interest rate
- Control of the blood glucose level in the human body
- Congestion control in the TCP protocol

Recent textbook aimed at computer science students:

Hellerstein, Diao, Parekh and Tilbury (2004): *Feedback Control of Computing Systems*

Basic Setting



Must handle two tasks:

- Make the measurement signal y follow the reference r
- Compensate for disturbances

How to

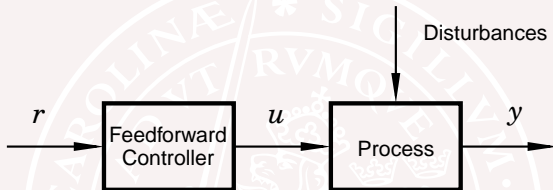
- do several things with the control signal u

Two fundamental control principles

- Feedforward control
- Feedback control

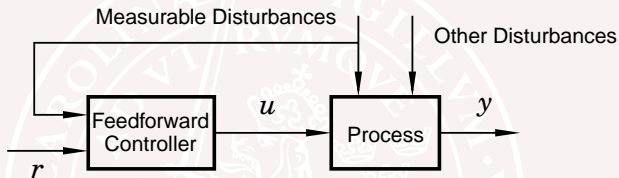


Feedforward control



- Adjust the control signal based on the reference signal, using knowledge of how the process works
- Open loop
- Real-world examples?

Feedforward from Measurable Disturbances



- If some disturbances are measurable, they may be compensated for
- Corrective action before an error has occurred in the process output

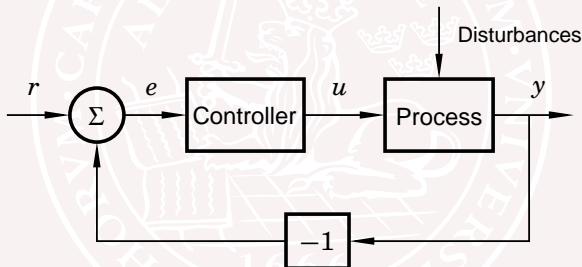
Properties of Feedforward Control

- + Allows fast response to set-point changes
- + Allows efficient suppression of measurable disturbances
- Requires an accurate process model
- Requires a stable process

Feedback Control

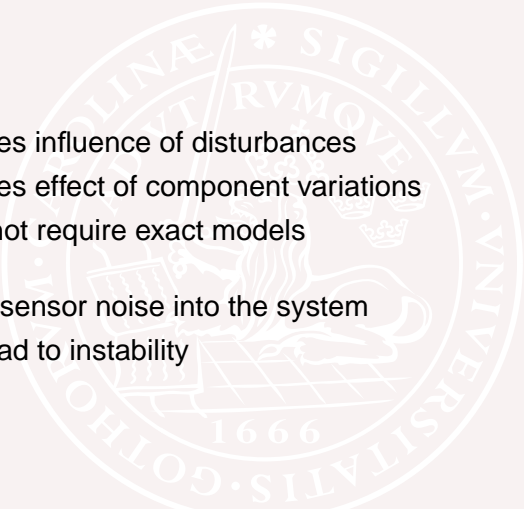
A very powerful principle, that often leads to revolutionary changes in the way systems are designed

The primary paradigm in automatic control

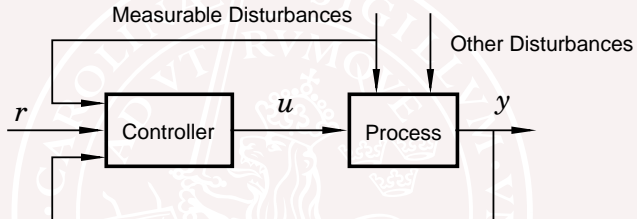


- Corrective action based on an error that has occurred
- Closed loop

Properties of Feedback Control

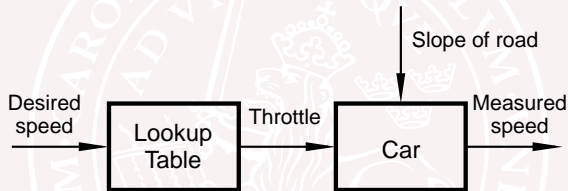
- 
- + Reduces influence of disturbances
 - + Reduces effect of component variations
 - + Does not require exact models
 - Feeds sensor noise into the system
 - May lead to instability

Putting It All Together



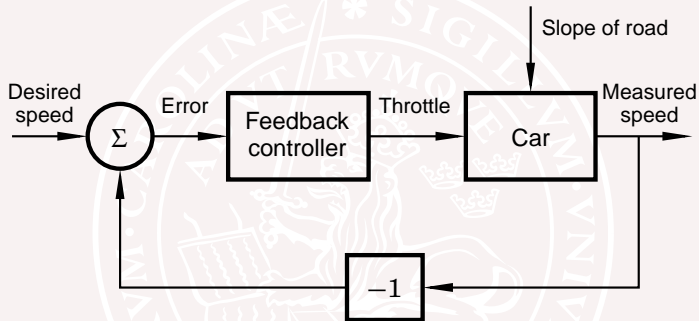
A good controller uses both **feedback** and **feedforward**

Example: Cruise Control Using Feedforward



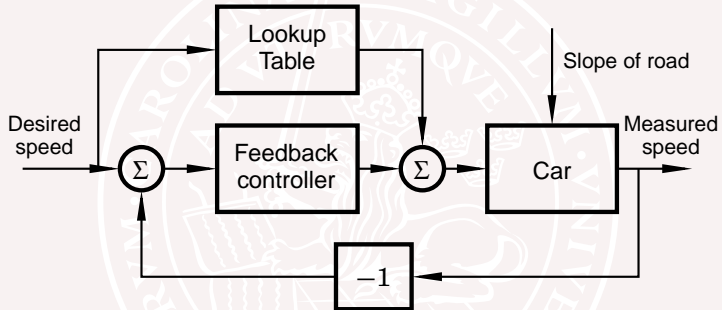
- Open loop
- Problems?

Example: Cruise Control Using Feedback



- Closed loop
- Controller:
 - Error > 0 : increase throttle
 - Error < 0 : decrease throttle
 - But how much?

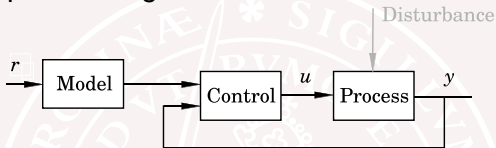
Exempel: Cruise Control Using Combination



- Both proactive and reactive

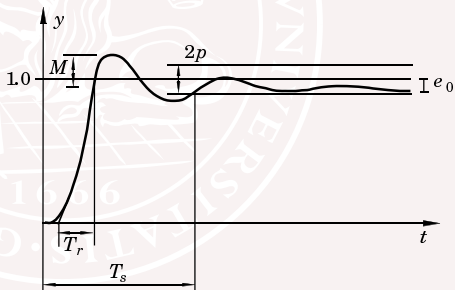
The servo problem

Focus on setpoint changes:



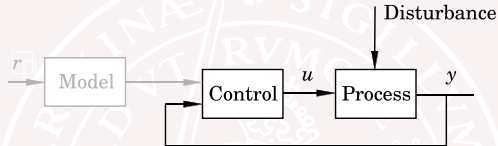
Typical design criteria:

- Rise time, T_r
- Overshoot, M
- Settling time, T_s
- Steady-state error, e_0
- ...



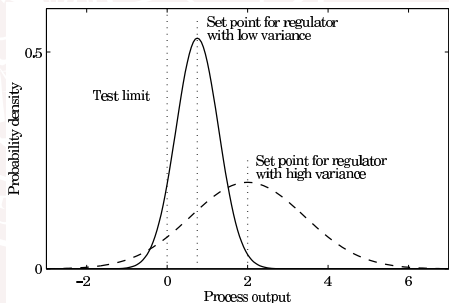
The regulator problem

Focus on process disturbances:



Typical design criteria:

- Output variance
- Control signal variance



Mathematical Models

Time domain:

- Differential equations, e.g.

$$\ddot{y} + a_1\dot{y} + a_2y = b_0\ddot{u} + b_1\dot{u}$$

- State space form

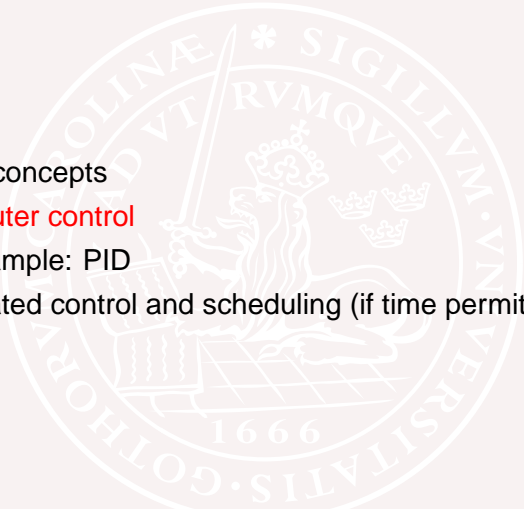
$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

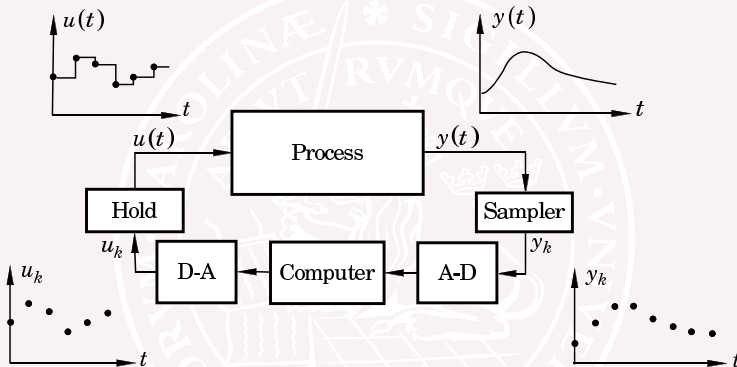
Frequency domain (linear systems only):

- Laplace transform of signals and systems
- Transfer function, $G(s) = C(sI - A)^{-1}B + D$
- Frequency response, $G(i\omega)$

Outline of Lecture

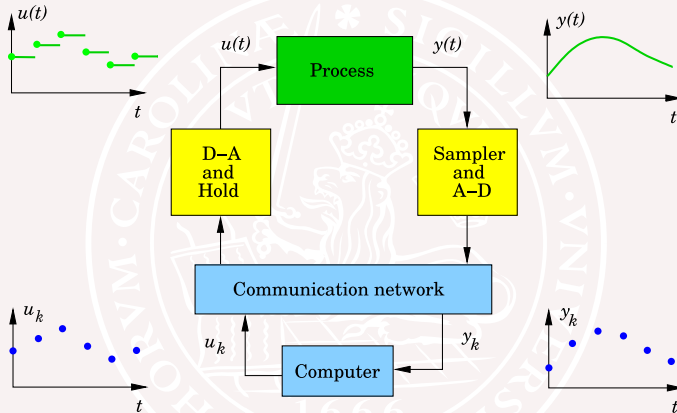
- 
- 1 Basic concepts
 - 2 **Computer control**
 - 3 An example: PID
 - 4 Integrated control and scheduling (if time permits)

Computer-controlled systems



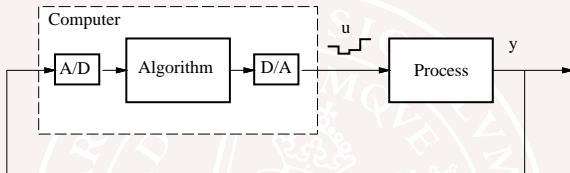
- Mix of continuous-time and discrete-time signals

Networked control systems

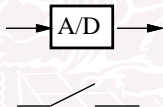


- Extra delay, possibly lost packets

Sampling



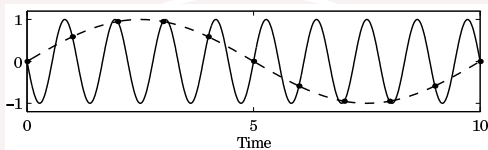
AD-converter acts as sampler



DA-converter acts as a hold device

Normally, zero-order-hold is used \Rightarrow piecewise constant control signals

Aliasing



$\omega_s = \frac{2\pi}{h} = \text{sampling frequency}$

$\omega_N = \frac{\omega_s}{2} = \text{Nyquist frequency}$

Frequencies above the Nyquist frequency are folded and appear as low-frequency signals.

The fundamental alias for a frequency f_1 is given by

$$f = |(f_1 + f_N) \bmod (f_s) - f_N|$$

Above: $f_1 = 0.9$, $f_s = 1$, $f_N = 0.5$, $f = 0.1$

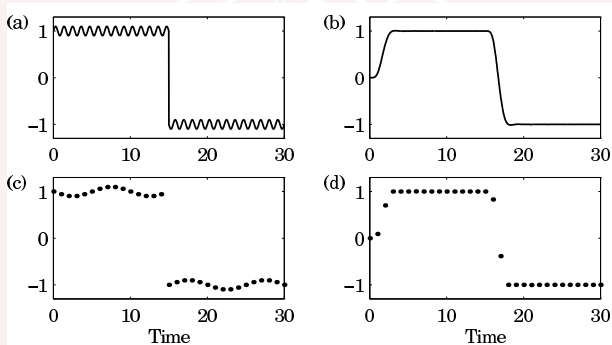
Anti-aliasing filter

Analog low-pass filter that eliminates all frequencies above the Nyquist frequency

- Analog filter
 - 2-6th order Bessel or Butterworth filter
 - Difficulties with changing h (sampling interval)
- Analog + digital filter
 - Fixed, fast sampling with fixed analog filter
 - Downsampling using digital LP-filter
 - Control algorithm at the lower rate
 - Easy to change sampling interval

The filter may have to be included in the control design

Example – Prefiltering



$$\omega_d = 0.9, \omega_N = 0.5, \omega_{alias} = 0.1$$

6th order Bessel filter with $\omega_B = 0.25$

Design approaches

Digital controllers can be designed in two different ways:

- Discrete-time design – sampled control theory
 - Sample the continuous system
 - Design a digital controller for the sampled system
 - Z-transform domain
 - discrete state-space domain
- Continuous time design + discretization
 - Design a continuous controller for the continuous system
 - Approximate the continuous design
 - Use fast sampling

Disk drive example

Control of the arm of a disk drive (double integrator)

$$G(s) = \frac{k}{Js^2}$$

Continuous time controller

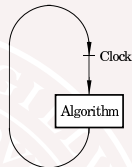
$$U(s) = \frac{bK}{a} U_c(s) - K \frac{s+b}{s+a} Y(s)$$

Discretized controller

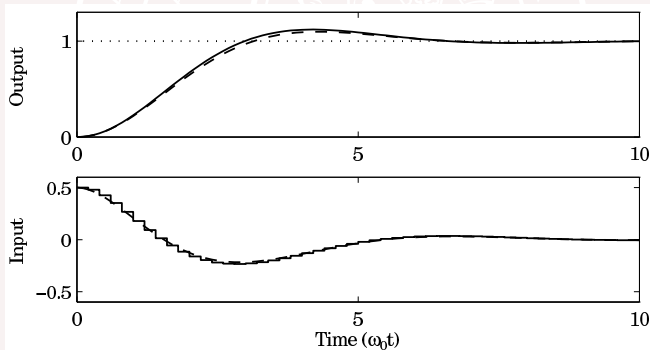
$$\begin{aligned} u(t_k) &= K \left(\frac{b}{a} u_c(t_k) - y(t_k) + x(t_k) \right) \\ x(t_k + h) &= x(t_k) + h \left((a-b)y(t_k) - ax(t_k) \right) \end{aligned}$$

Disk drive example

```
y := adin(in2)
u := K*(b/a*uc-y+x)
dout(u)
x := x+h*((a-b)*y-a*x)
```

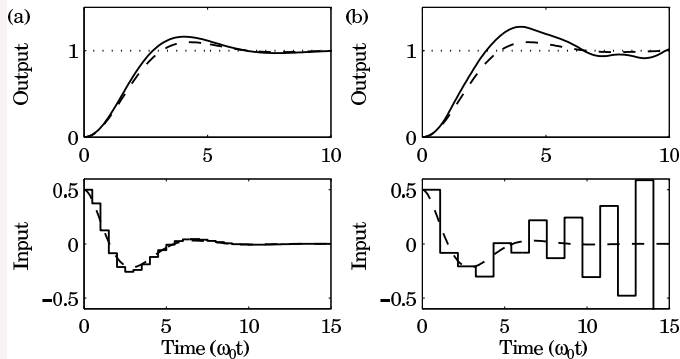


Sampling period $h = 0.2/\omega_0$



Increased sampling period

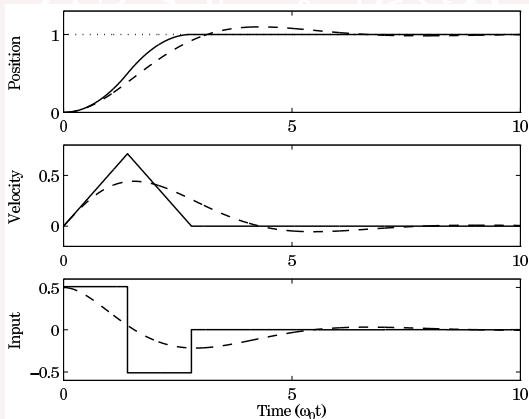
a) $h = 0.5/\omega_0$ b) $h = 1.08/\omega_0$



Better performance?

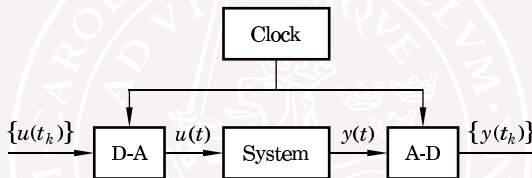
Dead-beat control, $h = 1.4/\omega_0$

$$u(t_k) = t_0 u_c(t_k) + t_1 u_c(t_{k-1}) - s_0 y(t_k) - s_1 y(t_{k-1}) - r_1 u(t_{k-1})$$



Sampling of systems

Look at the system from the point of view of the computer



Zero-order-hold sampling

- Let the inputs be piecewise constant
- Look at the sampling points t_k only
- Solve the system equation

Sampling a continuous-time system

Process:

$$\begin{aligned}\frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

Solve the system equation:

$$\begin{aligned}x(t) &= e^{A(t-t_k)}x(t_k) + \int_{t_k}^t e^{A(t-s')}Bu(s')ds' \\ &= e^{A(t-t_k)}x(t_k) + \int_{t_k}^t e^{A(t-s')}ds' Bu(t_k) \quad (u \text{ const.}) \\ &= e^{A(t-t_k)}x(t_k) + \int_0^{t-t_k} e^{As}ds Bu(t_k) \quad (\text{variable change}) \\ &= \Phi(t, t_k)x(t_k) + \Gamma(t, t_k)u(t_k)\end{aligned}$$

Periodic sampling

Assume periodic sampling, i.e. $t_k = kh$. Then

$$x(kh + h) = \Phi x(kh) + \Gamma u(kh)$$

$$y(kh) = Cx(kh) + Du(kh)$$

where

$$\Phi = e^{Ah}$$

$$\Gamma = \int_0^h e^{As} ds B$$

Time-invariant linear system!

Example: Sampling of double integrator

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

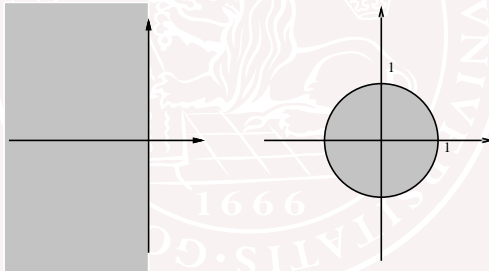
We get

$$\Phi = e^{Ah} = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix}$$
$$\Gamma = \int_0^h \begin{pmatrix} s \\ 1 \end{pmatrix} ds = \begin{pmatrix} h^2/2 \\ h \end{pmatrix}$$

Several ways to calculate Φ and Γ . Matlab

Stability region

- In continuous time the stability region is the complex left half plane, i.e., the system is stable if all the poles are in the left half plane.
- In discrete time the stability region is the unit circle.



Control design

A large variety of control design methods are available in digital control theory, e.g.:

- state-feedback control – pole-placement
- LQ control
- observer-based state feedback control
- LQG control
- ...

Outside the scope of this course

Computational Delay

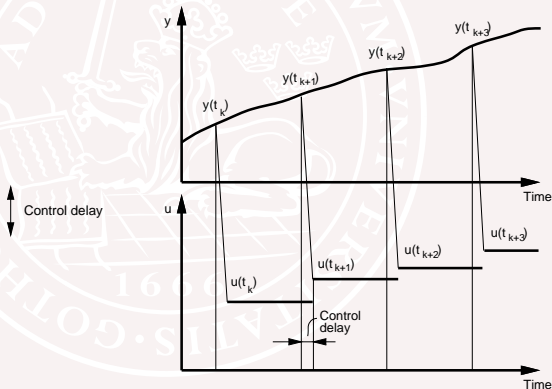
Problem: $u(t_k)$ cannot be generated instantaneously at time t_k when $y(t_k)$ is sampled

Control delay (computational delay) due to computation time

LOOP

wait for clock interrupt;
read analog input;
perform calculations;
set analog output;

END;



Three approaches

1. Ignore the computational delay
 - often justified, if it is small compared to h
 - write the code so that the delay is minimized, i.e., minimize the operations performed between AD and DA
 - divide the code into two parts: CalculateOutput and UpdateStates
2. Design the controller to be robust against variations in the computational delay
 - complicated
3. Compensate for the computational delay
 - include the computational delay in model and the design
 - write the code so that the delay is constant, e.g. one sample delay

Minimize Control Delays

General controller representation:

$$x(k+1) = Fx(k) + Gy(k) + G_r y_{ref}(k)$$

$$u(k) = Cx(k) + Dy(k) + D_r y_{ref}(k)$$

Do as little as possible between AdIn and DaOut:

```
PROCEDURE Regulate;  
BEGIN  
  AdIn(y);  
  (* CalculateOutput *)  
  u := u1 + D*y + Dr*yref;  
  DaOut(u);  
  (* UpdateStates *)  
  x := F*x + G*y + Gr*yref;  
  u1 := C*x;  
END Regulate;
```

Sampling Interval

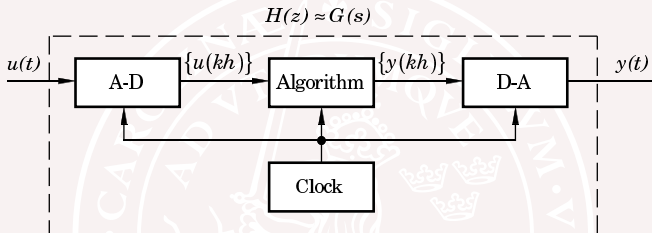
Number of samples per rise time, T_r , of the closed loop system

$$N_r = \frac{T_r}{h} \approx 4 - 10$$

With long sampling intervals it may take long before disturbances are detected

Discretization of continuous-time controllers

Basic idea: Reuse the analog design



Want to get:

- $A/D + \text{Algorithm} + D/A \approx G(s)$

Methods:

- Approximate derivatives by differences
- Other discretization methods (Matlab)

Some common discretization methods

Forward Difference (Euler's method):

$$\frac{dx(t)}{dt} \approx \frac{x(t_{k+1}) - x(t_k)}{h}$$

Backward Difference:

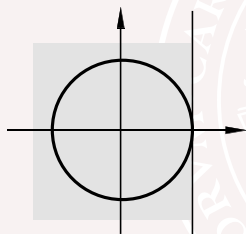
$$\frac{dx(t)}{dt} \approx \frac{x(t_k) - x(t_{k-1})}{h}$$

Tustin:

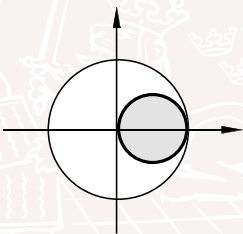
$$\frac{\frac{dx(t)}{dt} + \frac{dx(t_{k+1})}{dt}}{2} \approx \frac{x(t_{k+1}) - x(t_k)}{h}$$

Stability of discretizations

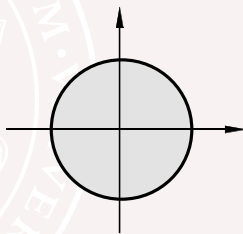
How is the continuous-time stability region (left half plane) mapped?



Forward differences

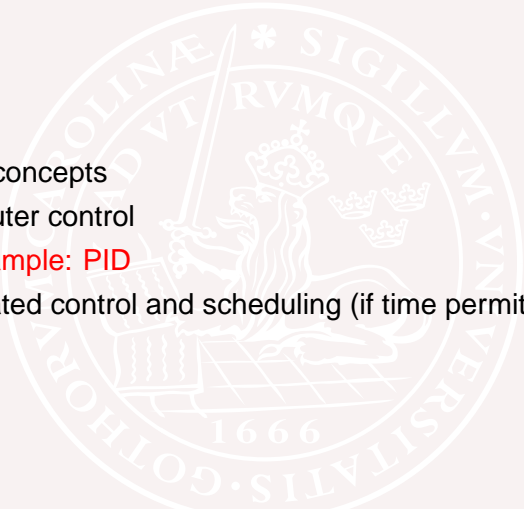


Backward differences



Tustin

Outline of Lecture

- 
- 1 Basic concepts
 - 2 Computer control
 - 3 **An example: PID**
 - 4 Integrated control and scheduling (if time permits)

An Example: PID Control

Proportional-Integral-Derivative control

- The oldest controller type (early 1900's)
- The most widely used
 - Pulp & paper 86%
 - Steel 93%
 - Oil refineries 93%
- Much to learn!

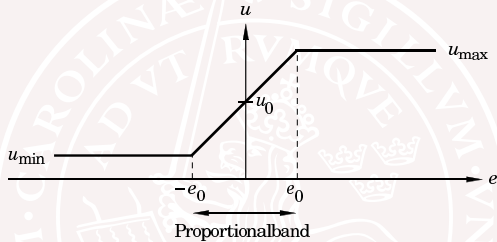
The Textbook Algorithm

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

$$U(s) = K E(s) + \frac{K}{s T_i} E(s) + K T_d s E(s)$$

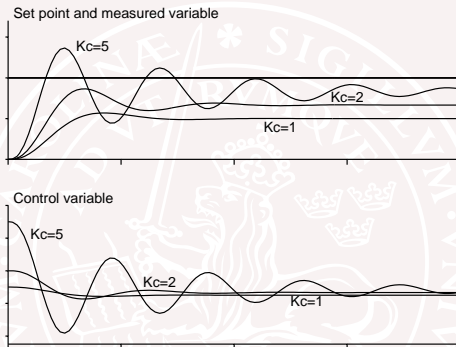
$$= P + I + D$$

Proportional Term



$$u = \begin{cases} u_{\max} & e > e_0 \\ K e + u_0 & -e_0 < e < e_0 \\ u_{\min} & e < -e_0 \end{cases}$$

Properties of P-Control



- stationary error
- increased K means faster speed, increased noise sensitivity, worse stability

Errors with P-control

Control signal:

$$u = Ke + u_0$$

Error:

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

Error removed if:

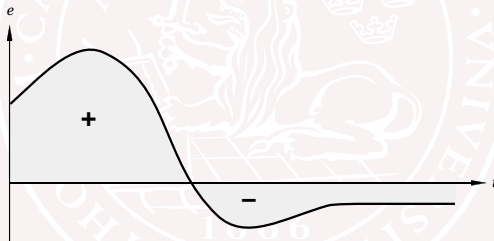
- 1 K equals infinity
- 2 $u_0 = u$

Solution: Automatic way to obtain u_0

Integral Term

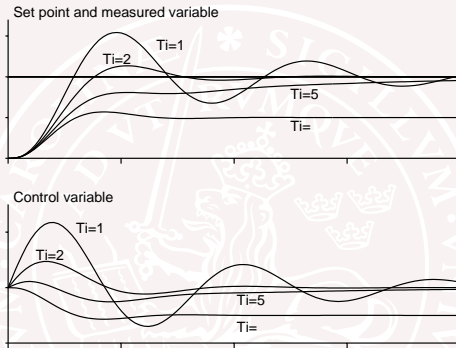
$$u = Ke + u_0$$

$$u = K \left(e + \frac{1}{T_i} \int e(t) dt \right) \quad (\text{PI})$$



Stationary error present $\rightarrow \int e dt$ increases $\rightarrow u$ increases $\rightarrow y$ increases \rightarrow the error is not stationary

Properties of PI-Control

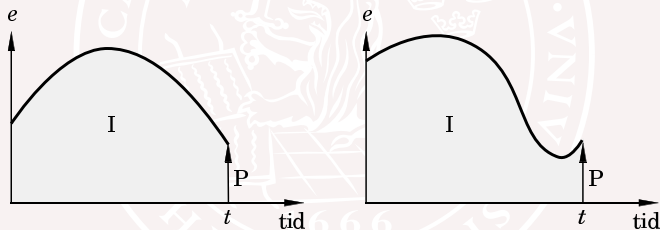


- removes stationary error
- smaller T_i implies worse stability, faster steady-state error removal

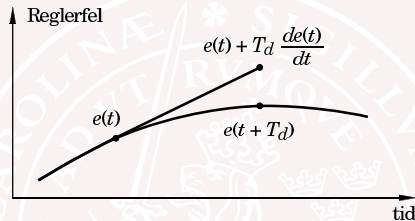
Prediction

A PI-controller contains no prediction

The same control signal is obtained for both these cases:



Derivative Part



P:

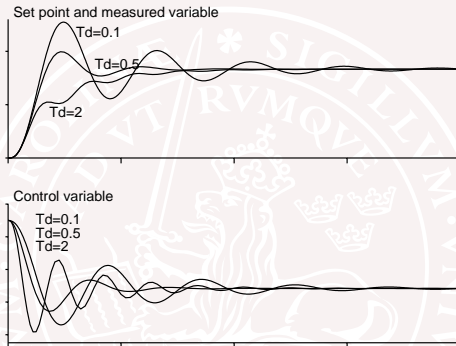
$$u(t) = K e(t)$$

PD:

$$u(t) = K \left(e(t) + T_d \frac{de(t)}{dt} \right) \approx K e(t + T_d)$$

T_d = Prediction horizon

Properties of PD-Control



- T_d too small, no influence
- T_d too large, decreased performance

In industrial practice the D-term is often turned off.

Algorithm Modifications

Modifications are needed to make the controller practically useful

- Limitations of derivative gain
- Derivative weighting
- Setpoint weighting
- Handle control signal limitations

Limitations of derivative gain

We do not want to apply derivation to high frequency measurement noise, therefore the following modification is used:

$$sT_d \approx \frac{sT_d}{1 + sT_d/N}$$

N = maximum derivative gain, often 10 – 20

Derivative weighting

The setpoint is often constant for long periods of time

Setpoint often changed in steps → D-part becomes very large.

Derivative part applied on part of the setpoint or only on the measurement signal.

$$D(s) = \frac{sT_d}{1 + sT_d/N} (\gamma Y_{sp}(s) - Y(s))$$

Often, $\gamma = 0$ in process control, $\gamma = 1$ in servo control

Setpoint weighting

An advantage to also use weighting on the setpoint.

$$u = K(y_{sp} - y)$$

replaced by

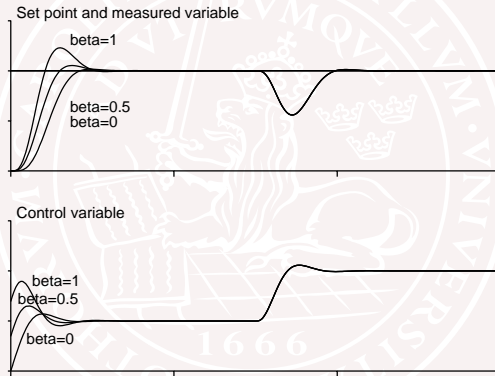
$$u = K(\beta y_{sp} - y)$$

$$0 \leq \beta \leq 1$$

A way of introducing feedforward from the reference signal

Improved set-point responses.

Setpoint weighting



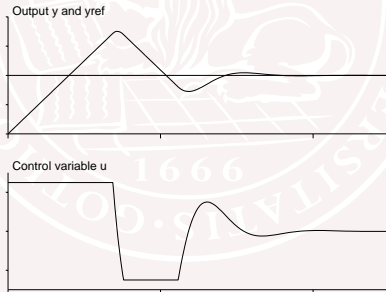
Control Signal Limitations

All actuators saturate.

Problems for controllers with integration.

When the control signal saturates the integral part will continue to grow – integrator (reset) windup.

When the control signal saturates the integral part will integrate up to a very large value. This may cause large overshoots.



Anti-Reset Windup

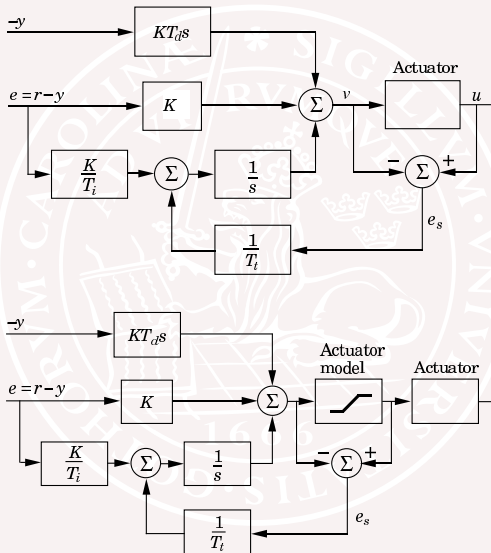
Several solutions exist:

- limit the setpoint variations (saturation never reached)
- conditional integration (integration is switched off when the control is far from the steady-state)
- tracking (back-calculation)

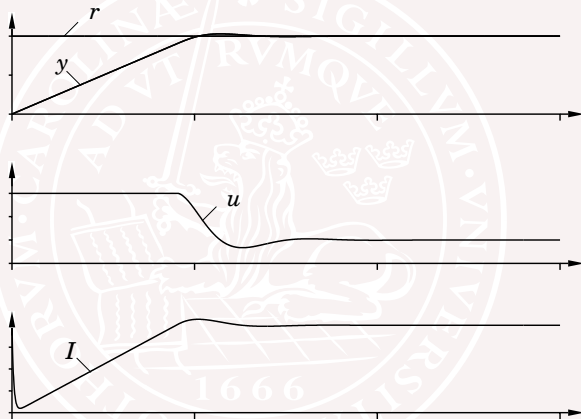
Tracking

- when the control signal saturates, the integral is recomputed so that its new value gives a control signal at the saturation limit
- to avoid resetting the integral due to, e.g., measurement noise, the recomputation is done dynamically, through a LP-filter with a time constant T_t .

Tracking



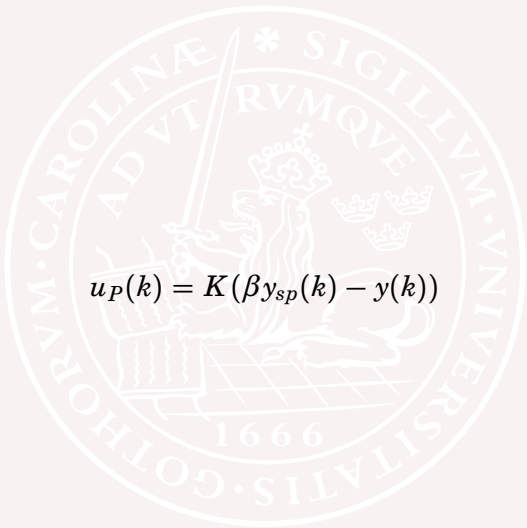
Tracking



Discretization

P-part:

$$u_P(k) = K(\beta y_{sp}(k) - y(k))$$



Discretization

I-part:

$$I(t) = \frac{K}{T_i} \int_0^t e(\tau) d\tau$$
$$\frac{dI}{dt} = \frac{K}{T_i} e$$

- Forward difference

$$\frac{I(t_{k+1}) - I(t_k)}{h} = \frac{K}{T_i} e(t_k)$$

$$I(k+1) := I(k) + (K \cdot h / T_i) \cdot e(k)$$

The I-part can be precalculated in UpdateStates

- Backward difference

The I-part cannot be precalculated, $i(k) = f(e(k))$

Discretization

D-part (assume $\gamma = 0$):

$$D = K \frac{sT_d}{1 + sT_d/N} (-Y(s))$$

$$\frac{T_d}{N} \frac{dD}{dt} + D = -KT_d \frac{dy}{dt}$$

- Forward difference (unstable for small T_d)
- Backward difference

$$\frac{T_d}{N} \frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -KT_d \frac{y(t_k) - y(t_{k-1})}{h}$$

$$D(t_k) = \frac{T_d}{T_d + Nh} D(t_{k-1}) - \frac{KT_d N}{T_d + Nh} (y(t_k) - y(t_{k-1}))$$

Discretization

Tracking:

```
v := P + I + D;  
u := sat(v,umax,umin);  
I := I + (K*h/Ti)*e + (h/Tt)*(u - v);
```

Tuning

Parameters: $K, T_i, T_d, N, \beta, \gamma, T_t$

Methods:

- empirically, rules of thumb, tuning charts
- model-based tuning, e.g., pole-placement
- automatic tuning experiments
 - Ziegler–Nichols' methods
 - relay method

PID code

PID-controller with anti-reset windup ($\gamma = 0$).

```
y = yIn.get();  
e = yref - y;  
D = ad * D - bd * (y - yold);  
v = K*(beta*yref - y) + I + D;  
u = sat(v,umax,umin)  
uOut.put(u);  
I = I + (K*h/Ti)*e + (h/Tt)*(u - v);  
yold = y
```

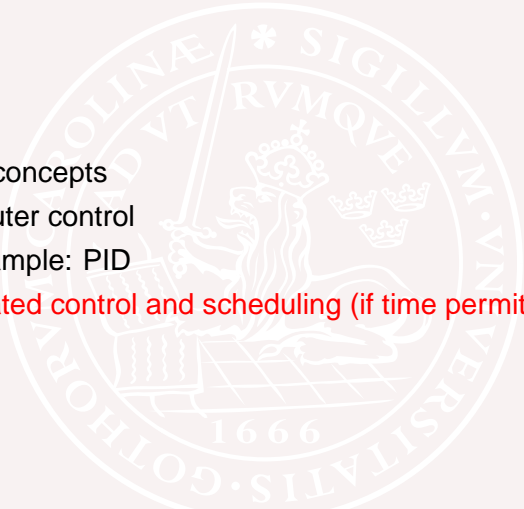
ad and *bd* are precalculated parameters given by the backward difference approximation of the D-term.

Industrial Reality

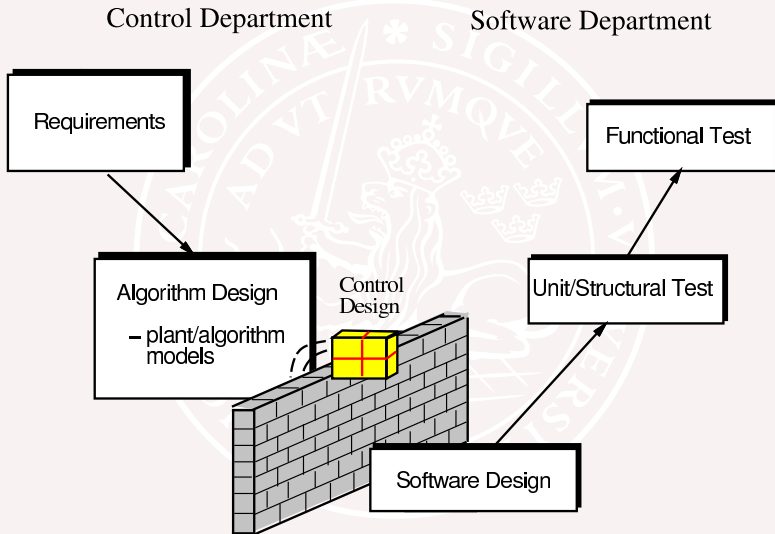
Canadian paper mill audit. Average paper mill: 2000 loops, 97% use PI, remaining 3% are PID, adaptive, ...

- default settings often used
- poor performance due to bad tuning and actuator problems

Outline of Lecture

- 
- 1 Basic concepts
 - 2 Computer control
 - 3 An example: PID
 - 4 Integrated control and scheduling (if time permits)

Control system development today



Problems

- The control engineer does not care about the implementation
 - “trivial”
 - “buy a fast computer”
- The software engineer does not understand controller timing
 - “ $\tau_i = (T_i, D_i, C_i)$ ”
 - “hard deadlines”
- Control theory and real-time scheduling theory have evolved as separate subjects for thirty years

In the beginning...

Liu and Layland (1973): “Scheduling algorithms for multiprogramming in a hard-real-time environment.” *Journal of the ACM*, **20**:1.

- Rate-monotonic (RM) scheduling
- Earliest-deadline-first (EDF) scheduling
- Motivated by process control
 - Samples “arrive” periodically
 - Control response computed before end of period
 - “Any control loops closed within the computer must be designed to allow at least an extra unit sample delay.”

Common assumptions about control tasks

In the simple task model, a task τ_i is described by

- a fixed period T_i
- a fixed, known worst-case execution time C_i
- a hard relative deadline $D_i = T_i$

Is this model suitable for control tasks?

Fixed period?

Not necessarily:

- Different sampling periods could be appropriate for different operating modes
- Some controllers are not sampled against time but are invoked by *events*
- The sampling period could be adjusted on-line by a supervisory task (“feedback scheduling”)

Fixed and known WCET?

Not always:

- WCET analysis is a very hard problem
 - May have to use estimates or measurements
- Some controllers switch between modes with very different execution times
 - Hybrid controllers
- Some controllers can explicitly trade off execution time for quality of control
 - “Any-time” optimization algorithms
 - Model-predictive control (MPC)
 - Long execution time \Rightarrow high quality of control

Hard deadlines?

Often not:

- Controller deadlines are often **firm** rather than hard
 - Often OK to miss a few outputs, but not too many in a row
 - Depends on what happens when a deadline is missed:
 - Task is allowed to complete late – often OK
 - Task is aborted at the deadline – worse
- At the same time, meeting all deadlines does not guarantee stability of the control loop
 - $D_i = T_i$ is motivated by runability conditions only

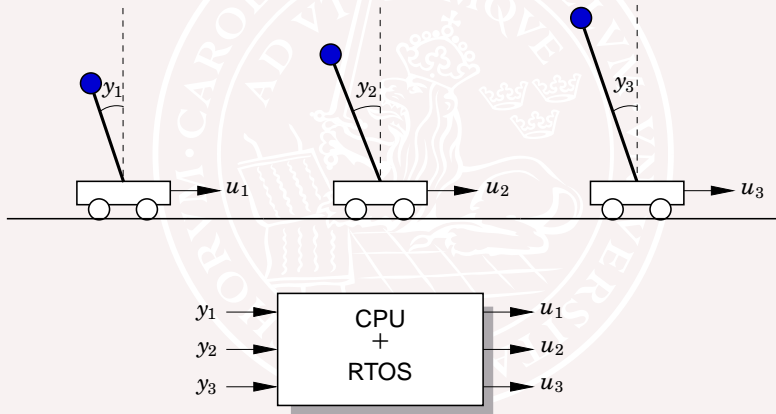
Inputs and outputs?

Completely missing from the simple task model:

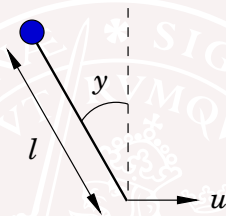
- When are the inputs (measurement signals) read?
 - Beginning of period?
 - When the task starts?
- When are the outputs (control signals) written?
 - When the task finishes?
 - End of period?

Inverted pendulum example

Control of three inverted pendulums using one CPU:



The pendulums



A simple second-order model is given by

$$\frac{d^2 y}{dt^2} = \omega_0^2 \sin y + u \omega_0^2 \cos y$$

where $\omega_0 = \sqrt{\frac{g}{l}}$ is the natural frequency of the pendulum.

Lengths $l = \{1, 2, 3\}$ cm $\Rightarrow \omega_0 = \{31, 22, 18\}$ rad/s

Control design

Linearization around the upright equilibrium gives the state-space model

$$\begin{aligned}\frac{dx}{dt} &= \begin{pmatrix} 0 & 1 \\ \omega_0^2 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ \omega_0^2 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} x\end{aligned}$$

- Model sampled using periods $h = \{10, 14.5, 17.5\}$ ms
- Controllers based on state feedback from observer, designed using pole placement

Control design, Cont'd

- State feedback poles specified in continuous time as

$$s^2 + 1.4\omega_c s + \omega_c^2 = 0$$

$$\omega_c = \{53, 38, 31\} \text{ rad/s}$$

- Observer poles specified in continuous time as

$$s^2 + 1.4\omega_o s + \omega_o^2 = 0$$

$$\omega_o = \{106, 75, 61\} \text{ rad/s}$$

Implementation

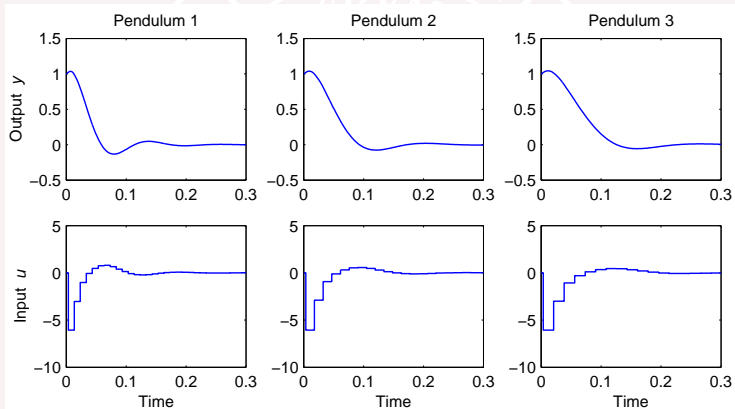
- A periodic timer interrupt samples the plant output and triggers control task
- Each controller i is implemented as a task:

```
y := ReadSample(i);  
u := CalculateControl(y);  
AnalogOut(i,u);
```

- Assumed execution time: $C = 3.5$ ms

Simulation 1 – Ideal case

Each controller runs on a separate CPU.



Schedulability analysis

- Assume $D_i = T_i$
- CPU utilization $U = \sum_{i=1}^3 \frac{C_i}{T_i} = 0.79$
- Schedulable under EDF, since $U < 1$
- Schedulable under RM?

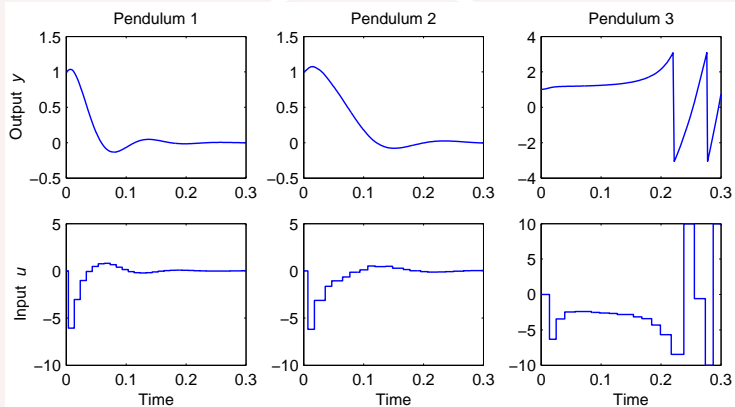
$$U > 3(2^{1/3} - 1) = 0.78 \Rightarrow \text{Cannot say}$$

Compute worst-case response times R_i :

Task	T	D	C	R
1	10	10	3.5	3.5
2	14.5	14.5	3.5	7.0
3	17.5	17.5	3.5	14.0

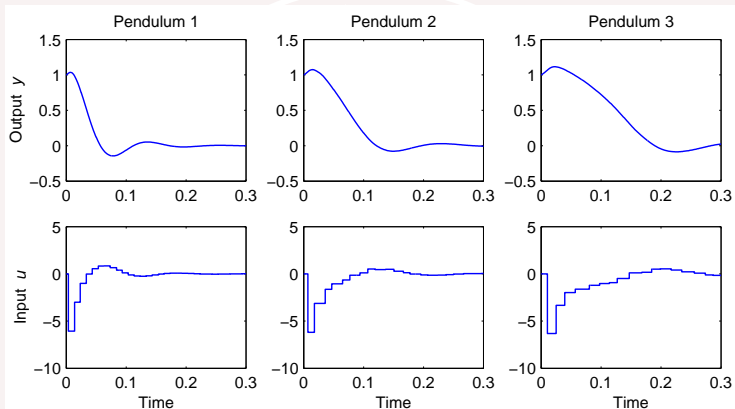
$$\forall i : R_i < D_i \Rightarrow \text{Yes}$$

Simulation 2 – Rate-monotonic scheduling



- Loop 3 becomes unstable

Simulation 3 – Earliest-deadline-first scheduling



- All loops are OK

Conclusion

- Schedulability does not imply stability
- Stability does not require schedulability
- The relation between scheduling parameters and the control performance is complex and can be studied through analysis or simulation