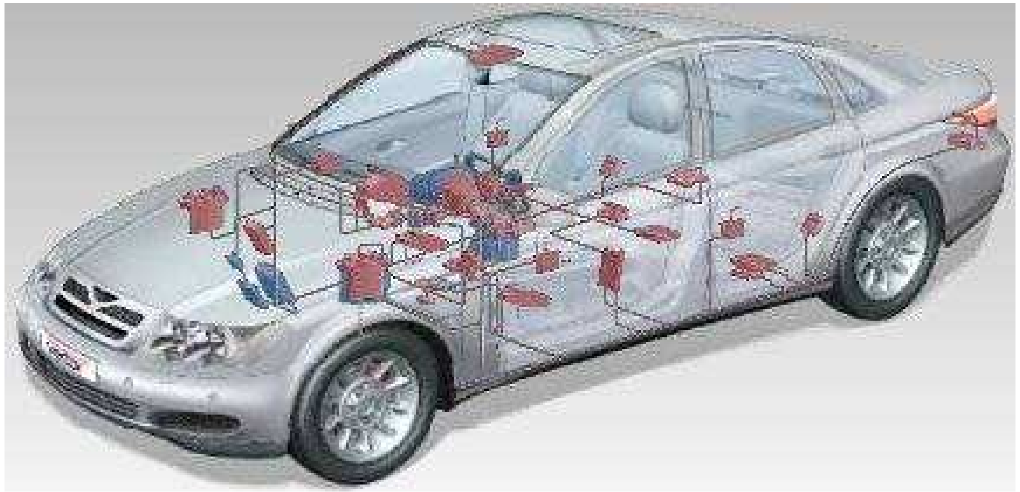
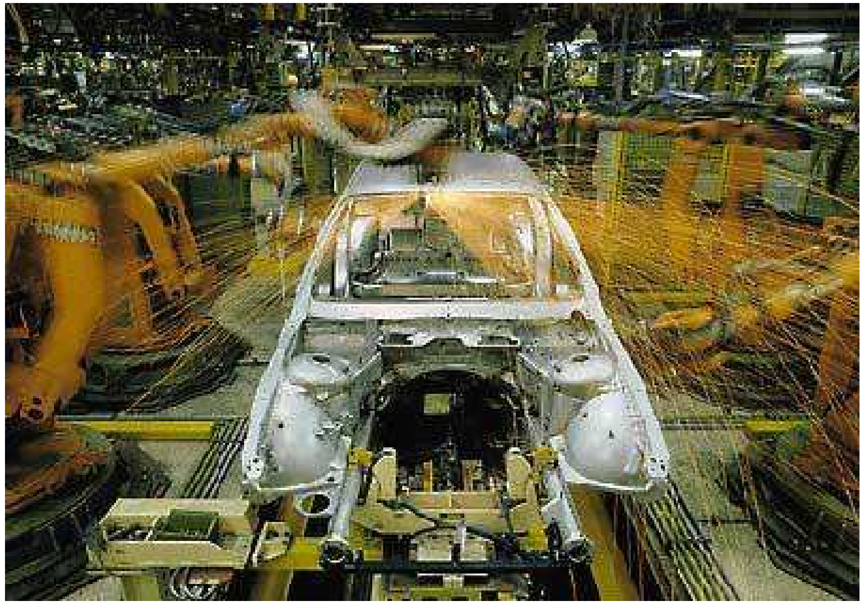
Networked and Embedded Control Systems

Josep M. Fuertes, Ricard Villà, Jordi Ayza,
Pau Martí, Manel Velasco,
José Yépez, Camilo Lozoya, Josep Guàrdia, Frederic Pérez

Automatic Control Dept., Technical University of Catalonia
pau.marti@upc.edu

Application scenarios

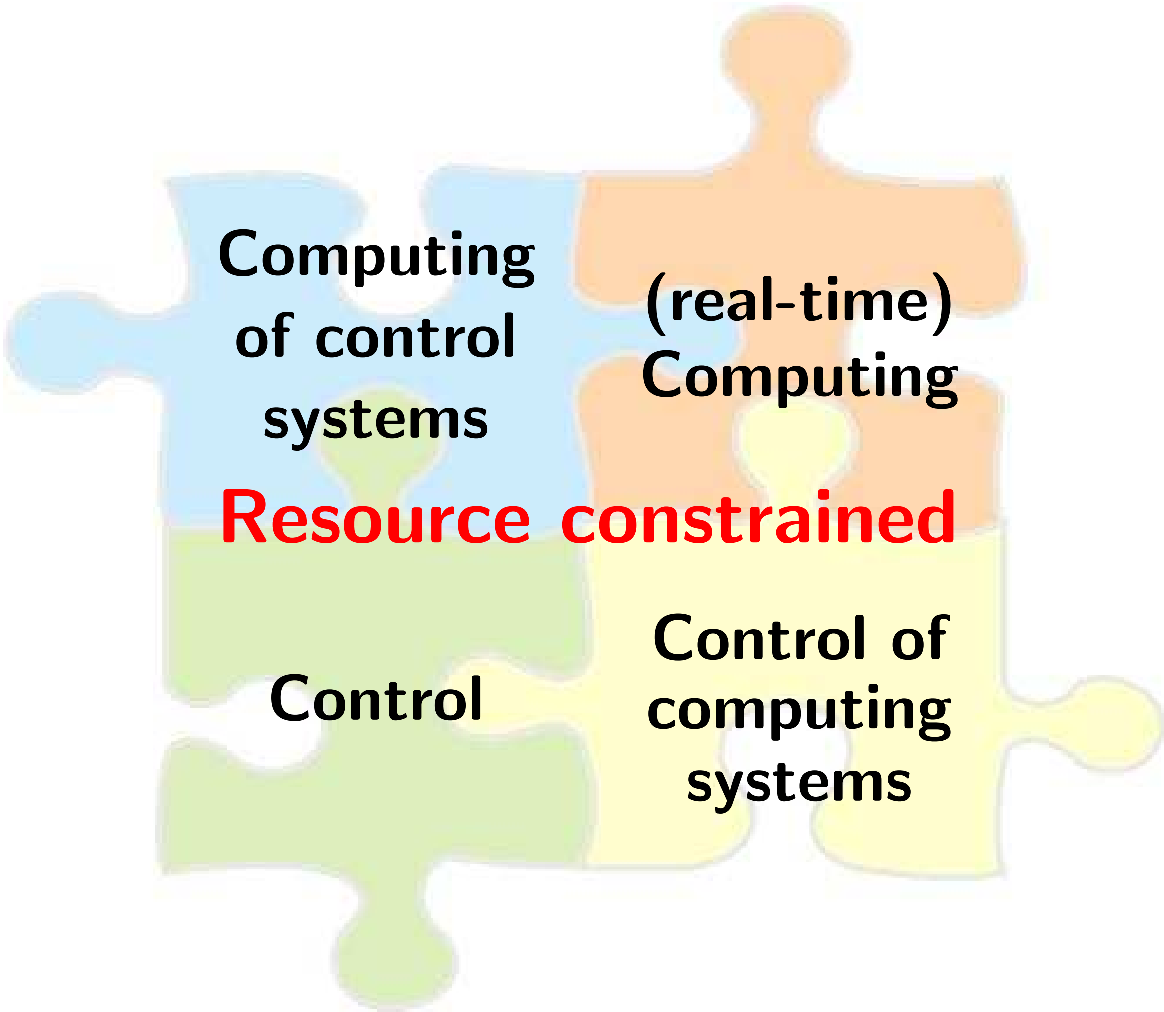




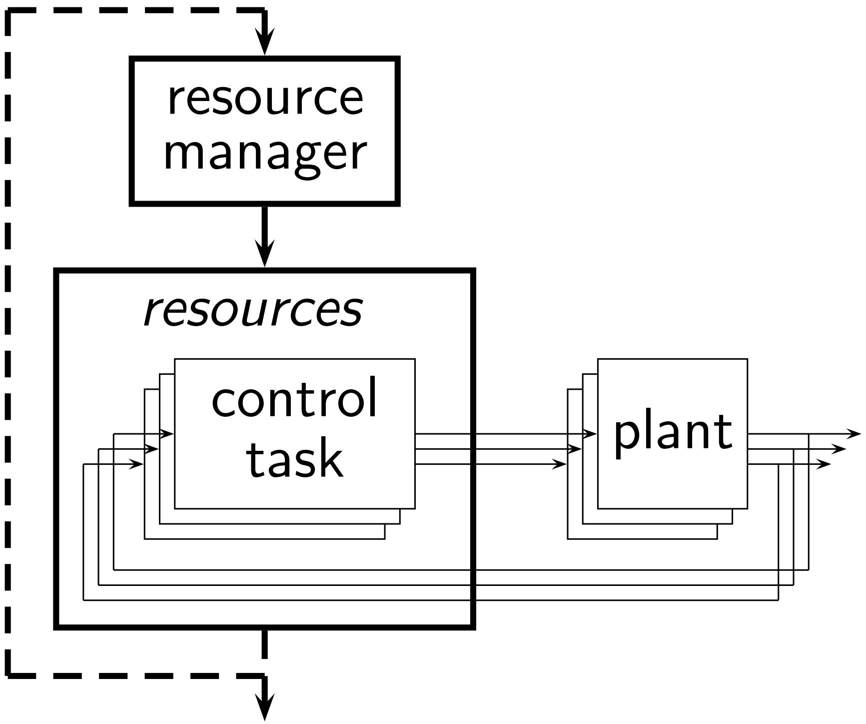
Common features

- Resource-constrained systems (mass-marked products subject to hard economic constraints)
- Often used in unpredictable environments
- The time when results are delivered is important
- Many simultaneously running control applications

Puzzle


Computing of control systems
(real-time) Computing
Resource constrained
Control
Control of computing systems

Understanding the puzzle

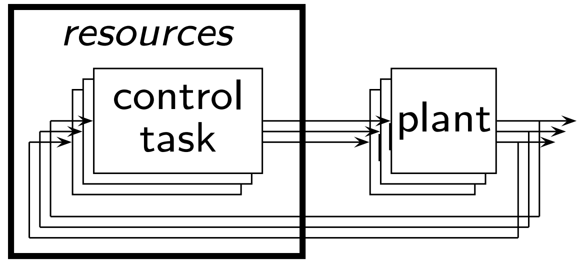


Building the puzzle

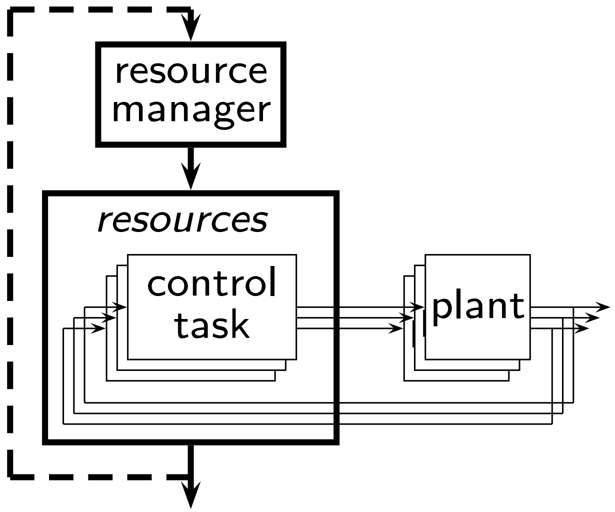
- Reality: gap between communities (real-time, control, ...)
- Why? Misconceptions
  - Real-time engineers assume hard deadlines for control algorithms
  - Control engineers assume determinism in the computing platform
- Need: Closer interaction between communities
- Today, emerging areas closing the gap
  - computing of control systems
  - control of computing systems

Contents

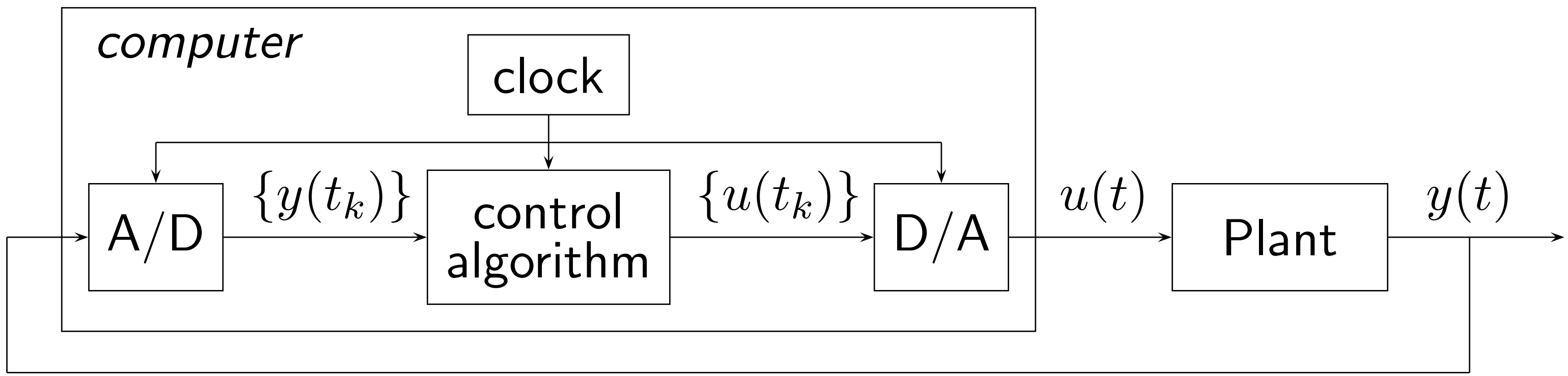
1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions



2. Control of real-time control systems
   (a) Overview
   (b) Representative examples

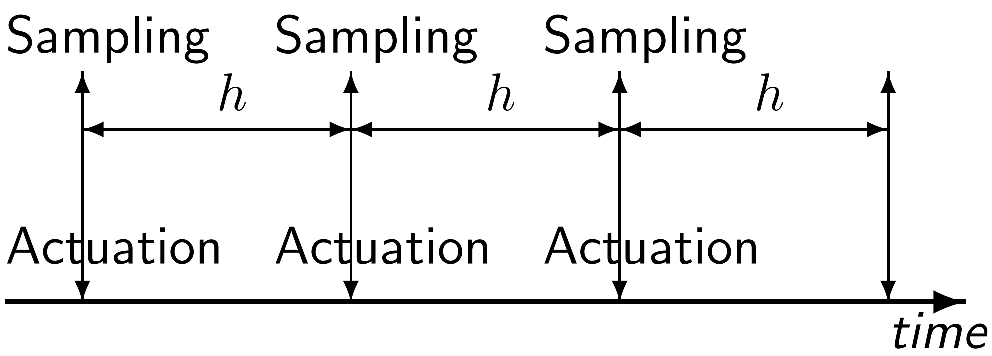


Timing and implementation



Simplest mathematical model with

- constant sampling period
- instantaneous input-output latency



Timing and implementation

Linear time-invariant continuous-time system state-space form

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

Discrete form, with sampling period h [1]

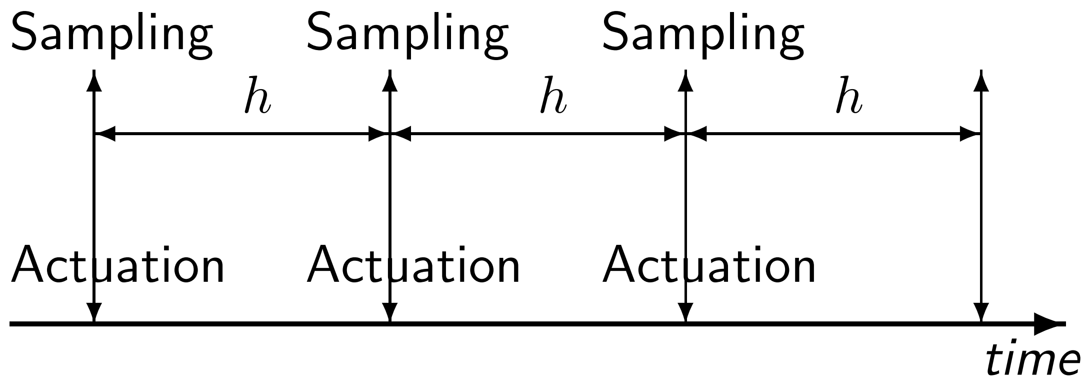
$$\begin{aligned} x_{k+1} &= \Phi(h)x_k + \Gamma(h)u_k \\ y_k &= Cx_k, \end{aligned} \tag{2}$$

where $\Phi(t)$ and $\Gamma(t)$ are obtained using the following

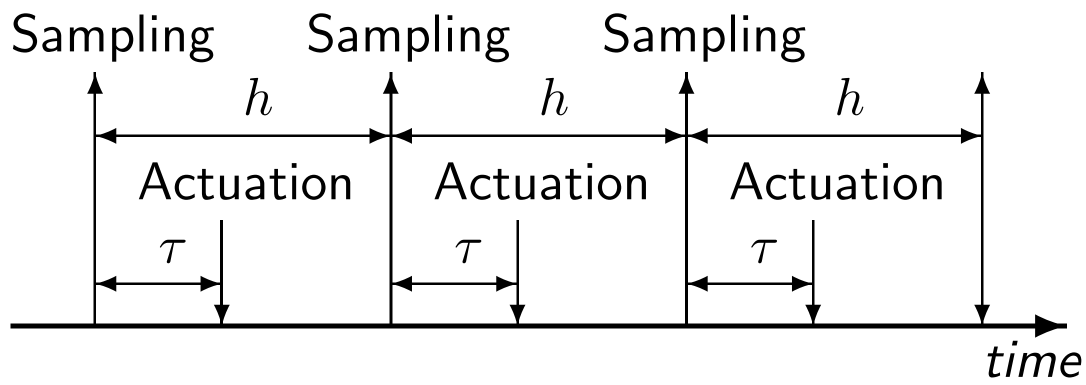
$$\Phi(t) = e^{At}, \quad \Gamma(t) = \int_0^t e^{As} B ds, \tag{3}$$

Timing and implementation

Timing of the basic model is not realistic



Adding a time delay to model an input/output latency due to the computation of the control algorithm or the insertion of a network



Timing and implementation

Continuous-time system with time delay τ

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t-\tau) \\ y(t) &= Cx(t) \end{aligned} \tag{4}$$

Discrete form, with $\tau \leq h$

$$\begin{aligned} x_{k+1} &= \Phi(h)x_k + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k. \\ y_k &= Cx_k, \end{aligned} \tag{5}$$

where $\Phi(t)$ and $\Gamma(t)$ are also obtained using (3).

Timing and implementation

State-space form for (5), extended model:

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} + \begin{bmatrix} \Gamma(h-\tau) \\ I \end{bmatrix} u_k \qquad (6)$$

where $z_k \in \mathbb{R}^{m\times 1}$ represent past control signals.

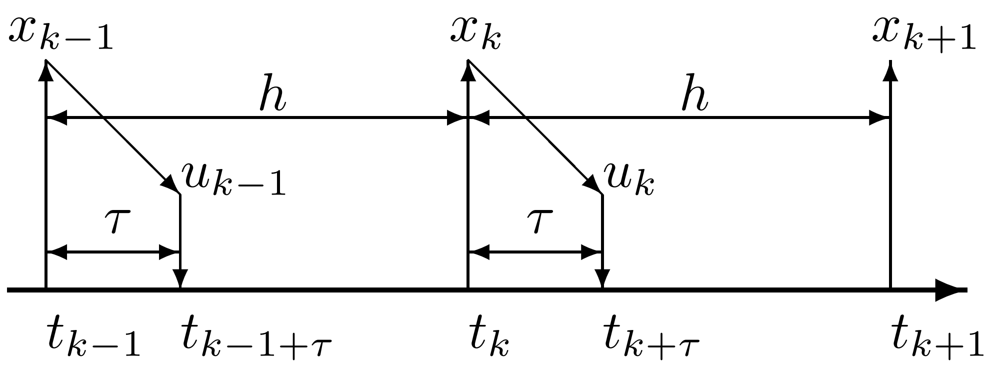
This notation slightly differs from conventional notation [1] to stress dependencies on h and τ.

The notation may be still misleading: u_k is applied τ time units after x_k is taken.

Timing and implementation

Notation issues:

k^{th} operation vs. timing



Let's obtain system (5) by looking at the dynamics from x_k to x_{k+1}. Denote the system state at time $t_{k+\tau}$ as $x_{k+\tau}$. Then

$$
\begin{aligned}
\text{From } x_k \text{ to } x_{k+\tau} \rightarrow x_{k+\tau} &= \Phi(\tau)x_k + \Gamma(\tau)u_{k-1} \\
\text{From } x_{k+\tau} \text{ to } x_{k+1} \rightarrow x_{k+1} &= \Phi(h-\tau)x_{k+\tau} + \Gamma(h-\tau)u_k \\
\text{All together} \rightarrow x_{k+1} &= \Phi(h-\tau)\left(\Phi(\tau)x_k + \Gamma(\tau)u_{k-1}\right) + \Gamma(h-\tau)u_k \\
&= \Phi(h-\tau)\Phi(\tau)x_k + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k \\
&= \Phi(h)x_k + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k \\
&= \text{model (5)}
\end{aligned}
$$

Timing and implementation

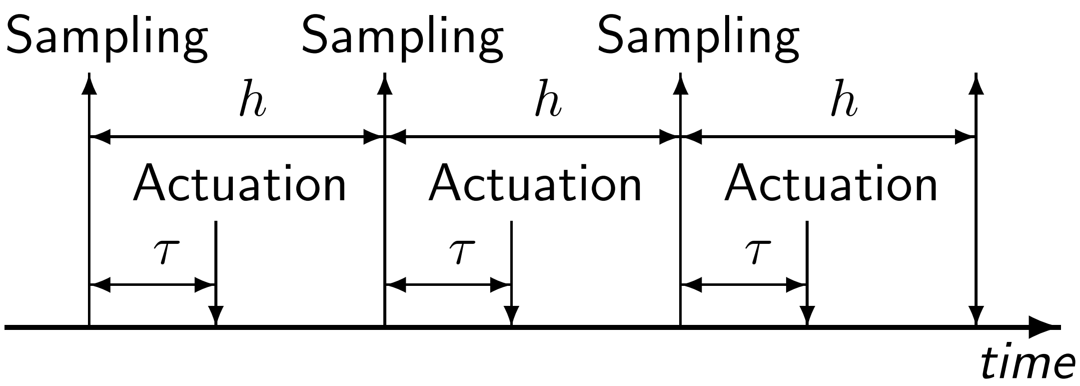
For closed loop operation of (6), given

$$u_k = -K(h,\tau)\begin{bmatrix} x_k \\ z_k \end{bmatrix} \tag{7}$$

where $K(h,\tau)$ is the state feedback gain, the system evolution is

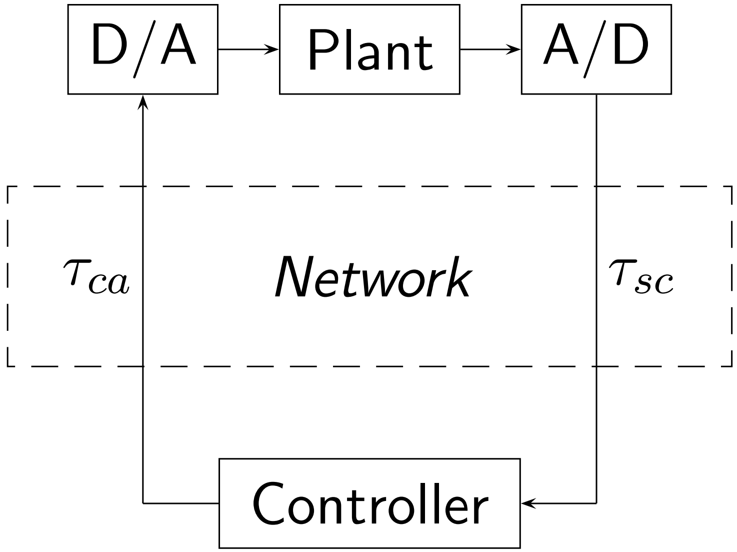
$$\begin{aligned}\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} &= \left(\begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \Gamma(h-\tau) \\ I \end{bmatrix} k(h,\tau)\right)\begin{bmatrix} x_k \\ z_k \end{bmatrix} = \\ &= \Phi_{cl}(h,\tau)\begin{bmatrix} x_k \\ z_k \end{bmatrix}\end{aligned} \tag{8}$$

Note: K depends on “future” parameters, h and τ



Timing and implementation

The extended form (6) also can model networked control systems



Delays controller-to-actuator τ_{ca} and sensor-to-controller τ_{sc} can be integrated into τ in (6).

Timing and implementation

Example. Double integrator differential equation:

$$\frac{d^2y}{dt^2} = u \tag{9}$$

If y and $\dot{y}$ are x_1 and x_2, a state space form is given by

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned} \tag{10}$$

Discrete-time model, with period h and delay τ

$$x_{k+1} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \tau\left(h - \frac{\tau}{2}\right) \\ \tau \end{bmatrix} u_{k-1} + \begin{bmatrix} \frac{(h-\tau)^2}{2} \\ h - \tau \end{bmatrix} u_k \tag{11}$$

Timing and implementation

If $h = 0.1$s and $\tau = 0.01$s, the state space form is given by

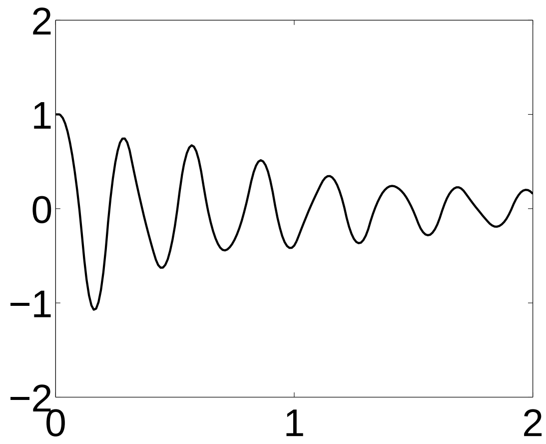
$$x_{k+1} = \begin{bmatrix} 1 & 0.1 & 0.001 \\ 0 & 1 & 0.01 \\ 0 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} 0.004 \\ 0.09 \\ 1 \end{bmatrix} u_k \qquad (12)$$

Closing the loop with $u_k = -\begin{bmatrix} 271.7 & 21.86 & 0.23 \end{bmatrix} x_k$, the closed loop poles are $\lambda_1 = -0.3,\ \lambda_2 = -0.1,\ \lambda_3 = -0.9$

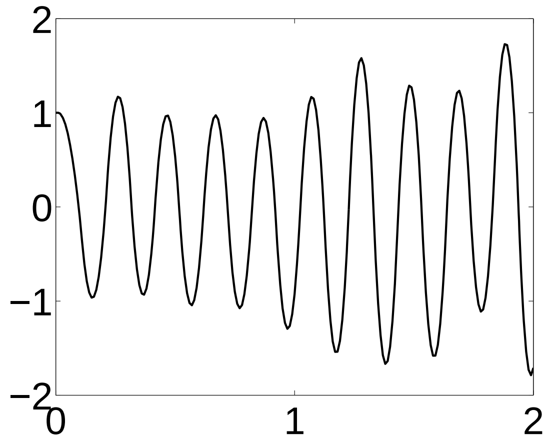
- With a faster micro $\tau = 0.005$, poles go at $\lambda_1 = 0.0082 - 0.2850i,\ \lambda_2 = 0.0082 + 0.2850i,\ \lambda_3 = -1.5513$
- With a slower micro $\tau = 0.02,\ \lambda_1 = -0.4910 + 0.9516i,\ \lambda_2 = -0.4910 - 0.9516i,\ \lambda_3 = 0.1316,$ with $|\lambda_1| = |\lambda_2| = 1.0708$

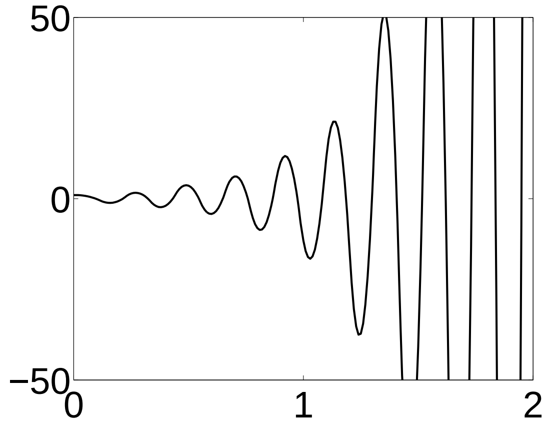
Timing and implementation

Random delay (with $\tau \in \left[\ 0.005 \quad 0.015\ \right]$, where $\tau_d = 0.01$)



Random period (with $h \in \left[\ 0.05 \quad 0.15\ \right]$, where $h_d = 0.1$)





Timing and implementation

- Timing is a key aspect !!!!
- The extended model (5) permits to model timing aspects.
- And the implementation should also enforce the timing.

Timing and implementation

Let's implement a controller: an infinite loop with a periodic activity to be executed every sampling period h

Periodic activity:

read_input(y_k); //assuming that $y_k = x_k$
$u_k = -Lx_k$;
write_output(u_k);

Which is the right code?

Timing and implementation

Let's implement a controller: an infinite loop with an algorithm to be executed every sampling period h. First attempt:

loop
  PeriodicActivity;
  WaitTime(h);
end loop

The computation time of *PeriodicActivity* is not accounted for.

Timing and implementation

Let's implement a controller: an infinite loop with an algorithm to be executed every sampling period h. Second attempt:

loop
Start = CurrentTime();
PeriodicActivity;
Stop = CurrentTime();
C := Stop - Start;
WaitTime(h - C);
end loop

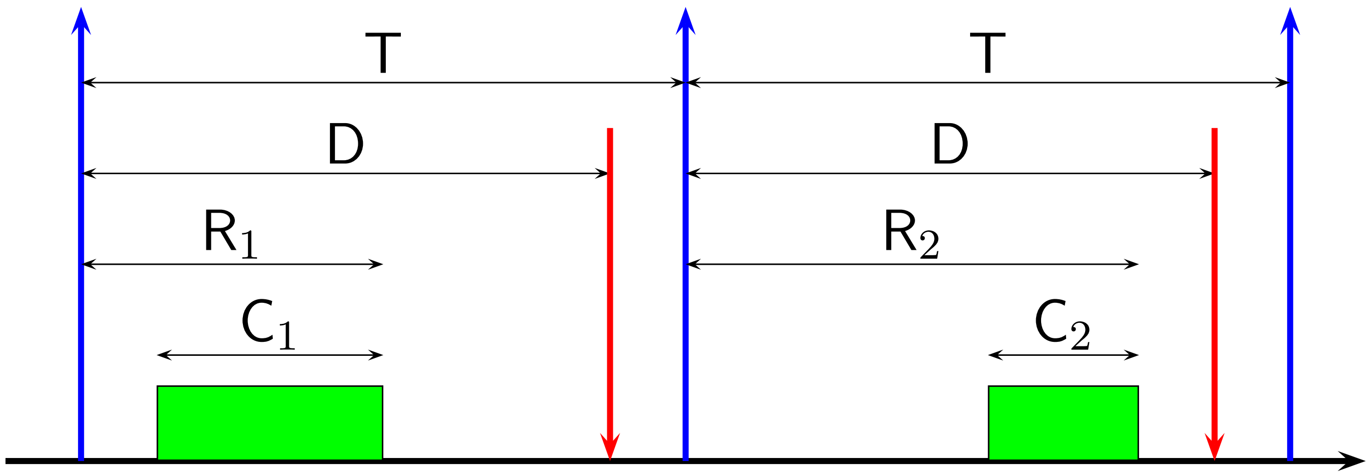
An interrupt causing suspension may occur between the assignment and *WaitTime*; or overrun problem.

Timing and implementation

- Methodologies for guaranteeing timeliness are required.
- Real-time technology ([2] or [3]) is the candidate: *in real-time computing the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced* [4].

Timing and implementation

Task (or message) basic parameters:



Tasks: periodic, sporadic, aperiodic
hard, soft, best effort
time or event triggered
pre-emptive, not pre-emptive

Timing and implementation

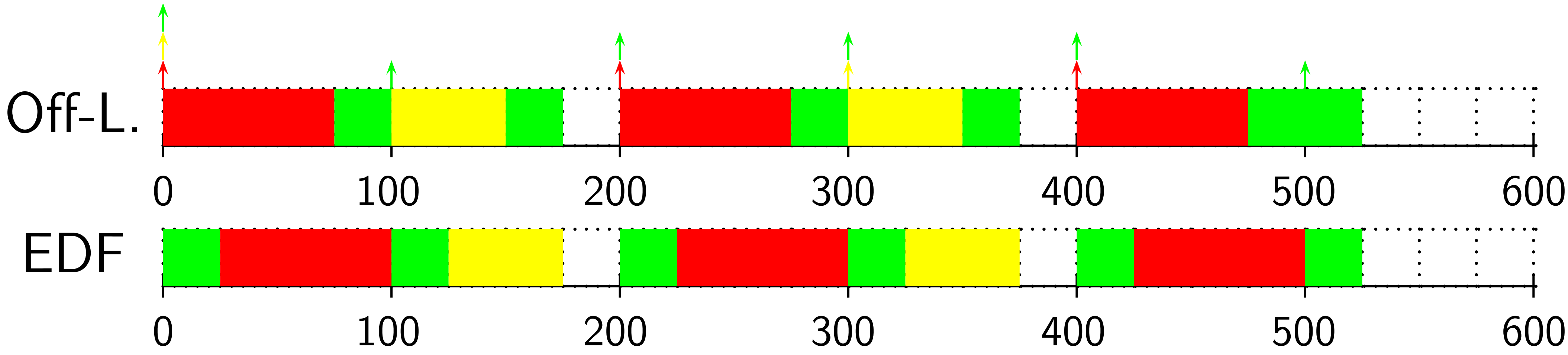
- Scheduling problem: how to assign tasks to the processor/network such that the set of (timing) constraints is met.

- Scheduling approaches:
  - offline scheduling: the time axis is divided in intervals of equal length (time slots), each task is statically allocated in a slot in order to meet the desired request rate, and the execution in each slot is activated by a timer.
  - online scheduling: each task is assigned a priority, scheduling feasibility is verified using analytical techniques, and tasks are executed on a priority-based kernel.

Timing and implementation

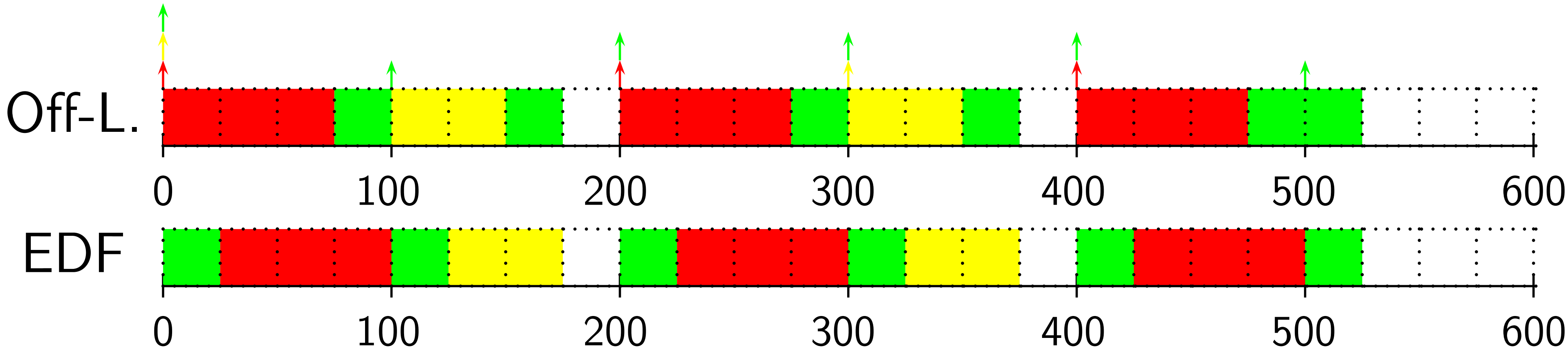
Example $(D = h)$

task	h	C
A	200 ms	75 ms
B	300 ms	50 ms
C	100 ms	25 ms

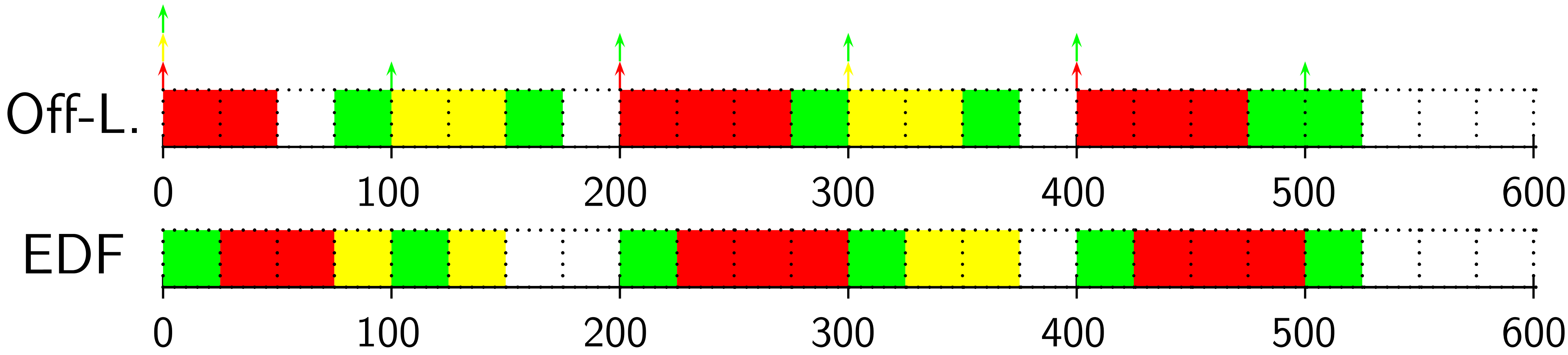


What happens if tasks execute less than C?

Timing and implementation



Run-time schedule with execution time of first job of A is 2



Timing and implementation

Observations on off-line scheduling

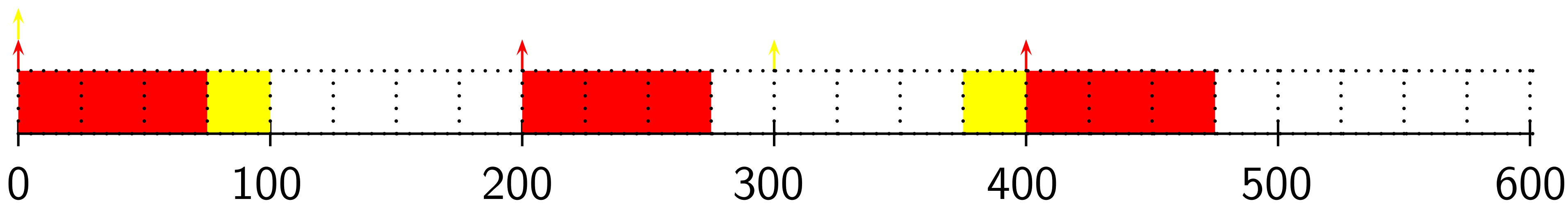
- Disadvantages
  - Keeping strict periodic execution is not possible
  - Hard to build, modify or expand
  - Lacks flexibility to adapt to resource availability or varying application demands
- Advantages
  - Simple implementation enforcing precise timing (no real-time operating system required)
  - Low run-time overhead

Timing and implementation

Example of what strict timing may mean....

Game: Let's try drawing a time line to execute tasks, keeping a constant distance between consecutive job start-times

task	h	C
A	200 ms	75 ms
B	300 ms	25 ms



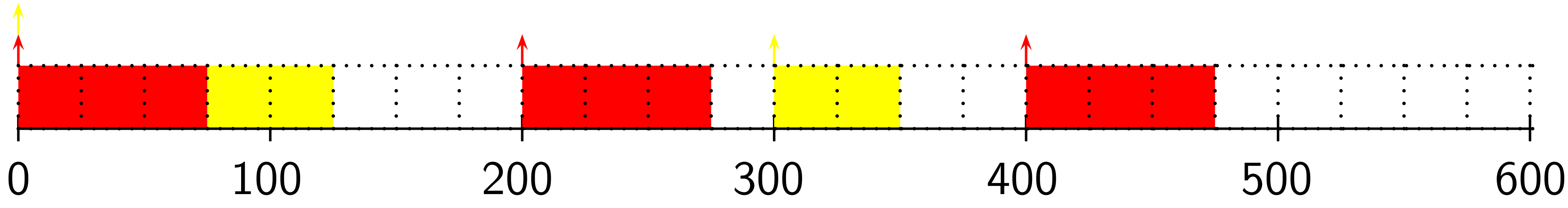
Assume **A** takes longer, e.g., C$= 100$ ms.

Can we still keep the constant distance?

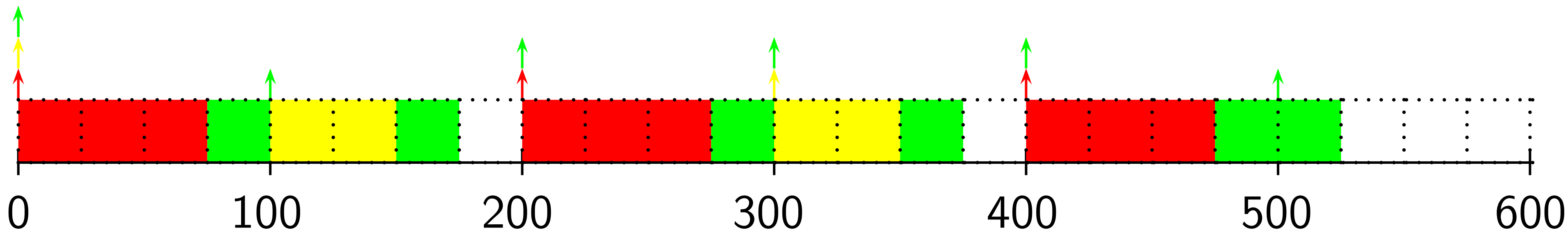
Timing and implementation

Example of *hard to modify*: Let's draw a time-line to execute

task	h	C
A	200 ms	75 ms
B	300 ms	50 ms

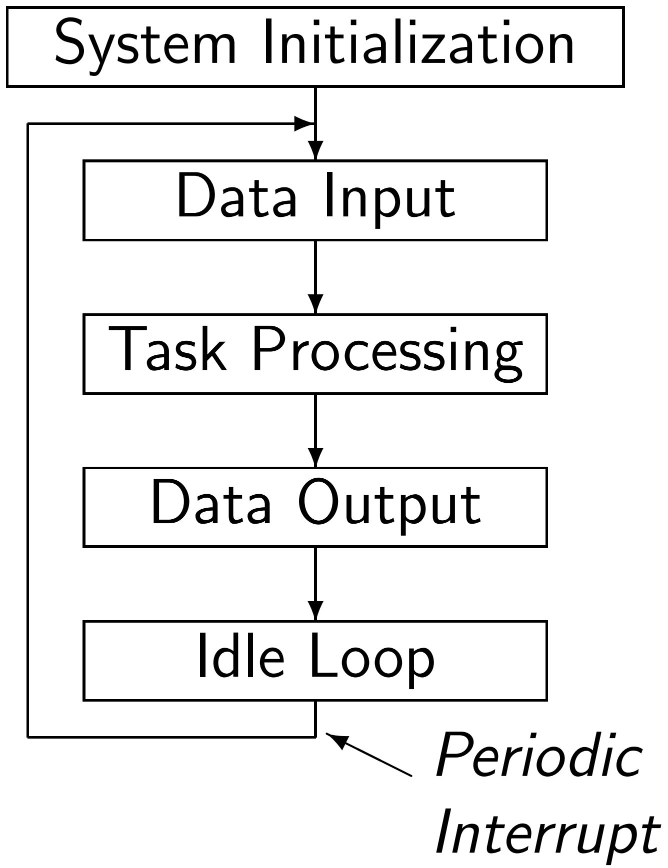


Assume a new task **C** with $h = 100$ ms and C$= 25$ ms. We have to rebuild the whole schedule !!!!



Timing and implementation

Note on offline scheduling: cyclic executives are the traditional offline scheduling approach for control applications [6], and widely used in industry.



Timing and implementation

Observations about on-line scheduling

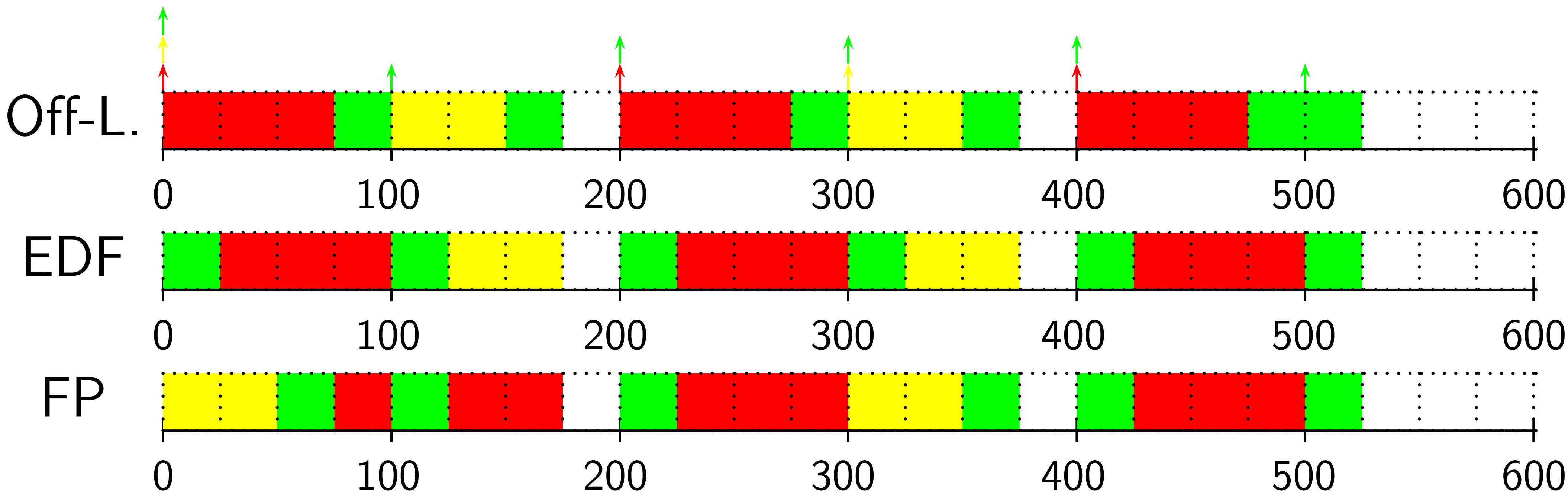
- Disadvantages
  - Keeping **strict** periodic execution is not possible
  - Needs operating system support (**overhead**)

- Advantages
  - Easy to build, analyze, modify or expand
  - Provides **flexibility** to adapt to resource availability or varying application demands

Timing and implementation

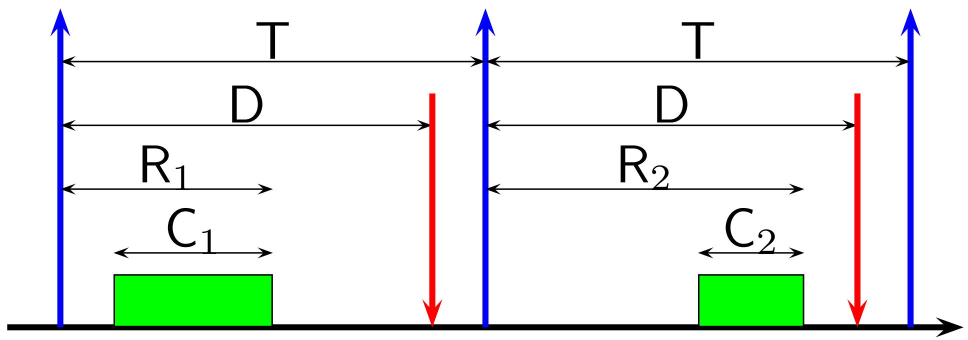
Previous example with a new schedule given by Fixed Priority online scheduling. Which is the priority assignment?

task	h	C
A	200 ms	75 ms
B	300 ms	50 ms
C	100 ms	25 ms

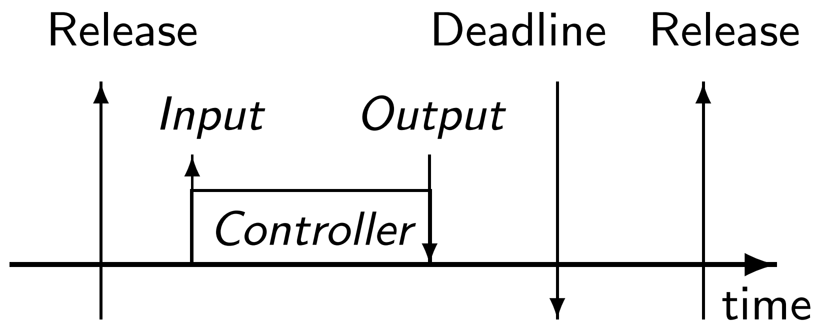


Timing and implementation

Enforcing the timing assumed in mathematical models using real-time technology



Hard real-time periodic task model



Naif control task model

- task period equal to sampling period T$=h$
- sampling and actuation occur at the beginning and termination of each job execution
- and task deadline bounding the time delay, D$\geq \tau$

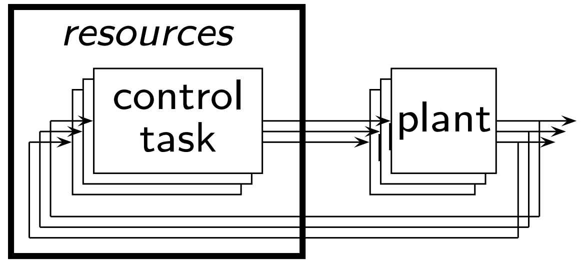
Timing and implementation

In the previous model for implementing controllers,

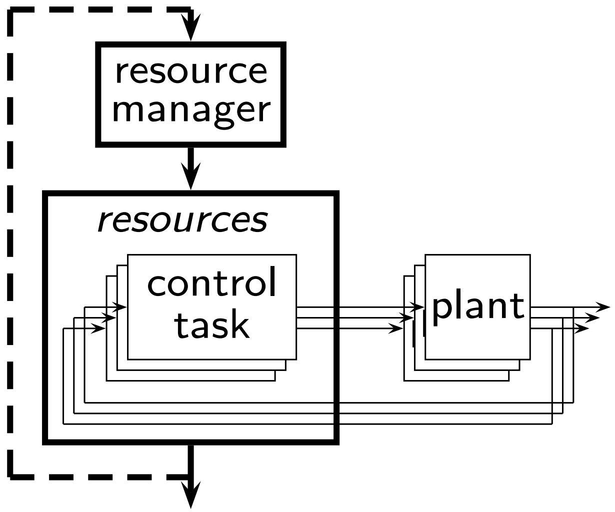
- if D$= \tau$: the expected timing from the model in closed loop operation (8) is perfectly kept !!! However, task set schedulability is severely reduced !!!
- if D$> \tau$: task set schedulability is increased (more tasks can be executed) at the expenses of introducing time uncertainty in sampling and actuation operations.

Contents

1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions

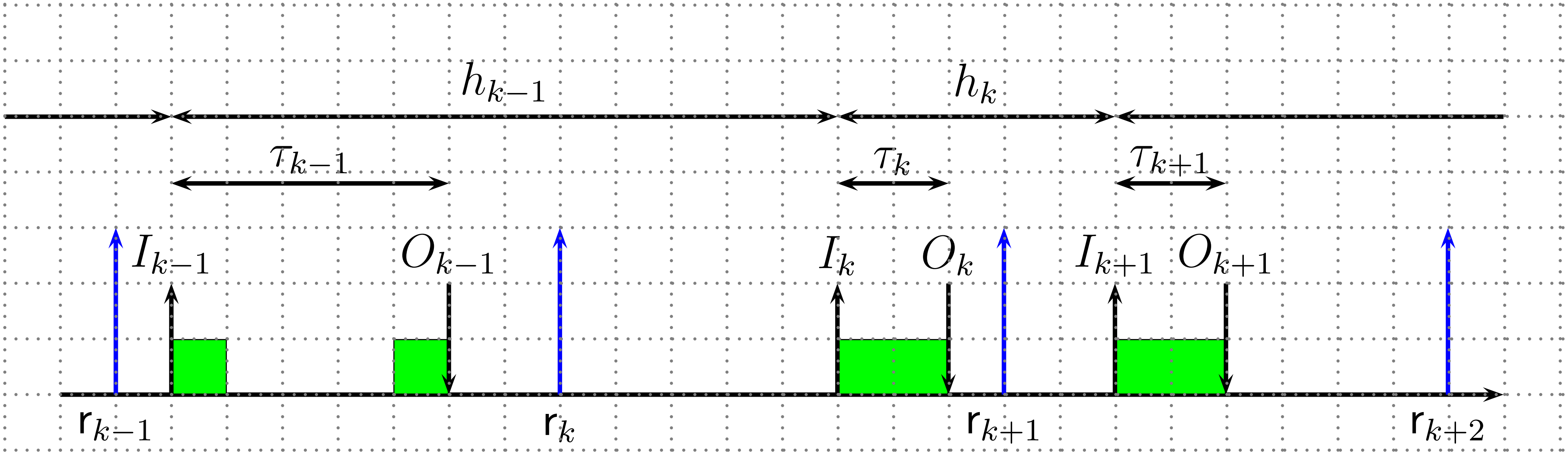


2. Control of real-time control systems
   (a) Overview
   (b) Representative examples



Problems and solutions

(RT) Computing introduces time uncertainty in periods and delays



- Job released at $r_k = kh$
- Sampling jitter: $\{h_k\}$
- Latency jitter: $\{\tau_k\}$

Problems and solutions

(RT) Computing introduces time uncertainty in periods and delays

Approaches:

- Ignore the problem
- Design the controller to be robust against time uncertainty
- Design the computing to minimize or eliminate the time uncertainty

Problems and solutions

Traditional approaches to time uncertainty

- Smith predictor
- Modified z-transform
- State-space lifting techniques

Limitation: ideal delays or mutirate (periodic) systems.

Alternative solutions

Problems and solutions

Remember the standard model (6)

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} + \begin{bmatrix} \Gamma(h-\tau) \\ I \end{bmatrix} u_k$$

where $x_{k+1} = x(kh+h)$ and $z_{k+1} = z(kh+h)$

If h and τ vary at each job execution, the model is given by

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi(h_k) & \Phi(h_k-\tau_k)\Gamma(\tau_k) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} + \begin{bmatrix} \Gamma(h_k-\tau_k) \\ I \end{bmatrix} u_k \quad (13)$$

where $x_{k+1} = x(\sum_{i=0}^{k+1} h_i)$ and $z_{k+1} = z(\sum_{i=0}^{k+1} h_i)$

Problems and solutions

(13) is a family of models. Observations:

- Given the $\{h_k, \tau_k\}$ values, a specific system is obtained and it can be analyzed
- Controllability and observability
- Stability

Problems and solutions

Given the $\{h_k, \tau_k\}$ values, a specific system is obtained and it can be analyzed. Example.

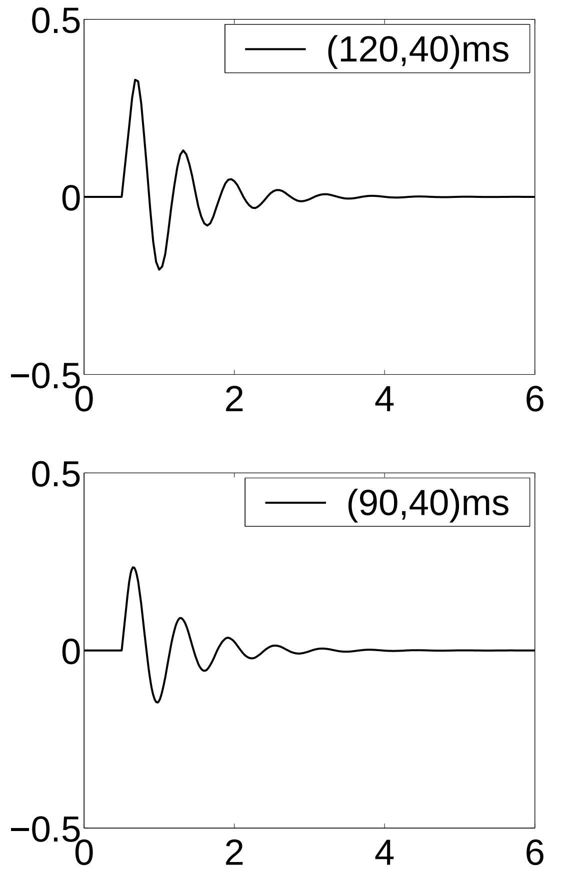
$$\frac{d^2y}{dt^2} = u$$

Let's locate the continuous closed loop poles at $\lambda_{1,2} = -1.5 \pm 10 * i$.

task	h	C=τ
A	120 ms	40 ms
B	90 ms	40 ms

$$K_{120,40} = \begin{bmatrix} 75.8572 & 10.1051 & 0.3435 \end{bmatrix}$$
$$K_{90,40} = \begin{bmatrix} 83.5998 & 9.7351 & 0.3225 \end{bmatrix}$$

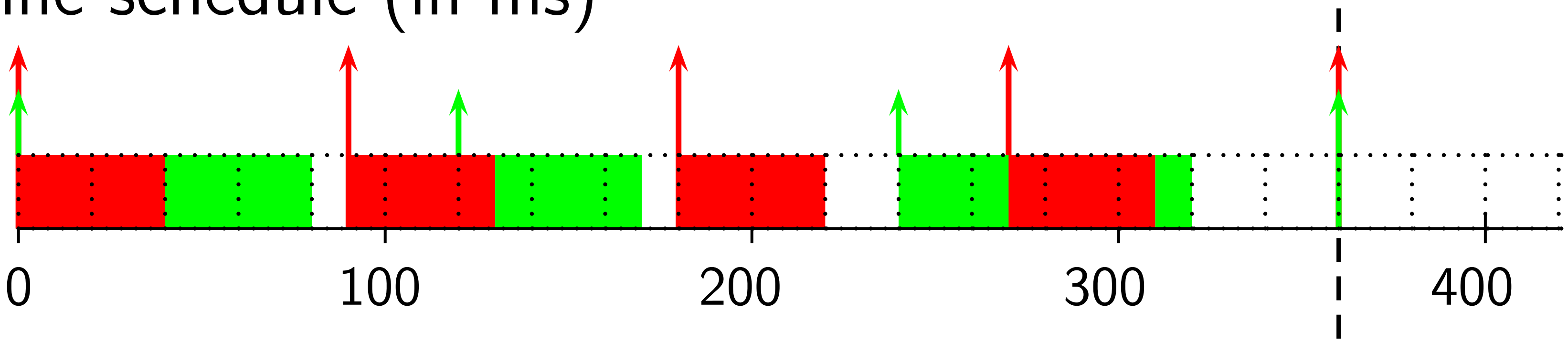
In isolation

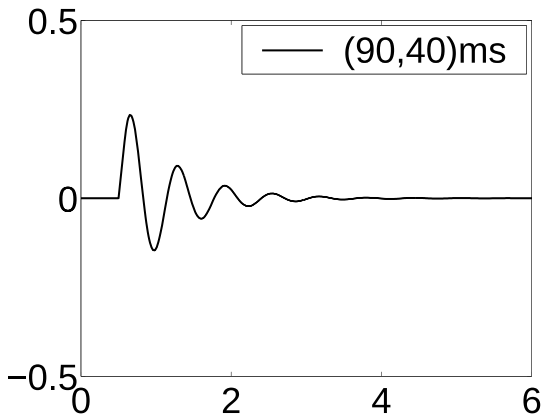


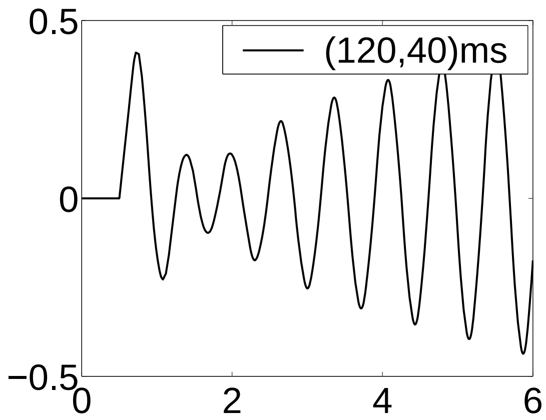
Problems and solutions

Given the $\{h_k, \tau_k\}$ values, a specific system is obtained and it can be analyzed. Example.

Offline schedule (in ms)

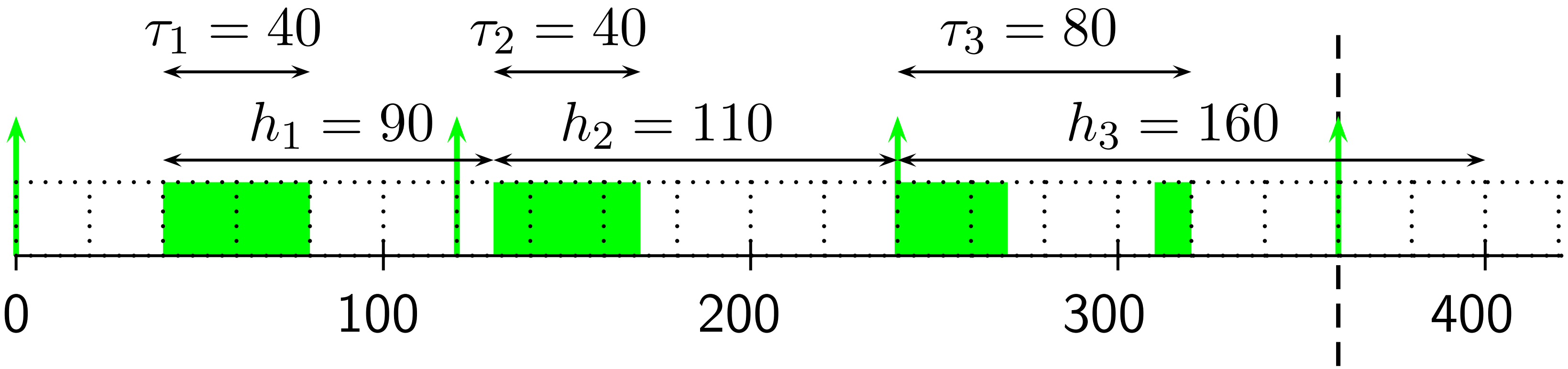






Problems and solutions

Given the $\{h_k, \tau_k\}$ values, a specific system is obtained and it can be analyzed. Example.

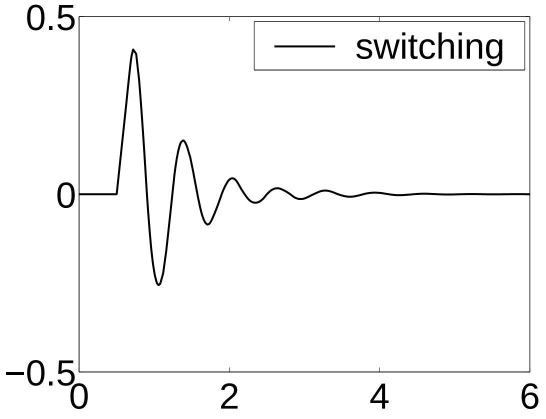


Let's approach A as a switched system:

$$\Phi_{cl}(h,\tau) \in \{\Phi_{cl}(90,40), \Phi_{cl}(110,40), \Phi_{cl}(160,80)\}$$

with

$$K_{90,40} = \left[\begin{array}{ccc} 83.5998 & 9.7351 & 0.3225 \end{array}\right]$$
$$K_{110,40} = \left[\begin{array}{ccc} 78.4894 & 10.0117 & 0.3377 \end{array}\right]$$
$$K_{160,80} = \left[\begin{array}{ccc} 65.0282 & 12.7871 & 0.8149 \end{array}\right]$$



Problems and solutions

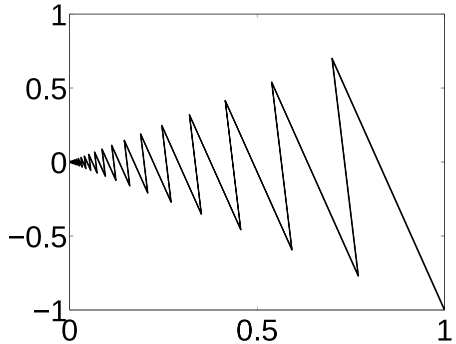
Given the $\{h_k, \tau_k\}$ values, a specific system is obtained and it can be analyzed. Is the previous analysis enough?

Example of unstable switched sequence $A_2A_1A_2A_1\ldots$ where each subsystem A_i is stable. Given

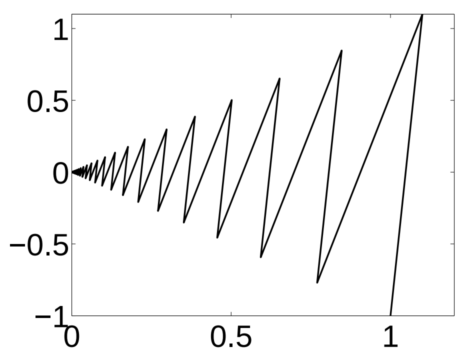
$$x_{k+1} = Ax_k, \quad k \geq 0, \quad x_0 = x0, \quad A \in \{A1, A2\} \tag{14}$$

with $A_1 = \begin{bmatrix} 0.9 & 0.2 \\ -0.2 & -0.9 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0.9 & -0.2 \\ 0.2 & -0.9 \end{bmatrix}, \quad x0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

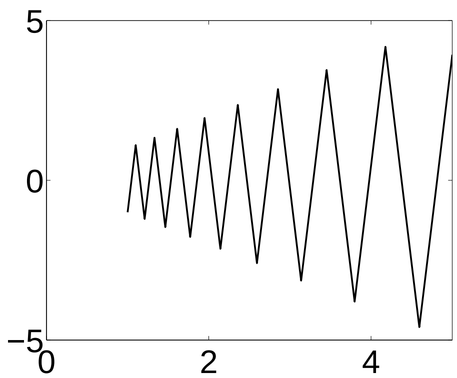
A_1^k



A_2^k



$(A_2A_1)^k$



Problems and solutions

Controllability

- Is (13) controllable? Yes. (See proof in appendix 1)
- Therefore, we can find $\{u_k\}$ to bring the system to x_{eq}
- However, to compute $\{u_k\}$, we need to know beforehand $\{h_k, \tau_k\}$
- ... which in the general case, it's not known !!!

Observation (feasibility problem): to compute u_k we need to know $\{h_k, \tau_k\}$

Problems and solutions

Observability

- Is (13) observable? Yes, if the output matrix outputs the additional variable. (See proof in appendix 2)
- No feasibility problems exist (past h_k, τ_k and u_k are known).

Remark

Although (13) is controllable and observable, the admissible time variability is not known.

Problems and solutions

Stability. Looking at matrices

- Single closed loop matrix

$$\text{stable} \Leftrightarrow \rho(\Phi_{cl}(h,\tau)) < 1 \tag{15}$$

- Sequence of closed-loop matrices

$$\text{stable} \Leftrightarrow \rho(\Phi_{cl}(h_1,\tau_1)\cdot\Phi_{cl}(h_2,\tau_2)\cdot\ldots\cdot\Phi_{cl}(h_n,\tau_n)) < 1 \tag{16}$$

- Closed-loop matrices randomly taken from a finite set Ω [7]

$$\Omega \text{ stable} \Leftrightarrow \exists P > 0 : \Omega^T P \Omega - P < 0,\ \forall \Omega \in \Omega^K, K \geq 0 \tag{17}$$

Problems and solutions

Stability. ... References

- Delays: For time-varying but bounded delays, simply checked in a Bode plot [8]
- Sampling periods: For uncertain sampled data systems, treated as hybrid system and using Lyapunov functions with discontinuities [9]

Problems and solutions

(RT) Computing introduces time uncertainty in periods and delays

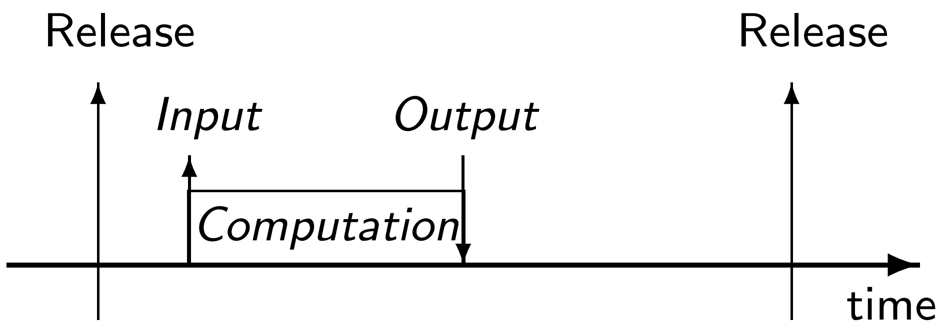
Approaches:

- Ignore the problem
- Design the controller to be robust against time uncertainty
- Design the computing to minimize or eliminate the time uncertainty

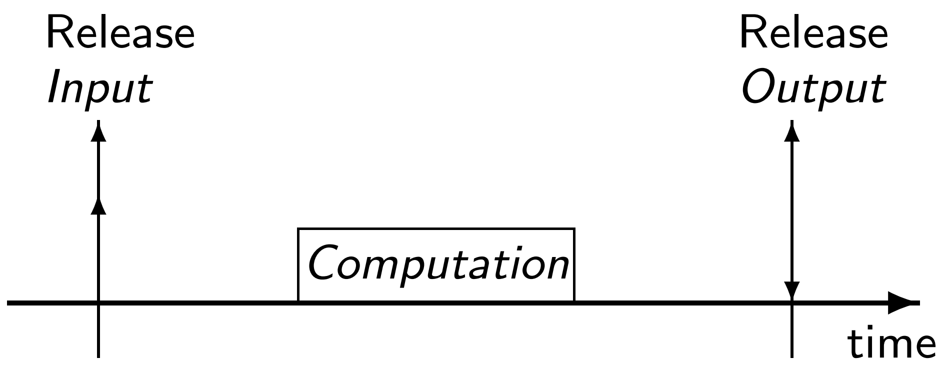
Problems and solutions

Design the computing to minimize/eliminate the time uncertainty

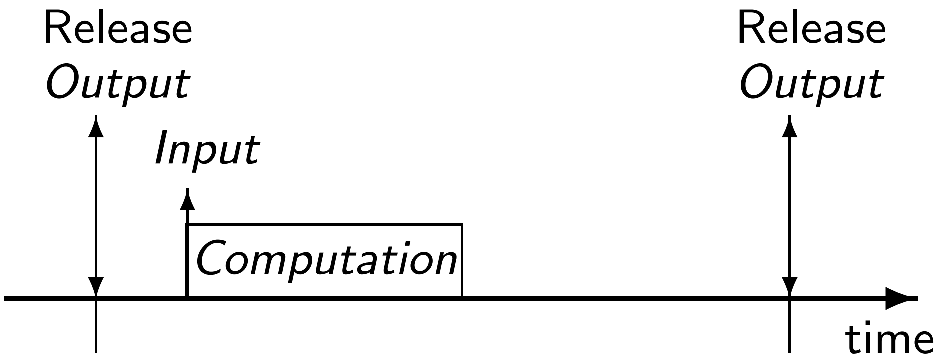
Naif approach [10]



One-sample approach [5], [11]



One-shot approach [12]



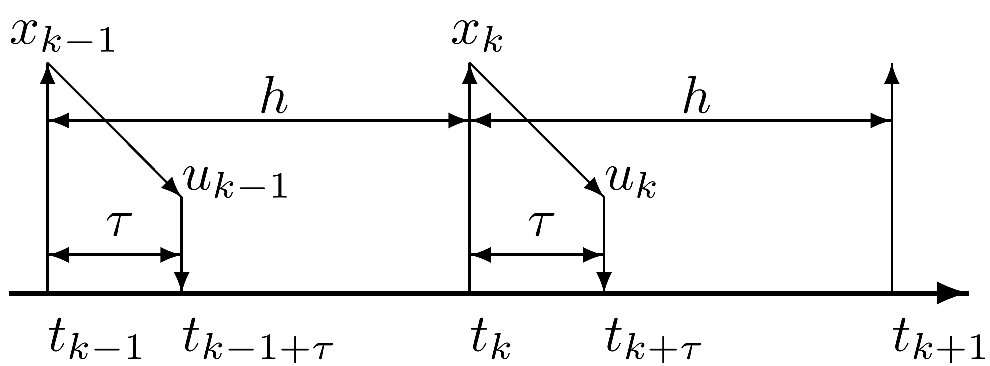
Problems and solutions

Design the computing to minimize/eliminate the time uncertainty

Model (6) in closed-loop form is based on two synchronization points, on a time reference given by the sampling instants.

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} + \begin{bmatrix} \Gamma(h-\tau) \\ I \end{bmatrix} u_k$$

$$u_k = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} = K_1 x_k + K_2 z_k \quad \text{with} \quad K_1 \in \mathbb{R}^{1\times n}, \quad K_2 \in \mathbb{R}^{1\times m}$$



Problems and solutions

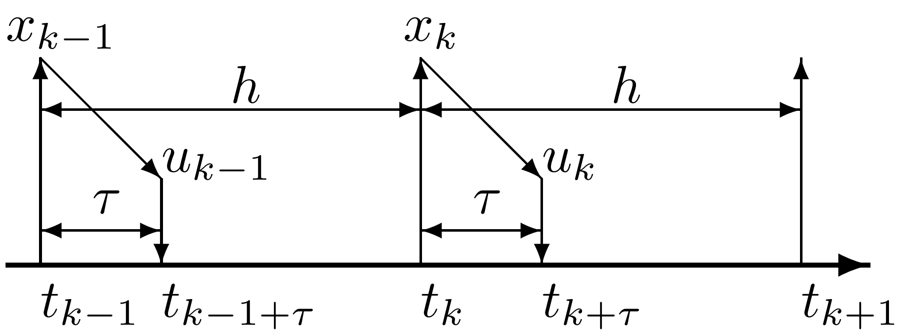
Design the computing to minimize/eliminate the time uncertainty

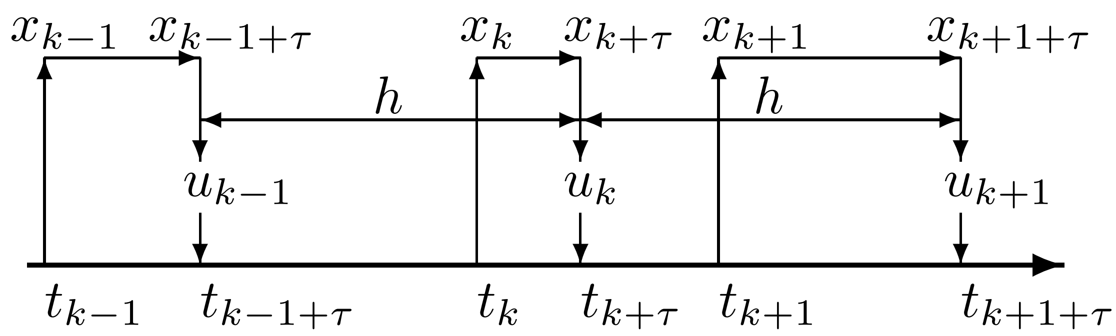
Constructing the one-shot task model: changing time coordinates to a time reference given by the actuation instants.

$$x_{k+\tau+1} = \Phi(h)x_{k+\tau} + \Gamma(h)u_k, \quad \text{with } u_k = Kx_{k+\tau} \quad \text{with } L \in \mathbb{R}^{1\times n}. \tag{18}$$

$x_{k+\tau}$ has to be predicted from x_k:

$$x_{k+\tau} = \Phi(\tau)x_k + \Gamma(\tau)u_{k-1}. \tag{19}$$



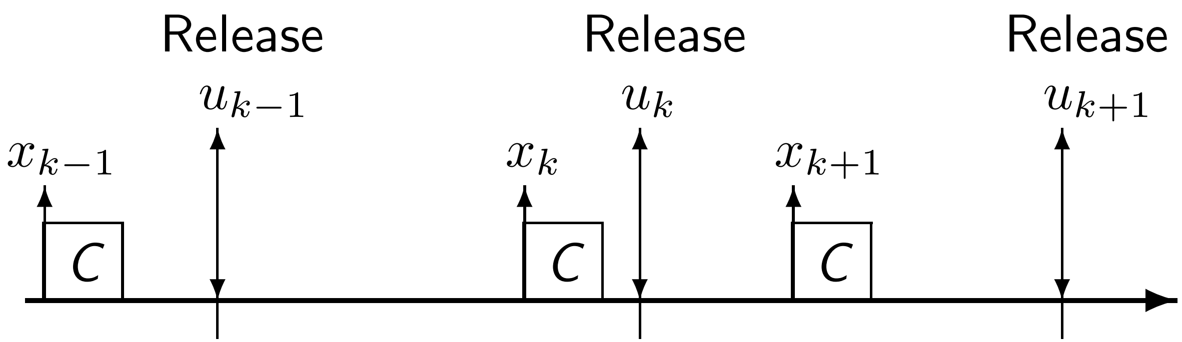


Problems and solutions

Design the computing to minimize/eliminate the time uncertainty

- All closed loop dynamics given by (18) and (19) (i.e., one-shot) can be obtained by (6) (standard).
- All closed loop dynamics given by (6) can be obtained by (18) and (19) if $m = n$.

The standard model is more general.... but one-shot admits irregular sampling

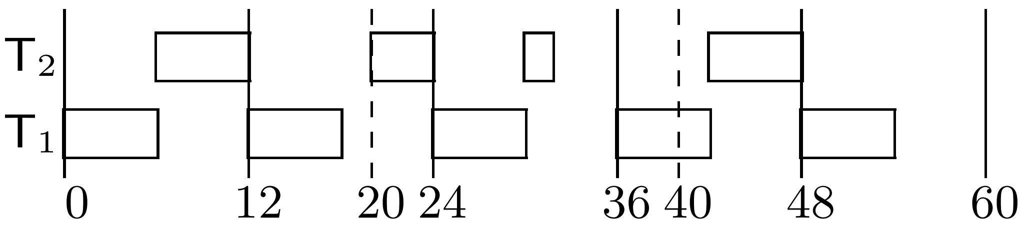


Problems and solutions

Design the computing to minimize/eliminate the time uncertainty

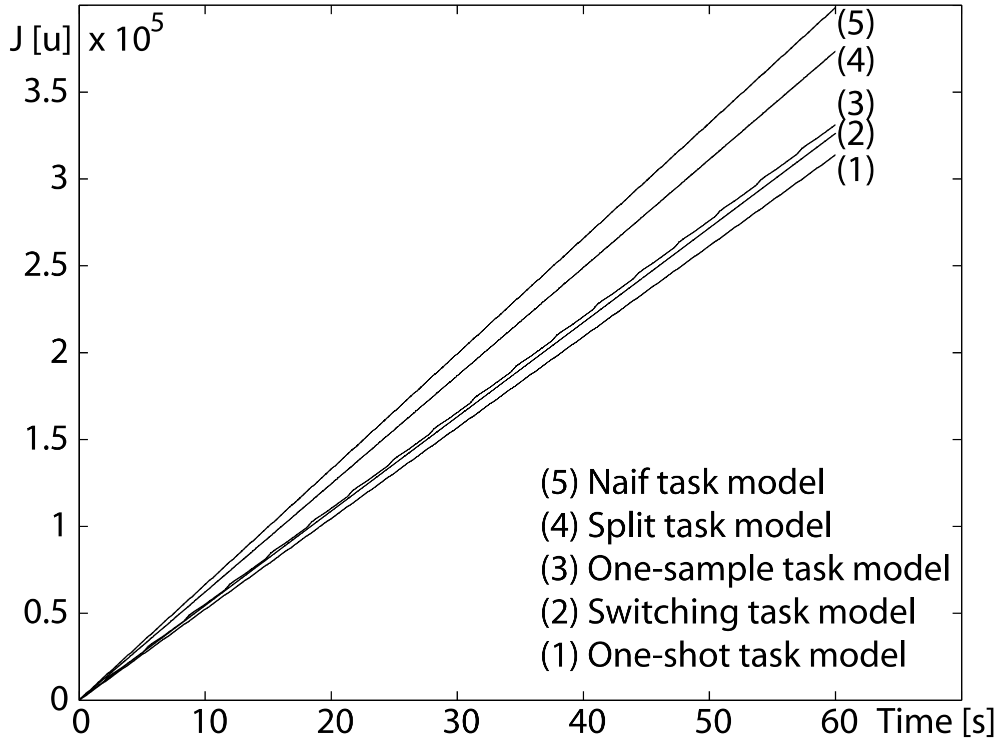
Evaluation: Naif, one-sample, switching, one-shot, split [13]

	h	C
T_1	12	6
T_2	20	6



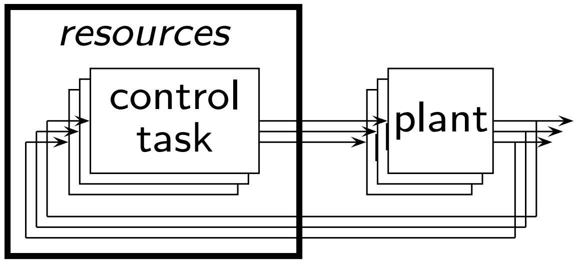
Voltage stabilizer (RCRC circuit)

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -918.2 & -90.9 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 918.2 \end{bmatrix} u(t)$$

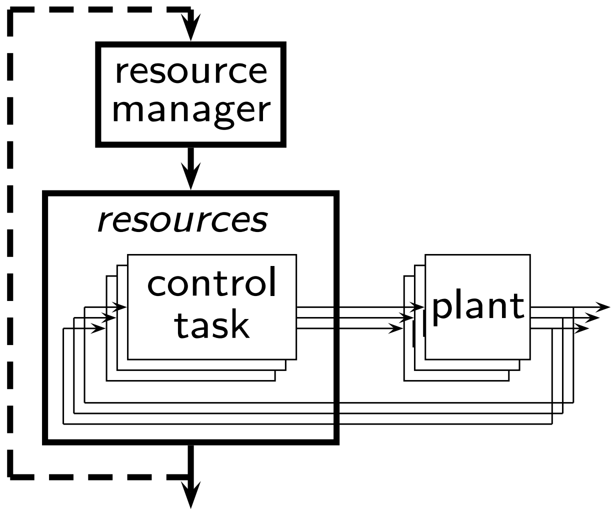


Contents

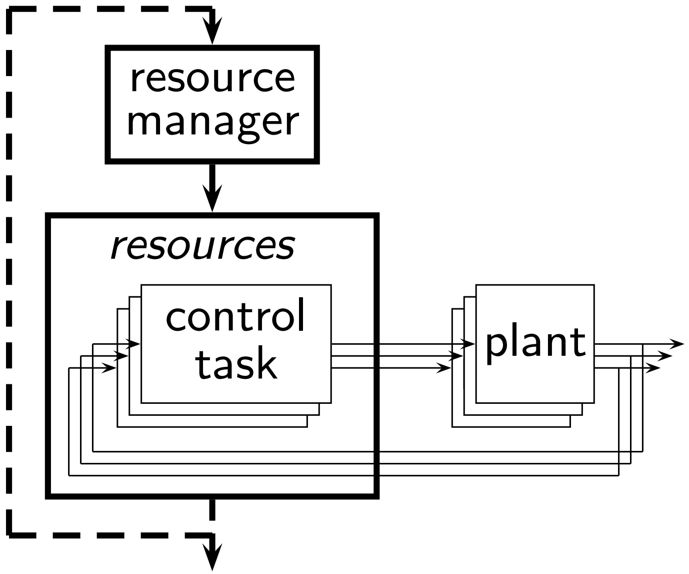
1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions



2. Control of real-time control systems
   (a) Overview
   (b) Representative examples



Overview



Control of real-time control systems, also known as

- (Optimal) Sampling period selection
- Feedback scheduling
- Event-based scheduling

Overview

Objective: to efficiently use resources when control loops share limited resources. Main flavors:

- Maximize aggregated control loop performance by fully and cleverly exploiting the available resources

$$\Downarrow$$

Feedback scheduling (FS)

- Minimize resource utilization while bounding inter-sampling dynamics

$$\Downarrow$$

Event-based scheduling (ES)

Overview * Feedback scheduling

Common formulation: optimization problem

minimize (maximize):	penalty (benefit) on control performance
with respect to:	sampling periods / job execution
subject to:	closed loop stability
	task set schedulability

Two type of results

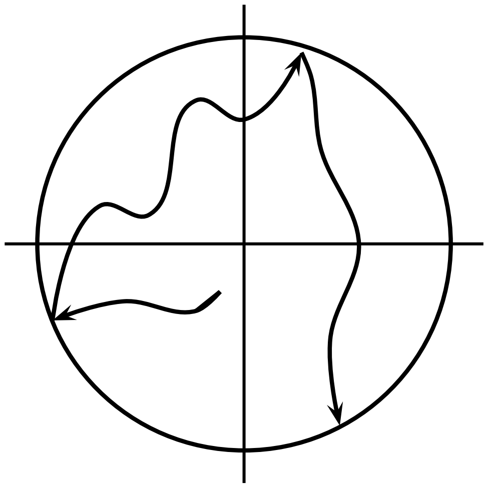
- Optimal sampling periods (e.g., [14], [15],[16],[17],[18])
- Optimal job sequence (e.g., [19], [20][1], [21])

[1] Based on bounding the inter-sampling dynamics.

Overview * Event-based scheduling

Common idea: to bound the inter-sample dynamics or to ensure stability (e.g., [22], [23], [24], [25], [26], [27]). Approaches:

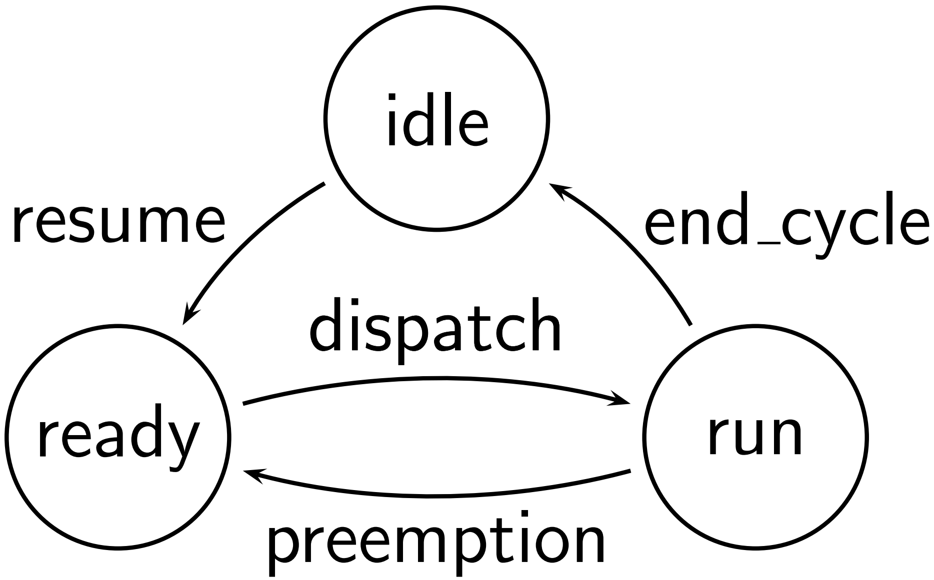
- Integrate an analog event detector, e.g. [22] or [24]
- Assume a coordinator aware of all plant states, e.g., [23]
- Enforce a minimum inter-execution time, e.g., [26]
- Observe the occurrence of the event (self-triggered), e.g., [25], [27]



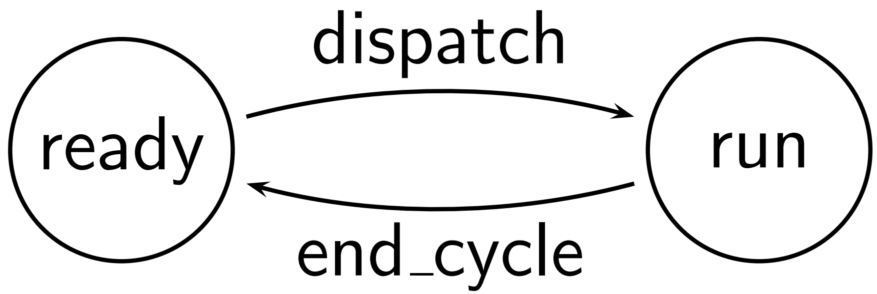
Overview * Taxonomy

	Which	What	Who	When	How		
	Criterion			it is solved	Solution	Timing Constraints	Sched.
[14] Set96	Optimizat.	TT	Coord.	Offline	Periods	Static periodic	EDF
[22] Arz99	Bound d.	ET	Task	Online	Periods	Aperiodic ET	Missing
[23] Zha99	Bound d.	ET	Coord.	Online	Job	Aperiodic ET	EFS
[15] Eke00	Optimizat.	TT	Coord.	Online	Periods	Varying periodic	EDF
[19] Reh00	Optimizat.	TT	Coord.	Offline	Sequences	Static pseudo periodic	Cyc. Ex.
[20] Hri01	Bound d.	TT	Coord.	Offline	Sequences	Static pseudo periodic	Cyc. Ex.
[16] Mar04	Optimizat.	TT	Coord.	Online	Periods	Varying periodic	EDF
[17] Pal05	Optimizat.	TT	Coord.	Offline	Periods	Static periodic	EDF
[18] Hen05	Optimizat.	TT	Coord.	Online	Periods	Varying periodic	EDF
[21] Ben06	Optimizat.	TT	Coord.	Online	Sequences	Dynamic pseudo per.	Flex.C.E
[25] Tab06	Bound d.	ET	Task	Online	Periods	Aperiodic TT	WiP
[26] Joh07	Bound d.	ET	Task	Online	Periods	Sporadic TT	Spor.
[27] Lem07	Bound d.	ET	Task	Online	Periods	Aperiodic TT	Elastic S.

Overview * FS vs. ES * Kernel



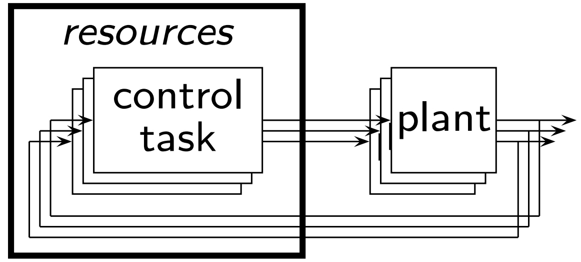
time-driven operation



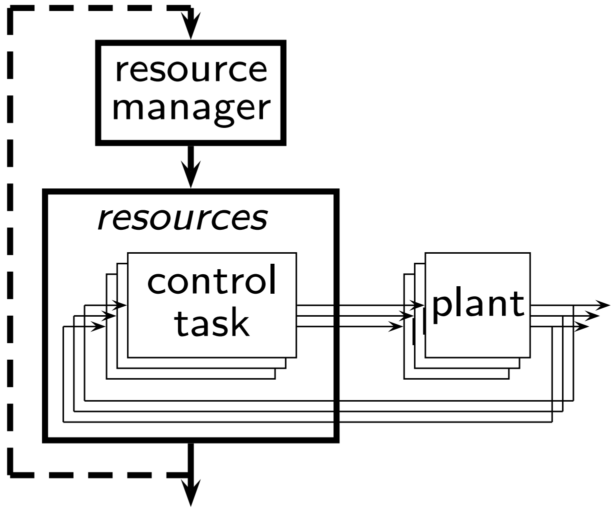
control-driven operation

Contents

1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions



2. Control of real-time control systems
   (a) Overview
   (b) Representative examples
      - periods (optimization)
      - sequences (optimization)
      - event-based

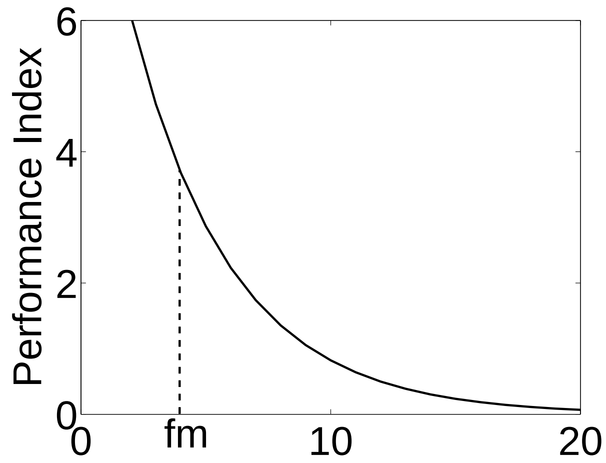


Representative examples (periods)

[14] On task schedulability in real-time control systems

- Set of n control tasks sharing a CPU
- Performance index for each control task (cost):

  $\Delta J(f_i) = J_D(f_i) - J$
- A minimum frequency for each task f_{mi} must be guaranteed



Representative examples (periods)

[14] On task schedulability in real-time control systems

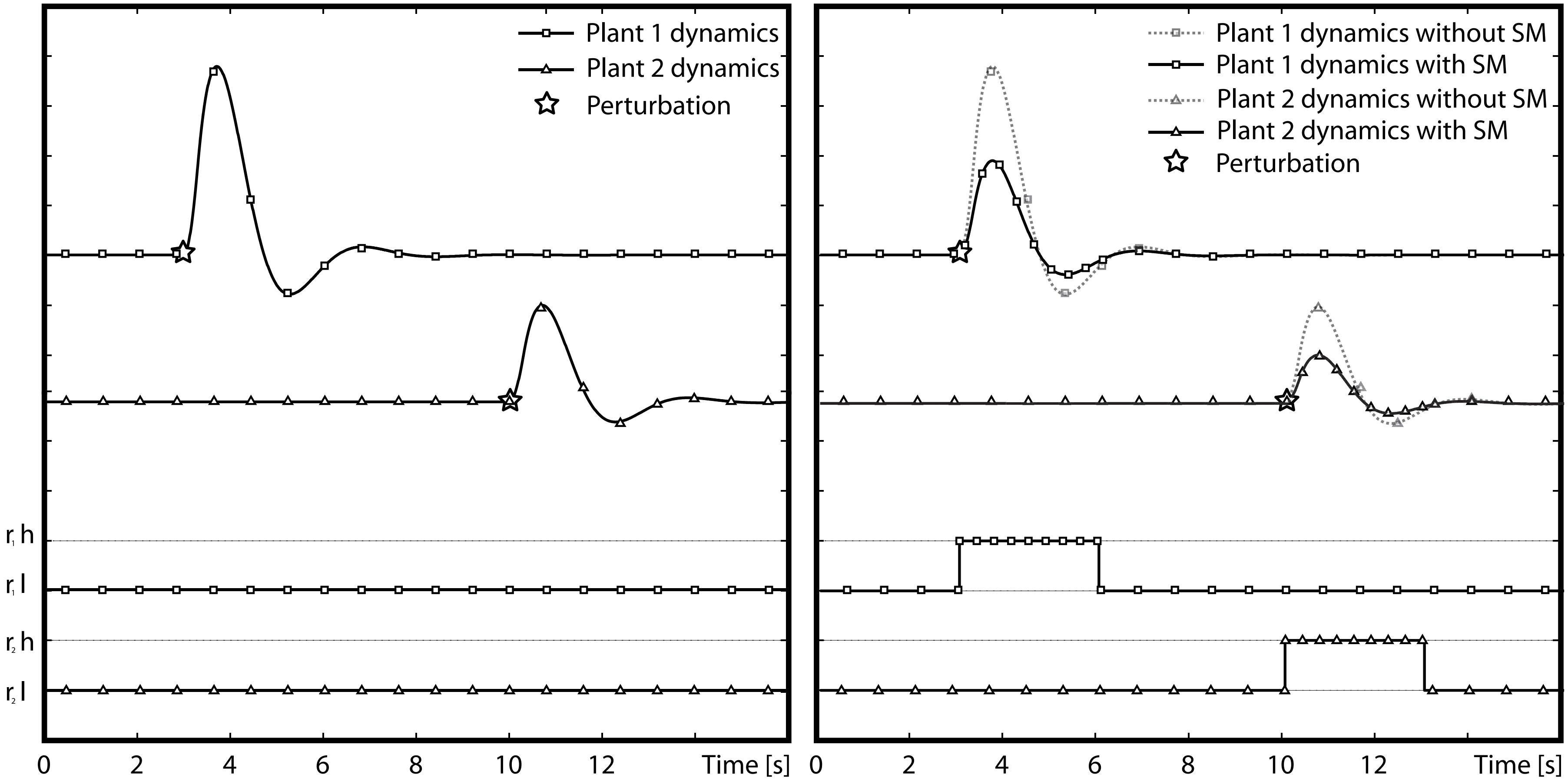
$$
\begin{aligned}
\text{min} \quad & \Delta J = \sum_{i=1}^{n} \omega_i \Delta J_i = \sum_{i=1}^{n} \omega_i \alpha_i e^{-\beta_i f_i} \\
\text{with respect to} \quad & f_1, f_2, \ldots, f_n \\
\text{subject to} \quad & \sum_{i=1}^{n} C_i f_i \leq A, \quad 0 < A \leq 1 \\
& f_i \geq f_{mi}, \quad i = 1, \ldots, n
\end{aligned} \tag{20}
$$

Solution:

It statically sets several frequencies $\geq f_{mi}$ and the rest $= f_{mi}$

Representative examples (periods)

[16] Optimal state feedback based resource allocation for resource-constrained control tasks. Key observation:



Representative examples (periods)

[16] Optimal state feedback based resource allocation for resource-constrained control tasks

- Set of n control tasks sharing a CPU
- A minimum resource share is guaranteed per task: $r_{i,min} = \frac{c_i}{h_{i,max}}$
- Performance index for each control task (benefit): $\alpha_i r_i + \beta_i$
- Instantaneous feedback: $e_i = |x_k|$

Representative examples (periods)

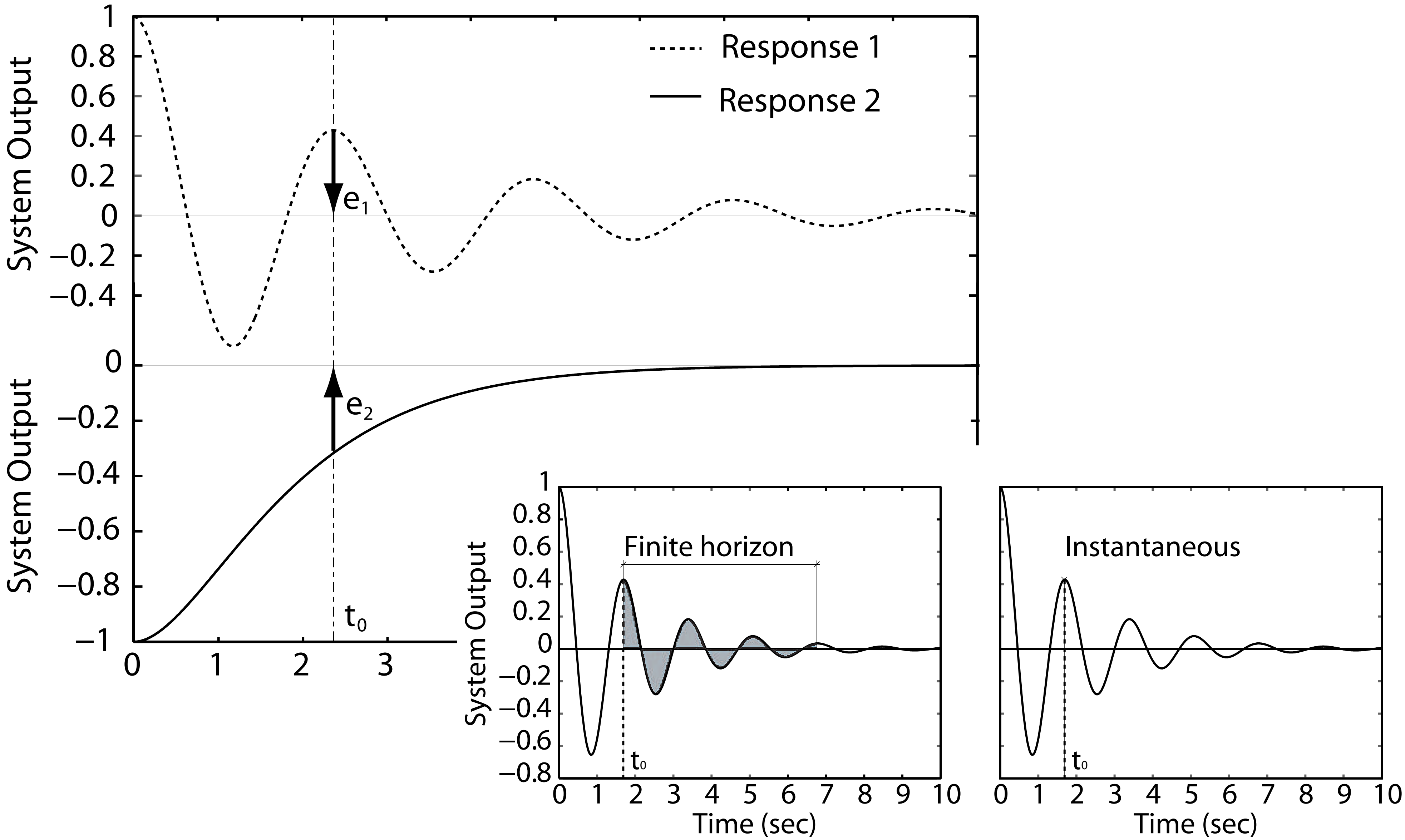
[16] Optimal state feedback based resource allocation for resource-constrained control tasks

$$
\begin{aligned}
\text{max} \quad & \sum_{i=1}^{n} \omega_i p_i(r_i) e_i = \sum_{i=1}^{n} \omega_i (\alpha_i r_i + \beta_i) |x_k| \\
\text{with respect to} \quad & r_1, r_2, \ldots, r_n \\
\text{subject to} \quad & \sum_{i=1}^{n} \Delta r_i \leq U_s(t) \quad \text{and} \quad \Delta r_i \geq 0, \quad i = 1, \ldots, n \\
\text{where} \quad & r_i = r_{i,min} + \Delta r_i \text{ and } U_s(t) = \text{available\_slack}(t)
\end{aligned} \tag{21}
$$

Solution: Assign all slack to the task whose plant has the largest error, where slack is the unused and thus available resources

Drawback: Instantaneous feedback, e.g. $|x_k|$, may be not helpful in certain scenarios

Representative examples (periods)



Representative examples (periods)

[18] Optimal on-line sampling period assignment for real-time control tasks based on plant state information

- Set of n control tasks sharing a CPU
- Performance index for each control task based on a finite horizon prediction (cost):

$$J(x_0, h, T_{fbs}) = x_0^T S x_0 + T_{fbs} \bar{J} \tag{22}$$

where

- $\bar{J} = \frac{1}{h}\big(\mathrm{tr}\ S(h) R_1(h) + J_v(h)\big)$ is the stationary cost per time unit
- $x_0^T S x_0$ is the transient cost, where S is the solution to:

Representative examples (periods)

[18] Optimal on-line sampling period assignment for real-time control tasks based on plant state information

- the algebraic Riccati equation (23) for optimal controllers providing the optimal cost (24) for a standard quadratic cost function (25)

$$S = \Phi^T S \Phi + Q_1 - (\Phi^T S \Gamma + Q_{12})(\Gamma^T S \Gamma + Q_2)^{-1}(\Gamma^T S \Phi + Q_{12}^T) \tag{23}$$

$$J = x_0^T S x_0 + \sum_{k=0}^{N-1} \Big(\mathrm{tr}\ S(h) R_1(h) + J_v(h) \Big) \tag{24}$$

$$J = \mathrm{E}_v \Big\{ \sum_{k=0}^{N-1} \Big(x(kh)^T Q_1 x(kh) + 2x(kh)^T Q_{12} u(kh) + u(kh)^T Q_2 u(kh) + J_v(h) \Big) \Big\} \tag{25}$$

- the Lyapunov equation (26) for an arbitrary state feedback control law $u(kh) = -Kx(kh)$, to be evaluated in (24)

$$S = (\Phi - \Gamma L)^T S (\Phi - \Gamma L) + Q_1 - Q_{12} L - L^T Q_{12}^T + L^T Q_2 L \tag{26}$$

Note that Φ, Γ, Q_1, Q_{12}, Q_2, J_v, R_1, and S all depend on the sampling interval h.

Representative examples (periods)

[18] Optimal on-line sampling period assignment for real-time control tasks based on plant state information

$$
\begin{aligned}
\text{min} \quad & \sum_{i=1}^{n} J_i(x_i(t_0), h_i, T_{fbs}) \\
\text{with respect to} \quad & h_1, h_2, \ldots, h_n \\
\text{subject to} \quad & \sum_{i=1}^{n} \frac{C_i}{h_i} \leq 1 \\
& h_i \geq 0, \quad i = 1, \ldots, n
\end{aligned} \tag{27}
$$

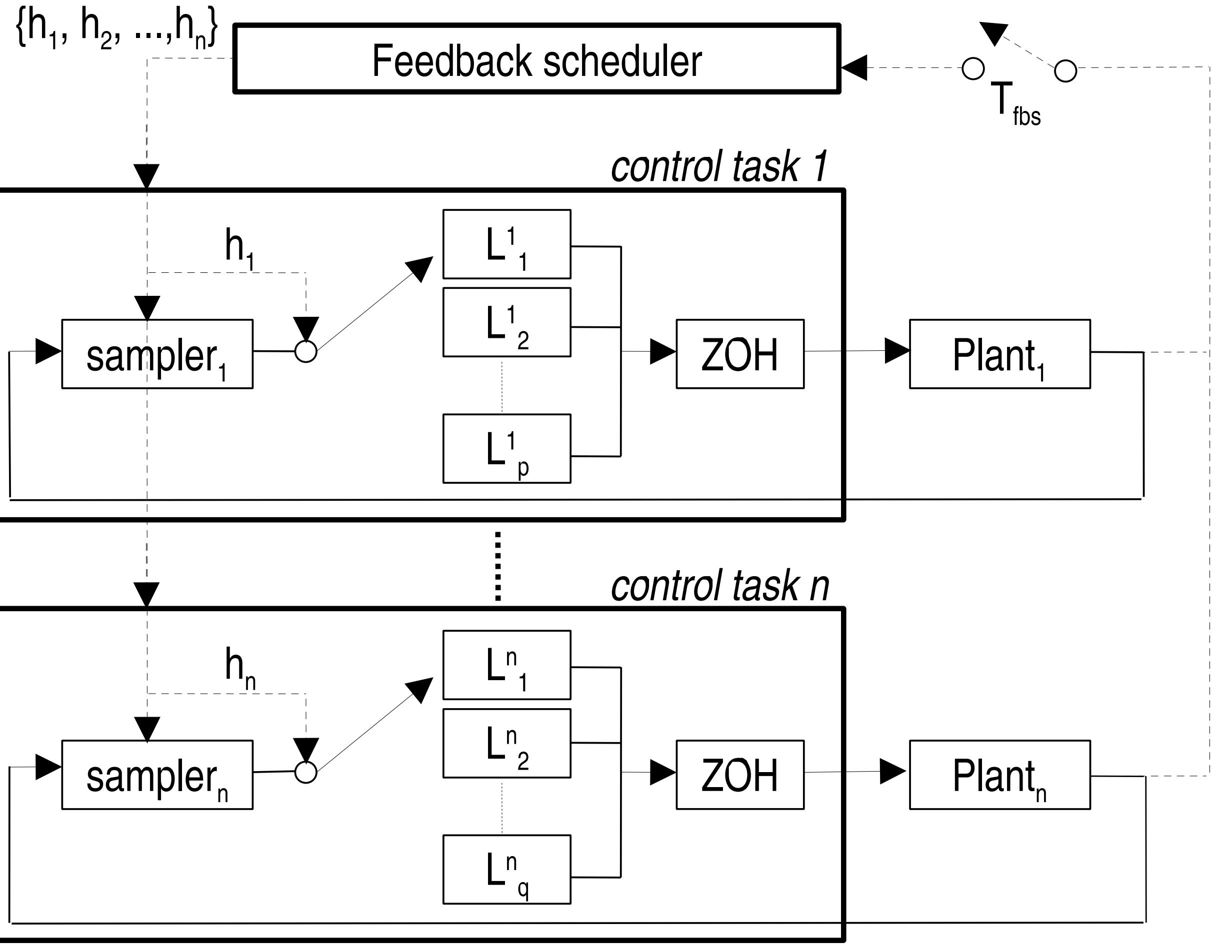
Solution:

Finding an analytical solution in the general case is not possible.

But an approximate general solution exists and works [28]

Representative examples (periods)

Operation of [16] ($T_{fbs} \rightarrow h_i$) or [18] ($T_{fbs} >> h$).



Representative examples (sequences)

[19] Integration of off-line scheduling and optimal control

- Set of n control tasks sharing a CPU
- Repeated cycle divided into p slots, cycle of length T_p
- Tasks execute within slots
- LQ controllers

$$\min_{\hat{u}_i} \quad E\left[\bar{x}_i^T \bar{S}_i \bar{x}_i(n) + \sum_{i=1}^{n} \begin{bmatrix} \bar{x}_i \\ \bar{u}_i \end{bmatrix}^T \begin{bmatrix} \bar{Q}_1 & \bar{Q}_{12} \\ \bar{Q}_{12}^T & \bar{Q}_2 \end{bmatrix}_i \begin{bmatrix} \bar{x}_i \\ \bar{u}_i \end{bmatrix}\right] \qquad (28)$$

$$\text{such that} \quad \bar{x}_i(k+1) = \bar{A}_i \bar{x}_i(k) + \bar{B}_i \bar{u}_i(k) + \bar{G}_i \bar{v}_i(k)$$

- Let s denote a scheduling sequence and S_p a set of schedules

Representative examples (sequences)

[19] Integration of off-line scheduling and optimal control

Finding the optimal schedule formulated as a combinatorial optimization problem

$$
\begin{array}{rl}
\text{min} & f^{per}(s,p) \\
\text{when} & s \in S_p \\
 & p = 1 \dots T_p
\end{array}
\qquad (29)
$$

where $f^{per}(s,p)$ is a performance measure derived from (28)

Solution: A periodic schedule $\hat{s}(t) = s_1 s_2 \dots s_p s_1 \dots,$ where $\hat{s}(t)$ indicates the controller run at time t, and a periodic linear feedback law such that $u_{\hat{s}(t)} = K_t x_{\hat{s}(t)}(t)$

Representative examples (event-based)

[23] Stable and real-time scheduling of a class of hybrid dynamic systems

- N continuous dynamic plants

$$\dot{x}_i = A_i x_i + b_i u_i, \quad i = 1, \ldots, N \tag{30}$$

- Discrete-event scheduling

$$Event(i, T_k) = \begin{cases} 1 & \text{if } \|x_i(T_k)\| = \max_{j=1,\ldots,N} \|x_j(T_k)\| \text{ at } T_k \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

During $t \in [\ T_k \quad T_k + h\ )$ *plant*$_i$ runs in closed loop (rest in open loop)

- Objective: to ensure stability

Representative examples (event-based)

[23] Stable and real-time scheduling of a class of hybrid dynamic systems

- Control design specification: to ensure asymptotical and exponential stability for all plants
- Outcome:
  - Sufficient conditions
  - Stabilizing feedback gains

Observation: similar to previous feedback-scheduling approaches but using an event-based scheduling and single feedback gains.

Representative examples (event-based)

[25] Preliminary results on state-triggered scheduling of stabilizing control tasks

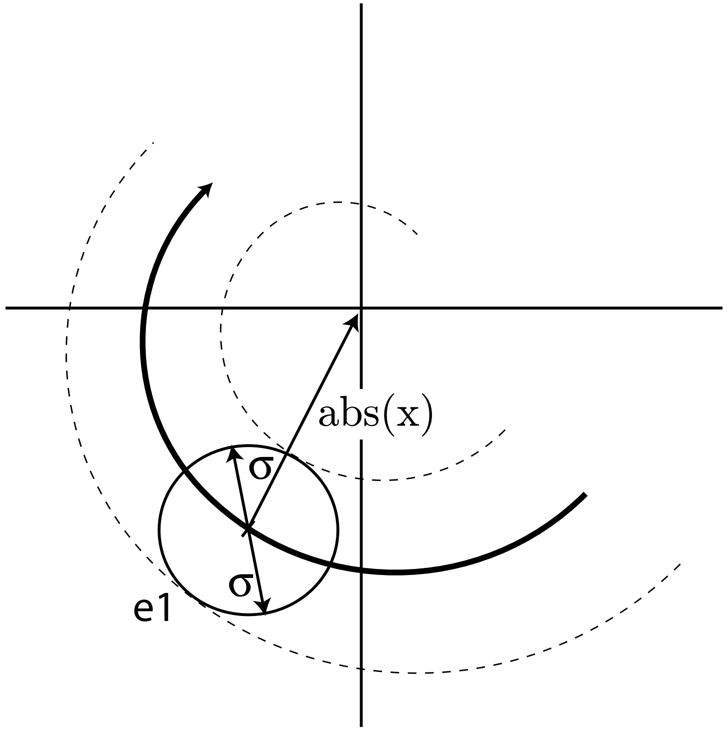
- Closed loop continuous time system with discrete controller

$$\dot{x} = f(x, k(x+e)) \quad \text{where} \quad e(t) = x(t_i) - x(t) \tag{32}$$

- Event-triggered executions:

  $|e(t)| \leq \sigma |x(t)|$

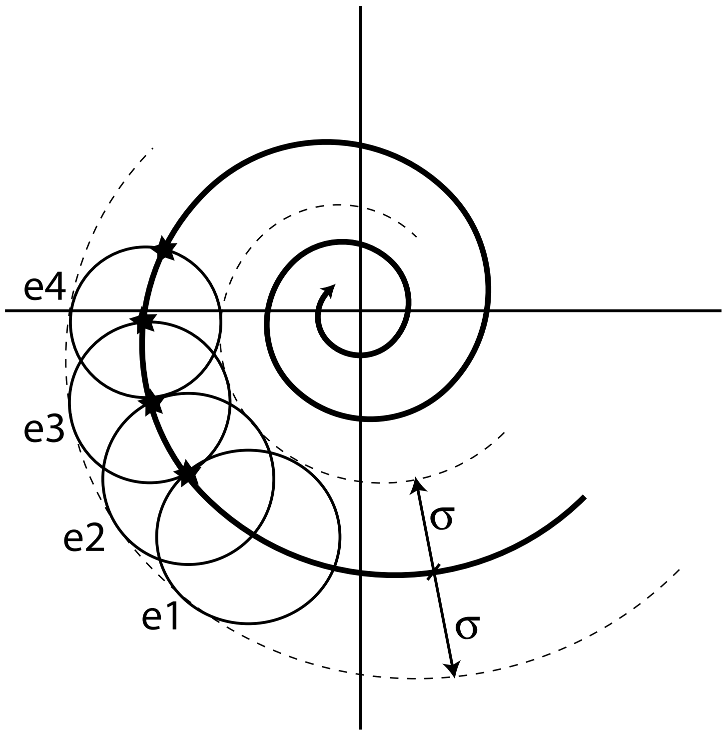
  to enforce stability



Representative examples (event-based)

[25] Preliminary results on state-triggered scheduling of stabilizing control tasks

- avoids accumulation points
- provides estimates of the time between consecutives executions



Miscellaneous

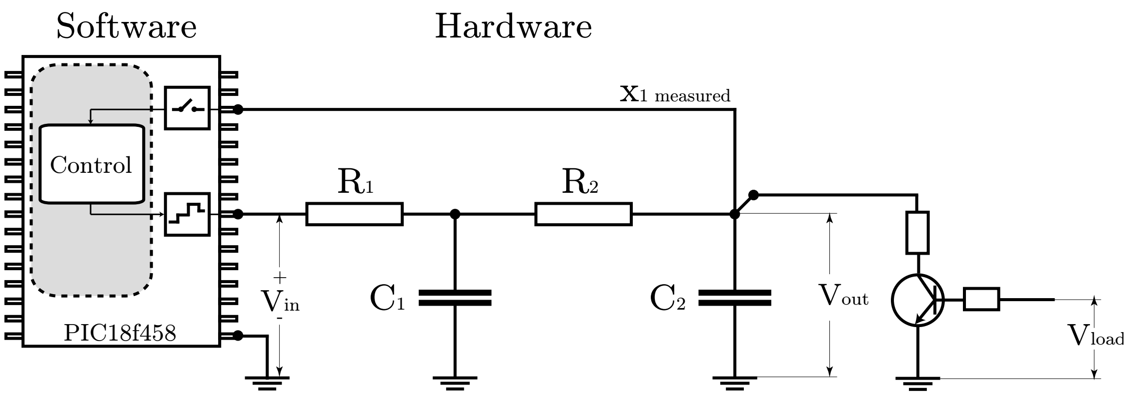
Miscellaneous - Jitters

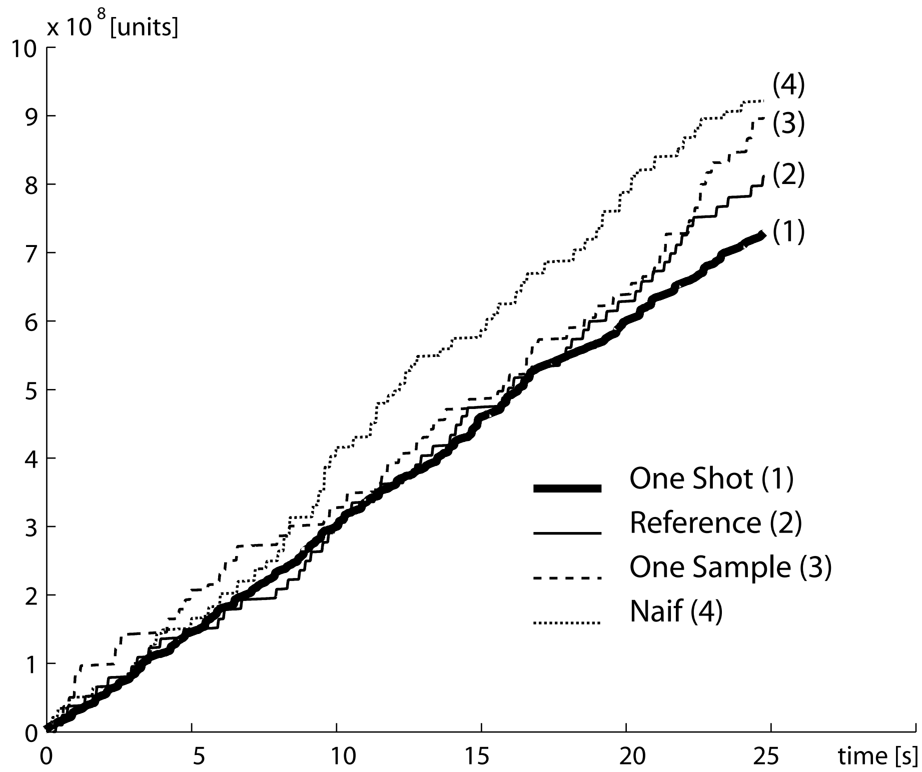
Simulation: Some feedback scheduling approaches vs. jitters. Three control tasks controlling RCRC circuits. Evaluation using a quadratic cost function.

Approach	Original	Indep. Proc.
Static approach	109.05	105.82
Off-line RM [14]	121.85	96.59
On-line FS [15]	99.92	98.74
On-line instantaneous FS [16]	90.63	64.41
On-line finite horizon FS [28]	100.61	86.99
Heuristic on-line cyclic scheduling [19]	62.43	62.48

Miscellaneous - No jitters

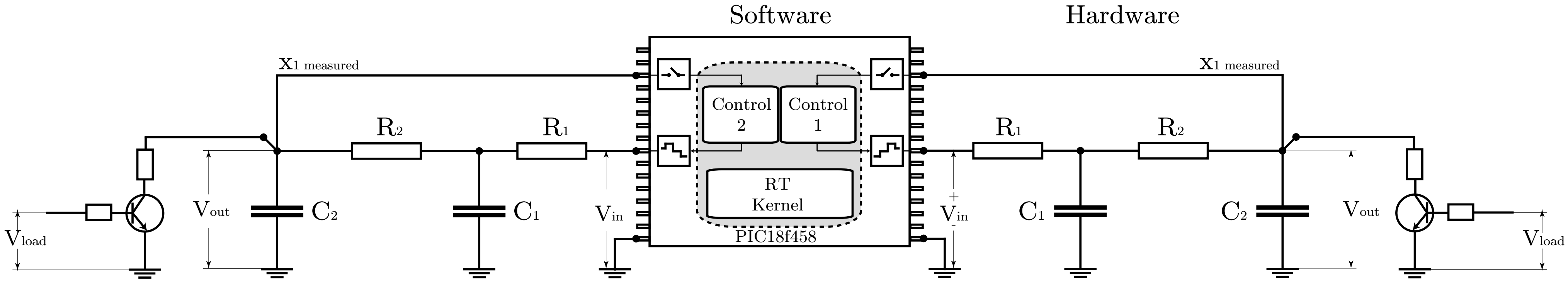
Implementation: one-shot task model.

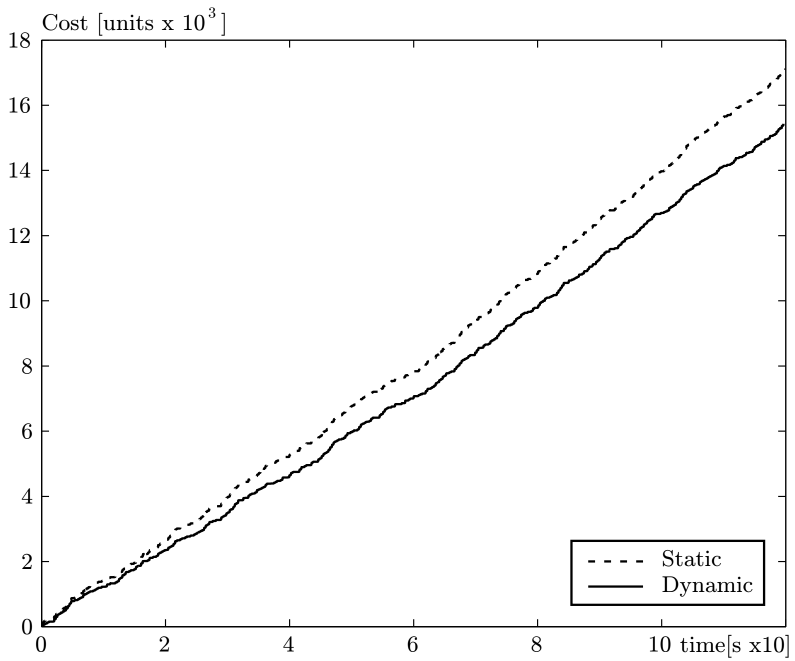


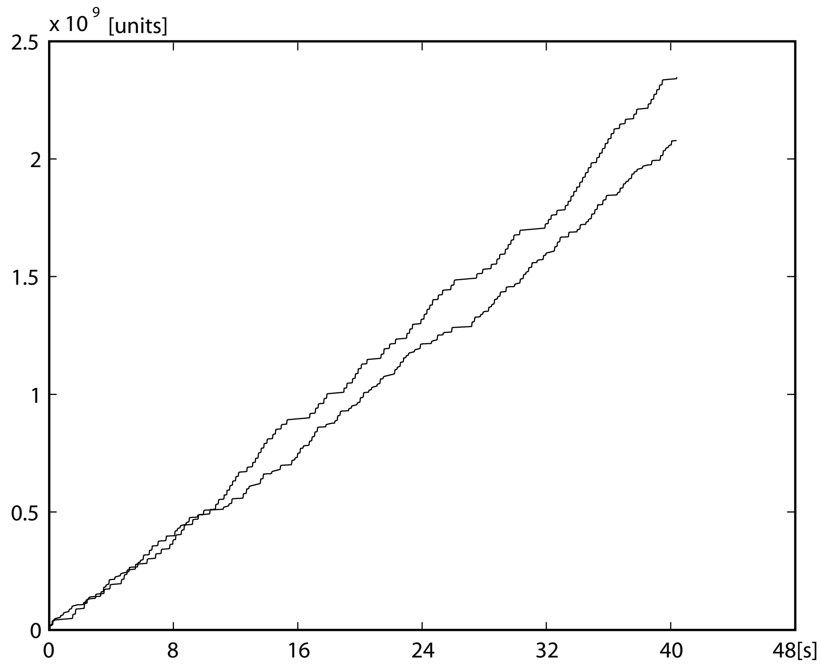


Miscellaneous - Feedback scheduling

Implementation: FS - [16] (left) and [18]+[28] (right)







Miscellaneous - Simulation tool

TrueTime (http://www.control.lth.se/truetime/)

Simulation of Networked and Embedded Control Systems

- Matlab/Simulink-based simulator for real-time control systems.
- Facilitates co-simulation of controller task execution in real-time kernels, network transmissions, battery-powered devices, and continuous plant dynamics.

Summary

- Networked and embedded control systems are everywhere
  - Resources
  - Timing
  - Dynamic behavior
- Overcoming separation of concerns
  - Real-time computing of control systems
  - Control of real-time control systems

Appendix 1 (controllability)

Controllability. Is (13) controllable? Yes.

Proof. We assume that the standard system (2) is controllable

$$W_c = \det\left(\begin{bmatrix}\Gamma \; \Phi\Gamma \; \cdots \; \Phi^{n-1}\Gamma\end{bmatrix}\right), \quad \det(W_c) \neq 0 \tag{33}$$

Let us define

$$\phi_a(h_k, \tau_k) = \begin{bmatrix} \Phi(h_k) & \Phi(h_k - \tau_k)\Gamma(\tau_k) \\ 0 & 0 \end{bmatrix} \tag{34}$$

$$\Gamma_a(h_k, \tau_k) = \begin{bmatrix} \Gamma(h_k - \tau_k) \\ I \end{bmatrix} \tag{35}$$

$$x_a(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \tag{36}$$

Appendix 1 (controllability)

Let the system state at $k = n$ be

$$x(n) = \prod_{i=1}^{n} \phi_a(h_{n-i+1}, \tau_{n-i+1}) x(0) + W_c U \tag{37}$$

with

$$\begin{aligned} W_c &= \left[\underbrace{\Gamma_a(h_n, \tau_n)}_{\text{for } j=n} \; \ldots \; \underbrace{\left(\prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1}) \right) \Gamma_a(h_1, \tau_1)}_{\text{for } j=1} \right] \\ U &= \left[u^T(n-1) \; \ldots \; u^T(0) \right]^T \end{aligned} \tag{38}$$

Appendix 1 (controllability)

Subtituting (34) and (35) into (38) we obtain

$$W_c = \begin{bmatrix} \overbrace{\Gamma_0(h_n \tau_n)}^{j=n} & \cdots & \overbrace{\left(\prod_{i=1}^{n-2} \Phi(h_{n-i+1})\right)\Gamma_1(h_2, \tau_2) + \left(\prod_{i=1}^{n-1} \Phi(h_{n-i+1})\right)\Gamma_0(h_1, \tau_1)}^{j=1} \\ I & \cdots & 0 \end{bmatrix} \tag{39}$$

For MIMO systems, (13) is controllable if $\det(W_c) \neq 0$. Developing the determinant from the last row, and setting $\tau_k = 0$ and $h_k = h$, we obtain condition (33)

Appendix 1 (controllability)

$\det(W_c)$ is a continuous function of a continuous variable

$$\begin{array}{rcl} det(W_c)\ :\ \mathbf{R}^{nxn} & \rightarrow & \mathbf{R} \\ (h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n) & \rightarrow & det(W_c\left[h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n\right]) \end{array}$$

If the original system (2) is controllable, then

$$\exists (h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n) \quad | \quad det(W_c) \neq 0$$

And due to continuity

$$\exists B((h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n), \delta) \quad | \quad det(W_c) \neq 0$$

$\square$

Appendix 2 (observability)

Observability. Is (13) observable? Yes, if the output matrix outputs the additional variable.

Proof. We assume that the standard system (2) is observable

$$W_o = \det\left(\begin{bmatrix} C \\ C\Phi(h) \\ \vdots \\ C\Phi^{n-1}(h) \end{bmatrix}\right), \quad \det(W_o) \neq 0 \tag{40}$$

and we use definitions (34), (35) and (36), and we set as output matrix

$$C_a = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} \tag{41}$$

Appendix 2 (observability)

Without losing generality, if $u_k = 0$, the initial state can be observed in n steps, being n the order of (13)

$$\begin{aligned} y_a(0) &= C_a x_a(0) \\ y_a(1) &= C_a x_a(1) = C\phi_a(h_1, \tau_1) x_a(0) \\ &\vdots \\ y_a(n) &= C_a \prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1}) x_a(0) \end{aligned} \tag{42}$$

From (42), the obserbability matrix is

$$W_o = \begin{bmatrix} C_a \\ \vdots \\ C_a \prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1}) \end{bmatrix} \tag{43}$$

Appendix 2 (observability)

Subtituting (41) and (34) into (43) we obtain

$$W_o = \begin{bmatrix} C & 0 \\ 0 & I \\ \vdots & \vdots \\ C\prod_{i=1}^{n-1}\Phi(h_{n-i+1}) & C\prod_{i=1}^{n-2}\Phi(h_{n-i+1})\Phi(h_1-\tau_1)\Gamma(\tau_1) \\ 0 & 0 \end{bmatrix} \quad (44)$$

For MIMO systems, $W_o \in \mathbb{R}^{2n\times n}$. Therefore, we can construct W_o^* with n rows of W_o. Then, (13) is observable if $\det(W_o^*) \neq 0$.

Appendix 2 (observability)

For W_o^* we pick rows $2, 3, 5, 7, \ldots, n-1$ of W_o

$$W_o^* = \begin{bmatrix} 0 & I \\ C\Phi(h_1) & C\Phi(h_1 - \tau_1)\Gamma(\tau_1) \\ \vdots & \vdots \\ C\prod_{i=1}^{n-1}\Phi(h_{n-i+1}) & C\prod_{i=1}^{n-2}\Phi(h_{n-i+1})\Phi(h_1 - \tau_1)\Gamma(\tau_1) \end{bmatrix} \quad (45)$$

With constant period and $\tau = 0$ we obtain

$$W_o^* = \begin{bmatrix} 0 & I \\ C\Phi & 0 \\ \vdots & \vdots \\ C\Phi^{n-1} & 0 \end{bmatrix} \quad (46)$$

Appendix 2 (observability)

Developing the determinant of (46) by the first row

$$\det(W_o^*) = \pm\det\left(\begin{bmatrix} C\Phi \\ \vdots \\ C\Phi^{n-1} \end{bmatrix}\right) = \pm\det\left(\begin{bmatrix} C \\ \vdots \\ C\Phi^{n-2} \end{bmatrix}\right)\det(\Phi) \quad (47)$$

Note: $det(\Phi) \neq 0$ and recall (40) $\Rightarrow \det(W_o^*) \neq 0$.

As before, $\det(W_o)$ is a continuous function of a continuous variable. If the original system (2) is controllable, then

$$\exists(h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n) \quad | \quad det(W_o^*) \neq 0$$

And due to continuity

$$\exists B((h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n), \delta) \quad | \quad det(W_o^*) \neq 0$$

□

References (1)

[1] Åström, K.J., Wittenmark, B., *Computer controlled systems*. Prentice Hall, 1997

[2] Burns, A. and Wellings, A. J., *Real-Time Systems and Programming Languages: ADA 95, Real-Time Java, and Real-Time POSIX*. 3rd. Edition, Addison-Wesley, 2001

[3] Buttazzo, G., *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Second Edition, Springer, 2005

[4] Stankovic, J. A., "Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems," *Computer* 21, 10, Oct. 1988

[6] Locke, C. D., "Software architecture for hard real-time applications: cyclic executives vs. fixed priority executives," *Real-Time Systems,* 4(1), pp. 37-53, 1992

[5] Liu, C.L., and Layland, J.W., "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM* 20(1), pp. 40–61, 1973

[7] Dogruel, M., and Özgüner, U, "Stability of a Set of Matrices: A Control Theoretic Approach," 34th IEEE Conference of Decision and Control, 1995

[8] Kao, C.-Y. and Lincoln, B., "Simple stability criteria for systems with time-varying delays", *Automatica,* 40(8), pp. 1429-1434, 2004

[9] Naghshtabrizi, P., Hespanha, J., and Teel, A. R., " On the robust stability and stabilization of sampled-data systems: A hybrid system approach", 45th IEEE Conference on Decision and Control, 2006

References (2)

[10] Årzén, K.-E., Cervin, A., Eker, J., Sha, L., "An introduction to control and scheduling co-design,", 39th IEEE Conference on Decision and Control, 2000)

[11] Henzinger, T.A., Horowitz, B., and Kirsch, C.M., "Giotto: a time-triggered language for embedded programming", First International Workshop on Embedded Software, LNCS 2211, pp. 166–184, 2001

[12] Martí, P., Velasco, M., "Toward Flexible Scheduling of Real-Time Control Tasks: Reviewing Basic Control Models", 10th International Conference on Hybrid Systems: Computation and Control, 2007

[13] Balbastre, P., Ripoll, I., Vidal, J., Crespo, A., "A task model to reduce control delays," *Journal of Real-Time Systems*, vol.27, n.3, pp. 215–236, 2004

[14] Seto, D., Lehoczky, J. P., Sha, L., Shin, K. G., "On task schedulability in real-time control systems," 17th IEEE Real-Time Systems Symposium, December 1996

[15] Eker, J., Hagander, P., Årzén, K.-E.: "A Feedback Scheduler for Real-time Control Tasks," *Control Engineering Practice,* Vol. 8 N. 12, December 2000

[16] Martí, P., Lin, C., Brandt, S., Velasco, M., Fuertes, J.M. "Optimal State Feedback Based Resource Allocation for Resource-Constrained Control Tasks", 25th IEEE Real-Time Systems Symposium, December 2004

References (3)

[17] Palopoli, L., Pinello, C., Bicchi, A., Sangiovanni-Vincentelli, A., “Maximizing the Stability Radius of a Set of Systems Under Real-Time Scheduling Constraints,” *IEEE Transactions on Automatic Control,* Vol. 50, N. 11, pp. 1790-1795, Nov. 2005

[18] Henriksson, D. and Cervin, A., “Optimal On-line Sampling Period Assignment for Real-Time Control Tasks Based on Plant State Information”, 44th IEEE Conference on Decision and Control and European Control Conference 2005, December 2005

[19] Rehbinder, H., Sanfridson, M., “Integration of off-line scheduling and optimal control”, 12th Euromicro Conference on Real-Time Systems, 2000

[20] Hristu-Varsakelis, D., “Feedback control systems as users of a shared network:communication sequences that guarantee stability”, 40th IEEE Conference on Decision and Control, 2001

[21] Ben Gaid, M-M., Çela, A., Hamam, Y., Ionete, C., “Optimal Scheduling of Control Tasks with State Feedback Resource Allocation,” 2006 American Control Conference, June 2006

[22] Årzén, K.-E., “A Simple Event-Based PID Controller,” 14th World Congress of IFAC, January, 1999

[23] Zhao, Q.C., Zheng, D.Z., “Stable and Real-Time Scheduling of a Class of Hybrid Dynamic Systems,” *Discrete Event Dynamic Systems,* Vol. 9, N. 1, pp. 45-64, 1999

References (4)

[24] Heemels, W.P.M.H., Sandee, J.H. , "Practical stability of perturbed event-driven controlled linear systems," 2006 American Control Conference
[25] Tabuada, P., Wang, X. "Preliminary results on state-triggered scheduling of stabilizing control tasks", 45th IEEE Conference on Decision and Control, December 2006
[26] Johannesson, E., Henningsson, T., Cervin, A., "Sporadic Control of First-Order Linear Stochastic Systems", Hybrid Systems: Computation and Control, April 2007
[27] Lemmon, M., Chantem, T., Hu, X., Zyskowski, M., "On Self-Triggered Full Information H-infinity Controllers", Hybrid Systems: Computation and Control, April 2007
[28] Castañé, R., Martí, P., Velasco, M., Cervin, A., "Resource Management for Control Tasks Based on the Transient Dynamics of Closed-Loop Systems", 18th Euromicro Conference on Real-Time Systems, 2006