Feedback control for computing systems

Alberto Leva

DEIB, Politecnico di Milano, Italy
alberto.leva@polimi.it

Lund University, 13–17 November 2017

Purpose of this course

- Provide a quick and basic, yet consistent introduction to the systems and control theory (centred primarily on the discrete-time framework);
- revisit some relevant problems related to computing systems design, at various levels, with a system- and control-theoretical attitude;
- present some results obtained with the proposed approach;
- highlight the advantages that the approach yields – where applicable – over heuristic techniques.

- In perspective, foster a better and methodologically grounded cooperation between the System & Control (SC) and the Computer Science & Engineering (CSE) communities.

Credits

Alphabetical order

- Anant K. Agarwal,
- Davide Arcelli,
- Luciano Baresi,
- Enrico Bini,
- Vittorio Cortellessa,
- Antonio Filieri,
- William Fornaciari,
- Carlo Ghezzi,
- Lars Grunske,
- Sam Guinea,
- Henry C. Hoffman,
- Martina Maggio,
- Alessandro V. Papadopoulos,
- Maria Prandini,
- Giovanni Quattrocchi,
- Marco D. Santambrogio,
- Federico Terraneo,
- ...

apologies to the forgotten.

Course outline

- **Day 1**
  Morning: dynamic systems.
  Afternoon: multi-physic modelling & simulation, case studies.
- **Day 2**
  Morning: feedback and control design.
  Afternoon: hands-on practice.
- **Day 3**
  Morning: applications 1
  – scheduling & resource allocation, WSN clock synchronisation.
  Afternoon: hands-on practice.
- **Day 4**
  Morning: applications 2
  – thermal/power/performance management, self-adaptive software (some cases).
  Afternoon: hands-on practice.
- **Day 5**
  Morning: feedback loops in computers – peculiarities, takeaways, coverage, discussion.
  Afternoon: conclusions, references, about the exam, question & proposal time.

Tools

- OpenModelica
  `https://www.openmodelica.org`
- wxMaxima
  `https://sourceforge.net/projects/wxmaxima`
- Scilab
  `https://www.scilab.org`
- Some Scilab scripts and a Modelica library provided as course material.

– Day 1 –

Outline

- **Morning**: *dynamic systems (our main tool)*
  - General definitions;
  - motion, equilibrium, stability;
  - the Discrete-Time Linear Time-Invariant (DT LTI) class:
    - state space representation,
    - the $\mathscr{Z}$ transform,
    - input-output representation;
  - block diagrams.
- **Afternoon**: *multi-physics dynamic modelling and simulation*
  - Definitions and motivations (in a nutshell);
  - the Modelica language and the OpenModelica translator;
  - commented examples.

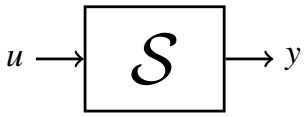
Dynamic systems

System

A mathematical representation (model) of something that evolves

Note: for the moment we restrict the scope to models involving real-valued quantities, a very few words on quantisation in due course.

Simple representation



u – input(s)
y – output(s)

Model ingredients

- what the system evolves upon:
  - the continuous time t;
  - an integer k counting some events, in the broadest sense of the term, called the discrete time (index).
- the evolution law:
  - $u[t_0,t] \rightarrow y[t_0,t]$;
  - $u[k_0,k] \rightarrow y[k_0,k]$.

Is anything missing? Sometimes, yes.

Dynamic system

The most general definition

If the knowledge of $u[t_0,t]$ — or $u[k_0,k]$
allows to determine $y[t_0,t]$ — or $y[k_0,k]$
the system is said to be non dynamic,
dynamic otherwise.

Dynamic system

Input, output, and *state*

- Non dynamic system:
  $y[t_0,t]$ or $y[k_0,k]$ depends only on $u[t_0,t]$ or $u[k_0,k]$.
- Dynamic system:
  $y[t_0,t]$ or $y[k_0,k]$ depends on $u[t_0,t]$ or $u[k_0,k]$,
  and on the initial values $x[t_0]$ or $x[k_0]$ of some quantities.
- These are called the state variables, and form the state (vector).
- The number of state variables is called the order of the system.

Dynamic system

How can we express this in mathematical terms?

In several ways. We see the only two relevant for us.

- Continuous-Time (CT) system:

$$\begin{cases} \frac{dx(t)}{dt} &= f\big(x(t),u(t),t\big) \\ y(t) &= g\big(x(t),u(t),t\big) \end{cases}$$

- Discrete-Time (DT) system:

$$\begin{cases} x(k) &= f\big(x(k-1),u(k-1),k\big) \\ y(k) &= g\big(x(k),u(k),k\big) \end{cases}$$

NOTE: in both cases we have two equations; the first is termed the state equation, the second the output equation (or sometimes transformation).

Dynamic system

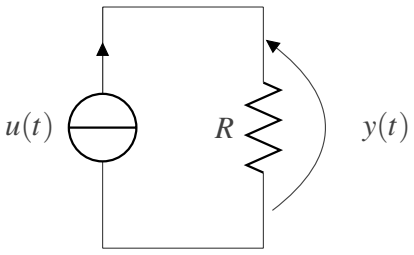
Some more definitions (for both the CT and the DT case)

- Linear system:
  $f(\cdot,\cdot,\cdot)$ and $g(\cdot,\cdot,\cdot)$ linear in x and u.
- Time-Invariant system:
  $f(\cdot,\cdot,\cdot)$ and $g(\cdot,\cdot,\cdot)$ not depending on t or k.
- Proper (sometimes, *strictly* proper) system:
  $g(\cdot,\cdot,\cdot)$ not depending on u,
  i.e., the input acts on the output only through the state.
- SISO (Single-Input, Single-Output) system:
  u and y – not necessarily x – scalars.

Let us see some examples, leading to a minimal taxonomy of dynamic systems (enough for us). Also, for our purposes we can limit the scope the SISO case.

Dynamic system

Examples (1/4) – resistor with prescribed voltage



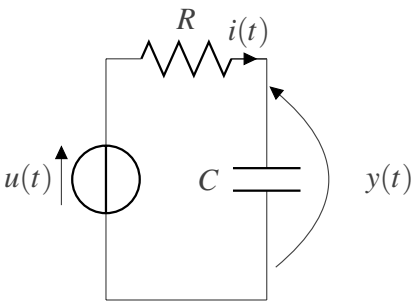
Model:

$$y(t) = Ru(t).$$

- Continuous-time,
- non dynamic,
- linear,
- time-invariant.

Dynamic system

Examples (2/4) – RC circuit with prescribed voltage



$$\underbrace{RC\frac{dy(t)}{dt}}_{i(t)}+y(t)=u(t)$$

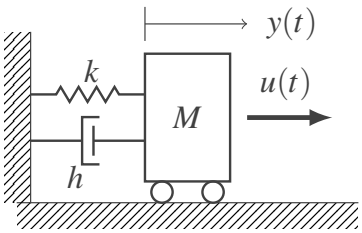
Model:

$$\begin{cases}\frac{dx(t)}{dt} &= -\frac{1}{RC}x(t)+\frac{1}{RC}u(t)\\ y(t) &= x(t)\end{cases}$$

- Continuous-time,
- dynamic,
- linear,
- time-invariant,
- proper,
- order 1.

Dynamic system

Examples (3/4) – mass with prescribed force, spring and damper



$$M\frac{d^2y(t)}{dt^2}=u(t)-ky(t)-h\frac{dy(t)}{dt}$$

$$x_1(t):=y(t),\ x_2(t):=\frac{dy(t)}{dt}$$

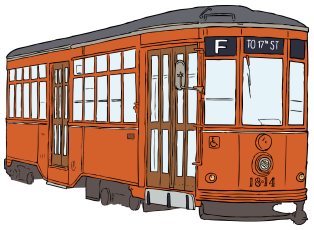
Model:

$$\begin{cases}\frac{dx_1(t)}{dt} &= x_2(t)\\ \frac{dx_2(t)}{dt} &= -\frac{k}{M}x_1(t)-\frac{h}{M}x_2(t)+\frac{1}{M}u(t)\\ y(t) &= x_1(t)\end{cases}$$

- Continuous-time,
- dynamic,
- linear,
- time-invariant,
- proper,
- order 2.

Dynamic system

Examples (4/4) – tram



$y(k)$: passengers after leaving stop k;
$u(k)$: passengers getting on minus passengers getting off at stop k
$\Rightarrow y(k) = y(k-1) + u(k)$.

Model (with proper state equation):

$$\begin{cases} x(k) &= x(k-1) + u(k-1) \\ y(k) &= x(k) + u(k) \end{cases}$$

- Discrete-time,
- dynamic,
- linear,
- time-invariant,
- not proper,
- order 1.

...and so on.

Motion

General definition

Note: in this lecture we mostly speak DT, for CT just replace k with t.

- Initial state + input at a certain start time $\Rightarrow$ motion, i.e.,

$$\left.\begin{matrix} x(k_0) \\ u(k), k \geq k_0 \end{matrix}\right\} \quad \Rightarrow \quad x(k), y(k), k \geq k_0.$$

- We call $x(k)$ and $y(k)$, respectively,
  - the state motion
  - and the output motion

  produced by the initial state $x(k_0)$ and the input $u(k)$, starting at time k_0.

Motion

for the Time-Invariant (TI) case, to which we restrict the scope from now on

- Initial state + input $\Rightarrow$ motion independently of the start time, i.e.,

$$\left.\begin{array}{l} x(0) \\ u(k), k \geq 0 \end{array}\right\} \quad \Rightarrow \quad x(k), y(k), k \geq 0.$$

- Alternatively, we can say that with TI systems one can set the time axis origin wherever one wants (for best convenience, at zero).

Equilibrium

Definition (in the TI case)

If there exist some state vectors $\bar{x}$
such that $x(0) = \bar{x}$ and $u(k) = \bar{u}$, $k \geq 0$,
produce the constant state motion $x(k) = \bar{x}$, $k \geq 0$,
then those vectors are called equilibrium states,
or equilibria for short,
corresponding to the constant input $\bar{u}$.

Equilibrium

Finding equilibrium states and outputs

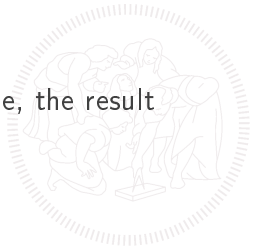
- A state vector $\bar{x}$ is an equilibrium for a given $\bar{u}$ if the consequent motion is $x(k) = x(k-1) = \bar{x}\ \forall k$.
- Thus to find equilibria one solves

$$\bar{x} = f(\bar{x}, \bar{u}).$$

- NOTE: the CT case is a bit different; one has to zero the state derivative, hence equilibria are found by solving

$$f(\bar{x}, \bar{u}) = 0.$$

- If some equilibrium state exists, and $g(\bar{x}, \bar{u})$ does not loose significance, the result is the corresponding equilibrium output $\bar{y}$.



Stability

Preliminaries

- The existence of an equilibrium implies NOTHING on what happens if the system does not start exactly at the equilibrium.
- Discussing this is a matter of stability.

- One can talk about stability of equilibria, motions, and sometimes systems.
- We define stability for an equilibrium, do not talk about motions, and move to systems later on.

Stability

Stable equilibrium

- Let $\bar{x}$ be an equilibrium for the constant input $\bar{u}$.
- Denote by $x_\Delta(k)$ the perturbed motion produced by
  - the input $\bar{u}$,
  - and the perturbed initial state $x(0) = \bar{x} + \Delta\bar{x}$.
- Not that in general, $x_\Delta(k)$ will not be constant.
- Denote by $||x||$ a norm (think here of the Euclidean norm) of vector x.

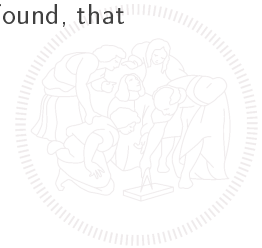
Stability

Stable equilibrium

- An equilibrium $\bar{x}$ corresponding to the constant input $\bar{u}$ is said to be stable if

$$\forall \varepsilon > 0 \; \exists \delta_\varepsilon > 0 \; : \; ||\Delta\bar{x}|| < \delta_\varepsilon \Rightarrow ||x_\Delta(k) - \bar{x}|| < \varepsilon \; \forall k \geq 0.$$

- Interpretation: stable equilibrium means that
  - no matter how close one wants the entire perturbed motion to remain to the equilibrium,
  - a maximum distance of the initial state from the equilibrium can be found, that fulfils the desire.
- If the above does not hold true, the equilibrium is unstable.



Stability

Asymptotically stable equilibrium

- An equilibrium $\bar{x}$ corresponding to $\bar{u}$ is said to be asymptotically stable if
  - it is stable,
  - and in addition
$$\lim_{k\to\infty} ||x_\Delta(k) - \bar{x}|| = 0.$$
- Clearly asymptotic stability implies stability, but not *vice versa*.

The DT LTI class

Preliminaries

- For the LTI class, that we see here in the DT context, there exists a very strong theory.
- The same is not true for more general (e.g., nonlinear) system classes.
- Therefore, control problems of the type addressed here, are cast in the LTI framework wherever possible.

- This motivates the importance of the DT LTI class, which we now come to examine.

State-space representation

Definition

$$\begin{cases} x(k) &= Ax(k-1)+bu(k-1) & \text{(state equation)} \\ y(k) &= cx(k)+du(k) & \text{(output equation)} \end{cases}$$

- $u(k)$ and $y(k)$ are real scalars;
- $x(k) \in \Re^n$, where n is the system's order;
- the real dynamic matrix A is $n \times n$;
- b is a real column vector $(n \times 1)$;
- c is a real row vector $(1 \times n)$;
- d is a real scalar.

State-space representation

Motion

- Given $x(0)$ and $u(k)$, $k \geq 0$, we get

$$\begin{array}{rcll} x(1) &=& Ax(0)+bu(0) & \\ x(2) &=& Ax(1)+bu(1) &= A^2x(0)+Abu(0)+bu(1) \\ x(3) &=& Ax(2)+bu(2) &= A^3x(0)+A^2bu(0)+Abu(1)+bu(2) \\ & & \cdots & \end{array}$$

- This readily generalises to the Lagrange state formula

$$x(k) = A^k x(0) + \sum_{h=0}^{k-1} A^{k-h-1} bu(h).$$

- NOTE: there is an analogous – but not identical – formula in the CT case, that we do not treat for brevity.

State-space representation

Free and induced state motion

- In LTI systems, the state motion $x(k)$ is the sum of a free motion $x_F(k)$ and an induced motion $x_I(k)$, i.e.,

$$x(k) = x_F(k) + x_I(k),$$

where

$$x_F(k) = A^k x(0), \quad x_I(k) = \sum_{h=0}^{k-1} A^{k-h-1} b u(h).$$

- $x_F(k)$ depends linearly on $x(0)$ and not on u,
- while $x_I(k)$ depends linearly only on $u(k)$ and not on $x(0)$.

State-space representation

Free and induced output motion

- In LTI systems, also the output motion $y(k)$ is the sum of a free motion $y_F(k)$ and an induced motion $y_I(k)$, i.e.,

$$y(k) = y_F(k) + y_I(k),$$

where

$$y_F(k) = cA^k x(0), \quad y_I(k) = c\sum_{h=0}^{k-1} A^{k-h-1} b u(h) + du(k).$$

- again, $y_F(k)$ depends linearly on $x(0)$ and not on u,
- while $y_I(k)$ depends linearly only on $u(k)$ and not on $x(0)$.

State-space representation

Equilibrium

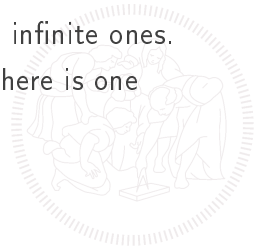
- For an LTI system, equilibrium states are found by solving

$$\bar{x} = A\bar{x} + b\bar{u}.$$

- Thus, if A has no unity eigenvalues, there exists the one equilibrium

$$\bar{x} = (I - A)^{-1} b\bar{u},$$

- while in the opposite case, either there is no equilibrium, or there are infinite ones.
- NOTE: in the CT case we have $0 = A\bar{x} + b\bar{u}$, hence $\bar{x} = A^{-1}b\bar{u}$, and there is one equilibrium if A has no *zero* eigenvalues.



State-space representation

Equilibrium – peculiarities of L(TI) systems

- They cannot have a finite number of equilibria, different from zero and one; note that this is not true for nonlinear systems.
- For each equilibrium state, there surely exists the one equilibrium output

$$\bar{y} = c\bar{x} + d\bar{u}.$$

State-space representation

Stability of an equilibrium

- Let $(\bar{x}, \bar{u})$ be an equilibrium for an LTI system.
- The Lagrange state formula leads to write

$$\bar{x} = A^k \bar{x} + \sum_{h=0}^{k-1} A^{k-h-1} b \bar{u}.$$

.

- Consider now the perturbed motion $x_\Delta(k)$ produced by $\bar{u}$ and $x(0) = \bar{x} + \Delta\bar{x}$; the same formula yields

$$x_\Delta(k) = A^k(\bar{x} + \Delta\bar{x}) + \sum_{h=0}^{k-1} A^{k-h-1} b \bar{u}.$$

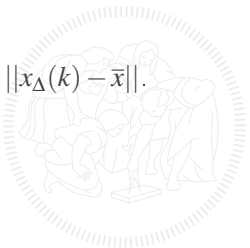
- Subtracting, therefore,

$$x_\Delta(k) - \bar{x} = A^k \Delta\bar{x}$$

State-space representation

Stability of an equilibrium

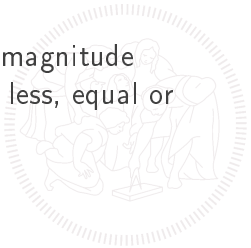
- The way $x_\Delta(k)$ moves with respect to $\bar{x}$ does not depend on $\bar{x}$.
- That is, contrary again to the nonlinear case, there cannot be equilibria with different stability characteristics.
- In the L(TI) class stability is a property of the system, not of the individual equilibria.
- Moreover,
  - for $k \to \infty$, $||x_\Delta(k) - \bar{x}|| \to 0 \forall x(0)$ iff A^k converges to a zero matrix,
  - the same norm generally diverges if at least one element of A^k does,
  - and if A^k neither converges to zero nor diverges, the same happens to $||x_\Delta(k) - \bar{x}||$.
- Thus, in the LTI case, system stability only depends on matrix A.



State-space representation

Stability and eigenvalues of A

- The following can be proven.
  - An LTI system is asymptotically stable iff all the eigenvalues of A have magnitude less than one (or, equivalently, lie in the open unit circle of the complex plane).
  - The same system is unstable if (but not only if) at least one eigenvalue of A has magnitude greater than one.
  - If all the eigenvalues of A have magnitude less than or equal to one, and there exists at least one with unity magnitude, the system can be either unstable or stable, but not asymptotically.
- NOTE: there is an analogous result for the CT case, where however "magnitude less, equal or greater than one" are respectively replaced by "real part less, equal or greater than zero".



State-space representation

Properties of asymptotically stable systems

- An asymptotically stable system has one and only one equilibrium for each constant input.
- The state and output free motions of an asymptotically stable system converge to zero (norm) for $k \to \infty$.
- As a consequence, asymptotically stable systems "forget their initial condition"...
- ...which is a definitely desired property for a controlled system.

The $\mathscr{Z}$ transform

Definition and main properties

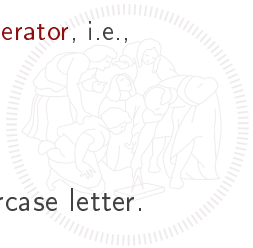
- Consider a real discrete-time signal $v(k)$, defined for $k \geq 0$ (or, equivalently, null for $k < 0$).
- Its $\mathscr{Z}$ transform is defined as

$$V(z) = \mathscr{Z}[v(k)] := \sum_{k=0}^{\infty} v(k) z^{-k}, \qquad z \in \mathbb{C}.$$

- Properties:
  - the $\mathscr{Z}$ transform is a linear operator (obvious);
  - The complex variable z can be interpreted as the one-step advance operator, i.e.,

$$\mathscr{Z}[v(k+1)] = z\mathscr{Z}[v(k)] - zv(0).$$

- We denote the $\mathscr{Z}$ transform of a signal with the corresponding uppercase letter.



The $\mathscr{Z}$ transform

Proof of the "one-step advance" property

- Expressing $\mathscr{Z}[v(k+1)]$ and then adding and subtracting $zv(0)$, we get

$$\begin{aligned}
\mathscr{Z}[v(k+1)] &= \sum_{k=0}^{\infty} v(k+1) z^{-k} \\
&= v(1) + v(2)z^{-1} + v(3)z^{-2} \ldots + zv(0) - zv(0) \\
&= z\left(v(0) + v(1)z^{-1} + v(2)z^{-2} \ldots\right) - zv(0) \\
&= z \sum_{k=0}^{\infty} v(k) z^{-k} - zv(0) \\
&= z\mathscr{Z}[v(k)] - zv(0).
\end{aligned}$$

- Thus, anticipating a signal by one step in the domain of the discrete time k...
- ...corresponds, if the signal is zero in $k = 0$, to just multiplying its transform by z.

The $\mathscr{Z}$ transform

The one-step advance and delay operators

- As z can be seen as the one-step advance'operator,
- in a similar way it can be proven that

$$\mathscr{Z}[v(k-1)] = z^{-1}\mathscr{Z}[v(k)].$$

- Hence, z^{-1} is called the one-step delay operator.

Input-output representation

Applying the $\mathscr{Z}$ transform

- Consider an LTI (DT SISO) system, and apply the $\mathscr{Z}$ transform to both sides of the state and the output equation; in force of the shown properties, this gives

$$\begin{aligned} zX(z) - zx(0) &= AX(z) + bU(z) \\ Y(z) &= cX(z) + dU(z) \end{aligned}$$

- Hence,

$$Y(z) = c(zI - A)^{-1}zx(0) + \big(c(zI - A)^{-1}b + d\big)U(z).$$

- Note that the two right hand side terms are the $\mathscr{Z}$ transforms of the free and the induced motion of y.

Input-output representation

The transfer function

- The ratio $Y_I(z)/U(z)$ does not depend on the input signal, but only on the system matrices.
- Thus, we define the complex function

$$G(z) = c(zI - A)^{-1}b + d$$

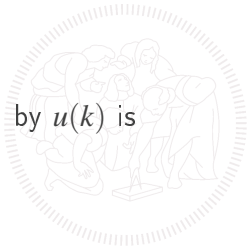
of the complex variable z as the transfer function of the system described in the state space by (A, b, c, d).

- The system is represented in input-output form by writing

$$G(z) = \frac{Y(z)}{U(z)}.$$

- This means that its transfer function is $G(z)$, and the motion induced by $u(k)$ is

$$y(k) = \mathscr{Z}^{-1}\left[G(z)\mathscr{Z}[u(k)]\right].$$



Input-output representation

Exercise 1 – From state space to transfer function

- Given the DT LTI SISO system represented in state space form by

$$A = \begin{bmatrix} 0.2 & 0 \\ 1 & 0.5 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, \; c = \begin{bmatrix} 1 & 1 \end{bmatrix}, \; d = 0,$$

compute its transfer function.

- Solution:

$$G(z) = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \left(z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.2 & 0 \\ 1 & 0.5 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} + 0$$

$$= \cdots = \frac{1.5z - 0.4}{(z - 0.2)(z - 0.5)}.$$

Input-output representation

Exercise 2 – From state space to transfer function

- Given the DT LTI SISO system represented in state space form by

$$A = \begin{bmatrix} 0.2 & 0.1 \\ 0 & 0.1 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, c = \begin{bmatrix} 0 & 1 \end{bmatrix}, d = 2,$$

  compute its transfer function.
- Solution:

$$\begin{aligned} G(z) = & \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \left(z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.2 & 0.1 \\ 0 & 0.1 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 2 \\ = & \cdots = \frac{2z - 0.8}{z - 0.1}. \end{aligned}$$

Input-output representation

Properties of the transfer function (1/2)

- It is the ratio of two polynomials;
- the roots of the numerator are called zeroes,
- those of the denominator poles;
- the denominator is the characteristic polynomial of A;
- the degree of the numerator can at most equal that of the denominator,
- and the two are equal iff $d \neq 0$.

Input-output representation

Properties of the transfer function (2/2)

- The poles of $G(z)$ are eigenvalues of A;
- however in computing $G(z)$ cancellations can occur, see exercise 2;
- in this case the system has hidden parts,
- i.e., something happens on the state that is not influenced by the input,
- or not visible on the output,
- or both.

Input-output representation

System stability and transfer function poles

- In the absence of cancellations, we can study stability based on the poles of $G(z)$;
- if a cancelled eigenvalue has magnitude less than one, the cancellation is non-critical,
- and we can still use the poles to assess stability.
- if a cancelled eigenvalue has magnitude one or more, the cancellation is critical,
- and stability needs discussing in the state space.

Input-output representation

A final remark on equilibrium

- Consider an asymptotically stable system described in the state space by (A,b,c,d), and apply the constant input $u(k)=\bar{u}$.
- We already know that the corresponding equilibrium state is

$$\bar{x}=(I-A)^{-1}b\bar{u}.$$

- Since the system is asymptotically stable, it will tend to this equilibrium at steady state, i.e., when the transient caused by $x(0)\neq\bar{x}$ is exhausted.
- Thus, the steady-state output will be

$$\bar{y}=c\bar{x}+d\bar{u}=c\big((I-A)^{-1}b+d\big)\bar{u}=G(1)\bar{u}.$$

- That is why $G(1)$ is called the system (static) gain.

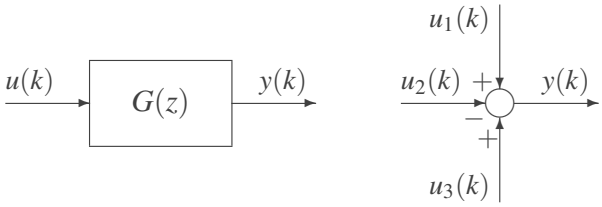
Block diagrams

Preliminaries

- Block diagrams (BDs) are a graphical formalism to represent dynamic systems, that we see here limited to the DT LTI class.
- They are useful to study interconnected systems, i.e., compounds of subsystems (e.g., a controller and the controlled object).
- Important *caveat*, that we state right from the beginning:
  - a BD MUST NOT BE CONFUSED with a flow diagram; although subsystems have inputs and outputs, their compound comes from assembling equations.

Block diagram components

Block and summation node



- Block (left):
  an LTI SISO system with the indicated transfer function – in the shown example, the equation $G(z)U(z) - Y(z) = 0$.
  Note: we use to write $G(z)u(k) - y(k) = 0$ with the same meaning.
- Summation node (right):
  a summation expression – in the shown example, the equation $u_1(k) - u_2(k) + u_3(k) - y(k) = 0$.

Properties of interconnected systems

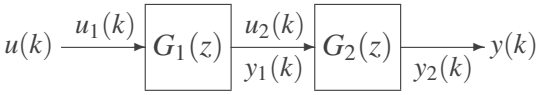
Questions

- A BD can represent a multivariable system, with many inputs and outputs.
- Since we associate blocks to transfer functions, we are assuming they have no hidden parts.
- We need however to answer some questions.
  - How do I compute the transfer function from a certain input u_i to a certain output y_j?
  - Assuming the stability properties of individual blocks to be known, what about the same properties for the compound system?
  - The blocks do not have hidden parts: can the compound system conversely have some?
- To this end, we study the three main BD manipulation (or simplification) operations.

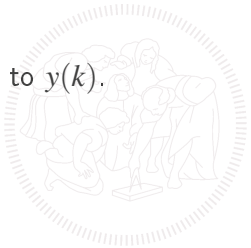
Block diagram manipulation

Series connection

- Two blocks are in series (or cascade) when the output of the first is the input of the second:

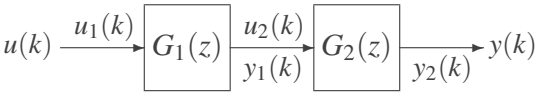


- Generalising to more than two blocks is obvious.
- We want to compute the compound transfer function $G(z)$ from $u(k)$ to $y(k)$.



Block diagram manipulation

Series connection



- Equations:

$$\begin{cases} y_1(k) &= G_1(z)u_1(k) \\ y_2(k) &= G_2(z)u_2(k) \\ u_2(k) &= y_1(k) \\ u(k) &= u_1(k) \\ y(k) &= y_2(k) \end{cases}$$

- Note the block constitutive and the connection equations.
- Solving, we get

$$y(k) = G_2(z)G_1(z)u(k) \quad \Rightarrow \quad G(z) = G_2(z)G_1(z).$$

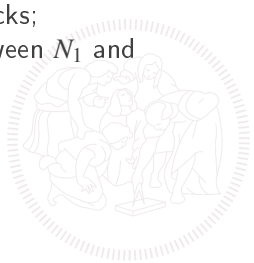
Block diagram manipulation

Series connection – properties of the compound

- Evidencing the numerator and denominator polynomials we have

$$G_1(z) = \frac{N_1(z)}{D_1(z)},\ G_2(z) = \frac{N_2(z)}{D_2(z)} \Rightarrow G(z) = \frac{N_1(z)N_2(z)}{D_1(z)D_2(z)}.$$

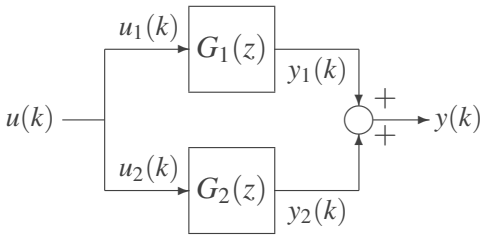
- Hence
  - the compound is (asymptotically) stable iff so are the component blocks;
  - the compound can have hidden parts in the case of cancellations between N_1 and D_2, or N_2 and D_1.



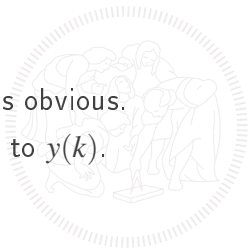
Block diagram manipulation

Parallel connection

- Two blocks are in parallel when their inputs are all equal and their outputs are summed together:

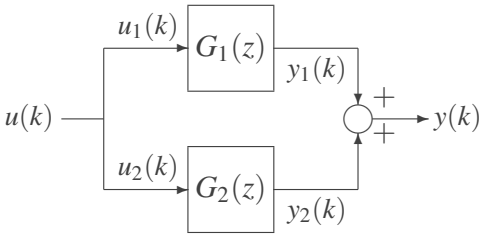


- Generalising to more than two blocks and different summation signs is obvious.
- We want to compute the compound transfer function $G(z)$ from $u(k)$ to $y(k)$.



Block diagram manipulation

Parallel connection

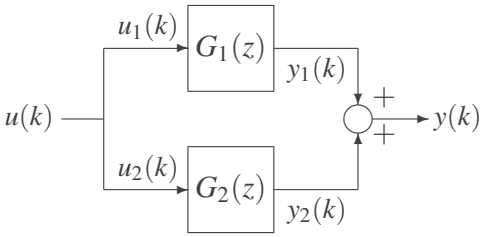


- Equations (same colour coding):

$$\begin{cases} y_1(k) &= G_1(z)u_1(k) \\ y_2(k) &= G_2(z)u_2(k) \\ u_1(k) &= u(k) \\ u_2(k) &= u(k) \\ y(k) &= y_1(k)+y_2(k) \end{cases}$$

Block diagram manipulation

Parallel connection



- Solving, we get

$$y(k) = (G_1(z)+G_2(z))u(k) \quad \Rightarrow \quad G(z) = G_1(z)+G_2(z).$$

Block diagram manipulation

Parallel connection – properties of the compound

- Evidencing the numerator and denominator polynomials we have in this case

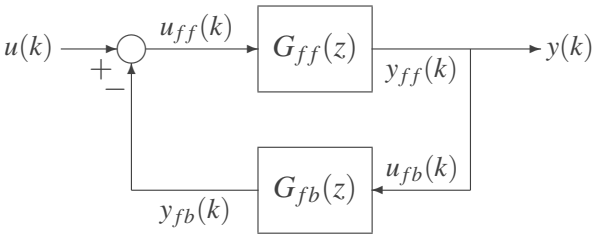
$$G(z) = \frac{N_1(z)D_2(z) + N_2(z)D_1(z)}{D_1(z)D_2(z)}.$$

- Hence
  - the compound is (asymptotically) stable iff so are the component blocks;
  - the compound can have hidden parts in the case of cancellations between the numerator and the denominator of $G(z)$;
  - a notable case is when D_1 and D_2 have a common factor.

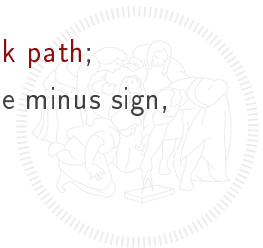
Block diagram manipulation

Feedback (loop) connection

- The two blocks below form a feedback loop:

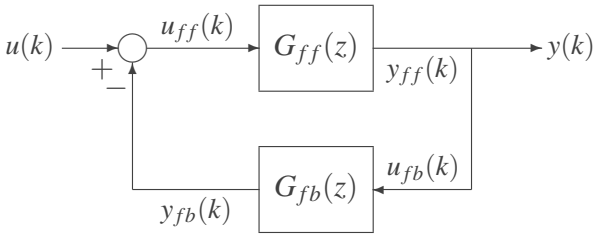


- $G_{ff}(z)$ and $G_{fb}(z)$ are respectively the forward path and the feedback path;
- the product of all transfer functions around the loop, disregarding the minus sign, is the (open) loop transfer function;
- this is typically indicated with $L(z)$, and equals here $G_{ff}(z)G_{fb}(z)$.



Block diagram manipulation

Feedback (loop) connection

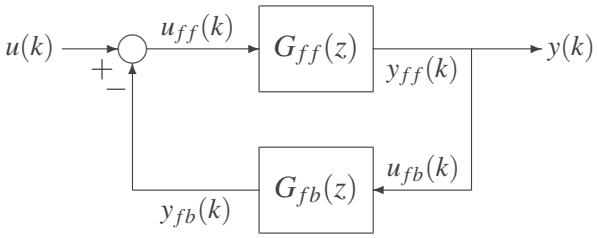


- Equations (same colour coding):

$$\begin{cases} y_{ff}(k) &= G_{ff}(z)u_{ff}(k) \\ y_{fb}(k) &= G_{fb}(z)u_{fb}(k) \\ u_{ff}(k) &= u(k) - y_{fb}(k) \\ y(k) &= y_{ff}(k) \\ u_{fb}(k) &= y_{ff}(k) \end{cases}$$

Block diagram manipulation

Feedback (loop) connection



- Solving, we get

$$G(z) = \frac{G_{ff}(z)}{1 + G_{ff}(z)G_{fb}(z)}.$$

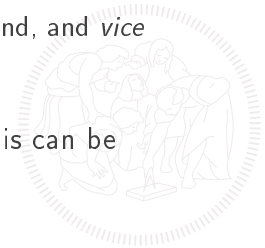
Block diagram manipulation

Feedback (loop) connection – properties of the compound

- Evidencing the numerator and denominator polynomials we here obtain

$$G(z) = \frac{N_1(z)N_2(z)}{N_1(z)N_2(z) + D_1(z)D_2(z)}.$$

- Hence
  - there is no immediate relationship between the poles of the components and those of the compound;
  - the compound can have hidden parts (we omit details);
  - but most important, unstable components can yield a stable compound, and *vice versa*.



- Feedback is extremely powerful. In the following we shall see how this can be exploited for control.

Multi-physics dynamic modelling and simulation

Dynamic modelling and simulation

- **What**
  - Creating a dynamic system to represent some phenomenon of interest, evidencing inputs, states and outputs;
  - turning that model into a program, and execute that program to make the modelled system evolve.
- **Why**
  - To gain insight into the system — e.g., what influences what, to what extent, and on what time scale;
  - to forecast the future behaviour of the system;
  - to size system components;
  - to set up controls;
  - to co-design the system and its control.

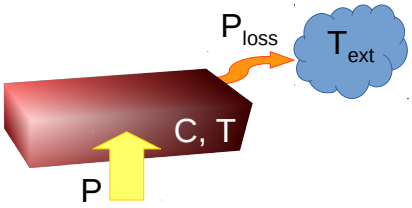
Multi-physics

- **What**
  - The distinctive character of a model whose equations pertain to different domains — e.g., electrical, mechanical, thermal, hydraulic, and so forth.
- **Why**
  - Because many interesting problems involve models with this character; and nowadays we have tools that natively support such models, thus allowing to treat the above problems in a unitary manner.

Case 1 – no multi-physics yet, just thermal (and CT)

"to gain insight into the system and forecast its behaviour"

- Solid plate with exogenous thermal power input and loss

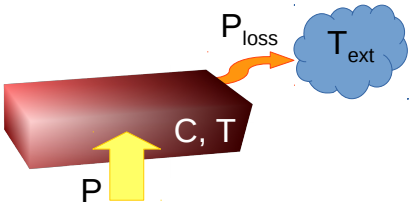


- Exogenous inputs:
  generated power P, external temperature T_{ext}.
- State:
  plate temperature T.
- Parameters:
  heat capacity $C = c\rho V$ where c is specific heat, ρ density and V volume;
  loss conductance $G = \gamma S$, where γ is convective coefficient and S surface.

Case 1

Model

- Dynamic energy balance — derivative of energy is sum of powers.



- Equations (dot means derivative with time, i.e., $\dot{x} := dx/dt$):

$$\begin{aligned} E &= CT \\ C &= c\rho V \\ \dot{E} &= P - P_{loss} \\ P_{loss} &= G(T - T_{ext}) \\ G &= \gamma S \end{aligned}$$

- Let us now write this in Modelica.

Case 1

Model Day1.Case_01 in the course Modelica package (subsequent listings not on slides for brevity)

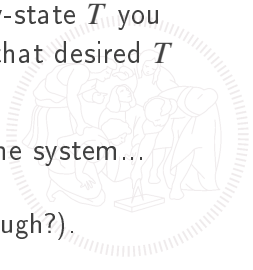
```
model Case_01 "Solid plate with exogenous thermal power input and loss"
  parameter Modelica.SIunits.Length                    L      = 0.01 "length";
  parameter Modelica.SIunits.Length                    W      = 0.01 "width";
  parameter Modelica.SIunits.Length                    H      = 0.001 "height";
  parameter Modelica.SIunits.Density                   ro     = 2300;
  parameter Modelica.SIunits.SpecificHeatCapacity      c      = 700;
  parameter Modelica.SIunits.CoefficientOfHeatTransfer gamma  = 6;
  parameter Modelica.SIunits.Temperature               Tstart = 273.15 + 20;

  final parameter Modelica.SIunits.HeatCapacity C = L*W*H*d*c;
  final parameter Modelica.SIunits.ThermalConductance G = (2*L*W+2*H*(L+W))*gamma;

  Modelica.SIunits.Temperature T(start = Tstart, fixed = true);
  Modelica.SIunits.Temperature Text;
  Modelica.SIunits.Power P, Ploss;

equation
  // Dynamic balance and expression of state derivatives
  C * der(T) = P-Ploss;
  Ploss      = G*(T-Text);
  // Exogenous inputs (just an example with pulsed power and constant external T)
  P          = if sin(2*Modelica.Constants.pi/30 * time) > 0.95 then 1 else 0;
  Text       = 273.15+20;
end Case_01;
```


Play around with the parameters and change the inputs' shape...

Case 1

Some considerations

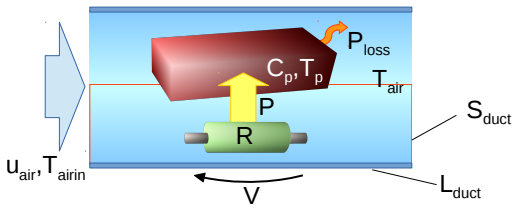
- Try a step power with constant T_{ext} (step responses in some sense portray a system, more on this when talking about control). What can we conclude?
- G influences the steady-state variation of T: lower G, better thermal insulation, need to raise T higher to release a given P to the exterior;
- C has no effect on the steady-state T, but together with G influences the time to reach it: more capacity, need to input more energy to reach the final T, and with the same power P this takes longer.
- Want it faster? Better insulated? Is P enough to maintain the steady-state T you desire in the face of expectable values of T_{ext}? Is P enough to reach that desired T within a given deadline?
- This is gaining insight, and as a consequence information for sizing the system...
  - both statically (is this number OK to keep a desired condition?)
  - and dynamically (is this OK to reach a new desired condition fast enough?).



Case 2 – multi-physics (thermal/electrical/aeraulic), still CT

"to size components"

- Solid plate with electric heating and air cooling

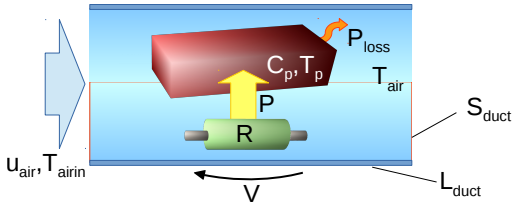


- Exogenous inputs:
  voltage V, inlet air velocity u_{air} and temperature T_{airin}.
- States:
  plate temperature T_p, air temperature T_{air}.
- Parameters:
  see the model on next slide.

Case 2

Model — don't panic, just to show how we SC guys reason in such cases

- Solid plate with electric heating and air cooling



- Equations:

$$
\begin{aligned}
c_p \rho_p V_p \dot{T}_p &= P - P_{loss} \\
c_{air} V_{air} \dot{T}_{air} &= c_{air} S_{duct} u_{air} (T_{airin} - T_{air}) + P_{loss}/\rho_{air} \\
P &= V^2/R \\
P_{loss} &= \gamma S_p (T_p - T_{air}) \\
\gamma &= \gamma_0 + (\gamma_{ref} - \gamma_0)(u_{air}/u_{airref})^{0.8}
\end{aligned}
$$

- Let us see this in Modelica (model Day1.Case_02 in the course package).

Case 2

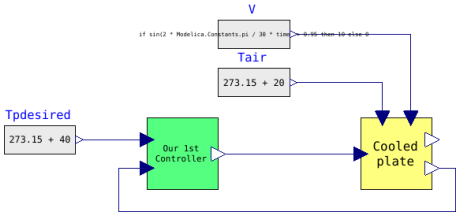
Some considerations

- Try various values for R, apply constant and pulsed V, change u_{air} and T_{airin}.
- Which air velocity do I need to keep an acceptable T? Which is the air outlet temperature I have to consequently accept? How fast do I need to vary u_{air} to shave the temperature peaks provoked by a pulsed power?
- This is sizing system components (e.g., a cooling fan and duct) and understanding the operational limits of their compound.
- ...wait, this is all physical; and the cyber stuff?

Case 3 – CT controlled system, DT controller

"to set up controls, and possibly co-design the system and its control"

- Solid plate with electric heating and temperature control via cooling air (fan) speed



- Models Day1.CooledPlate, Day1.OurFirstController and Day1.Case_03 in the course package.
- Controller, DT at fixed timestep (just the most relevant lines);

```
algorithm
  when sample(0,timestep) then
    uair := max(0,min(uairmax,K*(Tpdesired-Tpactual)));
  end when;
```

Case 3

Some considerations

- Try various values for the controller gain K (negative, as to make the temperature T go down the air velocity u_{air} must go up);
- What are the limits? Would you try a larger fan (larger u_{airmax})? Would you add a heat sink (as an approximation for that, here you can increase the γ coefficients)?
- This is setting up control to match a given goal in the face of the ambient conditions deemed reasonable to expect, and sometimes acknowledging that to make the goal feasible, the controlled system itself needs modifying.
- We can make a crappy system look less crappy, not fantastic: we make controls, not miracles :-)

Remark

- We just mixed CT and DT system (controlled object and controller, respectively) in the same model.
- The CT part was described as differential equations, the DT part as an algorithm to be executed periodically.
- This is OK for simulating, but can we analyse such a model?
- The question trespasses the boundary of this course...
  ...but nonetheless, the essentials on how to relate CT and DT are useful to know.
- Let us therefore ask ourselves a question.

A question about state space and transfer function form

limiting the scope to the LTI SISO case

DT system $\begin{cases} x(k) &= Ax(k-1)+bu(k-1) \\ y(k) &= cx(k)+du(k) \end{cases}$ $\xrightarrow{\mathscr{Z}\text{transform}}$ $G(z)=c(zI-A)^{-1}b+d$

$\Downarrow$

Block diagrams

CT system $\begin{cases} \dot{x}(k) &= Ax(t)+bu(t) \\ y(k) &= cx(t)+du(t) \end{cases}$ $\Longrightarrow$?

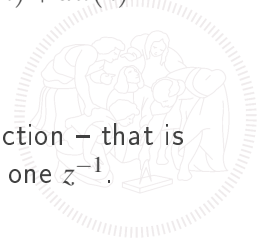
- It would be nice to have block diagrams with CT blocks as well...
  ...but do we have a transfer function also for CT systems?
- Yes we do. Rigorously we should go through the Laplace transform, but for our purposes let us be more operational.

Revisiting the DT (LTI SISO) case

- What gives the system its dynamic character?
- The one-step advance in the state equation ($x(k)$ depending on x and u at $k-1$).
- All right, let us introduce the one-step advance operator and call it z, i.e., set by definition $zx(k) := x(k+1)$.
- Rewriting the system with this operator we have

$$\begin{cases} zx(k) &= Ax(k)+bu(k) \\ y(k) &= cx(k)+du(k) \end{cases} \Rightarrow \begin{aligned} (zI-A)x(k) &= bu(k) \\ y(k) &= c(zI-A)^{-1}bu(k)+du(k) \end{aligned}$$

  and voilà, we get $G(z)$.
- Reasoning like this, we see a system as an operator – its transfer function – that is constructed using the elementary advance operator z — or the delay one z^{-1}.

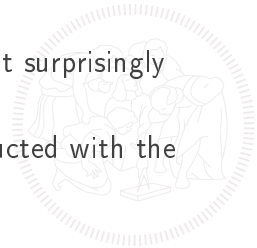


And now, for the CT case

- What gives the system its dynamic character?
- The derivative in the state equation ($\dot{x}(t)$ depending on x and u).
- All right, let us introduce the time derivative operator and call it s, i.e., set by definition $sx(t) := \dot{x}(t)$.
- Rewriting with this operator we have

$$\begin{cases} sx(t) &= Ax(t) + bu(t) \\ y(t) &= cx(t) + du(t) \end{cases}$$

  and by expressing $y(t)/u(t)$ we get the CT transfer function $G(s)$, not surprisingly with the same form as in the DT case, i.e., $G(s) = c(sI - A)^{-1}b + d$.
- Here we see a system as an operator – its transfer function – constructed with the elementary derivative operator s — or the integral one $1/s$.



And block diagrams?

- Identical to the CT case:
  - series of $G_1(s)$ and $G_2(s)$

$$G(s) = G_2(s)G_1(s),$$

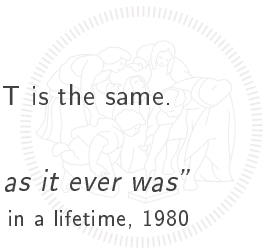
  - parallel of $G_1(s)$ and $G_2(s)$

$$G(s) = G_1(s) + G_2(s),$$

  - loop with negative feedback, forward path $G_1(s)$ and feedback path $G_2(s)$

$$G(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)},$$

  - for summation nodes (and non dynamic systems in general) CT or DT is the same.



"Same as it ever was"
Talking Heads, Once in a lifetime, 1980

Case 4

revisiting case 1 as block diagram

- Model (compacted):

$$C\dot{T} = P - G(T - T_{ext}).$$

- Bring in the derivative operator s and solve for the output:

$$sCT = P - G(T - T_{ext}),$$

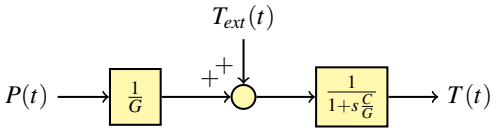
$$(sC + G)T = P + GT_{ext},$$

$$T = \frac{1}{1 + s\frac{C}{G}}\left(\frac{1}{G}P + T_{ext}\right).$$

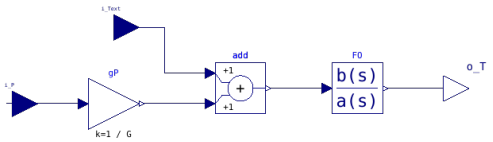
Case 4

revisiting case 1 as block diagram

- Block diagram:



- Modelica diagram (Day1.Plate_P_Ploss_as_BD in the course package):



- You can play around with a simple loop involving this system (Day1.Case_04).
- Now, something more relaxing (and back to DT).

Case 5 – DT block diagrams

Block diagram simplifications

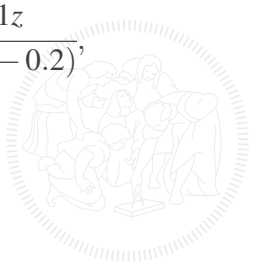
- Let us verify the series, parallel and feedback connections, by comparing the unit step response of the individual blocks' compound and that of the equivalent, single block.
- We shall use

$$G_1(z) = \frac{1}{z-0.5}, \quad G_2(z) = \frac{z-0.4}{z-0.2},$$

so that

$$G_1(z)G_2(z) = \frac{z-0.4}{(z-0.5)(z-0.2)}, \quad G_1(z)+G_2(z) = \frac{z^2+0.1z}{(z-0.5)(z-0.2)},$$

$$\frac{G_1(z)}{1+G_1(z)G_2(z)} = \frac{z-0.2}{z^2+0.3z+0.3}.$$



This morning

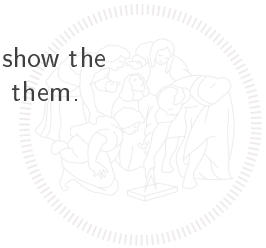
Takeaways

- Dynamic system: reacts to input depending on state, remembers the past...
- Treatise centred on the DT LTI case (the most interesting for CSE-related problems).
- State space representation, stability.
- Transfer function and its relationship with state space (hidden parts).
- Interconnected systems, block diagrams, and the power of feedback.

This afternoon

Takeaways

- Dynamic modelling and simulation — easier to introduce in the CT case.
- Multi-physics (Modelica as host environment for the examples).
- Model analysis and simulation as a means to design controls, and sometimes systems.
- Something more on the CT (LTI) case
  - for information completeness,
  - because the first control examples come easier,
  - because later on it will be useful to have loops CS as counterpart to show the peculiarities of CSE-related problems, thereby motivating research on them.



Tomorrow

Main topics and goals

- Control design.
- In the morning, back to DT for the theoretical treatise.
- In the afternoon practice and still mostly DT, however with some (more) words on how DT and CT relate.

- Objective for the next days: follow and understand descriptions of real-world cases, then address simple – yet realistic – examples completely.

– Day 2 –

Outline

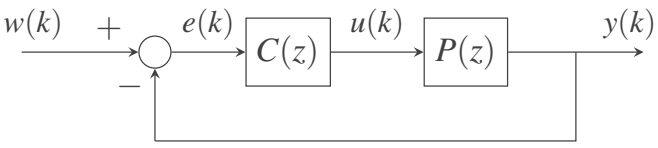
- **Morning**: *the magic of feedback (how we use our main tool)*
  - Preliminaries;
  - the control loop and its actors;
  - formal requirements;
  - control synthesis;
  - simulation examples;
  - from controller model to algorithm.
- **Afternoon**: *hands-on practice*
  - Modelling and simulating simple loops in Modelica;
  - synthesising and assessing feedback controllers;
  - something about the CT–DT relationship (discretisation).

The magic of feedback

Preliminaries

The big picture

- This is the simplest feedback control loop:



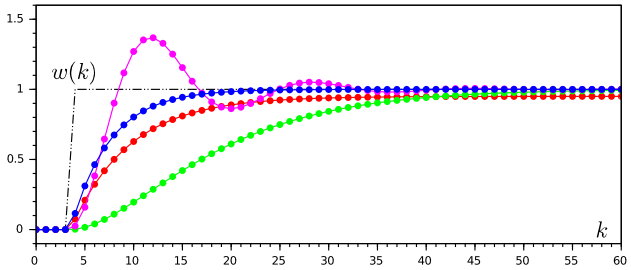
- Notation (more later on):
  - $w(k)$ is the set point or reference;
  - $y(k)$ is the controlled variable;
  - $e(k)$ is the error;
  - $u(k)$ is the control signal.

- Objective: make $y(k)$ follow $w(k)$ as closely as possible.

Objectives

Time versus $\mathscr{Z}$ transform domain

- Specifications are naturally expressed in the time domain;
- suppose we apply a step variation to $w(k)$, and consider four possible responses of $y(k)$:



- The control quality is apparently different:
  - blue – good;
  - red – does not reach the new value exactly;
  - green – too slow;
  - magenta – oscillatory.

Objectives

Time versus $\mathscr{Z}$ transform domain

- To determine the controller, it is very simple to operate with transfer functions.
- Thus, we need to turn control quality specifications in the time domain into a desired transfer function from $w(k)$ to $y(k)$.
- To this end, we now relate a transfer function to its *time responses*.
- Note: since we shall make any control system asymptotically stable, free motion vanishes for $k \to \infty$.
- This is why we can use transfer functions, which only determine induced motion.
- Let us proceed.

Time responses

From $G(z)$ to $u(k) \mapsto y(k)$

- Consider a generic proper transfer function and write it in the form

$$G(z) = \frac{b_m z^m + b_{m-1} z^{m-1} \cdots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} \cdots + a_1 z + a_0}, \quad m < n.$$

- Indicating the system input and output with $u(k)$ and $y(k)$, we have

$$\frac{b_m z^m + b_{m-1} z^{m-1} \cdots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} \cdots + a_1 z + a_0} = \frac{Y(z)}{U(z)}.$$

Time responses

From $G(z)$ to $u(k) \mapsto y(k)$

- Hence,

$$\big(z^n + a_{n-1} z^{n-1} \cdots + a_1 z + a_0\big) Y(z) = \big(b_m z^m + b_{m-1} z^{m-1} \cdots + b_1 z + b_0\big) U(z).$$

- Now, recall that we are treating induced motion, thus we assume zero initial conditions, and

$$\begin{aligned} &\mathscr{Z}\big[v(k+1)\big] = zV(z) \\ &\mathscr{Z}\big[v(k+2)\big] = z^2 V(z) \\ &\dots \end{aligned} \qquad \Rightarrow \mathscr{Z}\big[v(k+h)\big] = z^h V(z).$$

Time responses

From $G(z)$ to $u(k) \mapsto y(k)$

- Exploiting the last relationship we get

$$y(k+n)+a_{n-1}y(k+n-1)\cdots+a_1y(k+1)+a_0y(k)= \\ b_mu(k+m)+b_{m-1}u(k+m-1)\cdots+b_1u(k+1)+b_0u(k).$$

- We can now solve for the most recent output, and since the system is TI, shift time indices so that this be $y(k)$.

- Better show this with an example.

From $G(z)$ to $u(k) \mapsto y(k)$

Example – proper case

- Let the system transfer function be

$$G(z)=\frac{z-0.8}{(z-0.5)(z-0.2)}.$$

- We have

$$\frac{Y(z)}{U(z)}=\frac{z-0.8}{z^2-0.7z+0.1},$$

- thus, anti-transforming,

$$y(k+2)-0.7y(k+1)+0.1y(k)=u(k+1)-0.8u(k),$$

- whence, solving for the most recent output and shifting,

$$y(k)=0.7y(k-1)-0.1y(k-2)+u(k-1)-0.8u(k-2).$$

Time responses

From $G(z)$ to $u(k) \mapsto y(k)$

- If $G(z)$ is not proper, the numerator and denominator degrees are equal.
- We can thus write $G(z)$ as a constant plus a proper transfer function, i.e.,

$$G(z) = \frac{N(z)}{D(z)} = k + \frac{\widetilde{N}(z)}{D(z)},$$

  where if the degrees of $N(z)$ and $D(z)$ are both n, that of $\widetilde{N}(z)$ is at most $n-1$.
- Once this is done, we are back to the proper case.

- Here too, let us see an example.

From $G(z)$ to $u(k) \mapsto y(k)$

Example – not proper case

- Let the system transfer function be

$$G(z) = \frac{z^2 - 0.8}{(z-0.5)(z-0.2)}.$$

- We have

$$\frac{Y(z)}{U(z)} = 1 + \frac{0.7z - 0.9}{z^2 - 0.7z + 0.1}.$$

- This can be interpreted (question: why x?) as

$$Y(z) = X(z) + U(z); \quad \frac{X(z)}{U(z)} = \frac{0.7z - 0.9}{z^2 - 0.7z + 0.1}.$$

From $G(z)$ to $u(k) \mapsto y(k)$

Example – not proper case

- Solving and shifting, we get

$$\begin{cases} x(k) &= 0.7x(k-1) - 0.1x(k-2) + 0.7u(k-1) - 0.9u(k-2) \\ y(k) &= x(k) + u(k) \end{cases}$$

- BTW, answer: see which are the state variables, i.e., what you need other than $u(k)$ to know $y(k)$...

$$\begin{cases} x(k) &= 0.7x(k-1) - 0.1x(k-2) + 0.7u(k-1) - 0.9u(k-2) \\ y(k) &= x(k) + u(k) \end{cases}$$

Back to time responses

An intuitive means to assess control quality

- The response of y to a (unit) step on w allows to easily catch the quality of a control loop.
- Let us see a notable example, leading to a most desirable form for $Y(z)/W(z)$.
- Consider the transfer function

$$\frac{Y(z)}{W(z)} = \frac{1-p}{z-p}, \quad |p| < 1,$$

- that can be written in state space form as

$$\begin{cases} x(k) &= px(k-1) + (1-p)w(k-1) \\ y(k) &= x(k) \end{cases}$$

Time responses

An intuitive means to assess control quality

- The (induced) unit step response is obtained, from $k = 0$, with $x(0) = 0$ and $w(k) = 1$.
- Applying the Lagrange formula we get

$$\begin{aligned} y(k) = x(k) &= \sum_{h=0}^{k-1} A^{k-h-1} b w(h) \\ &= \sum_{h=0}^{k-1} p^{k-h-1}(1-p) \cdot 1 = 1 - p^k. \end{aligned}$$

- Let us verify:

$$\begin{aligned} &x(0) = 0 \Rightarrow y(0) = 0 \\ &x(1) = p \cdot 0 + 1 - p \Rightarrow y(1) = 1 - p \\ &x(2) = p(1-p) + 1 - p \Rightarrow y(2) = p - p^2 + 1 - p = 1 - p^2 \\ &x(3) = p(1-p^2) + 1 - p \Rightarrow y(3) = p - p^3 + 1 - p = 1 - p^3 \\ &\ldots \end{aligned}$$

Time responses

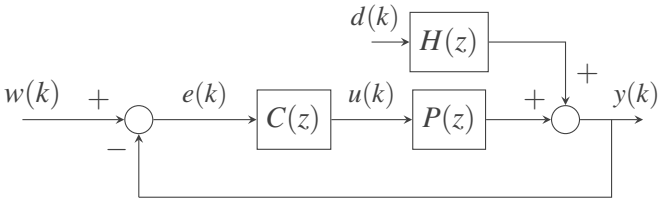
An intuitive means to assess control quality

- To avoid oscillations in the response, we need $0 < p < 1$.
- With such a choice, we ensure the following:
  - for $k \to \infty$, $y(k)$ tends to $w(k)$,
  - thus, after a transient, the error goes to zero;
  - parameter p controls the convergence time,
  - that increases if $p \Rightarrow 1^-$ and decreases if $p \Rightarrow 0^+$.
- Summarising, then,
  - one starts from "time domain" (and stability) requirements,
  - whence a viable way to determine $C(z)$ is to choose a desired $Y(z)/W(z)$ accordingly.

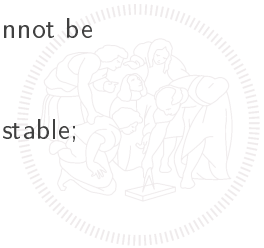
The control loop and its actors

A more detailed picture

- Let us see a more realistic loop representation:



- Additional notation:
  - $d(k)$ is a disturbance, that acts on the controlled variable $y(k)$ but cannot be manipulated (maybe, however, measured);
  - $P(z)$ and $H(z)$ compose the controlled system (or process) model;
  - note that $H(z)$ is outside the loop, thus we assume it asymptotically stable;
  - $C(z)$ is the controller, or regulator, transfer function.



The control loop and its actors

Main transfer functions of interest

- Loop:

$$L(z) := C(z)P(z).$$

- From set point to controlled variable:

$$T(z) := \frac{L(z)}{1+L(z)}.$$

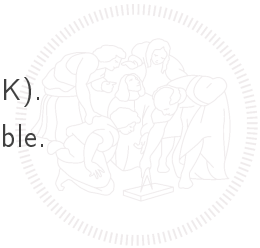
- From disturbance to controlled variable:

$$F(z) := \frac{H(z)}{1+L(z)}.$$

The control part of the loop

Functional view

- For the loop to operate
  - an event generator triggers a new control action (periodically or on demand);
  - a sensor is activated to get $y(k)$;
  - the controller $C(z)$ is invoked, and evolves as dynamic system by one step,
    - updating its state
    - and then computing the new control $u(k)$;
  - an actuator is activated to act on the process;
  - and finally the system waits for a new event.



- This view is quite easily related to CSE frameworks such as MAPE(-K).
- However, to design (not code) the controller, another view is preferable.

The control part of the loop

Systemic view – i.e., the SC standpoint

- For the loop to operate correctly
  - the closed-loop system must be asymptotically stable, which forbids critical cancellations between $C(z)$ and $P(z)$;
  - the reference-to-output transfer function $T(z)$, and/or the disturbance-to-output one $F(z)$, must yield "acceptable" time responses in the presence of expectable signals $w(k)$ and $d(k)$;
  - the controller must be realisable, i.e., the order of the numerator of $C(z)$ cannot exceed that of its denominator.

- This leads us to (further) discuss how requirements are stated, and how they reflect into desired closed-loop transfer functions.

Formal requirements

Just an add-on concerning asymptotic stability

- As per its definition, asymptotic stability is a state-space concept.
- However, we operate with transfer functions.
- A system could thus be asymptotically stable externally (no poles on or outside the circle) but not internally (at least one such eigenvalue, cancelled).
- To stress that we forbid this, we say to require – somehow redundantly – asymptotic internal stability.

Formal requirements

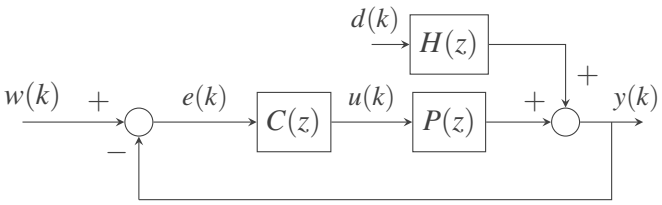
What we need to ensure (basic list)

- Asymptotic (internal) stability:
  asymptotic (external) stability and no critical cancellations.
- Static precision:
  at steady state y must be as close as possible to w.
- Dynamic precision:
  fast and not oscillatory responses.

- We now discuss the items above more in detail.

Asymptotic internal stability

The loop characteristic equation



- Let us set $H(z) = 1$ for simplicity and write, as usual,

$$C(z) = \frac{N_C(z)}{D_C(z)}, \quad P(z) = \frac{N_P(z)}{D_P(z)}.$$

Asymptotic internal stability

The loop characteristic equation

- This yields

$$\begin{aligned} T(z) &= \frac{C(z)P(z)}{1+C(z)P(z)} = \frac{N_C(z)N_P(z)}{N_C(z)N_P(z)+D_C(z)D_P(z)}, \\ F(z) &= \frac{1}{1+C(z)P(z)} = \frac{D_C(z)D_P(z)}{N_C(z)N_P(z)+D_C(z)D_P(z)}. \end{aligned}$$

- The closed-loop poles are thus the solutions of the characteristic equation

$$N_C(z)N_P(z)+D_C(z)D_P(z) = 0.$$

Asymptotic internal stability

Conditions

1. The roots of the characteristic equation must lie in the open unit circle.
2. There must be no critical cancellation between $C(z)$ and $P(z)$.

- In the following we shall see a cancellation-based synthesis technique...
- ...that is therefore applicable only to processes with neither poles nor zeroes on or outside the unit circle;
- for processes violating this we shall just say a few words...

  ..if not for a particular case, that in CSE-related control problems we encounter quite frequently.

Asymptotic internal stability

Example 1 – correct

- Consider

$$P(z) = \frac{0.5}{z-0.8}, \quad C(z) = \frac{z-0.8}{z-1}.$$

- This gives

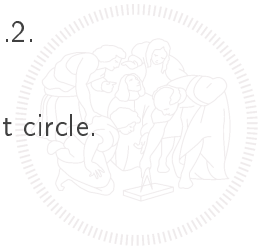
$$\begin{aligned} T(z) &= \frac{\frac{0.5}{z-0.8}\frac{z-0.8}{z-1}}{1+\frac{0.5}{z-0.8}\frac{z-0.8}{z-1}} = \frac{0.5(z-0.8)}{(z-0.8)(z-1)+0.5(z-0.8)} \\ &= \frac{0.5(z-0.8)}{(z-0.5)(z-0.8)} = \frac{0.5}{z-0.5}. \end{aligned}$$

- Note the (non critical) cancellation:
  - closed-loop poles $\Rightarrow$ 0.5;
  - closed-loop eigenvalues $\Rightarrow$ 0.5 and 0.8 (the cancelled one).
- GOOD, we have asymptotic internal stability.

Asymptotic internal stability

Example 2 – incorrect

- Consider
$$P(z) = \frac{0.5}{z-1.2}, \quad C(z) = \frac{z-1.2}{z-1}.$$
- Mind the process pole outside the circle. Here we have
$$T(z) = \cdots = \frac{0.5(z-1.2)}{(z-0.5)(z-1.2)} = \frac{0.5}{z-0.5}.$$
- The cancellation is critical; the closed-loop eigenvalues are 0.5 and 1.2.
- BAD, we have an unstable hidden part.
- Note: the same would happen if $P(z)$ had a zero not in the open unit circle.



Asymptotic internal stability

Example 3 – correct though $P(z)$ is unstable

- Consider
$$P(z) = \frac{0.5}{z-1.2}, \quad C(z) = 2.$$
- This time we have
$$T(z) = \frac{2\frac{0.5}{z-1.2}}{1+2\frac{0.5}{z-1.2}} = \frac{1}{z-1.2+1} = \frac{1}{z-0.2}.$$
- Clearly there can be no cancellation, as $C(z)$ is not dynamic.
- GOOD, we have asymptotic internal stability.

Static precision

Context

- One can impose various steady-state requirements.
- Also, one can consider different input signals for both reference and disturbance.

- Here we limit the scope to achieving zero steady-state error for a reference step.

Static precision

Condition

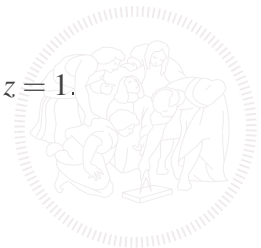
- To have zero steady-state error for a reference step we need the reference-to-output gain to be the unity, i.e.,

$$T(1) = 1.$$

- Simplifying a bit,

$$\frac{C(1)P(1)}{1+C(1)P(1)} = 1 \;\Rightarrow\; C(1)P(1) = \infty.$$

- It is thus required that the open-loop transfer function has a pole in $z = 1$.



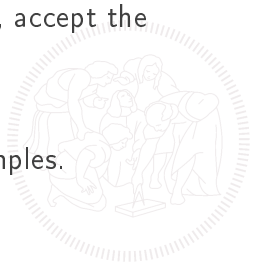
Dynamic precision

Quality of the response

- Here too, we have to limit our scope.
- A good way to have y respond "well" to a step on w is to aim (as anticipated) at

$$T(z) = \frac{1-p}{z-p}, \quad 0 < p < 1.$$

- Note that $T(1) = 1$, while p is the pole.
- If this is not possible (we shall shortly see why) one has to add poles, accept the presence of zeroes, or introduce a delay (a z^{-N} term).



- Better discuss these issues later on, when seeing some synthesis examples.

Control synthesis

Foreword

- There is a myriad of methods to synthesise $C(z)$.
- We here discuss just two quite simple techniques.
- In the end, however, we aim in fact at introducing two design paradigms, corresponding to the two major purposes of feedback control:
  - tracking the reference signal,
  - and rejecting disturbances.

Synthesis by transfer function assignment

Set point tracking

- Let $P(z)$ be the process under control, and $T^\circ(z)$ the desired closed-loop transfer function from set point to controlled variable.
- To find the controller, we solve for $C(z)$ the equation

$$\frac{C(z)P(z)}{1+C(z)P(z)} = T^\circ(z),$$

- obtaining

$$C(z) = \frac{1}{P(z)} \frac{T^\circ(z)}{1-T^\circ(z)}.$$

Synthesis for set point tracking

Limitations

- This technique *as is* has some applicability conditions:
  - $P(z)$ must have neither poles nor zeros on or outside the unit circle, otherwise we are generating a hidden part that is not asymptotically stable;
  - the relative degree (number of poles minus number of zeroes) of $T^\circ(z)$ must not be lower than that of $P(z)$, or we get an unrealisable controller (more zeroes than poles).
- Let us see some examples, and then reconsider the limitations above.

Synthesis for set point tracking

Examples

- Example 1 (OK):

$$P(z) = \frac{(z-0.5)}{(z-0.4)^2}, T^\circ(z) = \frac{0.2}{z-0.8} \quad \Rightarrow \quad C(z) = \frac{0.2(z-0.4)^2}{(z-1)(z-0.5)}.$$

- Example 2 (NO, process pole outside the circle):

$$P(z) = \frac{1}{z-2}, T^\circ(z) = \frac{0.2}{z-0.8} \quad \Rightarrow \quad C(z) = \frac{0.2(z-2)}{z-1}.$$

- Example 3 (NO, infeasible relative degree):

$$P(z) = \frac{1}{(z-0.4)^2}, T^\circ(z) = \frac{0.2}{z-0.8} \quad \Rightarrow \quad C(z) = \frac{0.2(z-0.4)^2}{z-1}.$$

Synthesis for set point tracking

Dealing with the limitations

- If the process has poles on or outside the circle, we cannot synthesise by cancellation. Period.
- If the process has zeroes on or outside the circle, we must include them in $T^\circ(z)$ so that they are not cancelled by $C(z)$; this generally requires to increase the relative degree of $T^\circ(z)$ accordingly.
- If the relative degree of $T^\circ(z)$ is infeasible, we need to add "fast" (near to the origin) poles and/or delay terms.

- Let us clarify with a few simulation examples (Scilab scripts).

Synthesis for set point tracking

Dealing with the limitations – Scilab script (1/3, just for reference)

```
// S01-SyhnthesisExamples-SetPointTracking.sce
clear; clc; z = %z;
// Reference and time vectors
w  = ones(1,30);
k  = 0:length(w)-1;
// Process 1, asymptotically stable, no zeroes
P1 = syslin('d',2/(z-0.5)^2);
// Infeasible relative degree
To = syslin('d',0.2/(z-0.8));
C  = 1/P1*To/(1-To); disp(C); y1 = dsimul(tf2ss(To),w);
// Adding a fast pole for a realisable C
To = syslin('d',0.16/(z-0.8)/(z-0.2));
C  = 1/P1*To/(1-To); disp(C); y2 = dsimul(tf2ss(To),w);
// Adding a faster pole
To = syslin('d',0.19/(z-0.8)/(z-0.05));
C  = 1/P1*To/(1-To); disp(C); y3 = dsimul(tf2ss(To),w);
// Adding a one-step delay
To = syslin('d',0.2/(z-0.8)/z);
C  = 1/P1*To/(1-To); disp(C); y4 = dsimul(tf2ss(To),w);
// Using two poles, both faster than the design one
p1 = 0.75;
p2 = 0.45;
To = syslin('d',(1-p1)*(1-p2)/(z-p1)/(z-p2));
C  = 1/P1*To/(1-To); disp(C); y5 = dsimul(tf2ss(To),w);
```


Synthesis for set point tracking

Dealing with the limitations – Scilab script (2/3, just for reference)

```
// Process 2, asymptotically stable,, 1 zero outside the circle
P2 = syslin('d',(z-2)/(z-0.5)^2);
// Infeasible To
To = syslin('d',0.2/(z-0.8));
C  = 1/P2*To/(1-To); disp(C); y6 = dsimul(tf2ss(To),w);
// Adding the required zero to To
To = syslin('d',-0.18*(z-2)/(z-0.8)/(z-0.1));
C  = 1/P2*To/(1-To); disp(C); y7 = dsimul(tf2ss(To),w);
// Improving by acting on the To poles
p1 = 0.78;
p2 = 0.02;
To = syslin('d',-(1-p1)*(1-p2)*(z-2)/(z-p1)/(z-p2));
C  = 1/P2*To/(1-To); disp(C);
y8 = dsimul(tf2ss(To),w);
```

Synthesis for set point tracking

Dealing with the limitations – Scilab script (3/3, just for reference)

```
// Plot the results for process 1
h               = scf(0); clf;
h.figure_size   = [500,500];
title("${\Large \text{Cases 1--5}$");
xlabel("k");
plot(k,y1,'r',k,y1,'r.');
plot(k,y2,'b',k,y2,'b.');
plot(k,y3,'m',k,y3,'m.');
plot(k,y4,'g',k,y4,'g.');
plot(k,y5,'k',k,y5,'k.');
ax              = gca();
ax.data_bounds  = [0,0;max(k),1.1];
ax.tight_limits = "on";
// Plot the results for process 2
h               = scf(1); clf;
h.figure_size   = [500,500];
title("${\Large \text{Cases 6--8}$");
xlabel("k");
plot(k,y6,'r',k,y6,'r.');
plot(k,y7,'b',k,y7,'b.');
plot(k,y8,'k',k,y8,'k.');
ax              = gca();
ax.data_bounds  = [0,-0.3;max(k),1.1];
ax.tight_limits = "on";
```

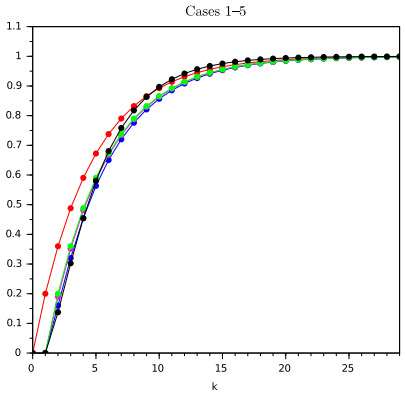

Synthesis for set point tracking

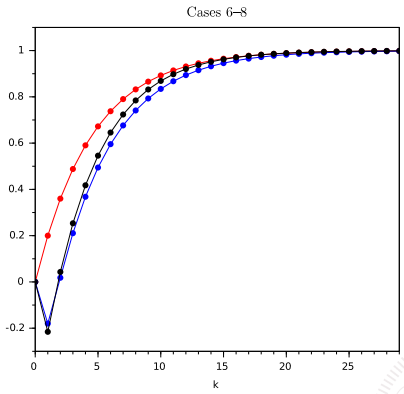
Dealing with the limitations – summary

Case	Process	Requirement	Controller
1	$P_1(z)=\frac{2}{(z-0.5)^2}$	$T_{o11}(z)=\frac{0.2}{z-0.8}$	Infeasible
2		$T_{o12}(z)=\frac{0.16}{(z-0.8)(z-0.2)}$	$C_{12}(z)$
3		$T_{o13}(z)=\frac{0.19}{(z-0.8)(z-0.05)}$	$C_{13}(z)$
4		$T_{o14}(z)=\frac{0.2}{z(z-0.8)}$	$C_{14}(z)$
5		$T_{o15}(z)=\frac{(1-0.75)(1-0.45)}{(z-0.75)(z-0.45)}$	$C_{15}(z)$
6	$P_2(z)=\frac{z-2}{(z-0.5)^2}$	$T_{o21}(z)=\frac{0.2}{z-0.8}$	Infeasible
7		$T_{o22}(z)=-\frac{0.18(z-2)}{(z-0.8)(z-0.1)}$	$C_{22}(z)$
8		$T_{o23}(z)=-\frac{(1-0.78)(1-0.02)(z-2)}{(z-0.78)(z-0.02)}$	$C_{23}(z)$

Synthesis for set point tracking

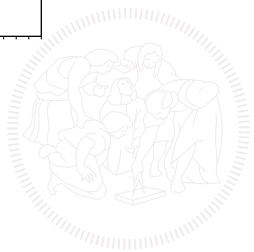
Dealing with the limitations – results





$P_1, T_{o11} \Rightarrow$ infeasible
$P_1, T_{o12} \Rightarrow C_{12}$
$P_1, T_{o13} \Rightarrow C_{13}$
$P_1, T_{o14} \Rightarrow C_{14}$
$P_1, T_{o15} \Rightarrow C_{15}$

$P_2, T_{o21} \Rightarrow$ infeasible
$P_2, T_{o22} \Rightarrow C_{22}$
$P_2, T_{o23} \Rightarrow C_{23}$



Synthesis by transfer function assignment

Disturbance rejection

- Let $P(z)$ be the process under control, and $F^\circ(z)$ the desired closed-loop transfer function from disturbance to controlled variable.
- To find the controller, we solve for $C(z)$ the equation

$$\frac{H(z)}{1+C(z)P(z)} = F^\circ(z),$$

- obtaining

$$C(z) = \frac{1}{P(z)} \frac{H(z)-F^\circ(z)}{F^\circ(z)}.$$

Synthesis for disturbance rejection

Limitations

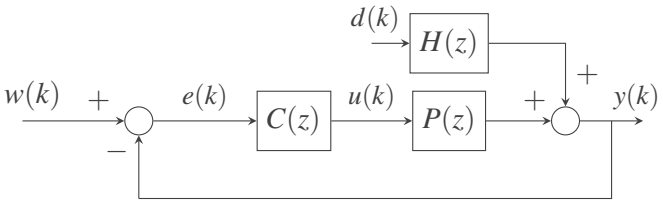
- As for internal stability, the same considerations of the set point tracking case hold, because $C(z)$ still contains $1/P(z)$.
- Realisability can be cumbersome to guarantee, however, and a solution may not exist for any $F^{\circ}(z)$ even if relative degrees are consistent.
- We do not have the space to deal with this issue.

- In general, to effectively reject disturbances, one aims at a fast set point tracking with zero steady-state error.
- Let us see some further simulation examples.

Simulation examples

Foreword

- We refer to the scheme



- We discuss just two cases...
- ...involving however a controlled system that is frequently encountered in the problems we address (the one announced on slide 104).

Example 1

Set point tracking

- Controlled system:
$$P(z) = H(z) = \frac{1}{z-1}.$$
- This is a delayed discrete integrator, as
$$y(k+1) = y(k) + u(k) + d(k),$$
thus
$$y(k) = \sum_{h=0}^{k-1} \big(u(h) + d(h)\big).$$
- Goal: track a step reference with zero steady-state error, exhausting a fraction $\alpha \in (0,1)$ of the whole transient in N steps at most.

Example 1

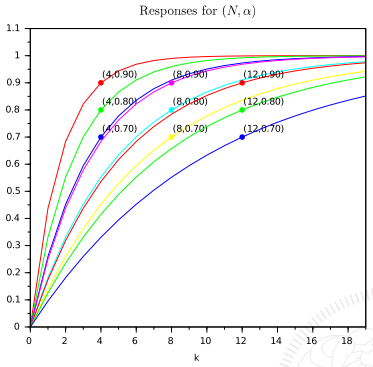
Set point tracking

- Recall that the (unit) step response of
$$T^\circ(z) = \frac{1-p}{z-p}, \quad |p| < 1,$$
is $y(k) = 1 - p^k$.
- This guarantees zero steady-state error since $T^\circ(1) = 1$.
- Then, we need
$$1 - p^N >= \alpha \Rightarrow p = (1-\alpha)^{1/N}.$$
- We have thus the simple proportional controller
$$C(z) = 1 - p = 1 - (1-\alpha)^{1/N}.$$

Example 1

Simulations – Scilab script and results

```
//S02-ControlSynthesis-SimExample1.sce
clear; clc; z = %z;
w               = ones(1,20);
k               = 0:length(w)-1;
P               = syslin('d',1/(z-1));
vN              = [4,8,12];
valpha          = [0.9,0.8,0.7];
colours         = ['r','g','b','m','c','y']
h               = scf(0);
clf;
h.figure_size   = [500,500];
title("${\Large \text{Responses for }(N,\alpha)}$");
xlabel("k");
for i=1:length(vN)
    for j=1:length(valpha);
        C    = 1-(1-valpha(j))^(1/vN(i));
        T    = tf2ss(C*P/(1+C*P));
        y    = dsimul(T,w);
        cdex = (i-1)*length(vN)+j-1;
        cdex = 1+modulo(cdex,size(colours,2));
        c    = colours(cdex);
        plot(k,y,c);
        plot(k(vN(i)+1),y(vN(i)+1),c+'.');
        s    = sprintf("(%d,%.2f)",vN(i),valpha(j));
        xstring(k(vN(i)+1),y(vN(i)+1),s);
    end
end
ax              = gca();
ax.data_bounds  = [0,0;max(k),1.1];
ax.tight_limits = "on";
```



Try with different values of N and α...

Example 2

Disturbance rejection

- Same controlled system.
- With the tracking-centred controller $C = 1 - p$ we get

$$F(z) = \frac{Y(z)}{D(z)} = \frac{H(z)}{1 - C(z)P(z)} = \frac{1}{z - p}.$$

- The disturbance is not rejected, as $F(1) = 1/(1-p)$.
- To reject it, we need a disturbance-to-output transfer function with zero gain.
- For example (mind the relative degree) we can take

$$F^{\circ}(z) = \frac{z-1}{(z-p)^2}, \quad |p| < 1,$$

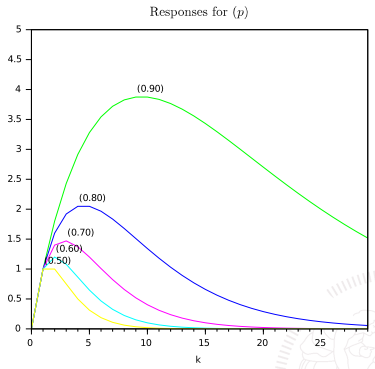
- yielding

$$C(z) = \frac{2(1-p)z + p^2 - 1}{z - 1}.$$

Example 2

Simulations – Scilab script and results

```
// S03-ControlSynthesis-SimExample2.sce
clear; clc; z = %z;
d               = ones(1,30);
k               = 0:length(d)-1;
P               = syslin('d',1/(z-1));
vp              = 0.9:-0.1:0.5;
colours         = ['r','g','b','m','c','y']
h               = scf(0);
clf;
h.figure_size   = [500,500];
title("${\Large \text{Responses for }(p)}$");
xlabel("k");
for i=1:length(vp)
    C     = syslin('d',...
              (2*(1-vp(i))*z+vp(i)^2-1)/(z-1));
    F     = tf2ss(P/(1+C*P)); // Note: H=P
    y     = dsimul(F,d);
    cdex  = 1+modulo(i,size(colours,2));
    c     = colours(cdex);
    plot(k,y,c);
    s     = sprintf("(%.2f)",vp(i));
    [m,j] = max(y);
    xstring(k(j),m,s);
end
ax              = gca();
ax.data_bounds  = [0,0;max(k),5];
ax.tight_limits = "on";
```



See how p controls the convergence speed...

From controller model to control algorithm

An unambiguous process

- A major *plus* of a system- and control-theoretical approach, is that once the controller model is obtained and the loop is assessed, the control algorithm follows unambiguously.
- We now briefly illustrate how this happens, limited to the DT LTI case.

- In fact we already know how to obtain time responses of a DT LTI system based on its transfer function, thus we just need to re-visit that matter.

The control algorithm

From $C(z)$ to the time-domain control law

- Let us consider the controller

$$C(z) = \frac{U(z)}{E(z)} = K\frac{z-b}{z-a}.$$

- We readily have

$$C(z) = K + \frac{K(a-b)}{z-a},$$

- whence

$$\begin{cases} x(k) &= ax(k-1) + K(a-b)e(k-1) \\ u(k) &= x(k) + Ke(k) \end{cases}$$

The control algorithm

From $C(z)$ to the time-domain control law

- This is easily realised in (pseudo-)code as

```
When a control event occurs
   Acquire w and y;
   e      := w-y;
   x      := a*x_prev+K*(a-b)*e_prev;
   u      := x+K*e;
   e_prev := e;
   x_prev := x;
   Apply u;
```

- We skip issues such as initialisation for space reasons.

The control algorithm

Just one add-on – dealing with control saturations

- Suppose that `u` is physically limited to the range `[umin,umax]`.
- This makes the controller nonlinear, apparently.
- However we want it to behave linearly when `u` is in range, i.e., not saturated,
- while keeping its state consistent with input and output in the opposite case.

The control algorithm

Just one add-on – dealing with control saturations

- A possible way to obtain this is state re-computation:

```
When a control event occurs
   Acquire w and y;
   e      := w-y;
   x      := a*x_prev+K*(a-b)*e_prev;
   u      := min(umax,max(umin,x+K*e));
   e_prev := e;
   x_prev := u-K*e;
   Apply u;
```

- Once again we are omitting many important issues, but the above is enough for this course.

Hands-on practice

Objectives

- Synthesise a feedback controller (DT):
  - track a step/ramp set point;
  - reject a step/ramp load disturbance;
  - focus on 1st order process (aka controlled system), define PI and deadbeat control laws.
- Understand the CT-DT relationship as for discretisation (*per exemplum*):
  - compare a CT process to its DT approximation,
  - learn to tune a PI from a step response.

Organisation

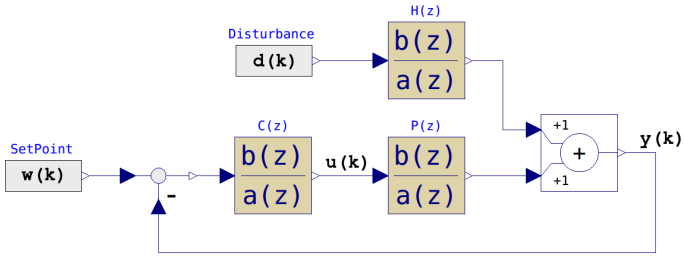
- Highly interactive:
- whiteboard, computations and much simulation.
- Use of wxMaxima and OpenModelica.

- In the case of doubts, stop and ask IMMEDIATELY.

Case 1

Synthesising a DT controller for a DT process

- Modelica scheme (Day2.Case_01):



- We limit the zoo of processes to the most commonly seen ones in CSE-related problems:
  - asymptotically stable (AS) 1st order system with no zeroes,
  - delayed integrator.

Case 1 – AS 1st order process

Set point tracking — i.e., $d(k) = 0$ and thus $H(z)$ irrelevant

- Process and desired setpoint-to-output transfer function:

$$P(z) = \frac{b}{z-a}, \quad T^o(z) = \frac{1-p}{z-p}.$$

- Note that $T^o(z)$ has unity gain to guarantee zero steady-state error.
- To get the controller we solve for $C(z)$ the equation $T(z) = T^o(z)$, i.e.,

$$\frac{C(z)P(z)}{1+C(z)P(z)} = T^o(z) \quad \Rightarrow \quad C(z) = \frac{1}{P(z)} \frac{T^o(z)}{1-T^o(z)}.$$

- In the case at hand,

$$C(z) = \frac{1-p}{b} \frac{z-a}{z-1}.$$

- Note the cancellation and the structure of $C(z)$: one pole in $z = 1$, one zero. Later on we shall call this a PI (Proportional and Integral) control law.

Case 1 – AS 1st order process

Set point tracking

- Start with a unit step $w(k)$:
  - try with $p < a$ (want closed-loop faster then the uncontrolled process) and notice the overshoot of $u(k)$;
  - try with $p > a$ (want closed-loop slower then the uncontrolled process) as a counterpart;
  - try different values of b — same $y(k)$, larger $u(k)$ required for lower b.
- Now try the same with a unit ramp $w(k)$, i.e., $w(k) = k$ (in the Modelica model we emulate this with $w(time) = time$).

- Reconsider the above in a view to carrying out both static and dynamic component sizing...

Case 1 – AS 1st order process

Rejection of "output" disturbance — i.e., $w(k) = 0$ and $H(z) = 1$

- We want the disturbance-to-output transfer function $1/(1+C(z)P(z))$ to have zero gain, i.e., a zero in $z = 1$.
- If we solve for C the equation

$$\frac{1}{1+C(z)P(z)} = \frac{z-1}{z-p}$$

- with our P, we obtain again

$$C(z) = \frac{1-p}{b}\frac{z-a}{z-1}.$$

- Try with step and ramp $d(k)$: the former can be rejected, the latter cannot (with *this* $C(z)$, of PI structure).

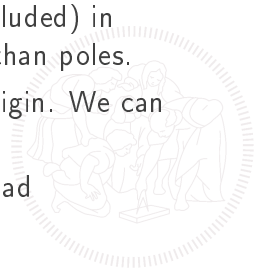
Case 1 – AS 1st order process

Rejection of "load" disturbance — i.e., $w(k) = 0$ and $H(z) = P(z)$

- The difference wrt the previous case is that here the disturbance "enters at the input" of the controlled system.
- Hence this time the disturbance-to-output transfer function is $P(z)/(1+C(z)P(z))$. To give it a zero in $z = 1$, we could solve for $C(z)$ an equation like

$$\frac{P(z)}{1+C(z)P(z)} = \frac{z-1}{(z-p)^n}, \quad n \geq 1$$

  but doing so (i.e., prescribing $Y(z)/D(z)$ completely, denominator included) in general has no feasible solution, as it gives a $C(z)$ with more zeroes than poles.
- However, a PI controller structurally gives $Y(z)/D(z)$ a zero in the origin. We can show the above in wxMaxima.
- Thus, with our $P(z)$, a PI asymptotically rejects a step (not ramp) load disturbance — we omit simulations for brevity



Case 1 – AS 1st order process

Deadbeat set point tracking

- Suppose that for a unit set point step you want to assign the sequence of y samples, ending with zero error (deadbeat response).
- For example (step of w at $k=0$) you may want y to respond as

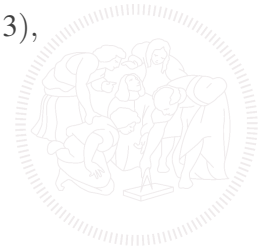
k	-2	-1	0	0	0	0	0	0	...
$w(k)$	0	0	1	1	1	1	1	1	...
$y(k)$	0	0	0	0.2	0.8	1	1	1	...

- This amounts to requiring

$$y(k) = 0.2w(k-1) + (0.8-0.2)w(k-2) + (1-0.8)w(k-3),$$

- that is,

$$\frac{Y(z)}{W(z)} = \frac{0.2}{z} + \frac{0.8-0.2}{z^2} + \frac{1-0.8}{z^3}$$



Case 1 – AS 1st order process

Deadbeat set point tracking

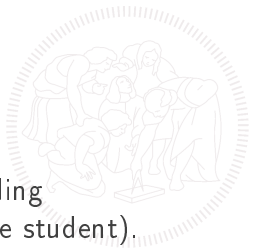
- Rewriting, we have to prescribe

$$T^o = \frac{Y(z)}{W(z)} = \frac{0.2z^2+0.6z+0.2}{z^3}$$

- Notice that the desired $Y(z)/W(z)$ has all the poles in the origin, and feasible relative degree — one, as $P)z)$.
- Hence, setting for example $P(z) = 1/(z-0.5)$ and solving the equation $C(z)P(z)/(1+C(z)P(z)) = T^o(z)$ for $C(z)$, we get

$$C(z) = \frac{2z^3+5z^2-z-1}{10z^3-2z^2-6z-2}$$

- Note that once again $C(z)$ has a pole in $z=1$.
- Let us try the set point step response in Modelica (other tests, including disturbance responses, and their analysis, are left as an exercise to the student).



Case 1 – delayed integrator

set point tracking and load disturbance rejection

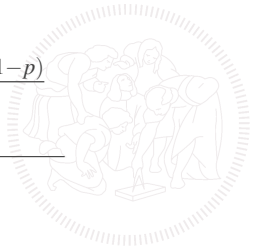
- We now consider the particular but very frequent case

$$P(z) = H(z) = \frac{1}{z-1} \quad \Rightarrow Y(z) = \frac{1}{z-1}\Big(W(z) + D(z)\Big).$$

- Synthetic table:

Objective	Equation	Controller
Track step w	$\frac{Y(z)}{W(z)} = \frac{1-p}{z-p}$	$C(z) = 1-p$
Track step w (deadbeat)	$\frac{Y(z)}{W(z)} = \frac{1}{z}$	$C(z) = 1$
Reject step d	$\frac{Y(z)}{D(z)} = \frac{z-1}{(z-p)^2}$	$C(z) = \frac{(1-p)(2z-1-p)}{z-1}$
Reject step d (deadbeat)	$\frac{Y(z)}{D(z)} = \frac{z-1}{z^2}$	$C(z) = \frac{2z-1}{z-1}$

- We can try some in Modelica.



Synthesis addendum

PI and deadbeat controller — two frequently used control laws

- PI – Proportional plus Integral: the control signal is the sum of two contributions,
  - one proportional to the error $e(k) := w(k) - y(k)$, i.e.

$$u_P(k) = K_P e(k),$$

  - one proportional to the time integral (in DT, sum of samples) of the error, i.e.

$$u_I(k) = u_I(k-1) + K_I e(k-1).$$

- In state space form, with state u_I, this reads

$$\begin{cases} u_I(k) &= u_I(k-1) + K_I e(k-1) \\ u(k) &= u_I(k) + K_P e(k) \end{cases}$$

- hence in transfer function form

$$C(z) = \frac{K_I}{z-1} + K_P = \frac{K_P z + K_P - K_I}{z-1}.$$

Synthesis addendum

PI and deadbeat controller — two frequently used control laws

- Deadbeat – specify a desired closed-loop transfer function with all poles in the origin, then obtain $C(z)$ by solving the convenient equation, which is often called direct synthesis.
- We omit discussions on feasibility.
- Remember that critical cancellations are not allowed — we omit further discussions on this aspect as well.
- In the application examples we shall meet both types of controller.

Problem statement and motivation

tailored to the purpose of this course

- Given an LTI CT system with transfer function $G(s)$, find a DT (LTI) one with transfer function $G^*(z)$ that approximates it.
- In our context, this is useful to turn a CT model into a DT one, to synthesise a DT controller (at fixed timestep) with the methods seen before.
- There is a vast theory, here we just propose and justify a method.
- NOTE: when CT and DT quantities co-exist, we shall denote the latter with an asterisk superscript, while when the treatise is entirely DT we shall omit this for compactness.

Main idea

replace the time derivative with the incremental ratio over one timestep

- The discrete time k here counts the timesteps h in the continuous one t, i.e.,

$$t = kh.$$

- This applows to write the k-th sample of the generic CT variable $v(t)$ as

$$v^*(k) = v(kh).$$

- Thus, in the discrete time k, we approximate the time derivative of the generic $v(t)$ as

$$\left.\frac{dv(t)}{dt}\right|_{t=kh} \cong \frac{v^*(k) - v^*(k-1)}{h},$$

- whence, bringing in the CT derivative operator s and the DT one-step advance operator z,

$$s\,v(t)|_{t=kh} \cong \frac{1-z^{-1}}{h}\,v^*(k).$$

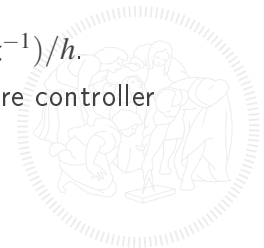
Main idea

known as the "backward difference" or "implicit Euler" method

- Take the CT system (assumed stable), simulate a step response, choose h so that the resulting number of samples is enough to represent that response precisely enough.
- Calculate the DT system transfer function as

$$G^*(z) = G\left(\frac{1-z^{-1}}{h}\right),$$

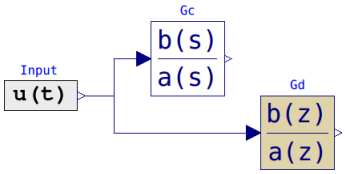
  i.e., taking the CT transfer function $G(s)$ and replacing s with $(1-z^{-1})/h$.
- Let us see an example in Modelica, and then synthesise a temperature controller for the heated plate.



Case 2

Discretisation example

- Modelica scheme (Day2.Case_02):

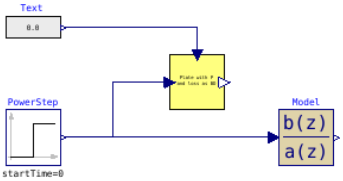


- Take a CT transfer function,
- observe its step response (in open loop) and decide h,
- compute the discrete-time transfer function,
- simulate the two and compare.

Case 3

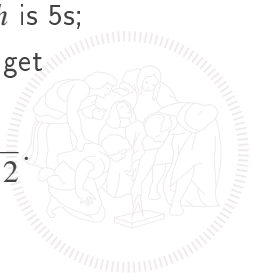
A controller for the heated plate of Case 1 this morning

- Model for the open-loop experiment (Day2.Case_03_Modelling):



- Starting with zero initial conditions and inputs other then P (i.e., interpreting everything as variations from an equilibrium) we find that a suitable h is 5s;
- Since with the given parameter values $1/G = 695$ and C/G=112, we get

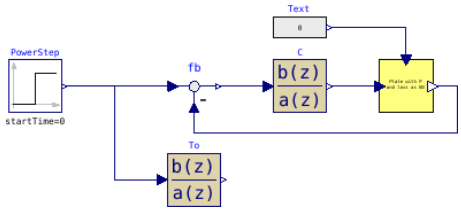
$$P(s) = \frac{695}{1+112s} \quad \Rightarrow \quad P^*(z) = P\left(\frac{1-z^{-1}}{5}\right) = \frac{3475z}{117z-112}.$$



Case 3

A controller for the heated plate of Case 1 this morning

- Model for closed-loop control (Day2.Case_03_Control):



- We prescribe $T^o(z)$ and get $C(z)$:

$$T^o(z) = \frac{(1-0.9)z}{z-0.9} \quad \Rightarrow \quad C(z) = \frac{117z-112}{31275(z-1)}.$$

- Let us see the result in Modelica, try with different h and T^o, and comment.

This morning

Takeaways

- The feedback loop as the main component of control systems.
- Functional view (MAPE-K) vs. systemic view (both with their purpose);
- From requirements in common language to requirements in SC terms (formalisation).
- A few control synthesis methods (DT).
- From controller transfer function to control algorithm.

This afternoon

Takeaways

- DT synthesis examples.
- PI and deadbeat controllers.
- Discretisation.
- A CT/DT synthesis case. i.e., from CT model through DT model to DT control law (and then, although not seen today, algorithm).

Tomorrow

Main topics and goals

- In the morning, applications 1: scheduling and clock synchronisation in wireless sensor networks.
- In the afternoon, practice: first on synthesis *in abstracto*, then further investigating the synchronisation application.
- At the end, discussion and question time.

– Day 3 –

Outline

- **Morning**: *applications 1*
  - Scheduling and resource allocation;
  - clock synchronisation in wireless sensor networks.
- **Afternoon**: *hands-on practice*
  - Control synthesis exercises;
  - the clock synchronisation case more in depth, and hands-on;
  - question time and discussion.

Caveat

- This morning, like tomorrow morning, is dedicated to a gallery of examples about what SC can do for CSE.
- The shown results are production-grade, hence sometimes we shall get to talk about things you are *not* expected to do in this course,
- or even not expected to do, period — we the SC guys must be there for something, in the end... :-)

- Therefore aim at the principles more than the details (but if you get lost, of course stop and ask).

Scheduling & resource allocation

Objectives

- Address the problem of resource allocation in computing systems in a control-theoretical manner:
- devise a simple solution, lightweight enough for massive online use;
- provide a formal stability proof;
- sketch out a proper interface for the system administrator (typically, not a control expert);
- apply the proposal to the (tough) context of task scheduling.

Problem overview

Foreword

- Computing systems need to manage the allocation of many resources:
  - CPU time, memory, disk space,
  - but also computing units (e.g., cores),
  - or power consumption allowance and battery energy budget,
  - and many other more or less "material" items.
- Traditionally, *ad hoc* (heuristic) solutions are used.
- Most of the said problems are control ones in nature, however.
- Can they hence be cast in our unifying framework?
- Let us try to provide here an answer.
- Note: we refer to task scheduling as an example, but the ideas are general.

Problem overview

(scheduling as example)

- Main ingredients of the problem:
  - a resource to be allocated (CPU time),
  - a generally time-varying pool of users (tasks),
  - user-generated time-varying resource requests,
  - limits for the resource, on a per-user and/or a global basis.
- Design choices:
  - no assumption on *how* the users will employ the resource,
  - all discrepancies between allotted and used resources treated as disturbances.
- Needs:
  - a general model for resource consumption,
  - a feedback control strategy,
  - a proof of stability,
  - a suitable setpoint generation mechanism.

The proposed methodology

Resource consumption model (scheduling as example)

- Model for the user (task) pool

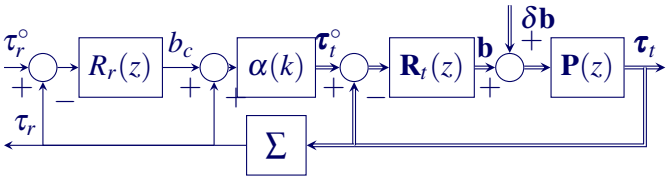
$$\mathbf{P}: \begin{cases} \boldsymbol{\tau}_t(k) = & \mathbf{b}(k-1) + \delta\mathbf{b}(k-1) \\ \tau_r(k) = & \sum_{i=1}^{N} \tau_{t,i}(k) \\ \tau(k) = & \tau(k-1) + \tau_r(k-1) \end{cases}$$

- Very simple and LTI, provided k counts allocation decisions.
- Straightforwardly generalised to other resources.
- Uncertainty-free.
- Limited to the core physical phenomenon
  (adopting the proposed methodology may require some re-design).

The proposed methodology

Control scheme (scheduling as example)

- A two-level strategy:

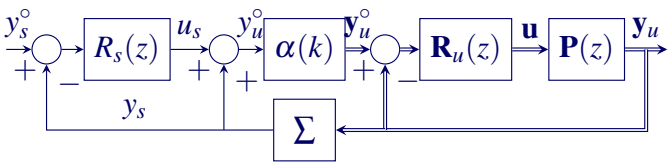


- Inner loop enforces allotted usage, outer loop distribution in between two allocation decisions.
- Already proven very effective for scheduling.
- Main problems:
  - ensure stability in the presence of the time variance induced by $\alpha(k)$,
  - devise a mechanism to generate set point(s) and distribution (here, $\tau_r^\circ(k)$ and $\alpha(k)$, respectively).

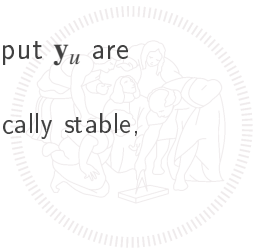
Stability proof

Just a sketch for brevity

- Scheme (generalised) with user- and system-level control:



- $\mathbf{P}(z)$ and $\mathbf{R}_u(z)$ diagonal, with identical elements.
- Hence, the matrices $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$ of the system with input $\mathbf{y}_u^\circ$ and output $\mathbf{y}_u$ are block diagonal. Assume
  - $\mathbf{A}$ Schur (all eigenvalues in the unit circle), i.e., inner loops asymptotically stable,
  - $\sum_{i=1}^{N} \alpha_i(k) = 1 \, \forall k$ (by construction).



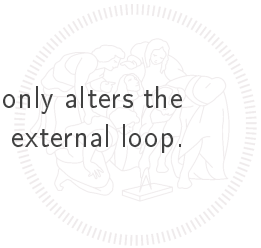
Stability proof

cont'd

- By means of a suitable state space transformation, the overall dynamic matrix can be written as

$$\tilde{\mathfrak{A}}_s = \left[\begin{array}{ccccc} A+BC & 0 & \cdots & \cdots & 0 \\ \hline BC\alpha_2(k) & A & 0 & \cdots & 0 \\ BC\alpha_3(k) & 0 & A & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ BC\alpha_N(k) & 0 & \cdots & 0 & A \end{array}\right]$$

- Hence, if $R_s(z)$ stabilises $A+BC$, the time variance induced by $\alpha(k)$ only alters the inputs of asymptotically stable systems that do not take part into the external loop.



Points to address

- Control structuring;
- classification of the users (tasks);
- consequent generation of set point(s) and distribution.

- Main issue: make the system manageable by a person who is not a control expert, and is often acquainted to metrics that are hardly possible to express (directly) in control-theoretical terms.

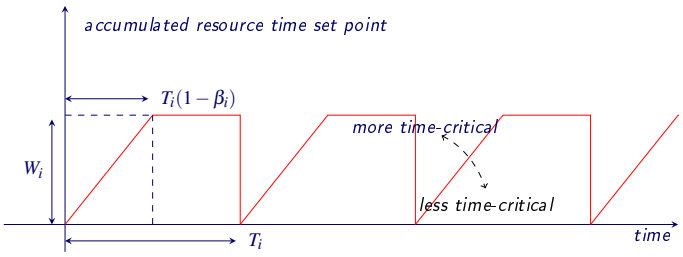
Control structuring

- The problem naturally indices a cascade structure;
- to minimise overhead, simple controllers should be selected;
  - internal controller – integral or deadbeat,
  - external controller – PI, pure proportional or deadbeat.
- Integral and proportional blocks give lowest orders;
- deadbeat blocks may require more stages, but allow to shape the set point responses.

Classification of the users for distribution

In the example, four task types

- Type 1 – tasks with periodic deadlines (T_i period, W_i workload):



- Type 2 – tasks with a single deadline: analogous to type 1, just triggered only once.

Classification of the users for distribution

In the example, four task types

- Type 3 – Tasks without deadlines: "fake" priorities

$$\widehat{\alpha}_i(k) = \alpha_{\min,i} + p_i(\alpha_{\max,i} - \alpha_{\min,i}), \quad 0 \leq p_i \leq 1$$

  where p_i can be mapped to "traditional" quantities like e.g. the nice number.
- Type 4 – event-triggered tasks: exponential share decay

$$\widehat{\alpha}_i(k) = \widehat{\alpha}_i(k-1) \cdot a_i^{-(k-k_{0,i})}, \quad 0 < a_i < 1,$$

  with $\widehat{\alpha}_i(k)$ reset to a_i when the task is (re-)triggered.

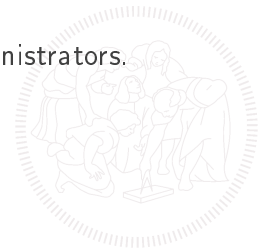
- Other types: devise analogous rules (stability never jeopardised).

A system-level management *vista*

Set point and distribution generation

- Set the system-level set point $\tau_r^\circ(k)$ for the desired responsiveness level;
- compute the $\widehat{\alpha}_i(k)$ for the current task pool;
- compute the components $\alpha_i(k)$ of vector $\alpha(k)$ by rescaling to unity sum.

- Note the use of concepts/quantities well understood by system administrators.



A benchmark OS

Miosix

- Web site:
  `https://miosix.org`
- Wiki:
  `https://miosix.org/wiki/index.php`
- Repository:
  `https://github.com/fedetft/miosix-kernel`

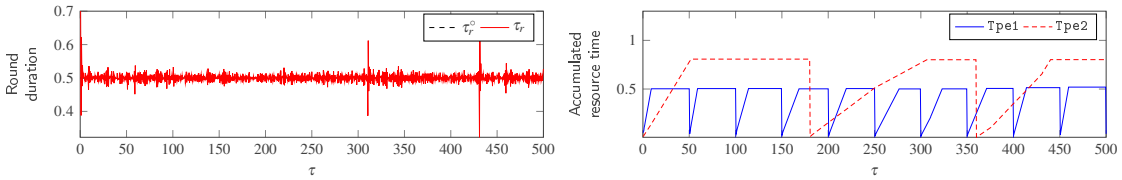
An example of experimental evaluation

Task pool

ID	**Periodic**	period	workload	β	
1	`Tpe1`	50	0.5	0.5	
2	`Tpe2`	180	0.8	0.2	
	Batch	arrival	workload	duration	β
3	`Tba1`	100	60	300	0.1
4	`Tba2`	150	70	400	0.1
	Priority	priority			
5	`Tpr1`	0.1			
6	`Tpr2`	if($\tau < 250 \wedge \tau > 375$) 0.4, else 1.0			
7	`Tpr3`	0.2			
	Ev-trig	in. share	dec. rate	trig. times	
8	`Teb1`	0.02	0.6	10, 20, 100, 280, 300	
9	`Teb2`	0.03	0.7	5, 15, 50, 80, 150, 400	

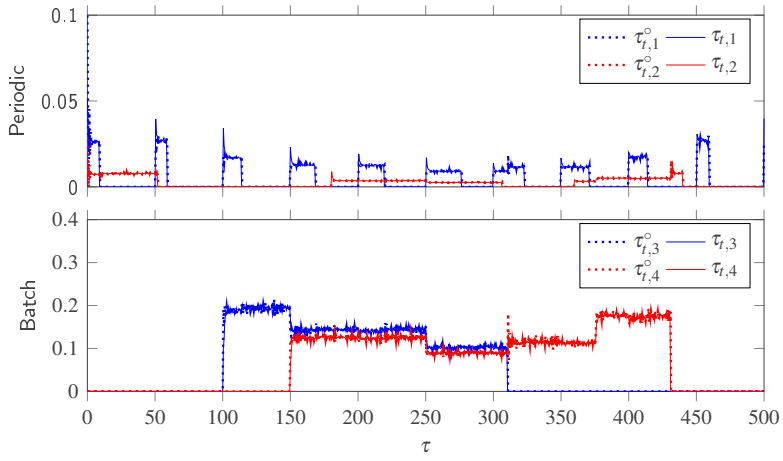
Results

Round duration and accumulated resource (CPU time) for periodic tasks (controlled)



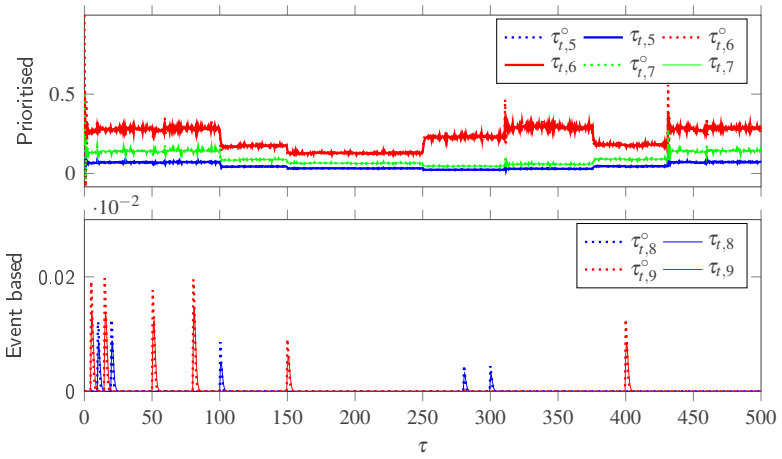
Results

Resource (CPU time) distribution (1/2)



Results

Resource (CPU time) distribution (2/2)



Conclusions drawn so far

- The problem of computing systems resource provisioning can be addressed from a control-theoretical perspective.
- A scheme for resource allocation was presented and the stability of the closed loop system was proved.
- The parametrisation of said control scheme is manageable by system administrators.

Future directions

- Address the extension of the presented solution to the allocation of other resources;
- carry out experiments with the devised control schemes in OS kernels (as already done for scheduling in `Miosix`);
- discuss issues on the API of the said kernels (syscalls and the like) for possible re-structuring induced by the new allocation solutions introduced.

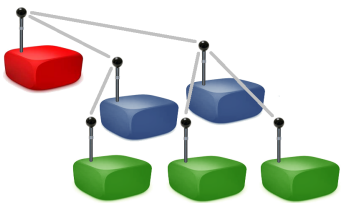
WSN clock synchronisation

Problem statement

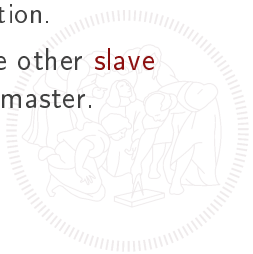
- Goal:
  make the clock of all the nodes in a wireless sensor network (WSN) agree on a unique time.
- Importance:
  - event ordering/correlation,
  - event interval measurements,
  - low power radio operation,
  - ...

Problem statement

- We focus on asymmetric master-slave synchronisation;
- we address both the single- and the multi-hop case.



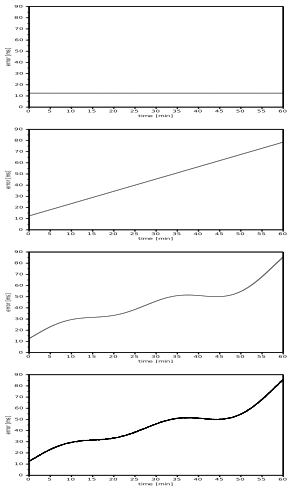
- One master node holds the "true" time
  - as it is the gateway,
  - as it has a GPS,
  - or simply by convention.
- We need to keep all the other slave nodes in sync with the master.



Synchronisation error sources

A simple example with two nodes

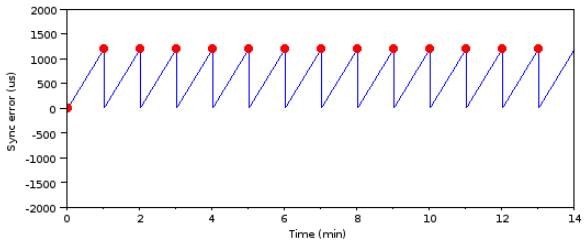
- Difference between master and slave clock in the absence of synchronisation:
  - Offset
  - Offset+skew
  - Offset+skew+drift
  - Offset+skew+drift+jitter



Mainstream solution

The "clock synchronisation plus skew compensation" (CS+SC) paradigm

- CS – the master periodically transmits a timestamp, the slaves overwrite their clock:

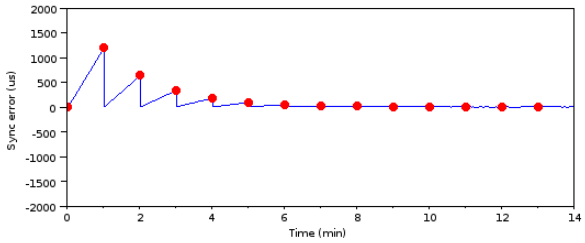


- The synchronisation error no longer grows arbitrarily,
- but the slave clock is inherently not continuous, and can loose monotonicity.

Mainstream solution

The "clock synchronisation plus skew compensation" (CS+SC) paradigm

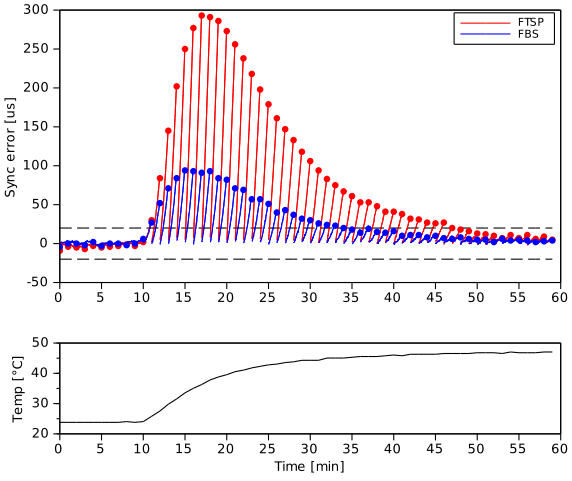
- SC – the slave estimates the error slope and compensates its clock in between two subsequent CS operations:



- If that skew estimate is correct, the slave clock is continuous and monotonic,
- but the presence of drift (i.e., a varying skew) can make the statement above highly questionable.

Mainstream solution

How the most employed (CS+SC) solutions react to drift



- A shade-sunlight transition causes a heat rate step,
- thus a temperature transient,
- thus drift, as the frequency of a quartz depends on its temperature.
- We show results with a 32 kHz off-the-shelf crystal, and two synchronisation schemes:
  - Flooding Time Synchronisation Protocol;
  - Feedback Based Synchronisation.

FLOPSYNC-2 (Feedback LOw Power SYNChronisation v2)

viewed from outside as a black box

- High accuracy when temperature is constant:
  $\sim$100ns average, $<1\mu$s standard deviation, even after 8 hops.
- Guaranteed sync bounds during temperature transients:
  offline Design Space Exploration (DSE) + tuning.
- Guaranteed monotonic and continuous clock:
  the clock of each node has no forward/backward jumps.
- Ultra-low power in typical conditions:
  $<1\mu$A current overhead per node.
- Ultra-low power also in extreme environments:
  offline DSE minimises power consumption subject to accuracy constraints.

FLOPSYNC-2

viewed from inside as a glass box — for the moment no *control-ese* :-)

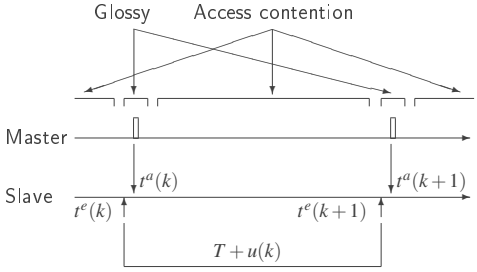
- FLOPSYNC-2 has four main components:
  - the error measurement block,
  - the monotonic virtual clock,
  - the feedback controller,
  - and the configuration tool.
- The first three run on the target node;
- the fourth is used offline to fine-tune a WSN prior to deployment.

The error measurement block

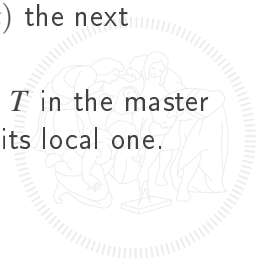
- For this component we build upon Glossy (Ferrari *et al.*, 2011).
- In a typical MAC protocol packets are subject to access contention, and retransmissions become necessary.
- Glossy is a flooding scheme that partitions radio usage on a temporal basis, eliminating access contention.
- We enhance Glossy with the following features:
  - no timestamps in synchronisation packets,
  - hardware packet rebroadcast,
  - online idle listening minimisation.

The error measurement block

Basic operation



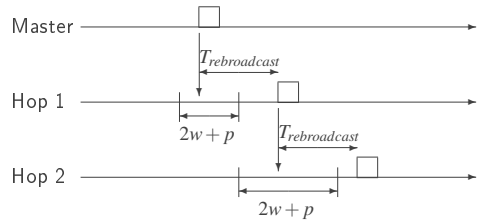
- The master floods a sync packet at fixed period T.
- The slave measures the error as $t^a(k) - t^e(k)$, where $t^e(k)$ and $t^a(k)$ are the estimated and actual arrival time of that packet.
- The slave alters by $u(k)$ the next expected arrival time...
- ...attempting to match T in the master clock with $T+u(k)$ in its local one.



The error measurement block

Minimising idle listening and per-hop jitter accumulation

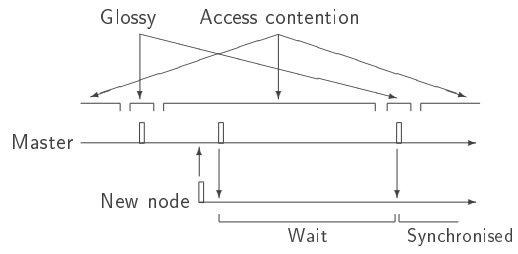
Zoom on one sync packet flood



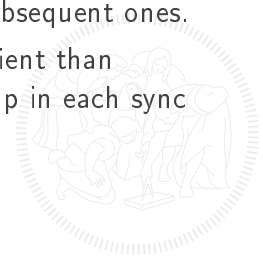
- We neglect the radio propagation time.
- The radio is turned on in advance to catch the sync packet;
- the *receiver window* w is adapted on-line to be three times the sync error standard deviation;
- the packet is rebroadcast after a fixed amount of time, accounted in hardware.

The error measurement block

Boot procedure to eliminate the initial offset

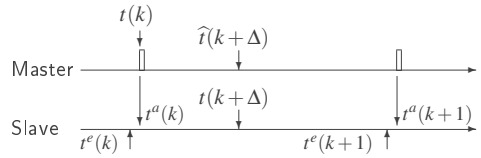


- A slave signals that it joins the WSN;
- the master answers with T and the time when the *next* sync packet will be sent;
- on receiving that packet the slave zeroes its offset, and knows the (master) time of all subsequent ones.
- This proved more efficient than embedding a timestamp in each sync packet.



The monotonic virtual clock

i.e., how to estimate the master time when some application requires it



$$\widehat{t}(k+\Delta) = t(k) + [t(k+\Delta) - t^e(k)] \cdot \frac{T}{T+u(k)}$$

- The slave hardware clock is never corrected;
- the master time is estimated by linear interpolation of the two closest sync packets' expected arrival times.

The feedback controller

A very few preliminaries

- The master operation is
  - send out sync packets (by flooding) every T;
  - respond (by normal MAC) to joining slaves.
- The slave operation (after joining) is
  - wait for sync packets and compute u on receiving one;
  - estimate the master time when required.
- The scheme is fully decentralised:
  - no slave-slave interaction;
  - nobody's workload depends on the WSN size.
- We now go for synthesis and assessment — and *control-ese* moves in.

The feedback controller

Model of the uncontrolled process

- Let t be the master (true) time, f_o the nominal frequency of the slave oscillator, and $\delta f(t)$ the variation of that frequency over time, no matter what its origin is.
- The local time t_{loc} evolves over the true one t as

$$t_{loc}(t) = \int_0^t \frac{f_o + \delta f(\tau)}{f_o} d\tau,$$

- hence the synchronisation error, in the absence of any correction, is

$$e(t) := t - t_{loc}(t) = t - \int_0^t \left(1 + \frac{\delta f(\tau)}{f_o}\right) d\tau = -\int_0^t \frac{\delta f(\tau)}{f_o} d\tau.$$

The feedback controller

Model of the uncontrolled process

- Sync packets are flooded at the master times $t(k) = kT$, thus

$$e(k) := t(k) - t_{loc}(t(k)) = -\int_0^{kT} \frac{\delta f(\tau)}{f_o} d\tau.$$

- Now we define the period-cumulated frequency variation effect

$$d(k) := -\int_{kT}^{(k+1)T} \frac{\delta f(\tau)}{f_o} d\tau,$$

  which is legitimate as $d(k)$ depends on frequency variations from kT till immediately before $(k+1)T$,
- and we get the uncorrected error as the sum of the said cumulated effects, i.e.,

$$e(k) = \sum_{\ell=0}^{k-1} d(\ell) \;\Rightarrow\; e(k) = e(k-1) + d(k-1).$$

The feedback controller

Model of the uncontrolled process

- Observe that up to now we have modelled the phenomenon, but said nothing about how we *measure* the error:
- time to formalise the intuitive idea of "expected minus actual arrival time, both counted in the local slave clock".
- The expected arrival time (u is the correction) turns out to be

$$t^e_{loc}(k) = kT + \sum_{\ell=0}^{k-1} u(\ell),$$

- while the actual one is

$$t^a_{loc}(k) = kT + \int_0^{kT} \frac{\delta_f(\tau)}{f_o} \mathrm{d}\tau = kT - \sum_{\ell=0}^{k-1} d(\ell).$$

The feedback controller

Model of the uncontrolled process

- Subtracting to get the error measurement we have

$$e(k) = t^e_{loc}(k) - t^a_{loc}(k) = \sum_{\ell=0}^{k-1} u(\ell) + \sum_{\ell=0}^{k-1} d(\ell),$$

- thus

$$e(k) = e(k-1) + u(k-1) + d(k-1),$$

- that in transfer function form reads

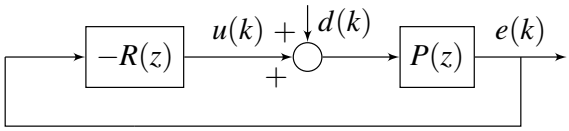
$$E(z) = \frac{1}{z-1}\big(U(z) + D(z)\big), \quad P(z) = \frac{1}{z-1}.$$

- Yes, the delayed integrator is back! :-)

The feedback controller

Model of the uncontrolled process

- Summing up, the controlled system has the block diagram



  where the minus sign on $R(z)$ is to not write the summation node, as the error set point is obviously zero.
- The loop is parameter- and uncertainty-free, as the error integrates $d(k)$ irrespective of its origin.
- Any uncertainty is in the generation of the disturbance $d(k)$, outside the loop: no effects of this on stability.
- Control design boils down to analysing the said disturbance, and setting up an $R(z)$ to reject it effectively.

The feedback controller

Choice of the regulator block transfer function

- A suitable $R(z)$ – we now check with wxMaxima – is

$$R(z) = \frac{3(1-\alpha)z^2 - 3(1-\alpha^2)z + 1 - \alpha^3}{(z-1)^2},$$

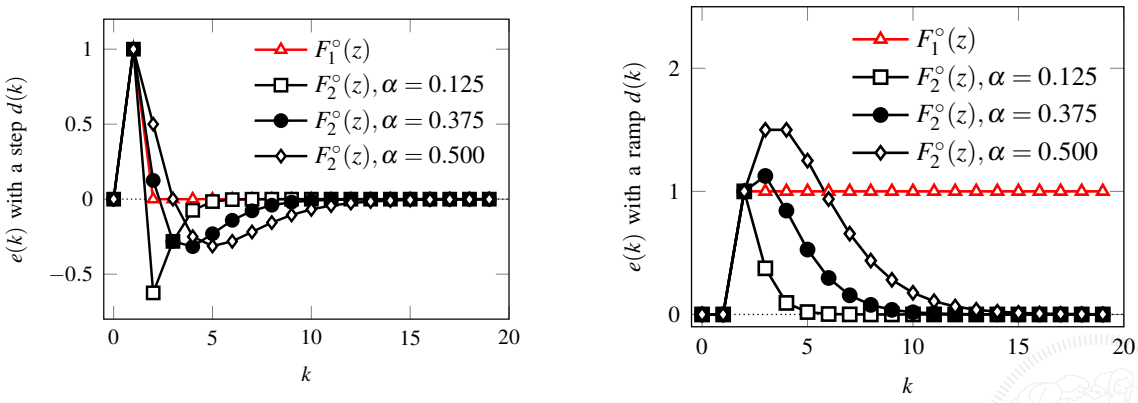
- yielding $F(z) := E(z)/D(z) = F^o(z)$, suitable to reject a ramp disturbance, with

$$F^o(z) = \frac{(z-1)^2}{(z-\alpha)^3}.$$

- Parameter α is chosen at design time to trade thermal versus jitter rejection; we designed the configuration tool for that.
- The algorithm for $R(z)$ above is quite lightweight (600 bytes of code, 28 bytes of RAM, 4 to 7μs on a 24MHz ARM CPU).

The feedback controller

Effect of parameter α



$F_2^o(z)$ is the design just suggested,

$F_1^o(z)$ is a baseline design with a PI controller; see Terraneo *et al.*, 2014 and Leva *et al.*, 2015 for details.

The configuration tool

Pre-tuning a WSN deployment based on quartz data and harshest assumed thermal stress

- FLOPSYNC-2 has two design parameters:
  - the synchronisation period T,
  - and the controller parameter α.
- The stability of the closed loop and the error convergence to zero is always guaranteed.
- The selection of parameters is only to optimise a tradeoff:

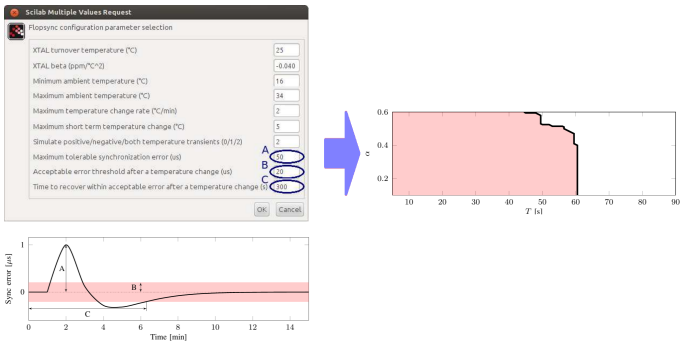
Tighter sync ← low — T — high → Lower power

Better thermal rejection ← low — α — high → Better jitter rejection
up to a max of 0.6

The configuration tool

Basic operation

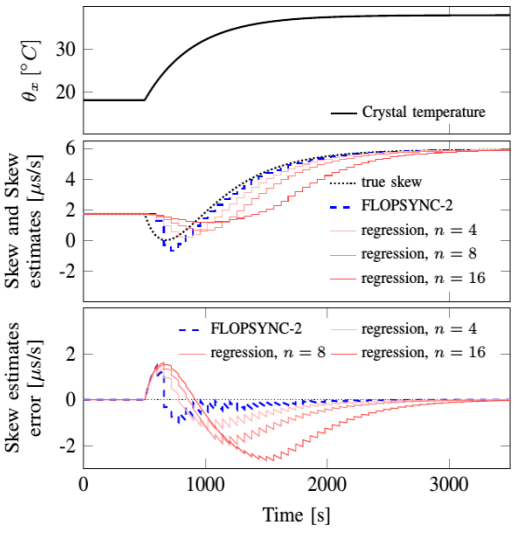
- The tool takes a simple thermal model and performs a design space exploration.



- Inputs:
  - sync error constraints;
  - maximum and minimum ambient temperature;
  - maximum short term temperature change;
  - crystal nominal parameters.
- Outputs:
  - feasible (T, α) region;
  - Pareto-optimal frontier.

Simulation results

Just an example – why *not* use linear regression to compensate for skew



- We created a Modelica model library mixing equation- and algorithm-based components;
- this afternoon we shall see more, and experiment.

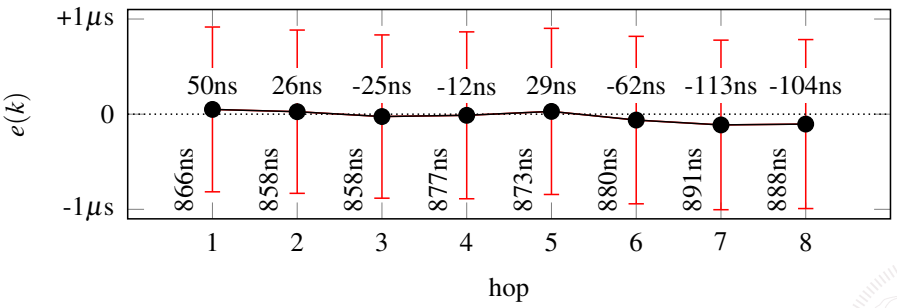
Experimental results

Overview

- We implemented FLOPSYNC-2 on a WSN node platform with an ARM Cortex-M3 microcontroller @ 24MHz and a CC2520 transceiver.
- In the following we present two results:
  - synchronisation error for an eight-hop WSN with negligible thermal stress;
  - synchronisation error over a single hop with severe thermal stress.

Experimental results

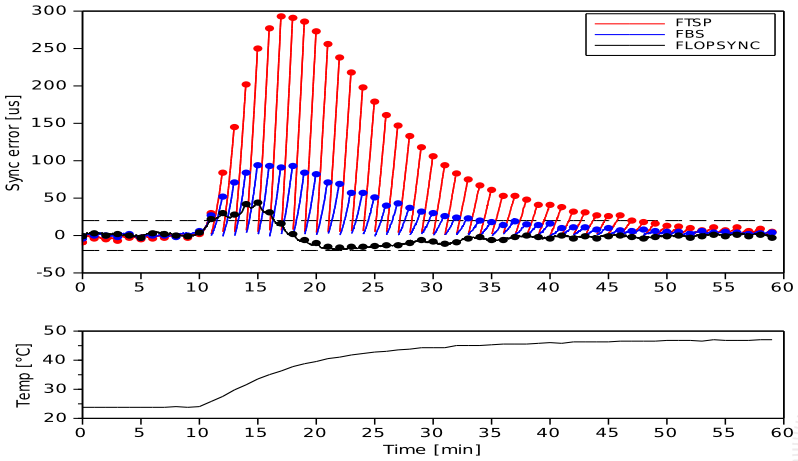
WSN with eight hops – error mean and standard deviation



FLOPSYNC-2 shows no relevant jitter accumulation over hops.

Experimental results

Single hop – shade-sunlight transition



FLOPSYNC-2 works better than mainstream alternatives;

in particular, it steers error toward zero with T still increasing (as it can reject a ramp disturbance).

Conclusions drawn so far

concerning WSN synchronisation

- A control-centric view helped formalise the problem and streamline the solution process.
- Simulation-based assessment was made with system-level models, and provided reliable results.
- FLOPSYNC-2 outperforms alternatives, and can be configured based on a dynamic first-principle model of the node, plus worst-case assumptions on the expectable thermal stress.

Present and future directions

- Flight time compensation (Terraneo *et al.*, 2015);
- integration in a radio stack;
- extensions to the wired case.

`https://github.com/fedetft/flopsync`

Hands-on practice

Interactive session

Whiteboard and computer, no slides

- Control synthesis exercises;
- the clock synchronisation case more in depth, and hands-on;
- question time and discussion.

– Day 4 –

Outline

- **Morning**: *applications 2*
  - Thermal/power/performance management;
  - self-adaptive software (some cases).
- **Afternoon**: *hands-on practice*
  - A few more control synthesis exercises;
  - modelling, simulating and controlling queues;
  - question time and discussion.

Caveat

- Also this morning is dedicated to a gallery of production-grade examples about what SC can do for CSE;
- hence the same considerations of yesterday morning apply

- Aim at the principles more than the details (but if you get lost, once again of course stop and ask).

Thermal/power/performance management

Problem overview

Preserve microprocessor thermal safety at the minimum performance cost

- Goal:
  deliver the computational power required by the load
  but maintain temperature at safe values.
- Importance:
  - CPU safety,
  - long-term reliability,
  - computational efficiency,
  - energy efficiency,
  - ...

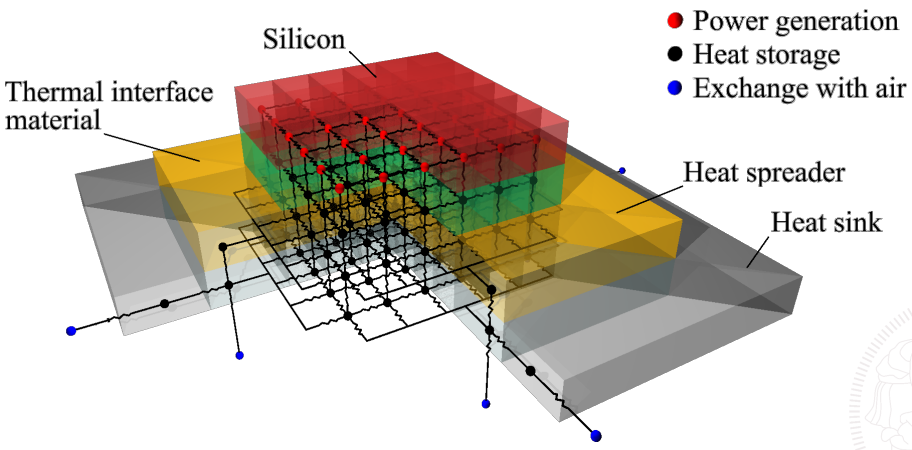
Problem overview

Critical issues

- Thermal dynamics with multiple time scales,
- including *really* fast ones — and getting faster;
- under-actuated system (per-core DVFS not standard);
- different installations (smarthphone, tablet, desktop...) for the same chip;
- totally unpredictable load-generated disturbances, only upper bounds available;
- need to not introduce (too much) overhead.

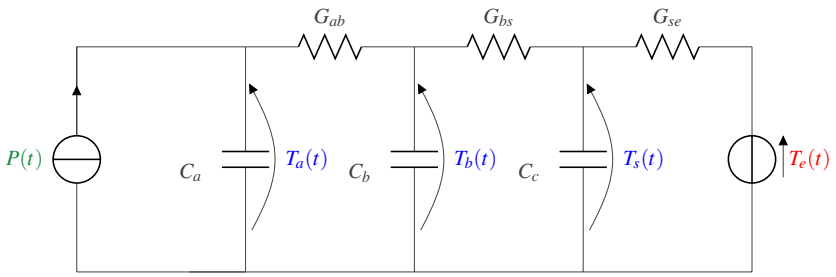
Thermal model

Physical layout of the addressed system



Thermal model

A simplified electric equivalent



- Subscripts: a for active silicon, b for bulk silicon, s for spreader/sink, e for exterior.
- Three states, a disturbance and an input that can be (partially) governed.

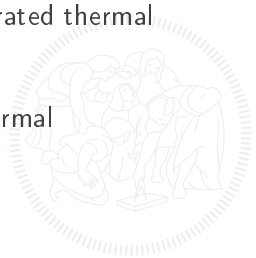
A bit of history

(1/2)

- Ancient age – pre-Pentium:
  - low power densities, no need for a heat sink;
  - thermal control was a no-problem;
  - power/performance management tools just stopped the CPU in the absence of load — useful for laptop batteries those days.
- Middle age – design for TDP and use a heat sink:
  - power densities still allowed for all CPU units at full power;
  - one had just to design for worst possible thermal stress (the Thermal Design Power),
  - and provide decent dissipation with the sink;
  - thermal control and power/performance management split,
    - as the former was dealt with by means of a fan,
    - while for the latter governors were created to act on DVFS (Dynamic Voltage and Frequency Scaling) based on the load.

A bit of history
(2/2)

- Modern age – the era of dark silicon:
  - breakdown of Dennard scaling (around 2006),
  - thus the focus moved to multicores;
  - however densities became so high to prevent operating all units at full power, or the chip burns;
  - "end of multicore scaling" foreseen (Esmaeilzadeh *et al.*, 2011);
  - academia and industry started diverging significantly.
- Contemporary age (in a nutshell):
  - from research, plenty of advanced techniques – e.g., MPC – for integrated thermal and power/performance management...
  - ...but significant applicability issues;
  - from industry, governor-like power/performance management plus thermal protection...
  - ...thus viable solutions, but the above integration is given up.



Back to the thermal model
A matter of sizes and times

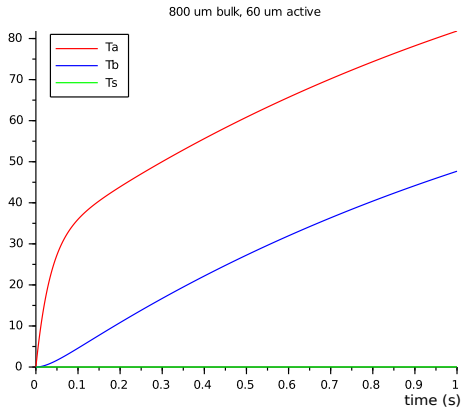
- Dynamic model (this time CT):

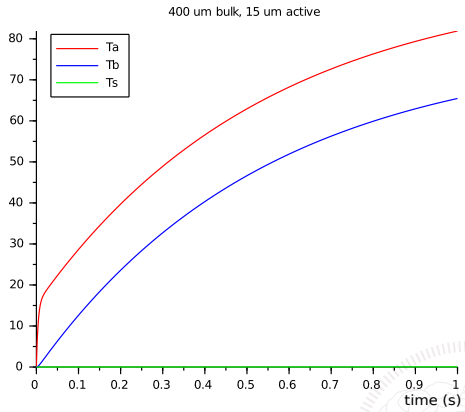
$$\begin{bmatrix} \dot{T}_a \\ \dot{T}_b \\ \dot{T}_s \end{bmatrix} = \begin{bmatrix} -\frac{G_{ab}}{C_a} & \frac{G_{ab}}{C_a} & 0 \\ \frac{G_{ab}}{C_b} & -\frac{G_{ab}+G_{bs}}{C_b} & \frac{G_{bs}}{C_b} \\ 0 & \frac{G_{bs}}{C_s} & -\frac{G_{bs}+G_{se}}{C_s} \end{bmatrix} \begin{bmatrix} T_a \\ T_b \\ T_s \end{bmatrix} + \begin{bmatrix} \frac{1}{C_a} & 0 \\ 0 & 0 \\ 0 & \frac{G_{se}}{C_s} \end{bmatrix} \begin{bmatrix} P \\ T_e \end{bmatrix}$$

- Let us vary the active silicon and bulk thicknesses, and simulate the response to a power step;
- figures are *very* first-cut, but the main facts emerge — and are confirmed in practice, as shown later on.

A matter of sizes and times

Temperature response to a power step (exterior as baseline)

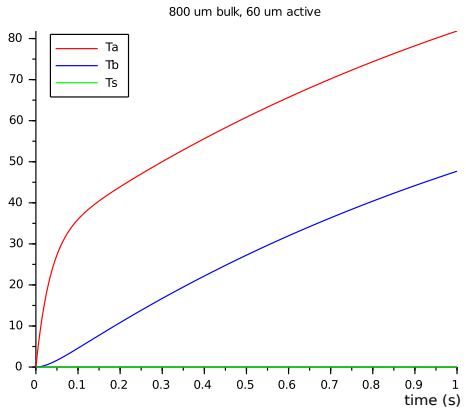


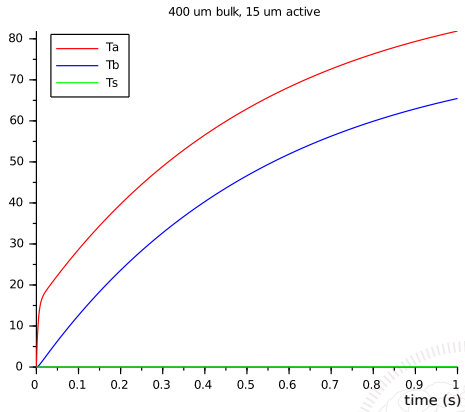


- On a 1s horizon the sink does not move;
- with smaller scales the bulk response stays quite similar,
- while the sharp swing of the active silicon is enhanced.

A matter of sizes and times

Temperature response to a power step (exterior as baseline)

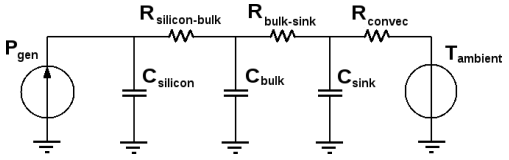


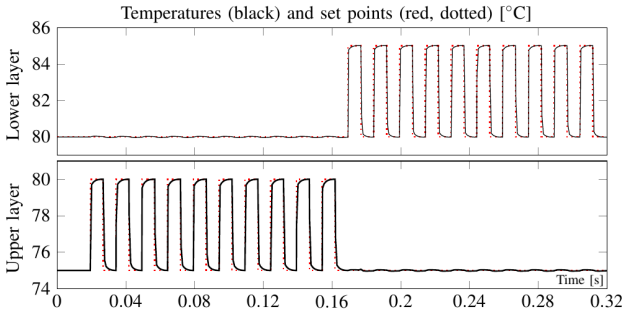


- In control-ese: the smaller the scale, the faster the fastest dynamics...
- ...but the larger the separation of it from the slower ones:
- smaller active volumes need faster control, but are more decoupled.

Control synthesis

(1/2)

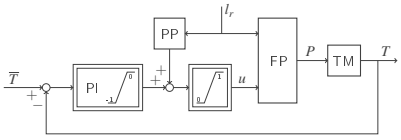




- Three main dynamics, the fastest being the relevant one.
- Actuator: DVFS command, the only one fast enough.
- Decentralised *high-bandwidth* control feasible for the harshest unit coupling.
- PI structure adequate.

Control synthesis

(2/2) – integration with Power/Performance DVFS management

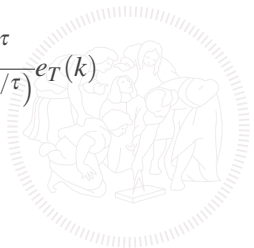


- Process (TM):

$$P(s) = \frac{\Gamma}{1+s\tau}$$

- PI controller:

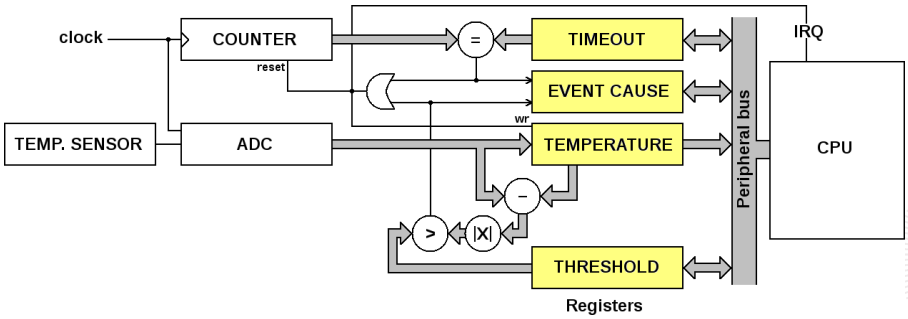
$$\begin{cases} x_R(k) = & x_R(k-1) + \dfrac{\mu T_s}{\Gamma} e_T(k-1) \\ u_R(k) = & x_R(k) + \dfrac{\mu T_s e^{-T_s/\tau}}{\Gamma\left(1-e^{-T_s/\tau}\right)} e_T(k) \end{cases}$$



Event-based realisation

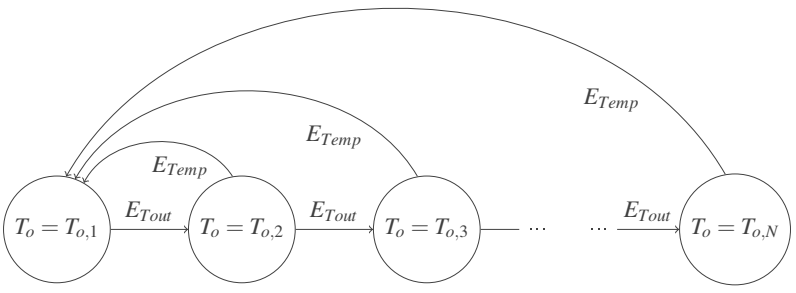
Controller and event triggering

- Stability guaranteed as per Leva and Papadopoulos, 2013.
- PI realised in software (39 clock cycles on average).
- Hardware event generator (send on delta plus timeout):



Event-based realisation

Timeout management



- Timeout increased when elapsed without temperature events, up to a maximum,
- and reset to minimum in the case of a temperature event (temperature differs from last control event more than a prescribed Δ).

The addressed architecture

A somehow (but not that much) futuristic one

- 24-core tiled 3D MPSoC;
- two silicon layers;
- 12 cores per layer, grouped in tiles of three.
- Tiles comprehend a shared L2 cache and a NoC router;
- routers organised in a 3D-mesh NoC;
- four routers per layer.

- Definitely a tough situation.

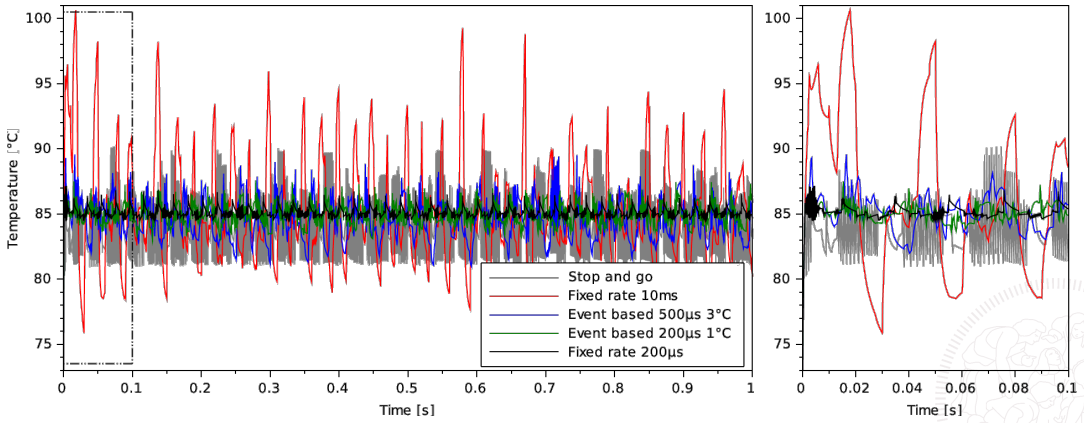
The compared control policies

- Per-core stop-and-go via clock gating (baseline);
- fixed-rate (FR) PIs – one per core – at 200μs;
- fixed-rate PIs at 10ms;
- event-based (EB) PIs with a triggering temperature variation ΔT of 1°C and a minimum inter-event time q of 200μs;
- event-based PIs with $\Delta T = 3$°C and $q = 500\mu$s.

- Benchmarks taken from the MiBench suite.

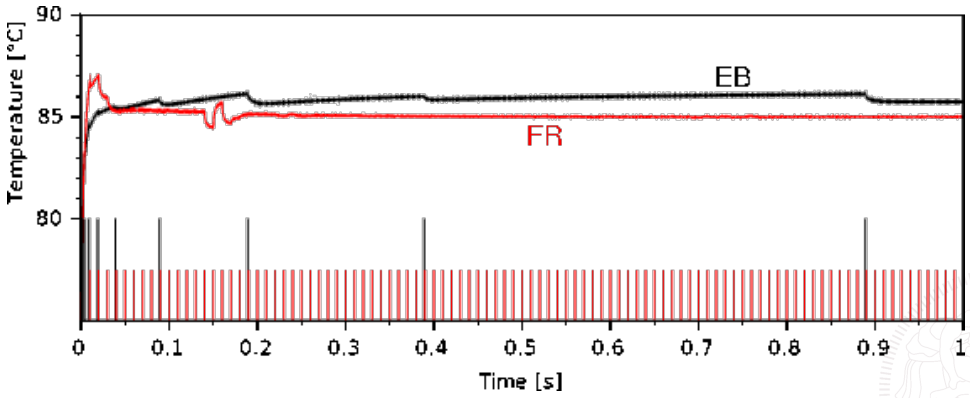
MiBench results

Core 0 temperature with `bitcount`



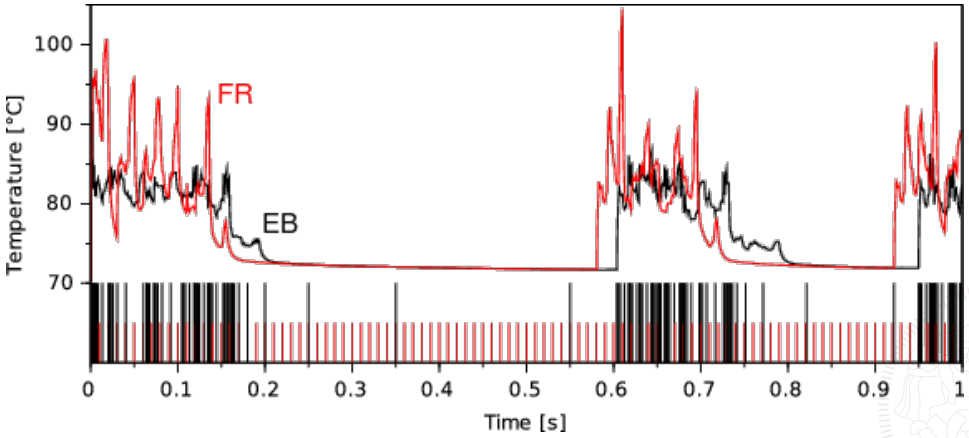
MiBench results

FR @ 10ms vs. EB @ $3°C$, $500\mu s$ on `stringsearch`: core 0 temperature and events



MiBench results

Same policies on `bitcount` interleaved with sleeps: core 0 temperature and events



MiBench results

Comparative figures

- Average events per second:

	basicmath	bitcount	crc32	fft	sha1	stringsearch
FR 200μs	5000	5000	5000	5000	5000	5000
EB 200μs/1°C	71	1153	387	218	958	28
EB 500μs/3°C	14	320	212	17	472	13
FR 10ms	100	100	100	100	100	100
Stop and go	5000	5000	5000	5000	5000	5000

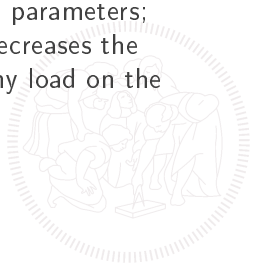
- Benchmark completion times (ms):

	basicmath	bitcount	crc32	fft	sha1	stringsearch
FR 200μs	+3%	+3%	+10%	+6%	+4%	+2%
EB 200μs/1°C	+2%	+1%	+1%	+2%	+4%	+2%
EB 500μs/3°C	8.453	40.683	+1%	88.872	+3%	0.220
FR 10ms	+1%	+1%	198.840	+4%	32.203	+1%
Stop and go	+24%	+34%	+8%	+23%	+6%	+24%

Counterpart – Linux governors

Preliminaries (1/2) from `https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt`

- Five governors are defined.
- Four allow very limited control from userspace:
  - `performance` sets the CPU statically (in CS-ese, constant) to the highest frequency within scaling_min_freq and scaling_max_freq;
  - `powersave` sets the CPU statically to the lowest frequency;
  - `ondemand` sets the CPU "depending on the current usage" with a heuristics inessential to discuss here, see the quoted URL for the many available parameters;
  - `conservative` is similar to `ondemand` but "gracefully increases and decreases the CPU speed rather than jumping to max speed the moment there is any load on the CPU".



Counterpart – Linux governors

Preliminaries (2/2) from `https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt`

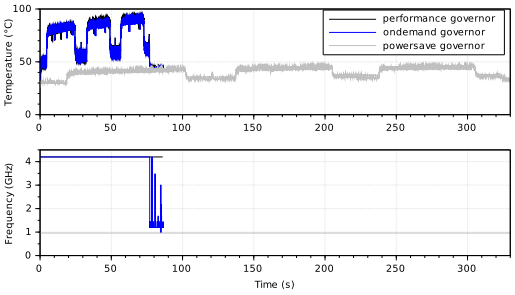
- One is built to be accessed from userspace:
  - `userspace` allows any program running with UID "root" to set the frequency.
- However `userspace` acts through `sysfs`, which makes it unacceptably slow and useless for the actuation speed we need.

- We take governors, plus the hardware thermal protection, as the state of the art for power/performance management.
- We test against `performance`, `powersave`, and `ondemand`.

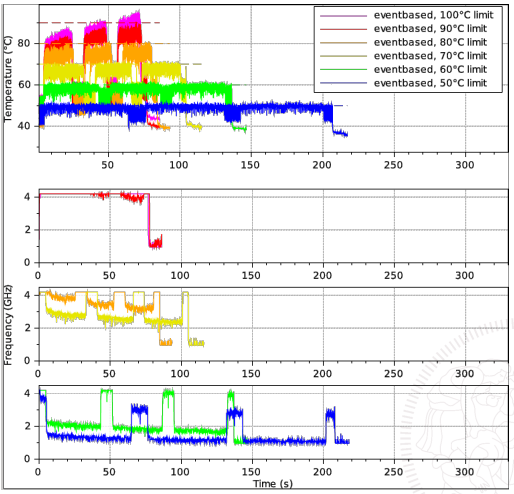
Counterpart – Linux governors

Benchmark taken from the LINPACK suite

Linux governors



Event-based controller



We have thermal control integrated, and can move with continuity between "powersave" and "performance" by acting on the temperature threshold.

Counterpart – the Intel `thermald` daemon

Preliminaries (1/2) from `https://01.org/linux-thermal-daemon/`

- `thermald` provides a Linux user mode daemon to system developers, reducing time to market with controlled thermal management using P-states, T-states, and the Intel power clamp driver. The Thermal Daemon uses the existing Linux kernel infrastructure and can be easily enhanced.
- The project is for system developers who want to enable application developers and their customers with the responsive and flexible thermal management, supporting optimal performance in desktop, clamshell, mobile and embedded devices.

Counterpart – the Intel thermald daemon

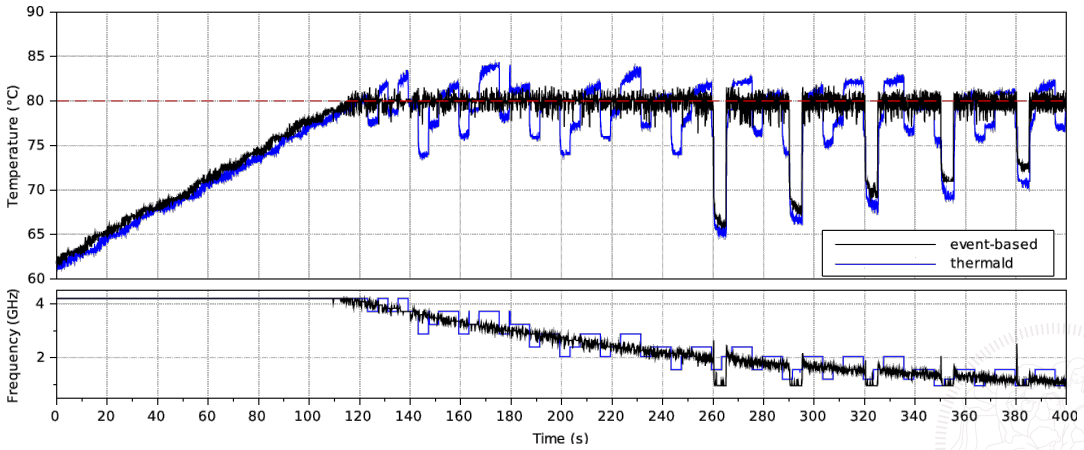
Preliminaries (2/2) from https://01.org/linux-thermal-daemon/

- This is an active open source project distributed under the LGPL open source license. With a mature and established codebase containing about 12,000 lines of code.
- Linux Thermal Daemon is currently used in distributions such as Ubuntu and Fedora and can be used any Linux-based system, including Chromium, Chrome OS or Android.

- We take thermald as the current evolution of the state of the art.

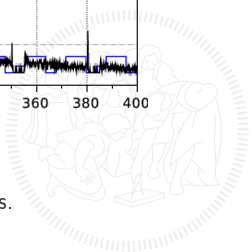
Counterpart – the Intel thermald daemon

Load given by cpuburn



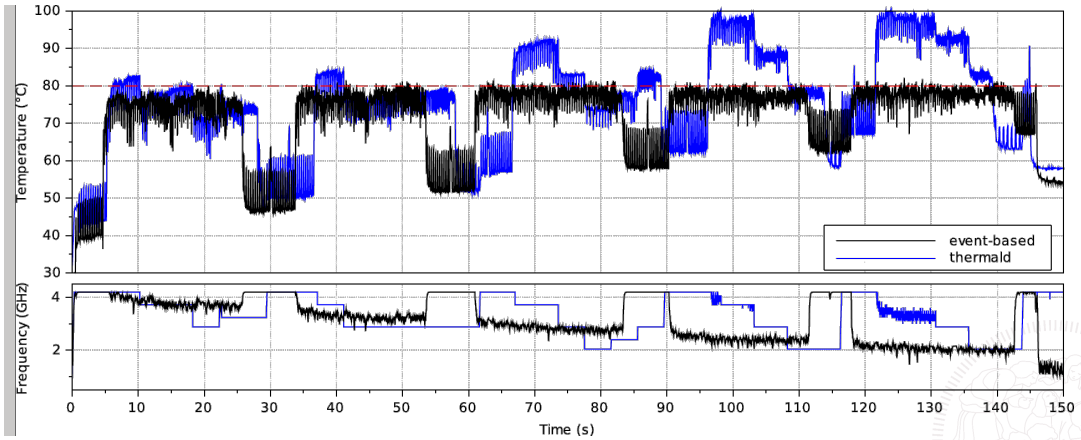
Constant load and power if not for some interruptions:

both work, but thermald suffers from the actuation speed limit of OS modules.



Counterpart – the Intel `thermald` daemon

Benchmark taken from the LINPACK suite



Constant load, variable power (cache miss rate):

as sink heats up, `thermald` cannot keep the disturbance pace (hw protection engages).

Retrospect

- Problem tractable with system-level models:
- no need for fine-grained thermal simulation,
- nor for cycle-accurate representation of the CPU behaviour.
- Simple and robust solution with a *decentralised* scheme;
- formal stability proof;
- advances over the state of the art, proven on real hardware.

Directions

- Per-core DVFS to enable multivariable control;
- SoCs, NoCs, and the like;
- characterisation of load-induced disturbances as stochastic processes;
- integration with larger (e.g., rack-level) thermal/energy management.

Self-adaptive software (some cases)

Engineering "self-adaptive" software

Motivations

- Software is nowadays required to preserve Quality of Service (QoS) in the presence of varying
  - deployment infrastructure,
  - usage profiles,
  - third-party component behaviours,
  - ...

⇒ SOA (Service Oriented Architectures): composition of abstract services, each possibly provided by multiple implementations.

⇒ Binding a concrete implementation to the abstract service is crucial for SOA systems.

Engineering "self-adaptive" software

Drivers

- Dynamic binding can be driven by many factors. We see (the major) two:
  - reliability – bind to maintain a certain "success rate";
  - performance – bind to maintain s certain "service level".
- In both cases, one has to avoid over-provisioning to save resources and money.
- We want to view the *scenario* with dynamic models (DT and CT)...
- ...although this time uncertainty matters.

Introduction

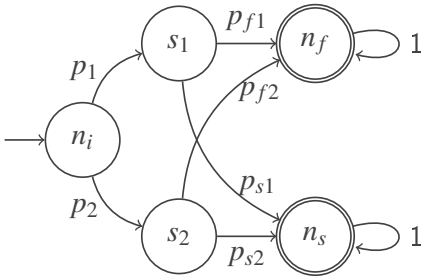
- Dynamic biding driven by "reliability", i.e.,
  probability of successfully accomplishing an assigned task
  (whatever "success" means).

- Problem re-stated as a set point tracking one in the presence of disturbances.
- Context: discrete-time dynamic systems.

$\Rightarrow$ Initial modelling effort but simple and general solutions.

Dynamic binding

Preliminaries – a typical DTMC scheme



- State: node queued requests $\mathbf{n}(k) = [\nu_i(k)\, \nu_1(k)\, \nu_2(k)\, \nu_s(k)\, \nu_f(k)]'$.
- Control input: one dispatch probability, here $p_1(k)$.
- Disturbance(s): input rate(s), here $w_i(k)$.
- Variability: services' reliabilities, here p_{s1} and p_{s2}.
- Controlled variable: overall reliability.

Model of the controlled system

- The model is

$$\begin{aligned} \mathbf{n}(k) &= \mathbf{n}(k-1) - \mathbf{r}(k-1) + \mathscr{P}(k-1)\cdot\mathbf{r}(k-1) + \mathbf{w}(k-1) \\ \mathbf{r}(k) &= \min\{\mathbf{t_m}, \mathbf{n}(k)\} \end{aligned}$$

  where $\mathbf{r}(k)$ are the served requests, and $\mathscr{P}$ the transition matrix of the chain.
- In the addressed case, having $\mathbf{y} = v_s$ as output,

$$\mathscr{P}(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ p_1(k) & 0 & 0 & 0 & 0 \\ 1-p_1(k) & 0 & 0 & 0 & 0 \\ 0 & 1-p_{s1} & 1-p_{s2} & 1 & 0 \\ 0 & p_{s1} & p_{s2} & 0 & 1 \end{bmatrix}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{n}(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{n}(k).$$

Completing the model

- Indicating by $q(k)$ the measured reliability, the measurement model is

$$q(k) = \frac{v_s(k) - v_s(k-1)}{v_s(k) + n_f(k) - v_s(k-1) - v_f(k-1)}$$

- If p_{s1} and p_{s2} are constant the compound model is NL but TI.
- Assumption 1: probabilities have small and slow fluctuations plus sporadic large abrupt changes.
- Assumption 2: reliable probabilities' estimates are available.

- This allows to *automatically tune* the controller (on demand and/or on error threshold).

A tuning-oriented model

- Reduced transition matrix (disregard absorbing states)

$$\mathscr{P}_{red}(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ p_1(k) & 0 & 0 & 0 & 0 \\ 1-p_1(k) & 0 & 0 & 0 & 0 \\ 0 & 1-p_{s1} & 1-p_{s2} & 0 & 0 \\ 0 & p_{s1} & p_{s2} & 0 & 0 \end{bmatrix}$$

- Corresponding reliability re-definition

$$q(k) = \frac{\nu_s(k)}{\nu_s(k)+\nu_f(k)}$$

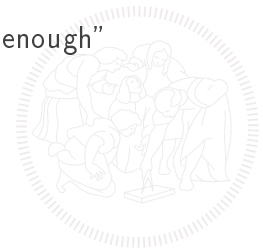
A tuning-oriented model

- This makes the chain model algebraic, thus by finding equilibrium and linearising one has

$$P(z) = \frac{\Delta Q(z)}{\Delta P_1(z)} = \frac{p_{s2}-p_{s1}}{z^2}$$

- Very simple structure, *but gain sign may change*.

- Hence, automatic tuning is disabled if the estimated gain is not "far enough" (configuration parameter) from zero.
- In the opposite case, an autotuning PI can be used.



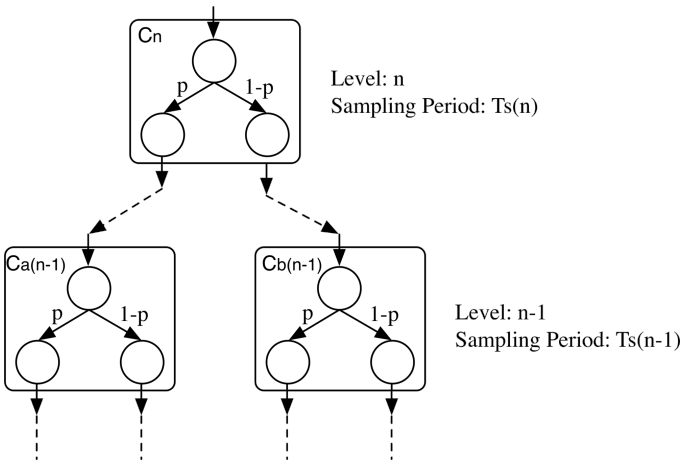
Control and its tuning

- Goal: reliability $\bar{q}$.
- Scheme: PI with reliability set point $\bar{q}(k)$ "slightly greater" than $\bar{q}$.
- Choice of the autotuning method:
  - better keep loop closed,
  - avoid stepwise *stimuli*,
  - and forecast closed-loop transients to quantify "slightly greater" above.

$\Rightarrow$ Contextual autotuning (Leva *et al.*, 2010) with relay plus integrator test.

Extension to the N-level case

Scheme having the two-lavel case seen so far as basic brick

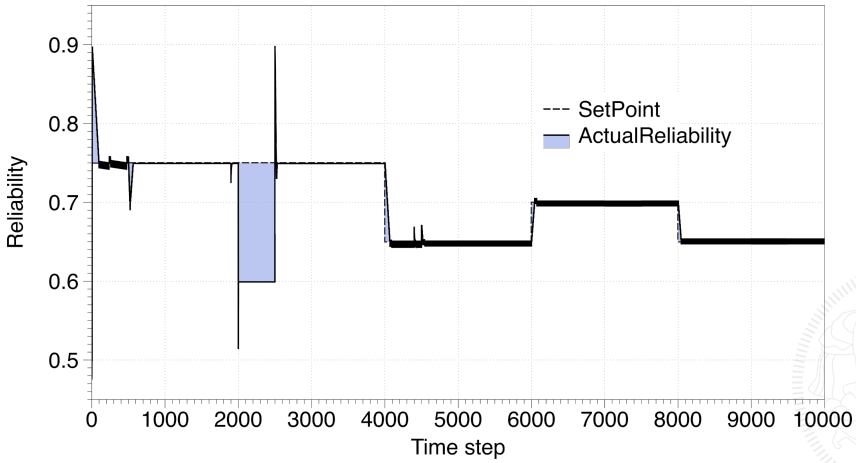


- Multirate scheme, analysis still underway.
- Monolithic alternative to be studied: less modular but maybe easier to assess.

Implementation and operation

Only one example for brevity

- Four-level binder with autotuning and various *stimuli*.



Performance-driven software adaptation

Preliminary analysis and problem statement

- Context: a service providing system modelled as a queuing network made of
  - servers, that store incoming jobs in their queue and process them,
  - and routers, that distribute incoming request to servers with certain probabilities.
- Goal:
  - process all the incoming jobs at the maximum service level
  - in the presence of server resource limitations,
  - and robustly in the face of changes, e.g., in the job exit probability;
  - provide graceful degradation if the goal is infeasible.

Performance-driven software adaptation

Controlled system model

- Given the huge rate of incoming jobs, we can describe a server with a continuous-time model.
- Denoting by $r_i(t)$ and $r_o(t)$ the input and output rate of a server and by $n(t)$ its queue occupation, we have
$$\dot{n}(t) = r_i(t) - r_o(t),$$
- A network with N nodes is seen as a continuous-time dynamic system of order N, with state variables $n_k(t)$, $k = 1 \ldots N$, i.e.,
$$\dot{\mathbf{n}}(t) = \mathbf{B_o}(\mathbf{p}(t))\mathbf{r_o}(t) + \mathbf{B_i}\mathbf{r_i}(\mathbf{t})$$
- Matrix $\mathbf{B_o}$ depends on some vector $\mathbf{p}(t)$ of time-varying routing probabilities $\Rightarrow$ LPV system.

Performance-driven software adaptation

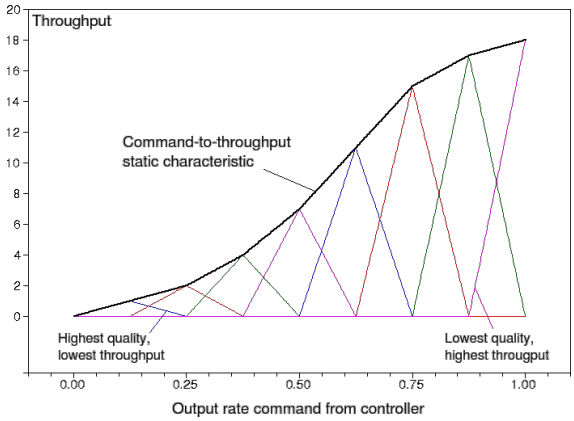
Control synthesis (1/2)

- Vector $\mathbf{p}(t)$ lies in a convex hull.
- Hence, any feedback control system with either an LTI controller, or an LPV one not introducing additional varying quantities besides $\mathbf{p}$, is asymptotically stable if those on the hull vertices are asymptotically stable with the same stability degree.
- A *sufficient* condition for the existence of a solution is that all the eigenvalues of the $\{\mathbf{A}_{cl,v}\}$ matrices be real negative for any $\mathbf{p}$.
- In practice, we set up a decentralised scheme with an independent controller per server.

Performance-driven software adaptation

Control synthesis (2/2)

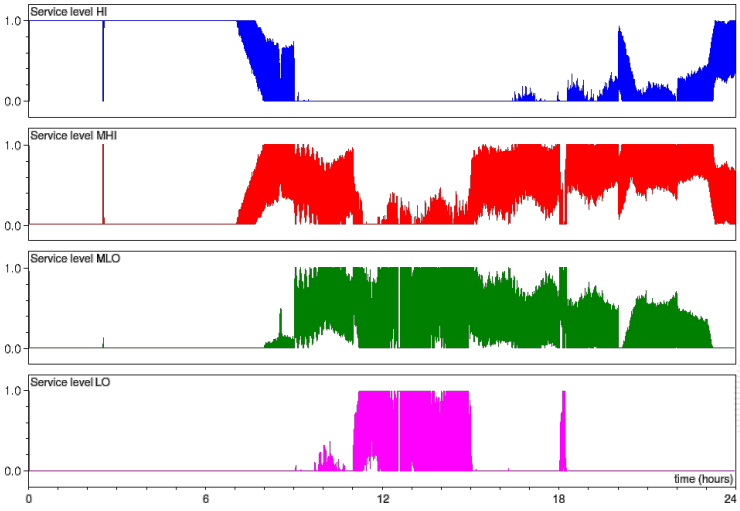
- Local controllers act on their $r_o(t)$ to keep $n(t)$ at $n^o(t)$:
  - once you control queue lengths, throughputs follow;
  - measuring a length requires no time window assumptions.
- To manage service levels, split-range actuation is used:



Performance-driven software adaptation

A simulation example

- Server with four service levels:



Retrospect

- Dynamic binding formulated as a discrete-time feedback control;
- automatic controller tuning techniques devised and applied when convenient;
- reliability binding strategy validated through simulations and subsequently implemented in Java, both within the Spring framework and as a standalone application;
- performance binding strategy (to date) just simulated;
- results are encouraging.

Future work

- Address different performance/quality metrics;
- implement in distributed environments hosting mutually related software components;
- bring in full-fledged multivariable control methodologies, and their adaptation.

Hands-on practice

Interactive session

Whiteboard and computer, no slides

- A few more control synthesis exercises;
- modelling, simulating and controlling queues;
- question time and discussion.

- NOTE: be ready for tomorrow's overall question time.

– Day 5 –

Outline

- **Morning**: *feedback loops in computers*
  - Peculiarities;
  - control vs. control-based design;
  - main course takeaways;
  - coverage of the presented system class and outlook;
  - discussion.
- **Afternoon**: *overall recap & question time*
  - Four questions and four answers;
  - conclusions;
  - references;
  - about the exam;
  - discussion, question & proposal time.

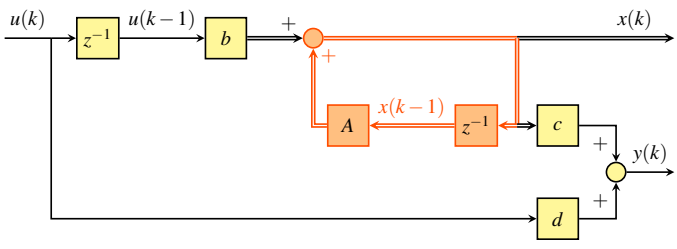
Feedback loops in computers

Is feedback just for control?

- NO. Feedback is inherent to ANY dynamic system.
- Let us see an example. Take the DT LTI system

$$\begin{cases} x(k) &= Ax(k-1)+bu(k-1) \\ y(k) &= cx(k)+du(k) \end{cases}$$

and writing it as block diagram, you see the loop:



Indeed...

A famous quote

on "creation"

"I'm personally convinced that computer science has a lot in common with physics. Both are about how the world works at a rather fundamental level.
The difference, of course, is that while in physics you're supposed to figure out how the world is made up, in computer science you create the world.
Within the confines of the computer, you're the creator. You get to ultimately control everything that happens.
If you're good enough, you can be God. On a small scale."

Linus Torvalds

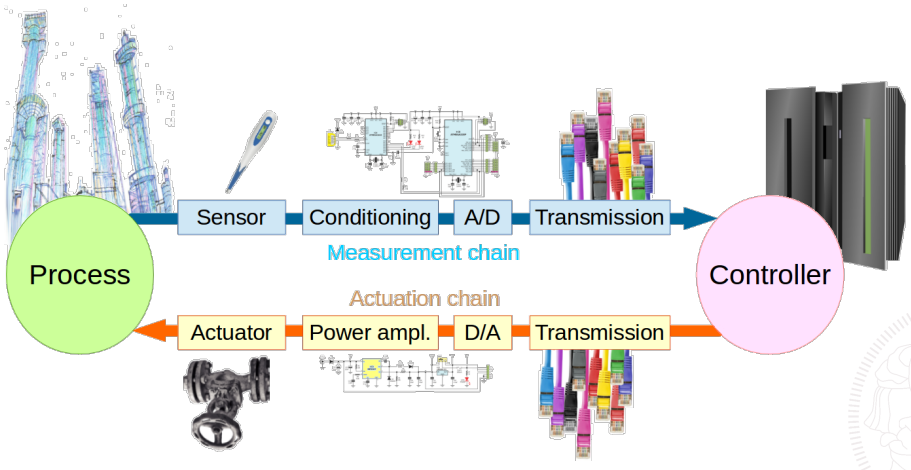
Consequences

- The "nature of physics" in CSE-related control problems may be *very* peculiar.
- As a result, control loops in computing systems can be peculiar as well, in several senses.

- Let us have a look at possible peculiarities,
  - to grasp the big picture (an exhaustive treatise is infeasible),
  - and in perspective, to set forth new research topics.

Peculiarity 1

A simplified picture matching virtually any control application but most CSE-related ones



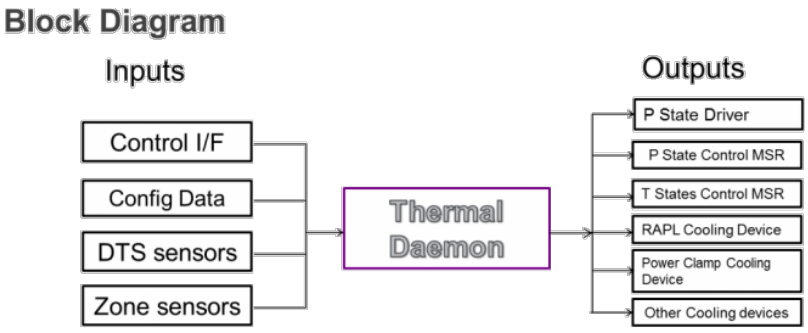
Difference between non-CSE and CSE-related control problems

valid in general – i.e., there may be *some* exception, but statistically irrelevant

- Non-CSE:
  - the nature of measured/governed quantities is clear (e.g., pressure, valve opening...);
  - the nature of sensors/actuators follows unambiguously if not for technological choices (e.g., for temperature, NTC vs. thermocouple).
- CSE:
  - none of the above is true in general—in fact, much of the above quite often does not apply;
  - on the contrary, a relevant part of the problem is selecting or creating sensors and/or actuators.

Example

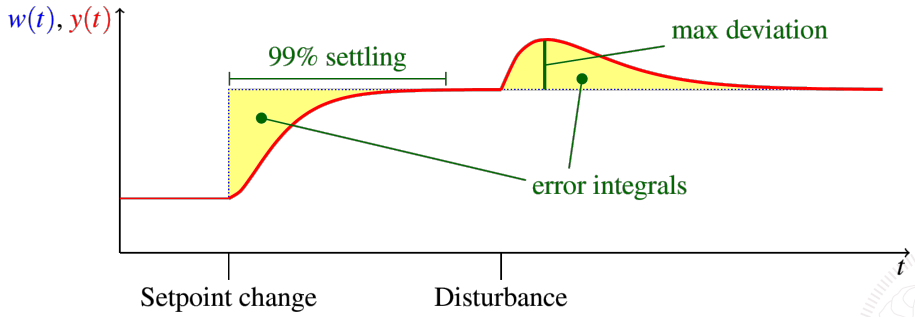
`thermald` – which (combination of) sensor(s) and actuator(s)?

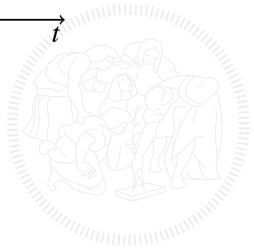


Taken from
https://01.org/linux-thermal-daemon/documentation/introduction-thermal-daemon

Peculiarity 2

How control quality is assessed in non-CSE applications (examples)





Difference between non-CSE and CSE-related control problems

valid in general

- Non-CSE:
  - control quality is judged based on set point tracking, disturbance rejection, or integral indices (e.g., the ISE, or Integrated Squared Error).
- CSE:
  - the above is not true in general—in fact, quite often does not apply;
  - and another relevant part of the problem is agreeing metrics that can be quantified so as to be used for feedback control.

Peculiarity 3

How timing for control is managed

Difference between non-CSE and CSE-related control problems

valid in general

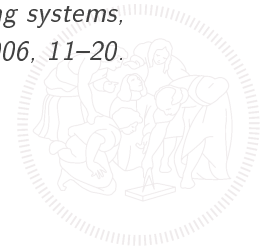
- Non-CSE:
  - strict timing;
  - quite often, specialised busses distributing the clock, or some means of synchronisation;
  - everything is timestamped.
- CSE:
  - accessing information (sensing) via different paths (e.g., /proc, MSRs,...);
  - not easy to avoid unpredictable delays and to ensure precise timestamping;
  - many "sensors" were not conceived for control...

Peculiarity 4

On the "nature of physics" – a bit more detail

"A computing system [...] is a man-made artifact whose internal behaviour is not governed by any laws of nature, at least not on the macroscopic level. This means that it is, generally, not possible to derive any first principles models."

K.E. Årzén et al.,
Conclusions of the ARTIST2 roadmap on control of computing systems,
SIGBED Review ***3****(3), 2006, 11–20.*



Difference between non-CSE and CSE-related control problems

valid in general

- Non-CSE:
  - physics is basically the one outside the computer;
  - first principles rule.
- CSE:
  - physics is also that inside the computer;
  - at the lowest level it can be ruled by first principles (think of the task pool model we used for scheduling),
  - but there can be software management layers made of heuristics, designed directly as algorithms and therefore hard to describe with dynamic systems.

Summarising

What makes it difficult to use controls (in the sense viewed here) for CSE problems

- May need to define sensors and actuators;
- may have problems translating "lexical" specifications into requirements in the SC sense;
- may come across software components (for sensing and actuating) unfit to guarantee adequate timing, and cumbersome to modify for that purpose;
- may be forced to pass through software layers that do exert the require action, but expose an interface not easy to connect to controllers;
- ...

Summarising

How one can use feedback control

- To close loops around an already functioning system.
  - Example: changing the priorities of a scheduler, that functions also in the absence of control;
  - quite likely to incur in the problems just listed.

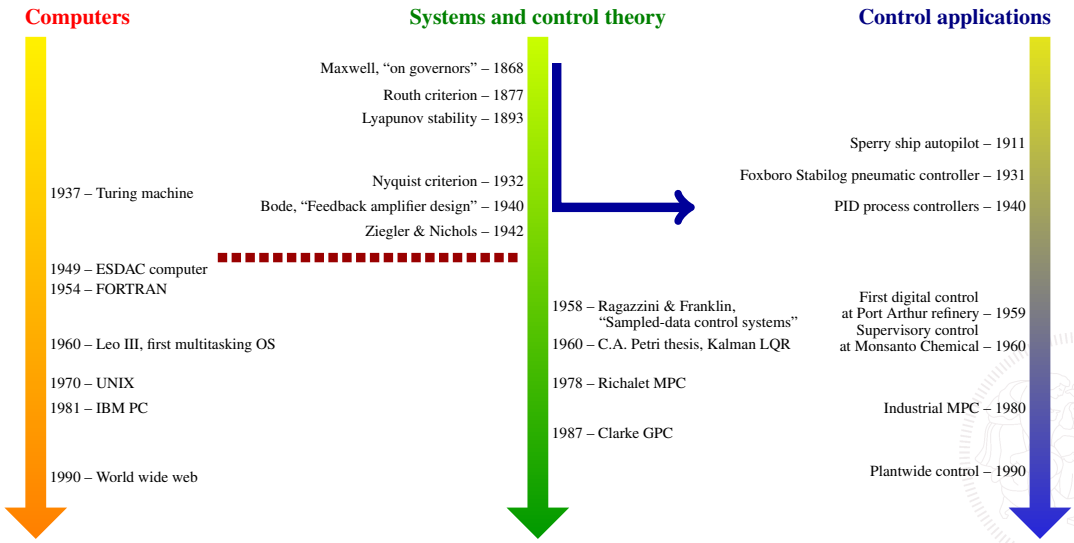
  $\Rightarrow$ This is computing systems control.
- To design part of a system as controllers.
  - Example: our scheduler, in the absence of which the system does not function;
  - less likely to incur in the problems just listed.

  $\Rightarrow$ This is control-based computing systems design.

- In the last days we did both things. Let us now reason on this matter a bit more in depth.

History of a competence gap (just a few markers)

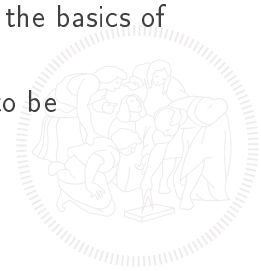
Sometimes theory was there at application time, sometimes it was not



History of a competence gap

resulting in two parallel lives

- As for "industrial" (basically CT) control, when applications became demanding the theory was there to use.
- The same did not happen for "digital" (simplifying, DT) control: the theory grew up in the same years as computers, and CSE specialists had to do without.
- The result is two communities with quite a bit of communication difficulties.
- You may have known this also before the course, but now you know the basics of "the other" viewpoint.
- Let us therefore re-consider some problems (we see one, the others to be discussed) in the light of the so gained double personality.

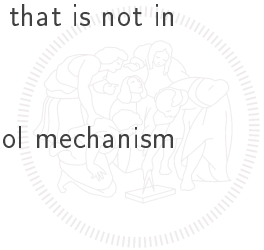


Example

A hypothetical short tale from industrial control

A certain number of **tanks** need receiving a given amount of **fluid** within a specified period. Once each of them has received enough **fluid**, be it within the deadline or not, it **is emptied** and put in a waiting state, and starts over asking for the same amount of **fluid** at the beginning of the next period. **Tanks** are managed in a certain number of queues depending on their priority level, and when moved from a queue to another, they enter the latter in the last position. At any given time only one of them can receive **fluid**, and is selected as the first one of the highest-priority queue that is not in the waiting state.

To ensure that each **tank** receives enough **fluid** within its period, a control mechanism is to be introduced having as control variables the **tank** priorities.

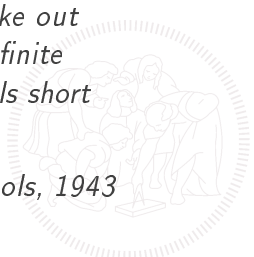


Example

Another quote

In the application of automatic controllers, it is important to realize that controller and process form a unit; credit or discredit for results obtained are attributable to one as much as the other.
A poor controller is often able to perform acceptably on a process, which is easily controlled. The finest controller made, when applied to a miserably designed process, may not deliver the desired performance.
True, on badly designed processes, advanced controllers are able to eke out better results than older models, but on these processes there is a definite end-point which can be approached by the instrumentation and it falls short of perfection.

J.G. Ziegler and N.B. Nichols, 1943



Example

Activate your new SC personality...

...and reconsider the tank system.

- Why act on priorities?
- They do influence the way tanks receive fluid,
- but the relationship is so indirect, and hard to model!

- Is this an "easily controlled" or a "miserably designed" process?
- Why not remove the queue and priority stuff, read the tank levels, and act on the valves directly?

- In general: are we addressing all and only the relevant phenomenon, using the right sensors and actuators?

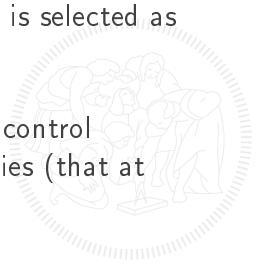
Example

Now switch back to the CS attitude for a re-interpretation

Tanks reloaded

A certain number of **tasks** need receiving a given amount of **CPU time** within a specified period. Once each of them has received enough **CPU time**, be it within the deadline or not, it **has the CPU time count reset** and is put in a waiting state, and starts over asking for the same amount of **CPU time** at the beginning of the next period. **Tasks** are managed in a certain number of queues depending on their priority level, and when moved from a queue to another, they enter the latter in the last position. At any given time only one of them can receive **CPU time**, and is selected as the first one of the highest-priority queue that is not in the waiting state.

To ensure that each **task** receives enough **CPU time** within its period, a control mechanism is to be introduced having as control variables the **task** priorities (that at present are managed by heuristics).



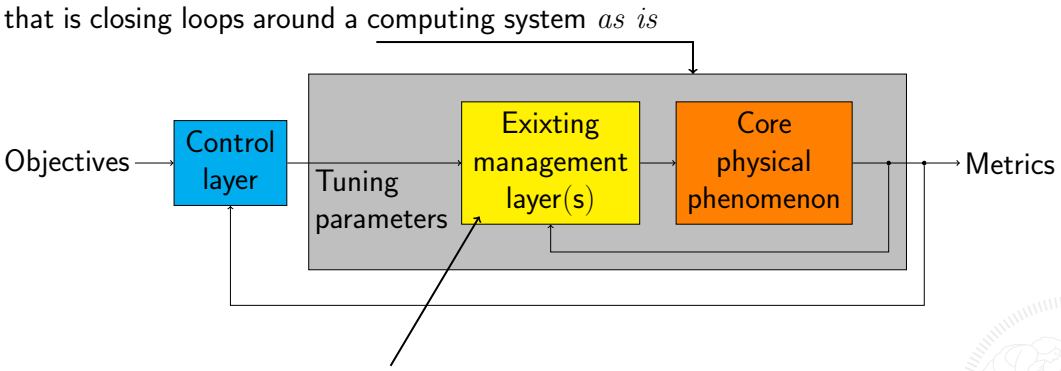
Example

Finally, scale up in abstraction and look at the two viewpoints from above

- The "tanks reloaded" tale is in fact one of the many flavours of (uniprocessor) "multilevel dynamic priority scheduling".
- More in general, many computing system components are controllers in nature, but are not designed as such.
- Brutalising for brevity, we need to reconcile two design attitudes.
  - CSE: problem ⇒ algorithm ⇒ (benchmark) testing;
  - SC: problem ⇒ dynamic model ⇒ assessment by analysis ⇒ algorithm specification.

Ultimate goal: widen the perspective

The starting point is computing systems control

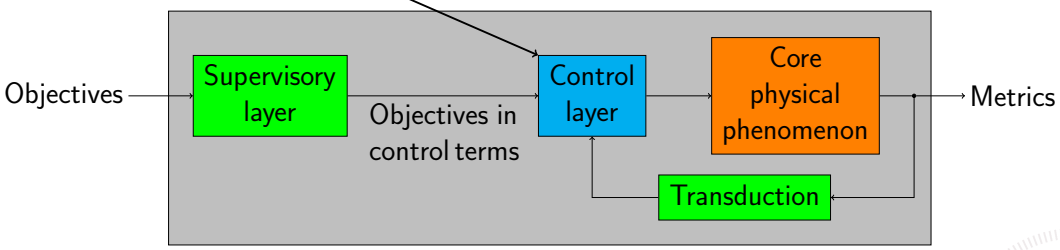


and where layers designed directly as algorithms, not as dynamic systems, often create physics that is difficult to model and control

Ultimate goal: widen the perspective

The target point is control-based computing systems design

that makes control loops vital for the computing system



and where at the cost of some redesign, but often with simple controllers, relevant properties can be achieved and formally guaranteed.

This is the *scenario* we and our research live in.
Let us now summarise our main takeaways
from this course.

Warning

- Sometimes we shall follow the topics covered, pretty much like when verifying a checklist.
- Sometimes we shall conversely re-combine ideas in a more comprehensive manner than we could do at the beginning of the course.
- We shall also just mention some new concepts in a view to talking about research directions later on, in our concluding discussion.

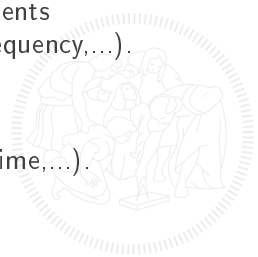
- If necessary, stop and ask questions at ANY time.

Takeaway #1

- The idea of control as conceived in the SC community...
- ...that we now review as a minimal taxonomy.

Terminology
(1/2)

- System – when required to avoid ambiguity, controlled system:
  the object or phenomenon to be governed
  (a server, the thermal behaviour of a CPU,...).
- Requirements:
  what you want the system to do
  (stay below 100°C, process this number of frames per second, respect an SLA,...)
- Controls or control actions or simply actions:
  what is done to the system with the purpose of fulfilling the requirements
  (boot/shutdown VMs, adjust the fan speed and/or the CPU clock frequency,...).
- Outcomes:
  the actual behaviour of the system
  (real fps, SLA adherence/violation, real temperature behaviour over time,...).



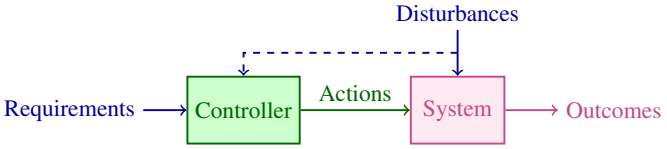
Terminology
(2/2)

- Disturbances:
  anything that affects the outcomes and cannot be manipulated, but possibly sensed
  (a user request burst, a VM crash, the failure of a fan, the rack temperature,...);
  in other words, actions from the environment to which the system must be resilient.
- Sensors:
  the entities that gather information from the system
  (request and completion counters, temperature and fan motor current probes,...).
- Actuators:
  the entities that act on the system to exert the actions
  (hypervisor, fan motor drive, DVFS,...).
- Controllers:
  the entities that determine the actions given the information deemed relevant
  (the object of your design).

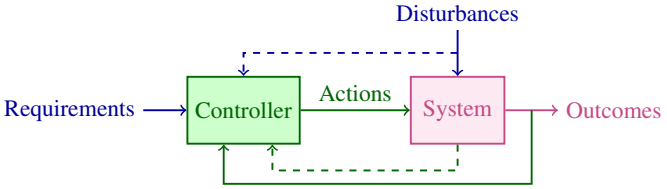
Taxonomy axis 1

What information is used by the controller – sensors and actuators not drawn for simplicity

- Requirements and possibly disturbances
  ⇒ open-loop control, possibly with disturbance compensation:

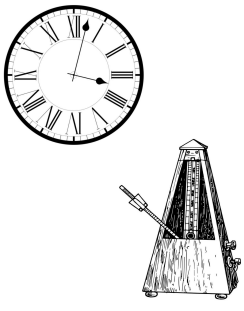


- Requirements, system output(s) and possibly disturbances
  ⇒ closed-loop or feedback control, possibly with disturbance compensation:



Taxonomy axis 2

When the controller determines the control action



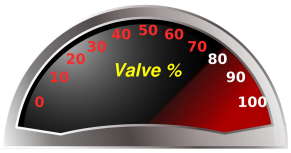
- Continuously ⇒ continuous-time control
  (e.g., analogue).

- At points known *a priori* ⇒ discrete-time control
  (e.g., ISR for a periodic interrupt).

- When requested ⇒ event-triggered control
  (e.g., every time a monitored signal varies by more than a given quantity).

Taxonomy axis 3

What is the nature of the control action



- Numeric, possibly quantised ⇒ modulating control (e.g., valve opening from 0 to 100%, CPU clock frequency in 50MHz steps,...).

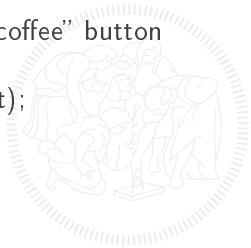
- Lexical (∼bool/enum) ⇒ logic control (e.g., gear, {forward,stop,backward}, Boost on/off,...).

Taxonomy axis 3

Remark

- In modulating control the requirements are normally that certain signals (variables varying over time)
  - follow a desired reference or set point (e.g., video rate = 30fps),
  - or stay in an admissible range (e.g., CPU temperature $< 100°$);
- hence in modulating control requirements are signals.

- In logic control the requirements are normally that certain input events
  - provoke certain sequences of response events (e.g., when I press the "coffee" button output a cup, then grind the coffee, then heat up water,...),
  - possibly with time constraints (...and make me a coffee in 30s at most);
- hence in logic control requirements are *not* signals.



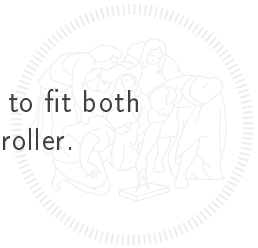
Summarising

brutally indeed

- Controller:

```
type                        = {modulating,logic}
timing                      = {continuous,discrete,event-triggered}
connection_with_system      = {open-loop,closed-loop}
disturbance_compensation    = {present,absent}
```

- There are corner cases to this taxonomy, but here we can safely neglect them.
- In complex systems, controllers of different nature co-exist.
- Here we focused mainly on DT control.



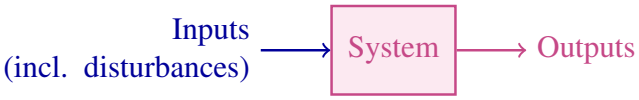
- In any case, to design and assess a controller, one needs a formalism to fit both the modelling of the controlled system, and the synthesis of the controller.
- That is why we started out with the concept of dynamic system.

Takeaway #2

- Dynamic systems—and at this point in the course, their relationship with control problems.

Dynamic system

General definition



- Suppose you know the values that the inputs have NOW.
- Can you tell which is NOW the values of the outputs?

- If so, the system is non dynamic, otherwise it is dynamic.

Continuous-time dynamic systems

In general

- They are made of differential equations.
- Input $u(t)$, state $x(t)$ with initial value x_0 at $t = t_0$, output $y(t)$;

$$\begin{cases} \frac{dx(t)}{dt} = f(x(t),u(t),t) & \text{[State equation]} \\ y(t) = g(x(t),u(t),t) & \text{[Output equation]} \end{cases} \qquad x(t_0) = x_0.$$

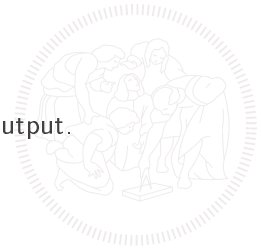
Discrete-time dynamic systems

In general

- An integer k is the "time index": it can really be time-related if it corresponds to the elapsing of a fixed period (giving rise to sampled-data systems) or just count the evolution steps (bus stops, scheduler interventions,...).
- Input $u(t)$k, state $x(k)$ with initial value x_0 at $k = k_0$, output $y(k)$;

$$\begin{cases} x(k) &= f(x(k-1), u(k-1), k) \quad \text{[State equation]} \\ y(k) &= g(x(k-1), u(k-1), k) \quad \text{[Output equation]} \end{cases} \qquad x(k_0) = x_0.$$

- Read: at each step
  - I take the old state an the old input, and compute the new state,
  - then I take the new state and the new input, and compute the new output.



Discrete-time LTI (Linear, Time-Invariant) dynamic systems

A specific but extremely useful subclass

- This is the announced class to discuss the full example at the end, after talking about feedback.
- Linear: functions $f(\cdot,\cdot,\cdot)$ and $g(\cdot,\cdot,\cdot)$ linear in x and u.
- Time-invariant: f and g do not depend explicitly on k
- Input $u(t)$k, state $x(k)$ with initial value x_0 at $k = k_0$, output $y(k)$;

$$\begin{cases} x(k) &= Ax(k-1) + bu(k-1) \\ y(k) &= cx(k) + du(k) \end{cases} \qquad x(0) = x_0.$$

- The time origin is zero because the system is TI.
- A, b, c and d are matrices of convenient dimensions.

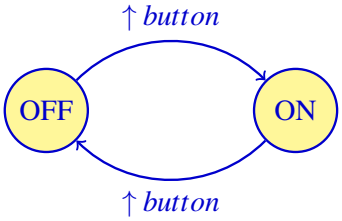
System representations and properties

- State space, transfer function (LTI), block diagrams;
- Stability of equilibria and systems (LTI), hidden parts.
- Relationships between CT and DT.

- This is what we concentrated upon. But there is more, like for example...

Discrete-event dynamic systems

- Not driven by time but rather by events.
- Represented as automata, Petri nets, and the like.
- Example automaton for the lamp:



- In this talk enough on this system class as well.
- Let us move to a *rough* (system class)–(control problem) pairing.

Dynamic systems classes and control problems

...in computers – just an overview on pairing to stimulate discussion and proposals

- Group 1 – problems directly related to physics *stricto sensu*:
  - typical example – CPU temperature;
  - system model – continuous-time dynamic system;
  - controller design – as continuous-time system, then converted to discrete-time (we do not treat this today).
- Group 2 – problems not in group 1 where requirements are translatable into desired behaviours of signals (reference or admissible range):
  - typical example – deadline enforcement, obtained by tracking a desired completion that reaches 100% by the deadline (many problems can be formulated this way);
  - system model – discrete-time dynamic system;
  - controller design – as discrete-time system.
- Group 3 – anything else:
  - mixed design strategy (we do not talk about this either),
  - or not suited for a control-centred design.
- The common idea for control design is feedback.

Takeaway #3

- Feedback and control synthesis.

Main topics

- Synthesis for set point tracking, disturbance rejection, or combinations thereof;
- major control laws – PI, deadbeat;
- structure-specific controllers, i.e., control laws the form of which is dictated by the model of the controlled system:
- CT design and discretisation.

Takeaway #4

- Examples:
  scheduling, clock synchronisation,
  thermal/power/performance management,
  self-adaptive software.

Takeaway #5

- Tools:
  Modelica, Maxima, Scilab...

An operational point of view

- Benefits of (feedback) control:
  - prescribe the response of a system to command inputs;
  - make that system insensitive to disturbance inputs;
  - reduce the effects of uncertainty;
  - reduce or eliminate the need to forecast exogenous actions on the system.
- Solid theory, proven effective in many fields
  (power, chemical, robotics, vehicles,...)
- Control already provides assurances/adaptiveness to *plants*:
- is it doing the best for the same purpose in computing systems?
- Yes and no.

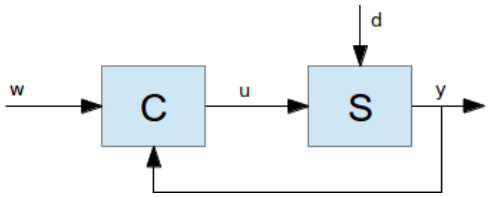
Why yes and why no?

- Yes, because
  - feedback can induce adaptiveness naturally;
  - in highly varying environments, reacting is better than attempting to forecast;
  - solutions can be assessed prior to deployment with extremely compact models.
- No, because
  - control methods require some model of the uncontrolled system;
  - the problem needs to be structured compatibly (not always immediate);
  - some solutions would require to re-design parts of the addressed system (otherwise, suboptimal performance).

- The real challenge is a common way of formulating problems, right from the system design phase...
- ...which calls in the first place for a shared jargon.

Jargon issues

Example 1/2 - "adaptive" vs. "state-dependent"



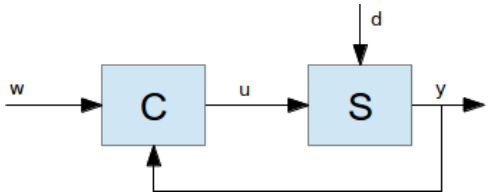
S system
C controller
w set point
y behaviour quality metric
u control action
d disturbance

- Role of **C**: make y follow w despite d, with no measurement of d.

- CSE jargon: **C** makes **S** adaptive.
- SC jargon: **C** and **S** form a dynamic feedback loop.

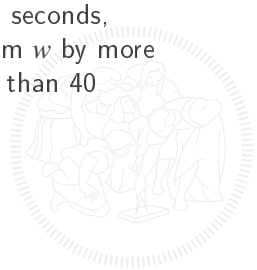
Jargon issues

Example 2/2 - "assurance" vs. "dynamic error bound"



S system
C controller
w set point
y behaviour quality metric
u control action
d disturbance

- Possible requirements:
  - when w varies in a step-like manner, y has to recover within 5% in 20 seconds,
  - a step-like d not exceeding 50 units must never cause y to deviate from w by more than 5 units, and y must then recover w within 0.01 units in no more than 40 seconds.
- CSE jargon: adaptation guarantees/assurances.
- SC jargon: dynamic error bounds.



Problem coverage of the presented system class

What can we do with DT (somentimes CT/DT) LTI models?

- Difficult to state in general.
- Better question: what do we need to exploit the approach?
  - quantitative performance metrics;
  - quantifiable actions;
  - first-principle (balance-like) dynamic relationships among them.

- Can we turn this into a rigorous characterisation of a problem class?
- Not yet.

Problem coverage of the presented system class

What we can say at the moment

- As far as the focus is restricted to the *scenario* just mentioned, control seems able to complement
  - a prescriptive paradigm,
    characterised by a mainly empirical objectives-to action/planning translation
  - with an abstract layer,
    reasoning on system properties instead of algorithms.

Research and technology perspectives

A set of open issues

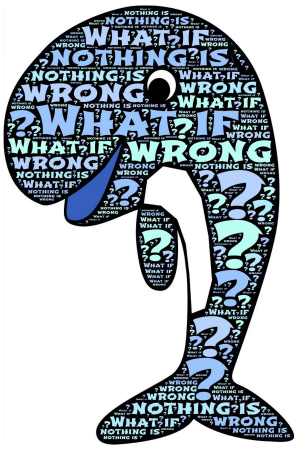
- At present, the most general statement we can make could sound like
  - when "adaptive" is well viewed as "state-dependent"
  - and "assurance" is well viewed as "dynamic error bound"...
  - ...then the theory of feedback control seems to help quite a lot,
  - especially when exploited for control-based design.
  - However, there are cases where such an approach leads to re-discussing part of the system.
- How really general is this?
- How to turn it into well assessed design guidelines/techniques?

- Addressing this matter is beneficial for both communities.
- Let us discuss.

Overall recap & question time

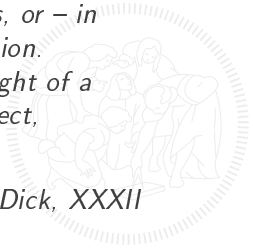
We somehow visited the control zoo

hence a couple of introductory words are in order



"Now the various species of whales need some sort of popular comprehensive classification, if only an easy outline one for the present, hereafter to be filled in all its departments by subsequent labourers. As no better man advances to take this matter in hand, I hereupon offer my own poor endeavours.
I promise nothing complete; because any human thing supposed to be complete, must for that very reason infallibly be faulty. I shall not pretend to a minute anatomical description of the various species, or – in this place at least – to much of any description.
My object here is simply to project the draught of a systematisation of Cetology. I am the architect, not the builder."

H. Melville, Moby Dick, XXXII



Four questions

to summarise and recap the path we went through in this course

- What is control and control-based design of computing systems?
- Why research on it?
- Results to date?
- Perspectives?

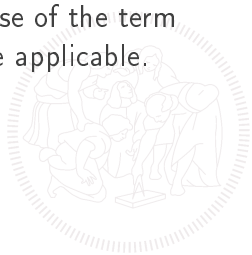
- Along the course we implicitly answered these questions.
- Let us now go for a final recap.

Four answers

...at least, to date

- What is control and control-based design of computing systems?

- Acknowledging that there is an "inside physics", some in the strict sense of the term and some "created", amd treating the matter with SC methods where applicable.

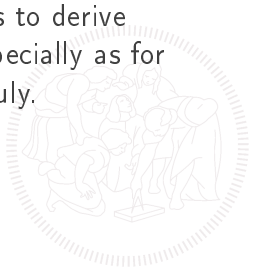


Four answers

...at least, to date

- Why research on it?

- Because specialising the SC theory to the encountered systems allows to derive powerful results, and the technological consequences are relevant, especially as for the ease in managing complexity, and in not creating complexity unduly.

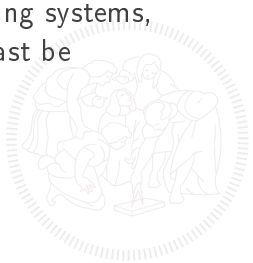


Four answers

...at least, to date

- Results to date?

- We showed some, and we can say that the deeper one digs into existing systems, the more frequently he/she encounters components that should at least be re-considered with a SC-centric attitude.



Four answers

...at least, to date

- Perspectives?

- Many, especially in those domains where complexity is increasing most rapidly.

At the end of the course

- We provided a basic, yet consistent introduction to the systems and control theory (centred primarily, but not exclusively on the discrete-time framework);
- we revisited some relevant problems related to computing systems design, at various levels, with a system- and control-theoretical attitude;
- we presented some results obtained with the proposed approach;
- we highlighted the advantages that the approach yields – where applicable – over heuristic techniques.

- the instructor hopes that this can foster a better and methodologically grounded cooperation between the SC and the CSE communities.

Next steps

- Quick tour of some references;
- about the exam;
- question & proposal time.

A couple of introductory words

- No exhaustiveness claimed.
- The proposed references are (physiologically) centred on work by the instructor and co-authors.
- Hence consider also the contained literature reviews, and the references quoted therein.

On control and control-based design of computing systems

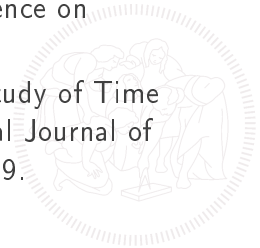
- J.L. Hellerstein, Y. Diao, S. Parekh, D.M. Tilbury, "Feedback Control of Computing Systems", Wiley-IEEE Press, Hoboken, NJ, 2004;
- K.E. Årzén, A. Robertsson, D. Henriksson, M. Johansson, H. Hjalmarsson, K.H. Johansson, "Conclusions of the ARTIST2 roadmap on control of computing systems", SIGBED Review **3**(3), 2006, 11–20.
- A. Leva, M. Maggio, A.V. Papadopoulos, F. Terraneo, "Control-based Operating System Design", IET, London, UK, 2012;
- P.K. Janert, "Feedback Control for Computer Systems ", O'Reilly, Sebastopol, CA, 2013;
- A.V. Papadopoulos, M. Maggio, F. Terraneo, A. Leva, "A Dynamic Modelling Framework for Control-based Computing System Design", Mathematical and Computer Modelling of Dynamical Systems **21**(3), 2015, 251–271.

On control-based scheduling & resource allocation

- A. Leva, M. Maggio, "Feedback Process Scheduling with Simple Discrete-time Control Structures", IET Control Theory & Applications **4**(11), 2010, 2331–2342.
- A.V. Papadopoulos, M. Maggio, S. Negro, A. Leva, "General Control-theoretical Framework for Online Resource Allocation in Computing Systems", IET Control Theory & Applications **6**(11), 2012, 1594–1602;
- M. Maggio, H. Hoffmann, M.D. Santambrogio, A. Agarwal, A. Leva, "Power Optimization in Embedded Systems via Feedback Control of Resource Allocation", IEEE Transactions on Control Systems Technology **21**(1), 2013,239–246.
- M. Maggio, F. Terraneo, A. Leva, "Task Scheduling: a Control-theoretical Viewpoint for a General and Flexible Solution", ACM Transactions on Embedded Computing Systems **13**(4), 2014, Article 76.
- A.V. Papadopoulos, M. Maggio, A. Leva, E. Bini, "Hard Real-Time Guarantees in Feedback-based Resource Reservations", Real-Time Systems **51**(3), 2015,221–246.

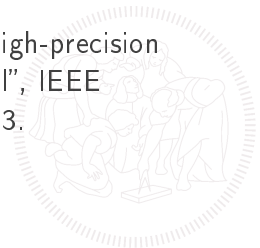
On clock synchronisation in WSNs
(1/2)

- M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "The Flooding Time Synchronization Protocol", Proc. 2nd international conference on Embedded networked sensor systems, Baltimore 2004, 39–49.
- J. Chen, Q. Yu, Y. Zhang, H. Chen, Y. Sun, "Feedback-Based Clock Synchronization in Wireless Sensor Networks: a Control Theoretic Approach", IEEE Transactions on Vehicular Technology **59**(6) 2010, 2963–2973.
- F. Ferrari, M. Zimmerling, L. Thiele, O. Saukh, "Efficient Network Flooding and Time Synchronization with Glossy", Proc. 10th International Conference on Information Processing in Sensor Networks, Chicago 2011, 73–84.
- S. El Khediri, N. Nasri, M. Samet, A. Wei, A. Kachouri, "Analysis Study of Time Synchronization Protocols in Wireless Sensor Networks", International Journal of Distributed and Parallel Systems 2012, arXiv preprint arXiv:1206.1419.



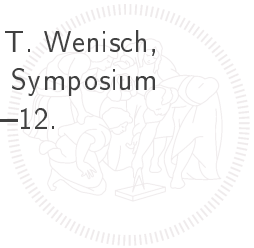
On clock synchronisation in WSNs
(2/2)

- F. Terraneo, L. Rinaldi, M. Maggio, A.V. Papadopoulos, A. Leva, "FLOPSYNC-2: Sub-μs, Sub-μA Clock Synchronisation for Wireless Sensor Networks", Proc. 35th IEEE Real-Time Systems Symposium, Roma 2014, 11–20.
- F. Terraneo, A. Leva, S. Seva, M. Maggio, A.V. Papadopoulos, "Reverse Flooding: Exploiting Radio Interference for Efficient Propagation Delay Compensation in WSN Clock Synchronization", Proc. 36th IEEE Real-Time Systems Symposium, San Antonio 2015, 175-–184.
- A. Leva, F. Terraneo, L. Rinaldi, A.V. Papadopoulos, M. Maggio, "High-precision Low-power Wireless Nodes Synchronization via Decentralized Control", IEEE Transactions on Control Systems Technology **24**(4), 2016, 1279–1293.



On thermal and power/performance management
(1/2)

- F. Zanini, C. Jones, D. Atienza, G. De Micheli, "Multicore Thermal Management using Approximate Explicit Model Predictive Control", Proc. 2010 IEEE International Symposium on Circuits and Systems (ISCAS), Paris 2010, 3321-–3324.
- H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, D. Burger, "Dark Silicon and the End of Multicore Scaling", Proc. 38th Annual International Symposium on Computer Architecture, San José 2011, 365—376.
- A. Raghavan, L. Yixin, A. Chandawalla, M. Papaefthymiou, K. Pipe, T. Wenisch, M. Martin, "Computational Sprinting", Proc. 18th IEEE International Symposium on High Performance Computer Architectures, New Orleans 2012, 1—12.



On thermal and power/performance management
(2/2)

- A. Leva, A. Papadopoulos, "Tuning of Event-based Industrial Controllers with Simple Stability Guarantees", Journal of Process Control 23(9), 2013, 1251–1260.
- A. Leva, F. Terraneo, W. Fornaciari, "Event-based Thermal Control for High-density Processors", Proc. 1st IEEE International Conference on Event-Based Control, Communication, and Signal Processing, Krakow 2015, 1–8.
- A. Leva, F. Terraneo, I. Giacomello, W. Fornaciari, "Event-based Power/performance-aware Thermal Management for High-density Microprocessors", IEEE Transactions on Control Systems Technology, 2017 (available online, DOI 10.1109/TCST.2017.2675841).

On self-adaptive software
(1/2)

- J. Kramer, J. Magee, "Self-managed systems: an architectural challenge", Proc. FoSE 2007, Minneapolis 2007, 259–268.
- R. de Lemos, H. Giese, H. Müller *et al.*, "Software engineering for self-adaptive systems", Proc. Dagstuhl Seminar 10431, 2010.
- C. Klein, M. Maggio, K.E. Årzén, F. Hernández-Rodriguez, "Brownout: building more robust cloud applications", Proc. 36th International Conference on Software Engineering, Hyderabad 2014, 700–711.

On self-adaptive software
(2/2)

- A. Filieri, C. Ghezzi, A. Leva, M. Maggio, "Self-adaptive software meets control theory: a preliminary approach supporting reliability requirements", Proc. 26th IEEE/ACM International Conference on Automated Software Engineering, Lawrence 2011, 283–292.
- A. Filieri, C. Ghezzi, A. Leva, M. Maggio, "Autotuning control structures for reliability-driven dynamic binding", Proc. 51st IEEE Conference on Decision and Control, Maui 2012, 418–423.
- D. Arcelli, V. Cortellessa, A. Filieri, A. Leva, "Control theory for model-based performance-driven software adaptation", Proc. 11th International ACM Sigsoft Conference on the Quality of Software Architectures, Montreal 2015, 11–20.

About the exam

Question & proposal time