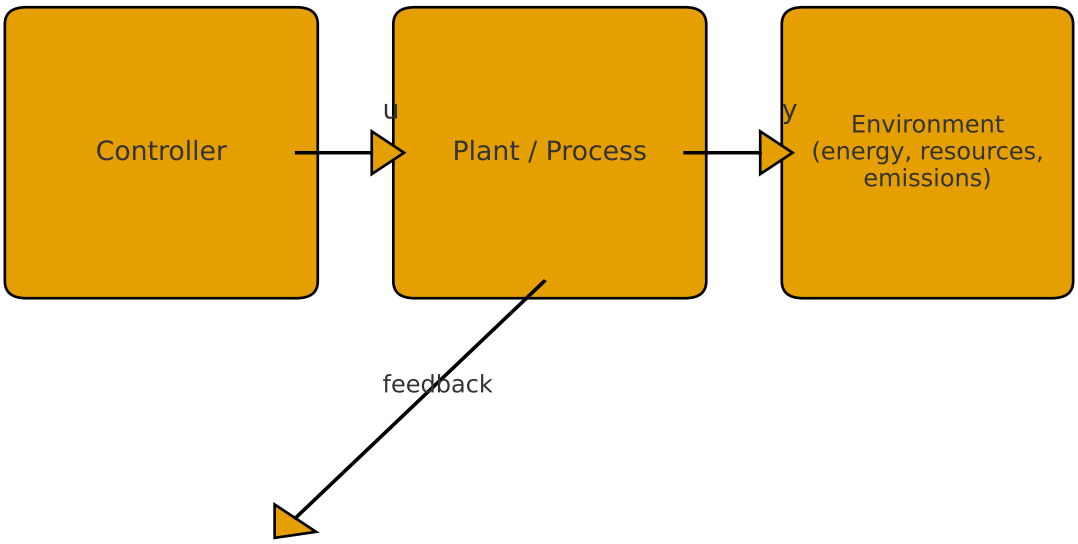
Environmental Sustainability in Control Systems

Supervision and Adaptive Systems (LM)

Department of Engineering — University of Ferrara

Handout (UK English) — designed for ~2 hours of in-class commentary

Lecturer: Prof. Silvio Simani



Intended Learning Outcomes

- Explain how environmental sustainability relates to control system design choices (sampling, actuation, computation).
- Define appropriate system boundaries and functional units for energy and resource accounting.
- Select and justify metrics for energy, resource use, and emissions suitable for your application.
- Compare alternative control strategies (PI/PID, fuzzy, adaptive, MPC) using performance–energy trade-offs.
- Quantify energy and emissions with transparent assumptions and uncertainty statements.
- Communicate design decisions clearly to technical stakeholders.

Why Sustainability in Control?

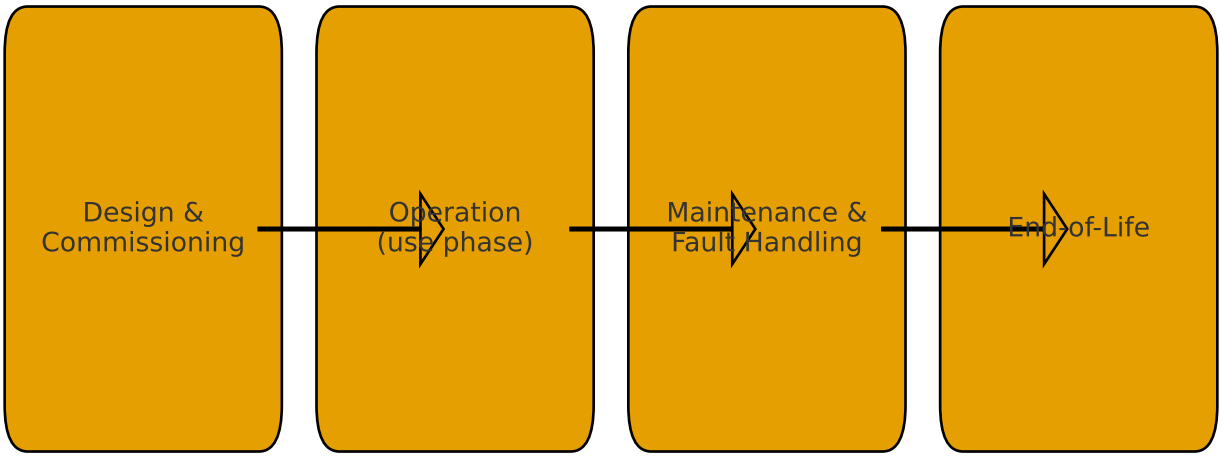
Control shapes how systems consume energy and resources throughout their life-cycle. Small design choices — such as sampling time or actuator constraints — can change energy demand, wear and tear, and emissions.

This handout supports a two-hour interactive lecture with short worked examples and prompts for discussion. It aligns with SDGs 7, 9, 12, and 13.

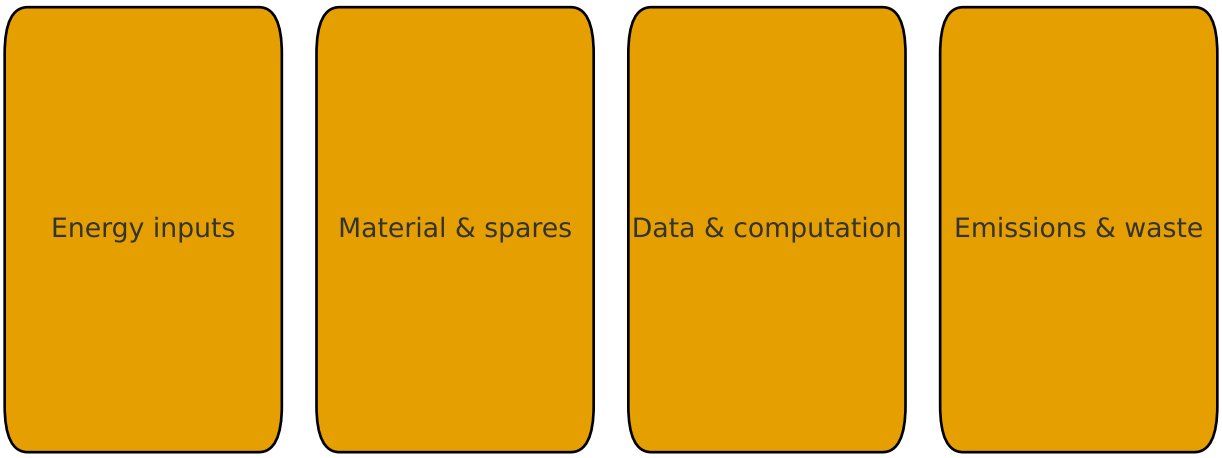
- Focus on the use phase (operation), without ignoring commissioning and end-of-life impacts.
- Seek solutions that are robust, safe, and maintainable — not merely efficient in a narrow operating point.

System Boundaries and Life-Cycle Thinking

From a control engineer’s perspective, define what is inside/outside your evaluation:



- Include controller computation, sampling rate, communications, and sensing/actuation
- energy.
- Use realistic duty cycles; document assumptions (e.g., 60% load, 24/7 operation).
- Express results per functional unit (e.g., per tonne produced, per kilometre travelled).

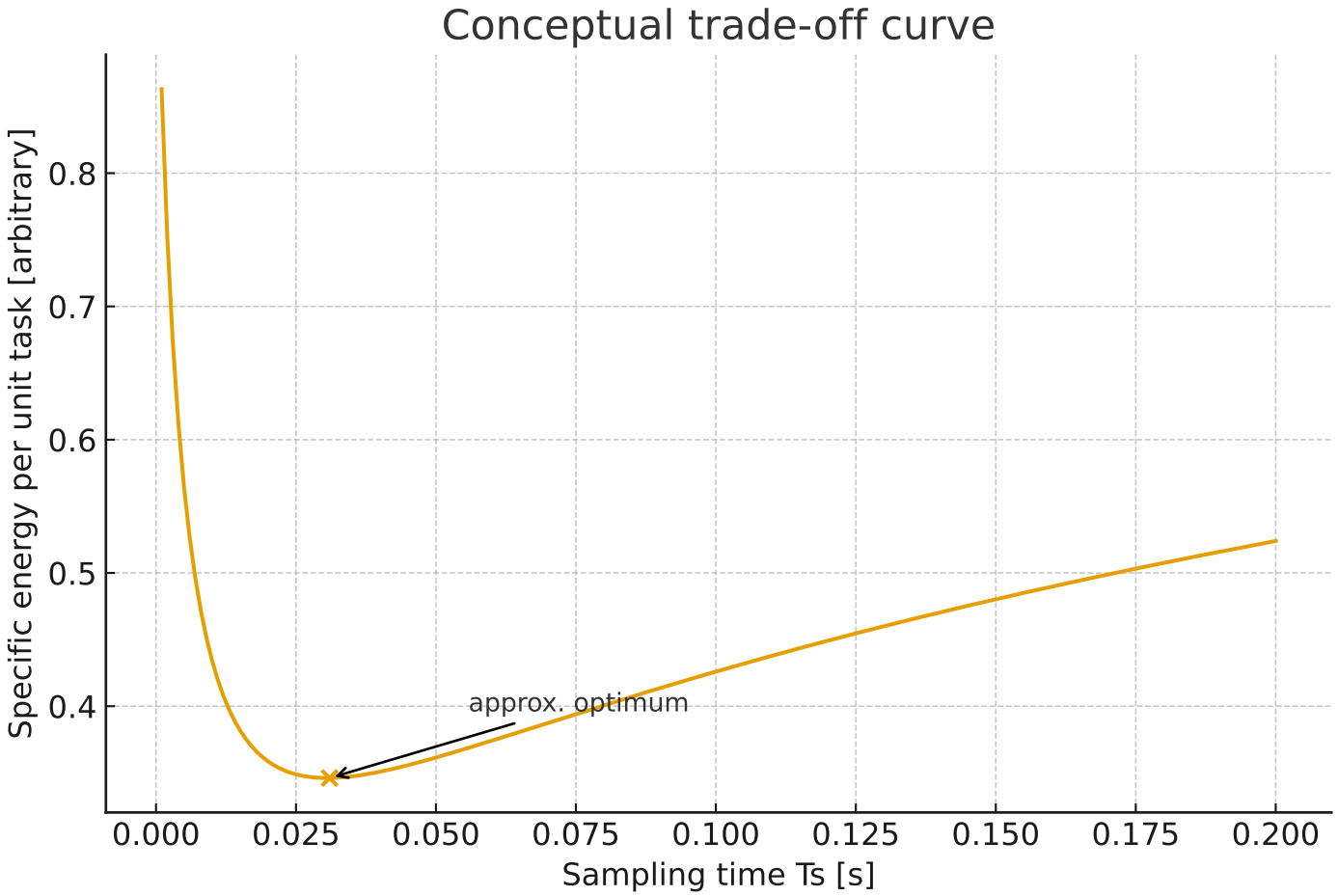


Metrics & Functional Units

- Energy per unit task (kWh/unit), peak power, and duty factors.
- CO_2e per unit task using a clearly stated grid factor (region and year).
- Resource-centric metrics: actuator wear events, filter replacements, coolant or compressed air usage.
- Computation: CPU time per step, solver iterations (MPC), memory footprint; when relevant, estimate CPU energy.
- Quality metrics to monitor alongside energy: settling time, overshoot, RMSE, constraint violations.

Illustration: Sampling Time vs Energy Use (conceptual)

Trade-off: aggressive sampling increases computation and I/O; slow sampling degrades control and en

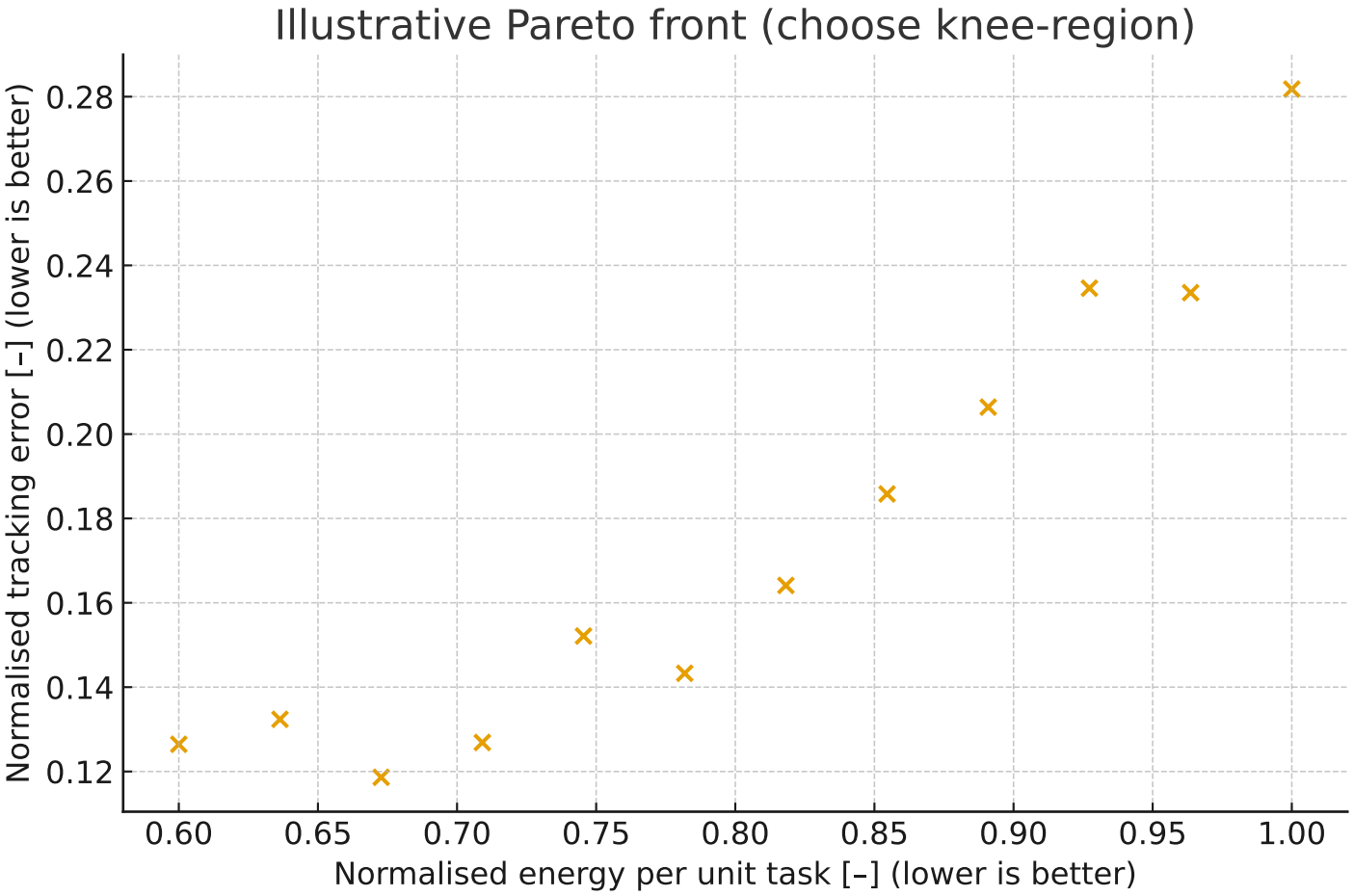


- Design guidance: evaluate energy per task for 2–3 candidate sampling times; include CPU load,
- sensor/actuator duty and control quality.
- Keep an engineering log of assumptions, data sources, and units to ensure reproducibility.

Design Levers in Control

- Sampling time and anti-windup to avoid unnecessary actuator saturation and rework.
- Setpoint shaping and reference governors to reduce aggressive transients.
- Gain scheduling or supervisory switching to stay near efficient modes.
- Predictive control horizons and constraints (shorter horizons may reduce computation but affect quality).
- Adaptive and fault-tolerant control to maintain efficiency under drift and faults.

Pareto View: Tracking Error vs Energy



- Interpretation: prefer solutions near the knee of the curve where further energy reduction causes
- disproportionate loss of performance.
- Use this view to compare PI/PID, fuzzy, adaptive, and MPC alternatives with identical constraints
- and workloads.

Worked Example A — PMSM Speed Control (illustrative)

Goal: reduce electrical energy per acceleration task while maintaining speed-tracking and torque ripple within specification.

Setup: PMSM test-bench model with inverter and DC link; compare classical PID, fuzzy PID, and model reference adaptive control (MRAC).

Assumptions (illustrative): 2 kW motor; 60 s drive cycle; three sampling times $T_s \in \{0.5\ \text{ms}, 1\ \text{ms}, 2\ \text{ms}\}$; identical current/voltage limits.

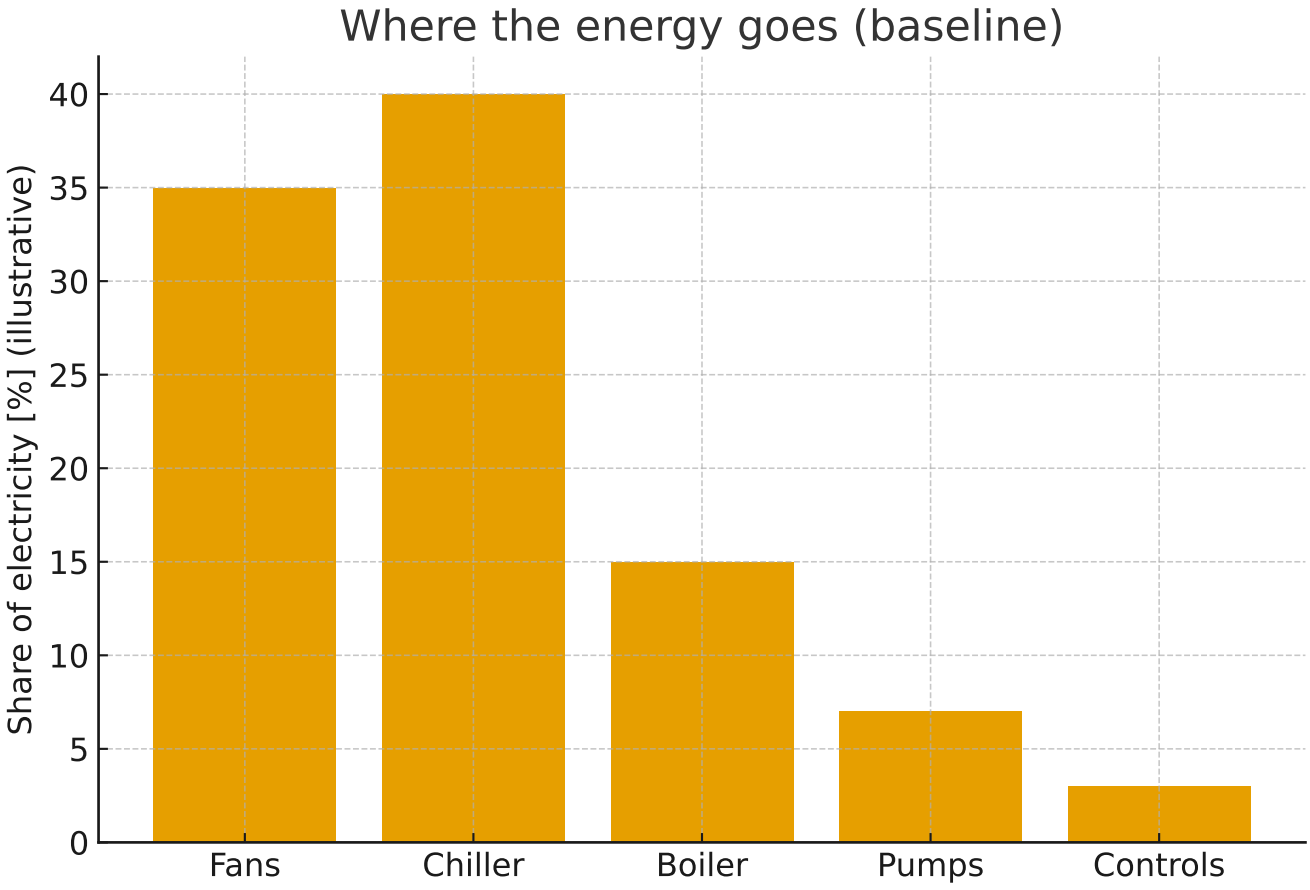
Illustrative results (normalised, lower is better):

- Energy per cycle: PID 1.00; Fuzzy PID 0.95; MRAC 0.92
- Tracking error (RMSE): PID 1.00; Fuzzy PID 0.93; MRAC 0.90
- CPU load (relative): PID 1.00; Fuzzy PID 1.20; MRAC 1.35

Design note: prefer MRAC at $T_s = 1$ ms; at $T_s = 0.5$ ms computation rises with small energy benefit.

Worked Example B — HVAC VAV Zone (illustrative)

Aim: reduce energy while preserving comfort (PMV/PPD). Use setpoint scheduling and supervisory sw



- Interventions: (i) occupancy-driven setpoints; (ii) night setback; (iii) anti-windup and smooth
- switching to reduce overshoot and reheat.
- Measurement: log air-flow and valve duty cycles; estimate fan power ∝ flow^3; convert to kWh and
- CO_2e with local factors.

Measurement & Estimation Methods

- Power measurement: prefer true-RMS meters; when unavailable, estimate from current, voltage and duty cycles with validated models.
- Functional unit: standardise results per unit task (e.g., per batch, per kilometre, per m^3 processed).
- CO_2e conversion: emissions = energy [kWh] × grid factor [$kgCO_2e/kWh$]; document the factor used and year/region.
- Edge vs cloud: account for networking and data centre energy when offloading computation.
- Algorithmic transparency: record sampling time, solver tolerances, horizon length (for MPC), and constraints.

Guided Lab Brief (MATLAB/Simulink)

- Simulink model: SISO nonlinear plant with measurable output; actuator saturations and rate limits.
- Task A (controller): tune PID, fuzzy PID, and an adaptive method. Keep the same safety constraints across all cases.
- Task B (energy): compute energy per task: $\int v(t)i(t)\,dt$ (electrical) or $\int |u(t)\cdot y(t)|\,dt$ (generic proxy).
- Task C (computation): log CPU time per step; estimate energy via power draw of the test PC or embedded target.
- Deliverable: 1–2 slides per method with numbers and a short justification of the preferred design.

In-Class Discussion Prompts

- Where is the knee of your Pareto curve, and how sensitive is it to operating conditions?
- What small change would deliver the largest energy reduction with minimal risk to robustness?
- Which assumptions affect results most (grid factor, duty cycle, sampling time, solver tolerances)?
- How would fault detection and predictive maintenance improve the sustainability of your system?

Formulae & Worked Calculation (illustrative)

- Energy per task (electrical): E = ∫ v(t) i(t) dt [J] ; For discrete logs: E ≈ Σk v[k] i[k]
- Δt.
- CO_2e estimate: m_CO2e = E_kWh × grid_factor [kg CO_2e].
- Example (illustrative): drive cycle energy E = 0.42 kWh; grid factor 0.35 kg/kWh → m_CO2e =
- 0.147 kg.
- CPU energy proxy: E_CPU ≈ P_idle·T + (P_load − P_idle)·CPU_util·T.
- Present results with uncertainty: ±(instrument error + modelling error).

References & Further Reading

- Chen, J., Patton, R.J. Robust Model-Based Fault Diagnosis for Dynamic Systems (Kluwer, 1999).
- Simani, S., Fantuzzi, C., Patton, R.J. Model-based Fault Diagnosis in Dynamic Systems using Identification Techniques (Springer, 2002).
- Skogestad, S., Postlethwaite, I. Multivariable Feedback Control (Wiley).
- Guidelines on energy-efficient control strategies in industrial automation (general best-practice notes).