Control System Architectures

K. J. Åström

Department of Automatic Control LTH
Lund University

Control System Architectures

1. Introduction
2. Examples
3. Control Paradigms
4. Three issues
5. Representations
6. Summary

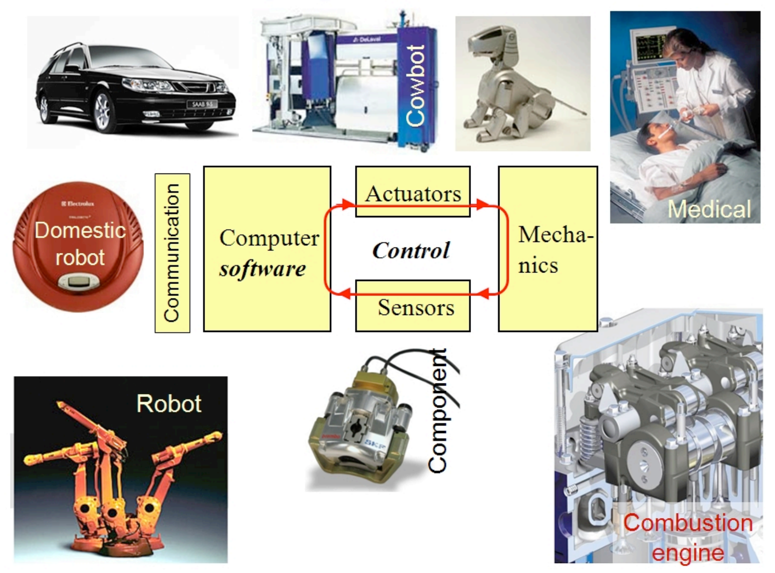
Theme: A meeting place of control and computer science.

Embedded Systems and Control

- The fact that traditional control engineers were skilled in analog computing was essential when control emerged
- Today there is a strong need to integrate knowledge about embedded systems and control
- Many embedded systems implement feedback control
- Feedback is useful in embedded systems - to provide flexibility, safety and efficiency (resource allocation)

  Adaptive scheduling and resource management
  Control of networks
- Many of the computing and scheduling models used in real-time computing are inspired by control applications
- Cluster in Artist2 on Control for Embedded Systems

Examples of Products using Embedded Systems



Control is a Rich Field

Essential to have an holistic systems view of process to be controlled, sensors, actuators, computers, software and theory

- Requirements and specifications
- Modeling
- Analysis and simulation
- Control design and simulation
- Implementation
- Validation verification
- Commissioning
- Operation
- Upgrading

Typical Tasks

- Regulation
- Servoing
- High level control tasks
  - Shape dynamics: ex stabilize and swing-up
  - Collision avoidance
  - Target tracking
  - Optimization, goal-seeking
  - Tuning
  - Adaptation
  - Learning

The Power of Feedback

- Create good systems from bad components
- Reduce effects of disturbances and component variations
- Follow command with High Fidelity
- Regulate, stabilize and shape dynamics
- Risk for instability
- Measurement noise is fed into the system
- PI(D) control the simplest way to use feedback

$$u = k_p e + k_i \int_0^t e(\tau) d\tau + T_d \frac{de}{dt}$$

The Amazing Property of Integral Action

Consider a PI controller

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

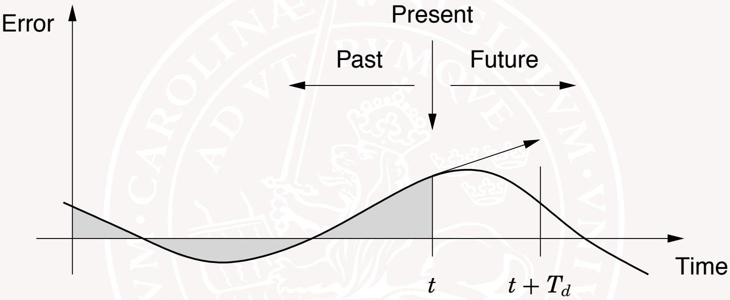
Assume that there is an equilibrium with constant $e(t) = e_0$ and constant $u(t) = u_0$. The error e_0 then must be zero. Proof: Assume $e_0 \neq 0$, then

$$u = ke_0 + k_i \int_0^t e(\tau)d\tau = ke_0 + k_i \int_0^t e_0 d\tau = ke_0 + k_i e_0 t$$

The right hand side is different from zero. Hence a contradiction unless $e_0 = 0$.

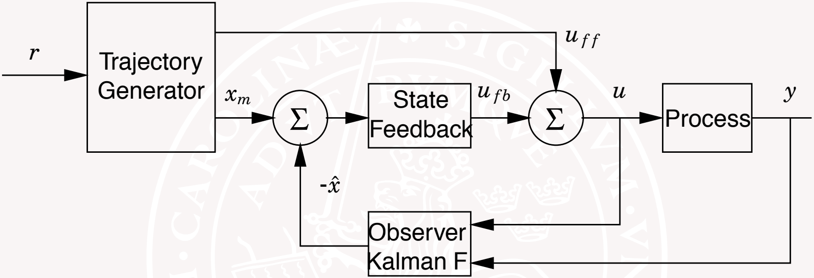
A controller with integral action will always give the correct steady state provided that a steady state exists (sometimes called adaptation).

PID versus More Advanced Controllers



- A PID controller predicts by linear extrapolation
- Advanced controllers predict using mathematical models of the process *and its environment*

Model Based Control - State Feedback Structure



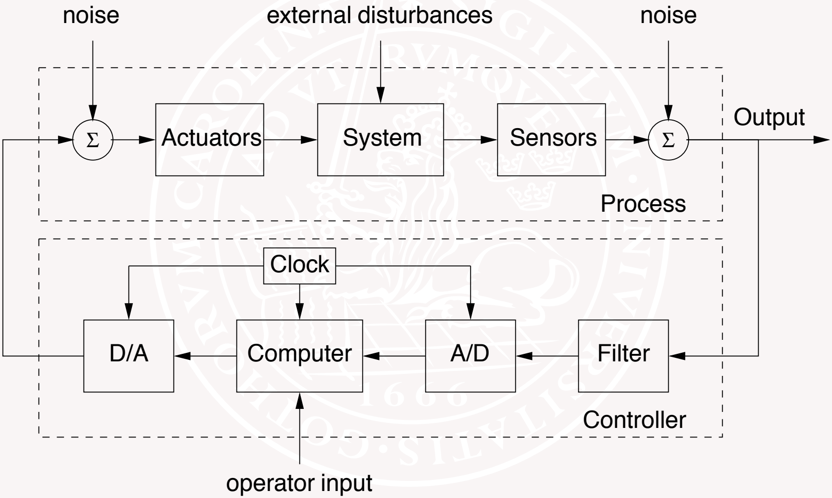
- Feedforward action by trajectory generation
- Observer (Kalman Filter) is based on a model of the process *and its environment*
- Feedback is based on the estimate the full state
- Many modeling and design theories available

Fundamental Limitations

Process and system dynamics imposes fundamental limitations to what can be done by control. Some systems cannot be controlled robustlty.

- Process instabilities: An unstable mode p_{RHP} requires a high bandwith controller. $\omega_B > 2p_{RHP}$. Think about stabilizing a pendulum!
- Time delays give an upper limit to the achievable bandwidth $\omega_B T_{delay} < 1$.
- Unstable systems cannot be controlled if the time delay is too large: Stability bound $p_{RHP} T_{delay} < 2$, robust control requires a smaller value. Stabilization of a pendulum with time delay.
- Unstable transmission zeros are similar to time delays $z \approx 0.5/T_{delay}$ gives $\omega_B < z_{RHP}/2$

Computer Control

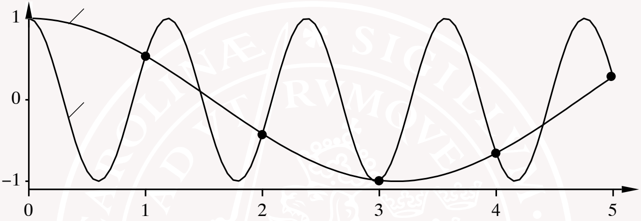


Some Issues in Computer Control

AD and DA converters are needed to connect sensors and actuators to the computer. A clock is also needed to synchronize the operations. Some important issues

- Sampling, aliasing and intersample behavior
- Control algorithms (converting differential equations to difference equations direct derivation of control law as a difference equation)
- Wordlength issues
- Bumpless parameter changes
- Variations in sampling rate
- Latency

Sampling, Aliasing and Antialiasing Filters



- Samples of signals of different frequencies may be identical
- Nyquist frequency = (Sampling frequency)/2
- To represent a continuous signal uniquely from its samples the continuous signal cannot have frequencies above the Nyqyist frequency which which is half the Nyquist frequency
- Antialiasing filters that reduce the frequency content above the Nyquist frequency are essential but they introduce time delays.

Intersample Behavior

Sample data theory is elegant because it abstracts a difficult problem (periodic systems) to a simple problem (time invariant system). Since the theory does not deal with what happens between the samples there may be problems. Can be dealt with in many ways. Some key issues are

- Proper antialias filters ensures that measured signal is smooth over the sampling period. Choise of sampling period important.
- Analyse the equations

$$x(t_k+\tau) = e^{A\tau}x(t_k) + \int_{t_k}^{t_k+\tau} e^{A(t_k+\tau-s)}Bu(t_k)ds$$

- Lifting: Consider the behavior of signals over the whole sampling interval as states.
- Sampling the whole problem (the system and the loss function)

Control System Architectures

1. Introduction
2. Examples
   - Process Control and Distributed Control Systems (DCS)
   - Cruise Control
   - Industrial PID Controller
3. Control Paradigms
4. Three issues
5. Representations
6. Summary

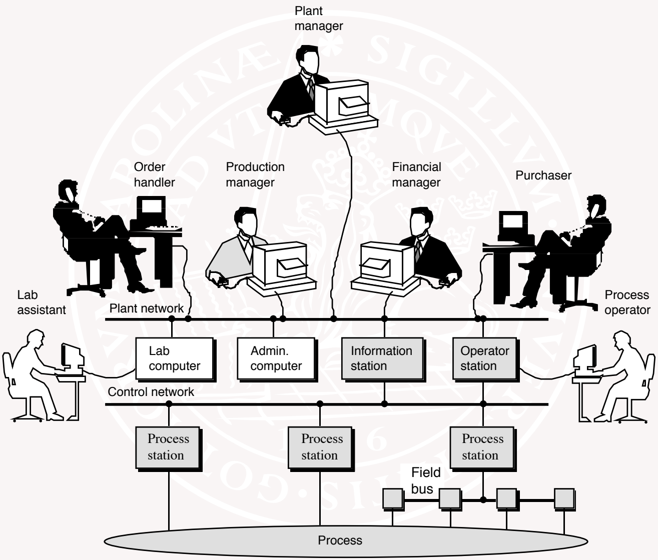
Theme: A missing link between control and computer science.

Distributed Control Systems for Process Control

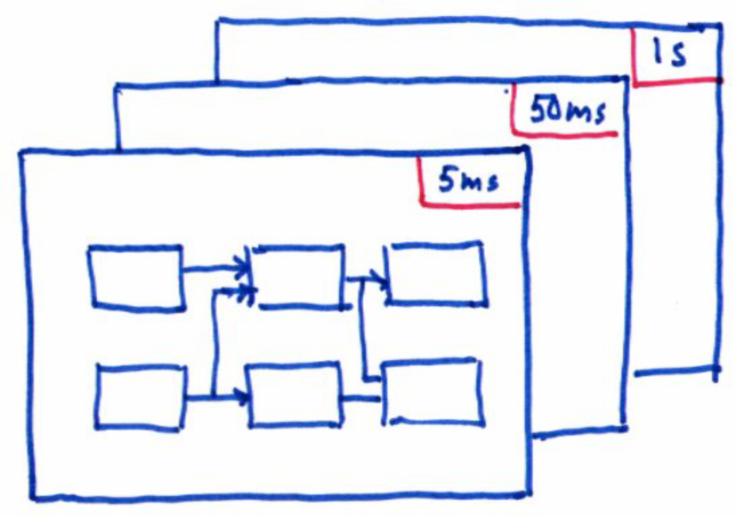
Distributed Control Systems for Process Control

- Computer control of industrial processes 1960-2010
- Early custom built systems TRW, IBM 1710 supervisory control
- Direct digital control
- Products based on special hardware, software and interfaces. Example IBM 1800, Honeywell, Foxboro,...
- Standard software and semiproprietory busses
- Microcontrollers, wireless?
- Migration of control functions to sensor and actuators

Distributed Control System (DCS)

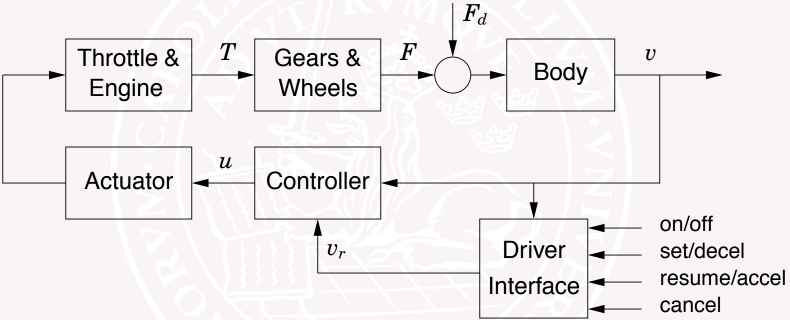


Block Diagram Programming

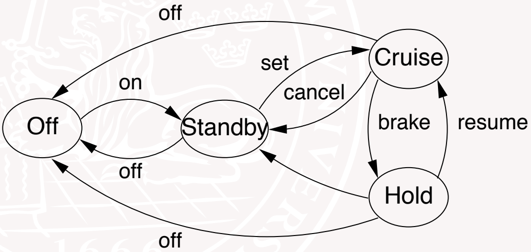


Cruise Control - A Common Embedded System

The block diagram view



Cruise Control - CC, FSM, HMI



An Industrial PID Controller

Some issues

- Operation modes: Manual, Automatic
- Parameter Changes
- Integrator windup
- Auto-tuning
- Auto-scheduling
- Adaptation
- HMI

Mode Switches: Manual/Automatic

- Hand/automatic

  Increase/decrease buttons

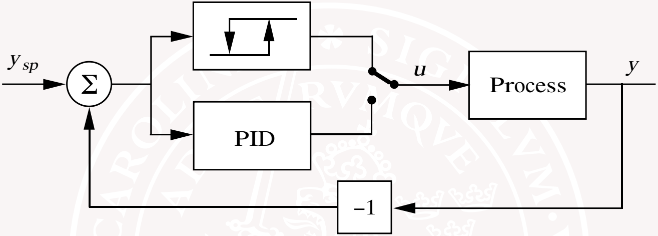
- Parameter changes

$$k_i \int_0^t e(s)ds, \qquad \int_0^t k_i e(s)ds$$

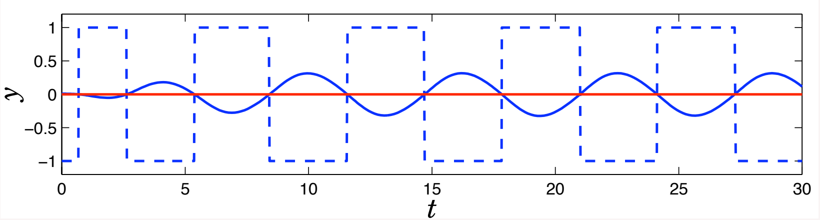
  Integration and mutliplication with a time function dont commute!

- Tune/Estimate/Schedule/Adapt
- How to handle states in mode and parameter changes

Relay Auto-tuning



What happens when relay feedback is applied to a system with dynamics? Think about a thermostat?

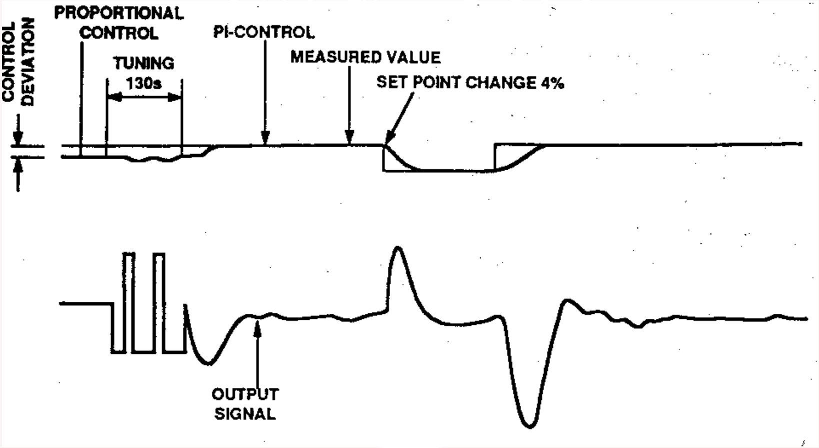


Practical Details

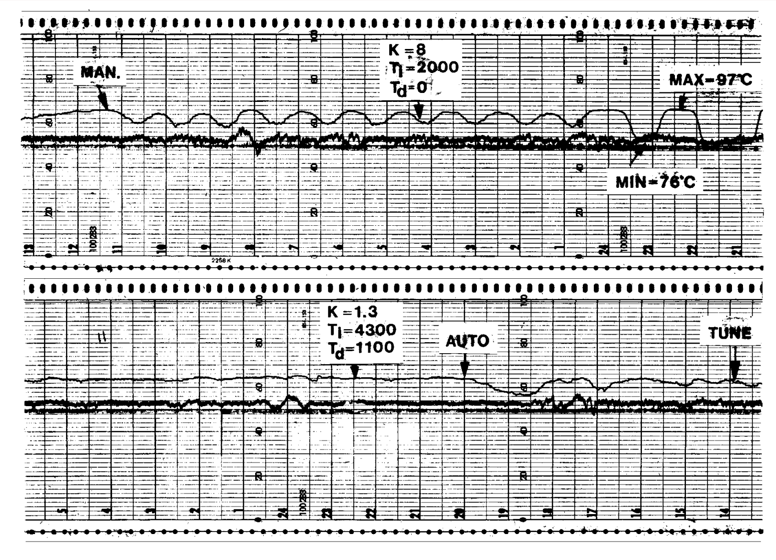
Basic controller

- Bring process to equilibrium
- Measure noise level
- Compute hysteresis width
- Initiate relay
- Monitor each half period
- Change relay amplitude automatically
- Check for steady state
- Compute controller parameters

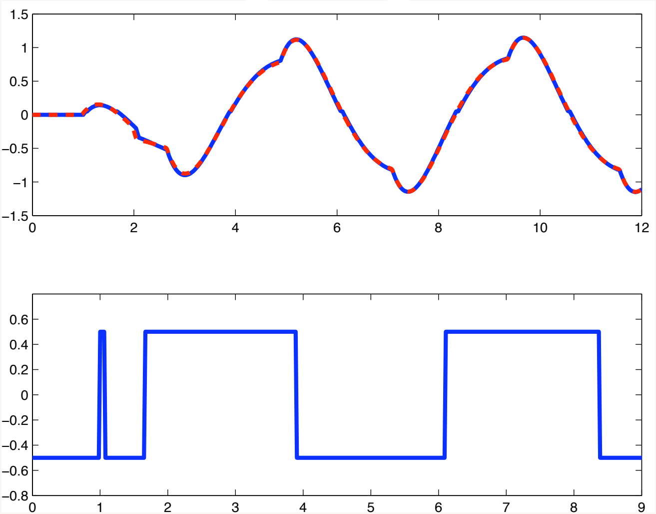
Automatic Tuning of a Level Controller



Temperature Control of Distillation Column



Fitting Better Models



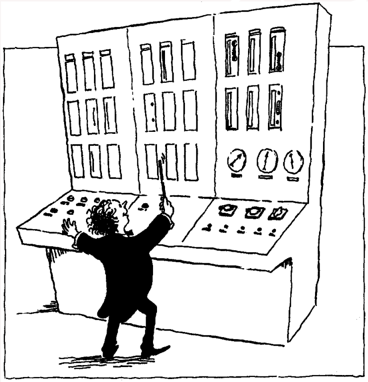
Summary of Relay Auto-tuning

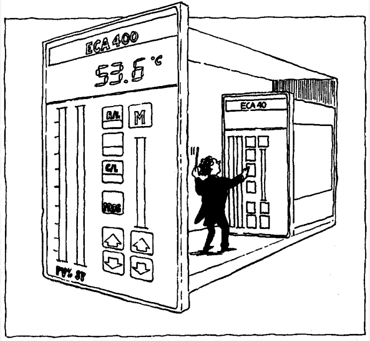
- A mixture of continuous algorithms, finite state machines
- Easy to use: One-button tuning
- Robust
- Automatic generation of test signal

  Automatically injects much energy at ω_{180} without for knowing ω_{180} apriori
- Many versions

  Stand alone
  DCS systems
  Estimation methods
  Control design
- Large numbers
- Excellent industrial experience for more than 25 years. Many patents are running out.

A View of Auto-tuning

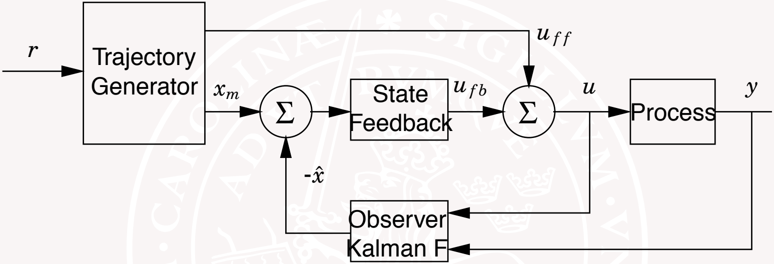




Increasing the automation level

Theme: A missing link between control and computer science.

State Feedback - Top Down



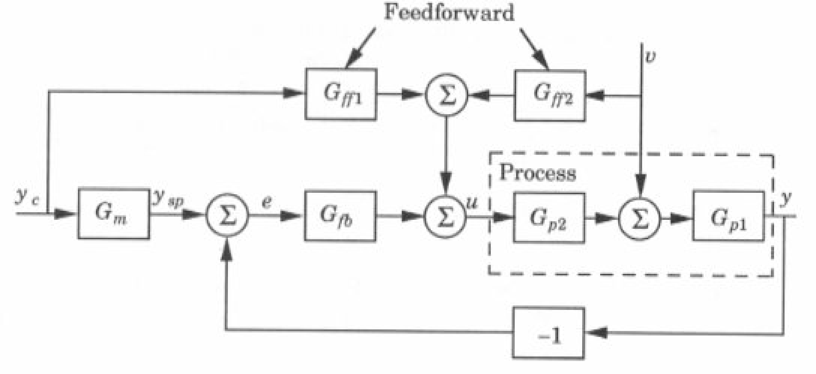
- Centralize all current information in the state
- State feedback often made hierarchical

Control Paradigms - Bottom Up

Useful to structure a control problem in several subtasks. Many schemes have emerged over the years.

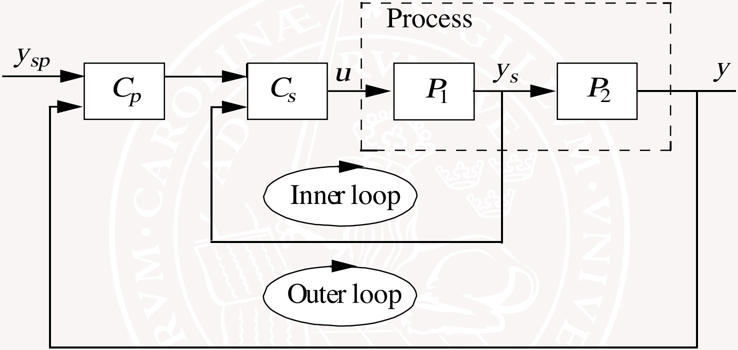
- Feedback
- Feedforward
- Model following
- Cascade control
- Split range
- Ratio control
- Disturbance attenuation
- Selector control
- State feedback
- Gain scheduling
- Adaptive control
- Neural networks
- Fuzzy control
- Intelligent control

Feedback and Feedforward



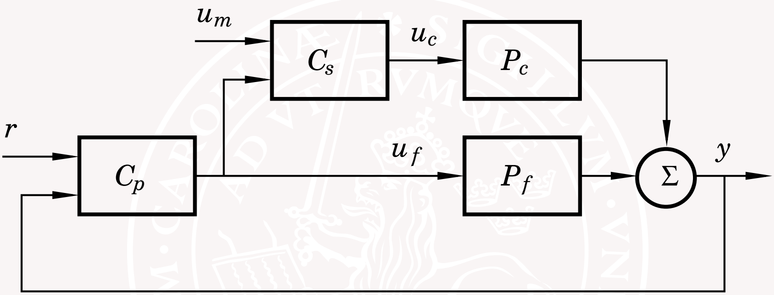
- Feedforward can be used to improve setpoint response and to reduce the effect of *measured* disturbances.
- Feedforward is more sensitive to modeling errors than feedback.

Cascade Control



- How to use several sensors?
- State feedback is the ultimate case!!

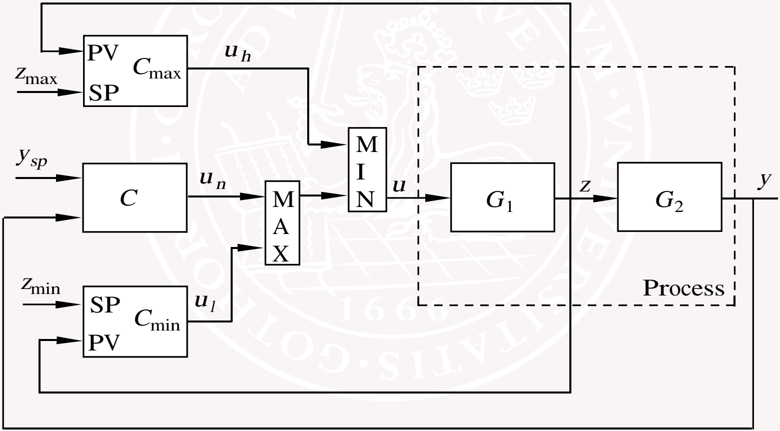
Midranging



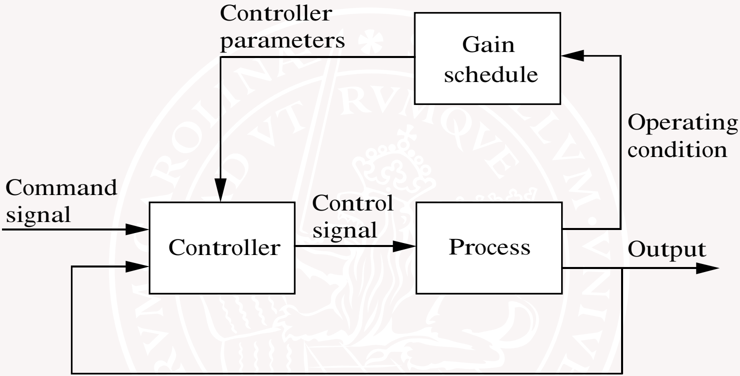
- Several actuators with different dynamics
- Fine adjustments - keep fine control in range
- Course adjustments
- Control the course range so that the fine system is in range

Selectors

Scheme used to achieve several control objectives, e.g. control temperature unless pressure is too high. A way to constrain process variables during operation.



Gain Scheduling



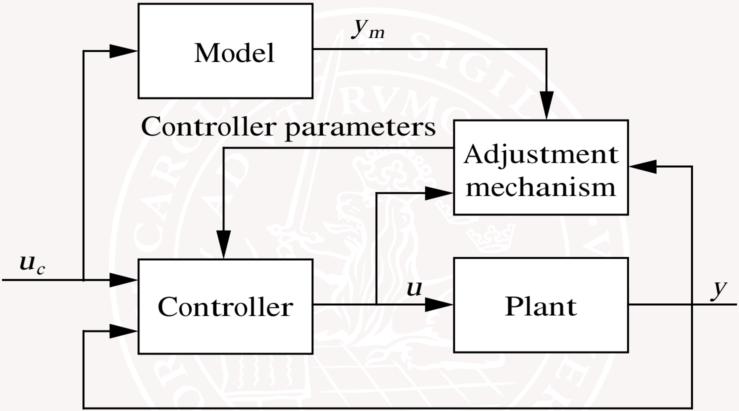
Example of scheduling variables

- Production rate
- Machine speed
- Mach number and dynamic pressure

Gain Scheduling

- Many uses
  - Linearization of actuators
  - Surge tank control
  - Control over wide operating regions
- Important issues
  - Choice of scheduling variables
  - Granularity of scheduling table
  - Interpolation schemes
  - Bump-less parameter changes
  - Man machine interfaces
- Importance of auto-tuning

Model Reference Adaptive Control MRAS

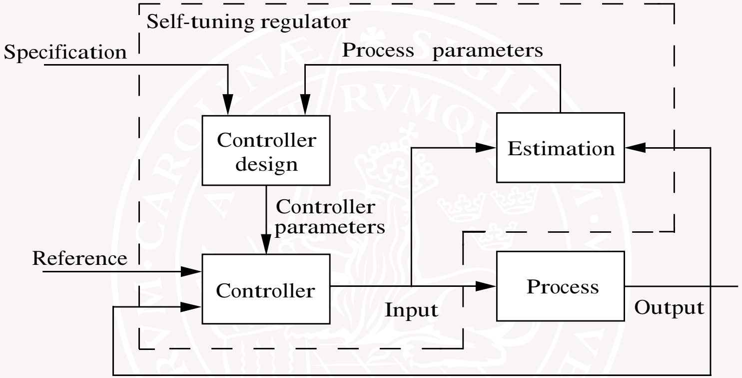


Linear feedback from $e = y - y_m$ does not work!

The MIT rule

$$\frac{d\theta}{dt} = -\gamma e \frac{\partial e}{\partial \theta}$$

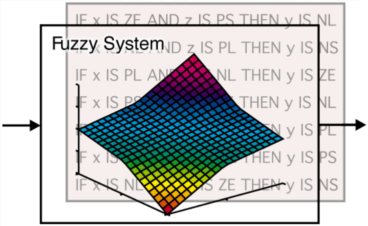
Self-Tuning Regulator STR

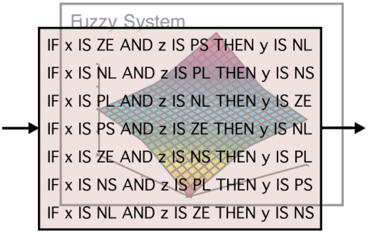


- Certainty Equivalence
- Many control and estimation schemes
- Dual control
- Control should be directing as well as investigating!

Fuzzy Control

- Rule based control
- Linguistic variables *high, low, medium*
- Membership functions
- If temperature *high* then increase flow *a little*





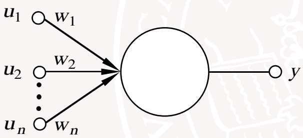
Courtesy of Karl-Erik Årzen

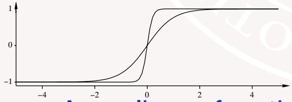
Neural Networks

Representation of functions of many variables

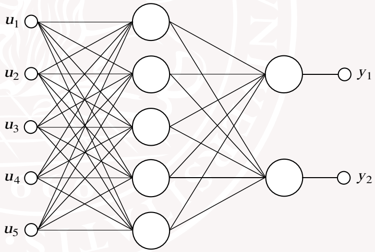
$$y(t) = f\left(\sum a_i u_i(t)\right)$$

Real and artificial neurons



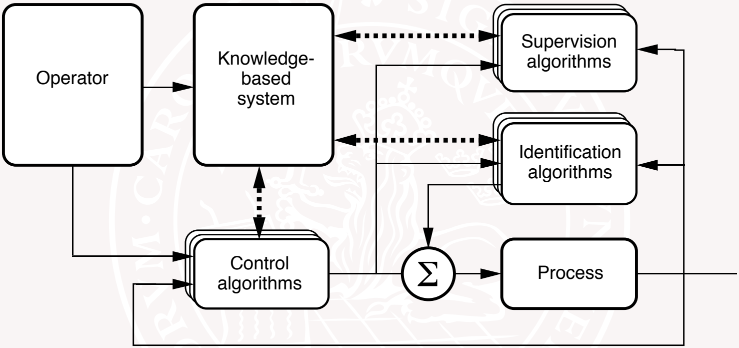


Feedforward neural network



A nonlinear function with a learning mechanism!

Intelligent Control



A knowledge bases system is used for monitoring, process supervision and switching of control and estimation algorithms.

Control System Architectures

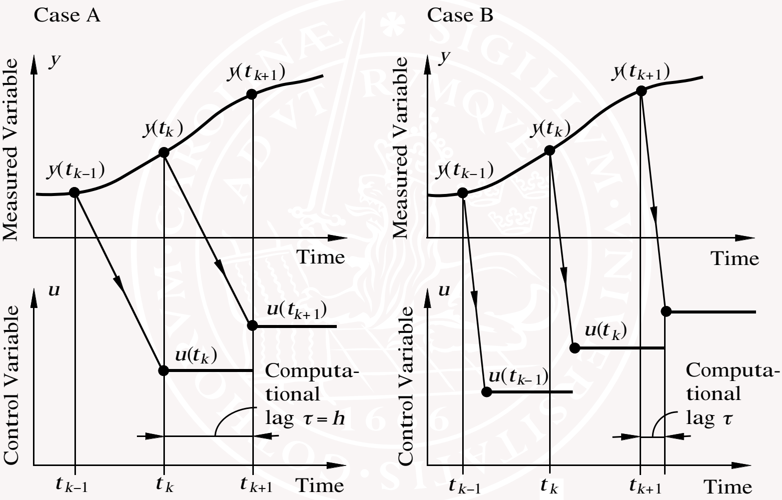
1. Introduction
2. Examples
3. Control Paradigms
4. Three issues
   - Timing and Latency
   - Saturation and Windup
   - Safety
5. Representations
6. Summary

Theme: A missing link between control and computer science.

Timing and Latency

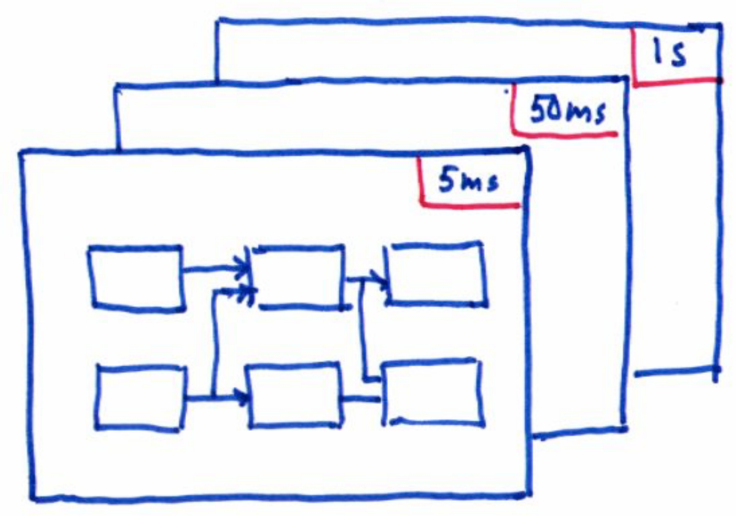
- Essential in early computer control because computers were so slow
- Became less important later as computing power increased and focus was diverted to HMI and graphics
- Analog inspired dataflow design poorly mapped on digital systems using the block diagram paradigm gave surprisingly long delays
- SattLine (Hilding Elmqvist) and ABB
- Rebirth of interest in connection with embedded systems

Latency



Case A
Measured Variable
y
y(t_{k-1})
y(t_k)
y(t_{k+1})
Time
Control Variable
u
u(t_k)
u(t_{k+1})
Computa-
tional
lag τ = h
t_{k-1}
t_k
t_{k+1}
Time
Case B
Measured Variable
y
y(t_{k-1})
y(t_k)
y(t_{k+1})
Time
Control Variable
u
u(t_{k-1})
u(t_k)
Computa-
tional
lag τ
t_{k-1}
t_k
t_{k+1}
Time

A Drawback with Blockdiagram Programming



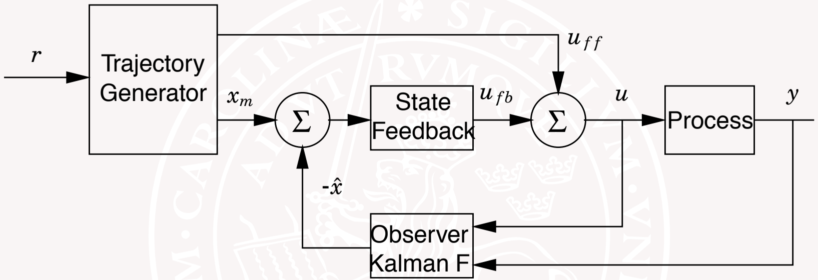
Reducing Latency

Rewrite code to minimize time between reading input signals and outputting control action

$$\% \; Compute\; controller\; output$$
$$r = adin(ch1)$$
$$y = adin(ch2)$$
$$[uff, xm] = ff(r)$$
$$u = K * (xm - xhat) + uff$$
$$daout(ch1, u)$$
$$\% \; Update\; controller\; state$$
$$xhat = A * xhat + B * u + L(y - \hat{y})$$
$$xm = f(xm, r)$$

Connected loops require knowledge of system structure (graph)

The State Feedback - A Natural Structure



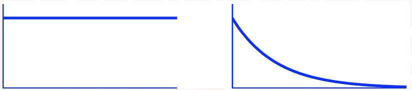
- Calculate output is done in the the Trajectory Generator and the State Feedback block
- Updating the Observer can be done when the output has been set

Sampling Jitter and Lost Data

Consider a normal digital controller that is closed over a wireless link. Almost like an ordinary digital control loop but with significant sampling jitter and lost data

- How to deal with sampling jitter?

  Build hold function in actuator
  Incremental algorithms
  Other data holds, exponential decay instead of piecewise constant



- How to deal with lost packages?

  Exploit the Kalman filter
  Exploit receding horizon control

- Eventbased control

Event Based State Feedback

Between observations integrate

$$\frac{d\hat{x}}{dt} = A\hat{x} + Bu$$
$$\frac{dP}{dt} = AP + PA + R_1$$

Initialized with $\hat{x}(t_{k-1}) = \hat{x}(t_{k-1}|t_{k-1})$ and $P^+(t_{k-1})$, at an observation update as follows

$$\hat{x}(t_k) = \hat{x}(t_k|t_{k-1}) + K\big(y_i(t_k) - C_i\hat{x}(t_k|t_{k-1})\big)$$
$$K = P(t_k)C[R_2 + CP(t)C^T]^{-1}$$
$$P^+(t_k) = P(t_k) - P(t_k)C[R_2 + CP(t)C^T]^{-1}C^T P(t_k)$$

- Update for each individual measurement
- Base control on estimates?

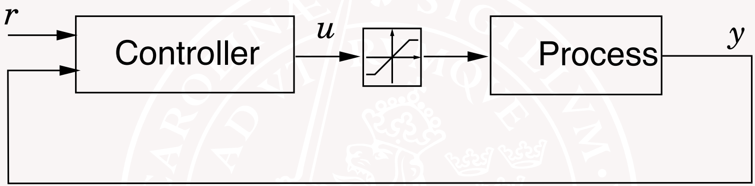
Event Based Receding Horizion Control

Receding horizon control

- Compute $u(t_k), u(t_{k+1}), u(t_{k+2}), \ldots$
- Apply $u(t_k)$
- Repeat

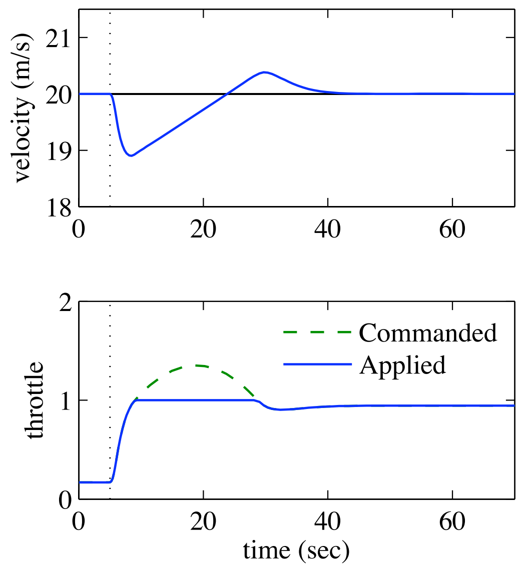
Use $u(t_{k+1})$ if no new measurement are obtained

Saturation and Windup

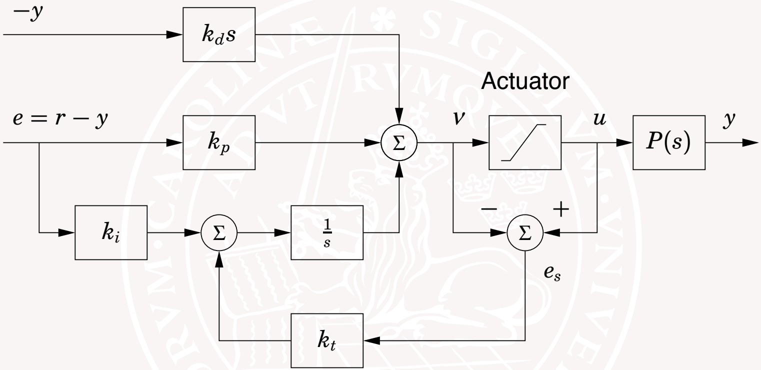


- Saturations may occur in the process and in the controller
- Practically all systems have saturations in actuators
- The feedback loop is broken when saturation occurs
- Unstable modes in process and controller will grow
- An integrator is an unstable and it will *wind up*
- All controllers with integral action require windup protection
- Instabilities are essential difficulties!

Integrator Windup in Cruise Control



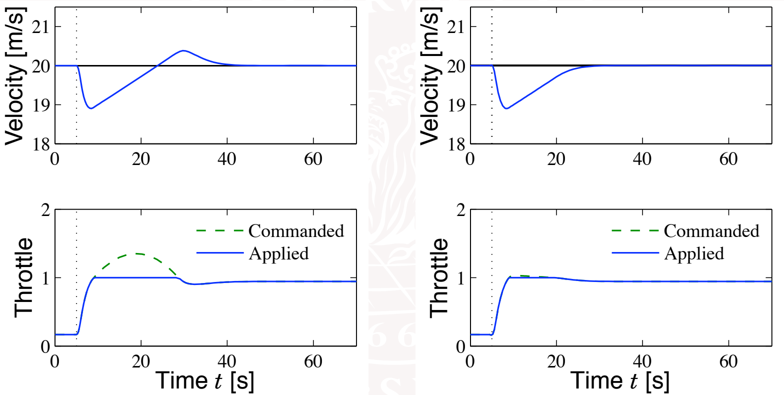
One Way to Avoid Windup



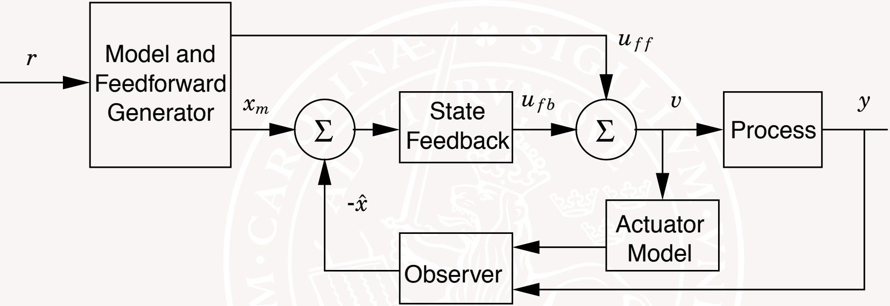
A local feedback loop keeps integrator output close to the actuator limits. The gain k_t or the time constant $T_t = 1/k_t$ determines the response time.

Cruise Control with Anti-Windup

Windup and anti-windup in a cruise controller



Avoiding Windup in State Based Controller



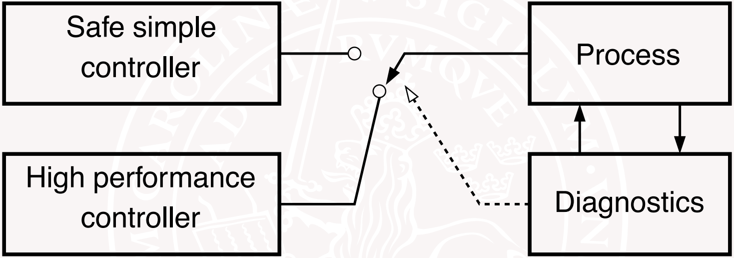
- Dont fool the observer!
- Model predictive control MPC
- Model predictive control (optimization with constraints)
- Easy to obtain tracking mode

$$\frac{d\hat{x}}{dt} = A\hat{x} + Bu + L(y - C\hat{x}), \quad u = \text{sat}(v), \quad v = u_{ff} + K(x_m - \hat{x})$$

Safety

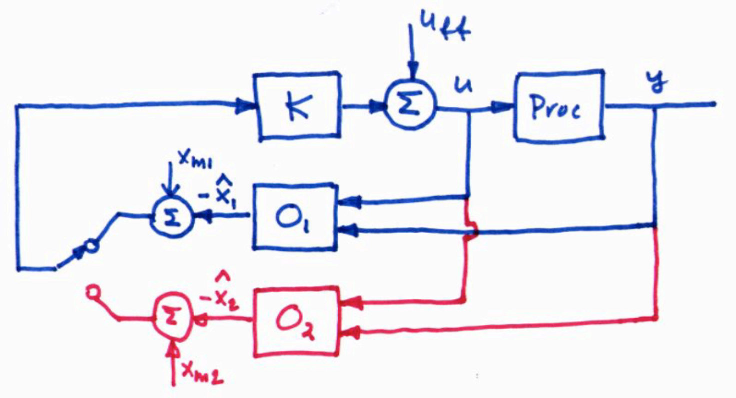
- A key issue
- Requirements-Design-Validation-Verification
- Difficult to guarantee against wrong assumptions
- Extensive hardware in the loop simulation
- Safety by design
- Safe upgrading (on-line?)

The Simplex Architecture - Lui Sha



- Use feedback to reduce effect of software errors
- Demonstator permits on-line test of control algorithm
- Has been applied successfully to real systems

Implementation



- Both observers track the state continuously
- Windup is easily accomodated by modeling actuator saturation before feeding it to the observers

Control System Architectures

1. Introduction
2. Examples
3. Control Paradigms
4. Three issues
5. Representations
6. Summary

Theme: A missing link between control and computer science.

Requirements

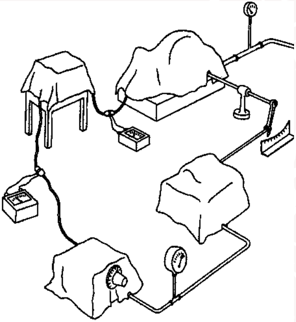
Representations to express

- Signal flow
- Logic and sequencing
- System structure and loops
- Timing

Representations should permit

- Should permit aggregation and refinement
- Should have an associated theory for analysis, design and verification

Block Diagrams - Blessing or Curse



- Very useful information hiding and abstraction
- Aggregation, refinement
- Theory: transfer functions, state models
- Dataflow - analog computing
- More issues to consider in computer control
- Logic and sequencing
- Timing and latency
- Block diagrams not well suited for physical modeling

- Control engineers are brainwashed by block diagrams.
- Dont implement old ideas in new technology.

Inputs, Outputs, States and Parameters

The block diagrams focus on inputs and outputs which is sufficient for regulation. To deal with mode changes we must also have a way to deal with states and parameters.

One possibility is to augment blocks to have

- At least two modes: tracking and controlling
- Facilities of parameter changes
- Ways of handling states in mode and parameter changes
- Facilites for handling time: latency

Logic and Sequencing

- Relays (Traditional systems had two cabinets one with controllers the other one with relays)
- PLCs
- IEEE 6.1131-3 (Ladder diagrams, logic, ...)
- Finite State Machines
- Petri Nets
- Statecharts, UML, Stateflow
- Grafcet (Sequential Function Charts)
- Grafchart

Logic and Sequencing

Reasons:

- For formal analysis
- To achieve well-structured implementations

Theoretical basis

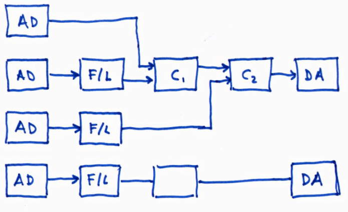
- Finite state machines
- Petri nets

Additions

- Hierarchy
- Concurrency
- History
- Sequences
- Abstractions

System Structure - Graphs

- Organize *compute output* in relevant blocks
- Allows partial compilation
- Handle windup protection in connected loops
- Much theory available



Structuring at the system (problem) level not at the software level gives a natural partition of the problem

Control System Architectures

1. Introduction
2. Examples
3. Control Paradigms
4. Three issues
5. Representations
6. Summary

Theme: A missing link between control and computer science.

Summary

- Important to develop architecture for control
- Clean up and structure the common control paradigms
- Some issues

  Windup
  Low latency
  Switching
  Diagnostics
- At least three elements

  Dataflow
  Logic and sequencing
  Structure
- Desirable properties
  - Aggregate and refine
  - Theory based
- A natural meeting place for control and computer science