Control Theory and Methods

Richard M. Murray
Control & Dynamical Systems and Bioengineering
California Institute of Technology

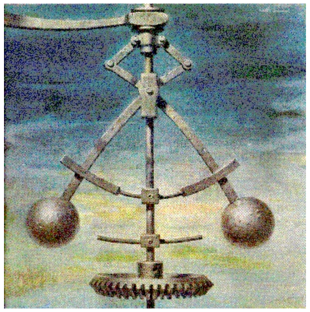
Engineering Resilient Space Systems: Leveraging Novel System Engineering Techniques and Software Architectures
30 July 2012

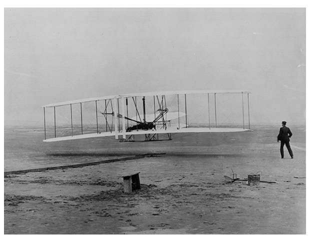
Goals

- Provide an overview of the key principles, concepts and tools from control theory that might be relevant for engineering resilient systems
  - "Classical" control - frequency domain, "inner loop" methods
  - Optimization-based control - exploit online computation, comms
- Describe current trends and recent work in control theory based on work at Caltech and JPL

Control in an Information Rich World
R. Murray (ed), SIAM, 2003
[Google: "CDS Panel Report"]

What is "Control Theory"?





Traditional view

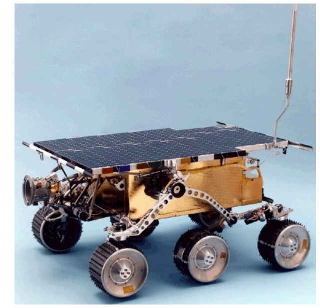
- Use of feedback for stability, performance, robustness
- DUFF: dynamics, uncertainty, feedback, feedforward

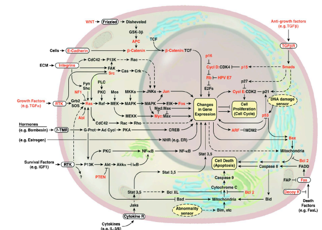
Emerging view

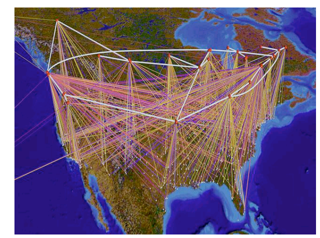
- Collection of tools and techniques for analyzing, designing and implementing complex *systems*
- Combination of dynamics, interconnection (fbk/ffd), communications, computing and software
- Successful implementation of complex systems requires combining traditional controls with CS view

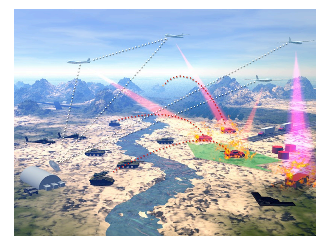
Key principles

- Feedback as a tool for *managing uncertainty*
- *Design of dynamics* through integration of sensing, actuation and computation
- Component/subsystem modularity (through feedback)









"Principles and methods used to design engineering systems that maintain desirable performance by automatically adapting to changes in the environment"

Some Important Trends in Control in the Last Decade

(Online) Optimization-based control

- Increased use of online optimization (MPC/RHC)
- Use knowledge of (current) constraints & environment to allow performance and adaptability

Layering and architectures

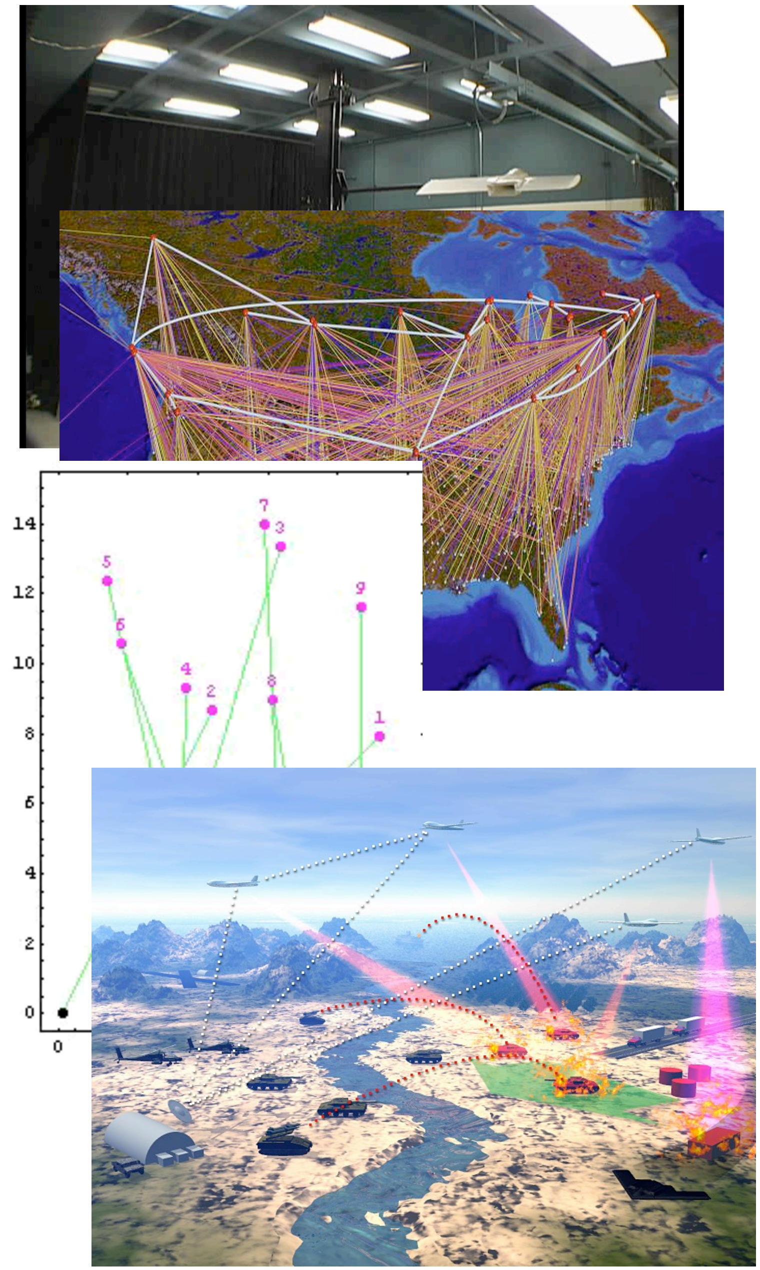
- Command & control at multiple levels of abstraction
- Modularity in product families via layers

Formal methods for analysis, design and synthesis

- Combinations of continuous and discrete systems
- Formal methods from computer science, adapted for cyberphysical systems

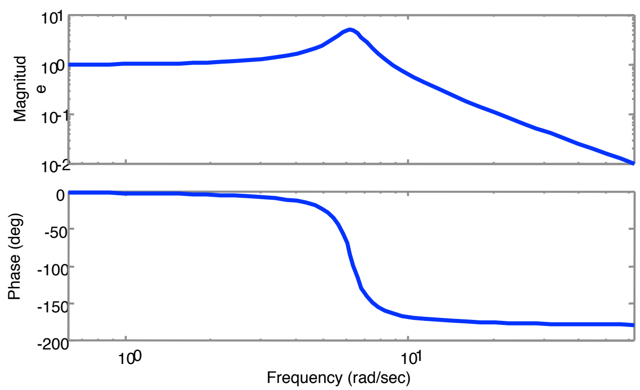
Components → Systems → Enterprise

- Movement of control techniques from “inner loop” to “outer loop” to entire enterprise (eg, supply chains)
- Use of *systematic* modeling, analysis and synthesis techniques at all levels
- Integration of “software” with “controls”



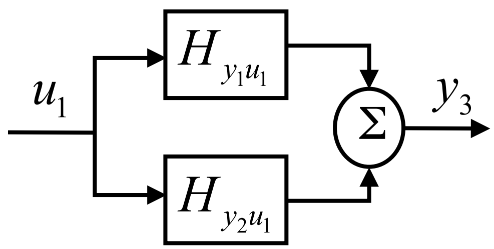
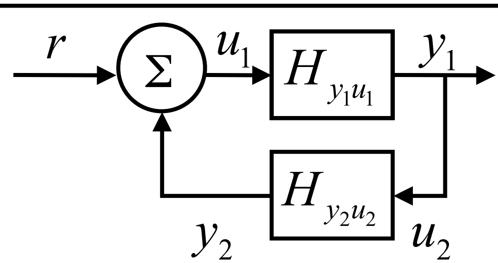
Frequency Response, Transfer Functions, Block Diagrams

$$u = A\sin(\omega t) \quad\longrightarrow\quad \begin{array}{c} \dot{x} = Ax + Bu \\ y = Cx + Du \\ x(0) = 0 \end{array} \quad\longrightarrow\quad y_{ss} = A\cdot|G(i\omega)|\times \sin(\omega t + \arg G(i\omega))$$



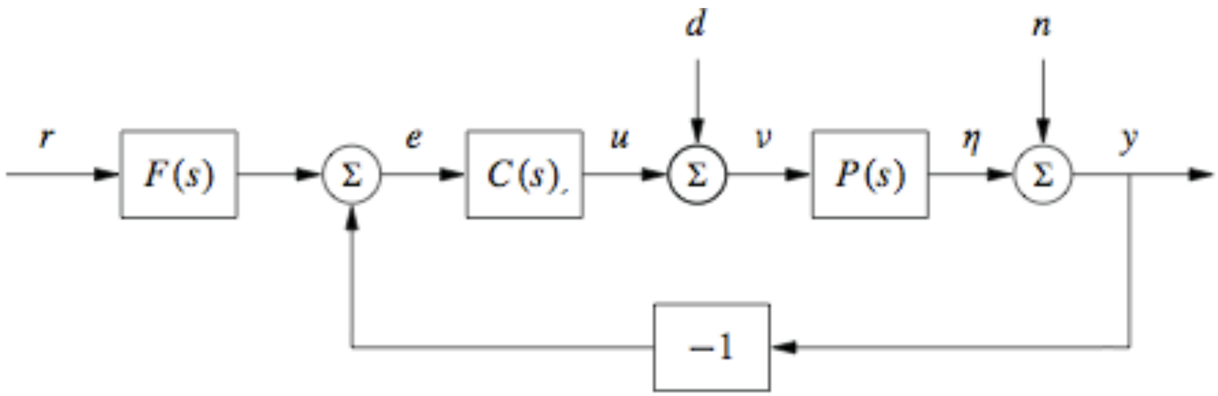
$$G(s) = C(sI - A)^{-1}B + D$$

$$G_{y_2u_1} = G_{y_2u_2}G_{y_1u_1} = \frac{n_1n_2}{d_1d_2}$$

Parallel		$H_{y_3u_1} = H_{y_2u_1} + H_{y_1u_1} = \dfrac{n_1d_2 + n_2d_1}{d_1d_2}$
Feedback		$H_{y_1r} = \dfrac{H_{y_1u_1}}{1 + H_{y_1u_1}H_{y_2u_2}} = \dfrac{n_1d_2}{n_1n_2 + d_1d_2}$

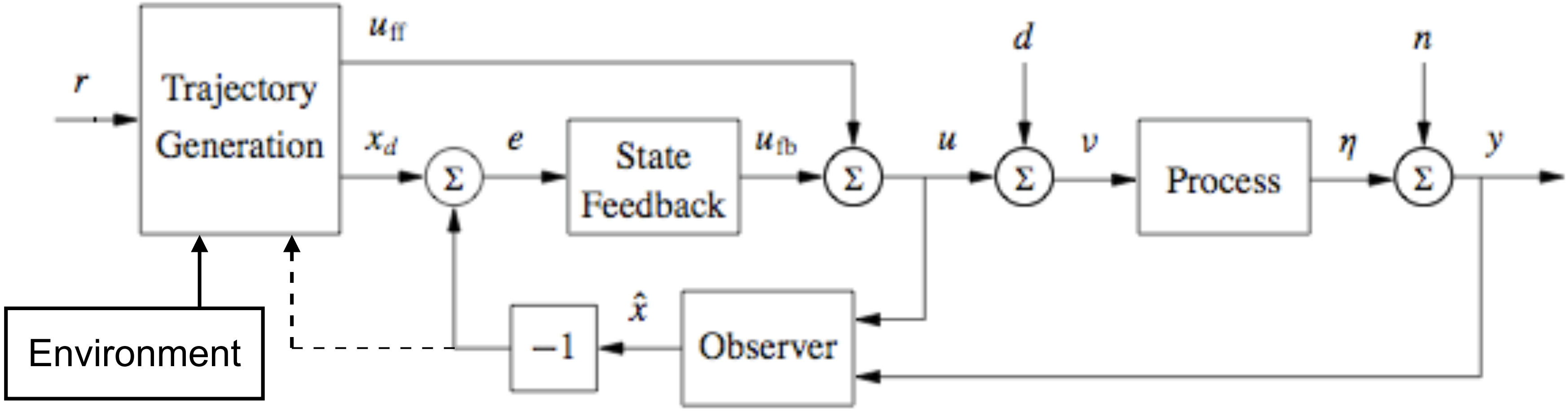
Design Patterns for Control Systems

Reactive compensation



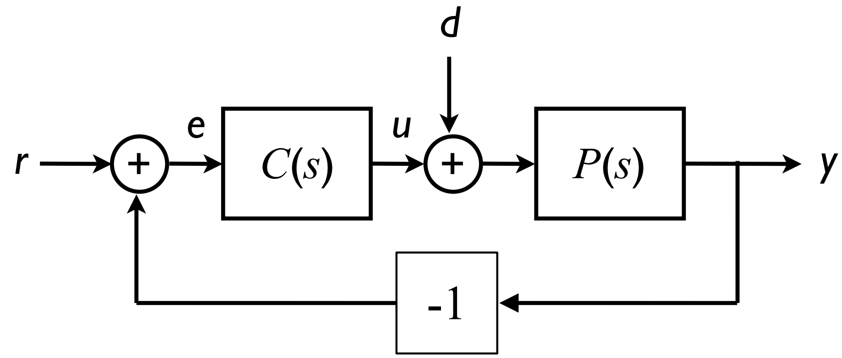
- Reference input shaping
- Feedback on output error
- Compensator dynamics shape closed loop response
- *Uncertainty* in process dynamics + external disturbances and noise
- Goals: stability, performance (tracking), robustness

Predictive compensation



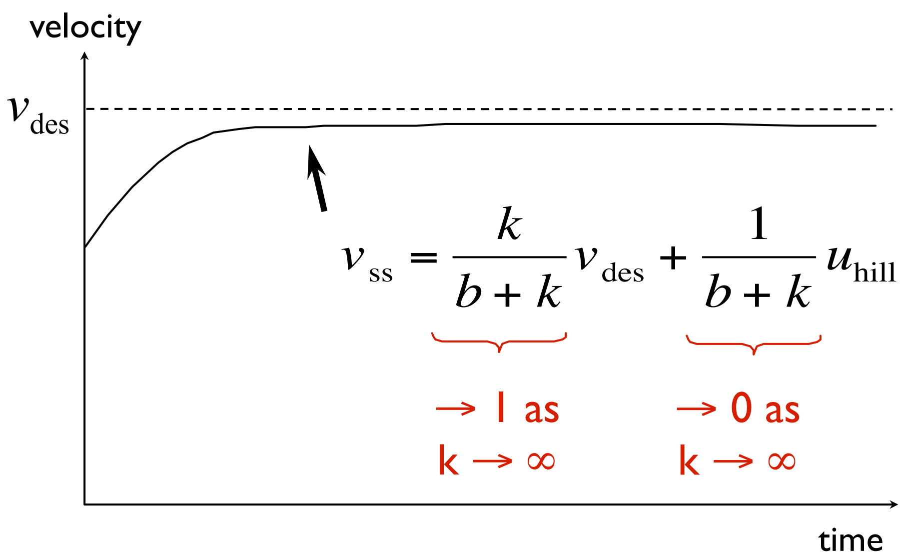
- Explicit computation of trajectories given a model of the process and environment

Example #1: Cruise Control



$$m\dot{v} = -bv + u_{\text{engine}} + d_{\text{hill}}$$

$$u_{\text{engine}} = k(\underbrace{v_{\text{des}}}_{r} - \underbrace{v}_{y})$$



$$v_{ss} = \underbrace{\frac{k}{b+k}}_{\to 1 \text{ as } k \to \infty} v_{\text{des}} + \underbrace{\frac{1}{b+k}}_{\to 0 \text{ as } k \to \infty} u_{\text{hill}}$$

Stability/performance

- Steady state velocity approaches desired velocity as k $\to \infty$
- Smooth response; no overshoot or oscillations

Disturbance rejection

- Effect of disturbances (hills) approaches zero as k $\to \infty$

Robustness

- None of these results depend on the specific values of *b, m,* or *k* for *k* sufficiently large

Control Tools: 1940-2000

Modeling

- Input/output representations for subsystems + interconnection rules
- System identification theory and algorithms
- Theory and algorithms for reduced order modeling + model reduction

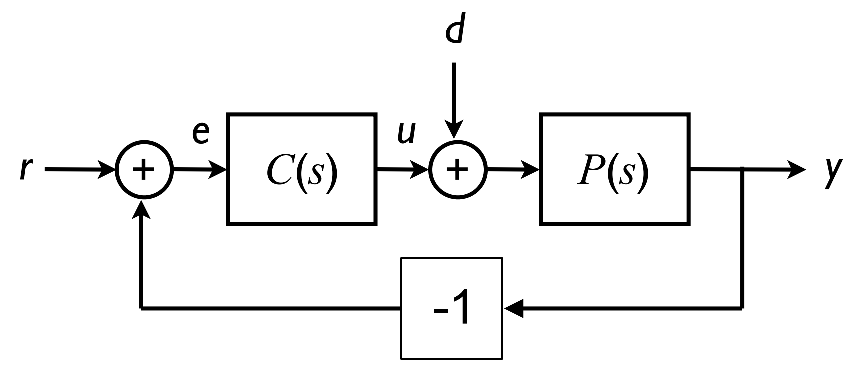
Analysis

- Stability of feedback systems, including robustness "margins"
- Performance of input/output systems (disturbance rejection, robustness)

Synthesis

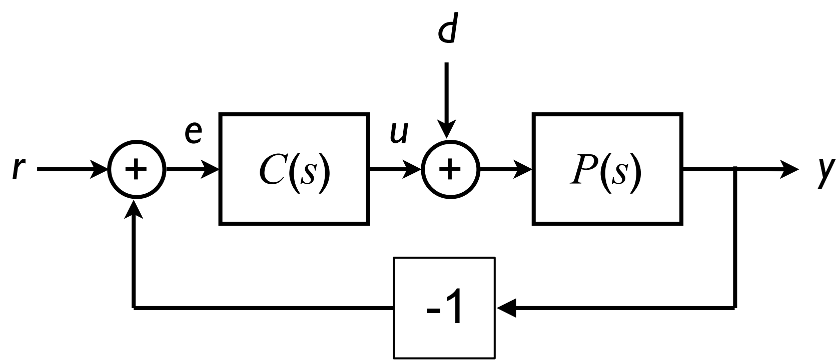
- Constructive tools for design of feedback systems
- Constructive tools for signal processing and estimation

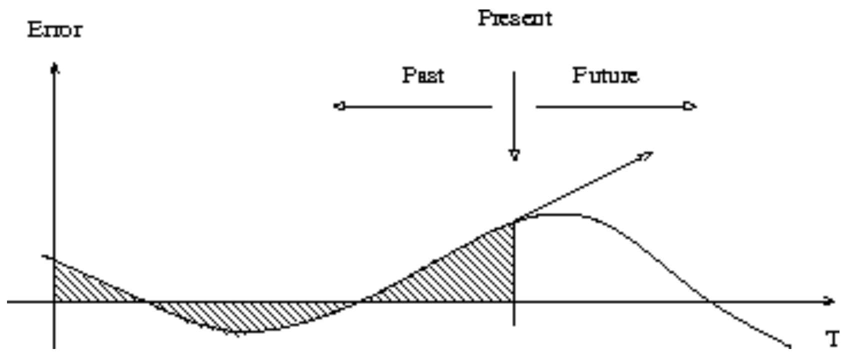
Basic feedback loop



- Plant, *P* = process being regulated
- Reference, *r* = external input (often encodes the desired setpoint)
- Disturbances, *d* = external environment
- Error, *e* = reference - actual
- Input, *u* = actuation command
- Feedback, *C* = closed loop correction
- Uncertainty: plant dynamics, sensor noise, environmental disturbances

Canonical Feedback Example: PID Control





$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

$$u(t) = ke(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de(t)}{dt}$$

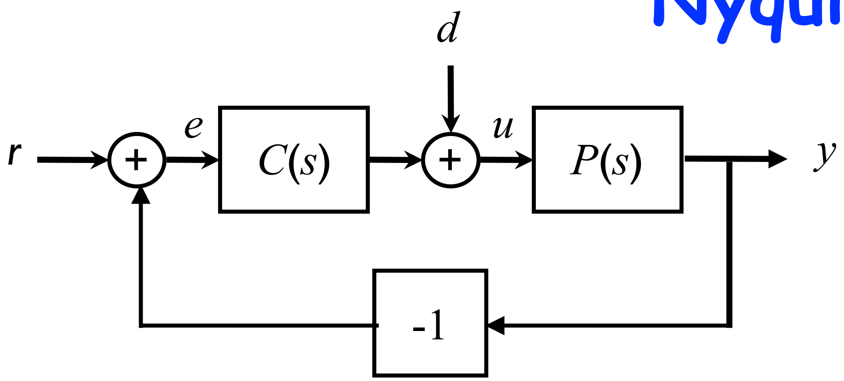
Three term controller

- Present: feedback proportional to current error
- Past: feedback proportional to *integral* of past error
  - Insures that error eventual goes to 0
  - Automatically adjusts setpoint of input
- Future: derivative of the error
  - *Anticipate* where we are going

PID design

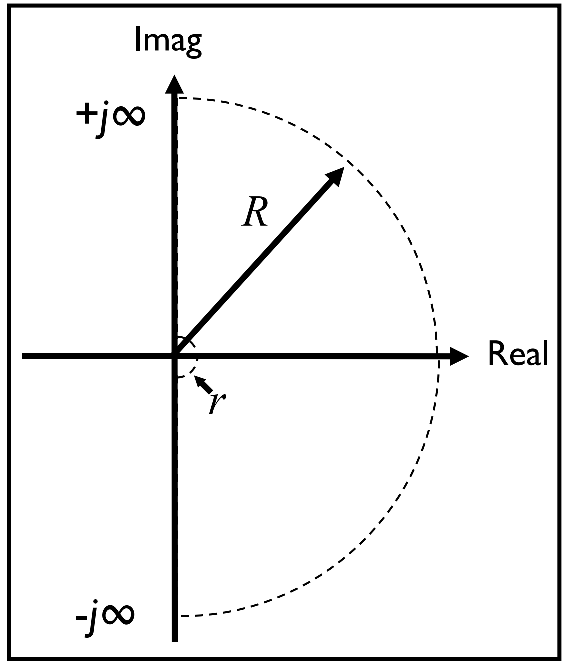
- Choose *gains* k, k_i, k_d to obtain the desired behavior
- *Stability*: solutions of the closed loop dynamics should converge to eq pt
- *Performance*: output of system, y, should track reference
- *Robustness*: stability & performance properties should hold in face of disturbances and plant uncertainty

Nyquist Criterion



Determine stability from (open) loop transfer function, $L(s) = P(s)C(s)$.

- Use "principle of the argument" from complex variable theory (see reading)
- Enables *loop shaping*: design open loop to enable closed loop properties



- Nyquist "D" contour
- Take limit as $r \to 0$, $R \to \infty$
- Trace from −1 to +1 along imaginary axis

Thm (Nyquist). Consider the Nyquist plot for loop transfer function $L(s)$. Let

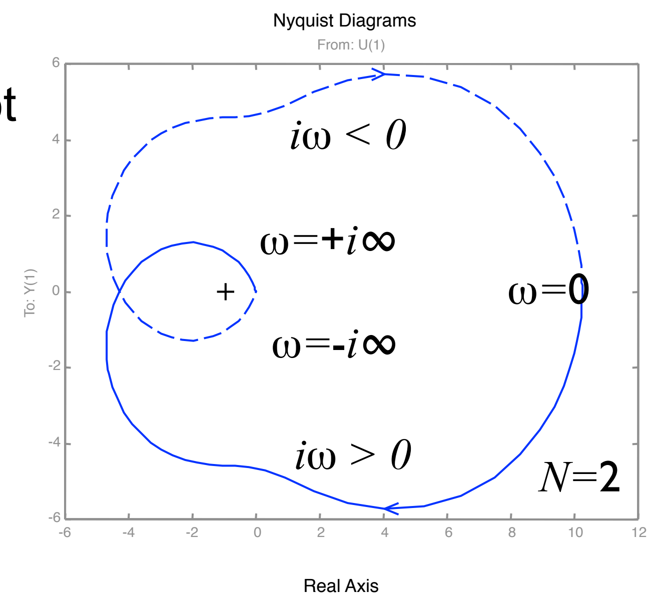
P # RHP poles of $L(s)$

N # clockwise encirclements of -1

Z # RHP zeros of $1 + L(s)$

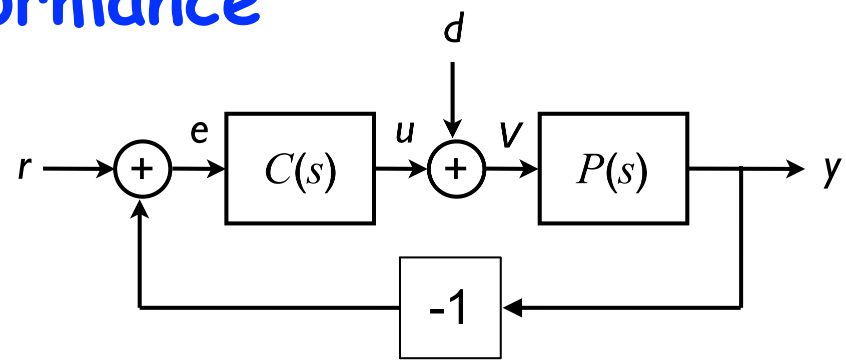
Then

$$Z = N + P$$



- Trace frequency response for $L(s)$ along the Nyquist "D" contour
- Count net # of clockwise encirclements of the -1 point

Limits of Performance



Q: How well can you reject a disturbance?

- Would like *v* to be as small as possible
- Assume that we have signals *v*(*t*), *d*(*t*) that satisfy the loop dynamics
- Take Fourier transforms *V*(ω), *D*(ω)
- *Sensitivity function*: *S*(ω) = *V*(ω)/*D*(ω); want *S*(ω) « 1 for good performance

Thm (Bode) Under appropriate conditions (causality, non-passivity)

$$\int_0^\infty \log |S(\omega)| d\omega \geq 0$$

Consequences: achievable performance is bounded

- Better tracking in some frequency band ⇒ other bands get worse
- For linear systems, formula is known as the *Bode integral formula* (get equality)
- "Passive" (positive real) systems can beat this bound

Extensions

- Discrete time nonlinear systems: similar formula holds (Doyle)
- Incorporate Shannon limits for communication of disturbances (Martins et al)

Example: Magnetic Levitation System

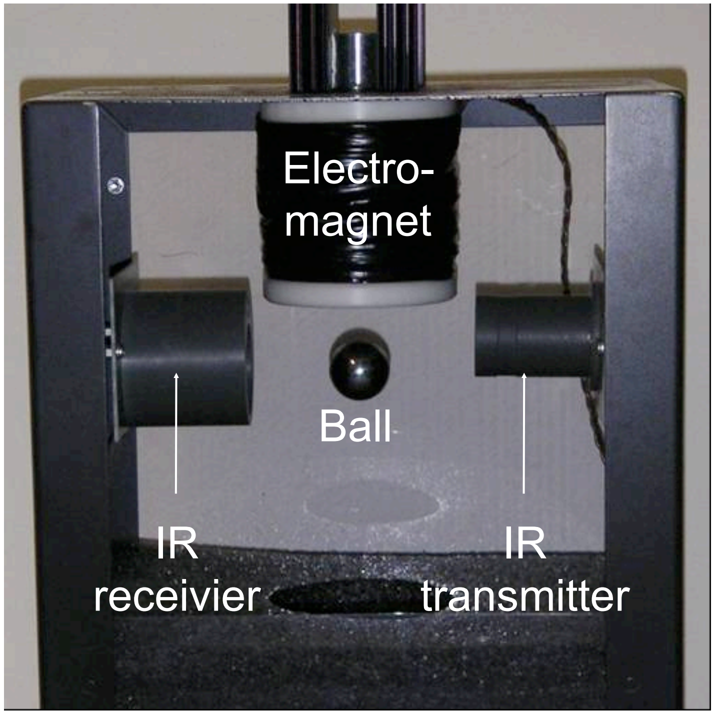
Nominal design gives low perf

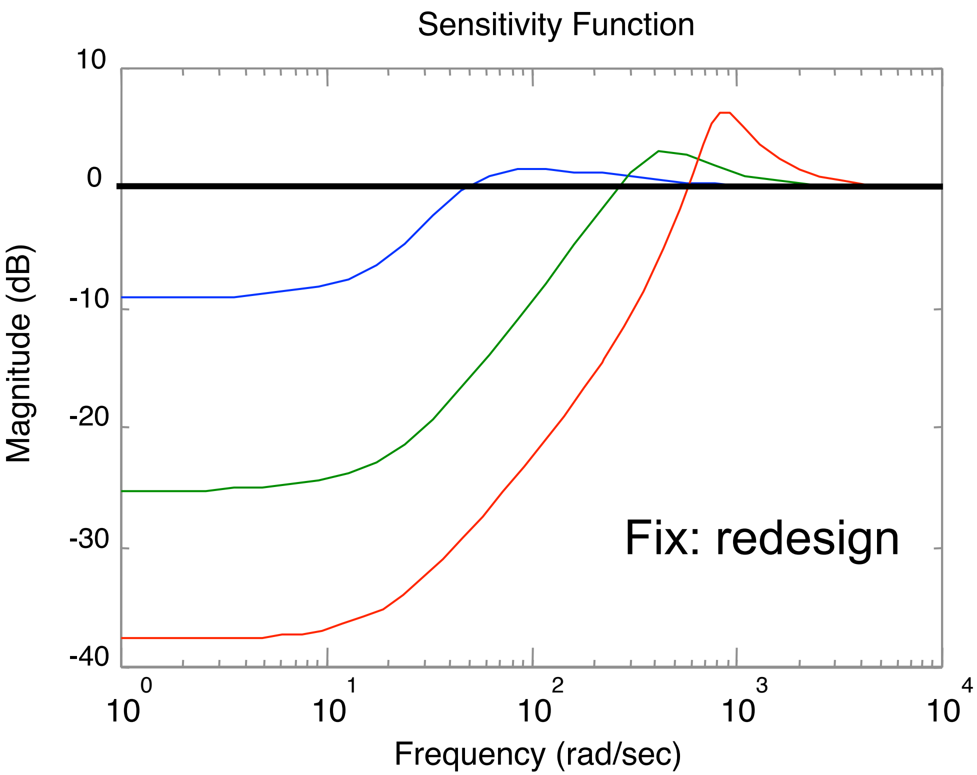
- Not enough gain at low frequency
- Try to adjust overall gain to improve low frequency response
- Works well at moderate gain, but notice waterbed effect

Bode integral limits improvement

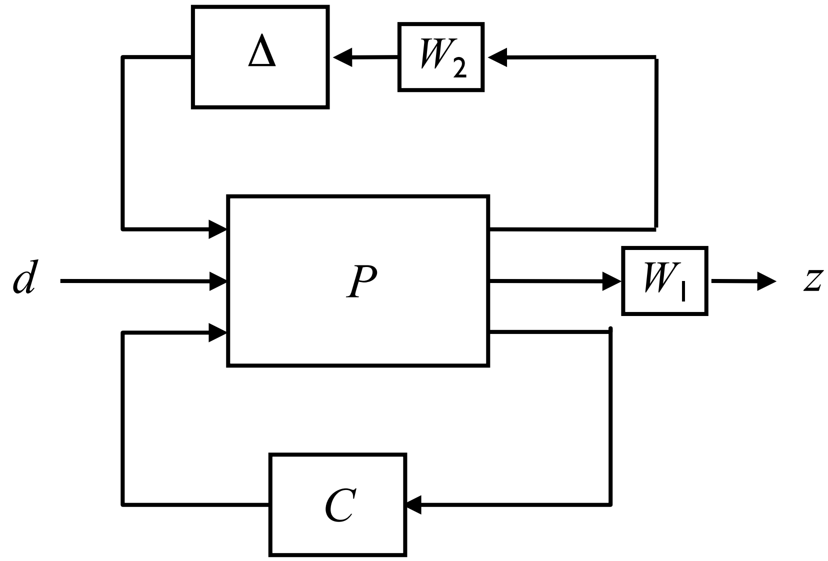
$$\int_0^\infty \log |S(j\omega)| d\omega = \pi r$$

- Must increase sensitivity at some point





Robust Control Theory



Model components as I/O operators

y(·) = P(u(·), d(·), w(·))

d disturbance signal

z output signal

Δ uncertainty block

W_1 performance weight

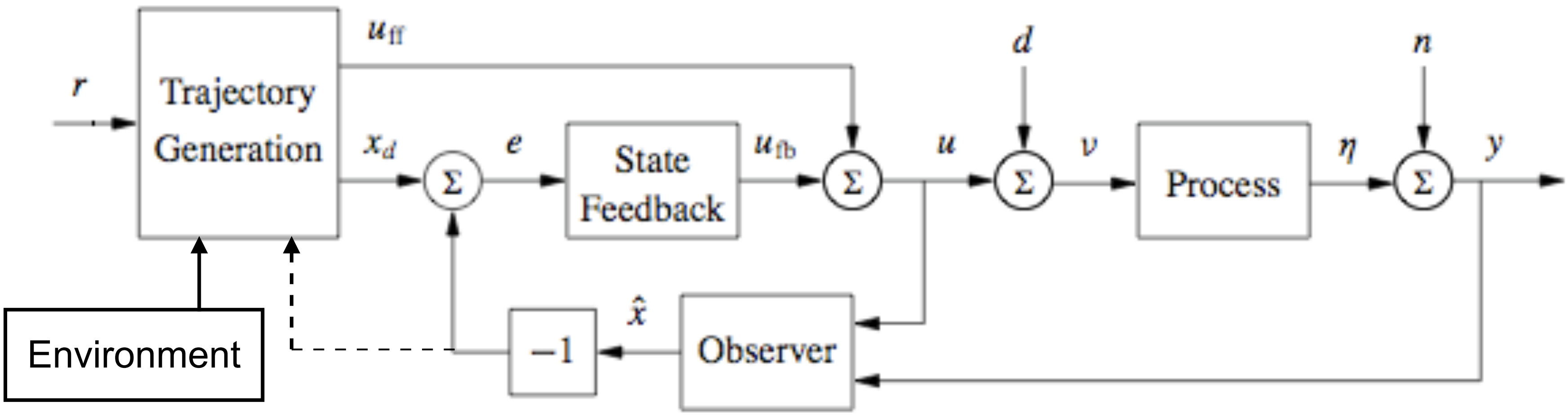
W_2 uncertainty weight

Goal: guaranteed performance in presence of uncertainty

$$\|z\|_2 \leq \gamma \, \|d\|_2 \quad \text{for all} \quad \|\Delta\| \leq 1$$

- Compare energy in disturbances to energy in outputs
- Use frequency weights to change performance/uncertainty descriptions
- “Can I get X level of performance even with Y level of uncertainty?”
- Generalizations to nonlinear systems (along trajectories) available [Tierno et al]

Feedforward and Feedback

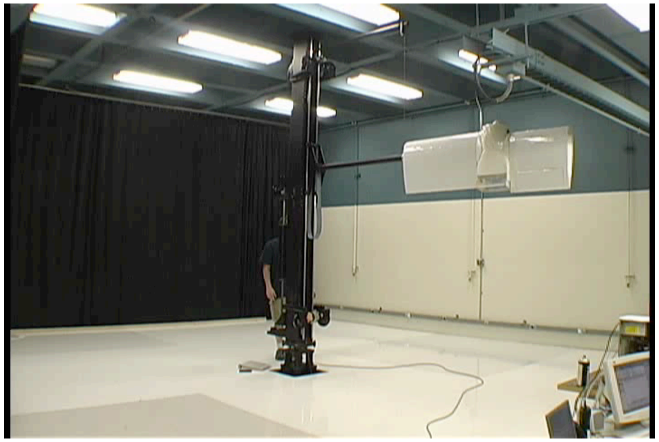


Benefits of feedforward compensation

- Allows online generation of trajectories based on current situation/environment
- Optimization-based approaches can handle constraints, tradeoffs, uncertainty
- Trajectories can be pre-stored and used when certain conditions are met

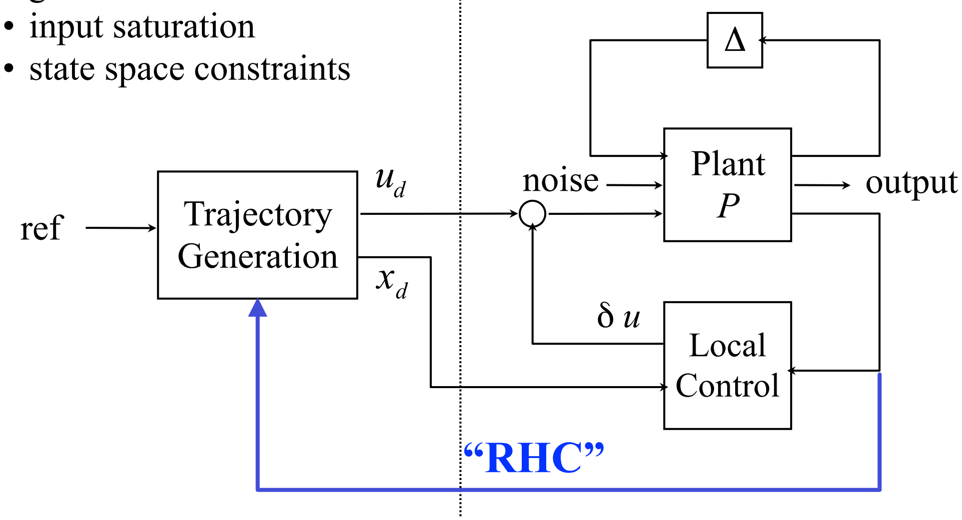
Replanning using receding horizon

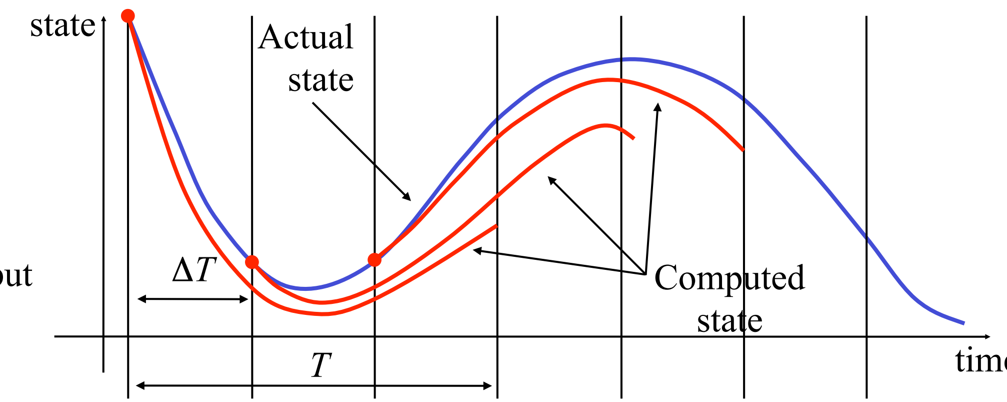
- Idea: regenerate trajectory based on new states, environment, constraints, etc
- Provides "outer loop" feedback at slower timescale
- Stability results available



Optimization-Based Control

Nonlinear design
- global nonlinearities
- input saturation
- state space constraints





$$u_{[t,t+\Delta T]} = \arg\min \int_t^{t+T} L(x(\tau),u(\tau))\,d\tau + V(x(t+T))$$

$$x_0 = x(t) \quad x_f = x_d(t+T)$$

$$\dot{x} = f(x,u) \quad g(x,u) \le 0$$

Offline design + analysis → online design

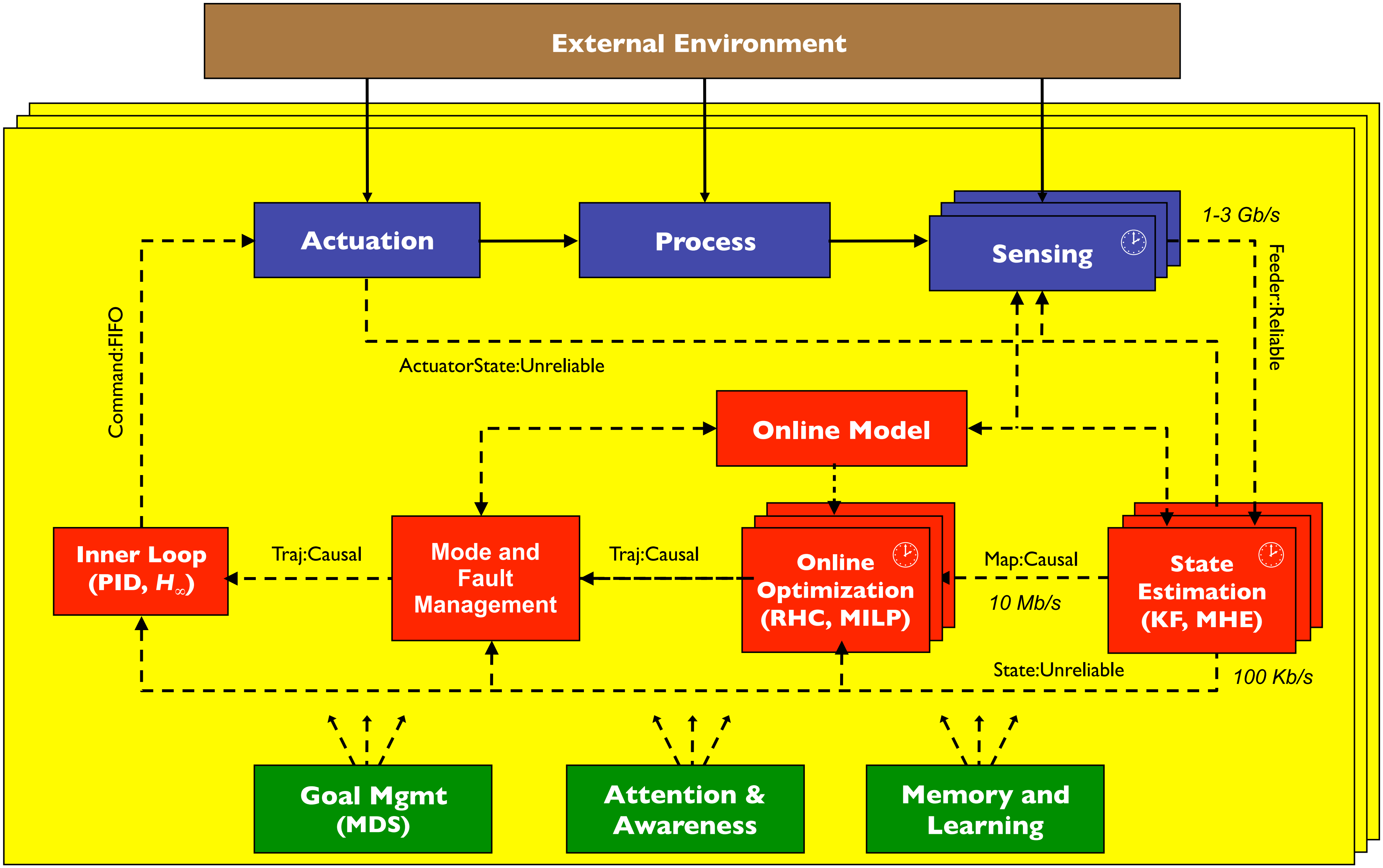
- Traditional: design (simple) controller, analyze performance, check with constraints
- Modern: specify performance and constraints, design trajectory/control to satisfy
- Problem: overall space too large; *online* optimization allows simplification
- Example of a "correct by construction" technique. Cost function = Lyapunov function

Links to resiliency

- Can "re-solve" the design problem in presence of (measurable) failures
- Still limited by our ability to formally specify behavior, computational tractability, etc

Networked Control Systems

(following P. R. Kumar)



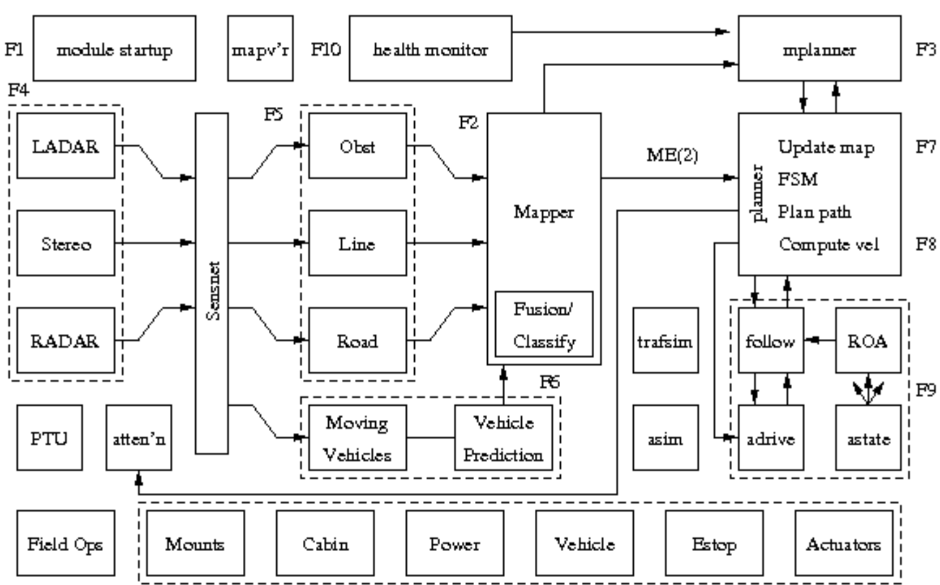
Recent Example: Alice (DGC07)

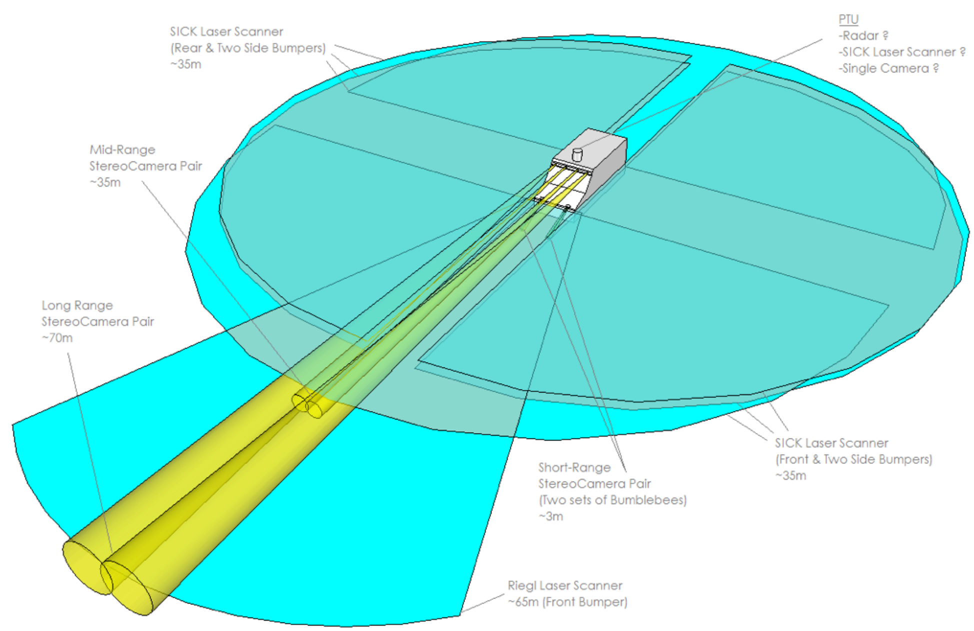
Alice

- 300+ miles of fully autonomous driving
- 8 cameras, 8 LADAR, 2 RADAR
- 12 Core 2 Duo CPUs + Quad Core
- ~75 person team over 18 months

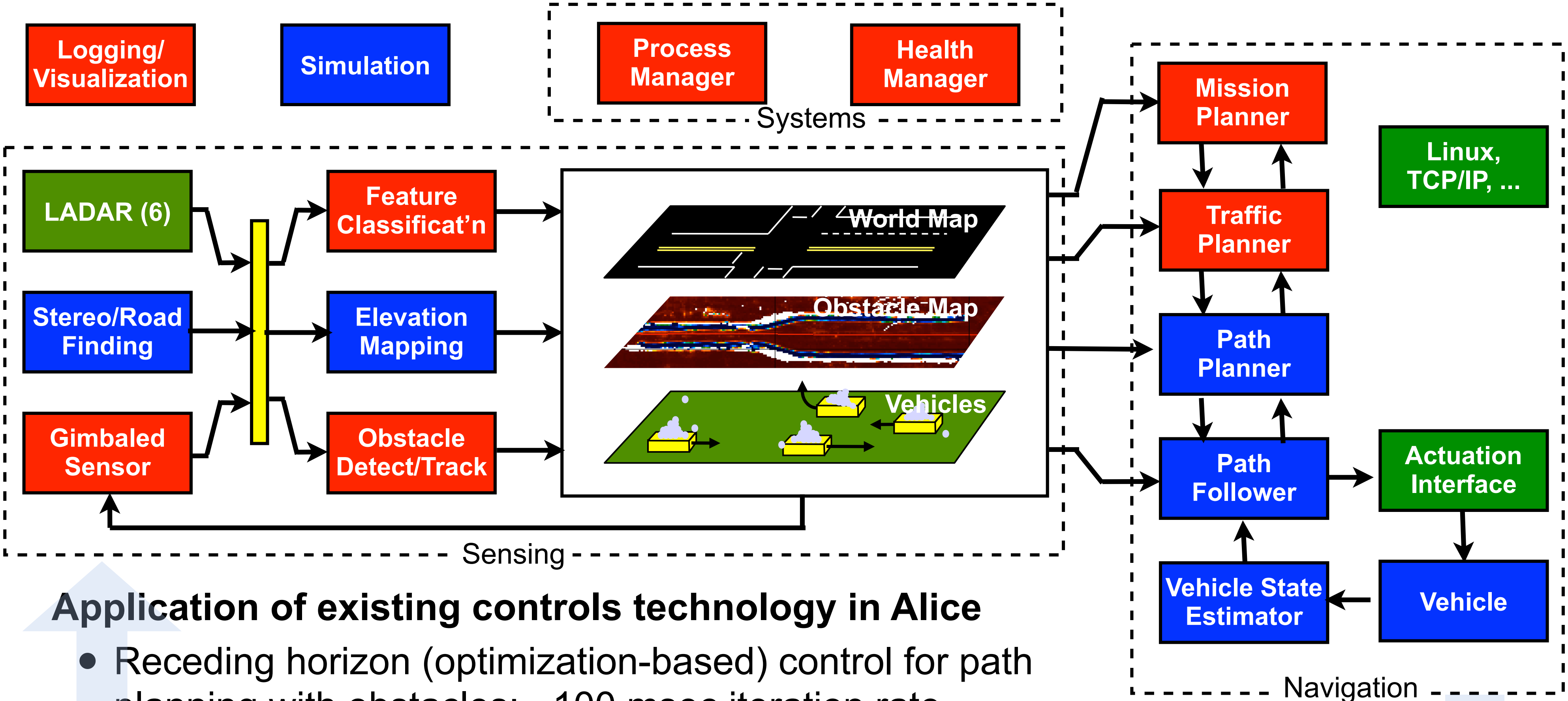
Software

- 25 programs with ~200 exec threads
- 237,467 lines of executable code





DGC07 System Architecture



Application of existing controls technology in Alice

- Receding horizon (optimization-based) control for path planning with obstacles; ~100 msec iteration rate
- Multi-layer sensor fusion: sensor "bus" allows different combinations of sensors to be used for perceptors + fusion at "map" level
- Low level (inner loop) controls: PID w/ anti-windup (but based on a feasible trajectory from RHC controller)

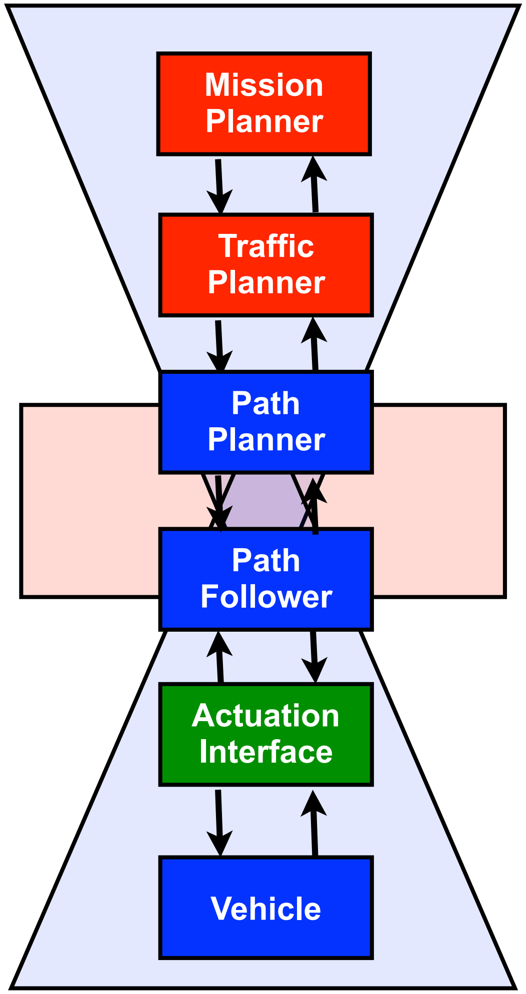
Properties

- Highly modular
- Rapidly adaptable
- Constantly viable
- Resilient ???

Planning Hourglass

Protocol stack based architecture

- Planners uses directives/responses to communicate
- Each layer is isolated from the ones above and below
- Had 4 different path planners under development, two different traffic planners.



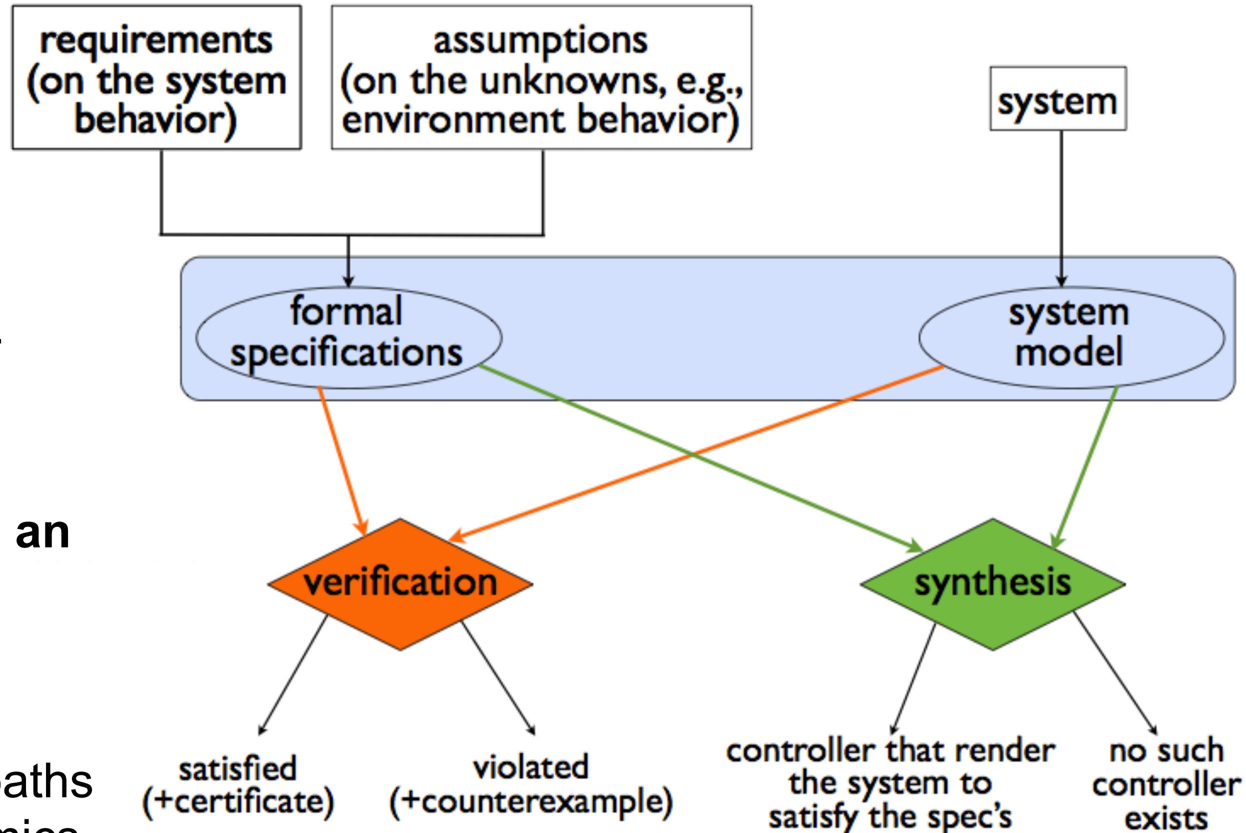
Engineering principle: layered protocols isolate interactions

- Define each layer to have a specific purpose; don't rely on knowledge of lower level details
- Important to pass information back and forth through the layers; a fairly in an actuator just generate a change in the path (and perhaps the mission)
- Higher layers (not shown) monitor health and can act as "hormones" (affecting multiple subsystems)

Hybrid system control methodology

- Finite state automata control interactions between layers and mode switches (intersection, off road, etc)
- Formal methods for analysis of control protocol correctness (post race)
  - Eg: make sure that you never have a situation where two layers are in conflict

Formal Methods for System Verification & Synthesis



Specification using LTL

- Linear temporal logic (LTL) is a math'l language for describing linear-time prop's
- Provides a particularly useful set of operators for constructing LT properties without specifying sets

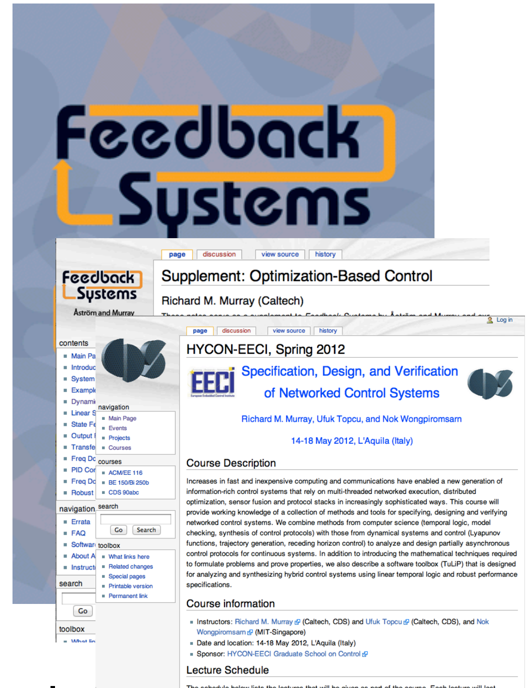
Existing methods for verifying an LTL specification

- *Theorem proving*: use formal logical manipulations
- *Model checking*: search for paths that satisfy the system dynamics (transition system) and violate the system specification (LTL formula)
  - If none, system is correct. Otherwise, return a counter example

Methods for *synthesis*: paths + finite state automata

- Feasible paths: use model checking to find a "counter-example" (= feasible solution)
- Finite state automata: determine how to react to environment to satisfy a spec

Summary: Control Theory



Two main principles of (feedback) control theory

- Feedback is a tool to provide robustness to uncertainty
  - Uncertainty = noise, disturbances, unmodeled dynamics
  - Useful for modularity: consistent behavior of subsystems
- Feedback is a tool to design the dynamics of a system
  - Convert unstable systems to stable systems
  - Tune the performance of a system to meet specifications
- Combined, these principles enable modularity and hierarchy

Control theory: past, present and future

- Tools were originally developed to help engineers design low-level control systems
- Increasing application of control theory for networked (hybrid) control systems
- New challenge: systematic design of layered architectures and control protocols

More information

- *Feedback Systems*: http://www.cds.caltech.edu/~murray/FBSwiki
- *Optimization-Based Control*: http://www.cds.caltech.edu/~murray/FBSwiki/OBC
- *Networked Control Systems*: http://www.cds.caltech.edu/~murray/wiki/NCS_course
- Additional references will be posted on the workshop wiki