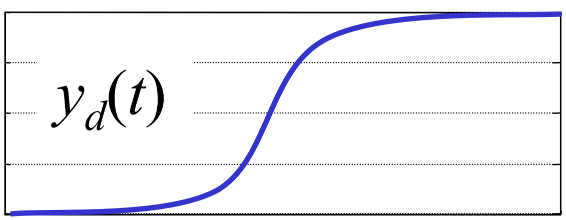
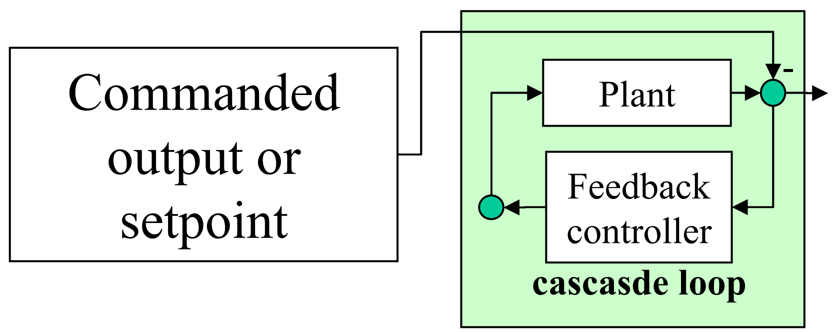
Lecture 6 – Outer Loop

- Setpoint profile generation
- Gain scheduling
- Feedforward and 2DOF design
- System inversion problem
- Feedforward for simple models
  - Zero order, first order, second order, oscillatory (input shaping)
- Iterative update of feedforward
  - Run-to-run, cascade loop

Setpoint profile generation

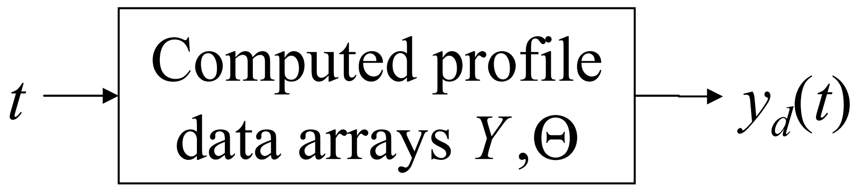
- Setpoint profile generation = path/trajectory planning
- Changing setpoint acts as a disturbance for the feedback loop.
- The closed-loop output follows the command accurately within the loop bandwidth
- A practical approach: choose the setpoint command (path) as a smooth function that has no/little high-frequency components.
- The smooth function can be a spline function etc





Setpoint profile

- Real-time replay of a pre-computed reference trajectory $y_d(t)$ or feedforward $v(t)$
- Reproduce a nonlinear function $y_d(t)$ in a control system

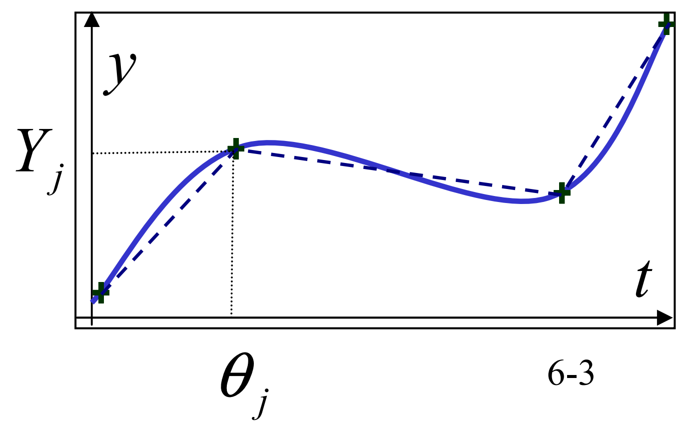


$$Y = \begin{bmatrix} Y_1 = y_d(\theta_1) \\ Y_2 = y_d(\theta_2) \\ \vdots \\ Y_n = y_d(\theta_n) \end{bmatrix}, \quad \Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

On-line computations:

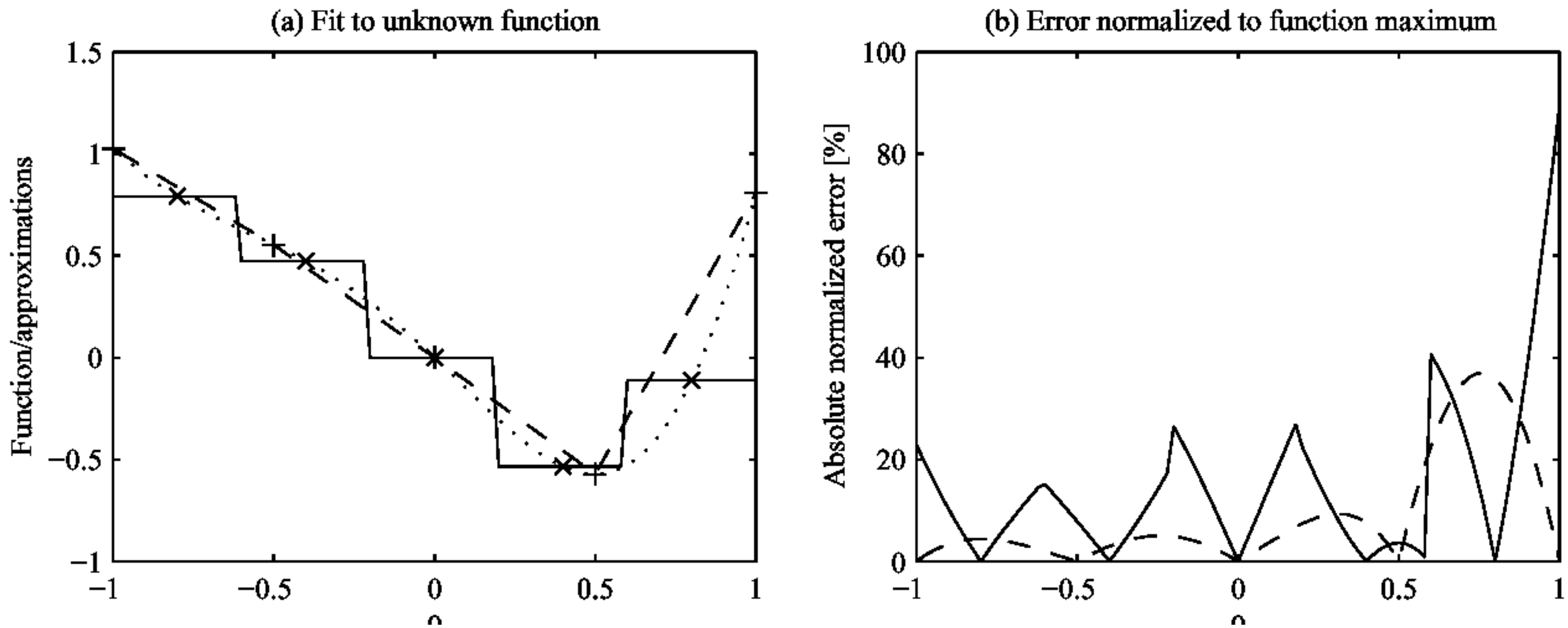
1. Find j, such that $\theta_j \le t \le \theta_{j+1}$
2. Compute linear interpolation

$$y_d(t) = Y_j \frac{\theta_{j+1} - t}{\theta_{j+1} - \theta_j} + Y_{j+1} \frac{t - \theta_j}{\theta_{j+1} - \theta_j}$$



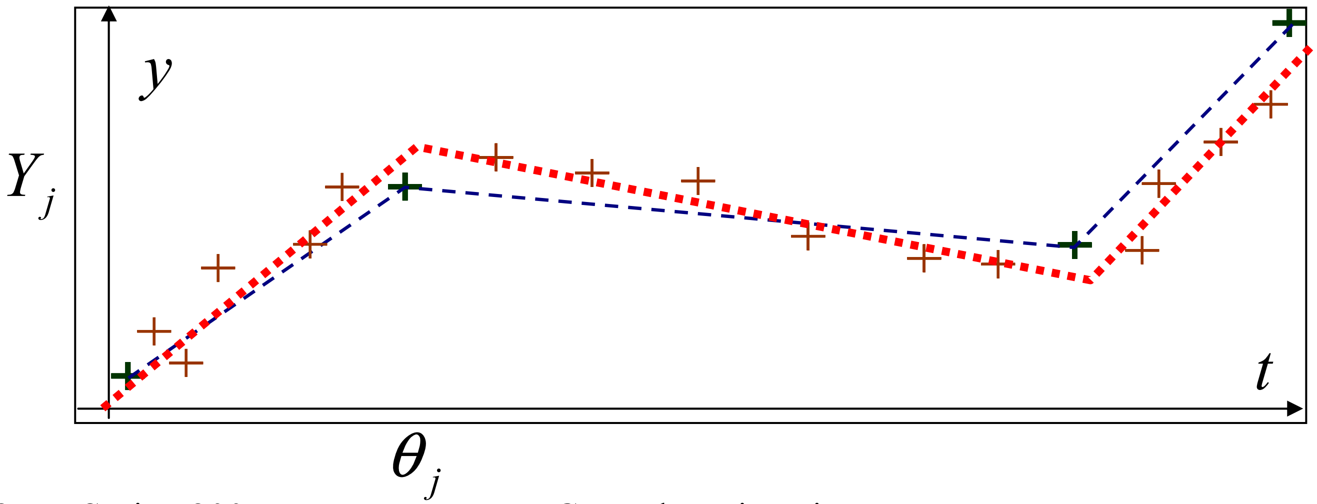
Linear interpolation vs. table look-up

- Linear interpolation is more accurate than a table look-up
- Requires less data storage
- At the expense of simple computation

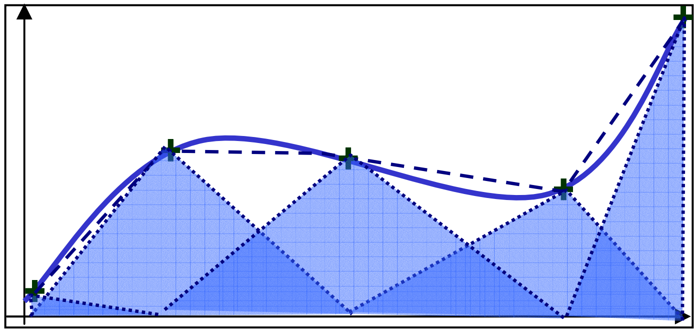


Approximation

- Interpolation:
  - compute function that will provide given values Y_j in the nodes θ_j
- Approximation
  - compute function that closely corresponds to given data, possibly with some error
  - might provide better accuracy throughout

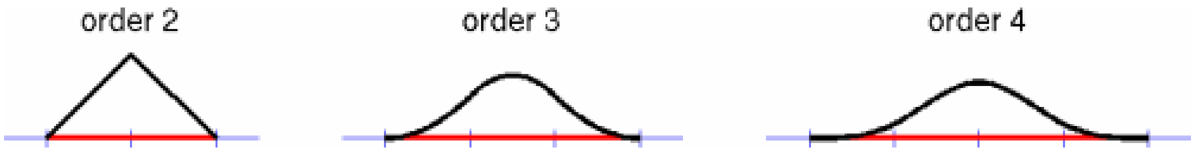


B-spline interpolation



- 1st-order
  - look-up table, nearest neighbor
- 2nd-order
  - linear interpolation

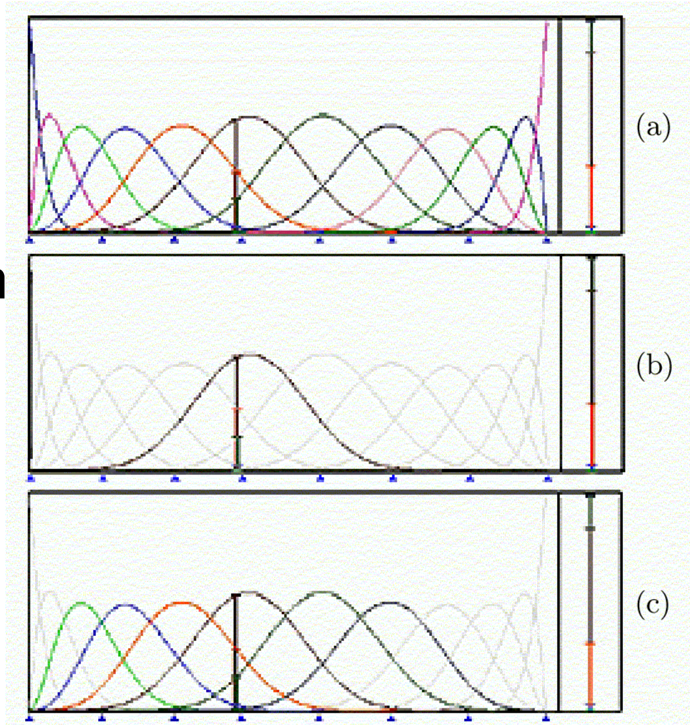
$$y_d(t) = \sum_j Y_j B_j(t)$$



- n-th order:
  - Piece-wise *n*-th order polynomials, continuous *n*-2 derivatives
  - Is zero outside a local support interval
  - The support interval extends to *n* nearest neighbors

B-splines

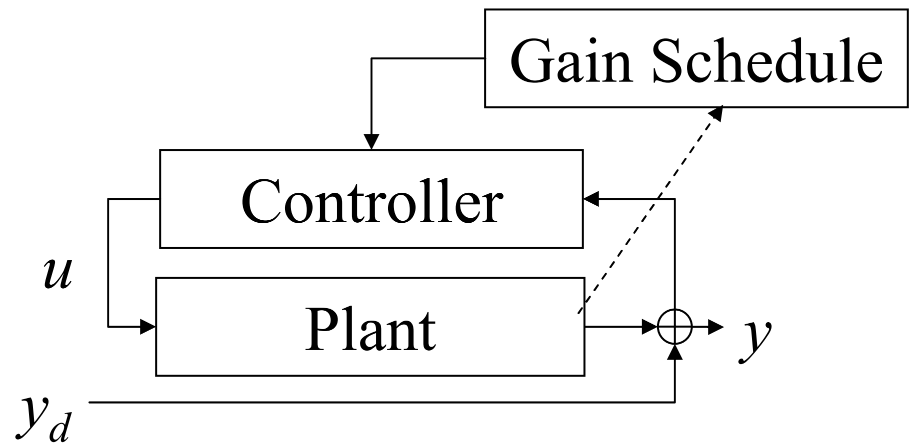
- Accurate interpolation of smooth functions with relatively few nodes
- For 1-D function the gain from using high-order B-splines is often not worth the added complexity
- Introduced and developed in CAD for 2-D and 3-D curve and surface data
- Are used for defining multidimensional nonlinear maps
- All you need to know that B-splines are useful. Actually using them would require learning available software.



Gain Scheduling

- Simple example

$$y = f(x) + g(x)u$$
$$u = -k(x)(y - y_d)$$

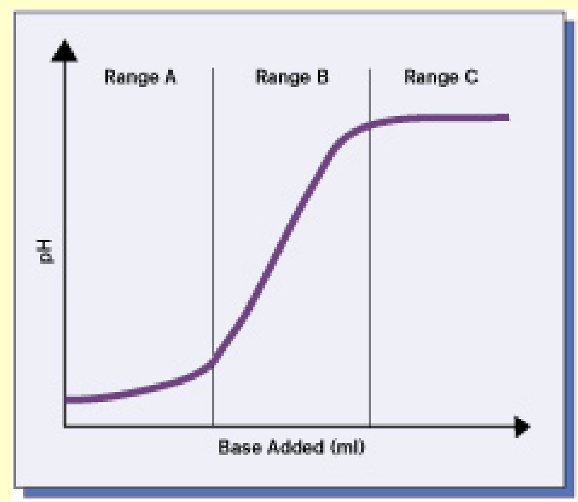


- Control design requires

$$k(x)$$

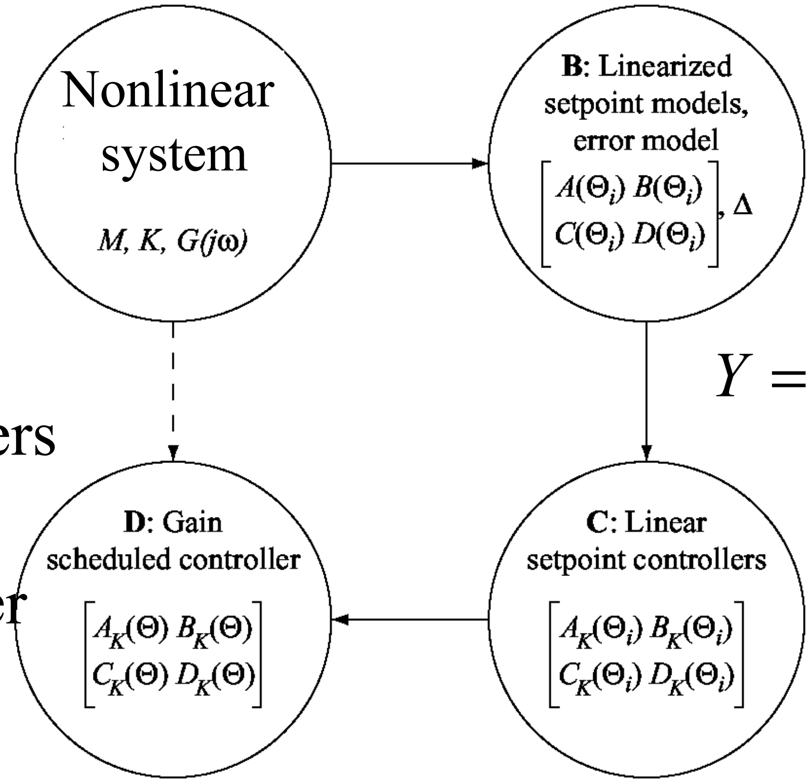
- The gain k is *scheduled* on x

Example: varying process gain



Gain scheduling

- Single out several regimes - model linearization or experiments
- Design linear controllers for these regimes
- Approximate controller dependence on the regime parameters



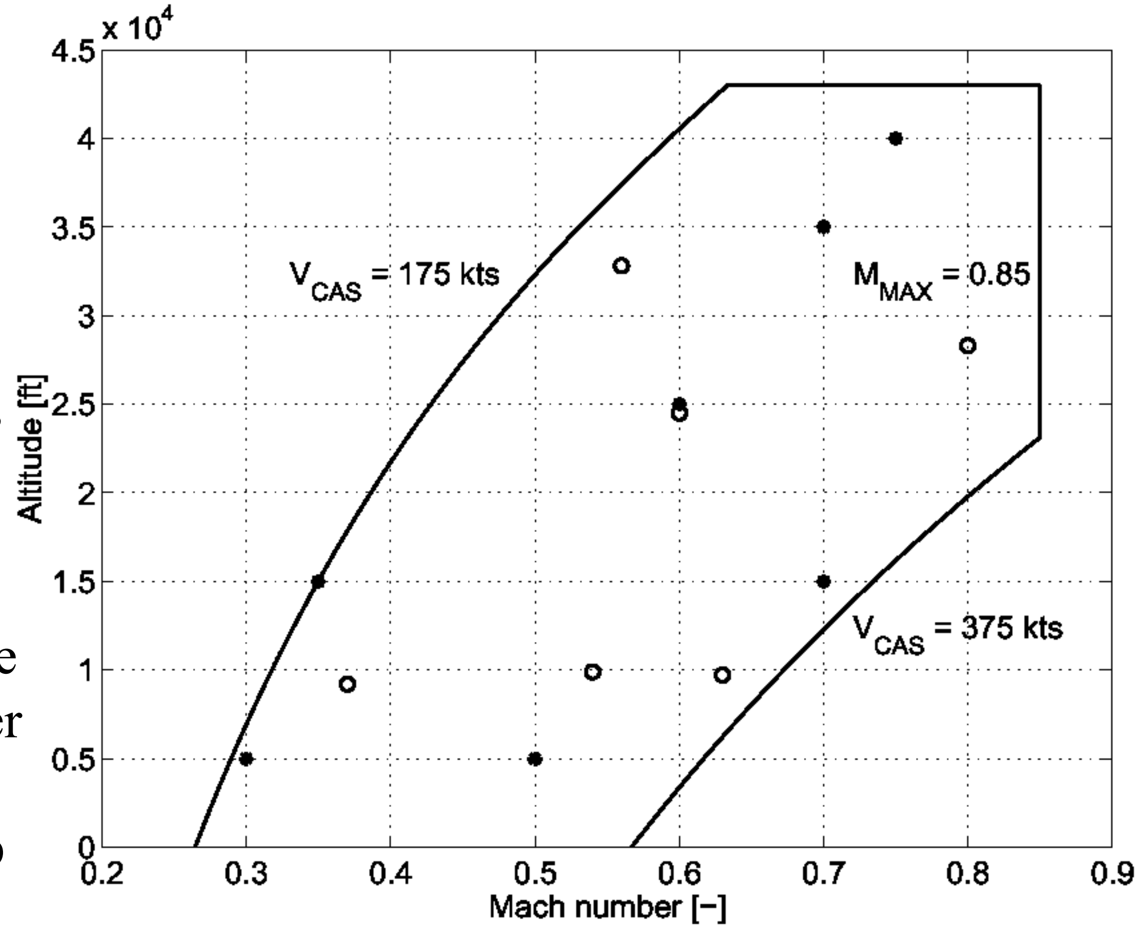
$$Y = \begin{bmatrix} \mathrm{vec}(A) \\ \mathrm{vec}(B) \\ \mathrm{vec}(C) \\ \mathrm{vec}(D) \end{bmatrix}$$

Linear interpolation:

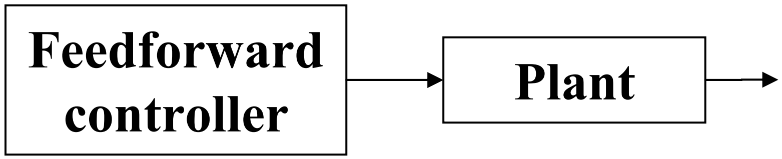
$$Y(\Theta) = \sum_j Y_j \varphi_j(\Theta)$$

Gain scheduling for aircraft

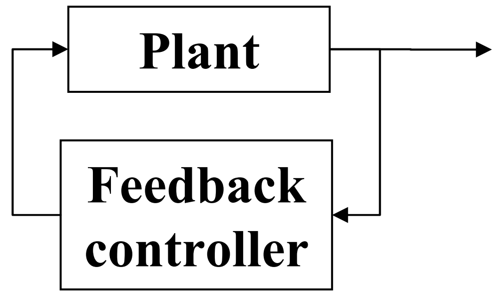
- Flight control
- Main trim condition parameters are used for scheduling
- Shown
  - Approximation nodes
  - Evaluation points
- Key assumption
  - Altitude and Mach are changing much slower than time constant of the flight control loop



Feedforward

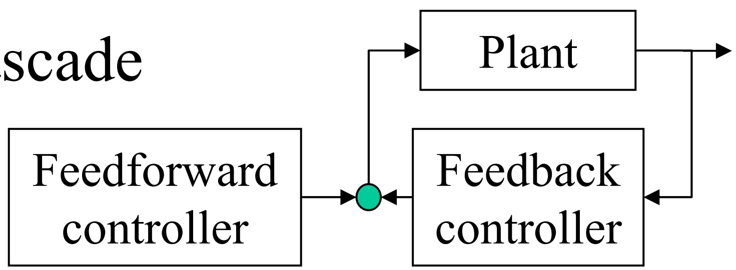


– this Lecture 6



– Lectures 3-5
– Lectures 7-8

- Main premise of the feedforward control: a model of the plant is known
- Model-based design of feedback control - the same premise
- The difference: feedback control is less sensitive to modeling error
- Common use of the feedforward: cascade with feedback



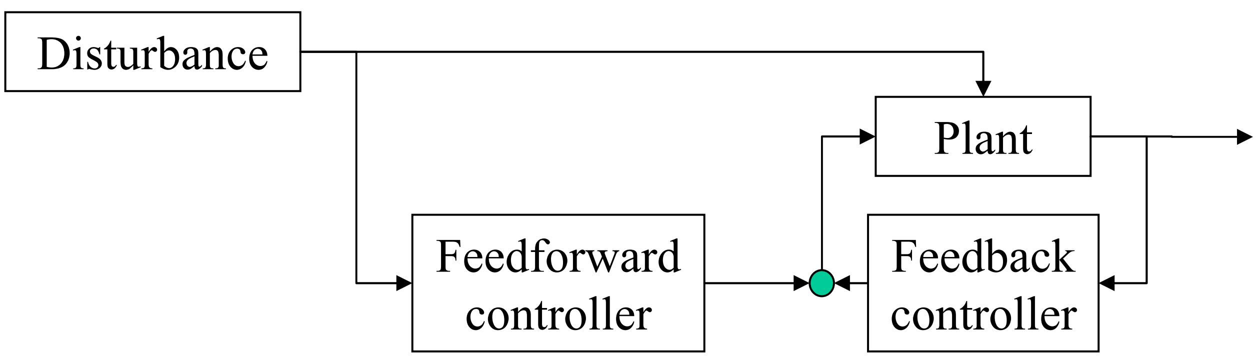
Why Feedforward?

- Model-based design means we know the system in advance
- The performance can be often greatly improved by adding open-loop control based on our system knowledge (models)

Disturbance feedforward

- Disturbance acting on the plant is measured
- Feedforward controller can react *before* the effect of the disturbance shows up in the plant output

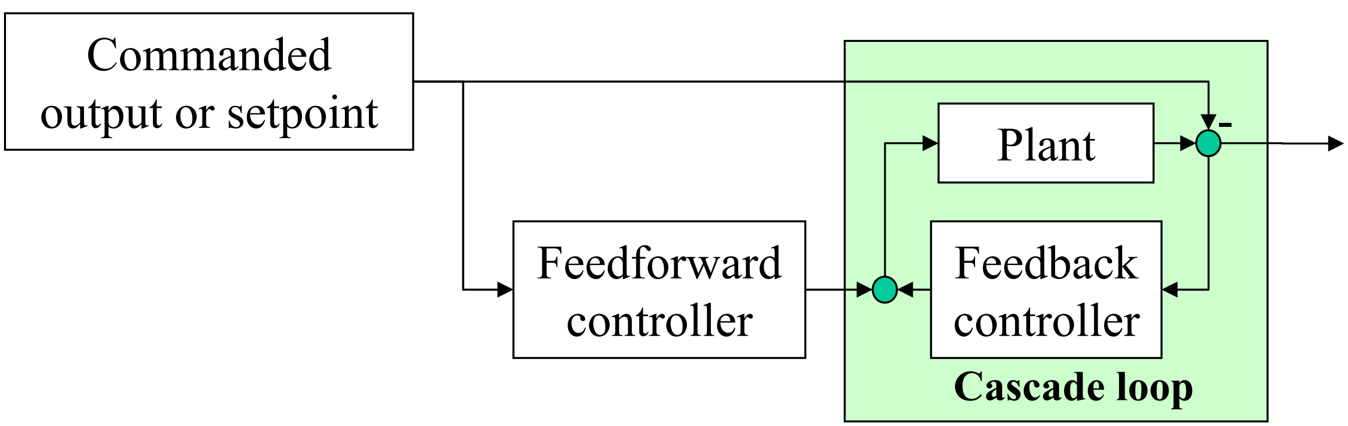
Example: Temperature control
- Measure ambient temperature and adjust heating/cooling
- homes and buildings
- district heating
- industrial processes
  - growing crystals
- electronic or optical components



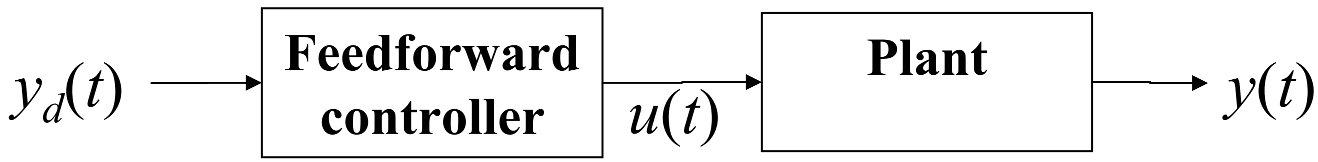
Command/setpoint feedforward

- The setpoint change acts as disturbance on the feedback loop.
- This disturbance can be measured
- 2-DOF controller architecture
  - Feedback controller
  - Feedforward controller
  - Joint design

Examples:
- Servosystems
  - Robotics
  - Aerospace
- Process control
  - RTP
- Propulsion (aero + auto)
  - Engine power demand



Feedforward as system inversion



$$y = P(s)u$$

- To get the output $y = y_d$ need to apply control $u_{FF} = [P(s)]^{-1} y_d$
- Simple example – feedthrough system with known gain

$$P(s) = g \qquad\qquad y = gu + w$$

- System inverse and feedforward

$$[P(s)]^{-1} = \frac{1}{g} \qquad\qquad u_{FF} = \frac{1}{g} y_d$$

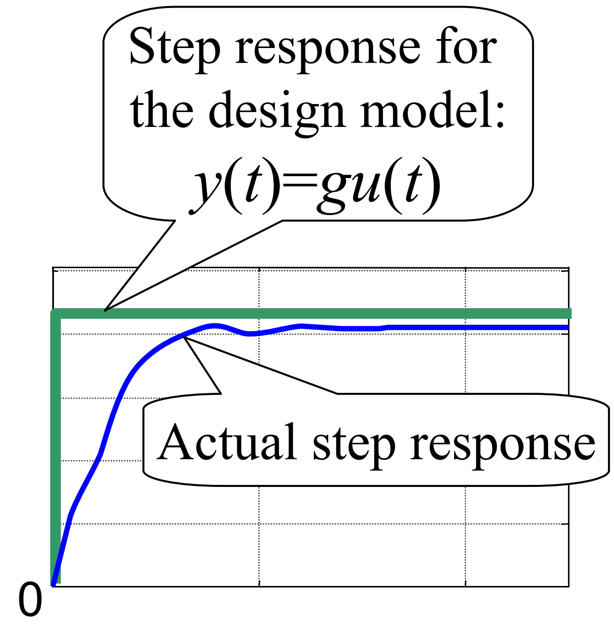
Feedforward for 0th order model

- Constant gain model (approximate)

$$y = gu + w$$

- There is a modeling error
- It might be desirable to introduce low pass filtering such that high frequencies are not excited by the feedforward

$$u_{FF} = \frac{1}{1+\tau s} \cdot \frac{1}{g} y_d$$



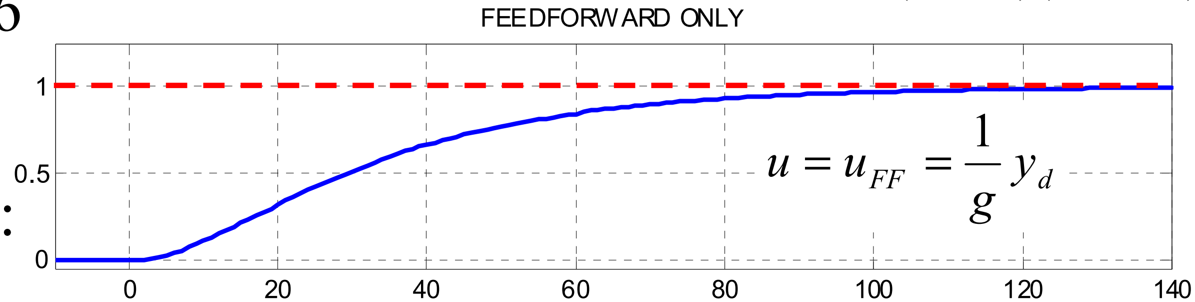
Setpoint Feedforward

- Example: processor thermal control – Lecture 4, Slide 6

$$H(s) = \frac{0.4}{(5s+1)(50s+1)}$$

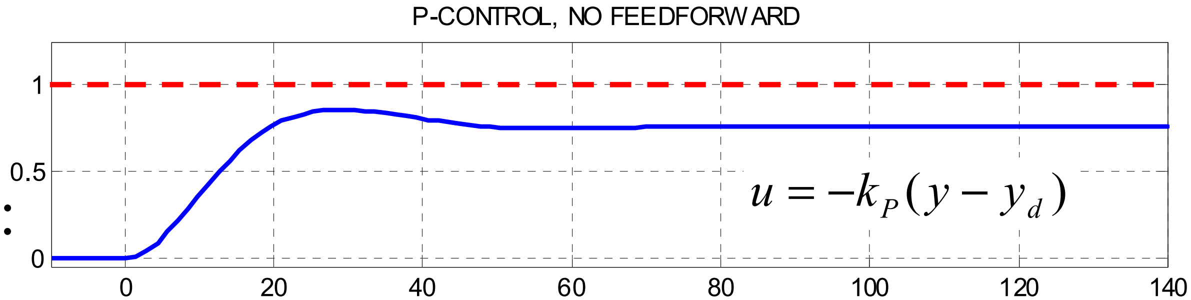
$g = 0.4$

Feedforward:

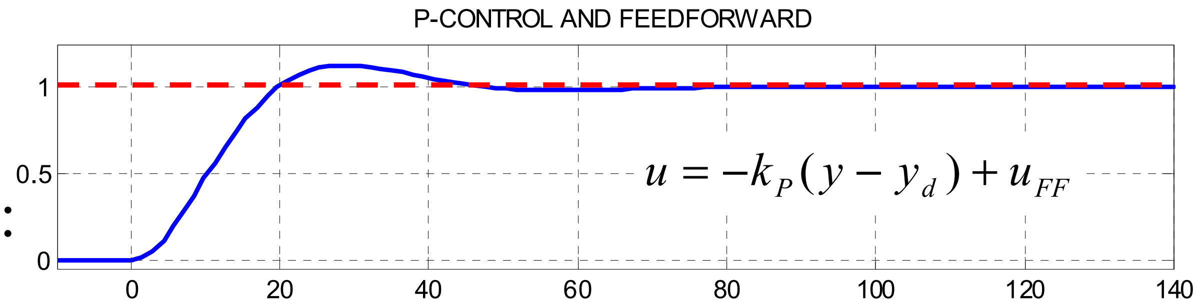


$k_P = 8$

P-feedback:



Feedforward + P-feedback:



Feedforward for 1st order model

- Simple 1st order model – integrator

$$y = \frac{1}{s}u + w$$

- Inverse system = differentiator

$$u_{FF} \approx s y_d$$

- Differentiating estimator (with low pass filtering)

$$u_{FF} = \frac{s}{1 + \tau s} y_d$$

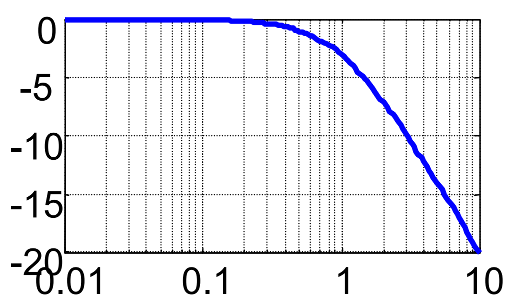
Feedforward as system inversion

$$y = P(s)u$$

$$y = y_d \Rightarrow u = [P(s)]^{-1} y_d$$

$$\tilde{u}(i\omega) = \frac{\tilde{y}_d(i\omega)}{P(i\omega)}$$

- Issue
  - High-frequency roll-off of the frequency response
  - Attempting inversion would result at growing high-frequency gain



$$P(s) = \frac{1}{1+s}$$ proper transfer function

$$[P(s)]^{-1} = 1 + s$$ not proper

- Approximate inverse solution:
  - ignore high frequency in some way

Proper transfer functions

- Proper means deg(Denominator) ≥ deg(Numerator)
- Strictly proper ←→ high-frequency roll-off, all physical dynamical systems are like that
- State space models are always proper
- Exact differentiation is noncausal, non-proper

Acceleration measurement example

Attempted perfect control

$$m\ddot{x} = u$$

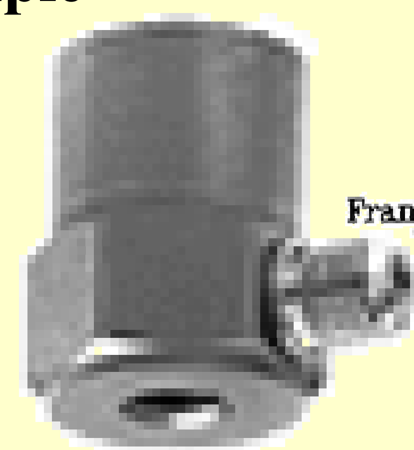
$$u = ma - k(x - x_d)$$

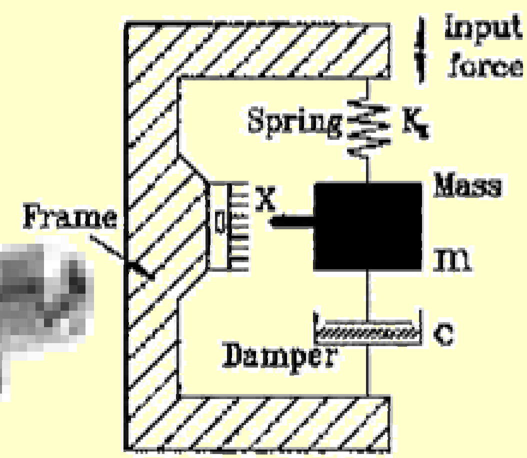
$$\Rightarrow x = x_d$$

$$a = \ddot{x}$$

this is wrong!

accelerometer





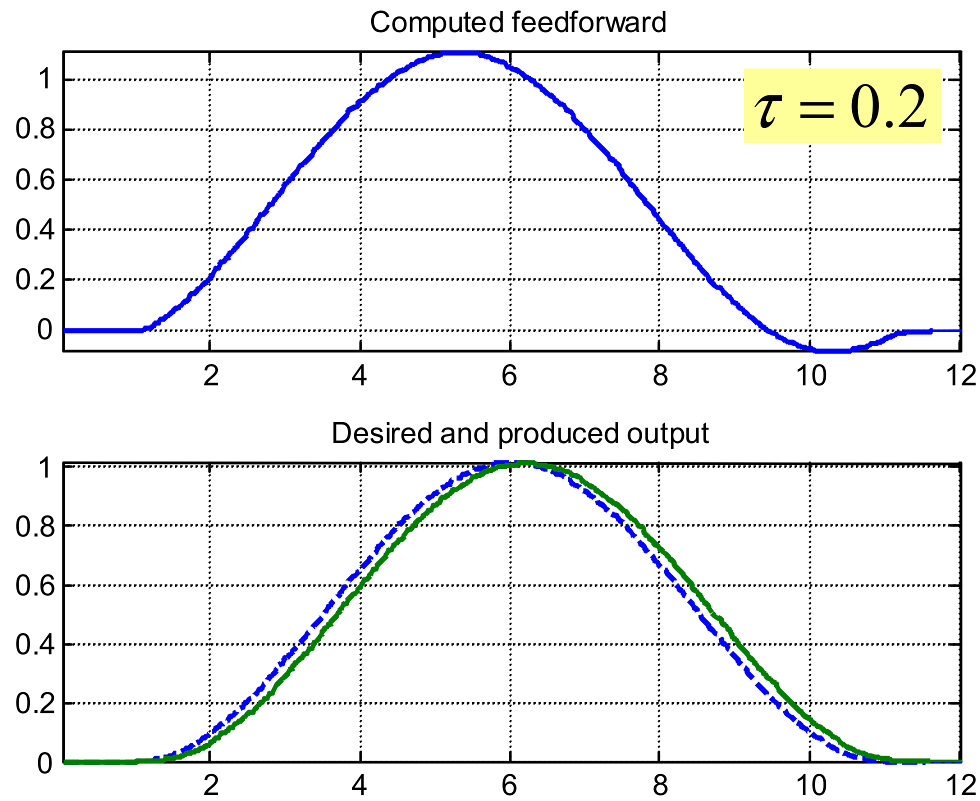
Approximate Differentiation

- Add low pass filtering:

$$P^{\dagger}(s) = \frac{1}{(1+\tau s)^n} \cdot \frac{1}{P(s)}$$

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

$$P^{\dagger}(s) = \frac{1}{1+\tau s} \cdot (1+s)$$



Differentiation

- Setpoint profile = path/trajectory planning
- The derivative can be computed if $y_d(t)$ is known ahead of time (no need to be causal then).

$$P^{-1}(s)y_d = \frac{1}{P(s)} \cdot \frac{1}{s^n} y_d^{[n]}, \qquad y_d^{[n]}(t) = \frac{d^n y}{dt^n}(t)$$

- This could be done by computing the profile $y_d(t)$ as an output of an integrator chain

$$y_d(t) = \frac{1}{s^n} a(t) \qquad y_d^{[k]}(t) = s^k y_d(t) = \frac{1}{s^{n-k}} a(t)$$

Example

$$P(s) = \frac{1}{s^2} \qquad P^{-1}(s)y_d = s^2 y_d = a(t)$$

Compute the setpoint profile as $a(t)$

Double integrator example

- Double integrator model

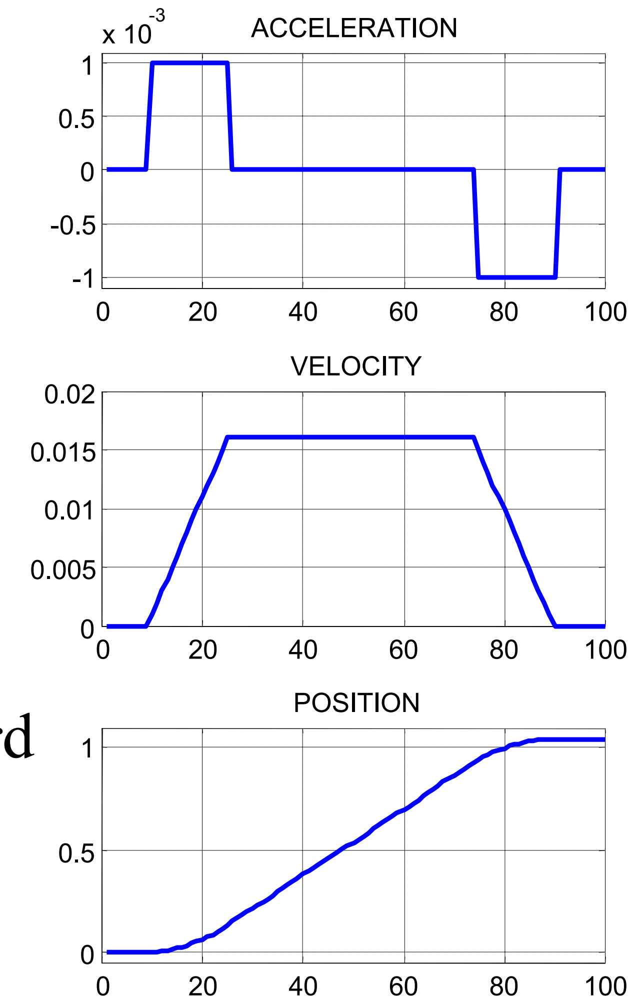
$$y = \frac{1}{s^2} u$$

- Setpoint profile

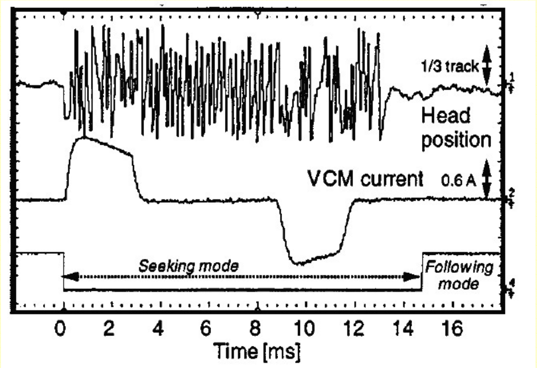
$$y_d = \frac{1}{s^2} a$$

- Feedforward

$$u_{FF} = a(t)$$

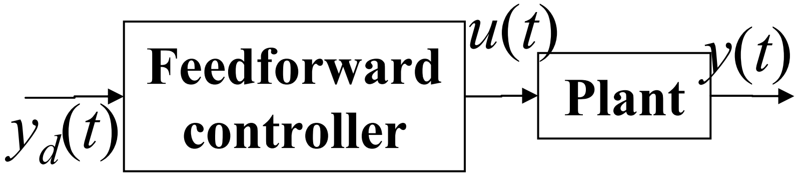


Example:
Disk drive long seek.
Move the R/W head a target track



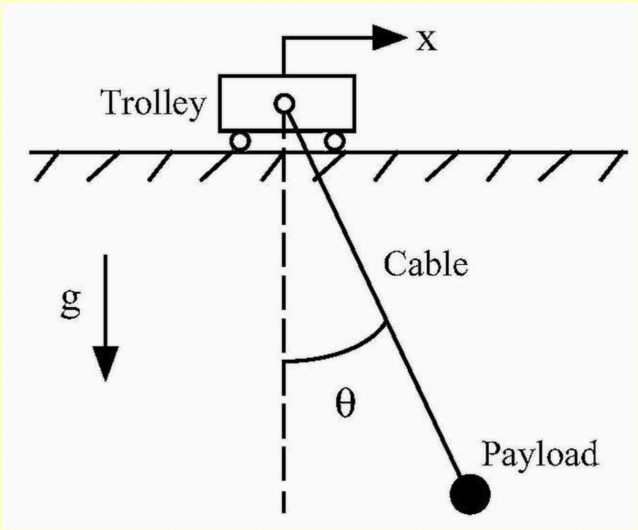
Hara et at. IEEE Tr. on Mechatronics, March 2000

Input Shaping: point-to-point control



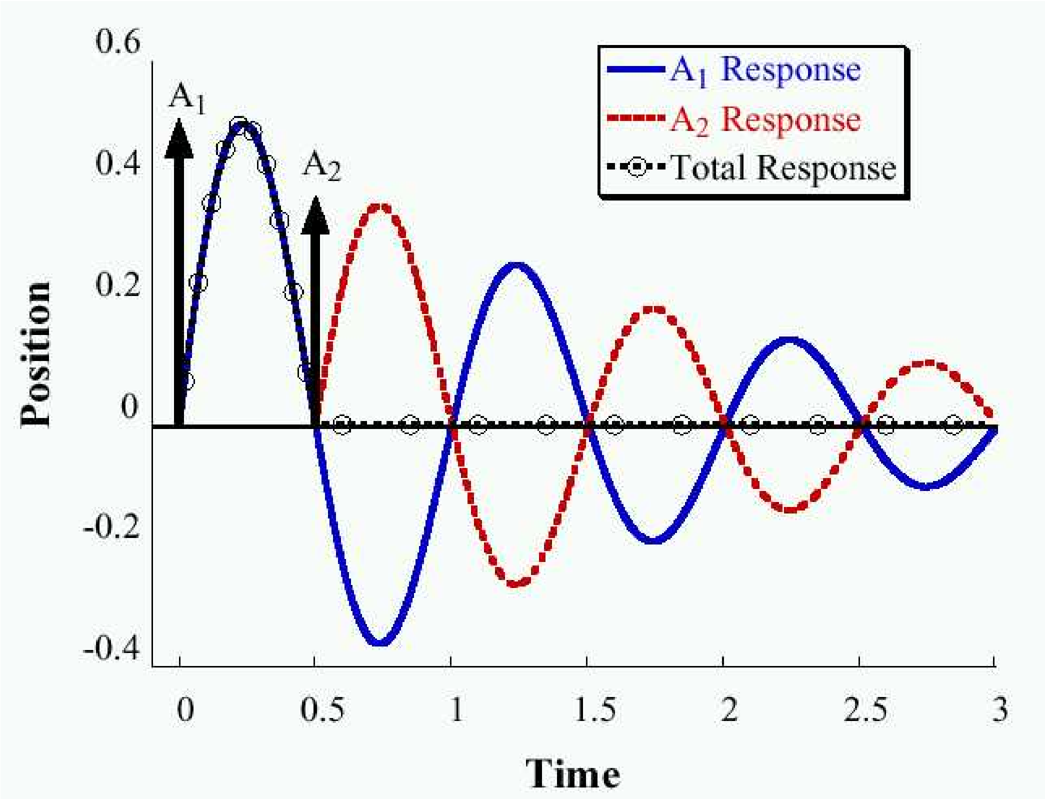
- Given initial and final conditions find control input
- No intermediate trajectory constraints
- Lightly damped, imaginary axis poles
  - inversion methods do not work well
- FIR notch fliter
  - Seering and Singer, MIT
  - Convolve Inc.

Examples:
- Disk drive long seek with flexible modes
- Flexible space structures
- Overhead gantry crane



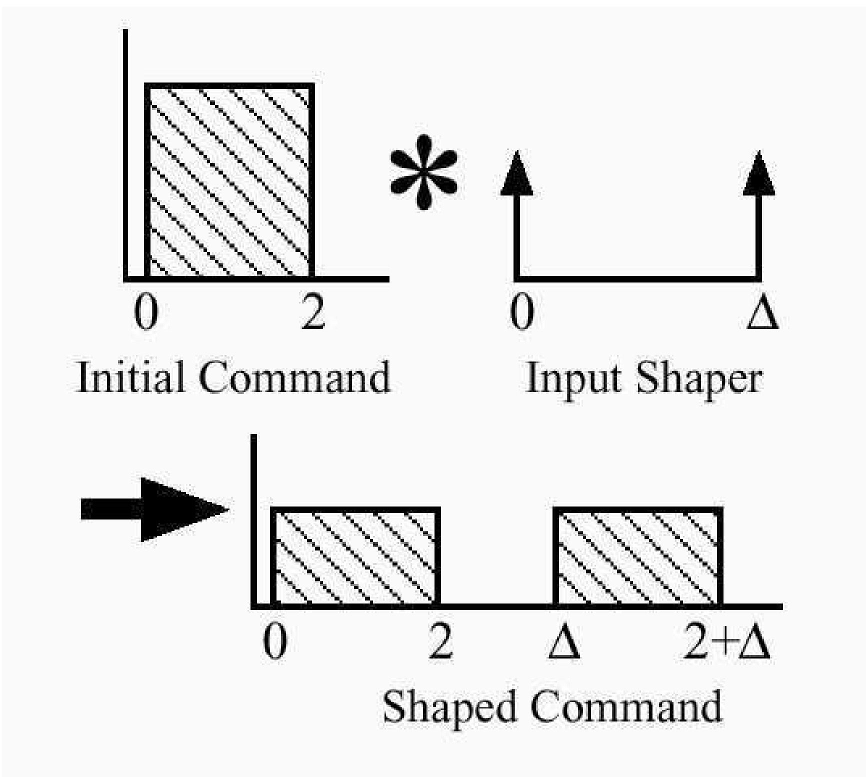
Pulse Inputs

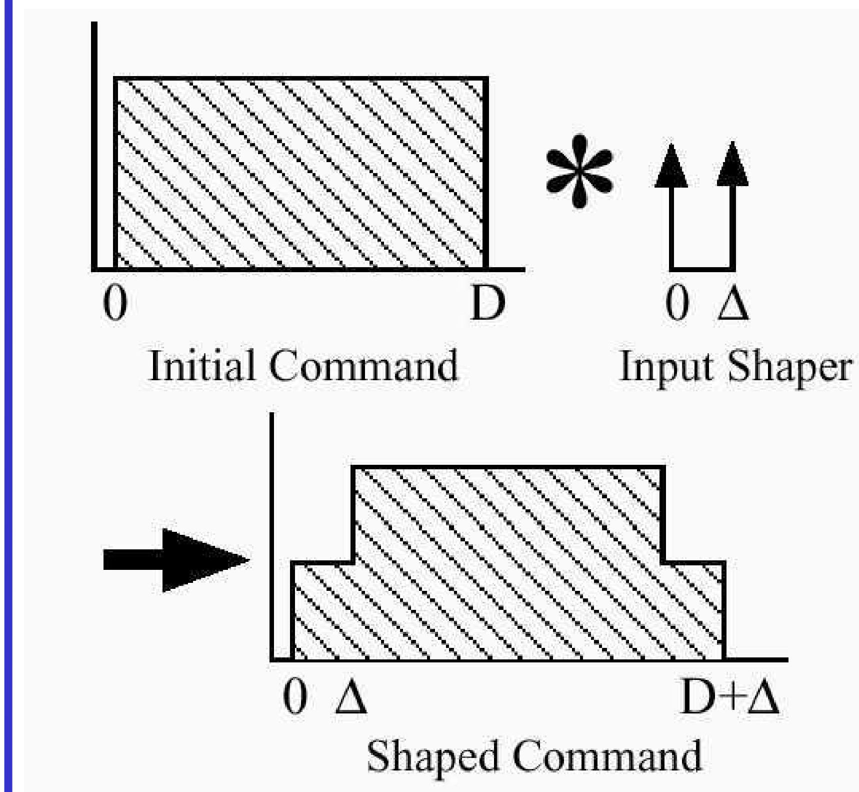
- Compute pulse inputs such that there is no vibration.
- Works for a pulse sequence input
- Can be generalized to *any* input



Input Shaping as signal convolution

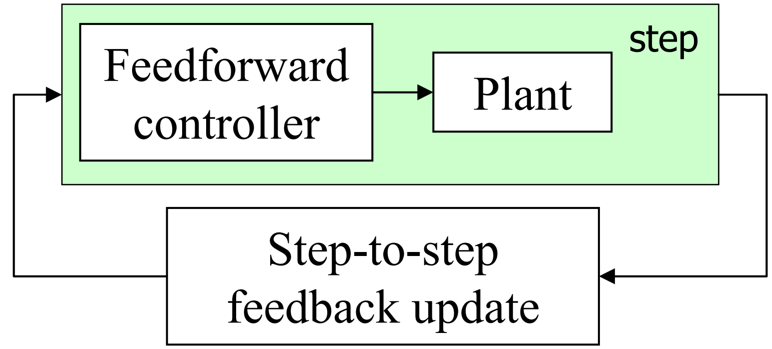
- Convolution: $f(t) * \left(\sum A_i \delta(t - t_i)\right) = \sum A_i f(t - t_i)$





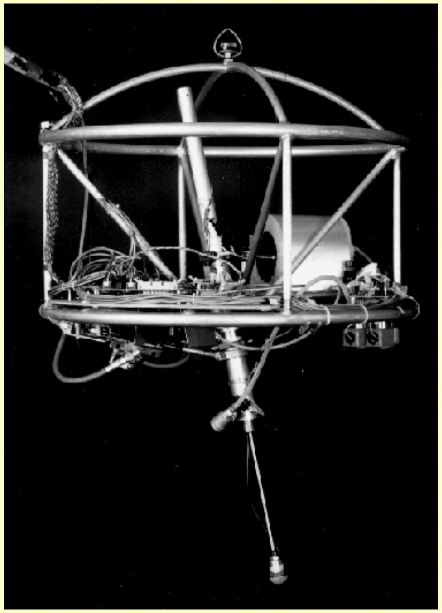
Iterative update of feedforward

- Repetition of control tasks
- Robotics
  - Trajectory control tasks: Iterative Learning Control
  - Locomotion: steps
- Batch process control
  - Run-to-run control in semiconductor manufacturing
  - Iterative Learning Control (*IEEE Control System Magazine*, Dec. 2002)



Example:
One-legged hopping machine (M.Raibert)

Height control:
$y_d = y_d(t-T_n;a)$
$h(n+1)=h(n)+Ga$



More on Feedforward…

- Iterative update
  - Iterative Learning Control, run-to-run update
  - Repetitive dynamics (repeating robotics mechanism motion)
- Replay pre-computed sequences
  - Look-up tables, maps
- Also used in practice
  - Servomechanism, disturbance model
  - Adaptive feedforward
    - LMS update
    - Sinusoidal disturbance tracking, e.g. in disk drives (related to PLL)