Lecture 5 - Feedforward

- Programmed control
- Path planning and nominal trajectory feedforward
- Feedforward of the disturbance
- Reference feedforward, 2-DOF architecture
- Non-causal inversion
- Input shaping, flexible system control
- Iterative update of feedforward
Why Feedforward?

- Feedback works even if we know little about the plant dynamics and disturbances
- Was the case in many of the first control systems
- Much attention to feedback - for historical reasons

- Open-loop control/feedforward is increasingly used
- Model-based design means we know something
- The performance can be greatly improved by adding open-loop control based on our system knowledge (models)
Feedforward

- 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

Feedforward controller → Plant

Plant

Feedback controller

– this Lecture 5
– Lecture 4 PID
– Lecture 6 Analysis
– Lecture 7 Design
Open-loop (programmed) control

• Control $u(t)$ found by solving an optimization problem. Constraints on control and state variables.
• Used in space, missiles, aircraft FMS
  – Mission planning
  – Complemented by feedback corrections
• Sophisticated mathematical methods were developed in the 60s to overcome computing limitations.
• Lecture 12 will get into more detail of control program optimization.

\[
\dot{x} = f(x, u, t) \\
J(x, u, t) \rightarrow \min \\
x \in X, u \in U \\
\text{Optimal control: } u = u_*(t)
\]
Optimal control

• Performance index and constraints
• Programmed control
  – compute optimal control as a time function for particular initial (and final) conditions
• Optimal control synthesis
  – find optimal control for any initial conditions
  – at any point in time apply control that is optimal now, based on the current state. This is feedback control!
  – simplified model, toy problems, conceptual building block
• MPC - will discuss in Lecture 12
Path/trajectory planning

• The disturbance caused by the change of the command $r$ influences the feedback loop.
• The error sensitivity to the reference $R(s)$ is bandpass: $|R(i\omega)|<<1$ for $\omega$ small
• A practical approach: choose the setpoint command (path) as a smooth function that has no/little high-frequency components. No feedforward is used.
• The smooth function can be a spline function etc
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 - crystallization
- 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

Examples:
- Servosystems
  - robotics
- Process control
  - RTP
- Automotive
  - engine torque demand
Feedforward as system inversion

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

- Simple example:

\[ P(s) = \frac{1 + 2s}{1 + s} \]
\[ [P(s)]^{-1} = \frac{1 + s}{1 + 2s} \]

\[ e = P(s)u + D(s)d \]
\[ y_d = -D(s)d \]

More examples:
- Disk drive long seek
- Robotics: tracking a trajectory
Feedforward as system inversion

\[ y = P(s)u \]
\[ y = y_d \implies u = [P(s)]^{-1} y_d \]
\[ \tilde{u}(i\omega) = \frac{\tilde{y}_d(i\omega)}{P(i\omega)} \]

- Issue
  - high-frequency roll-off

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

\[ P(s) = \frac{1}{1 + s} \]
\[ [P(s)]^{-1} = 1 + s \]
Proper transfer functions

- Proper means $\deg(\text{Denominator}) \geq \deg(\text{Numerator})$
- Strictly proper $\iff$ high-frequency roll-off, all physical dynamical systems are like that
- Proper $= \text{strictly proper} + \text{feedthrough}$
- State space models are always proper
- Exact differentiation is noncausal, non-proper
- Acceleration measurement example

\[
m \ddot{x} = u \\
u = ma - k(x - x_d) \\
\Rightarrow x = x_d
\]
Differentiation

- Path/trajectory planning - mechanical servosystems
- The derivative can be computed if $y_d(t)$ is known ahead of time (no need to be causal then).

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

\[
P(s) = \frac{1}{1 + s}
\]

\[
P^{-1}(s) y_d = \frac{1 + s}{s} \dot{y}_d = \left(1 + \frac{1}{s}\right) \dot{y}_d = \dot{y}_d + y_d
\]
Approximate Differentiation

- Add low pass filtering:

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

\[
P(s) = \frac{1}{1 + s}
\]

\[
P^\dagger(s) = \frac{1}{1 + \tau s} \cdot (1 + s)
\]
‘Unstable’ zeros

- Nonminimum phase system
  - r.h.p. zeros → r.h.p. poles
  - approximate solution: replace r.h.p. zeros by l.h.p. zeros

\[ P(s) = \frac{1-s}{1+0.25s}, \quad P^+(s) = \frac{1+0.25s}{1+s} \]

- RHP zeros might be used to approximate dead time
  - exact causal inversion impossible

\[ P(s) = e^{-2Ts} \approx \frac{1-sT}{1+sT} \]

- If preview is available, use a lead to compensate for the deadtime
Two sided $z$-transform, non-causal system

- Linear system is defined by a pulse response. Do not constrain ourselves with a causal pulse response anymore

\[ y(x) = \sum_{k=-\infty}^{\infty} h(x - k)u(k) \]

- 2-sided $z$-transform gives a “transfer function”

\[ P(z) = \sum_{k=-\infty}^{\infty} h(k)z^{-k} \]

- Fourier transform/Inverse Fourier transform are two-sided

Impulse response decay

- Decay rate from the center = $\log r$
Non-causal inversion

- Causal/anti-causal decomposition
  - 2-sided Laplace-transform

\[ P(s) = \frac{1 - s}{1 + 0.25s} \]

\[ P^{-1}(s) = \frac{1 + 0.25s}{1 - s} = -0.25 + \frac{1.25}{1 - s} \]

\[ P^{-1}(i\omega) = \frac{1}{P(i\omega)} \]

iFFT

causal \[ \frac{1}{1 - s} \]

anti-causal \[ \frac{1}{1 - s} \]
Frequency domain inversion

• Regularized inversion: \[ \|y_d - Pu\|_2^2 + \rho \|u\|_2^2 \rightarrow \text{min} \]
  \[ \int \left( \|y_d(i\omega) - P(i\omega)u(i\omega)\|^2 + \rho |u(i\omega)|^2 \right) d\omega \rightarrow \text{min} \]
  \[ u(i\omega) = \frac{P^*(i\omega)}{P^*(i\omega)P(i\omega) + \rho} \quad y_d(i\omega) = P^+(i\omega) y_d(i\omega) \]

• Systematic solution
  – simple, use FFT
  – takes care of everything
  – noncausal inverse
  – high-frequency roll-off
  – Paden & Bayo, 1985(?)
Input Shaping: point-to-point control

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

Examples:
- Disk drive long seek
- 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

![Diagram of pulse inputs and responses over time](image-url)
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 

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

Height control:
\[ y_d = y_d(t-T_n; a) \]
\[ h(n+1) = h(n) + Ga \]
Feedforward Implementation

- Constraints and optimality conditions known ahead of time
  - programmed control
- Disturbance feedforward in process control
  - has to be causal, system inversion
- Setpoint change, trajectory tracking
  - smooth trajectory, do not excite the output error
  - in some cases have to use causal ‘system inversion’
  - preview might be available from higher layers of control system, noncausal inverse
- Only final state is important, special case of inputs
  - input shaping - notch filter
  - noncausal parameter optimization
Feedforward Implementation

• Iterative update
  – ILC
  – run-to-run
  – repetitive dynamics

• Replay pre-computed sequences
  – look-up tables, maps

• Not discussed, but used in practice
  – Servomechanism, disturbance model
  – Sinusoidal disturbance tracking - PLL
  – Adaptive feedforward, LMS update