

# Feedforward Design

## Real-Time Systems, Lecture 10

---

Karl-Erik Årzén

8 February 2018

Lund University, Department of Automatic Control

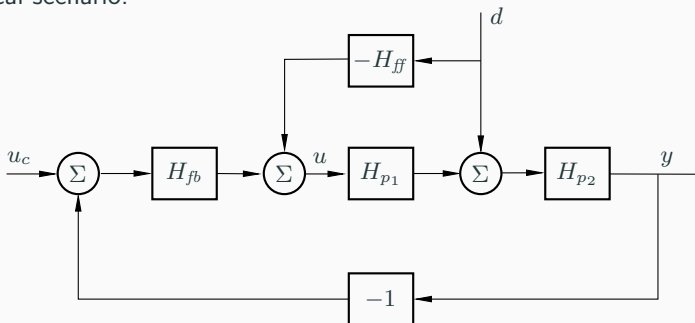
# Lecture 10 – Feedforward Design

[IFAC PB Chapter 9; These slides]

- Reduction of measurable disturbances by feedforward
- Using feedforward to improve setpoint response
  - The servo problem
  - Reference generation – input–output approach
  - Reference generation – state-space approach
    - Nonlinear reference generation

# Reduction of measurable disturbances by feedforward

Typical scenario:



Pulse transfer function from measured disturbance  $d$  to output  $y$ :

$$Y(z) = \frac{H_{p2}(z)(1 - H_{p1}(z)H_{ff}(z))}{1 + H_{p2}(z)H_{p1}(z)H_{fb}(z)}D(z)$$

To completely eliminate the disturbance, select

$$H_{ff}(z) = H_{p1}^{-1}(z)$$

# System Inverses

Assume

$$H(z) = \frac{B(z)}{A(z)} \quad \Rightarrow \quad H^{-1}(z) = \frac{A(z)}{B(z)}$$

Potential problems:

- Inverse not causal if pole excess  $d = \deg A - \deg B \geq 1$
- Inverse not stable if  $B(z)$  has zeros outside unit circle

One possible solution:

- Factor  $B(z)$  as  $B^+(z)B^-(z)$ 
  - $B^+(z)$  has all its zeros inside unit circle
  - $B^-(z)$  has all its zeros outside unit circle
- Use the approximate inverse

$$H^\dagger(z) = \frac{A(z)}{z^d B^+(z) B^{*-}(z)}, \quad \text{where } B^{*-}(z) = z^{\deg B^-} B^-(z^{-1})$$

- $B^{*-}(z)$  - the mirror in the unit circle of the zeros outside the unit circle

## Approximate Inverse – Example

Let

$$G(s) = \frac{6(1-s)}{(s+2)(s+3)}$$

ZOH sampling with  $h = 0.1$  gives

$$H(z) = \frac{-0.4420(z - 1.106)}{(z - 0.8187)(z - 0.7408)} = \frac{B(z)}{A(z)}$$

$H^{-1}(z)$  noncausal and unstable. Approximate inverse:

$$B^+(z) = 1, \quad B^-(z) = -0.4420(z - 1.106), \quad d = 1, \quad \deg B^- = 1$$

$$B^{*-}(z) = -0.4420z(z^{-1} - 1.106) = -0.4420(1 - 1.106z)$$

$$H^\dagger(z) = \frac{(z - 0.8187)(z - 0.7408)}{-0.4420z(1 - 1.106z)} = \frac{(z - 0.8187)(z - 0.7408)}{0.488z(z - 0.904)}$$

Stable and causal

# Two Classes of Control Problems

**Regulation problems:** compromise between rejection of load disturbances and injection of measurement noise

- Feedback
- Lecture 9 - Last lecture
- Feedforward
- Lecture 10 - Today

**Servo problems:** make the output respond to command signals in the desired way

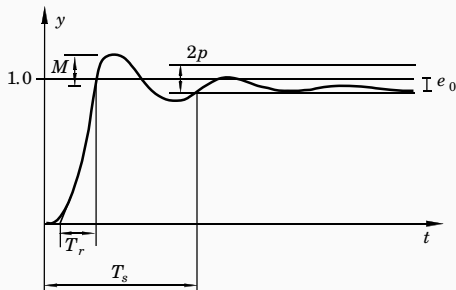
- Feedforward
- Lecture 10 - Today

# Using feedforward to improve setpoint response

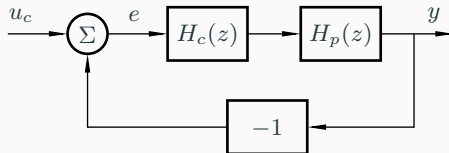
**The servo problem:** Make the output respond to setpoint changes in the desired way

Typical design criteria:

- Rise time,  $T_r$
- Overshoot,  $M$
- Settling time,  $T_s$
- Steady-state error,  $e_0$
- ...



# Simplistic Setpoint Handling – Error Feedback



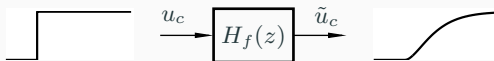
Potential problems:

- Step changes in the setpoint can introduce very large control signals
- The same controller  $H_c(z)$  must be tuned to handle both disturbances and setpoint changes
  - No separation between the regulator problem and the servo problem

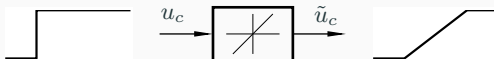


# Common Quick Fixes

- Filter the setpoint signal



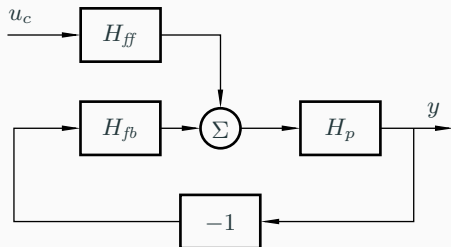
- Rate-limit the setpoint signal



- Introduce setpoint weighting in the controller
  - E.g. PID controller with setpoint weightings  $\beta$  and  $\gamma$

# A More General Solution

Use a two-degree-of-freedom (2-DOF) controller, e.g.:



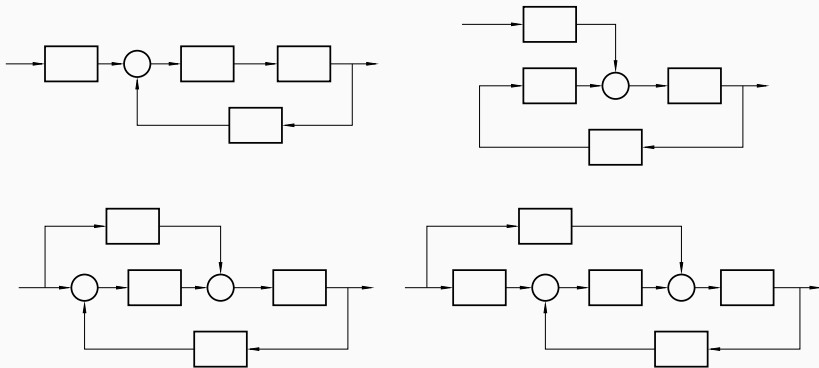
Design procedure:

1. Design feedback controller  $H_{fb}$  to get good regulation properties (attenuation of load disturbances and measurement noise)
2. Design feedforward compensator  $H_{ff}$  to obtain the desired servo performance

Separation of concerns

## 2-DOF Control Structures

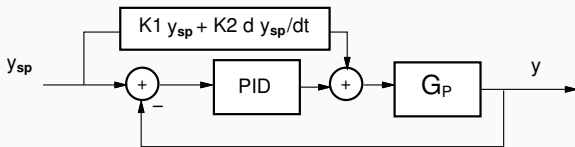
A 2-DOF controller can be represented in many different ways, e.g.:



For linear systems, all these structures are equivalent

## Example: PID with Setpoint Weighting

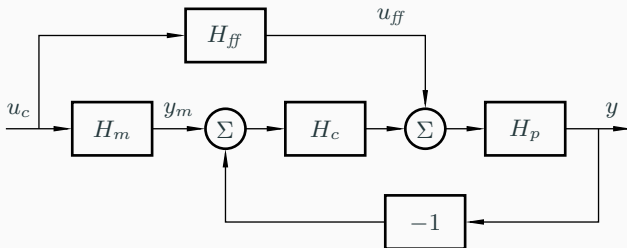
$$\begin{aligned} u &= K \left( \beta y_{sp} - y + \frac{1}{T_I} \int (y_{sp} - y) d\tau + T_D \frac{d}{dt} (\gamma y_{sp} - y) \right) \\ &= K \left( e + \frac{1}{T_I} \int e d\tau + T_D \frac{de}{dt} \right) \\ &\quad + \underbrace{K(\beta - 1) y_{sp}}_{K_1} + \underbrace{T_D K(\gamma - 1) \frac{dy_{sp}}{dt}}_{K_2} \end{aligned}$$



Interpretation: Error feedback + feedforward from  $y_{sp}$

# Reference Generation – Input–Output Approach

2-DOF control structure with reference model and feedforward:



- $H_m$  – model that describes the desired setpoint response
- $H_{ff}$  – feedforward generator that makes  $y$  follow  $y_m$ 
  - Goal: perfect following if there are no disturbances or model errors

# Reference Generation – Input–Output Approach

The pulse transfer function from  $u_c$  to  $y$  is

$$H = \frac{H_p(H_{ff} + H_c H_m)}{1 + H_p H_c}$$

Choose

$$H_{ff} = \frac{H_m}{H_p}$$

Then

$$H = \frac{H_p\left(\frac{H_m}{H_p} + H_c H_m\right)}{1 + H_p H_c} = H_m$$

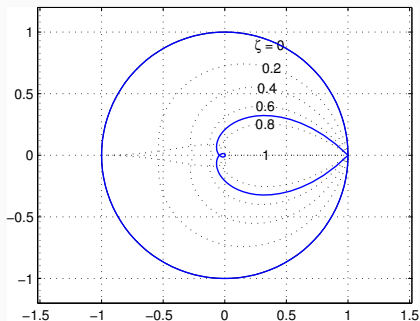
Perfect model following!

# Restrictions on the Model

In order for  $H_{ff} = \frac{H_m}{H_p}$  to be implementable (causal and stable),

- $H_m$  must have at least the same pole excess as  $H_p$
- any zeros of  $H_p$  outside unit circle must also be included in  $H_m$

In practice, also poorly damped zeros of  $H_p$  (e.g., outside the heart-shaped region below) should be included in  $H_m$



## Example: PID Control of the Double Tank

Process:

$$G_p(s) = \frac{3}{(1 + 60s)^2}$$

ZOH-sampled process ( $h = 3$ ):

$$H_p(z) = \frac{0.003627(z + 0.9672)}{(z - 0.9512)^2}$$

PID controller tuned for good regulation performance:

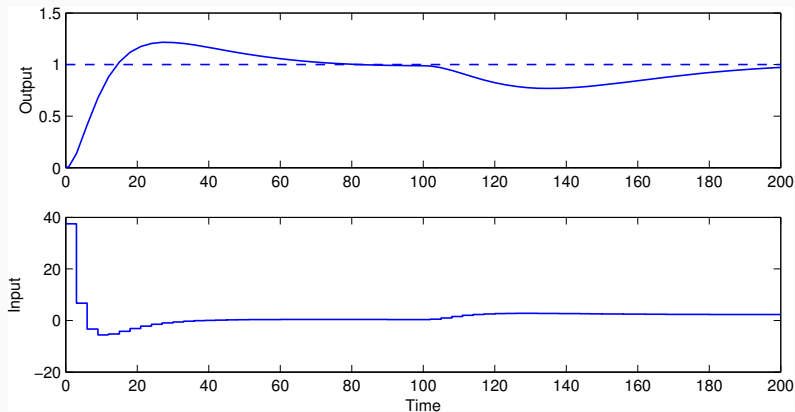
$$G_c(s) = K \left( 1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d/N} \right)$$

with  $K = 7$ ,  $T_i = 45$ ,  $T_d = 15$ ,  $N = 10$ , discretized using FOH



# Example: PID Control of the Double Tank

Simulation with simple error feedback:



- Load disturbance at time 100 regulated as desired
- Too large control signal at time 0 and overshoot in the step response

## Example: PID Control of the Double Tank

Reference model (critically damped – should not generate any overshoot):

$$G_m(s) = \frac{1}{(1 + 10s)^2}$$

Sampled reference model:

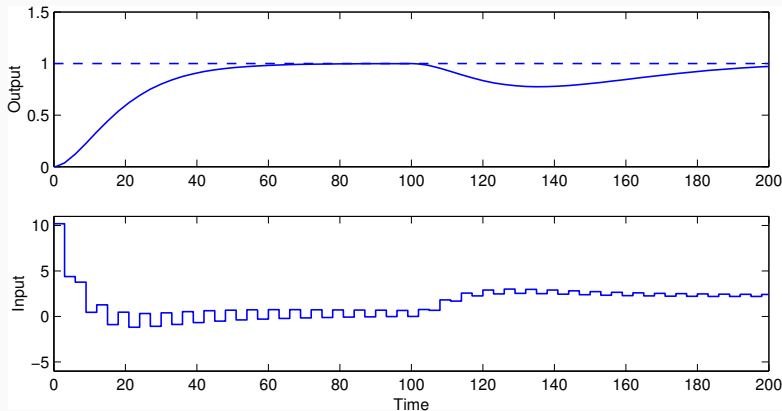
$$H_m(z) = \frac{0.036936(z + 0.8187)}{(z - 0.7408)^2}$$

Feedforward filter:

$$H_{ff}(z) = \frac{H_m(z)}{H_p(z)} = \frac{10.1828(z + 0.8187)(z - 0.9512)^2}{(z - 0.7408)^2(z + 0.9672)}$$

# Example: PID Control of the Double Tank

Simulation with reference model and feedforward:



- Perfect step response according to the model
- Unpleasant ringing in the control signal
  - due to cancellation of poorly damped process zero

## Example: PID Control of the Double Tank

Modified reference model that includes the process zero:

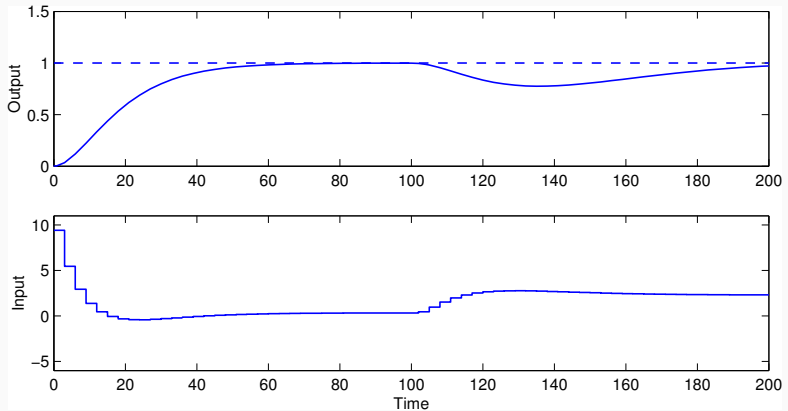
$$H_m(z) = \frac{0.034147(z + 0.9672)}{(z - 0.7408)^2}$$

New feedforward filter:

$$H_{ff}(z) = \frac{H_m(z)}{H_p(z)} = \frac{9.414(z - 0.9512)^2}{(z - 0.7408)^2}$$

# Example: PID Control of the Double Tank

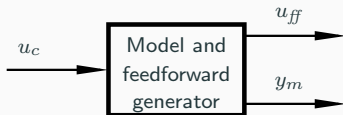
Simulation with modified reference model:



- Very similar step response
- Ringing in control signal eliminated

## Remark

In the implementation, both  $u_{ff}$  and  $y_m$  can be generated by a single dynamical system:



Matlab:

```
>> Hp = ...           % define process  
>> Hm = ...           % define reference model  
>> refgen = [Hm/Hp; Hm] % concatenate systems  
>> minreal(ss(refgen)) % make minimal state-space realization
```

# Simplistic Setpoint Handling in State Space

Replace  $u(k) = -Lx(k)$  with

$$u(k) = L_c u_c(k) - Lx(k)$$

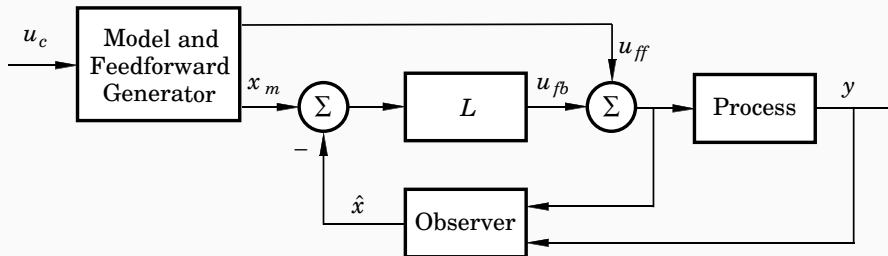
The pulse transfer function from  $u_c(k)$  to  $y(k)$  is

$$H_{yu_c}(z) = C(zI - \Phi + \Gamma L)^{-1} \Gamma L_c = L_c \frac{B(z)}{A_m(z)}$$

In order to have unit static gain ( $H_{yu_c}(1) = 1$ ),  $L_c$  should be chosen as

$$L_c = \frac{1}{C(I - \Phi + \Gamma L)^{-1} \Gamma}$$

## Reference Generation – State Space Approach



The model should generate a reference trajectory  $x_m$  for the process state  $x$  (one reference signal per state variable)

The feedforward signal  $u_{ff}$  should make  $x$  follow  $x_m$

- Goal: perfect following if there are no disturbances or model errors



# Reference Generation – State Space Approach

Linear reference model:

$$x_m(k+1) = \Phi_m x_m(k) + \Gamma_m u_c(k)$$

Control law:

$$u(k) = L\left(x_m(k) - \hat{x}(k)\right) + u_{ff}(k)$$

- How to generate model states  $x_m$  that are compatible with the real states  $x$ ?
- How to generate the feedforward control  $u_{ff}$ ?

# Design of the Reference Model

Start by choosing the reference model identical to the process model, i.e.,

$$x_m(k+1) = \Phi x_m(k) + \Gamma u_{ff}(k)$$

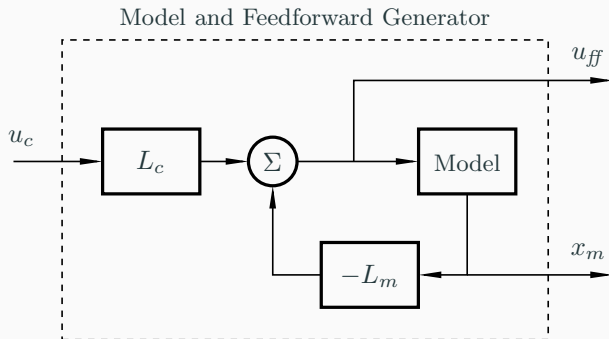
Then modify the dynamics of the reference model as desired using state feedback (“within the model”)

$$u_{ff}(k) = L_c u_c(k) - L_m x_m(k)$$

Gives the reference model dynamics

$$x_m(k+1) = \underbrace{(\Phi - \Gamma L_m)}_{\Phi_m} x_m(k) + \underbrace{\Gamma L_c}_{\Gamma_m} u_c(k)$$

# Design of the Reference Model



# Design of the Reference Model

Design choices:

- $L_m$  is chosen to give the model the desired eigenvalues (poles)
- $L_c$  is chosen to give the desired static gain (usually 1)

**Remark:** The reference model will have the same zeros as the process, so there is no risk of cancelling poorly damped or unstable zeros

Additional zeros and poles can be added by extending the model

# Complete State-Space Controller

The complete controller, including state feedback, observer, and reference generator is given by

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K(y(k) - C\hat{x}(k)) \quad (\text{Observer})$$

$$x_m(k+1) = \Phi x_m(k) + \Gamma u_{ff}(k) \quad (\text{Reference model})$$

$$u(k) = L(x_m(k) - \hat{x}(k)) + u_{ff}(k) \quad (\text{Control signal})$$

$$u_{ff}(k) = -L_m x_m(k) + L_c u_c(k) \quad (\text{Feedforward})$$

# Pseudo-Code Structure

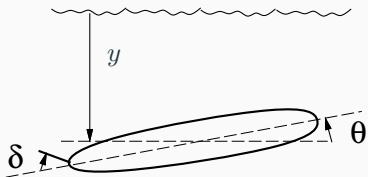
Observer with one sample delay

```
1  y = ReadInput();
2  u_c = getCommandSignal();
3  u_ff = -Lm*x_m + Lc*u_c;
4  u = L*(x_m - x_hat) + u_ff;
5  WriteOutput(u);
6  x_m = Phi*x_m + Gamma*u_ff;
7  x_hat = Phi*x_hat + Gamma*u + K*(y - c*x_hat);
```

The computational delay can be further minimized

```
1  y = ReadInput();
2  u_c = getCommandSignal();
3  u_ff = u_ff_temp + Lc*u_c;
4  u = u_temp + u_ff;
5  WriteOutput(u);
6  x_m = Phi*x_m + Gamma*u_ff;
7  x_hat = Phi*x_hat + Gamma*u + K*(y - c*x_hat);
8  u_ff_temp = -Lm*x_m;
9  u_temp = L*(x_m - x_hat);
```

## Design Example: Depth Control of Torpedo



State vector:

$$x = \begin{pmatrix} q \\ \theta \\ y \end{pmatrix} = \begin{pmatrix} \text{pitch angular velocity} \\ \text{pitch angle} \\ \text{depth} \end{pmatrix}$$

Input signal:

$$u = \delta = \text{rudder angle}$$

# Torpedo: Continuous-Time Model

Simple model:

$$\frac{dq}{dt} = aq + b\delta$$

$$\frac{d\theta}{dt} = q$$

$$\frac{dy}{dt} = -V\theta \quad (+ c\delta)$$

where  $a = -2$ ,  $b = -1.3$ , and  $V = 5$  (speed of torpedo)

$$\dot{x} = \begin{pmatrix} a & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -V & 0 \end{pmatrix} x + \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix} u$$

$$y = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} x$$



# Torpedo: Sampled Model

Sample with  $h = 0.2$

$$x(k+1) = \begin{pmatrix} 0.67 & 0 & 0 \\ 0.165 & 1 & 0 \\ -0.088 & -1 & 1 \end{pmatrix} x(k) + \begin{pmatrix} -0.214 \\ -0.023 \\ 0.008 \end{pmatrix} u(k)$$

Matlab:

```
>> A = [a 0 0; 1 0 0; 0 -V 0];  
>> B = [b; 0; 0];  
>> C = [0 0 1];  
>> Gp = ss(A,B,C,0);  
>> h = 0.2;  
>> Hp = c2d(Gp,h);  
>> [Phi,Gamma] = ssdata(Hp);
```

# Torpedo: State Feedback Design

- $u(k) = -Lx(k)$
- rejection of (impulse) load disturbances

Desired continuous-time dynamic behaviour:

- two complex-conjugated poles with relative damping 0.5 and natural frequency  $\omega_c$
- one pole in  $-\omega_c$
- a single parameter decides the dynamics

Desired characteristic polynomial

$$(s^2 + 2 \cdot 0.5 \cdot \omega_c s + \omega_c^2)(s + \omega_c) = s^3 + 2\omega_c s^2 + 2\omega_c^2 s + \omega_c^3$$

Each pole translated into discrete time as  $z_i = e^{s_i h}$

# Torpedo: State Feedback Design in Matlab

Matlab:

```
>> wc = 1; % speed of state feedback
>> pc = wc*roots([1 2 2 1]); % control poles in cont time
>> pcd = exp(pc*h); % control poles in disc time
>> L = place(Phi, Gam, pcd)
L =
    -0.145    -1.605     0.153
```

# Torpedo: Observer Design

- $\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K(y(k) - C\hat{x}(k))$
- state estimation + measurement noise rejection

Observer Dynamics:

- the same pole layout as in the state feedback design
- parametrized by  $\omega_o$  instead of  $\omega_c$
- typically faster dynamics than the state feedback, e.g.,  $\omega_o = 2\omega_c$

Desired continuous-time characteristic polynomial:

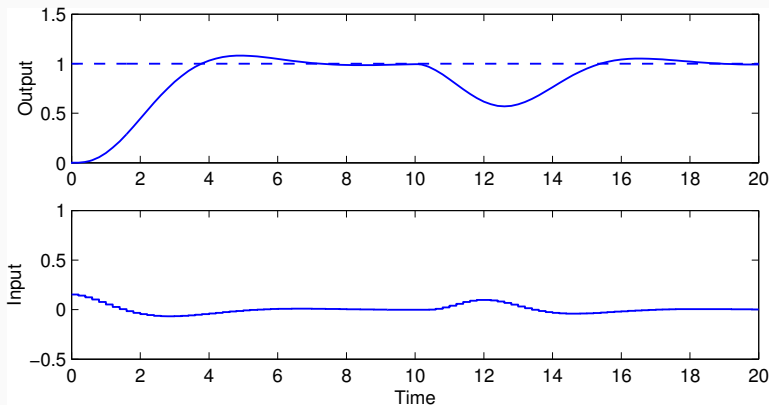
$$(s^2 + \omega_o s + \omega_o^2)(s + \omega_o) = s^3 + 2\omega_o s^2 + 2\omega_o^2 s + \omega_o^3$$

## Torpedo: Observer Design in Matlab

```
>> wo = 2; % speed of observer
>> po = wo*roots([1 2 2 1]); % observer poles in cont time
>> pod = exp(po*h); % observer poles in disc time
>> K = place(Phi',C',pod) '
K =
    0
-0.130
    0.460
```

## Torpedo: Simplistic Setpoint Handling

Simulation assuming simplistic approach,  $u(k) = -L\hat{x}(k) + L_c u_c(k)$ ,  
 $L_c = (C(I - \Phi + \Gamma L)^{-1}\Gamma)^{-1}$



- Step response slower than desired; overshoot in response

# Torpedo: Reference Model and Feedforward Design

Reference model:

$$x_m(k+1) = \Phi x_m(k) + \Gamma u_{ff}(k)$$

Feedforward:

$$u_{ff} = -L_m x_m + L_{cm} u_c$$

Desired characteristic polynomial:

$$(s + \omega_m)^3 = s^3 + 3\omega_m s^2 + 3\omega_m^2 s + \omega_m^3$$

(critically damped – important!)

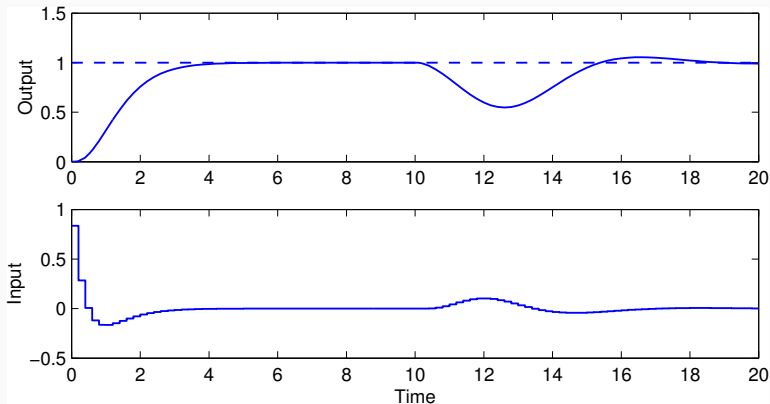
- Parametrized using  $\omega_m$
- Chosen as  $\omega_m = 2\omega_c$

## Torpedo: Reference Model and Feedforward in Matlab

```
>> wm = 2; % speed of model
>> pm = wm*roots([1 3 3 1]); % model poles in cont time
>> pmd = exp(pm*h); % model poles in disc time
>> Lm = place(Phi,Gam,pmd)
Lm =
    -2.327    -6.744     0.886
>> Hm = ss(Phi-Gam*Lm,Gam,C,0,h);
>> Lcm = 1/dcgain(Hm)
Lcm =
    0.836
```



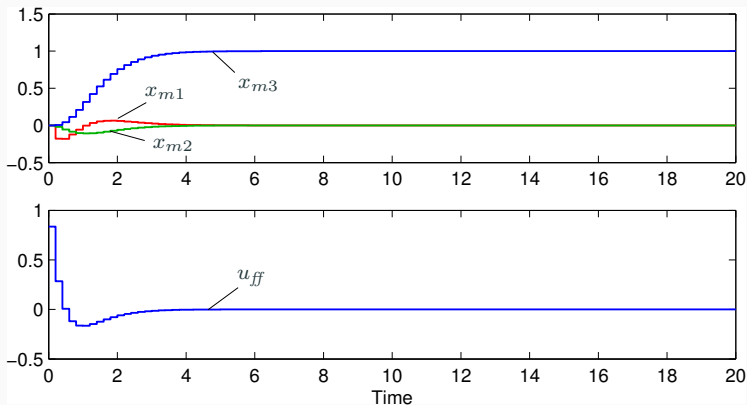
# Torpedo: Final Controller



- Faster step response without overshoot

# Torpedo: Final Controller

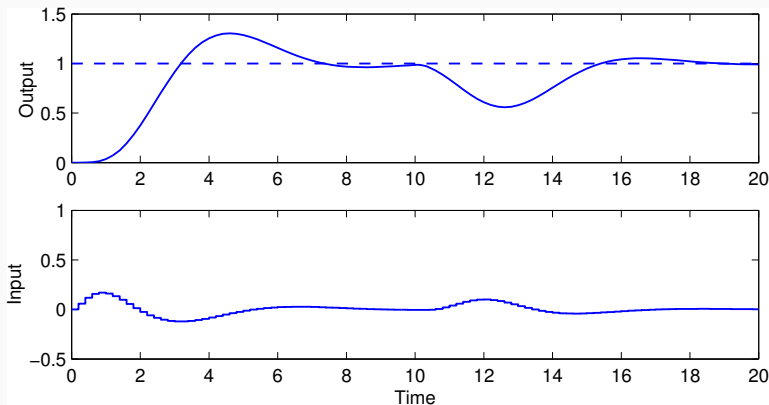
Model states and feedforward signal:



- The model states and the feedforward signal are not affected by the load disturbance
- Open loop

# Torpedo: Final Controller without Feedforward

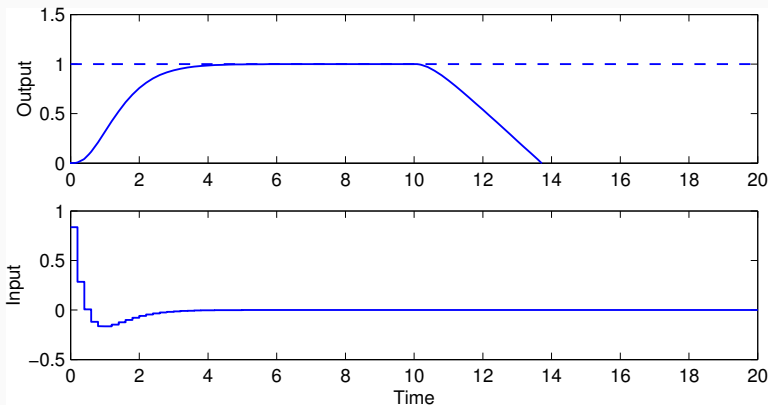
Simulation without the feedforward signal,  $u(k) = L(x_m(k) - \hat{x}(k))$ :



- Does not work very well – the feedforward term is needed to get the desired setpoint response

# Torpedo: Final Controller without Feedback

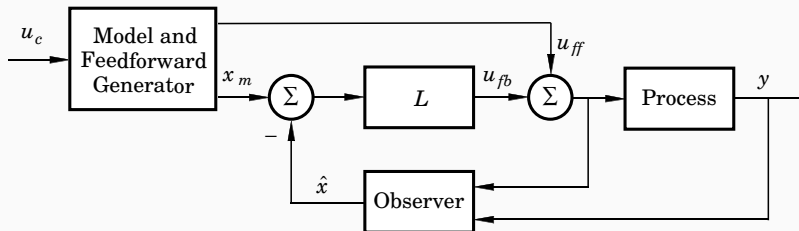
Simulation without the feedback signal,  $u(k) = u_{ff}(k)$ :



- Does not work – the feedback term is needed to stabilize the process and handle the load disturbance

# Nonlinear Reference Generation

Recall the state-space approach to reference generation:



$u_{ff}$  and  $x_m$  do not have to come from linear filters but could be the result of solving an optimization problem, e.g.:

- Move a satellite to a given altitude with minimum fuel
- Position a mechanical servo in as short time as possible under a torque constraint
- Move the ball on the beam as fast as possible without losing it

# General Solution for Linear Processes

Assume linear process

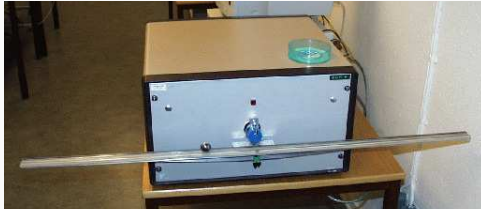
$$\frac{dx}{dt} = Ax + Bu$$

- Derive the feedforward (open-loop) control signal  $u_{ff}$  that solves the stated optimization problem
  - Course in Nonlinear Control (FRTN05, Lp 2)
- Generate the model state trajectories by solving

$$\frac{dx_m}{dt} = Ax_m + Bu_{ff}$$

Similar approach can be used for sampled systems

## Example: Time-Optimal Control of Ball on Beam



State vector:

$$x = \begin{pmatrix} z \\ v \\ \phi \end{pmatrix} = \begin{pmatrix} \text{ball position} \\ \text{ball velocity} \\ \text{beam angle} \end{pmatrix}$$

Continuous-time state-space model:

$$\frac{dz}{dt} = v$$

$$\frac{dv}{dt} = -k_v \phi \quad (k_v \approx 10)$$

$$\frac{d\phi}{dt} = k_\phi u \quad (k_\phi \approx 4.5)$$

## Example: Time-Optimal Control of Ball on Beam

**Optimization problem:** Assume steady state. Move the ball from start position  $z(0) = z_0$  to final position  $z(t_f) = z_f$  in minimum time while respecting the control signal constraints

$$-u_{\max} \leq u(t) \leq u_{\max}$$

Optimal control theory gives the optimal open-loop control law

$$u_{ff}(t) = \begin{cases} -u_0, & 0 \leq t < T \\ u_0, & T \leq t < 3T \\ -u_0, & 3T \leq t < 4T \end{cases}$$

where

$$u_0 = \operatorname{sgn}(z_f - z_0)u_{\max}$$

$$T = \sqrt[3]{\frac{|z_f - z_0|}{2k_\phi k_v u_{\max}}}$$

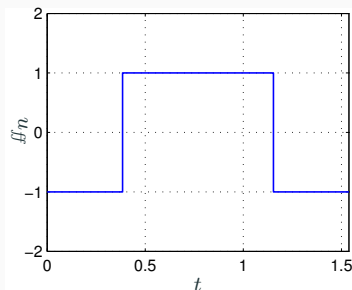
$$t_f = 4T$$



## Example: Time-Optimal Control of Ball on Beam

Assume  $u_{\max} = 1$ ,  $z_0 = 0$ , and  $z_f = 5 \Rightarrow t_f = 1.538$

Optimal control signal:



(“bang-bang” control)

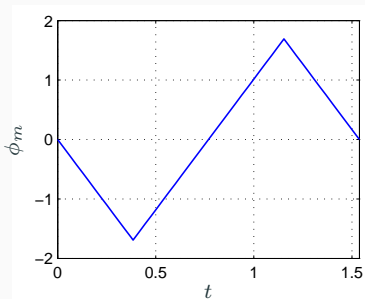
## Example: Time-Optimal Control of Ball on Beam

Solving

$$\frac{d\phi_m}{dt} = k_\phi u_{ff}$$

gives the optimal beam angle trajectory

$$\phi_m(t) = \begin{cases} -k_\phi u_0 t, & 0 \leq t < T \\ k_\phi u_0 (t-2T), & T \leq t < 3T \\ -k_\phi u_0 (t-4T), & 3T \leq t \leq 4T \end{cases}$$



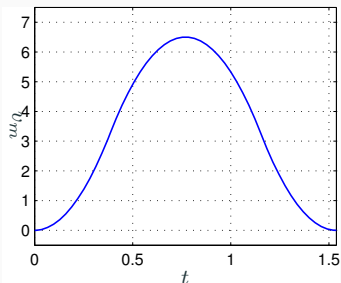
## Example: Time-Optimal Control of Ball on Beam

Solving

$$\frac{dv_m}{dt} = -k_v \phi_m$$

gives the optimal ball velocity trajectory

$$v_m(t) = \begin{cases} k_\phi k_v u_0 t^2/2, & 0 \leq t < T \\ -k_\phi k_v u_0 (t^2/2 - 2Tt + T^2), & T \leq t < 3T \\ k_\phi k_v u_0 (t^2/2 - 4Tt + 8T^2), & 3T \leq t \leq 4T \end{cases}$$



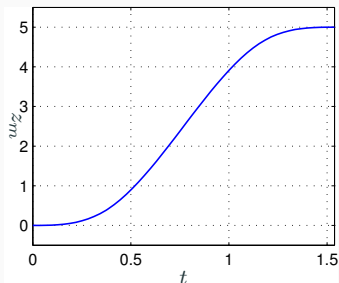
## Example: Time-Optimal Control of Ball on Beam

Finally, solving

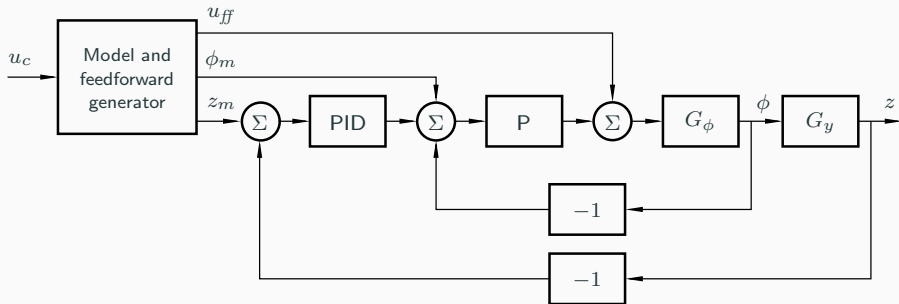
$$\frac{dz_m}{dt} = v_m$$

gives the optimal ball position trajectory

$$z_m(t) = \begin{cases} z_0 + k_\phi k_v u_0 \, t^3/6, & 0 \leq t < T \\ z_0 - k_\phi k_v u_0 (t^3/6 - Tt^2 + T^2t - T^3/3), & T \leq t < 3T \\ z_0 + k_\phi k_v u_0 (t^3/6 - 2Tt^2 + 8T^2t - 26T^3/3), & 3T \leq t \leq 4T \end{cases}$$



# Using the Time-Optimal Feedforward Generator in a Cascade Control Structure



- The PID controller should have derivative weighting  $\gamma = 1$

## Lectures 9 and 10: Summary

- Regulator problem – reduce impact of load disturbances and measurement noise
  - Feedforward from measurable disturbances
  - Input–output approach: design of feedback controller  $H_{fb}(z)$ , e.g. PID controller
  - State space approach: design of state feedback and observer, including disturbance estimator
- Servo problem – make the output follow the setpoint in the desired way
  - Input–output approach: design of reference model  $H_m(z)$  and feedforward filter  $H_{ff}(z)$
  - State space approach: design of combined reference and feedforward generator
    - Linear or nonlinear reference generation