Lecture 10: Reference Generation

[IFAC PB Ch 7, Ch 9 p. 68–72, these slides]

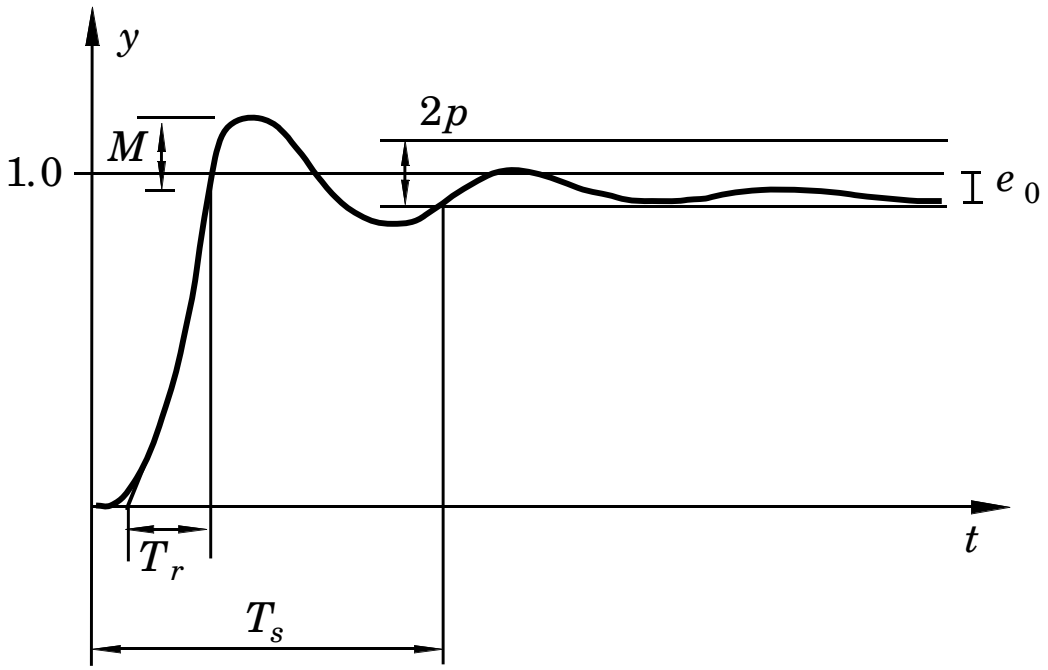
- Input–output approach
- State-space approach
- Nonlinear reference generation

The Servo Problem

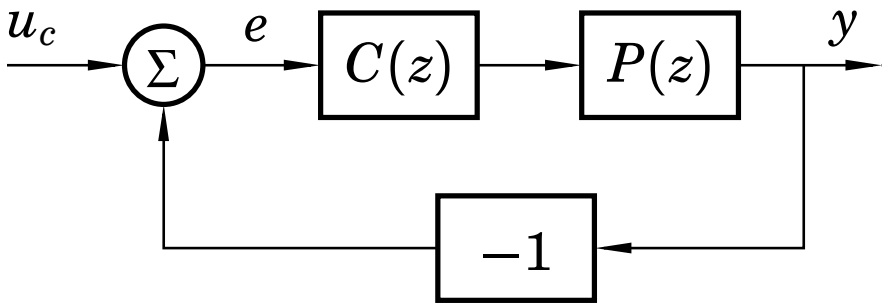
How to make the output respond to command signal changes in the desired way

Typical design criteria:

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



Simplistic Setpoint Handling – Error Feedback

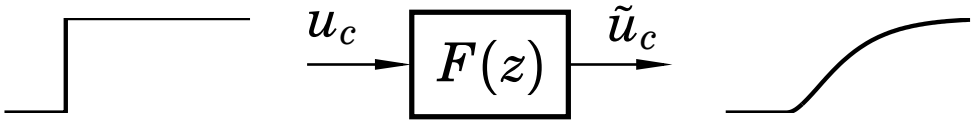


Problems:

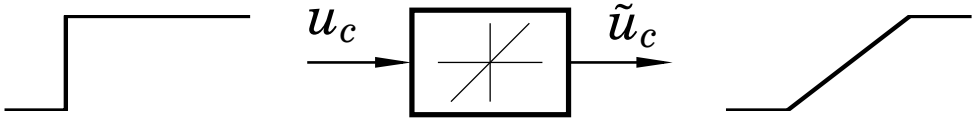
- Step changes in the setpoint can introduce very large control signals
- The same controller $C(z)$ must be tuned to handle both disturbances and setpoint changes
  - No separation between the regulator problem and the servo problem

Common Quick Fixes

- Filter the setpoint signal



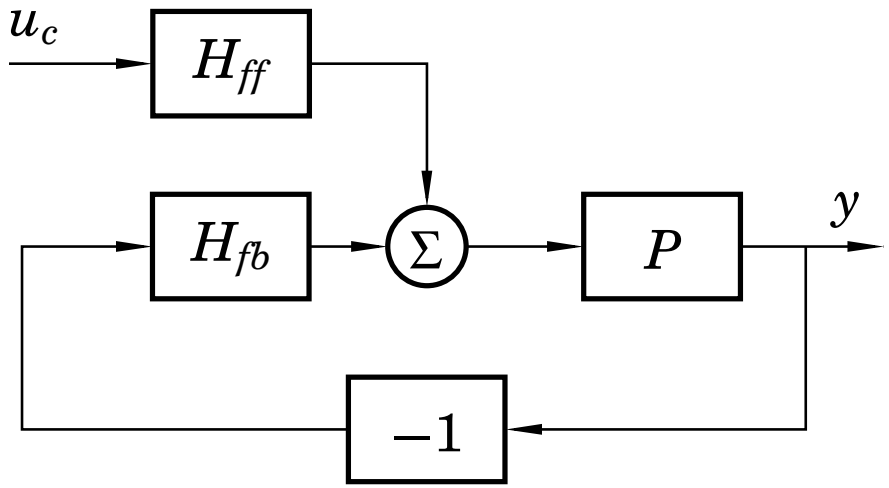
- Rate-limit the setpoint signal



- Introduce setpoint weighting in the controller
  - E.g. PID controller with setpoint weightings β and γ

A More General Solution

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



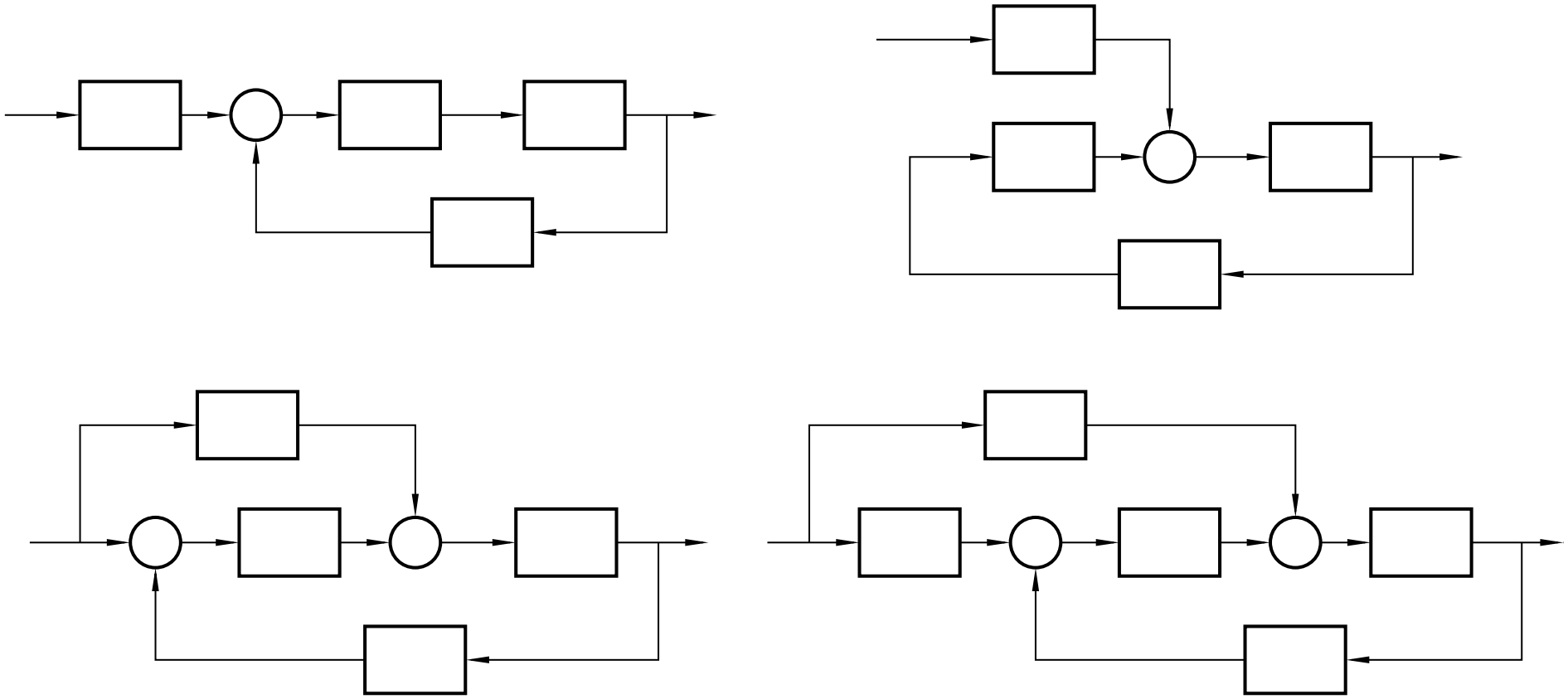
Design procedure:

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 ways, e.g.:



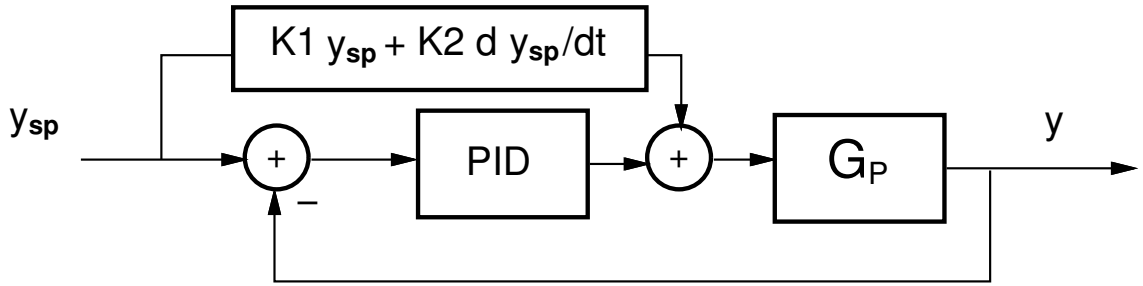
(For linear systems, all these structures are equivalent)

Example: PID with Setpoint Weighting

$$u = K\left(\beta y_{sp} - y + \frac{1}{T_I}\int (y_{sp} - y)d\tau + T_D\frac{d}{dt}(\gamma y_{sp} - y)\right)$$

$$= K\left(e + \frac{1}{T_I}\int e\,d\tau + T_D\frac{de}{dt}\right)$$

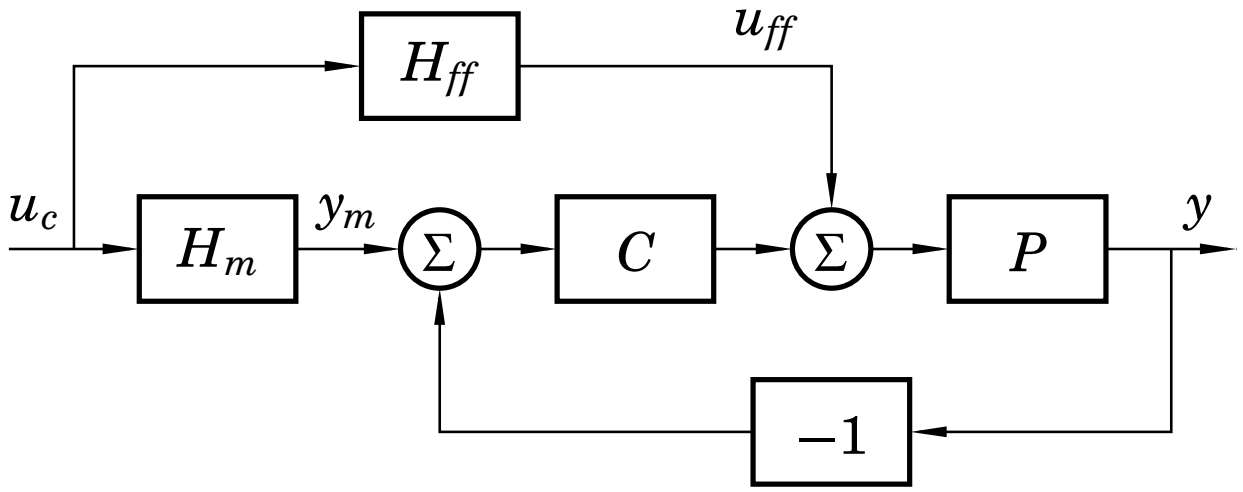
$$+ \underbrace{K(\beta - 1)}_{K_1} y_{sp} + \underbrace{T_D K(\gamma - 1)}_{K_2}\frac{dy_{sp}}{dt}$$



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 servo performance
- H_{ff} – feedforward generator that makes y follow y_m
  - Goal: perfect following if there are no disturbances or model errors

The transfer function from u_c to y is

$$H_{yu_c} = \frac{P(H_{ff} + CH_m)}{1 + PC}$$

Choose

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

Then

$$H_{yu_c} = \frac{P(\frac{H_m}{P} + CH_m)}{1 + PC} = H_m$$

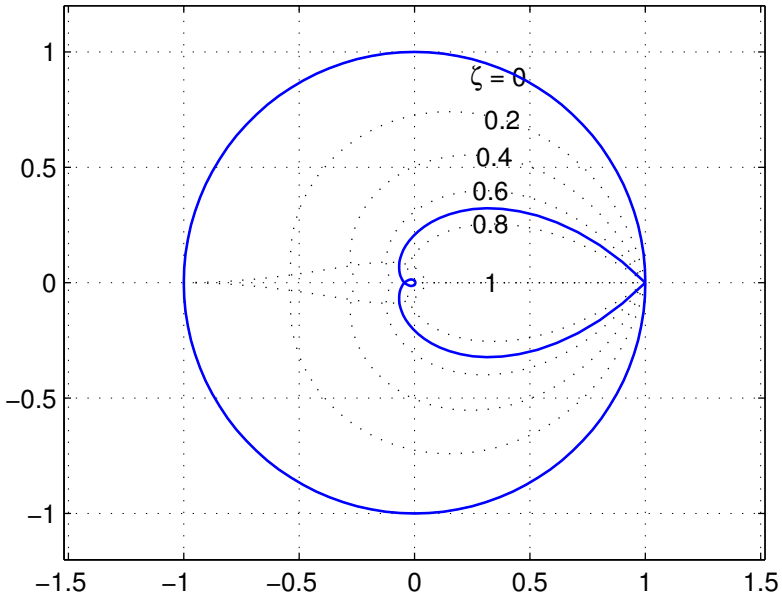
Perfect model following!

Restrictions on the Model

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

- H_m must have at least the same pole excess as P
- any unstable zeros of P must also be included in H_m

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



Example: PID Control of the Double Tank

Process:

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

Sampled process ($h = 3$):

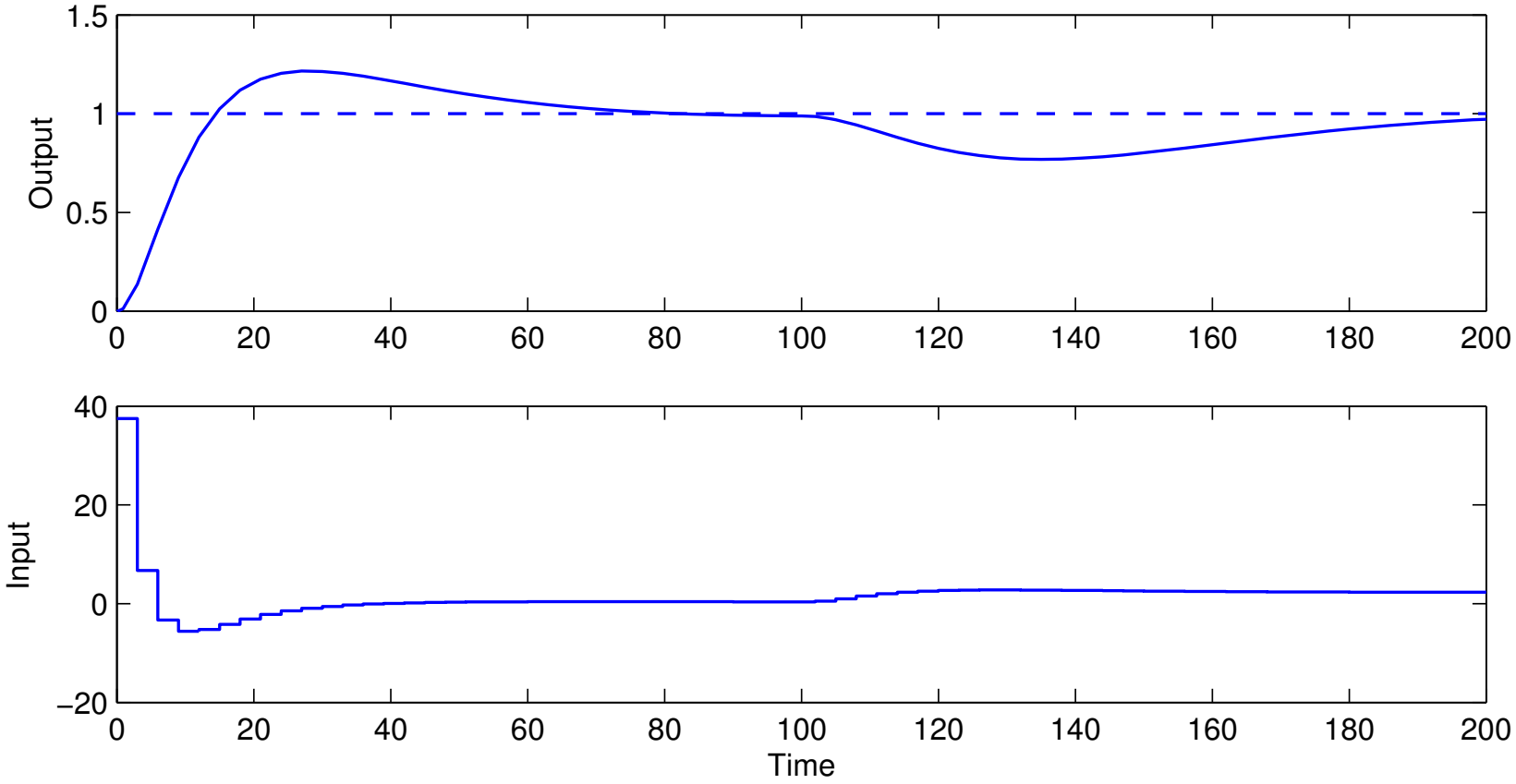
$$H(z) = \frac{0.003627(z+0.9672)}{(z-0.9512)^2}$$

PID controller tuned for good regulation performance:

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

with $K = 7$, $T_i = 45$, $T_d = 15$, $N = 10$, discretized using first-order hold.

Simulation with error feedback:



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

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

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

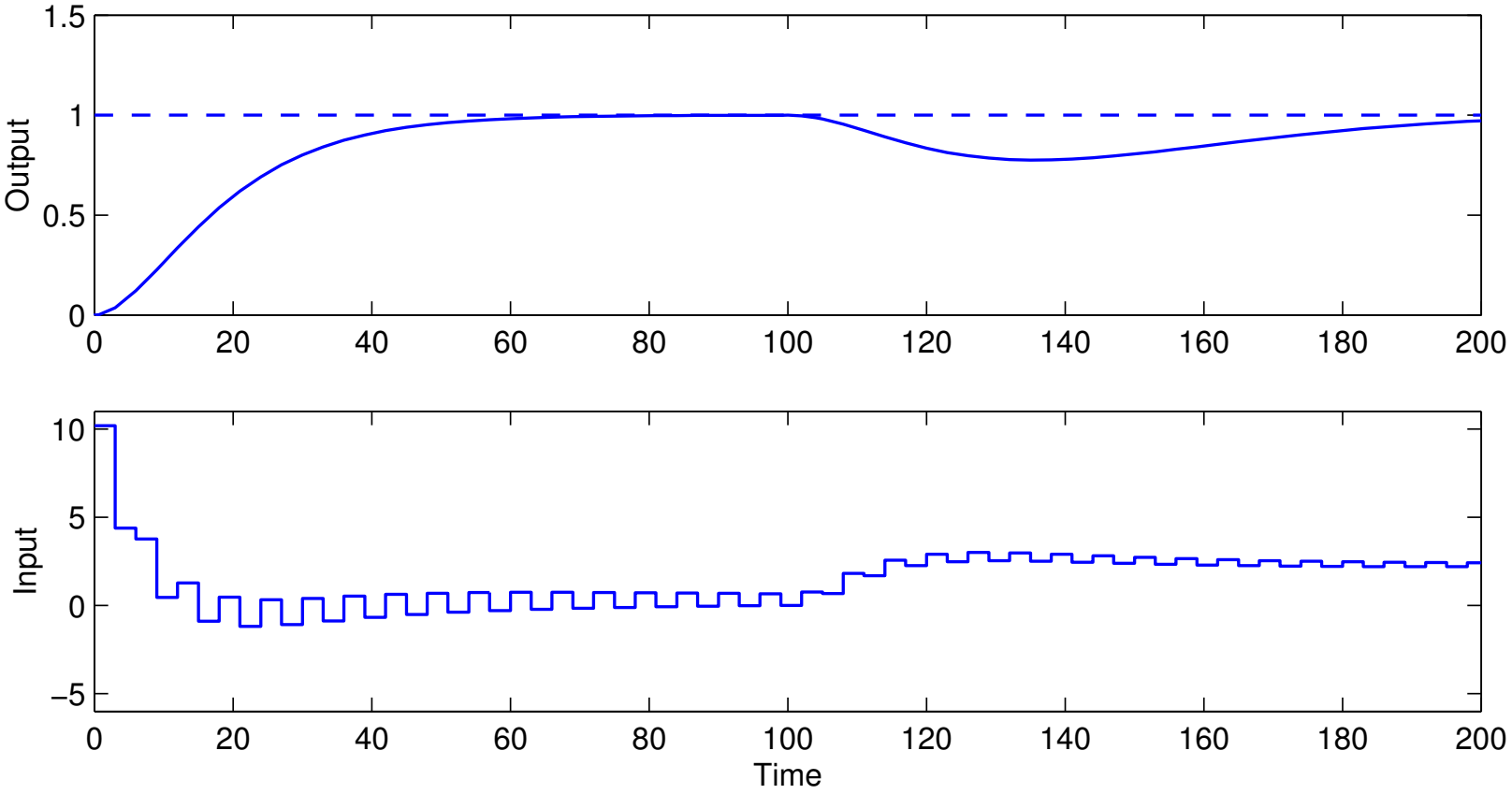
Sampled reference model:

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

Feedforward filter:

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

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; note that $u_{ff}(k) = \frac{H_m(q)}{H(q)} u_c(k)$

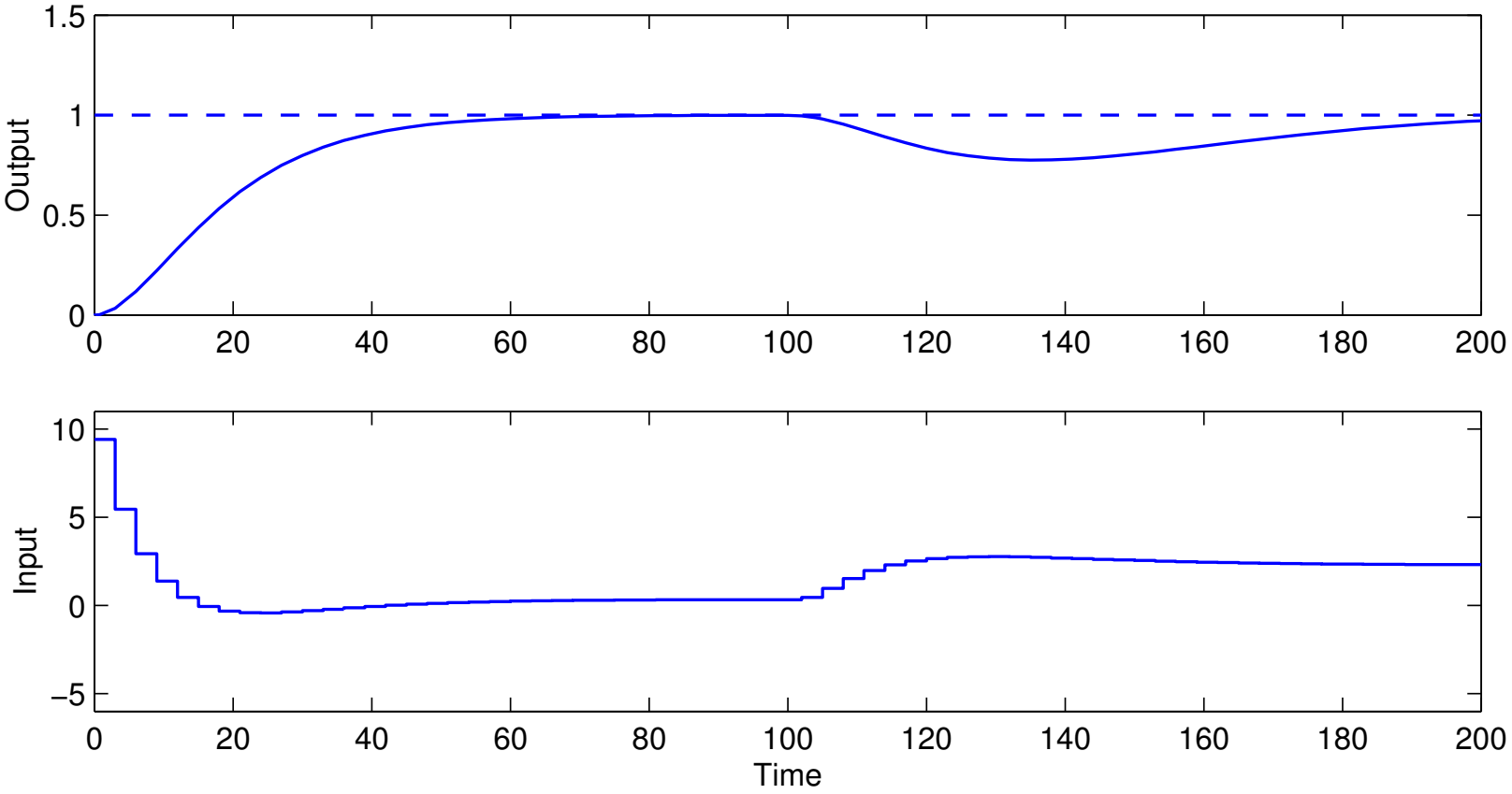
Modified reference model that includes the process zero:

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

New feedforward filer:

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

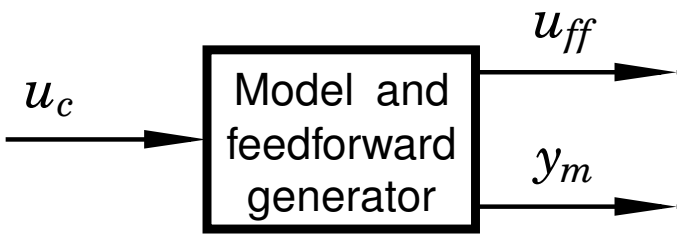
Simulation with modified reference model:



- Ringing eliminated

Remark

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



Matlab:

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

Simplistic Setpoint Handling in State Space

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

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

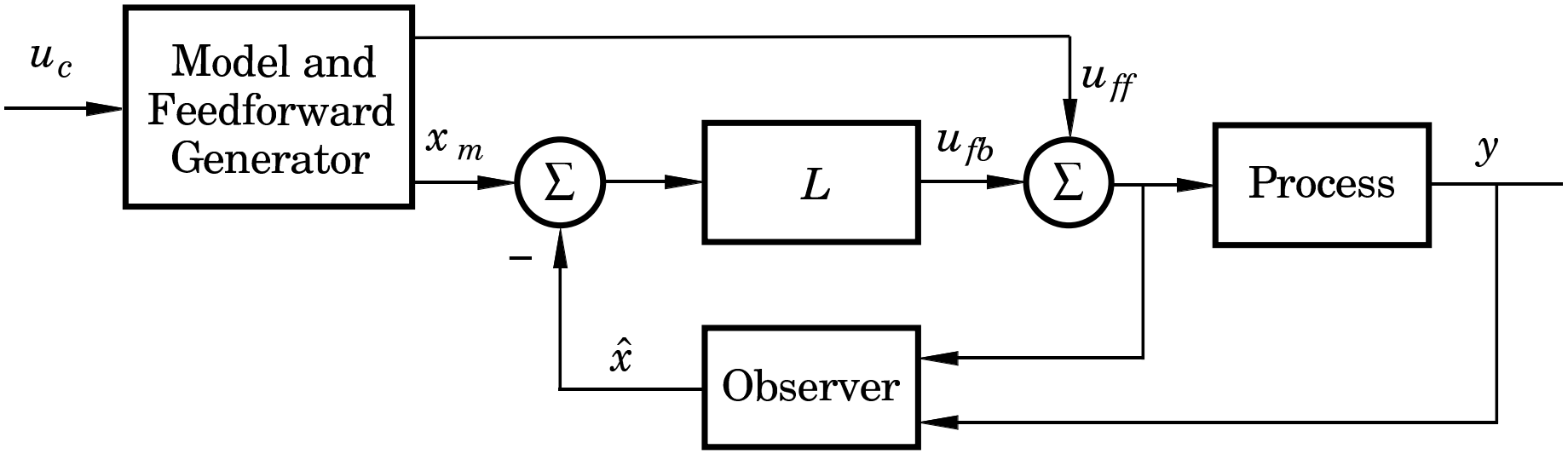
The pulse transfer function from $u_c(k)$ to $y(k)$ is

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

In order to have unit static gain ($H(1) = 1$), 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

Linear reference model:

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

Control law:

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

- 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.,

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

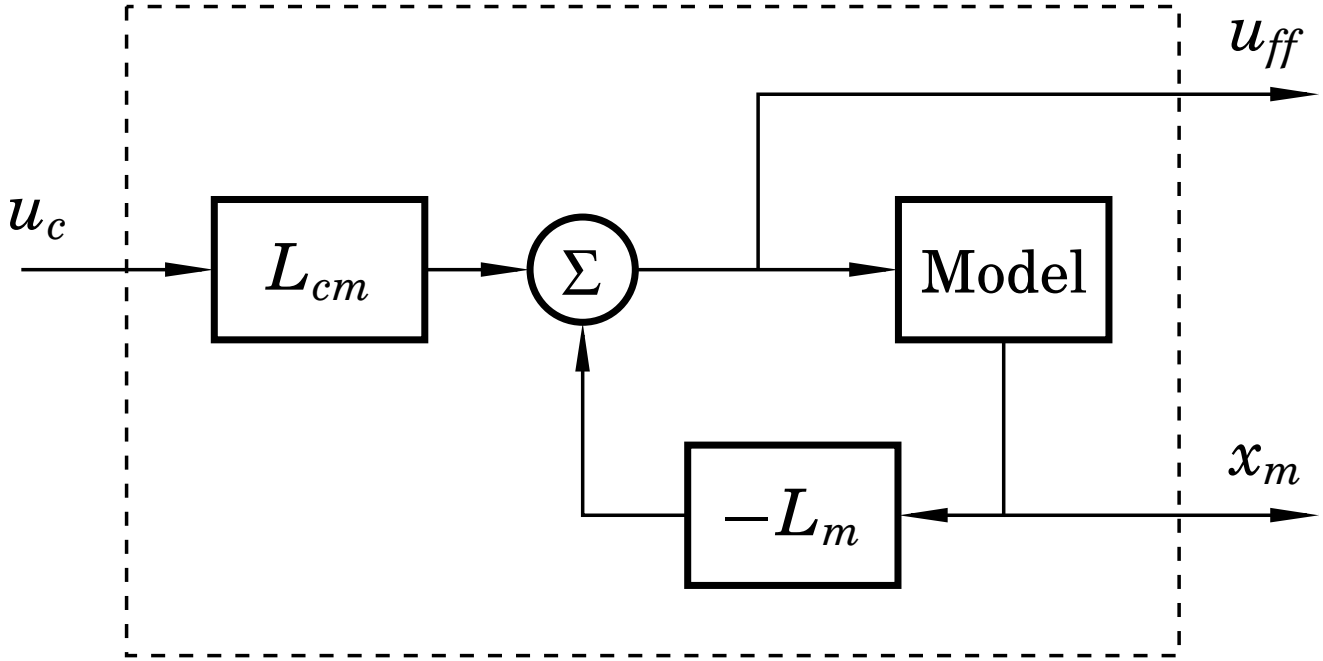
Then modify the dynamics of the reference model as desired using state feedback ("within the model")

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

Gives the reference model dynamics

$$x_m(k+1) = (\Phi - \Gamma L_m) x_m(k) + \Gamma L_{cm} u_c(k)$$
$$y_m(k) = C x_m(k)$$

Model and Feedforward Generator



Design of the Reference Model

Design choices:

- L_m is chosen to give the model the desired eigenvalues
- L_{cm} is chosen to give the model the desired static gain (typically 1)

(The above model will have the same zeros as the plant. 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

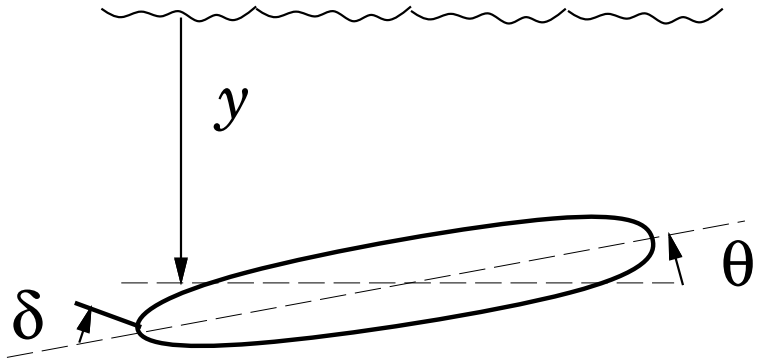
$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K(y(k) - C\hat{x}(k)) \qquad \text{(Observer)}$$

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

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

$$u_{ff}(k) = -L_m x_m(k) + L_{cm} u_c(k) \qquad \text{(Feedforward)}$$

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:

$$\frac{dq}{dt} = aq + b\delta$$
$$\frac{d\theta}{dt} = q$$
$$\frac{dy}{dt} = -V\theta \ (+ \ c\delta)$$

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 $h = 0.2$

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

Torpedo: State Feedback

- $u(k) = -Lx(k)$
- load disturbance rejection

Desired continuous-time dynamic behaviour:

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

Desired characteristic polynomial

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

Corresponds in discrete time to

$$(z^2 + a_1 z + a_2)\,(z - e^{-\omega_c h})$$

where $a_1 = -2e^{-\zeta\omega_c h}\cos\left(\sqrt{1-\zeta^2}\,\omega_c h\right)$, $a_2 = e^{-2\zeta\omega_c h}$ with $\zeta = 0.5$

Torpedo: State Feedback in Matlab

Matlab code

```
>> h = 0.2;
>> 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

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

Observer Dynamics:

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

Desired continuous-time characteristic polynomial:

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

Discrete time characteristic polynomial given from previous slide

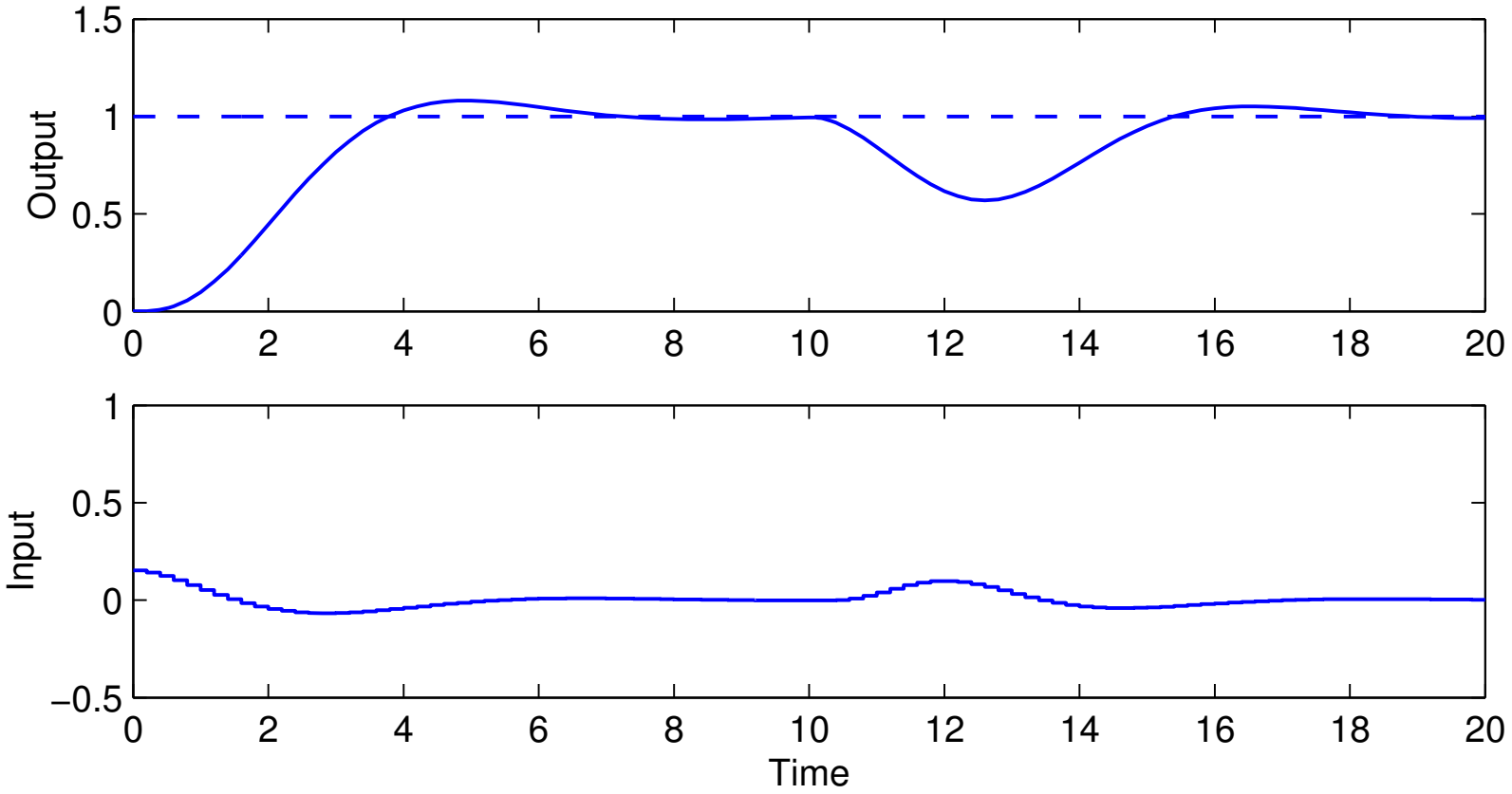
Torpedo: Observer Design in Matlab

```
>> 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, $u(k) = -L\hat{x}(k) + L_c u_c(k)$



- Slow step response with overshoot

Torpedo: Model and Feedforward Design

Reference model:

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

Feedforward:

$$u_{ff} = -L_m x_m + l_r r$$

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!)

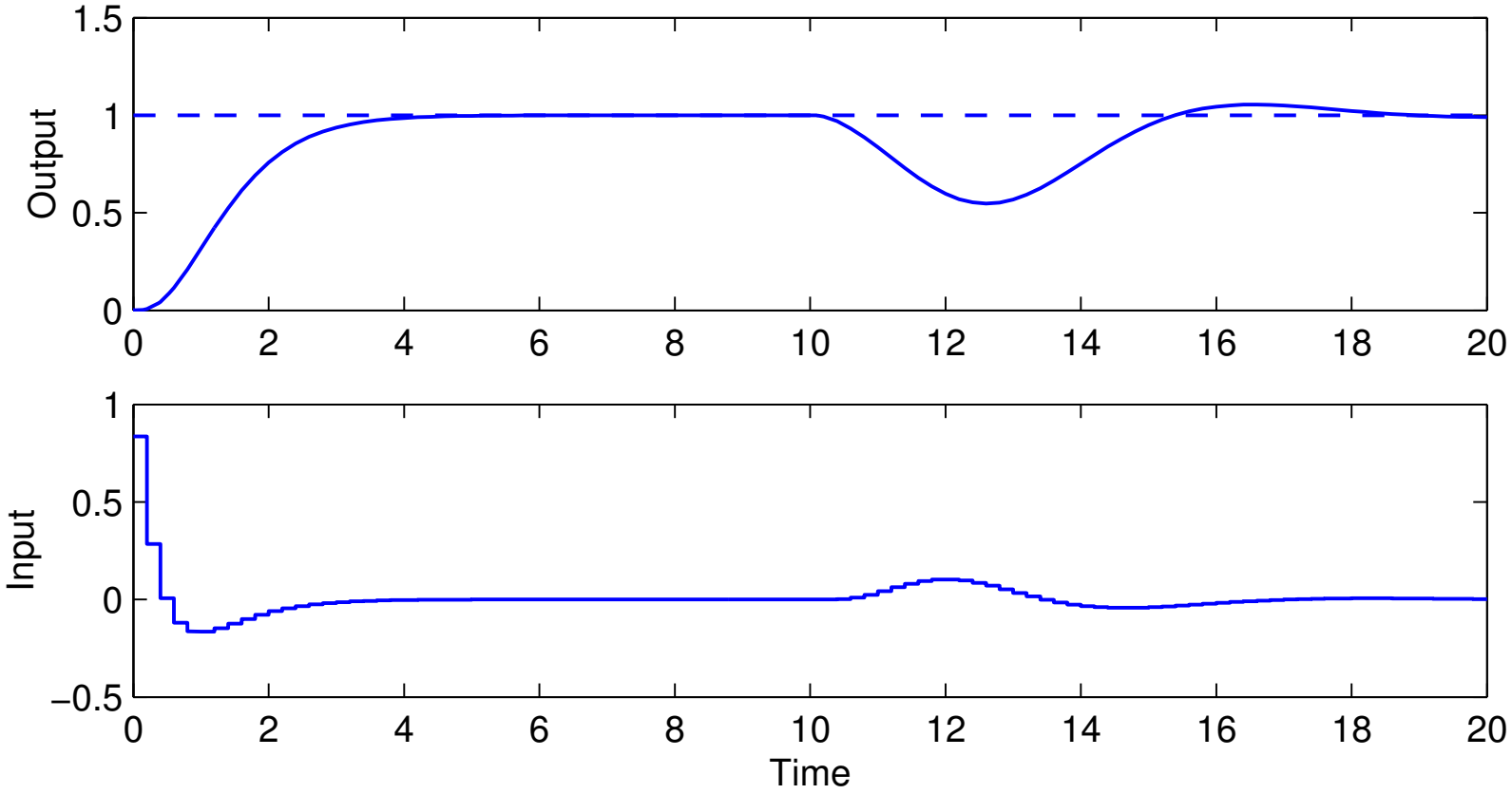
- Parametrized using ω_m
- Chosen as $\omega_m = 2\omega_c$

Torpedo: Model and Feedforward Design in Matlab

Corresponding discrete time characteristic polynomial from Matlab

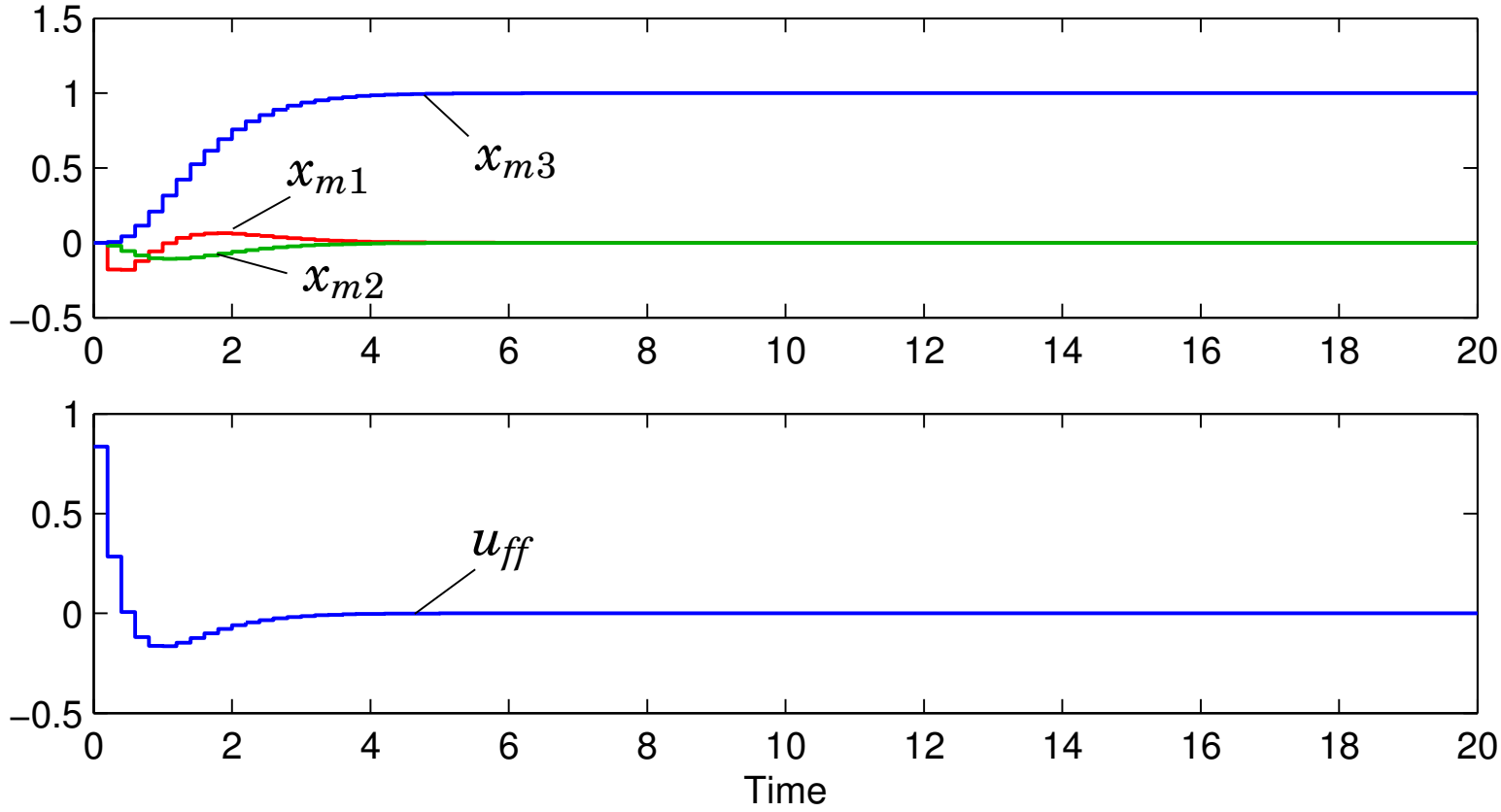
```
>> wm = 2;                          % speed of model
>> pm = wm*roots([1 3 3 1]);        % model poles in cont time
>> pmd = exp(pm*h);                 % model poles in disc time
>> Lm = place(Phi,Gam,pmd)
  Lm =
     -2.327   -6.744   0.886
>> Hm = ss(Phi-Gam*Lm,Gam,C,0,h);
>> lcm = 1/dcgain(Hm)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

Torpedo: Final Controller



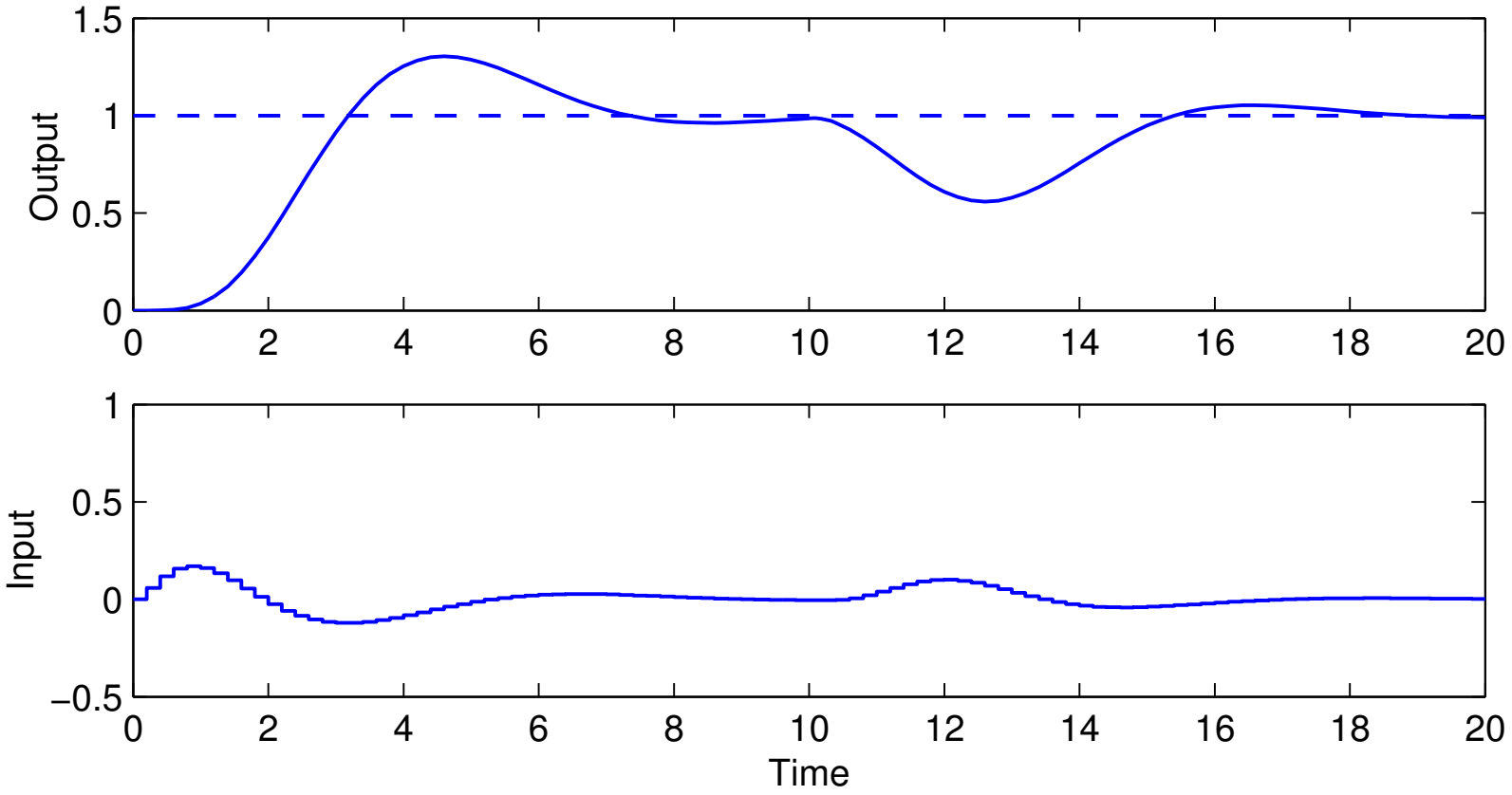
- Faster step response without overshoot

Model states and feedforward signal:



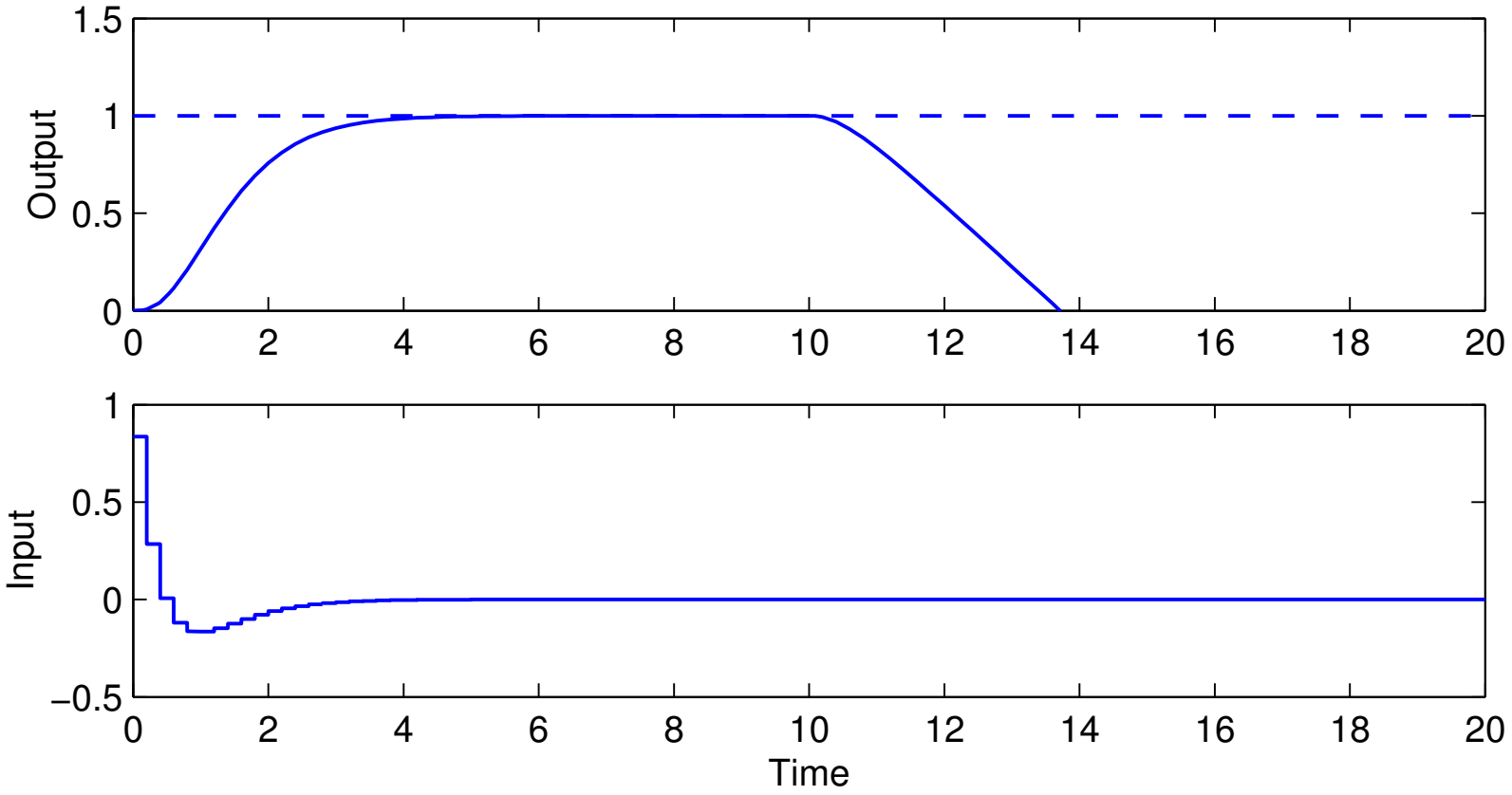
- The model states and the feedforward signal are not affected by the load disturbance
- Open loop

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 reference step response

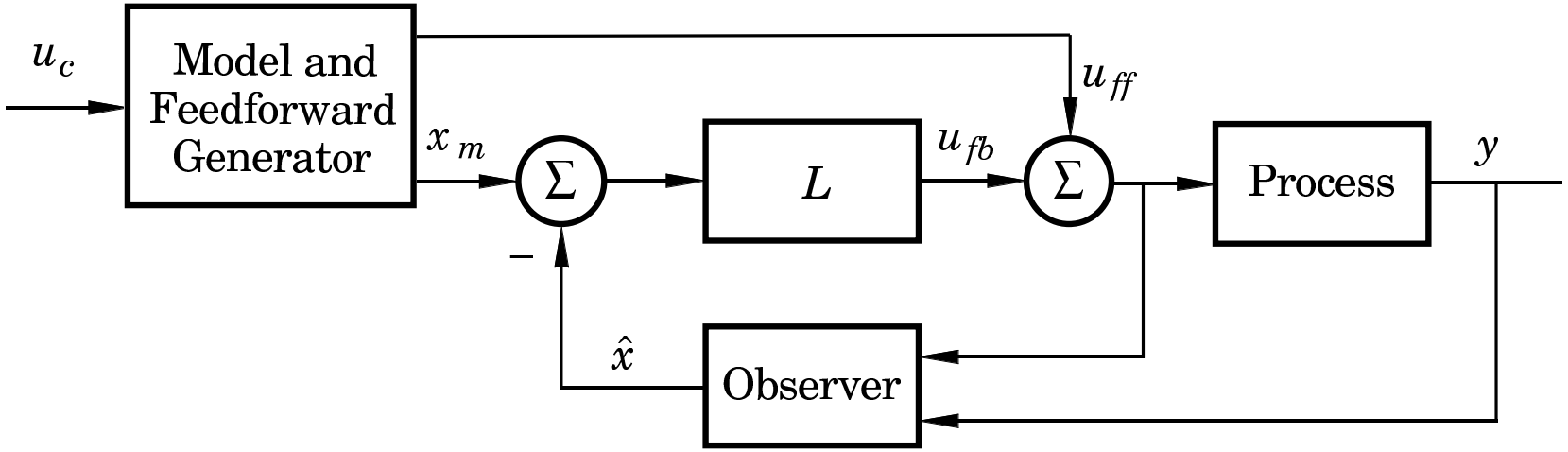
Simulation without the feedback term, $u(k) = u_{ff}(k)$:



- Does not work at all – the feedback is needed to handle the load disturbance

Nonlinear Reference Generation

Recall the state-space approach to reference generation:



Often, u_{ff} and x_m do not come from linear filters but are the result of solving an optimization problem, e.g.:

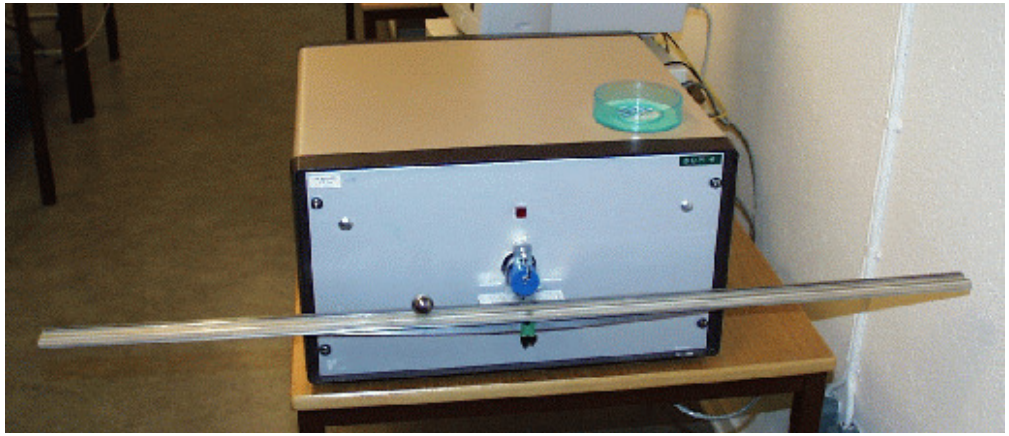
- Move a satellite to a given altitude with minimum fuel
- Position a mechanical servo in as short time as possible under a torque constraint
- Move the ball on the beam as fast as possible without losing it

General Solution

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

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

Example: Time-Optimal Control of Ball on Beam



State vector:

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

Continuous-time state-space model:

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -k_v \phi \qquad (k_v \approx 10) \\ \frac{d\phi}{dt} &= k_\phi u \qquad (k_\phi \approx 4.5) \end{aligned}$$

Optimization problem: Assume steady state. Move the ball from start position $x(0) = x_0$ to final position $x(t_f) = x_f$ 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

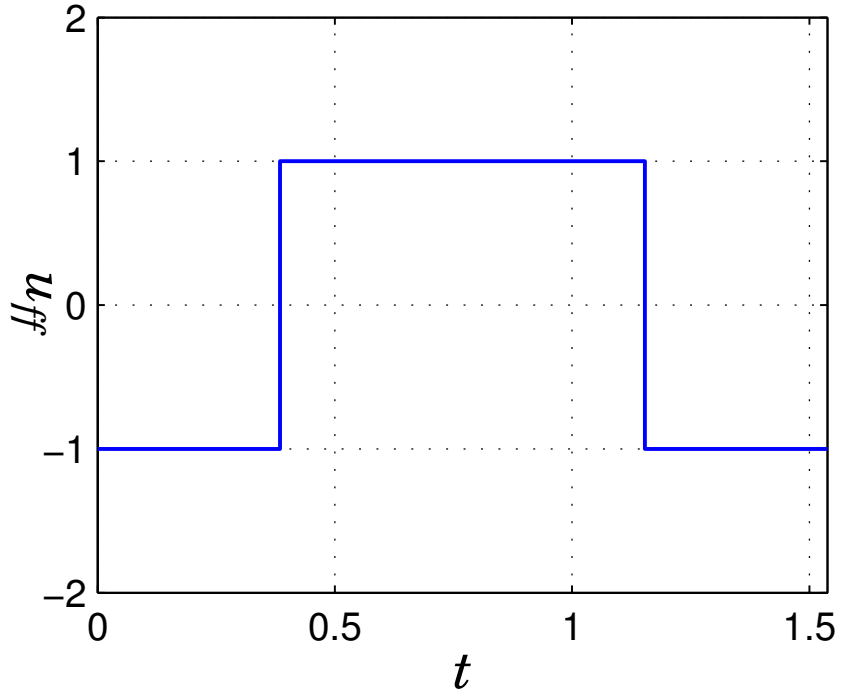
$$u_0 = \operatorname{sgn}(x_f - x_0) u_{\max}$$

$$T = \sqrt[3]{\frac{|x_f - x_0|}{2 k_\phi k_v u_{\max}}}$$

$$t_f = 4T$$

Example: $u_{\max} = 1$, $x_0 = 0$, and $x_f = 5 \;\Rightarrow\; t_f = 1.538$

Optimal control signal:



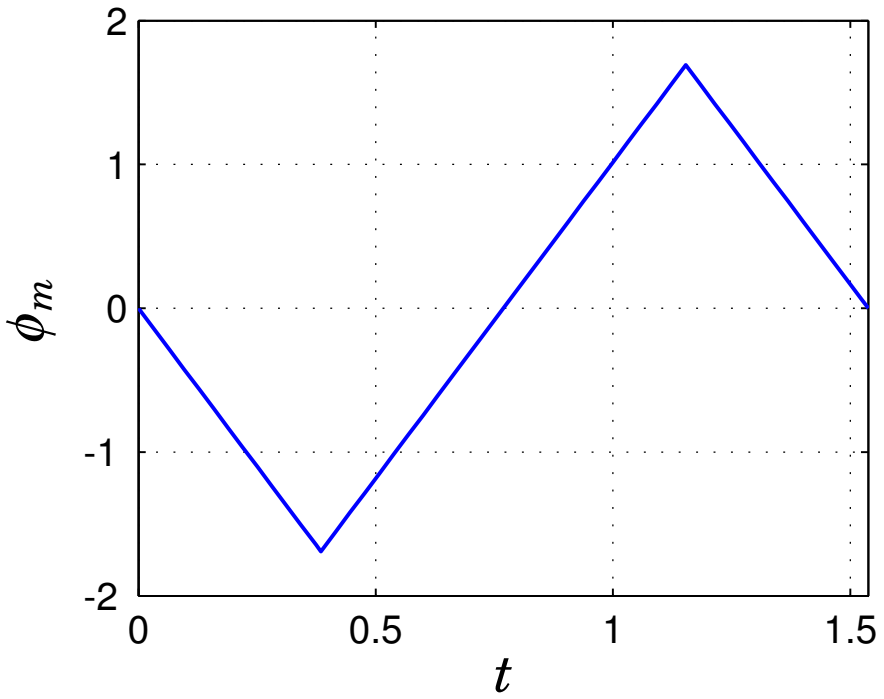
("bang-bang" control)

Solving

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

gives the optimal beam angle trajectory

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

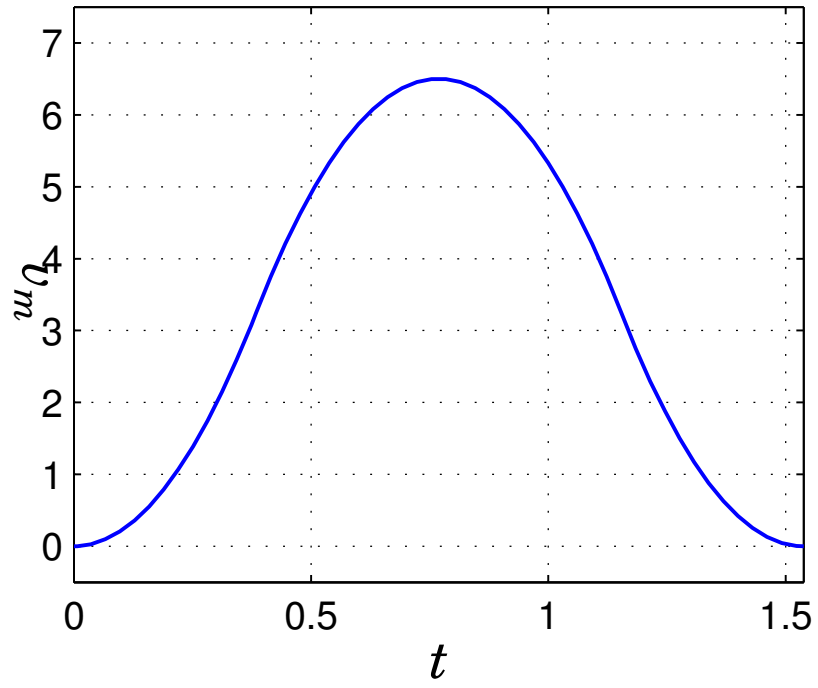


Solving

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

gives the optimal ball velocity trajectory

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

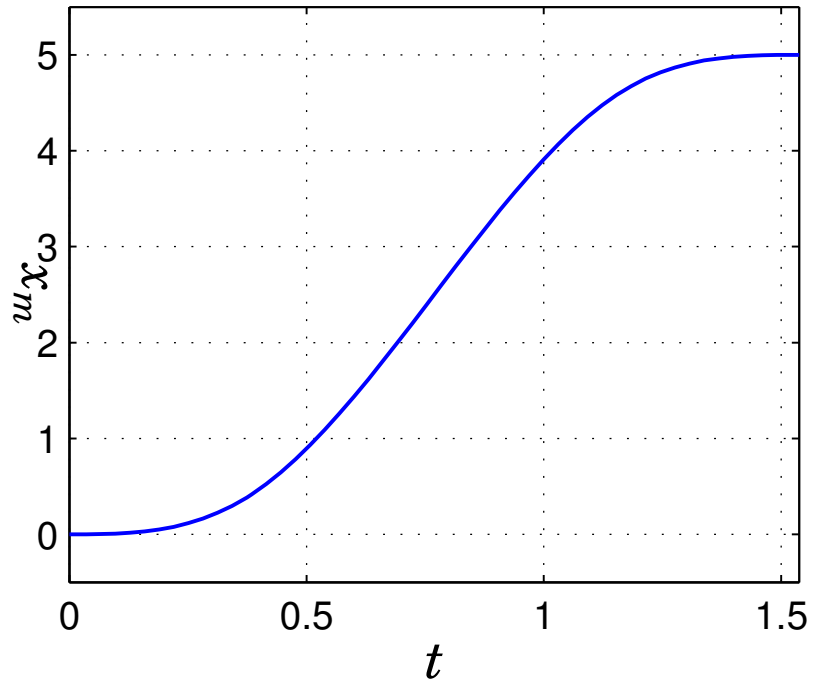


Finally, solving

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

gives the optimal ball position trajectory

$$x_m(t) = \begin{cases} x_0 + k_\phi k_v u_0 \; t^3/6, & 0 \le t < T \\ x_0 - k_\phi k_v u_0 \; (t^3/6 - Tt^2 + T^2 t - T^3/3), & T \le t < 3T \\ x_0 + k_\phi k_v u_0 \; (t^3/6 - 2Tt^2 + 8T^2 t - 26T^3/3), & 3T \le t \le 4T \end{cases}$$



Lectures 9 and 10: Summary

- Regulator problem – reduce impact of load disturbances and measurement noise
  - 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 model reference $H_m(z)$ and feedforward filter $H_{ff}(z)$
  - State space approach: design of combined reference and feedforward generator
    - Linear or nonlinear reference generation