4CM00: Control Engineering

FeedForward control

Dr.ir. Gert Witvoet

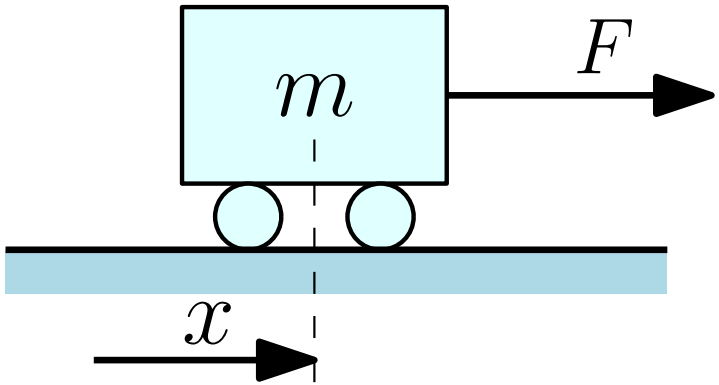
Technische Universiteit
Eindhoven
University of Technology

September 2020

Where innovation starts

Influence of the setpoint

Consider a simple motion system

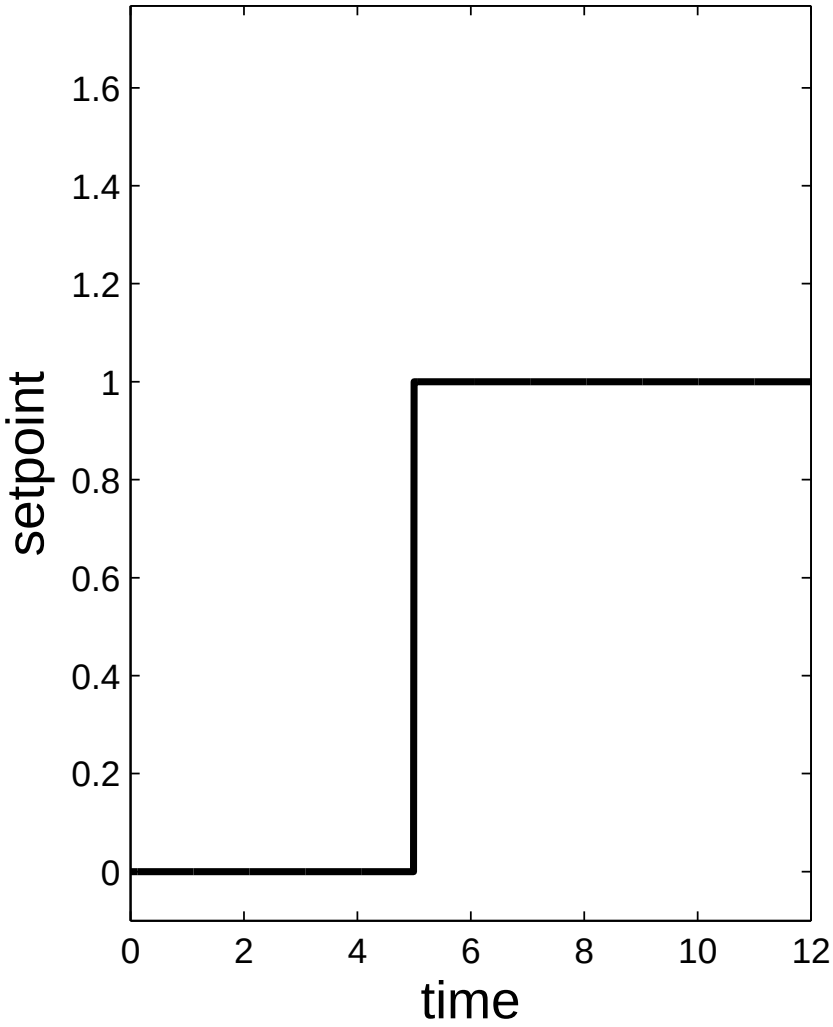


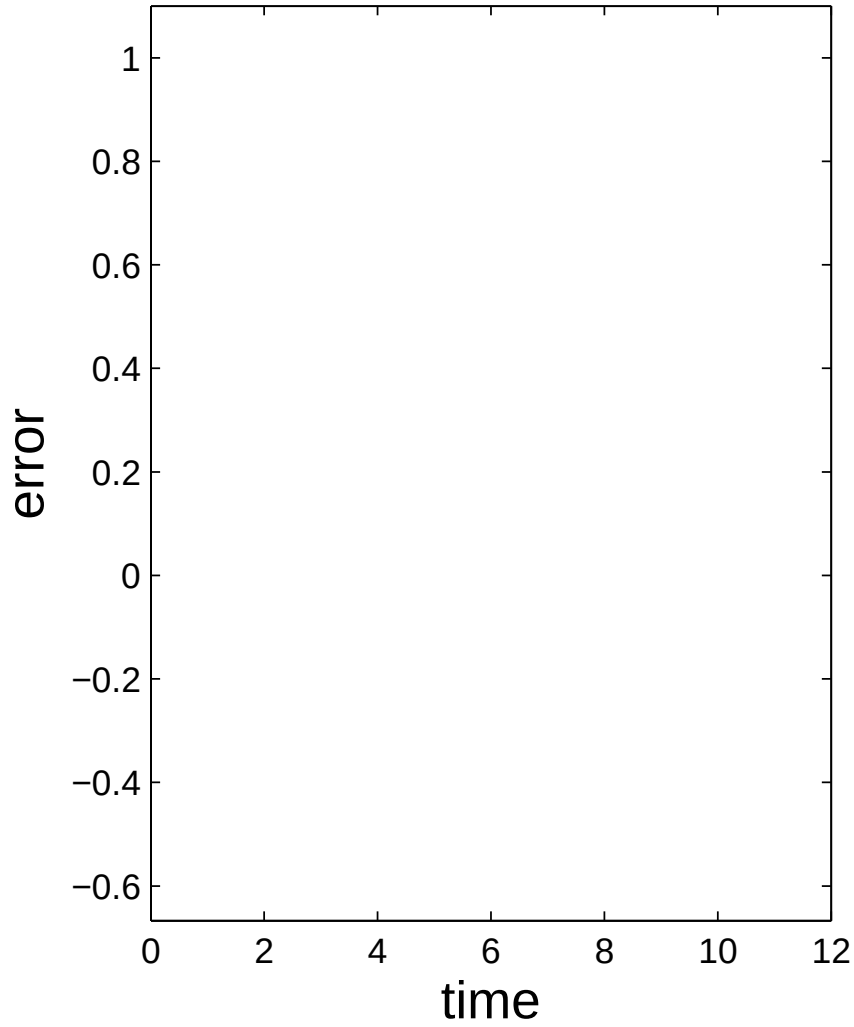
Objective: Move the mass move from position 0 to 1

Question 1: How does the error depend on the setpoint trajectory x_s? I.e. how do we get from 0 to 1?

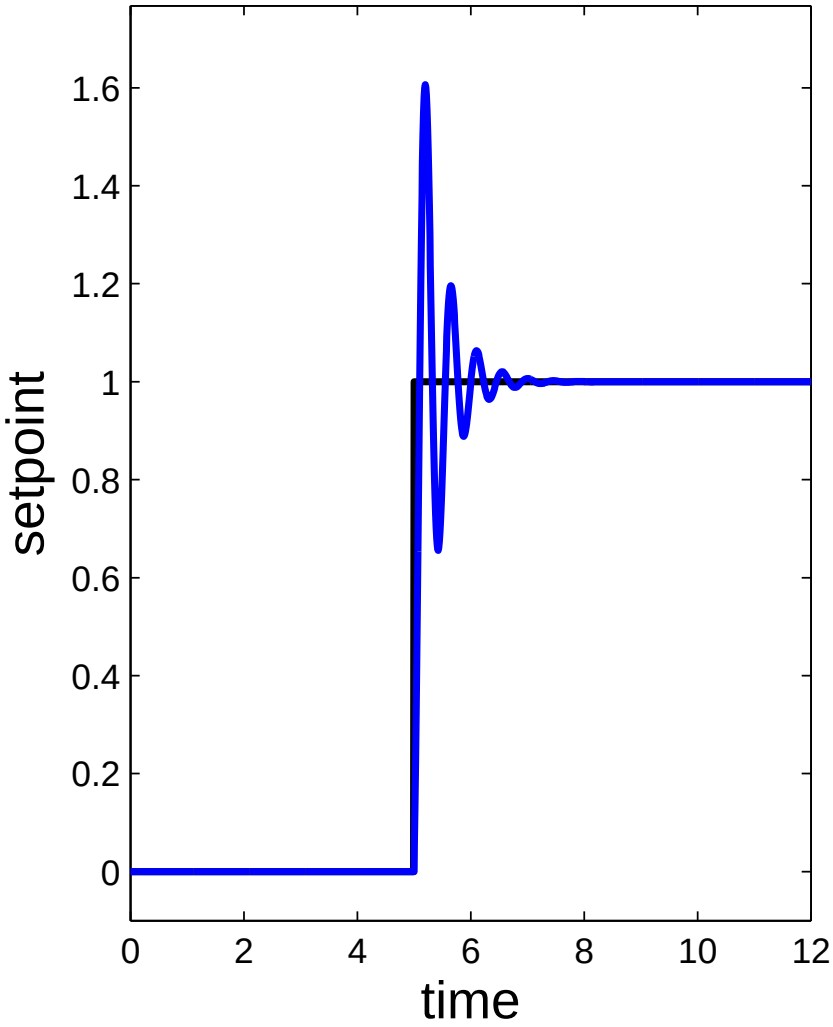
Question 2: Can we obtain (nearly) perfect tracking using just a simple PD-controller?

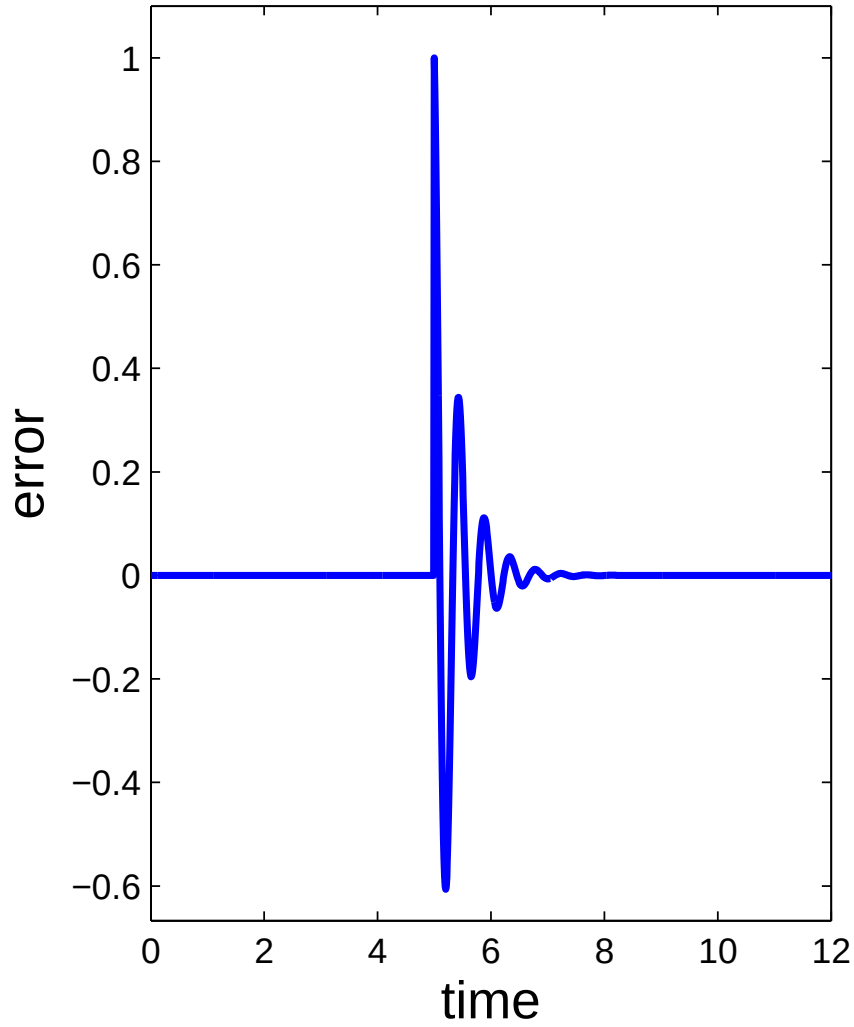
Closed loop response: 0th order setpoint





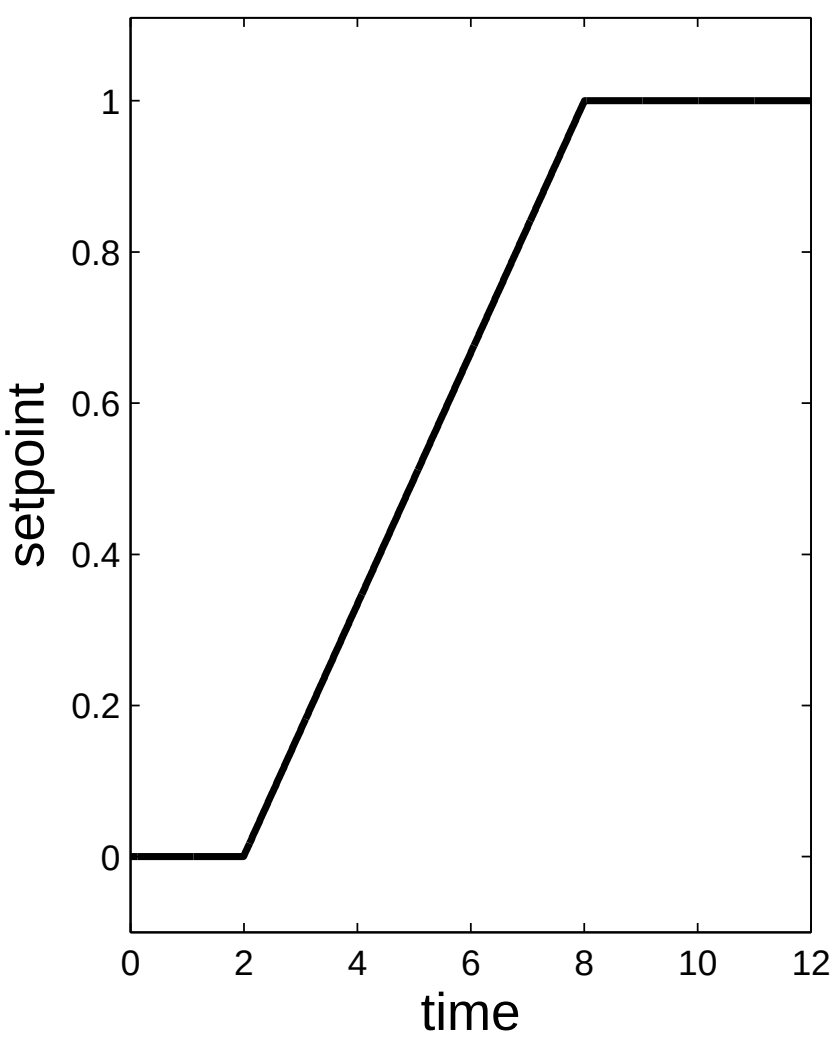
Closed loop response: 0th order setpoint

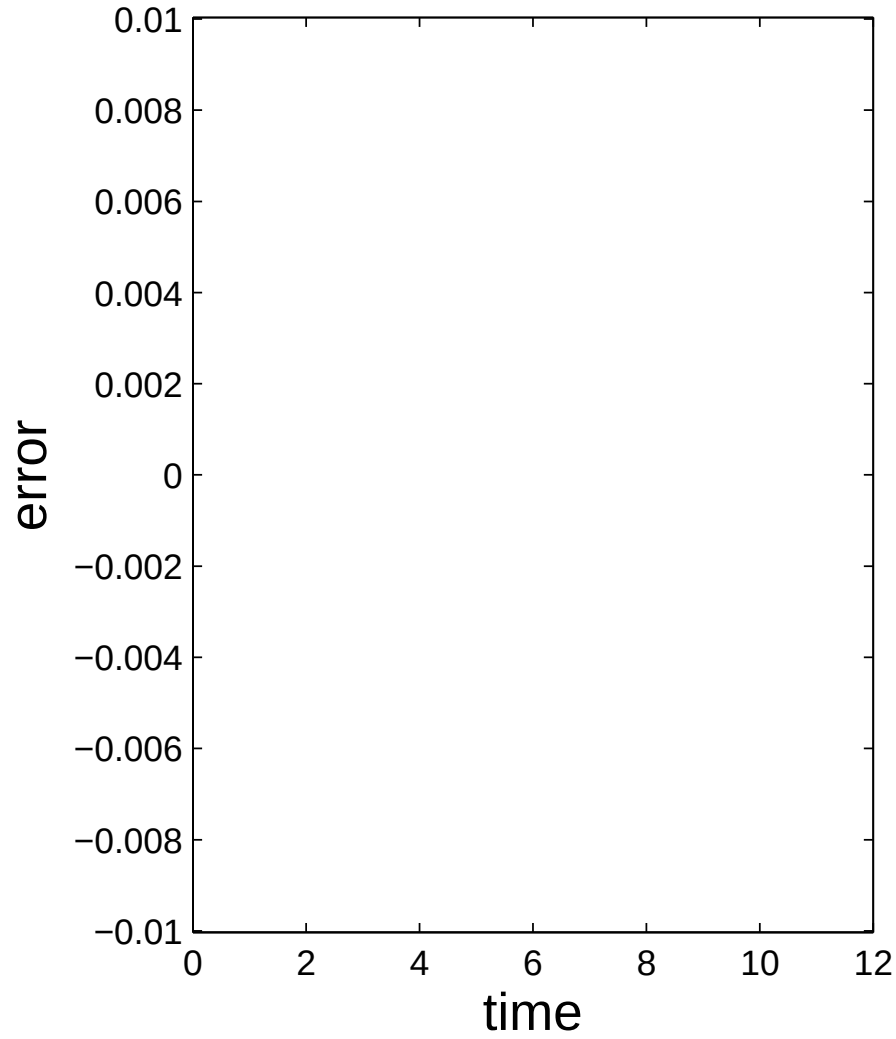




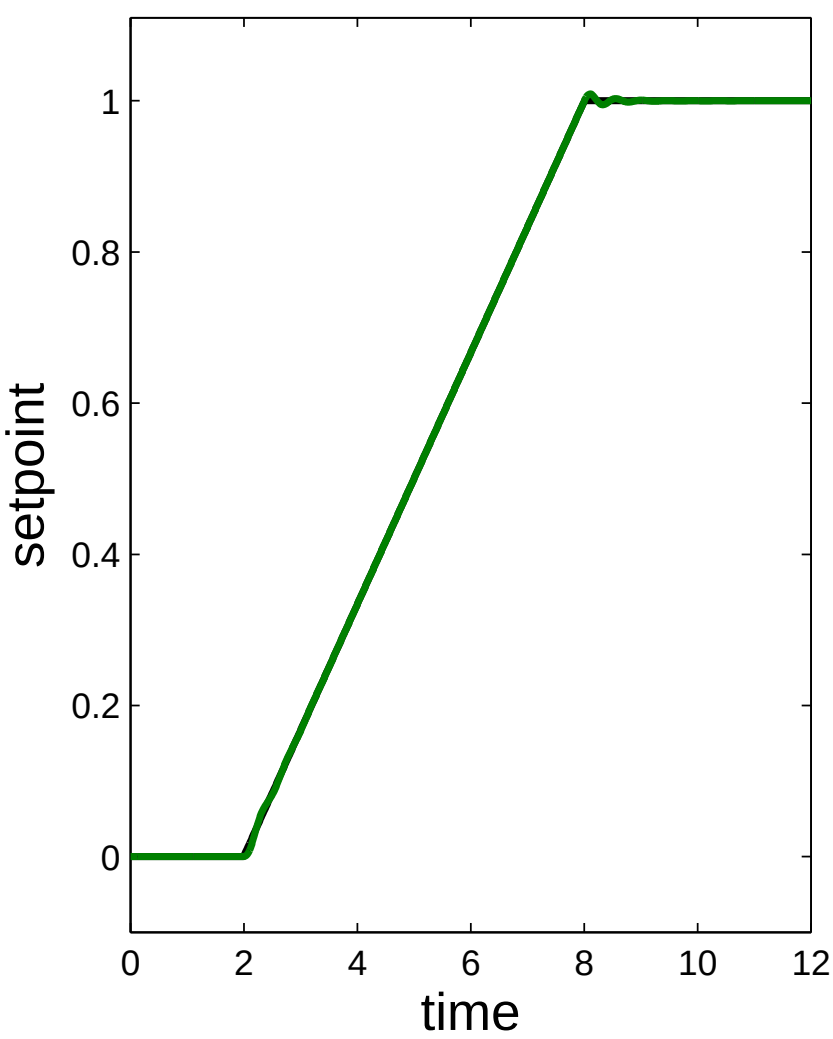
Maximal error / overshoot: 0.6

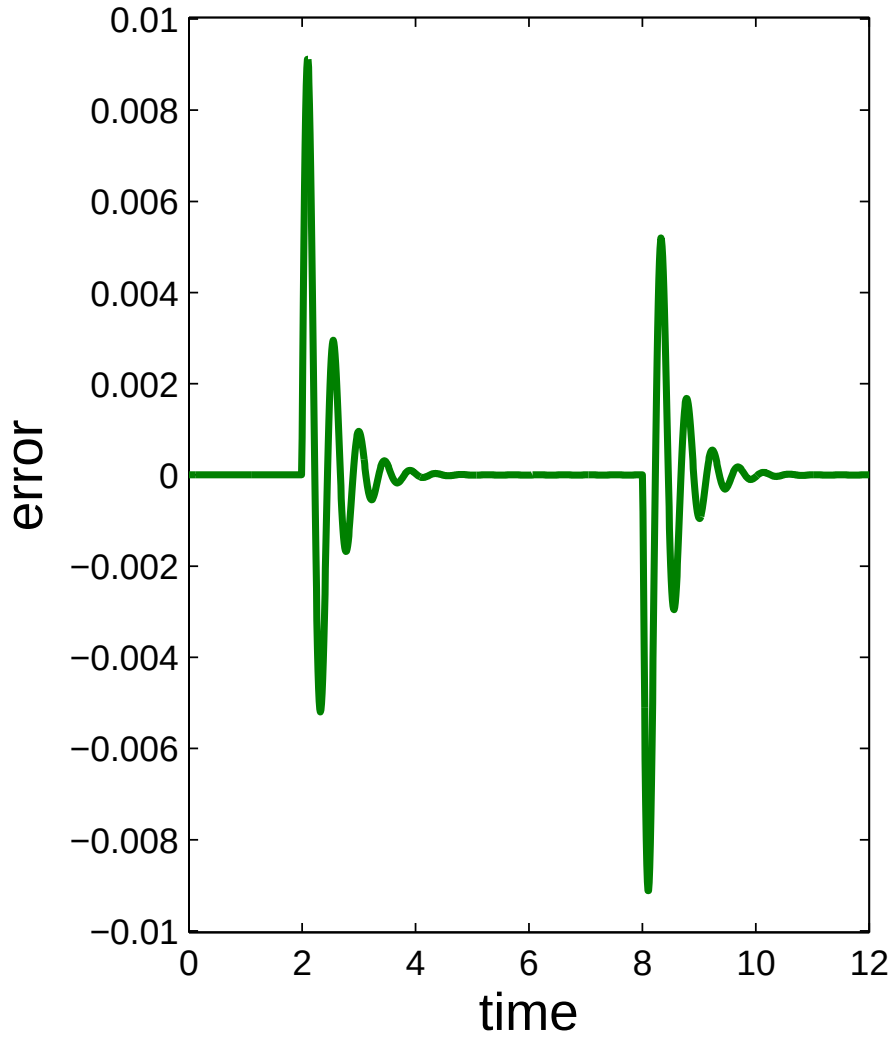
Closed loop response: 1st order setpoint





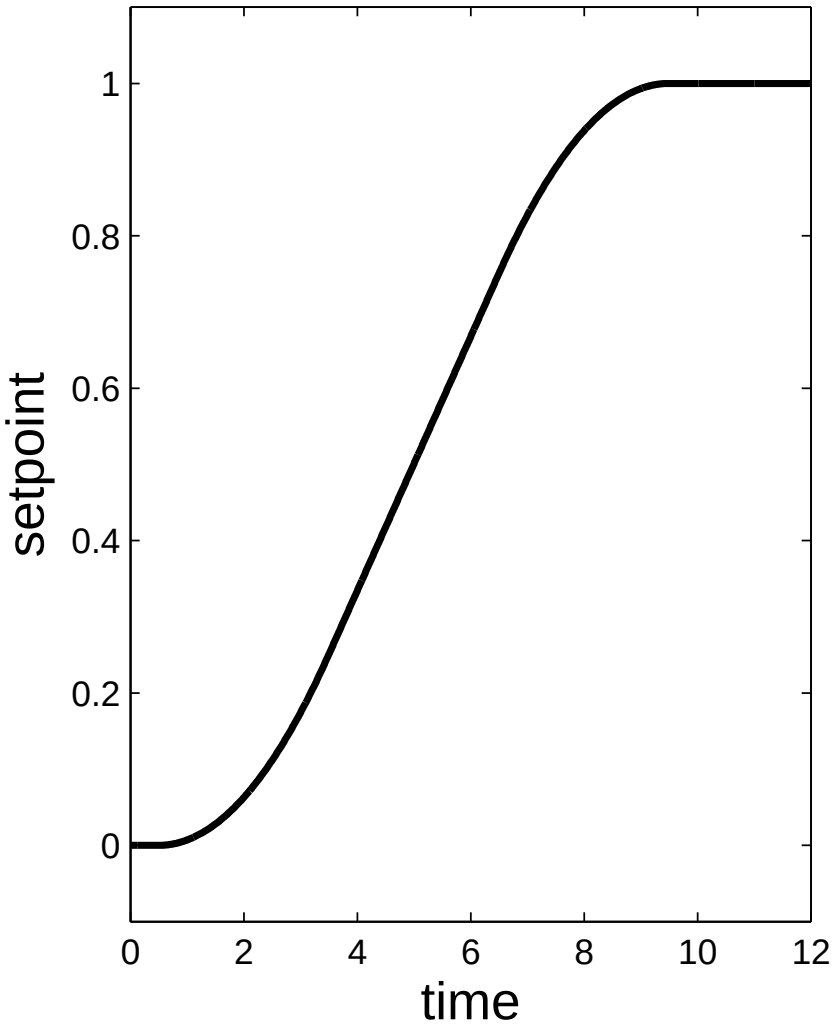
Closed loop response: 1st order setpoint

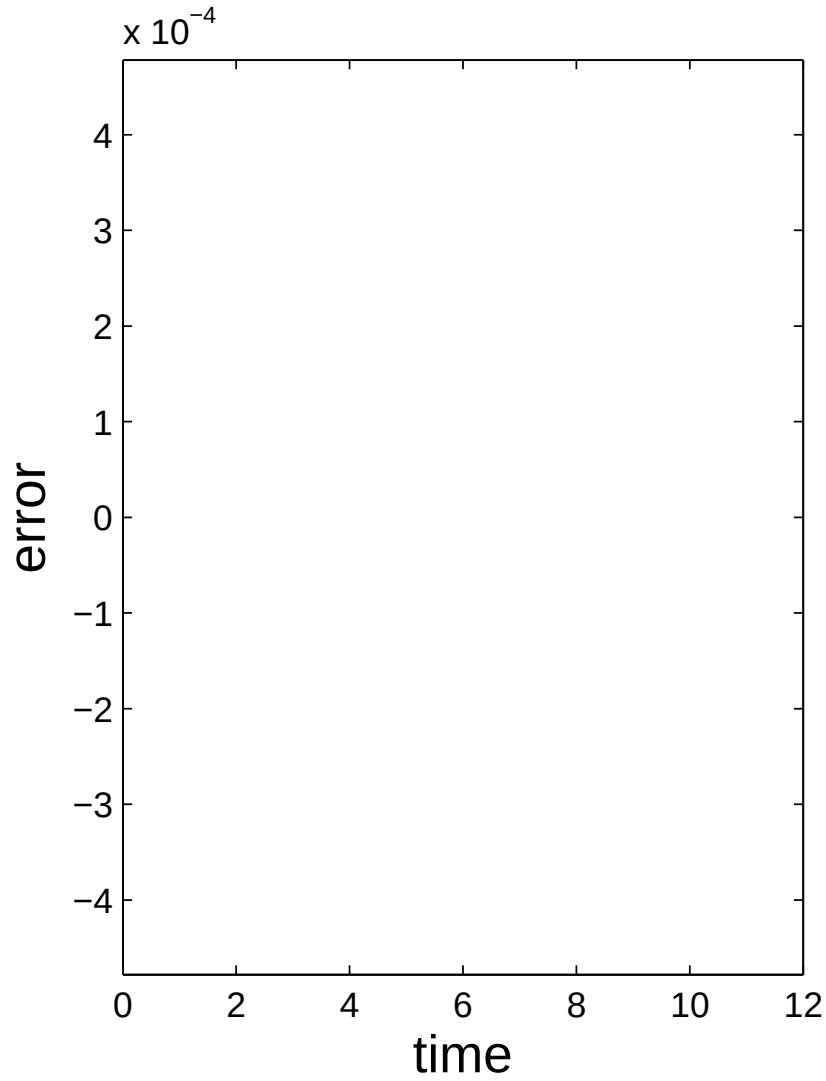




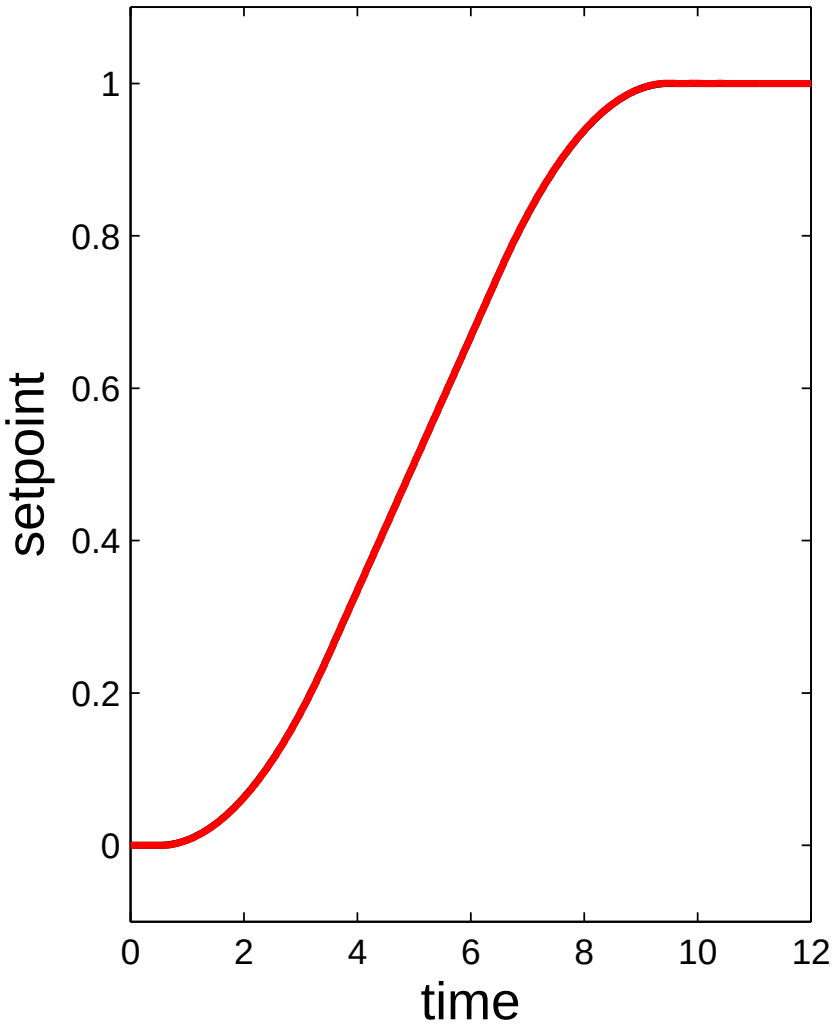
Maximal error / overshoot: 0.009

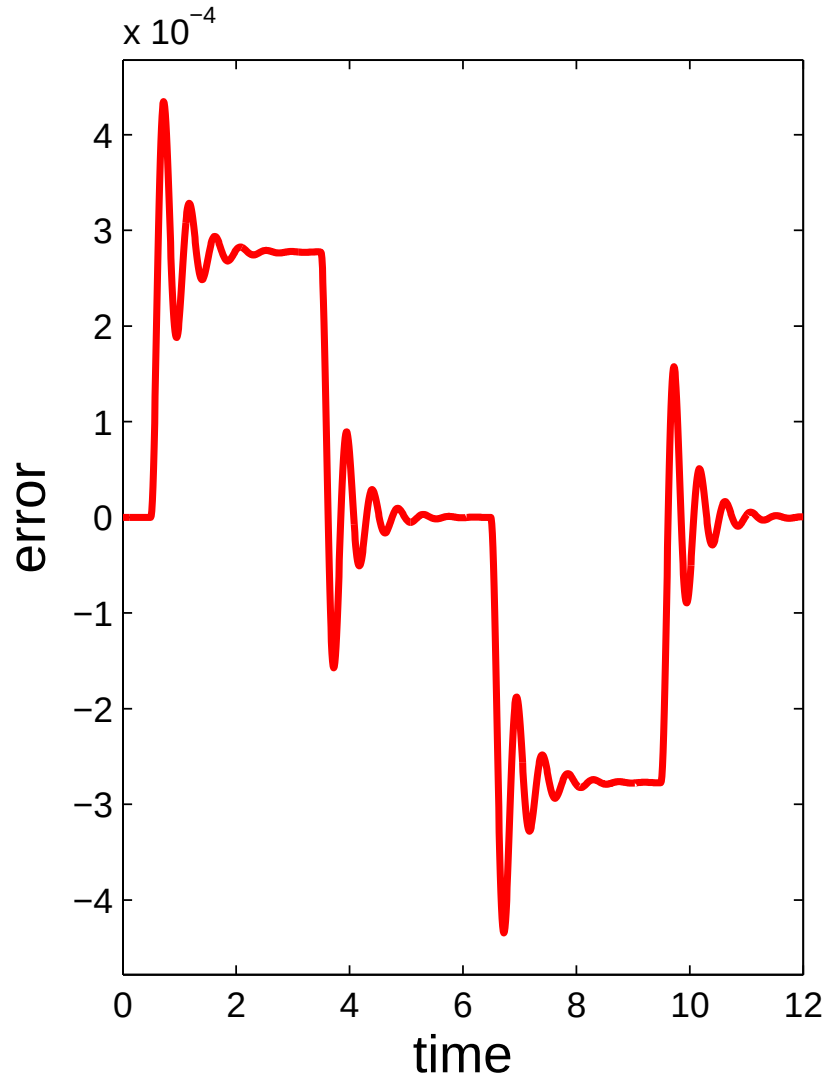
Closed loop response: 2nd order setpoint





Closed loop response: 2nd order setpoint





Maximal error / overshoot: 0.0004

Trajectory planning

Very good tracking results can be obtained

- using a simple (low bandwidth) controller
- using a 'smart' trajectory for x_s
- basic idea: shape trajectory $\rightarrow$ restrict $\dot{x}_s$, $\ddot{x}_s$, etc

Trajectory planning

Very good tracking results can be obtained

- using a simple (low bandwidth) controller
- using a 'smart' trajectory for x_s
- basic idea: shape trajectory $\rightarrow$ restrict $\dot{x}_s$, $\ddot{x}_s$, etc

Advantages of trajectory planning:

- less oscillations
- possibly faster settling times
- smaller control signals (less saturation issues)

Trajectory planning

Very good tracking results can be obtained

- using a simple (low bandwidth) controller
- using a 'smart' trajectory for x_s
- basic idea: shape trajectory $\rightarrow$ restrict $\dot{x}_s$, $\ddot{x}_s$, etc

Advantages of trajectory planning:

- less oscillations
- possibly faster settling times
- smaller control signals (less saturation issues)

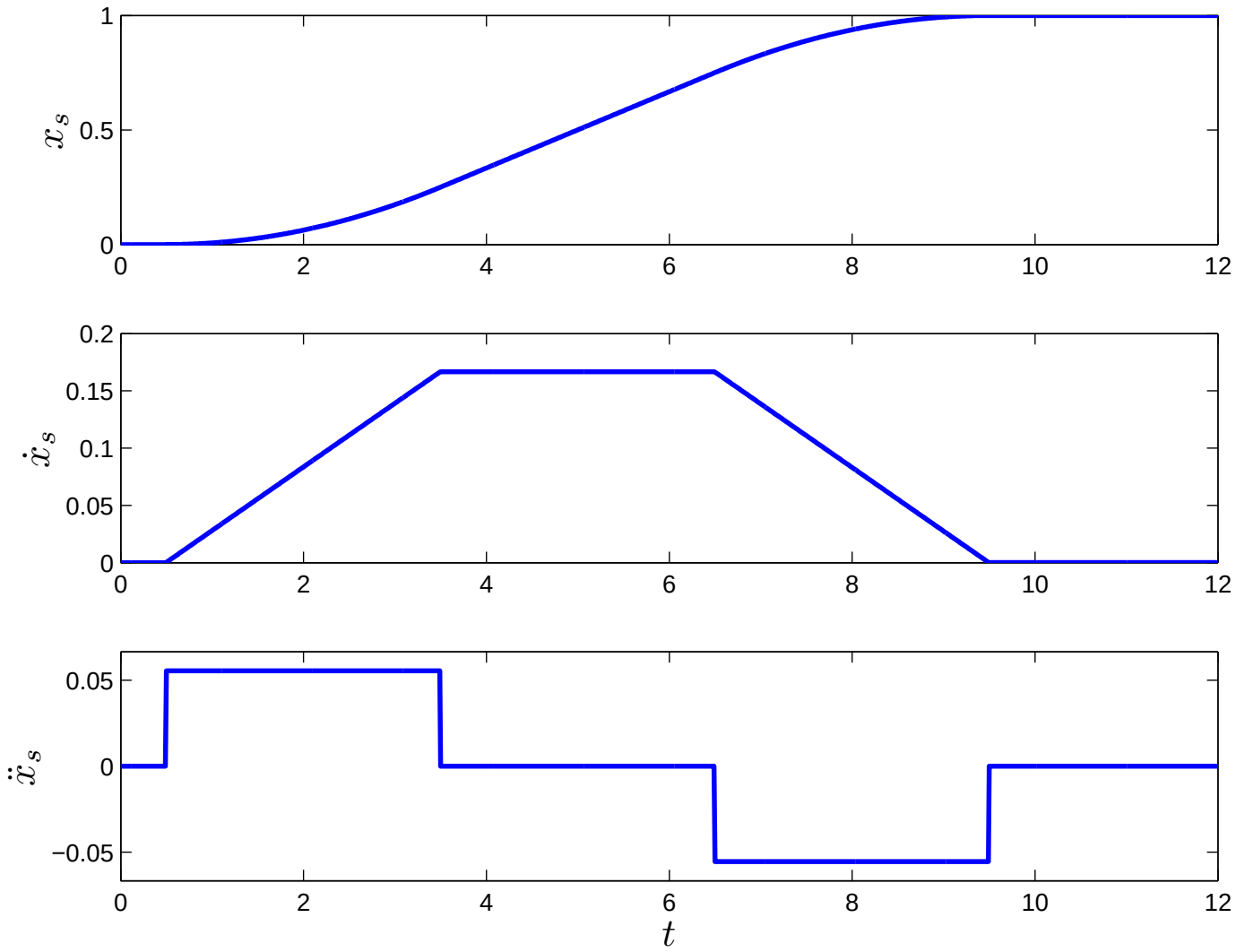
One can wonder...

- Can we quantify the tracking error?
- Is there a way to further reduce this tracking error?

Understanding the servo error

2nd order trajectory

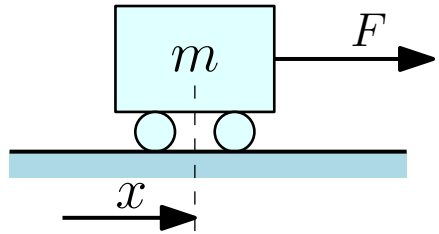
For now, assume only x_s, $\dot{x}_s$ and $\ddot{x}_s$ are restricted



Tracking example 1: error dynamics

Equation of motion:

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



PD-controller:

$$\begin{aligned} F &= K_p e + K_v \dot{e} \\ &= K_p (x_s - x) + K_v (\dot{x}_s - \dot{x}) \end{aligned}$$

Hence

$$\begin{aligned} m\ddot{x} + K_v (\dot{x} - \dot{x}_s) + K_p (x - x_s) &= 0 \\ m (\ddot{x}_s - \ddot{x}) + K_v (\dot{x}_s - \dot{x}) + K_p (x_s - x) &= m\ddot{x}_s \end{aligned}$$

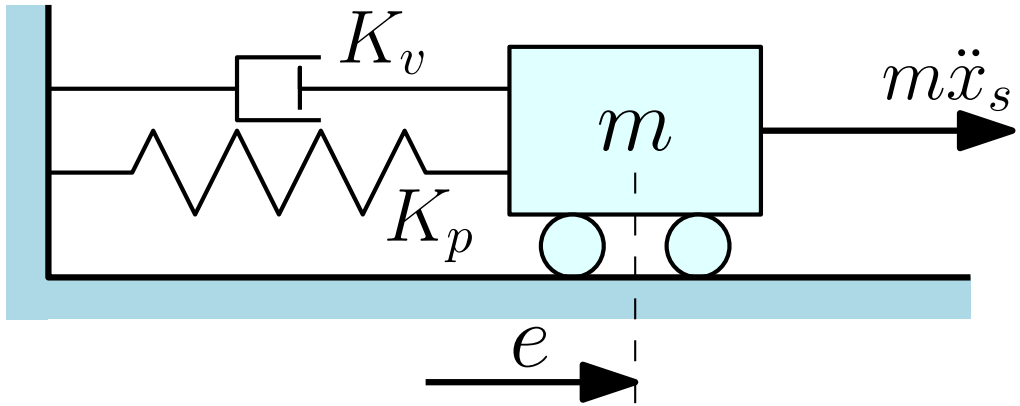
Using $e = x_s - x$, the tracking error dynamics becomes

$$m\ddot{e} + K_v \dot{e} + K_p e = m\ddot{x}_s$$

Tracking example 1: error dynamics

$$\underbrace{m\ddot{e} + K_v\dot{e} + K_p e}_{\text{mass - spring - damper}} = \underbrace{m\ddot{x}_s}_{\text{force on mass}}$$

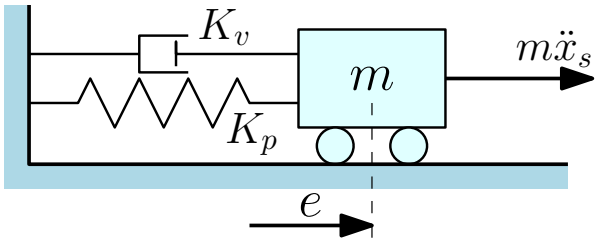
Does e go to zero?



- Iff $\ddot{x}_s = 0$ then $e \to 0$
- Tracking error whenever $\ddot{x}_s \neq 0$

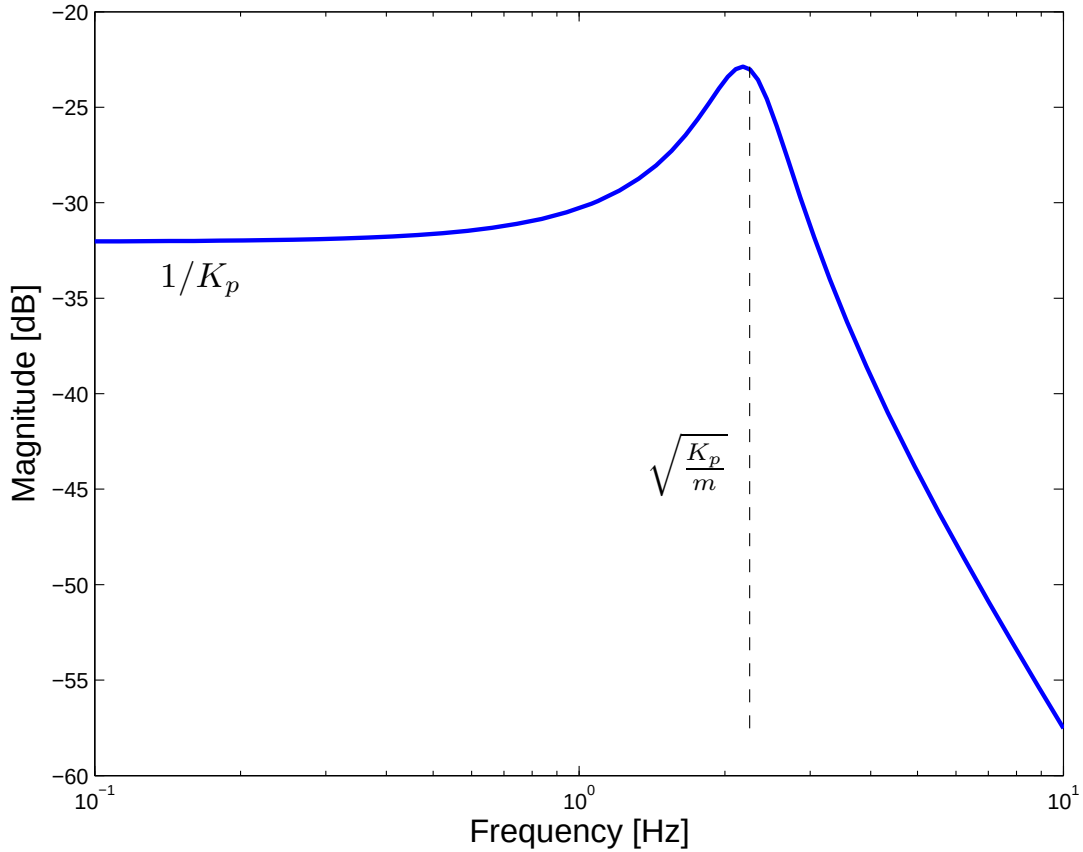
Whenever the mass needs to be accelerated, there is a tracking error!

Tracking example 1: error dynamics

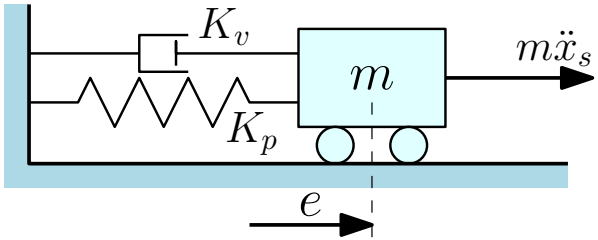


$$m\ddot{e} + K_v\dot{e} + K_p e = u$$

- DC-gain: $\frac{1}{K_p}$
- Eigenfrequency: $\sqrt{\frac{K_p}{m}}$
- $u = m\ddot{x}_s$

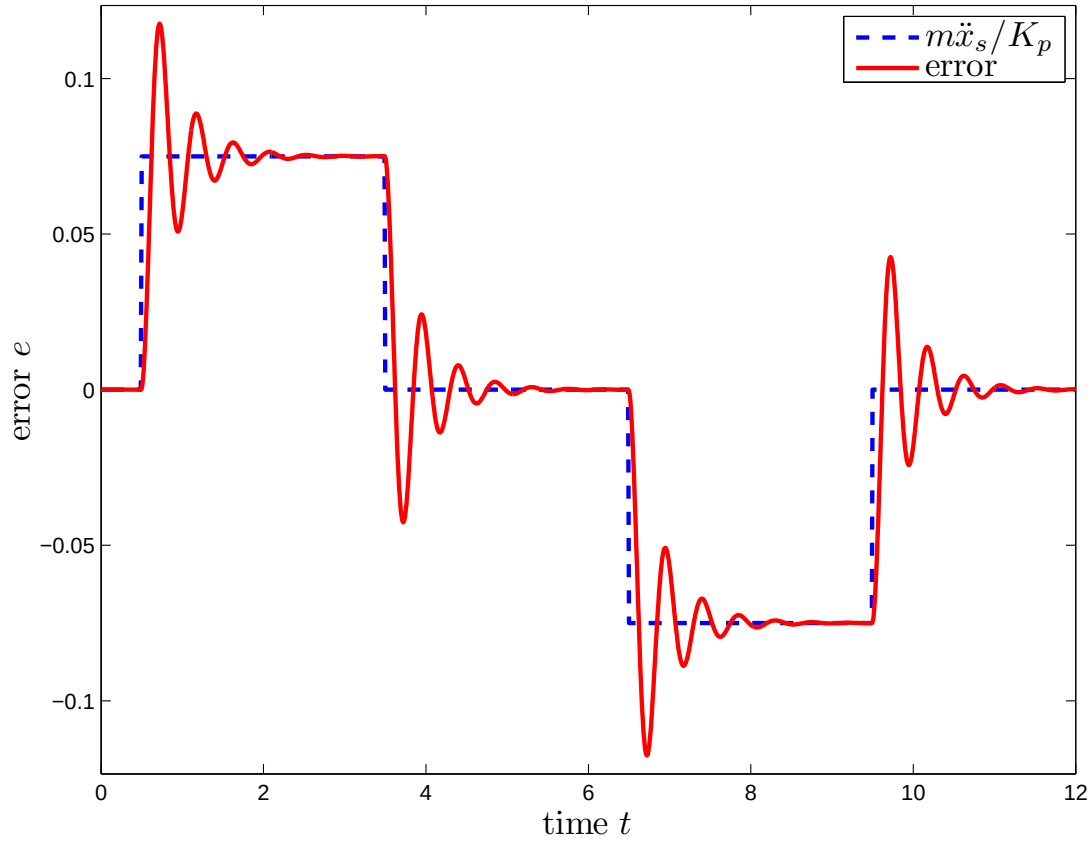


Tracking example 1: error dynamics



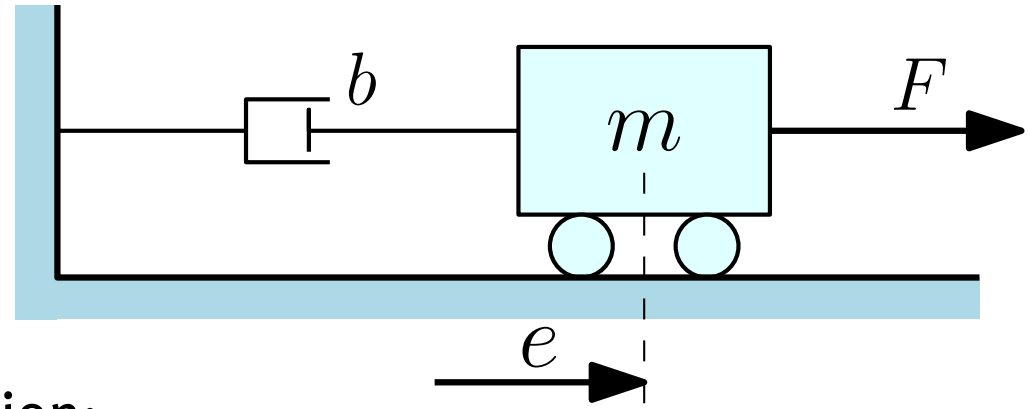
$$m\ddot{e} + K_v\dot{e} + K_p e = u$$

- DC-gain: $\frac{1}{K_p}$
- Eigenfrequency: $\sqrt{\frac{K_p}{m}}$
- $u = m\ddot{x}_s$



Tracking example 2: error dynamics

Suppose the mass encounters friction:



Equation of motion:

$$m\ddot{x} + b\dot{x} = \underbrace{K_p\left(x_s - x\right) + K_v\left(\dot{x}_s - \dot{x}\right)}_{\text{PD-controller}}$$

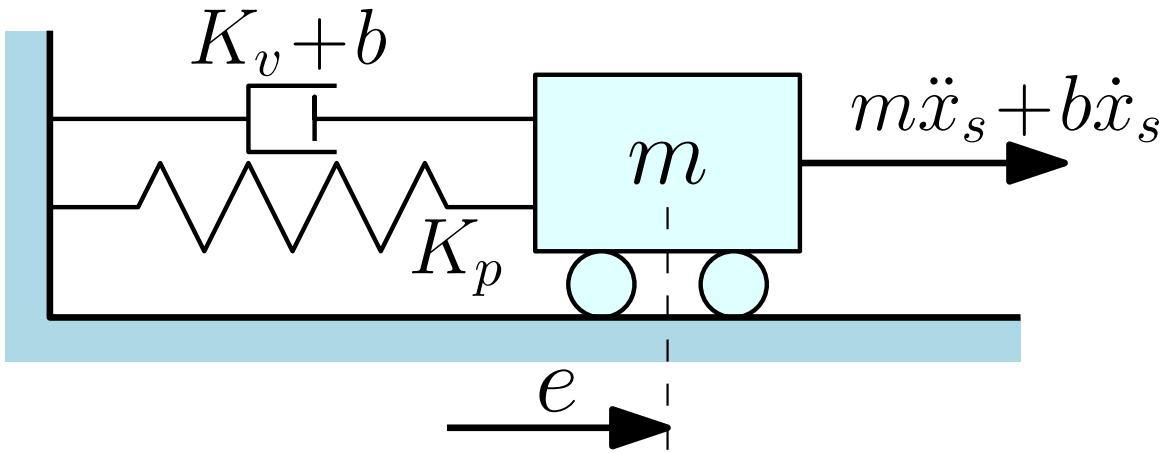
Hence, the error dynamics becomes

$$\begin{aligned} m\left(\ddot{x}_s - \ddot{x}\right) + \left(K_v + b\right)\left(\dot{x}_s - \dot{x}\right) + K_p\left(x_s - x\right) &= m\ddot{x}_s + b\dot{x}_s \\ m\ddot{e} + \left(K_v + b\right)\dot{e} + K_p e &= m\ddot{x}_s + b\dot{x}_s \end{aligned}$$

Tracking example 2: error dynamics

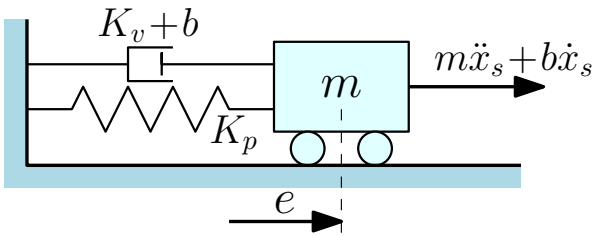
$$\underbrace{m\ddot{e} + (K_v + b)\,\dot{e} + K_p\,e}_{\text{mass - spring - damper}} \quad = \quad \underbrace{m\ddot{x}_s + b\dot{x}_s}_{\text{force on mass}}$$

Does e go to zero?



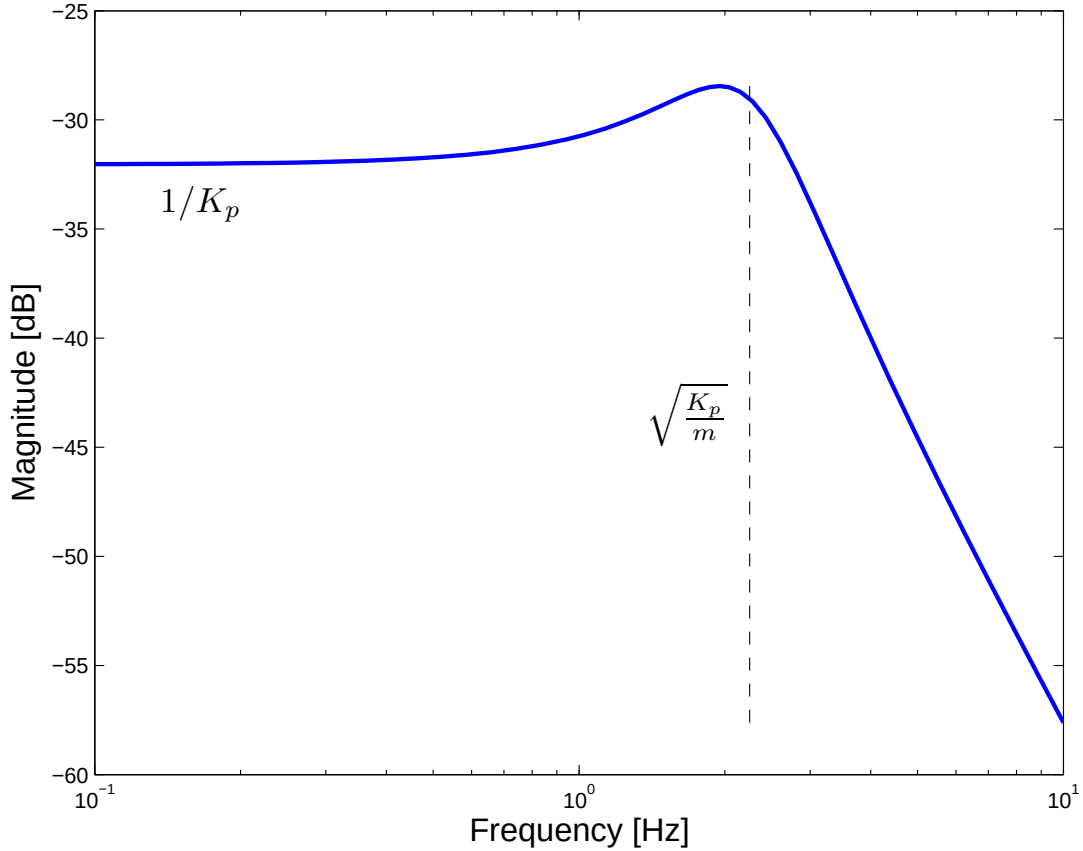
- Iff $\ddot{x}_s = 0$ and $\dot{x}_s = 0$, then $e \to 0$
- Tracking error whenever $\ddot{x}_s \neq 0$ and/or $\dot{x}_s \neq 0$

Tracking example 2: error dynamics

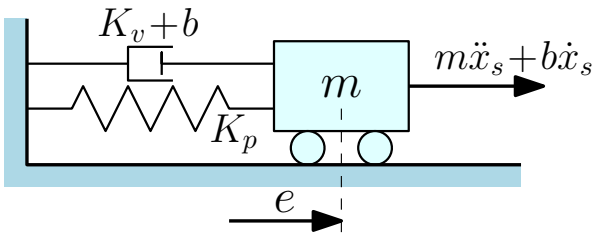


$$m\ddot{e} + (K_v + b)\dot{e} + K_p e = u$$

- DC-gain: $\frac{1}{K_p}$
- Eigenfrequency: $\sqrt{\frac{K_p}{m}}$
- $u = m\ddot{x}_s + b\dot{x}_s$

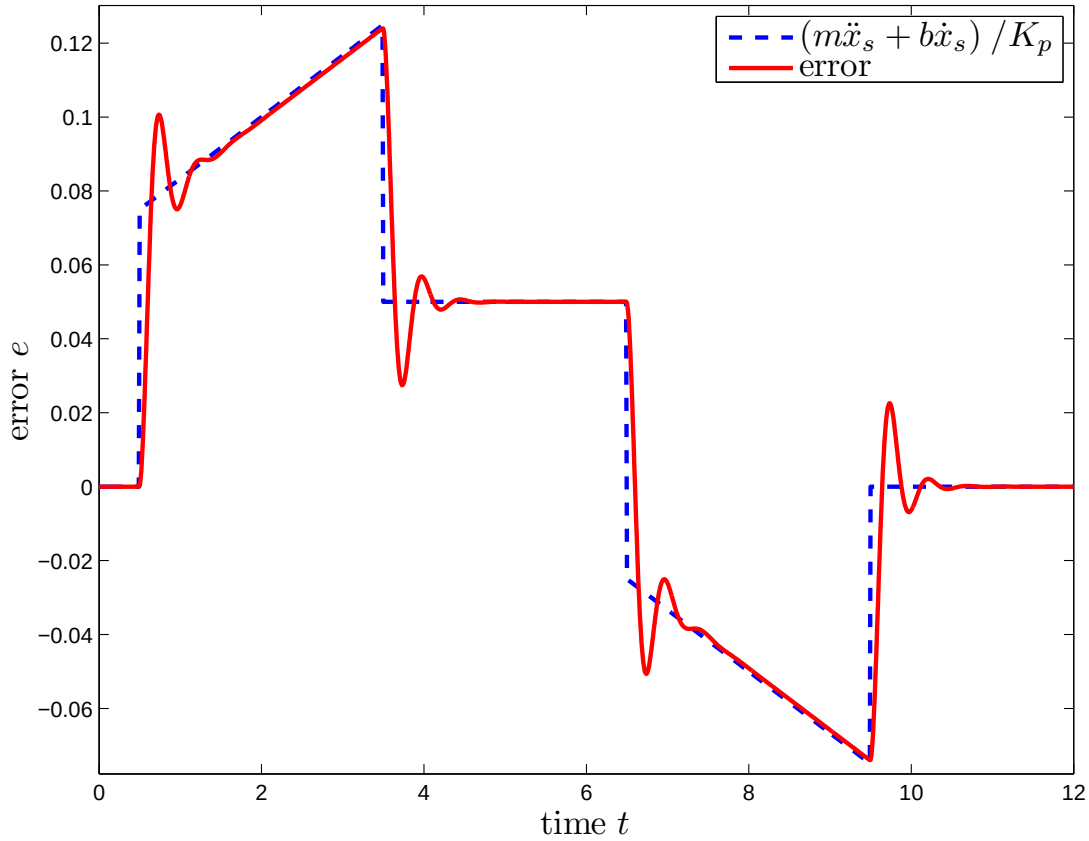


Tracking example 2: error dynamics



$$m\ddot{e} + (K_v + b)\,\dot{e} + K_p\,e = u$$

- DC-gain: $\frac{1}{K_p}$
- Eigenfrequency: $\sqrt{\frac{K_p}{m}}$
- $u = m\ddot{x}_s + b\dot{x}_s$



Error dynamics

How can we decrease this error?

Error dynamics

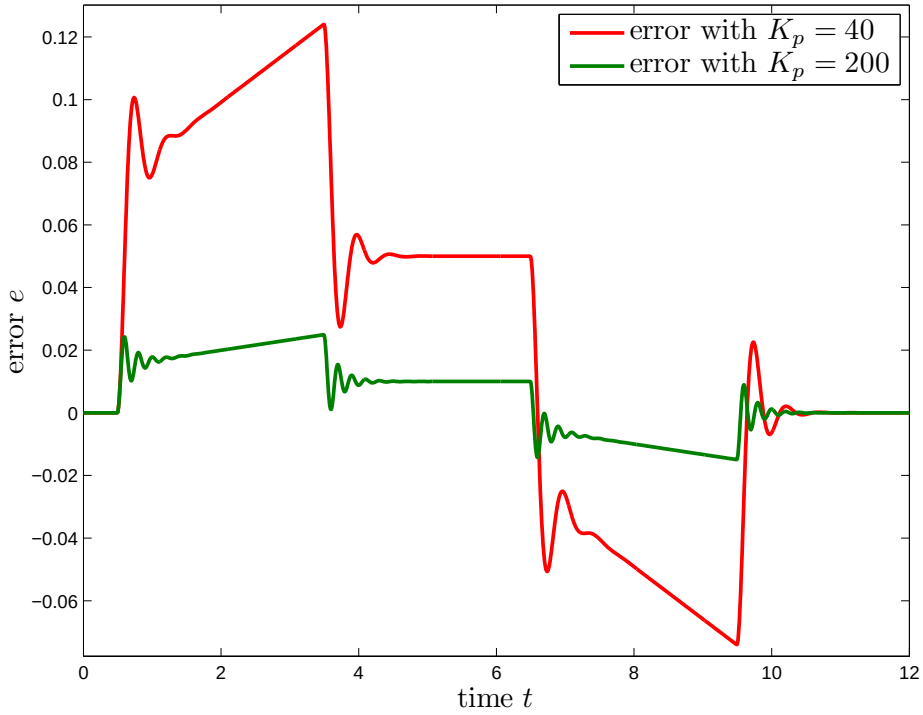
How can we decrease this error?

- Using feedback
  - increase K_p

Error dynamics

How can we decrease this error?

- Using feedback
  - increase K_p

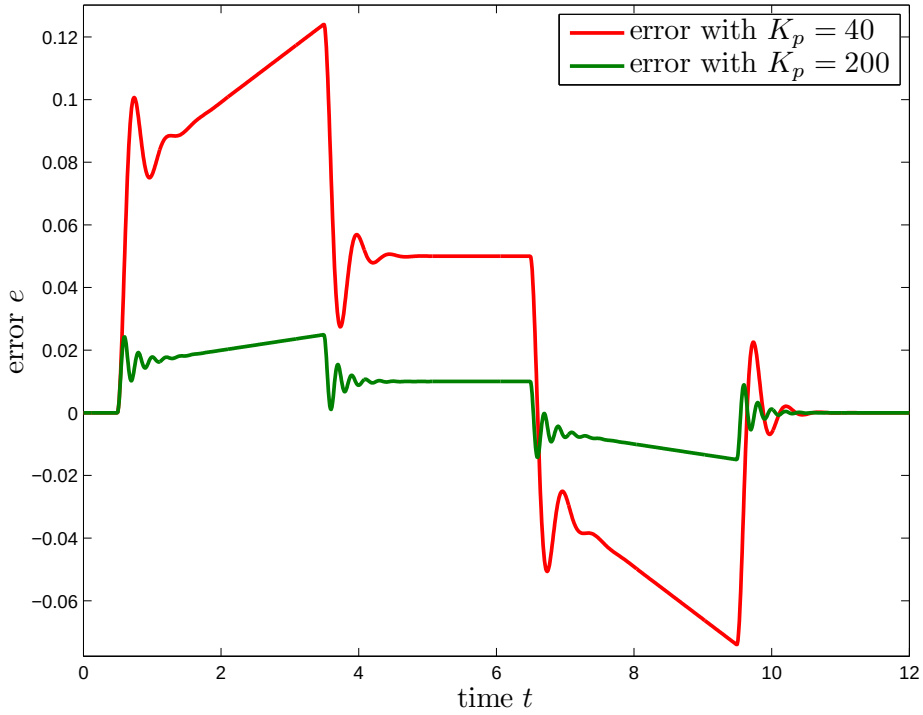


There is still a (steady-state) error...

Error dynamics

How can we decrease this error?

- Using feedback
  - increase K_p



There is still a (steady-state) error...

- Using feedforward

$$m\ddot{e} + K_v\dot{e} + K_p e = \underbrace{m\ddot{x}_s}_{\text{is known!}}$$

- Given x_s, can we compensate the error *beforehand*?

Feedback vs feedforward

Feedback:

- Controller responds to an error
- Error thus already exists
- Feedback = looking back
  - you’re always lagging behind

Feedback vs feedforward

Feedback:

- Controller responds to an error
- Error thus already exists
- Feedback = looking back
  - you're always lagging behind

Feedback vs feedforward

Feedback:

- Controller responds to an error
- Error thus already exists
- Feedback = looking back
  - you're always lagging behind

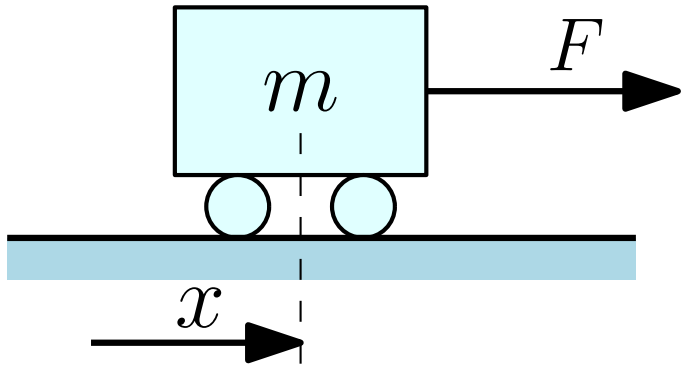
Feedforward:

- Use knowledge about the input
- Correct *before* the error can occur
- Feedforward = looking ahead
  - anticipate what happens or what is needed

Feedforward for motion systems

The inverse problem (1)

How does feedforward work?

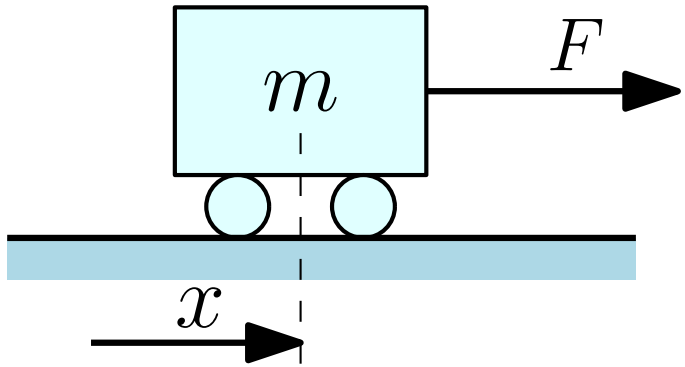


- Goal: zero error for $t \to \infty$ for all x_s:

$$m\ddot{e} + K_v\dot{e} + K_p e = 0$$

The inverse problem (1)

How does feedforward work?



- Goal: zero error for $t \to \infty$ for all x_s:

$$m\ddot{e} + K_v\dot{e} + K_p e = 0$$

- Since $m\ddot{x} = F$, we can obtain this when

$$F = \underbrace{m\ddot{x}_s}_{\text{feedforward}} \quad \underbrace{-K_v\dot{e} - K_p e}_{\text{PD-controller}}$$

Feedforward in the loop (1)

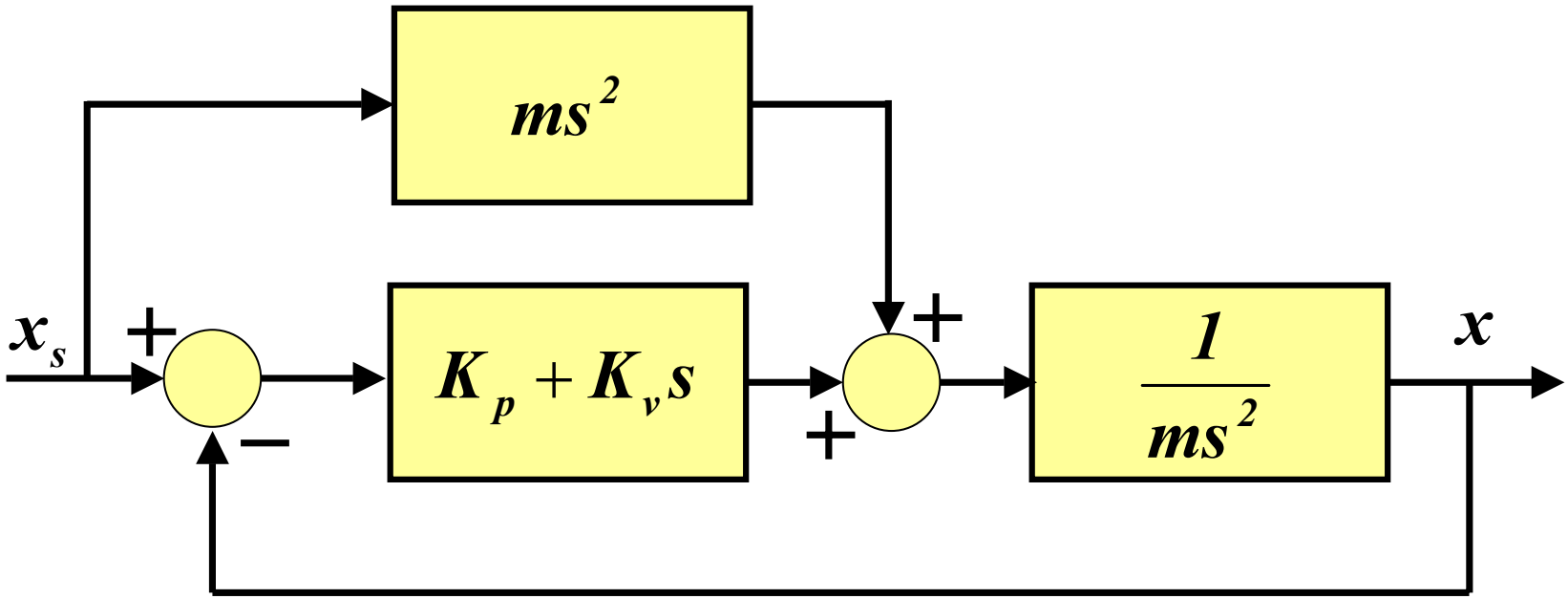
Note that x_s is known and

$$\mathcal{L}\{m\ddot{x}_s\} = ms^2\mathcal{L}\{x_s\}$$

Feedforward in the loop (1)

Note that x_s is known and

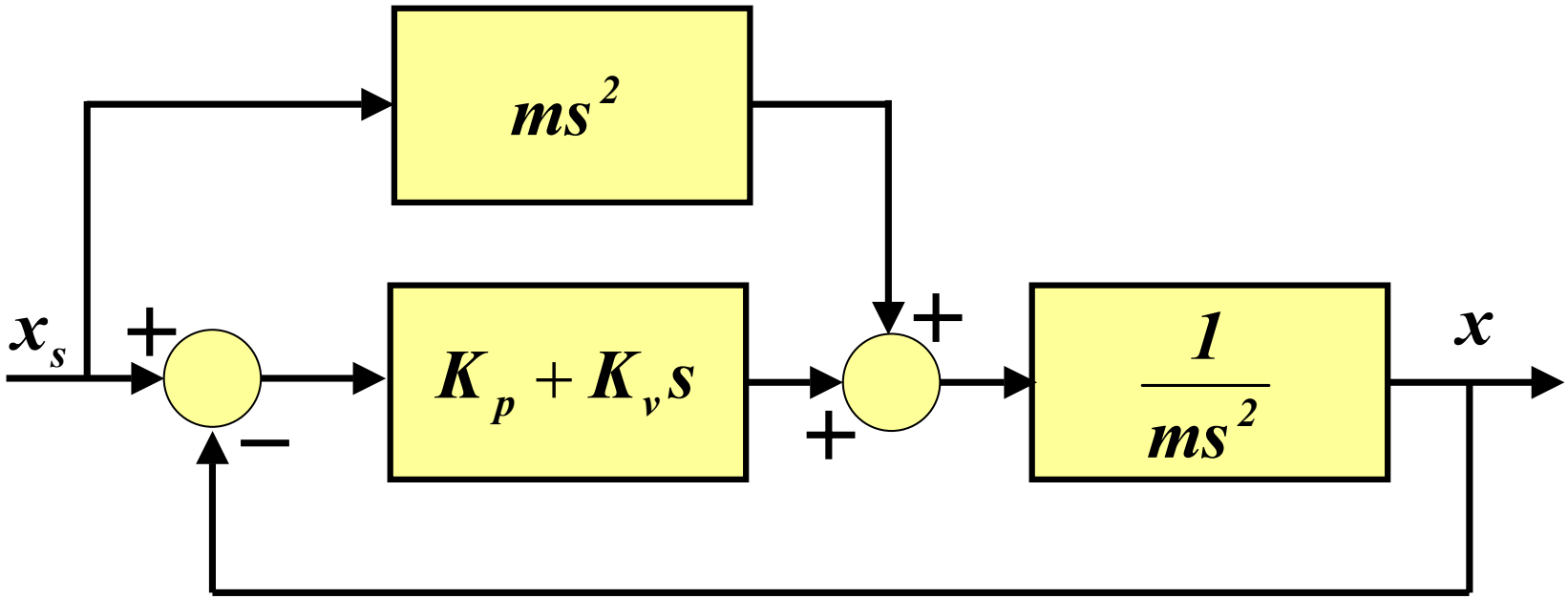
$$\mathcal{L}\{m\ddot{x}_s\} = ms^2\mathcal{L}\{x_s\}$$



Feedforward in the loop (1)

Note that x_s is known and

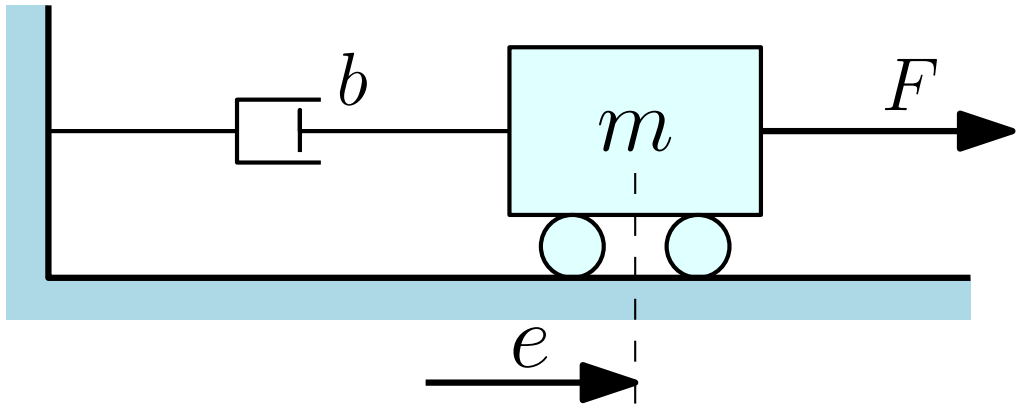
$$\mathcal{L}\{m\ddot{x}_s\} = ms^2\mathcal{L}\{x_s\}$$



Forward feed / prescribe in advance . . .

- . . . the force necessary to accelerate the rigid mass.

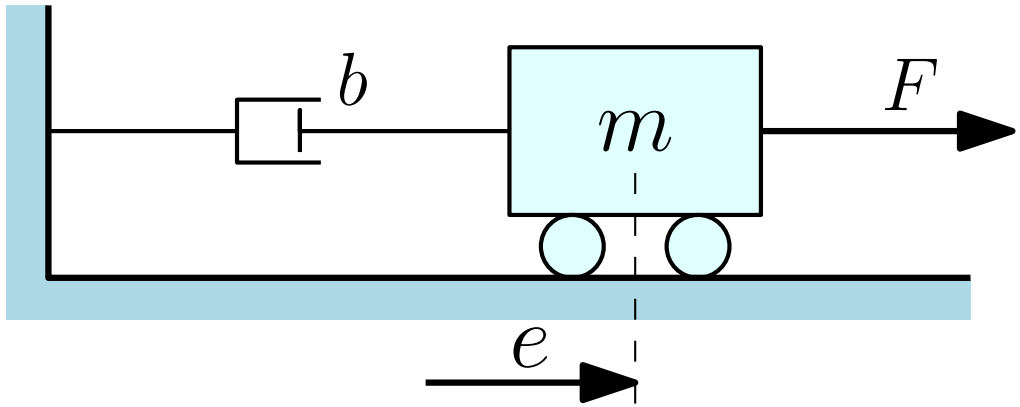
The inverse problem (2)



- Goal: zero error for $t \to \infty$ for all x_s:

$$m\ddot{e} + K_v\dot{e} + K_p e = 0$$

The inverse problem (2)



- Goal: zero error for $t \to \infty$ for all x_s:

$$m\ddot{e} + K_v\dot{e} + K_p e = 0$$

- Since $m\ddot{x} + b\dot{x} = F$, we can obtain this when

$$F = \underbrace{m\ddot{x}_s + b\dot{x}_s}_{\text{feedforward}} \quad \underbrace{-K_v\dot{e} - K_p e}_{\text{PD-controller}}$$

Feedforward in the loop (2)

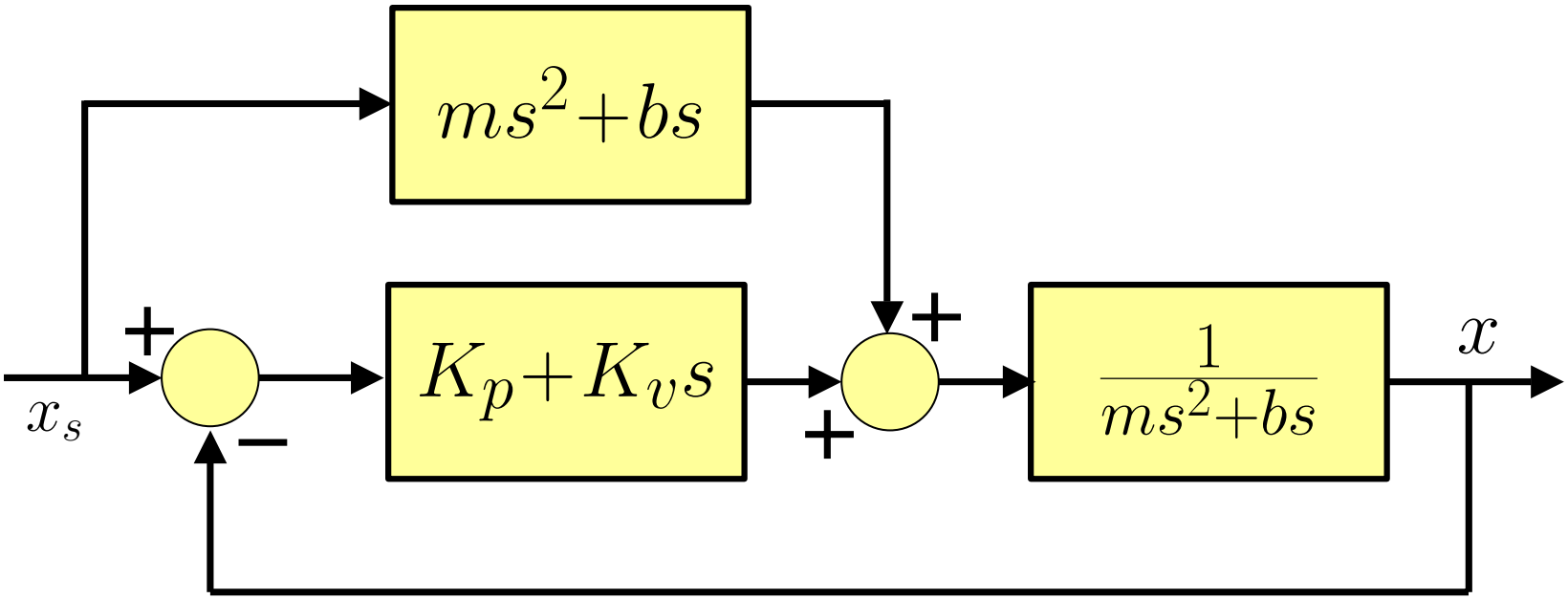
Note that x_s is known and

$$\mathcal{L}\{m\ddot{x}_s + b\dot{x}_s\} = \left(ms^2 + bs\right)\mathcal{L}\{x_s\}$$

Feedforward in the loop (2)

Note that x_s is known and

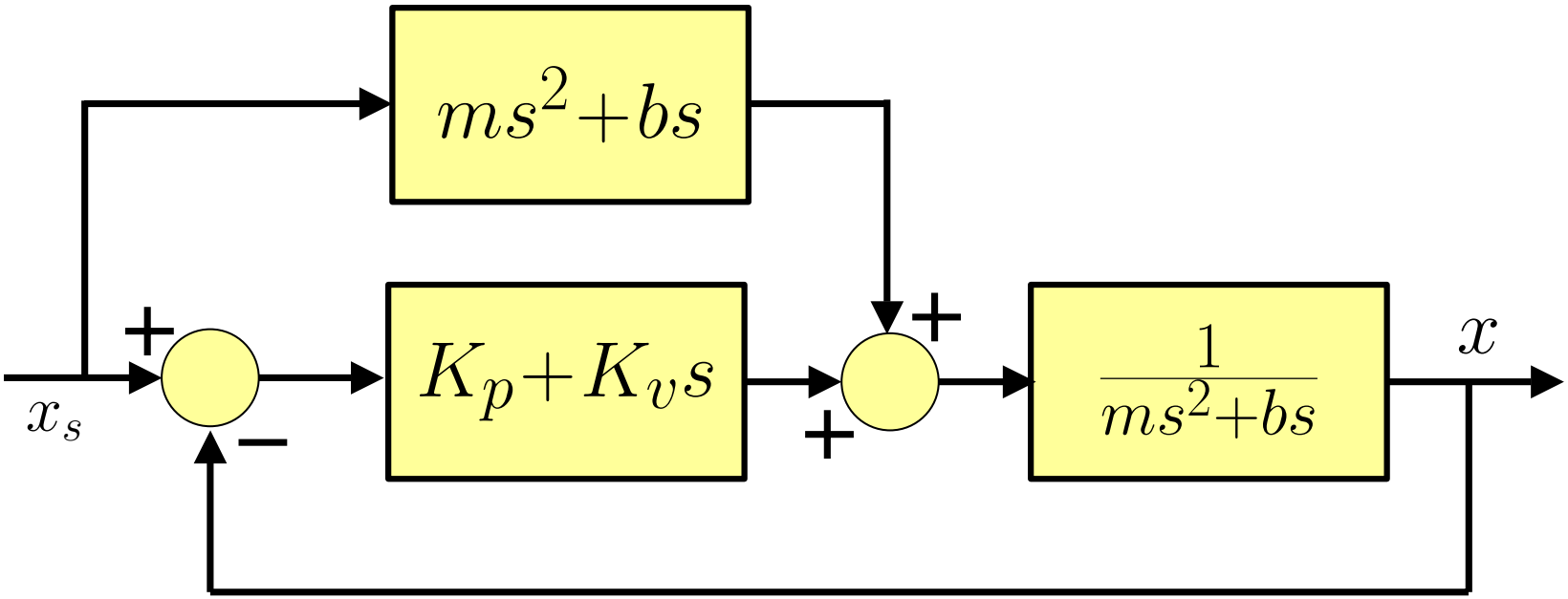
$$\mathcal{L}\{m\ddot{x}_s + b\dot{x}_s\} = \left(ms^2 + bs\right)\mathcal{L}\{x_s\}$$



Feedforward in the loop (2)

Note that x_s is known and

$$\mathcal{L}\{m\ddot{x}_s + b\dot{x}_s\} = \left(ms^2 + bs\right)\mathcal{L}\{x_s\}$$

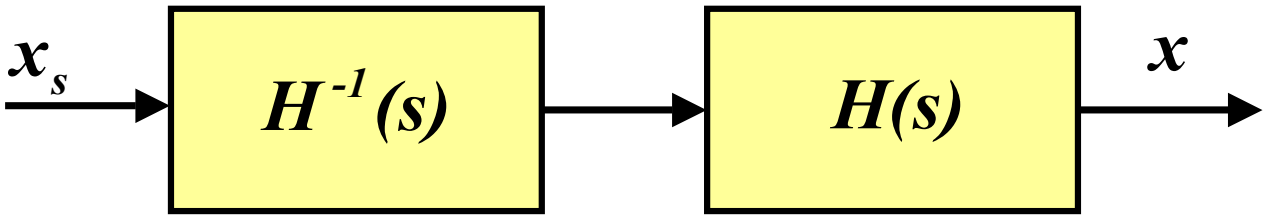


Forward feed / prescribe in advance . . .

- . . . the force necessary to accelerate the rigid mass,
- . . . the force required to overcome the friction.

Pure feedforward

In theory, a feedback controller is not necessary:

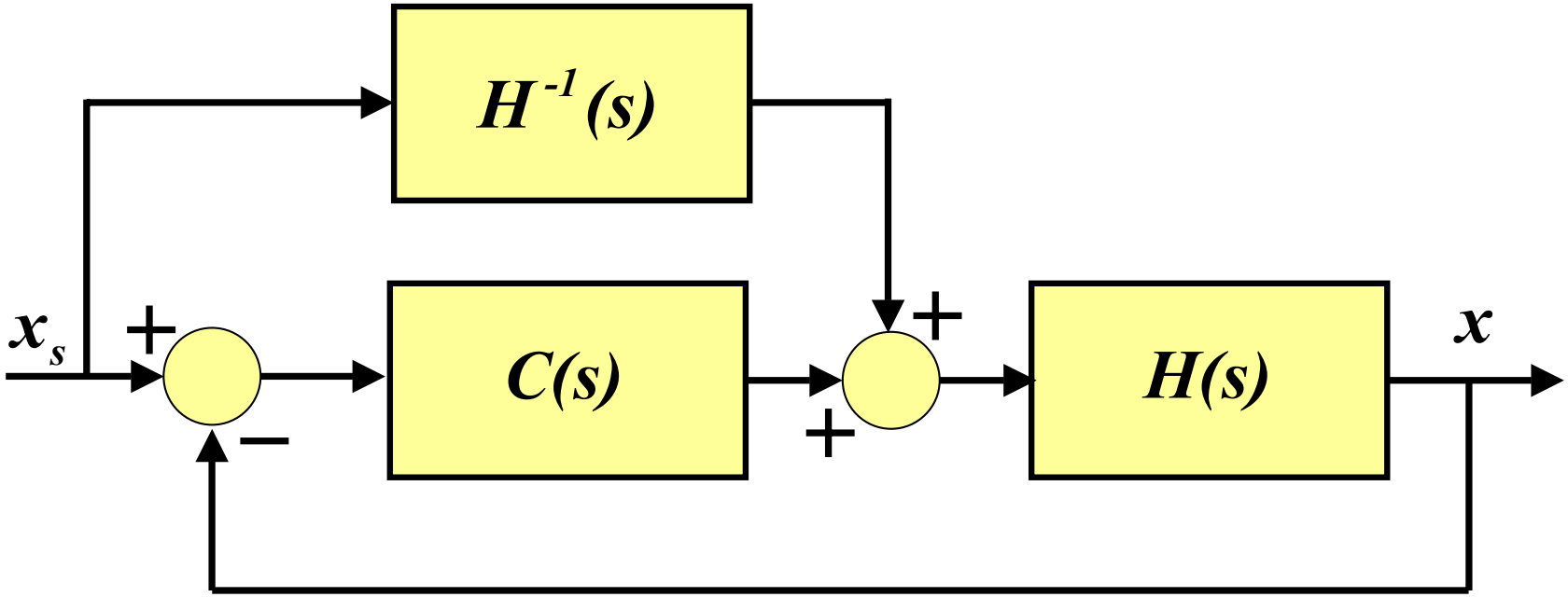


Ideal feedforward:

- $C_{ff}(s) = H^{-1}(s)$
- $x = H(s)H^{-1}(s)x_s = x_s$
- use the plant inverse to compute the plant input *in advance*

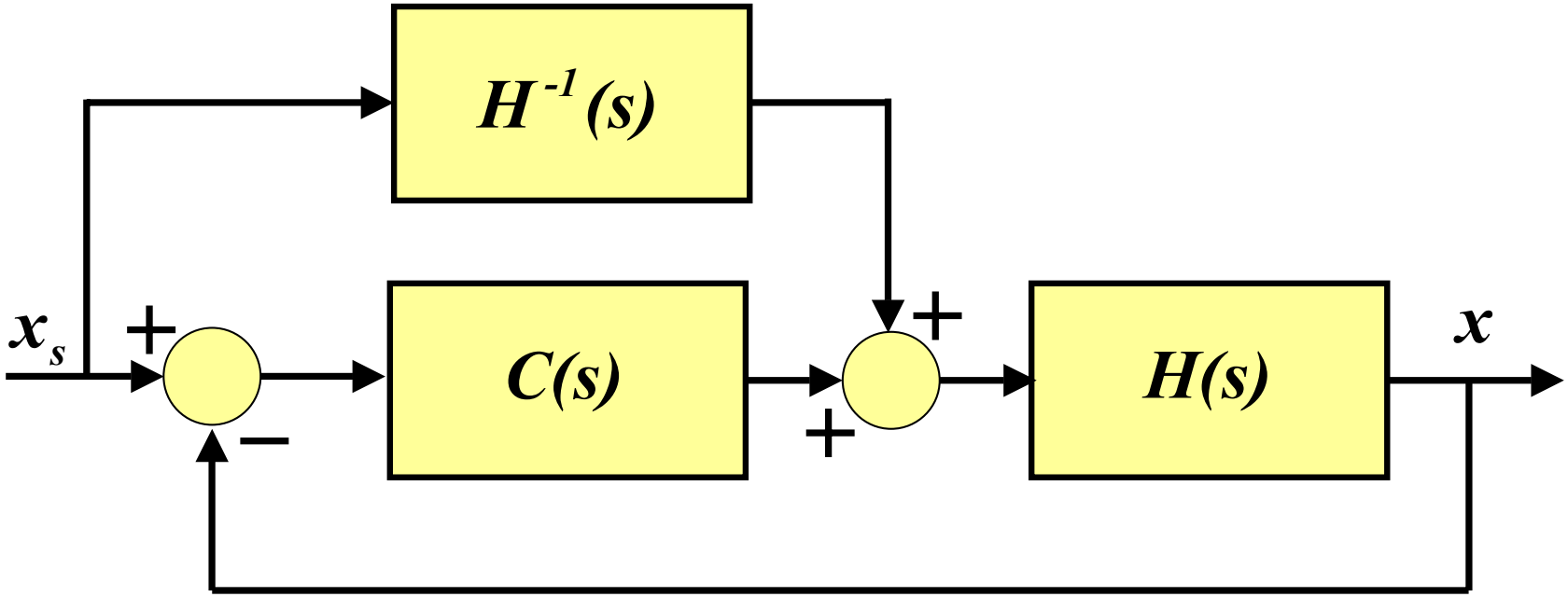
Combination feedback and feedforward

In practice, feedback is needed to compensate disturbances around x_s:



Combination feedback and feedforward

In practice, feedback is needed to compensate disturbances around x_s:



However,

- what if $H^{-1}(s)$ is unstable?
- what if $H(s)$ is unknown?

Then we try to approximate $H^{-1}(s)$ in time domain.

General feedforward (1)

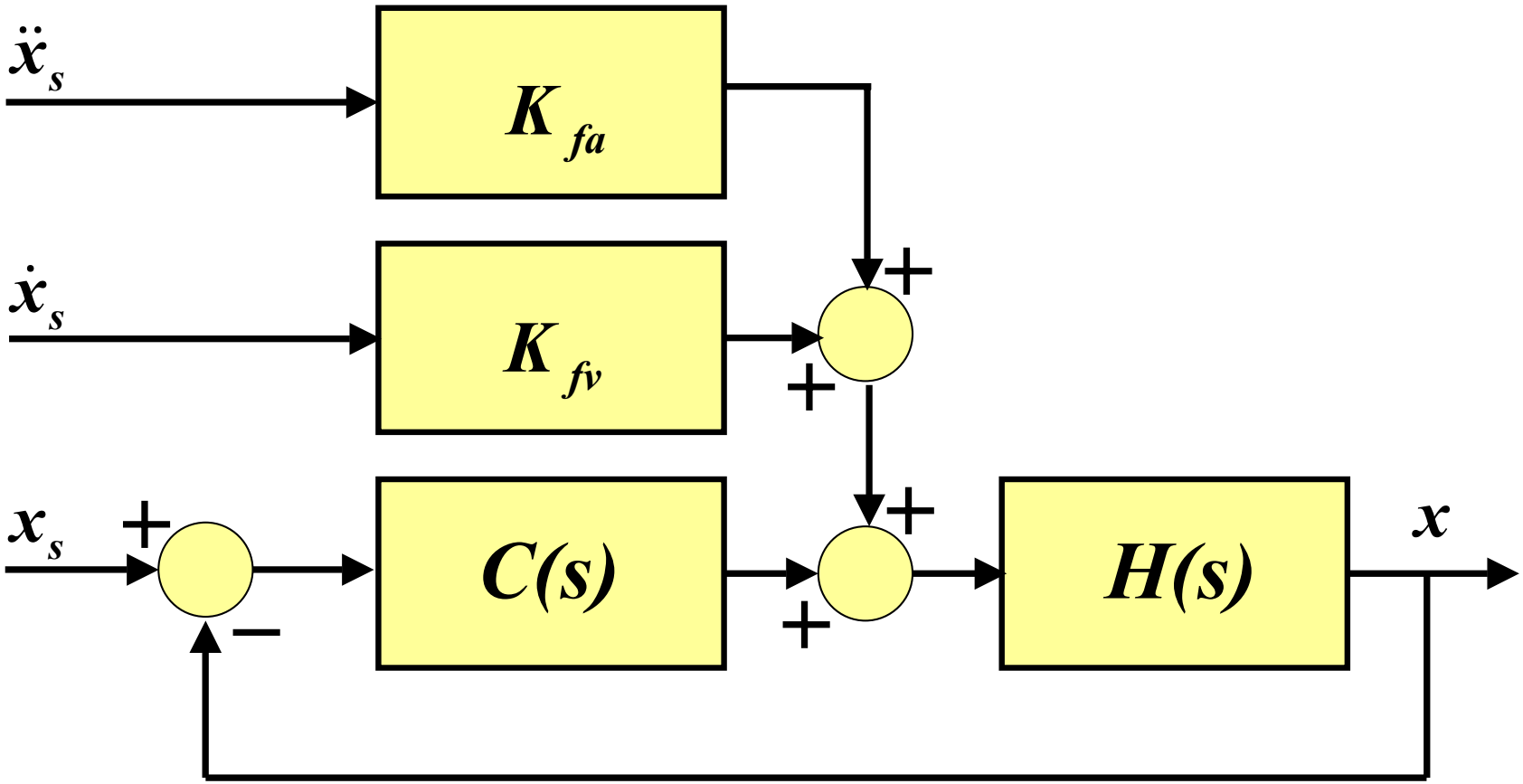
Forward feeding the force required to...

- ...accelerate the rigid body mode of the plant: $F_a = K_{fa}\ddot{x}_s$
- ...overcome the viscous friction with the fixed world: $F_v = K_{fv}\dot{x}_s$

General feedforward (1)

Forward feeding the force required to...

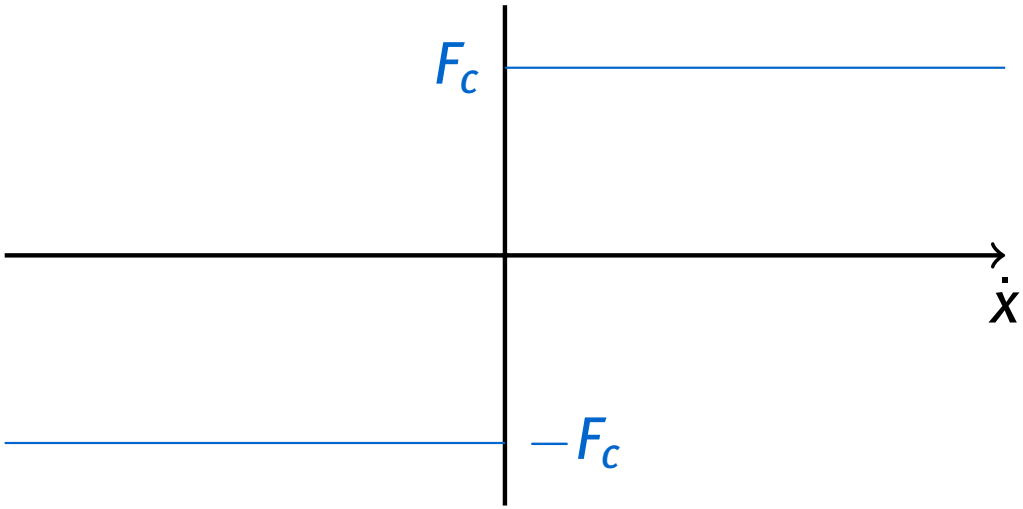
- ...accelerate the rigid body mode of the plant: $F_a = K_{fa}\ddot{x}_s$
- ...overcome the viscous friction with the fixed world: $F_v = K_{fv}\dot{x}_s$



Coulomb friction

What if Coulomb friction (dry friction) is present?

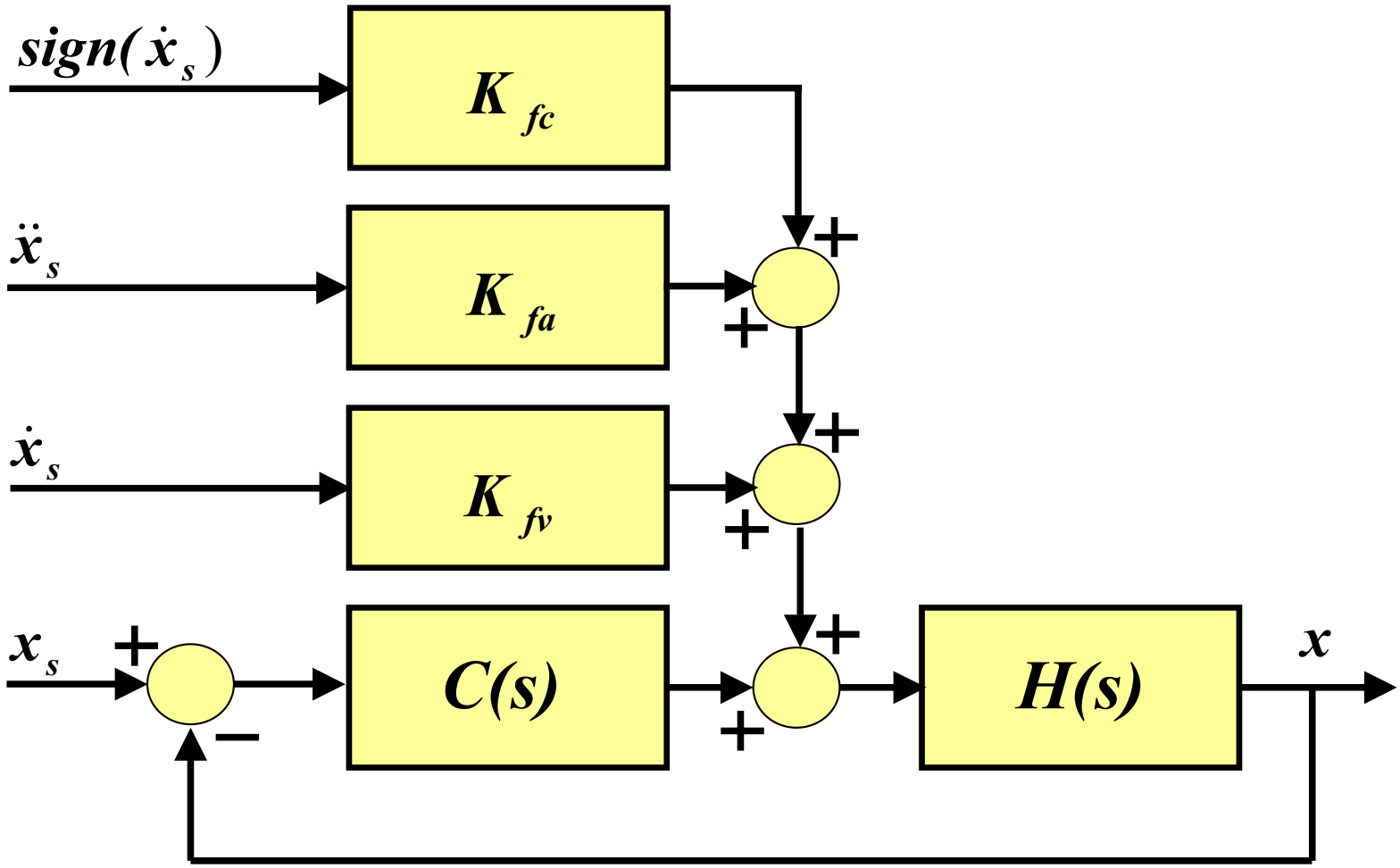
- constant force, opposing direction of motion
- direction dependent



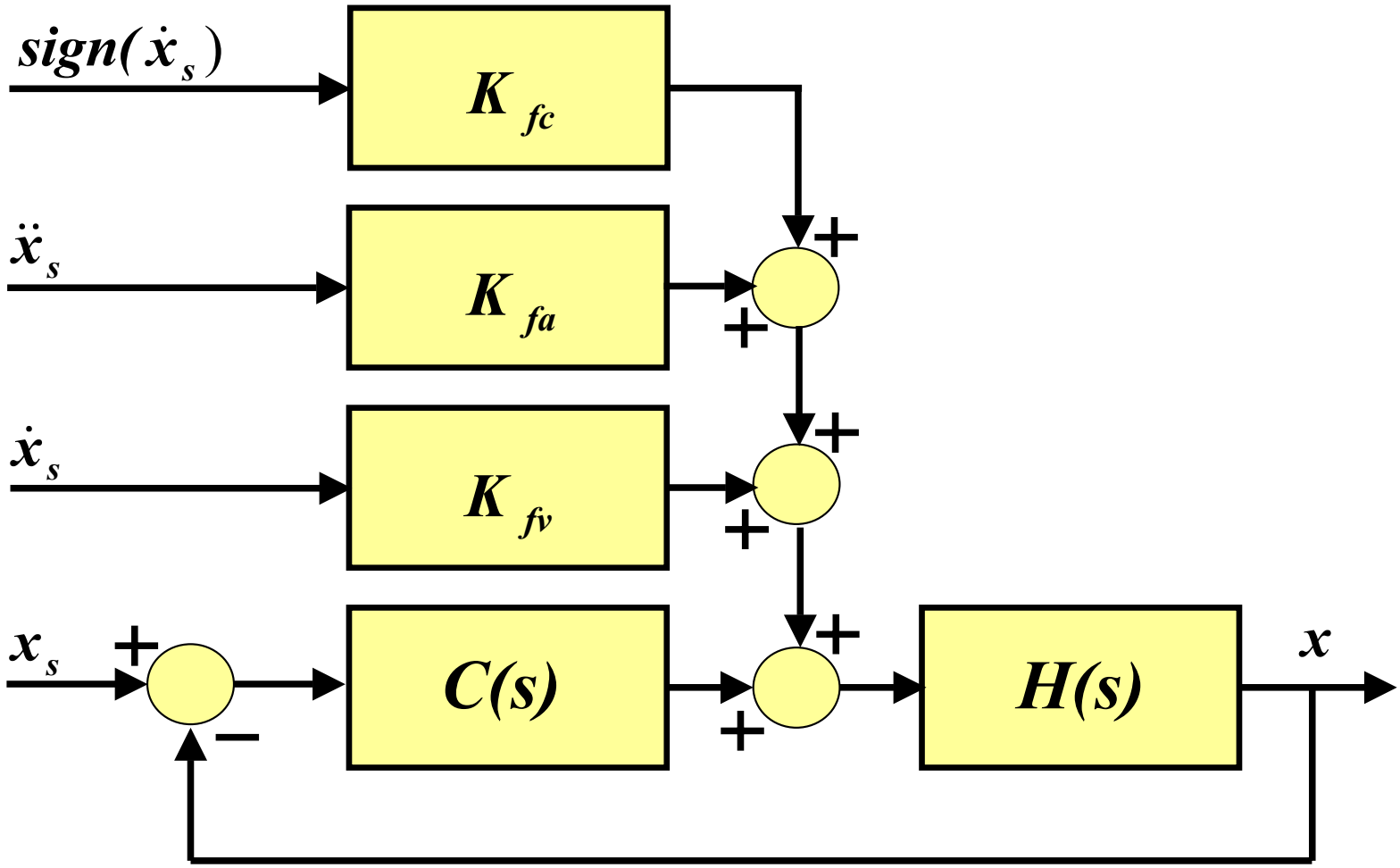
Hence, to overcome this force in advance, we need a feedforward

$$F_c = K_{fc} \cdot \text{sign}\left(\dot{x}_s\right)$$

General feedforward (2)



General feedforward (2)

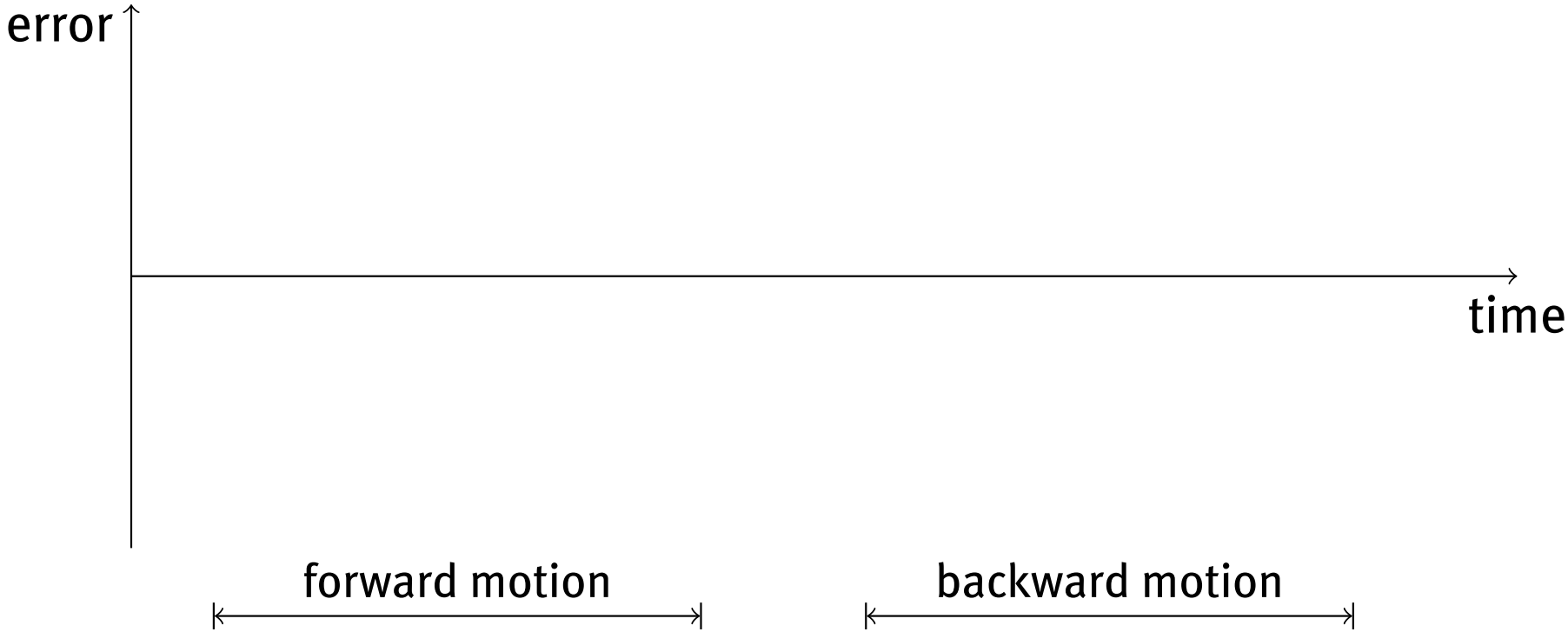


$$F_{ff} = K_{fa}\ddot{x}_s + K_{fv}\dot{x}_s + K_{fc}\,\mathrm{sign}\,(\dot{x}_s)$$

Tuning feedforward parameters

Error profile

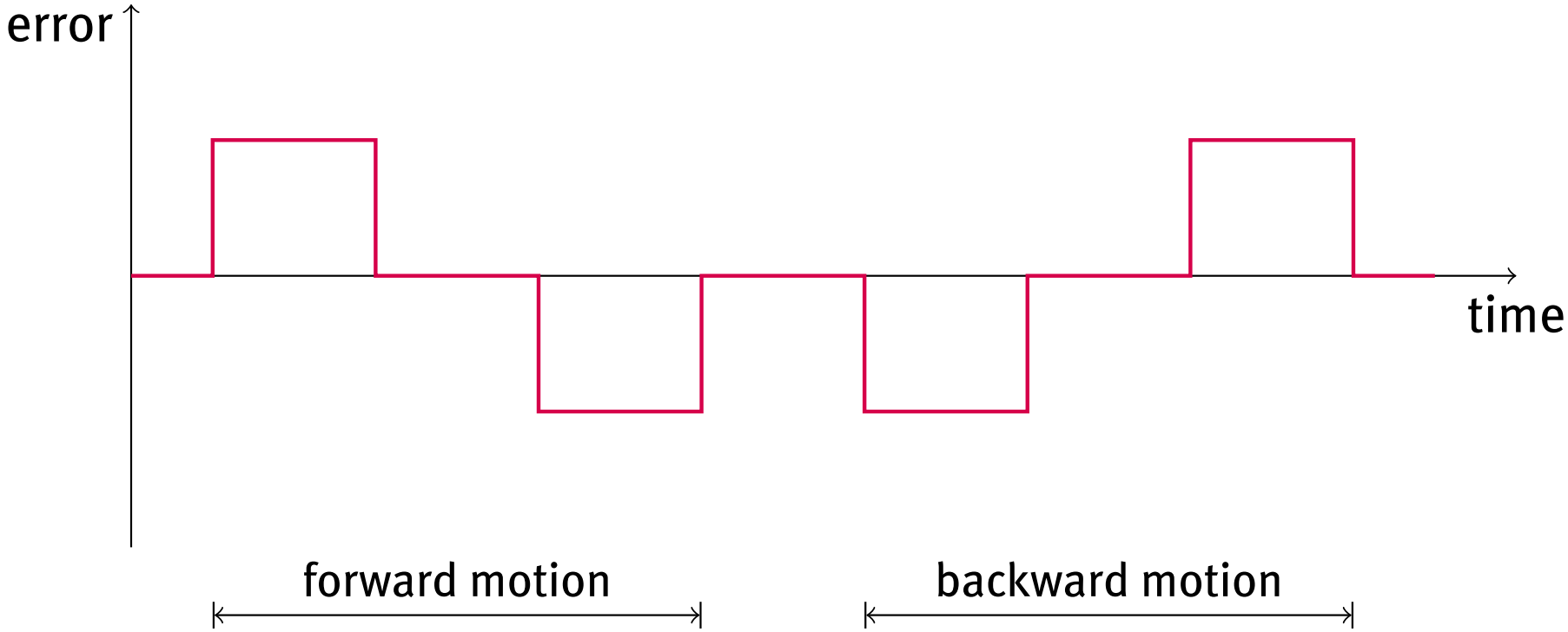
Feedforward tuning is based on error profile in time domain



There is an error due to:

Error profile

Feedforward tuning is based on error profile in time domain

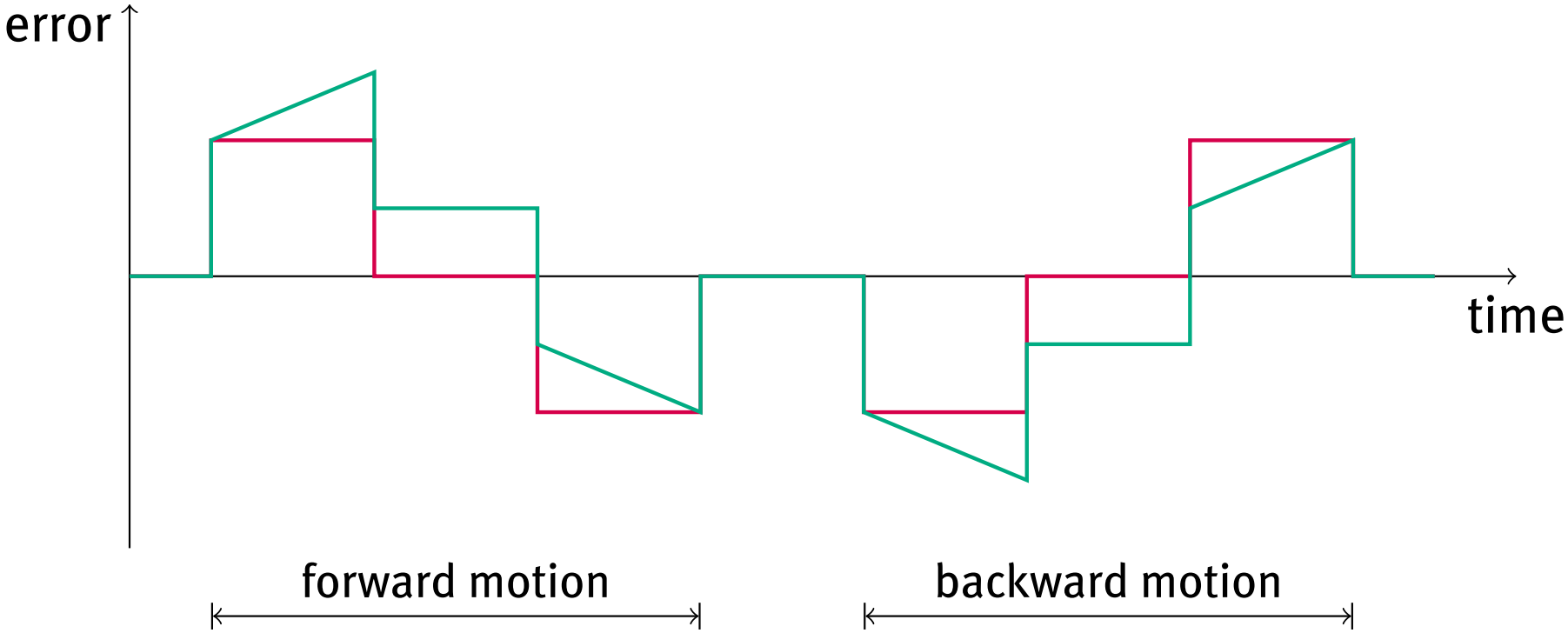


There is an error due to:

- lack of force to accelerate (K_{fa})

Error profile

Feedforward tuning is based on error profile in time domain

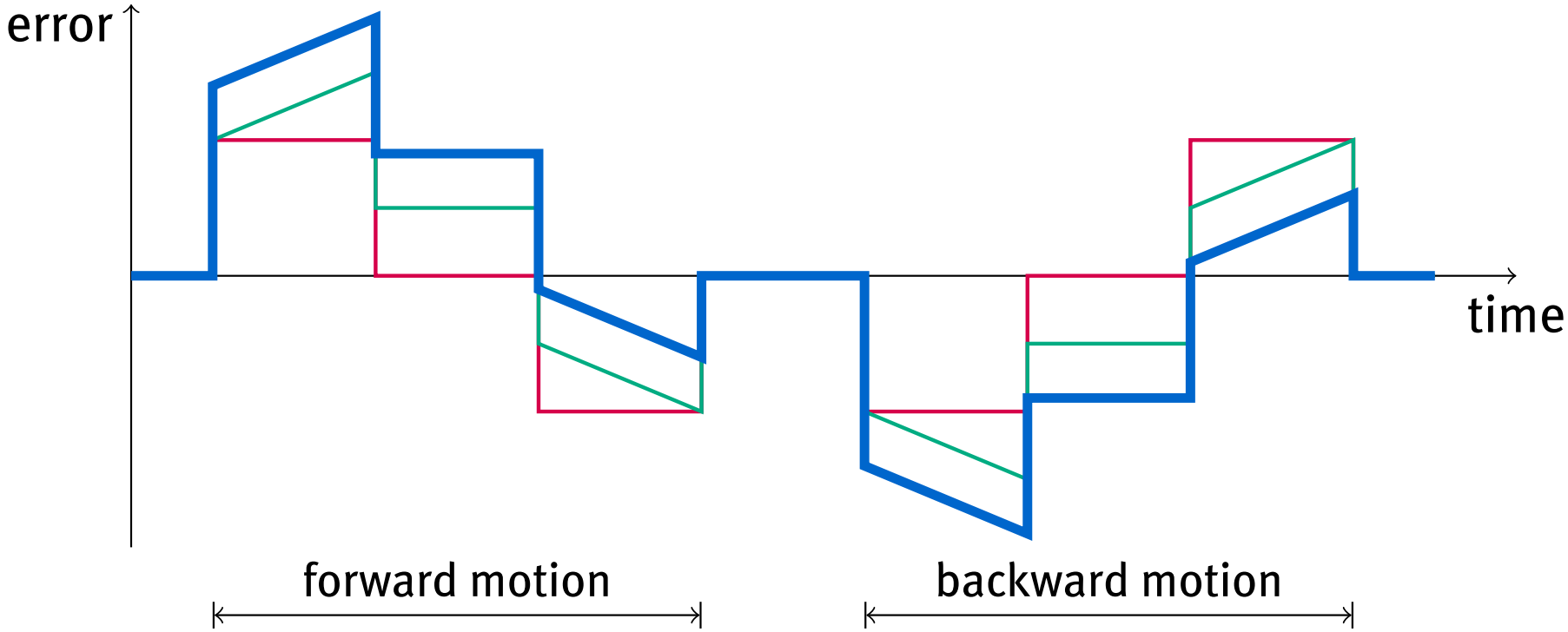


There is an error due to:

- lack of force to accelerate (K_{fa})
- lack of force to compensate viscous friction (K_{fv})

Error profile

Feedforward tuning is based on error profile in time domain



There is an error due to:

- lack of force to accelerate (K_{fa})
- lack of force to compensate viscous friction (K_{fv})
- lack of force to compensate Coulomb friction (K_{fc})

Feedforward tuning procedure

- Define a suitable (2nd or higher order) setpoint trajectory
  - both forward and backward movement (with a small pause)
  - rather ‘challenging’ (it must generate some error profile)
  - approx 1/3 acceleration, 1/3 constant velocity, 1/3 deceleration

Feedforward tuning procedure

- Define a suitable (2nd or higher order) setpoint trajectory
  - both forward and backward movement (with a small pause)
  - rather ‘challenging’ (it must generate some error profile)
  - approx 1/3 acceleration, 1/3 constant velocity, 1/3 deceleration
- Monitor the error
  - the error should be large enough
  - each feedforward component should be recognizable
  - maybe feedback bandwidth should be lowered?

Feedforward tuning procedure

- Define a suitable (2nd or higher order) setpoint trajectory
  - both forward and backward movement (with a small pause)
  - rather ‘challenging’ (it must generate some error profile)
  - approx 1/3 acceleration, 1/3 constant velocity, 1/3 deceleration
- Monitor the error
  - the error should be large enough
  - each feedforward component should be recognizable
  - maybe feedback bandwidth should be lowered?
- Tune K_{fc}
  - deceleration forward movement and acceleration backward movement should have same error

Feedforward tuning procedure

- Define a suitable (2nd or higher order) setpoint trajectory
  - both forward and backward movement (with a small pause)
  - rather 'challenging' (it must generate some error profile)
  - approx 1/3 acceleration, 1/3 constant velocity, 1/3 deceleration
- Monitor the error
  - the error should be large enough
  - each feedforward component should be recognizable
  - maybe feedback bandwidth should be lowered?
- Tune K_{fc}
  - deceleration forward movement and acceleration backward movement should have same error
- Tune K_{fv}
  - remove slopes during acceleration and deceleration
  - no error during constant velocity phase when K_{fc} and K_{fv} are correct

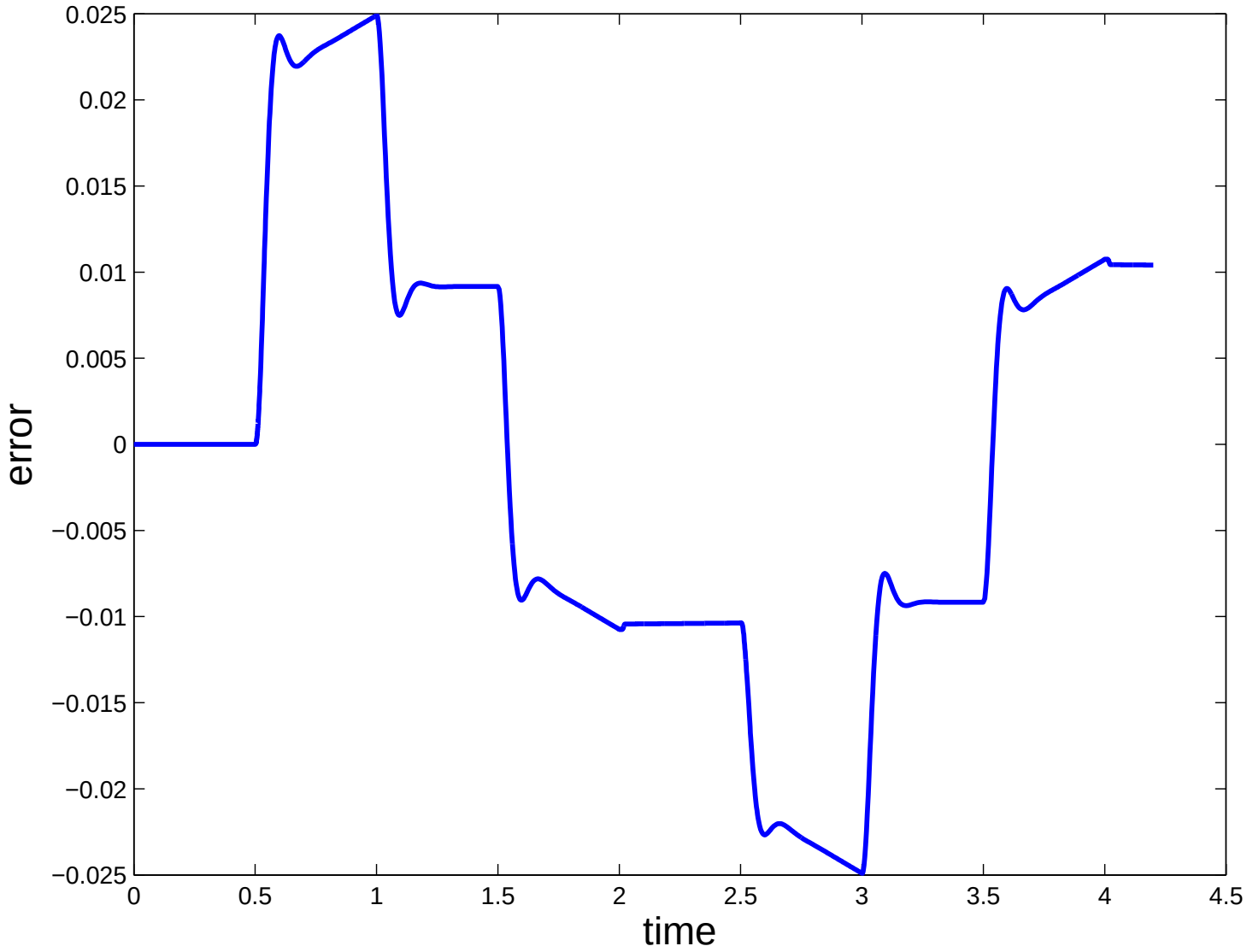
Feedforward tuning procedure

- Define a suitable (2nd or higher order) setpoint trajectory
  - both forward and backward movement (with a small pause)
  - rather 'challenging' (it must generate some error profile)
  - approx 1/3 acceleration, 1/3 constant velocity, 1/3 deceleration
- Monitor the error
  - the error should be large enough
  - each feedforward component should be recognizable
  - maybe feedback bandwidth should be lowered?
- Tune K_{fc}
  - deceleration forward movement and acceleration backward movement should have same error
- Tune K_{fv}
  - remove slopes during acceleration and deceleration
  - no error during constant velocity phase when K_{fc} and K_{fv} are correct
- Tune K_{fa}
  - minimize error during acceleration and deceleration

Feedforward tuning procedure

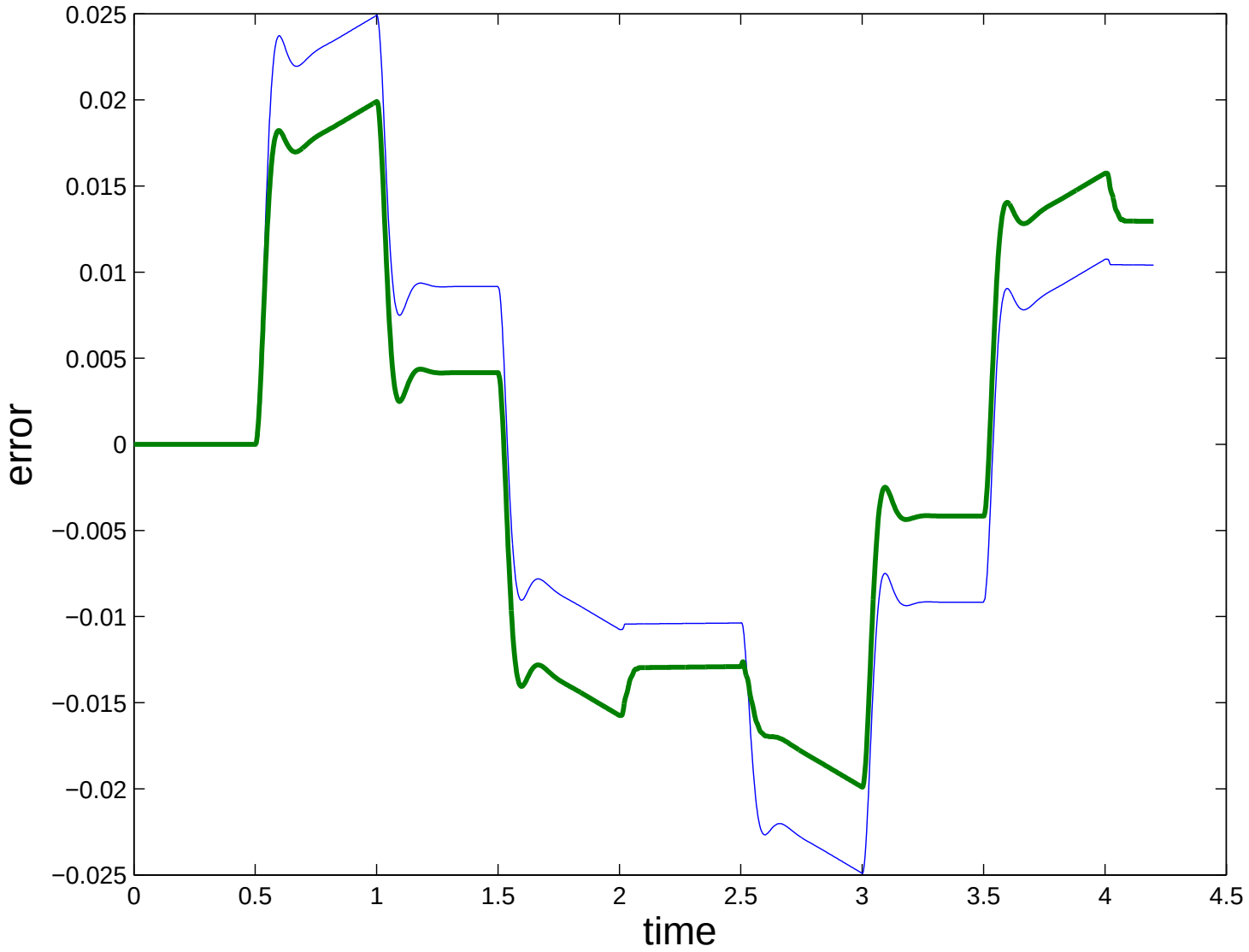
- Define a suitable (2nd or higher order) setpoint trajectory
  - both forward and backward movement (with a small pause)
  - rather 'challenging' (it must generate some error profile)
  - approx 1/3 acceleration, 1/3 constant velocity, 1/3 deceleration
- Monitor the error
  - the error should be large enough
  - each feedforward component should be recognizable
  - maybe feedback bandwidth should be lowered?
- Tune K_{fc}
  - deceleration forward movement and acceleration backward movement should have same error
- Tune K_{fv}
  - remove slopes during acceleration and deceleration
  - no error during constant velocity phase when K_{fc} and K_{fv} are correct
- Tune K_{fa}
  - minimize error during acceleration and deceleration
- If desired, tune up the feedback controller

Feedforward tuning



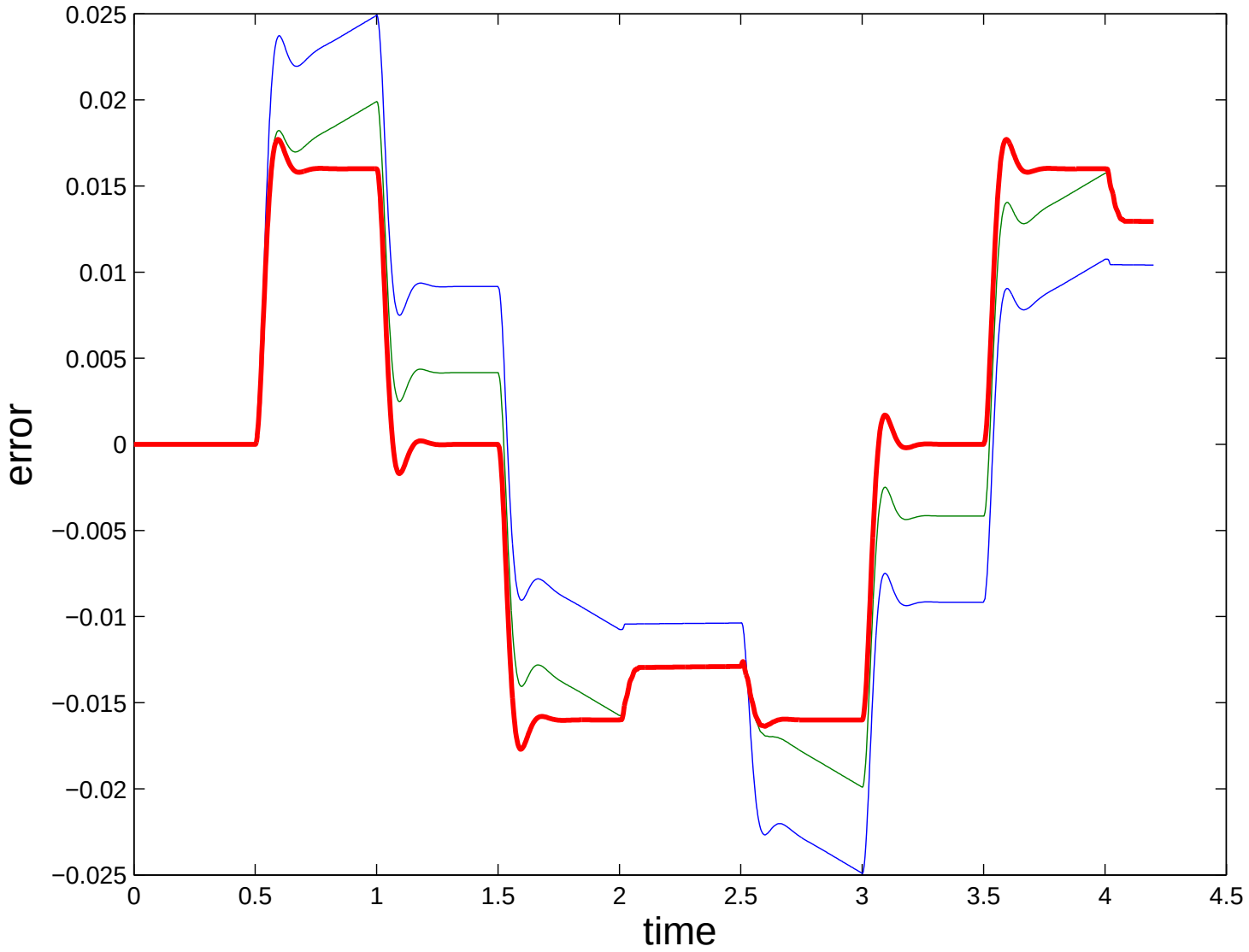
- No FF

Feedforward tuning



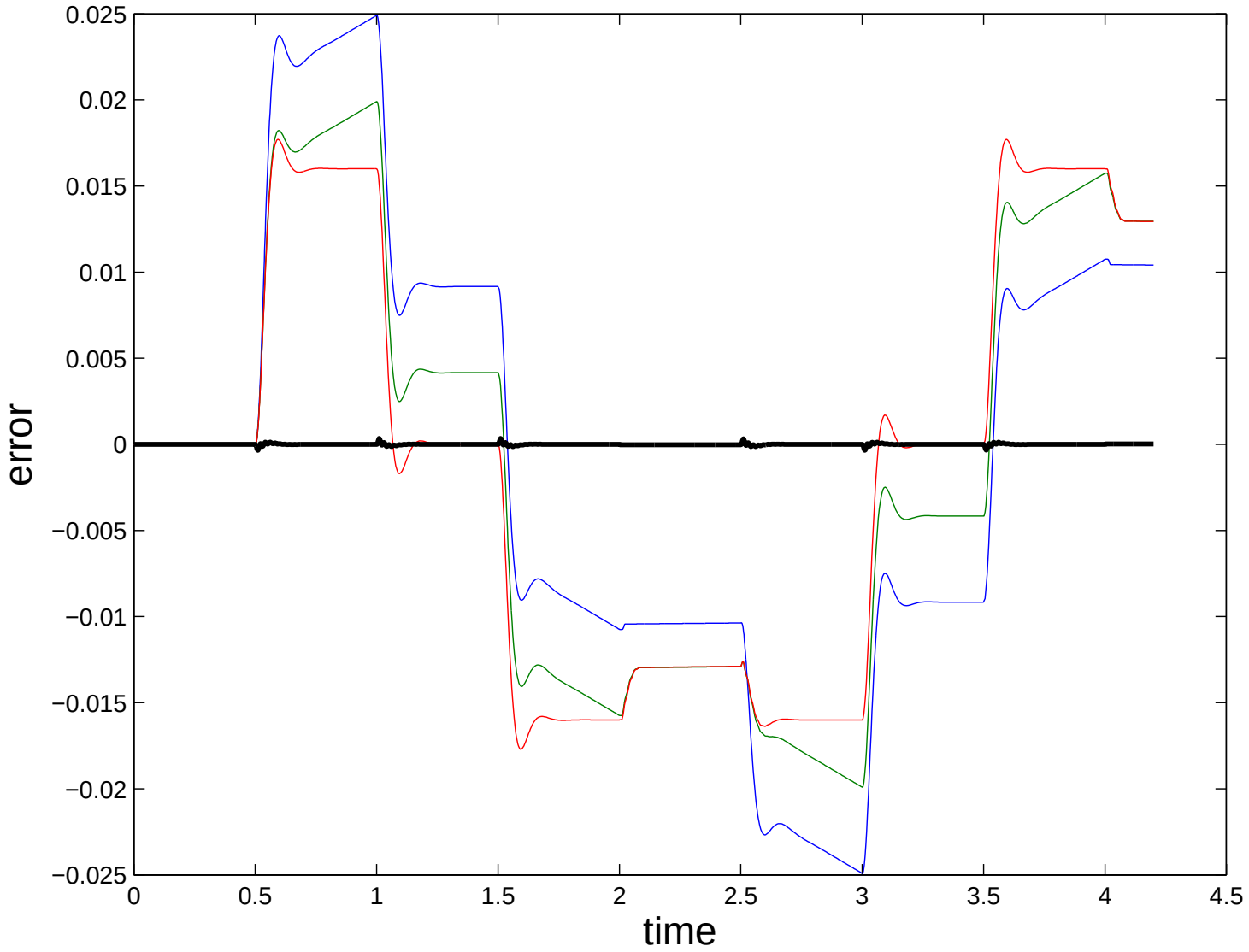
- No FF
- K_{fc} tuned

Feedforward tuning



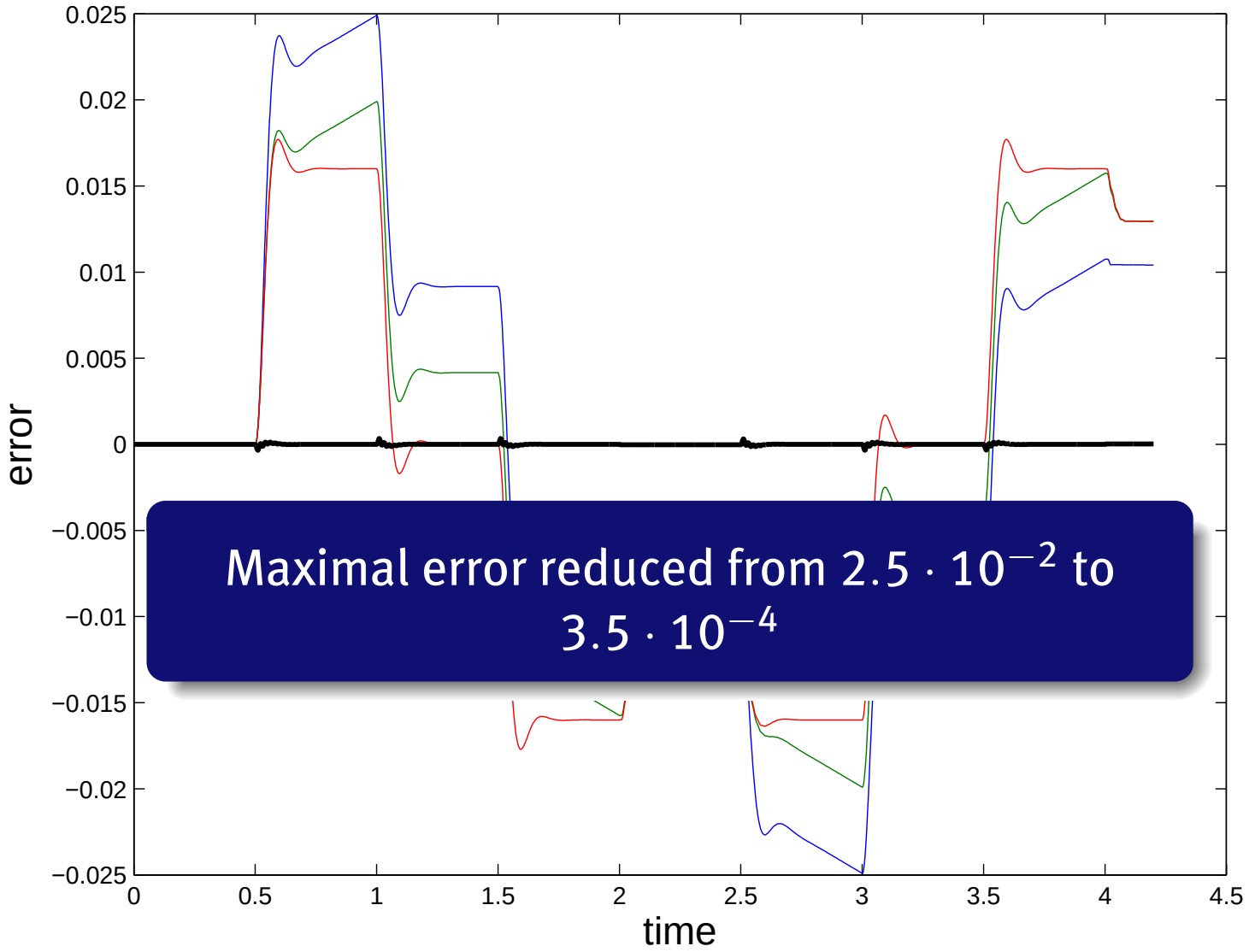
- No FF
- K_{fc} tuned
- K_{fv} tuned

Feedforward tuning



- No FF
- K_{fc} tuned
- K_{fv} tuned
- K_{fa} tuned

Feedforward tuning

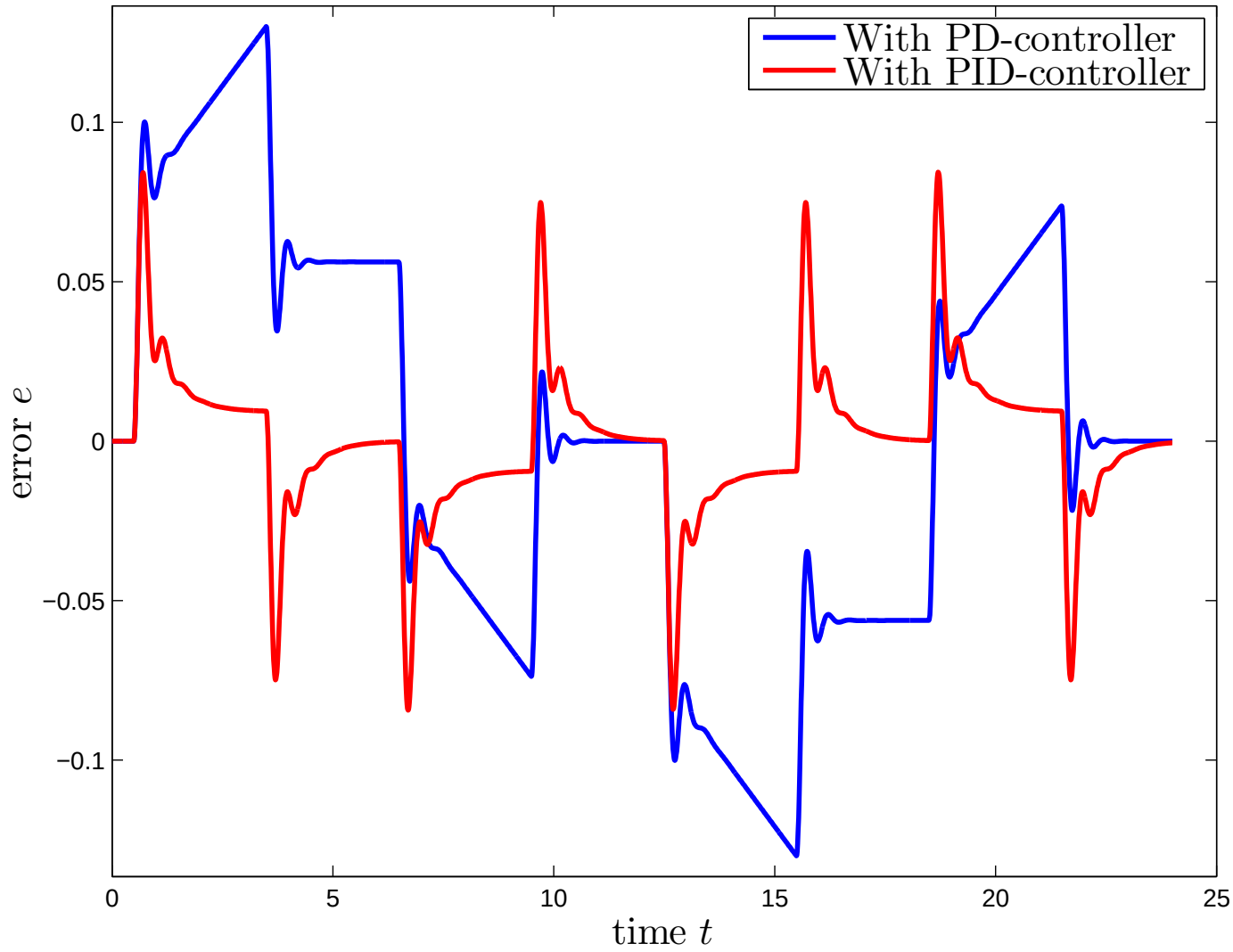


Maximal error reduced from $2.5 \cdot 10^{-2}$ to $3.5 \cdot 10^{-4}$

- No FF
- K_{fc} tuned
- K_{fv} tuned
- K_{fa} tuned

Integral controller action

To tune feedforward, don't use a high performance feedback controller!



Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

- Stability
  - Can a stable closed loop be destabilized by feedforward?
  - Can an unstable closed loop be stabilized by feedforward?

Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

- Stability
  - Can a stable closed loop be destabilized by feedforward? No.
  - Can an unstable closed loop be stabilized by feedforward? No.

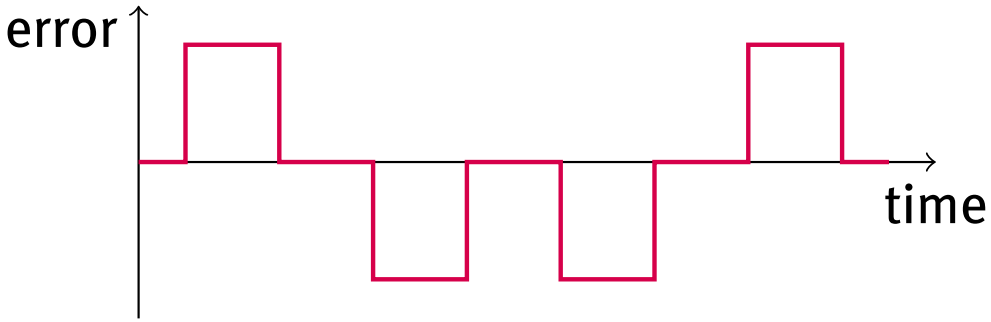
  $\Rightarrow$ Feedforward is open loop, in parallel with closed loop

Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

- Stability
  - Can a stable closed loop be destabilized by feedforward? No.
  - Can an unstable closed loop be stabilized by feedforward? No.

  $\Rightarrow$ Feedforward is open loop, in parallel with closed loop
- Over-tuning
  - What happens if we over-tune feedforward parameters?



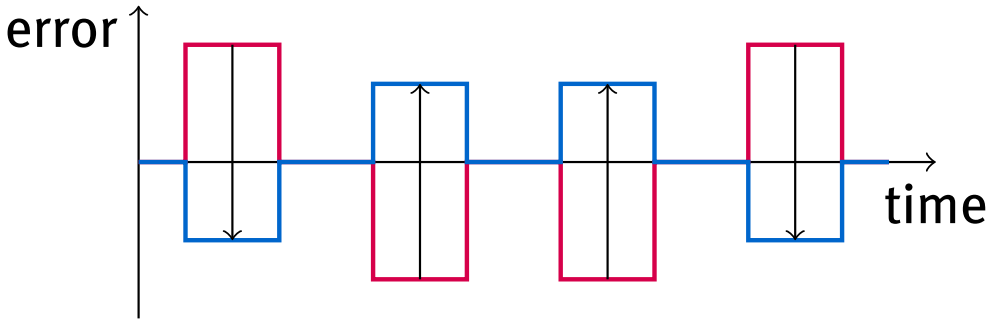
Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

- Stability
  - Can a stable closed loop be destabilized by feedforward? No.
  - Can an unstable closed loop be stabilized by feedforward? No.

  $\Rightarrow$ Feedforward is open loop, in parallel with closed loop
- Over-tuning
  - What happens if we over-tune feedforward parameters?

  $\Rightarrow$ Over-compensation of required forces reverses the error



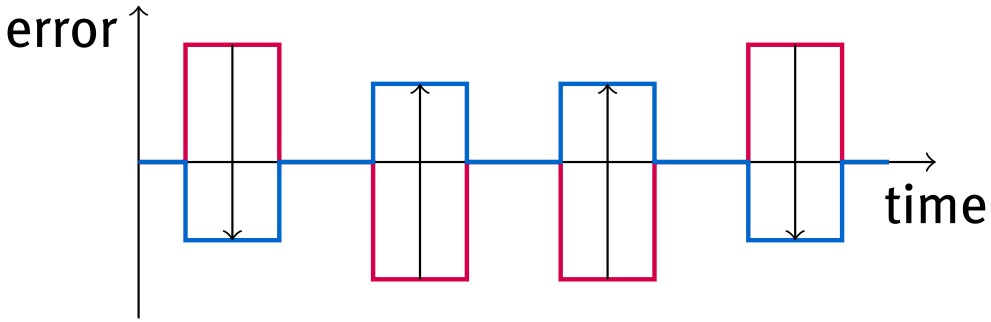
Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

- Stability
  - Can a stable closed loop be destabilized by feedforward? No.
  - Can an unstable closed loop be stabilized by feedforward? No.

  $\Rightarrow$ Feedforward is open loop, in parallel with closed loop
- Over-tuning
  - What happens if we over-tune feedforward parameters?

  $\Rightarrow$ Over-compensation of required forces reverses the error



- Trajectory dependence
  - Do the feedforward terms change when the trajectory changes?

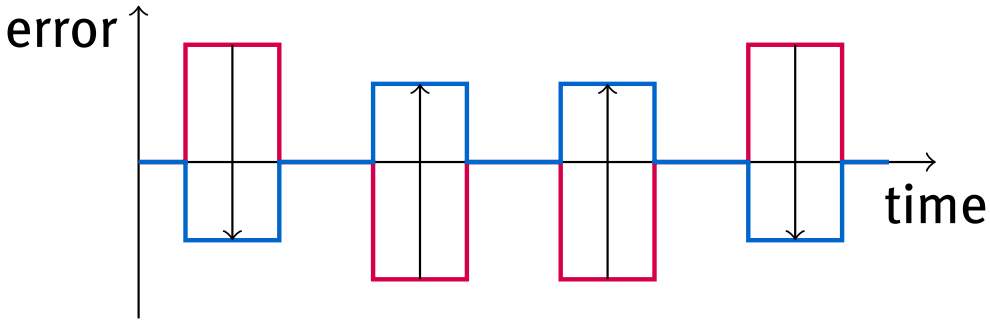
Feedforward issues to be aware of

Feedforward tuning is time domain trial and error.

- Stability
  - Can a stable closed loop be destabilized by feedforward? No.
  - Can an unstable closed loop be stabilized by feedforward? No.

  $\Rightarrow$ Feedforward is open loop, in parallel with closed loop
- Over-tuning
  - What happens if we over-tune feedforward parameters?

  $\Rightarrow$ Over-compensation of required forces reverses the error



- Trajectory dependence
  - Do the feedforward terms change when the trajectory changes? No.

  $\Rightarrow$ Terms represent system parameters, which do not change

Smoothing feedforward

Second order trajectory may not be sufficient

- jump in $\ddot{x}_s \Rightarrow$ jump in feedforward force
- can excite parasitic (higher order) dynamics
- can introduce oscillations
- can cause actuator saturation

Smoothing feedforward

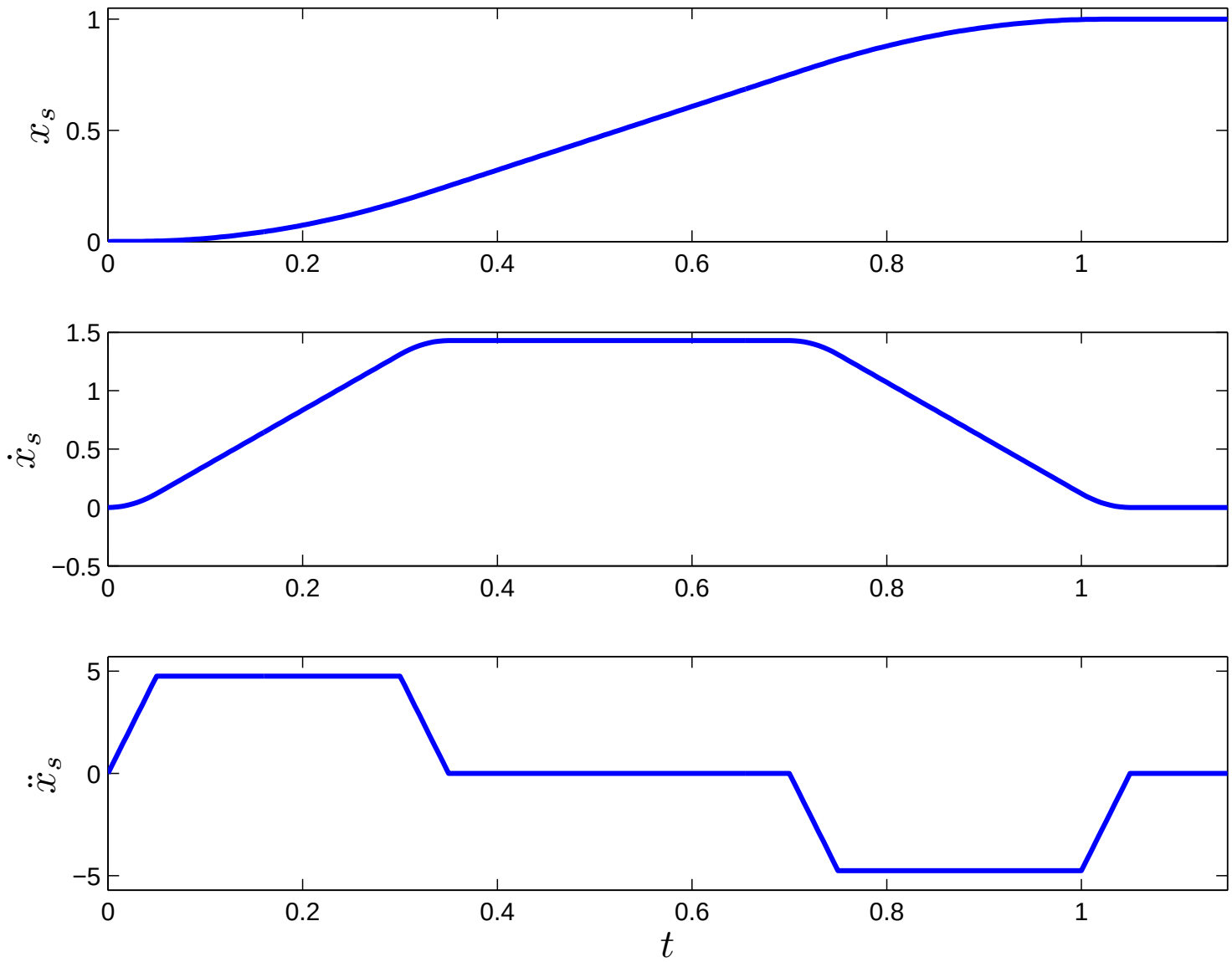
Second order trajectory may not be sufficient

- jump in $\ddot{x}_s \Rightarrow$ jump in feedforward force
- can excite parasitic (higher order) dynamics
- can introduce oscillations
- can cause actuator saturation

How to solve this?

- Higher order setpoints
  - gradual increase of feedforward forces (smoothing)
  - by limiting higher derivatives (e.g. jerk)
- Here: using 3rd order setpoints

3rd order setpoint



Higher order setpoints

$$\begin{aligned} \textit{Momentum} &= \textit{mass} \times \textit{velocity} \\ \textit{Force} &= \textit{mass} \times \textit{acceleration} \\ \textit{Yank} &= \textit{mass} \times \textit{jerk} \\ \textit{Tug} &= \textit{mass} \times \textit{snap} \\ \textit{Snatch} &= \textit{mass} \times \textit{crackle} \\ \textit{Shake} &= \textit{mass} \times \textit{pop} \end{aligned}$$

More advanced methods

Other more advanced techniques:

- Model-based input shaping
- Learning input shaping
  - adapt the trajectory after each trial
- Iterative Learning Control (ILC)
  - adapt the feedforward signal after each trial
  - frequency domain, lifted domain, or using basis functions

See other courses (4SC070: Learning Control) and current research.