Dynamics and motion control

Lecture 5

Model following control -servo design

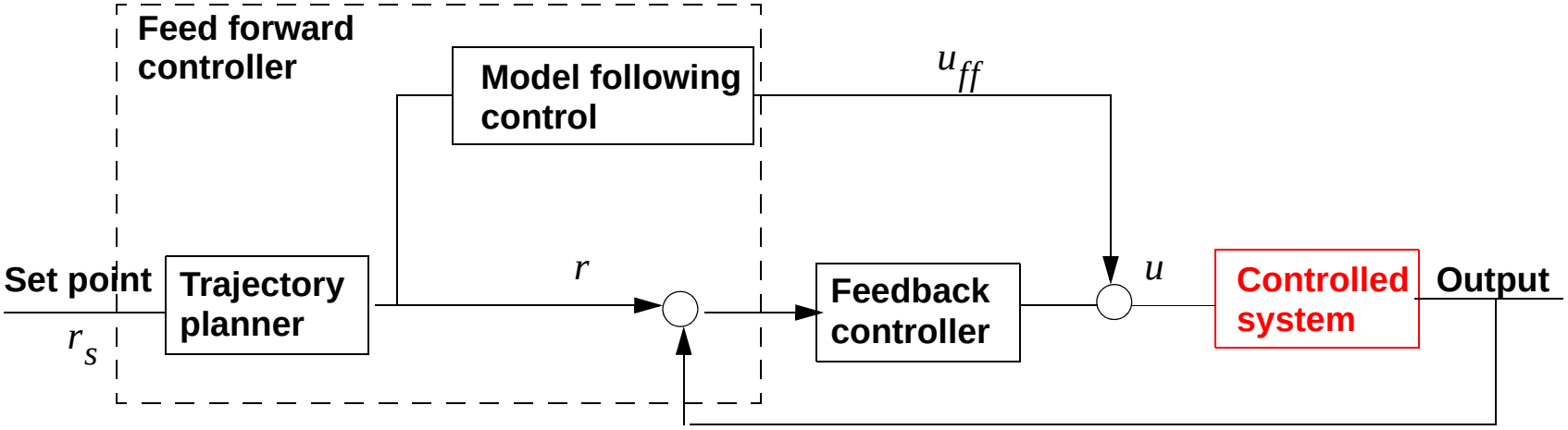
Jan Wikander,Bengt Eriksson
KTH, Machine Design
Mechatronics Lab
e-mail: benke@md.kth.se

5.1. Lecture outline

- **1. Introduction**
- 2. Transfer function based model following
- 3. Time domain based model following
- 4. An example
- 5. Matlab example for the dc motor

5.1.1. The servo problem

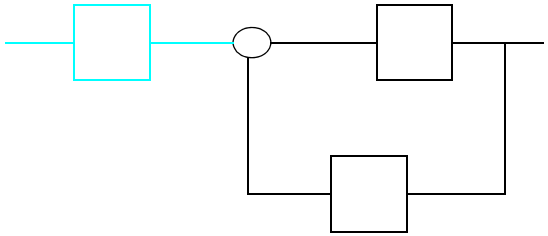
- **The servo problem, here with error feedback.**



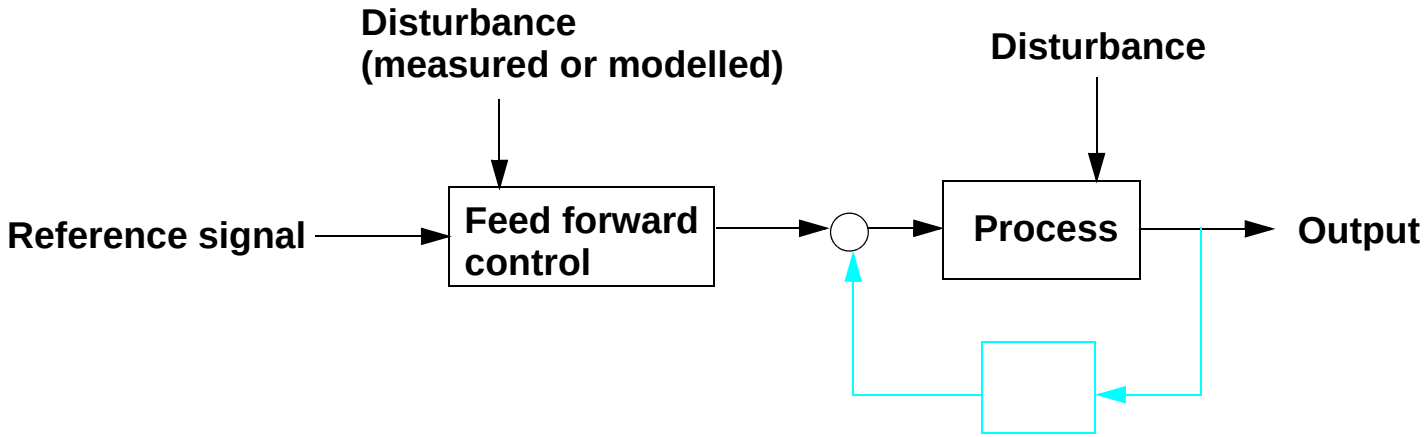
- Characterized by frequent changes in set point (e.g in an industrial robot)
- Control design mainly to follow (track) the reference (e.g. position or velocity, or position, velocity, acceleration and jerk).

5.1.2. Feedback control properties

- The main principle in control engineering
- Typically model based (but not required to be)
- Produces control signals after an error has occurred
- Disturbance rejection is achieved
- Effect of process parameter variations is reduced
- Leads to a closed loop
- Sensor noise may be amplified and deteriorate performance
- May lead to instability if designed incorrectly

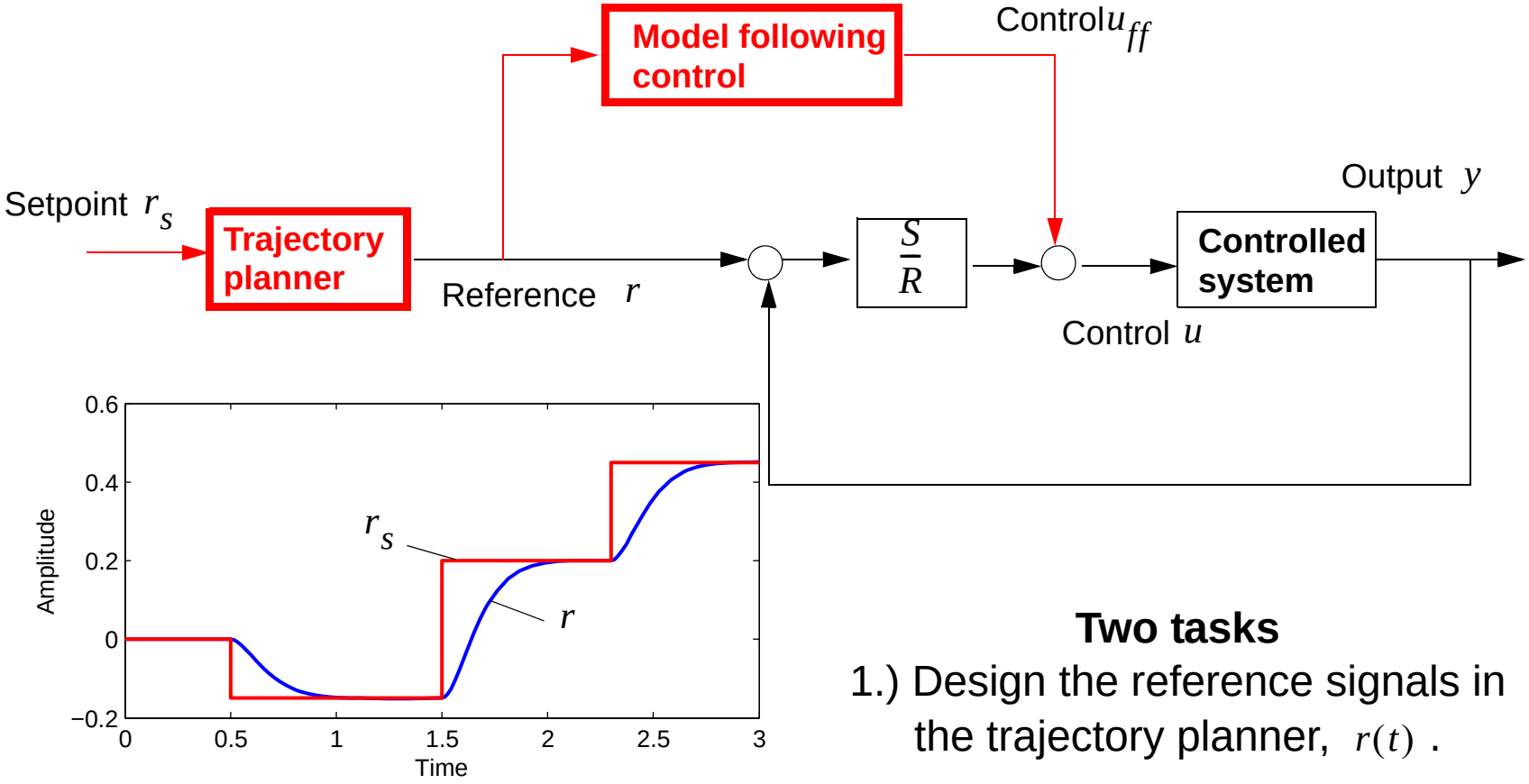


5.1.3. Feed forward control properties



- Model based (but fairly simple models typically helps a lot)
- Produces control signals **before** error has occurred
- Uses measured or modelled disturbance and compensates for it
- Uses carefully designed reference signals to make the process follow the references “exactly” and without saturating the control signal.

5.1.4. The model based control concept

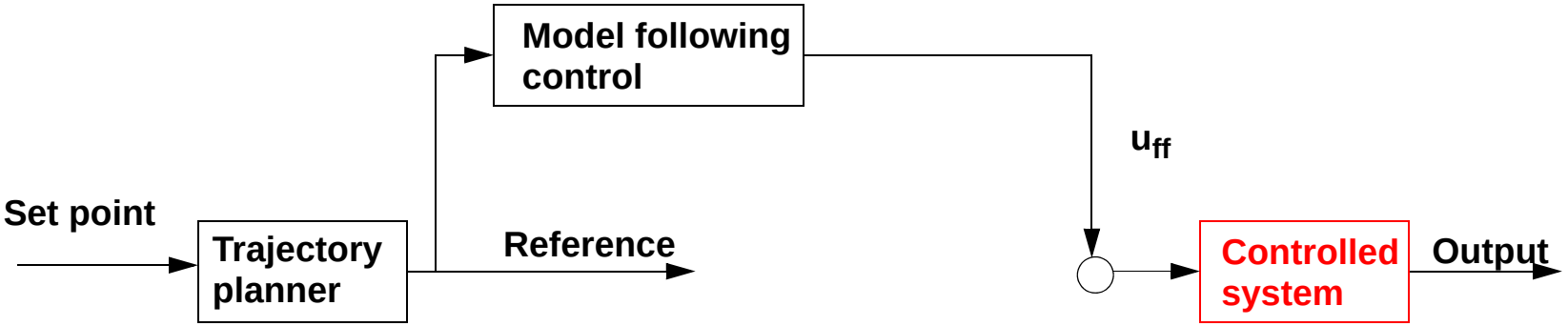


Two tasks

1.) Design the reference signals in the trajectory planner, $r(t)$.

2.) Design a model following controller, $u_{ff}(t)$.

5.1.5. The ultimate goal, Exact model following

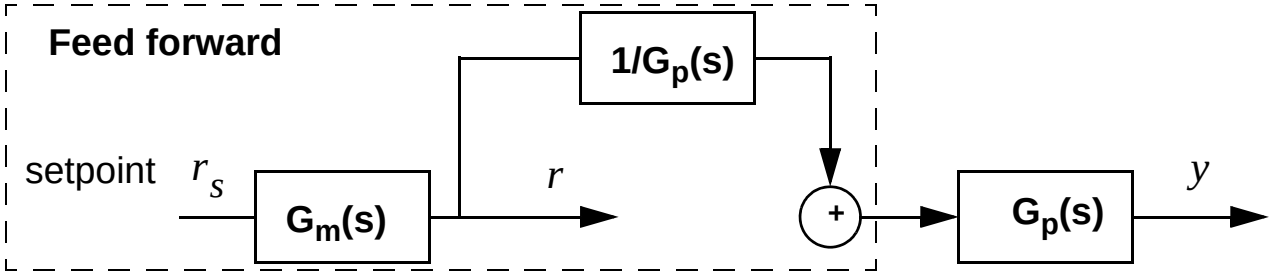


- Design the trajectory planner and the model following controller such that the feed forward input signal $u_{ff}(t)$ drives the output to the desired setpoint
- Note that the model following control part can be a nonlinear system
- Two main concepts, Transfer function and Time domain.

5.2. Lecture outline

- 1. Introduction
- **2. Transfer function based model following**
- 3. Time domain based model following
- 4. An example
- 5. Matlab example for the dc motor

5.2.1. Model following TF-based



- $G_m(s)$ is the reference model, i.e it is designed to provide reasonable references to the controller, and to make G_m / G_p proper.
- $G_p(s)$ is the transfer function of the process. No feedback controller yet!
- Ideally the inverse of the process, i.e. $1/G_p(s)$, would provide exact model following: $1/G_p(s) * G_p(s) = 1$, and the closed loop $y / r_s = G_m(s)$
- Clearly, the model following concept is directly dependent on the accuracy of the process model.

5.2.2. The reference model $G_m(s)$

- The task of the reference model is to generate smooth references that the process is able to follow.
- To implement complete model following, i.e a full process inverse, we must require that the pole excess d_m of the reference model is larger than or equal to the pole excess d of the process, i.e if

$$G_p(s) = \frac{B(s)}{A(s)}$$

is the pole excess of the process d_p

$$d_p = deg(A) - deg(B)$$

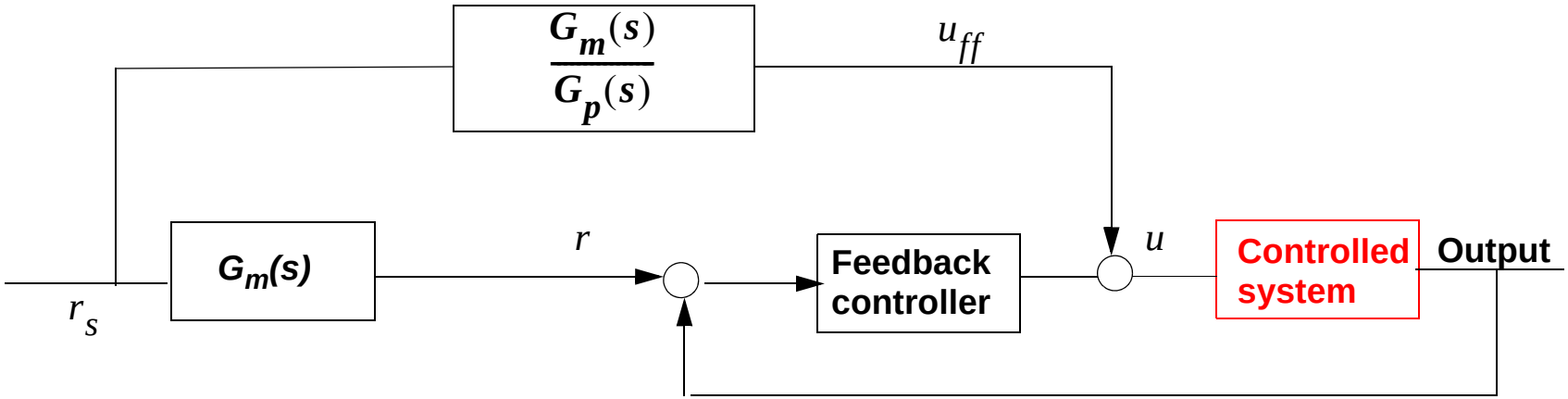
and finally the requirement

$$d_m \geq d_p \qquad G_m(s) = \frac{B_m(s)}{A_m(s)} \qquad d_m = deg(A_m) - deg(B_m)$$

5.2.3. Model following TF-based cont.

- Since exact model following involves taking the inverse of the process model, the inverse must be possible to implement. I.e. the inverse must constitute a stable dynamic system.
- A stable dynamic system has no right half plane (RHP) poles. This means for model following that the process must not have any RHP zeros.
- Care must be taken when doing the discrete time implementation because, as we know, a continuous time system with only LHP zeros may become zeros outside the unit circle when transformed into discrete time.
  This often happens for very low sampling periods.

5.2.4. Implementing the TF based servo control



5.3. Lecture outline

- 1. Introduction
- 2. Transfer function based model following
- **3. Time domain based model following**
- 4. An example
- 5. Matlab example for the dc motor

5.3.1. Model following Time domain (TD)

- Instead of the combined trajectory and model following transfer function $\frac{G_m(s)}{G_p(s)}$ are the trajectory and model following implemented separately and based on trajectories in time.
- Model following: If the inverse model (from *y* -> *u*) can be written as a function of the output and n-times the derivatives of the output (n:th order model).

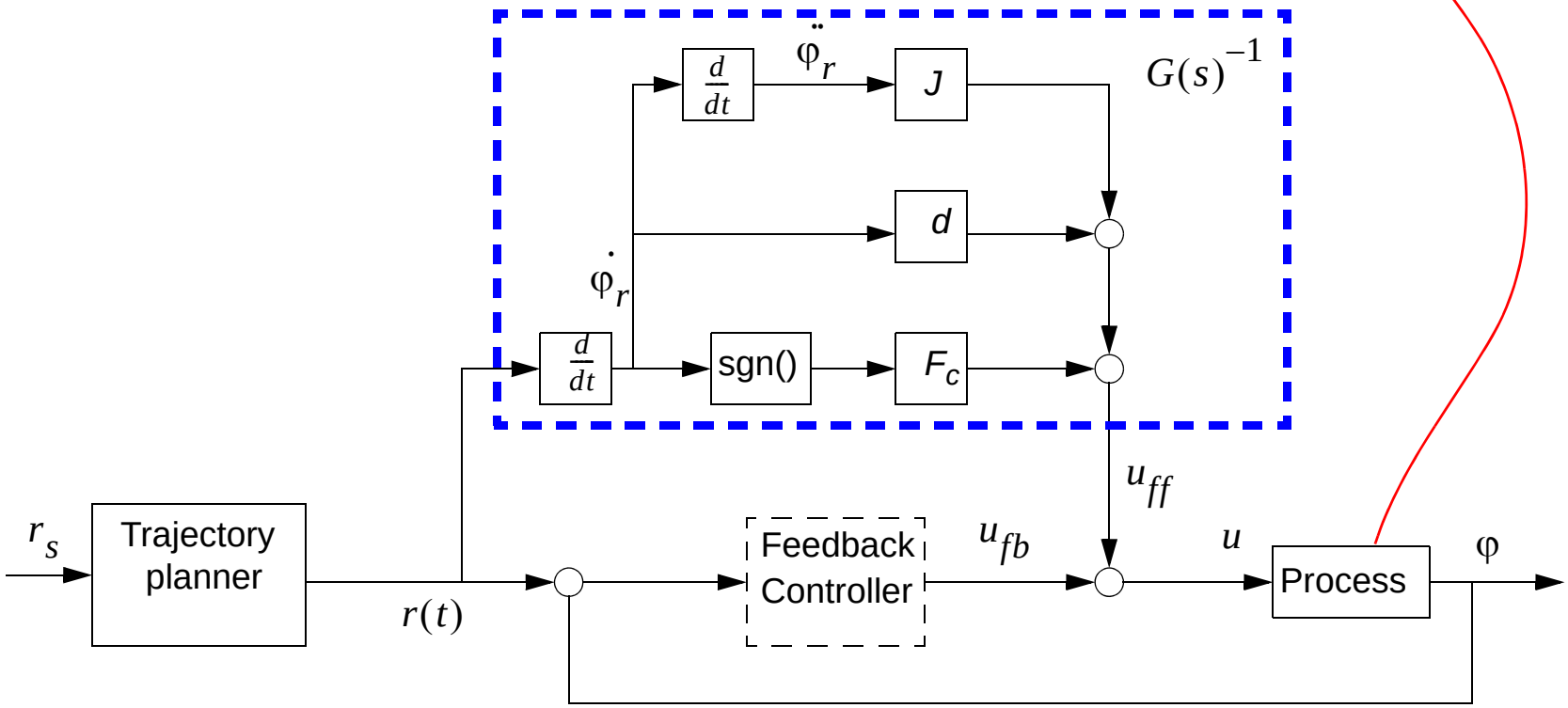
$$u(t) = f\left(y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, \ldots, \frac{d^ny}{dt^n}\right)$$

- Trajectory planner: Design $y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, \ldots, \frac{d^ny}{dt^n}$ based on closed loop specifications and process limitations, e.g., saturation.

5.3.2. The concept, an example

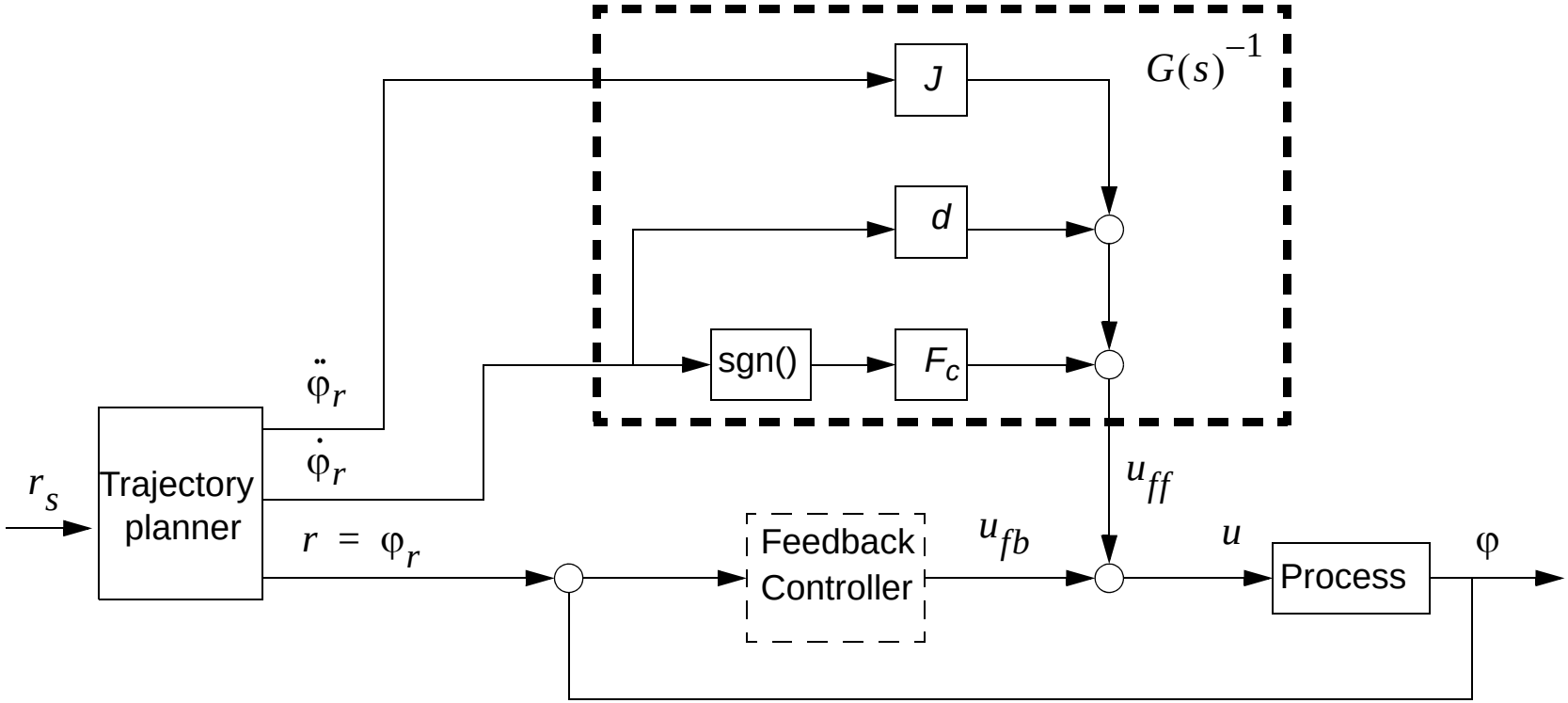
Process model DC-motor with non-linear friction $J\ddot{\varphi} = u - d\dot{\varphi} - F_c \mathrm{sgn}(\dot{\varphi})$.

Gives the feed forward control, $u_{ff}(t) = J\ddot{\varphi}_r(t) + d\dot{\varphi}_r + F_c \mathrm{sgn}(\dot{\varphi}_r)$.



5.3.3. Practical implementation

- **We want to avoid to take the derivative of signals.**
  -> The Trajectory planner must provide the derivatives



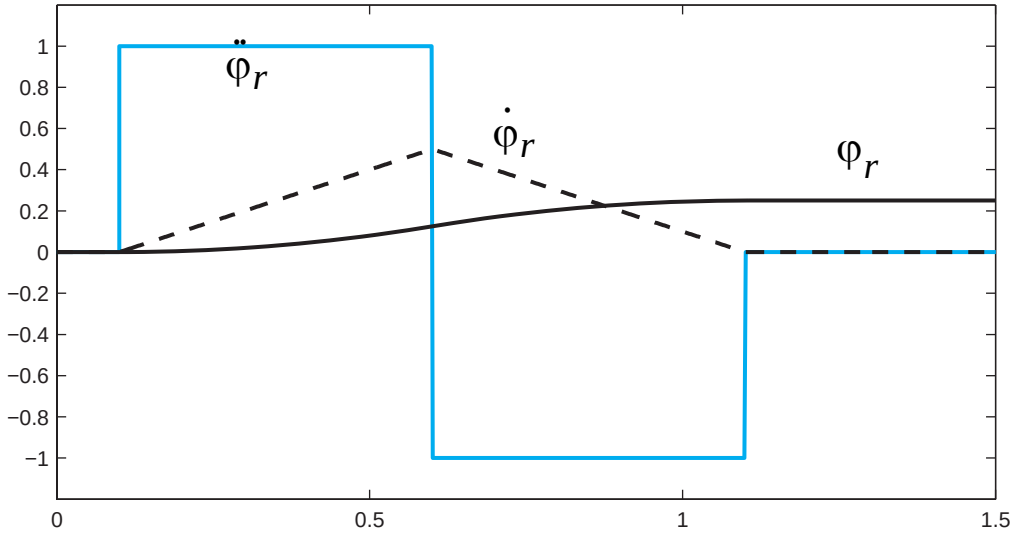
5.3.4. n-times differentiable reference signals

- **The DC-motor is a second order diff. eq. from input to position**

  - The reference position must be two times differentiable (velocity and acceleration)

  - Which is the same as: The acceleration reference must be finite

  ex.



5.3.5. How to choose the reference signals

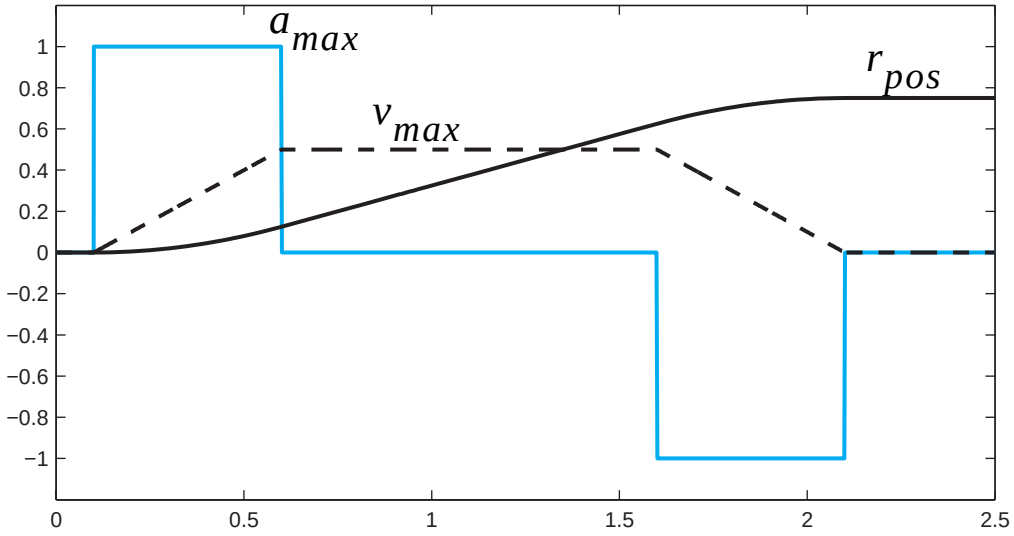
- **Fastest possible positioning (or specific time trajectories)**

  - Calculate max acceleration and velocity of the process, without saturating the input to the motor.

  - For the DC-motor: If max input torque is $\pm M_{max}$ then max acceleration at zero velocity is $a_{max} = (\pm M_{max})/J$.

    Max velocity at **zero** acceleration is $v_{max} = (\pm M_{sat})/(d + F_c)$

Example reference trajectories



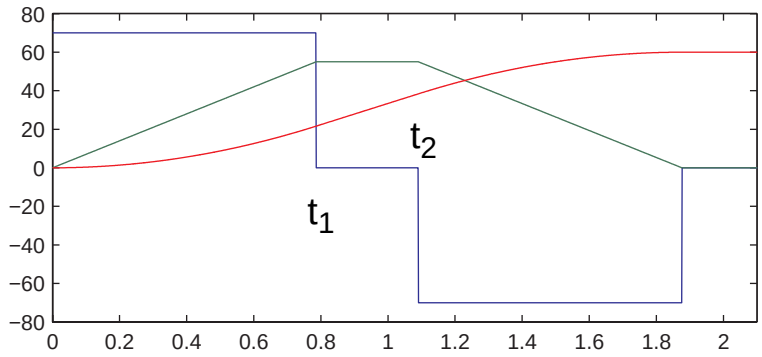
5.3.6. Example: how to calculate the ref. traj.

$v = at$, zero initial condition!

$t_1 = \frac{v_{max}}{a_{max}}$, the time when maximum velocity is reached using maximum acceleration

$y(t_1) = a_{max}\int_0^{t_1} t\,dt = \frac{1}{2}a_{max}t_1^2 = \frac{1}{2}\frac{v_{max}^2}{a_{max}}$, position reached at time t_1

$r_s - 2y(t_1) = v_{max}(t_2 - t_1)$ -> $t_2 = \frac{r_s}{v_{max}} - \frac{v_{max}}{a_{max}} + t_1$ is the latest time to start braking.



Example::

$r_s = 60,$
$v_{max} = 55$
$a_{max} = 70$

5.3.7. General remarks

- **The two concepts of servo control are often combined**
- **Some process models are inherently difficult to invert.**

  - Flexible links: Approximate the flexible model with a stiff model, the inertia and the friction of the process is still fed forward.

- **You need some margin to the theoretical maximum acceleration and speeds.**
  - delays in computer implementation, flexibility.
  - Maximum torque is a function of motor speed.

5.4. Lecture outline

- 1. Introduction
- 2. Transfer function based model following
- 3. Time domain based model following
- **4. An example**
- 5. Matlab example for the dc motor

5.4.1. Model following control example I

- Consider a second order motion control system. The output is position. The electrical motor (the coil inductance is neglected) is operating against a linear spring with spring constant K

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\frac{K_f}{J} & -\left\langle \frac{K_{emk} \cdot K_m}{J} \right\rangle \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_m}{J} \end{bmatrix} \cdot u \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Or in transfer function form

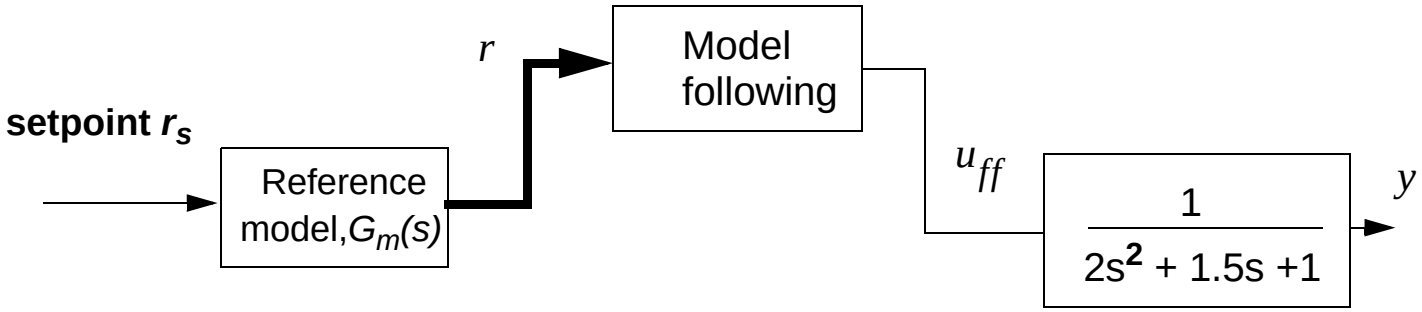
$$G_p(s) = \frac{K_m / J}{s^2 + s\left(\frac{K_m \cdot K_{emk}}{J}\right) + \frac{K_f}{J}}$$

5.4.2. Model following control example II

- Assuming

$$G_p(s) = \frac{1}{2s^2 + 1.5s + 1}$$

- We get the following principle model following control scheme



- Task: Design the reference model and the model following controller

5.4.3. Implementing the model following

- **The perfect model following is the inverse of the model** $G_m(s) = 1$

$$\frac{G_m(s)}{G_p(s)} = \frac{u_{ff}(s)}{r_s(s)} = \frac{2s^2 + 15s + 1}{1}$$ (not proper)

- **Use a reference model with pole excess dm > 2 (version 1)**

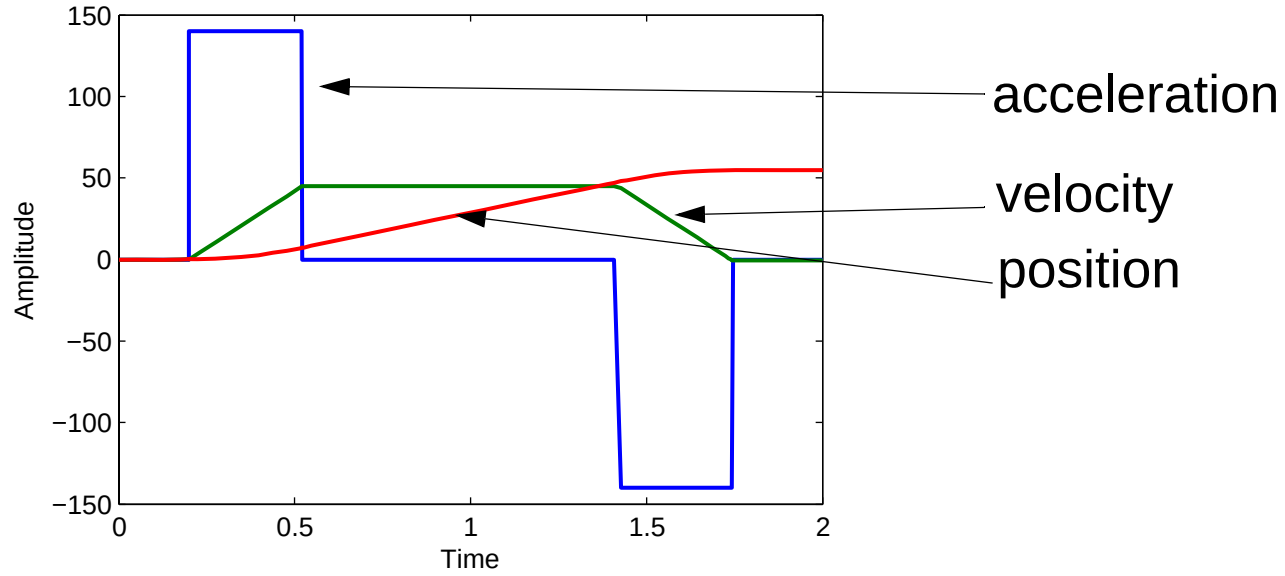
$$G_m(s) = \omega^2/(s^2 + 2\xi\omega s + \omega^2)$$

$$\frac{u_{ff}(s)}{r_s(s)} = \frac{(2s^2 + 15s + 1)\omega^2}{s^2 + 2\xi\omega s + \omega^2}$$ (proper)

- **Or use a trajectory planner, position, velocity and acceleration (version 2)**

$$u_{ff}(s) = (2s^2 + 15s + 1)r_s(s)$$
$$u_{ff}(t) = 2\ddot{r}_s(t) + 15\dot{r}_s(t) + r_s(t)$$

5.4.4. Output from the trajectory planner

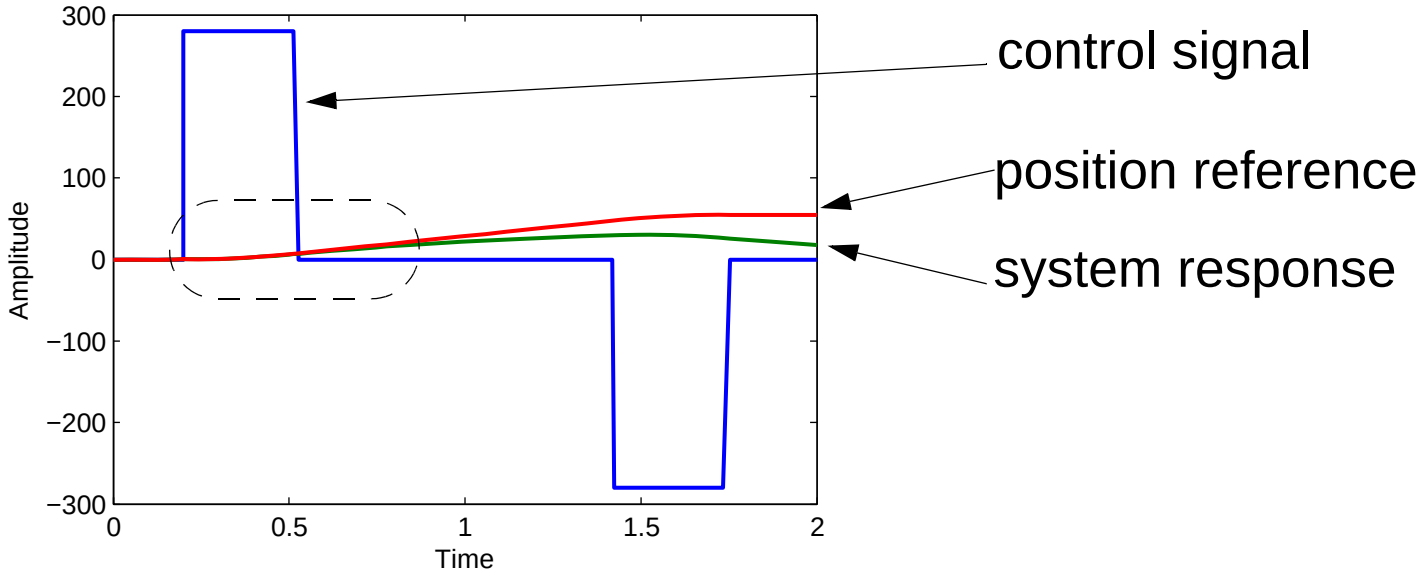


- The reference model provides acceleration, velocity and position references (acceleration step, velocity ramp and a smooth position reference)

$$r_{vel} = \frac{1}{t} r_{acc} \qquad r_{pos} = \frac{1}{t} r_{vel} = \frac{1}{t^2} r_{acc}$$

Observe, initial conditions

5.4.5. Acceleration feed forward

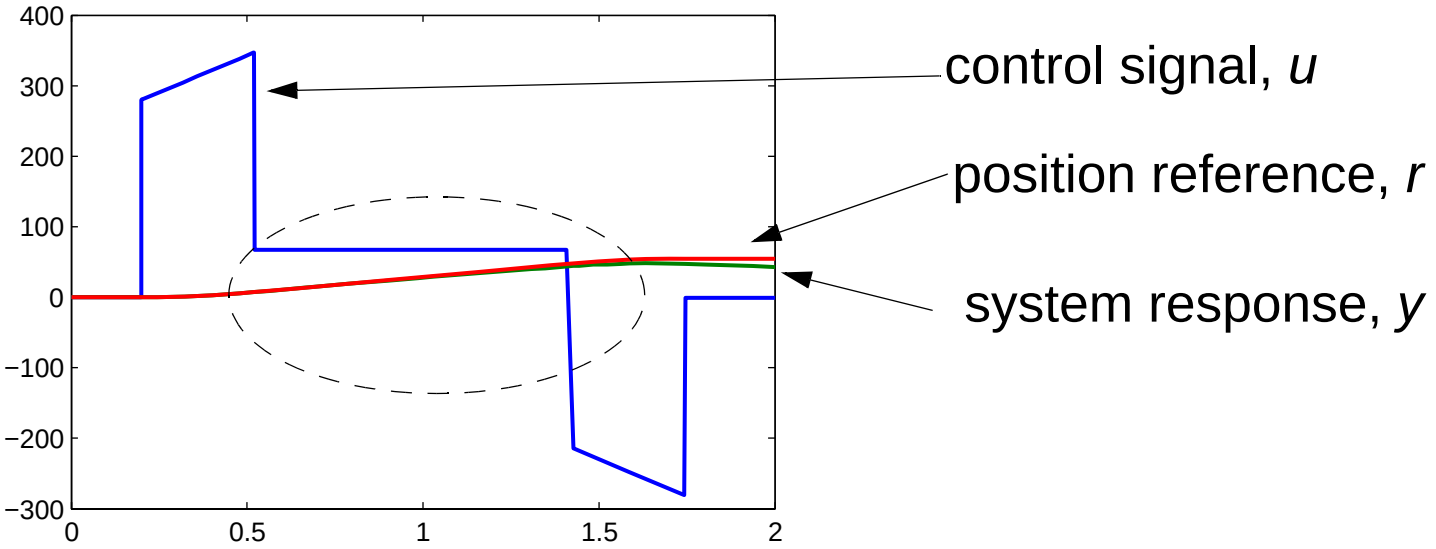


- Let's start out with feed forward to cope better with the acceleration step

$$u_{ff} = 2\ddot{r}.$$

- The process follows fairly good initially when the acceleration signal dominates

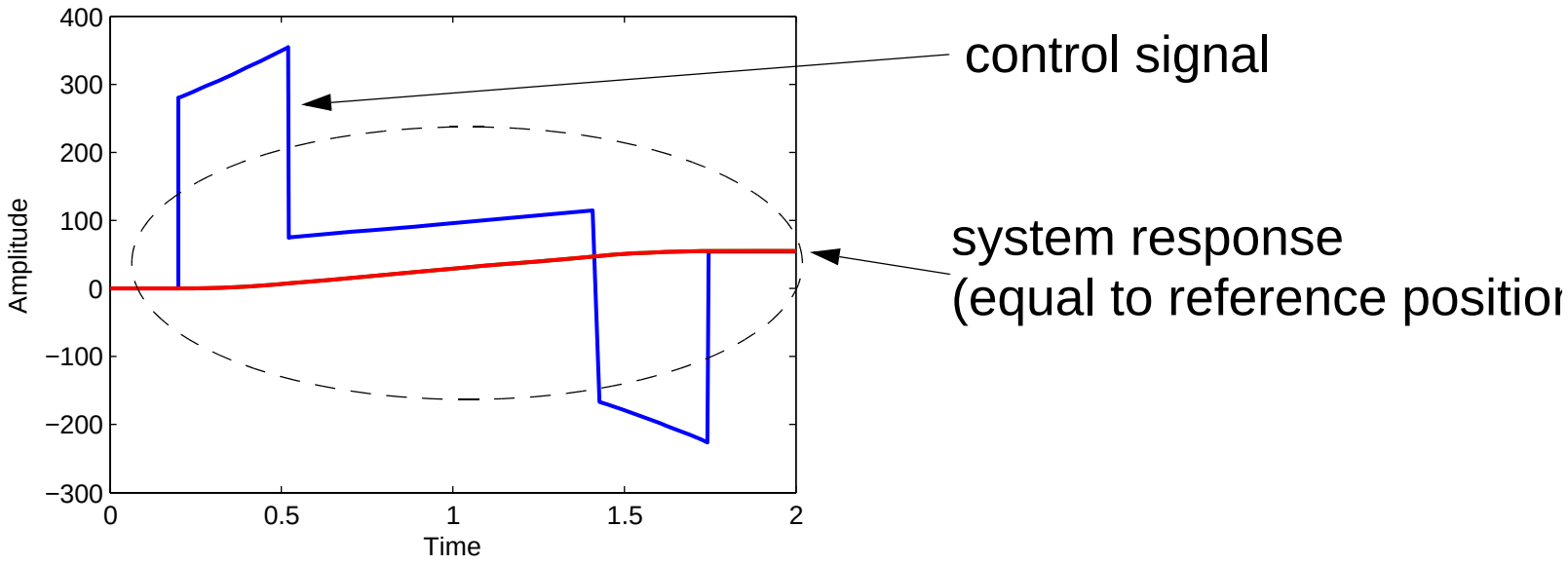
5.4.6. Acceleration and velocity feed forward



- Next we add velocity feed forward which gives improved following of the reference.

$$u_{ff} = 2\ddot{r} + 15\dot{r}$$

5.4.7. Exact model following

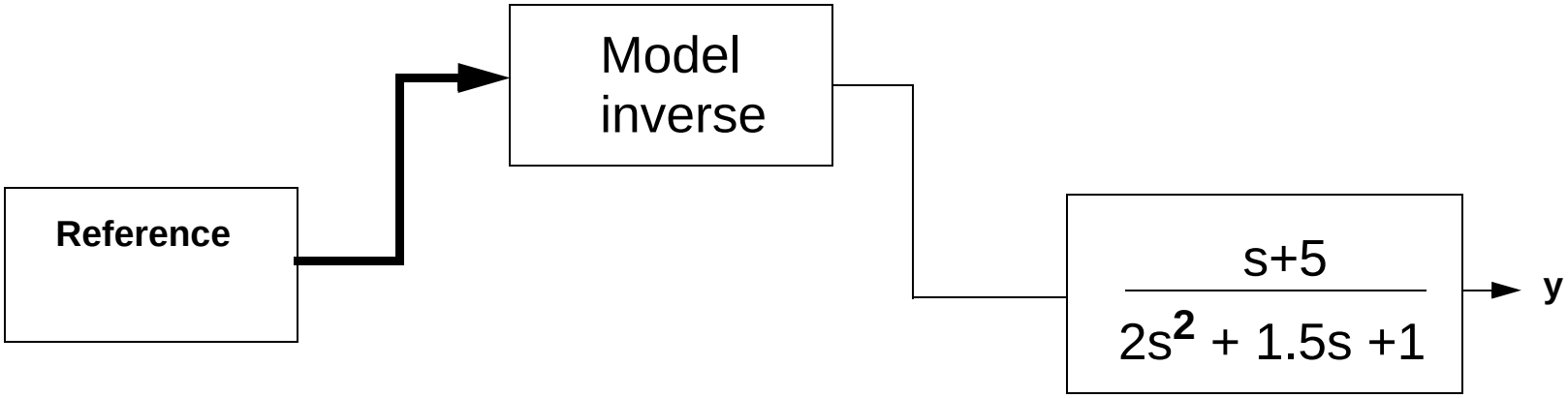


- Finally we add position feed forward which gives exact model following.

$$u_{ff} = 2\ddot{r} + 15\dot{r} + r$$

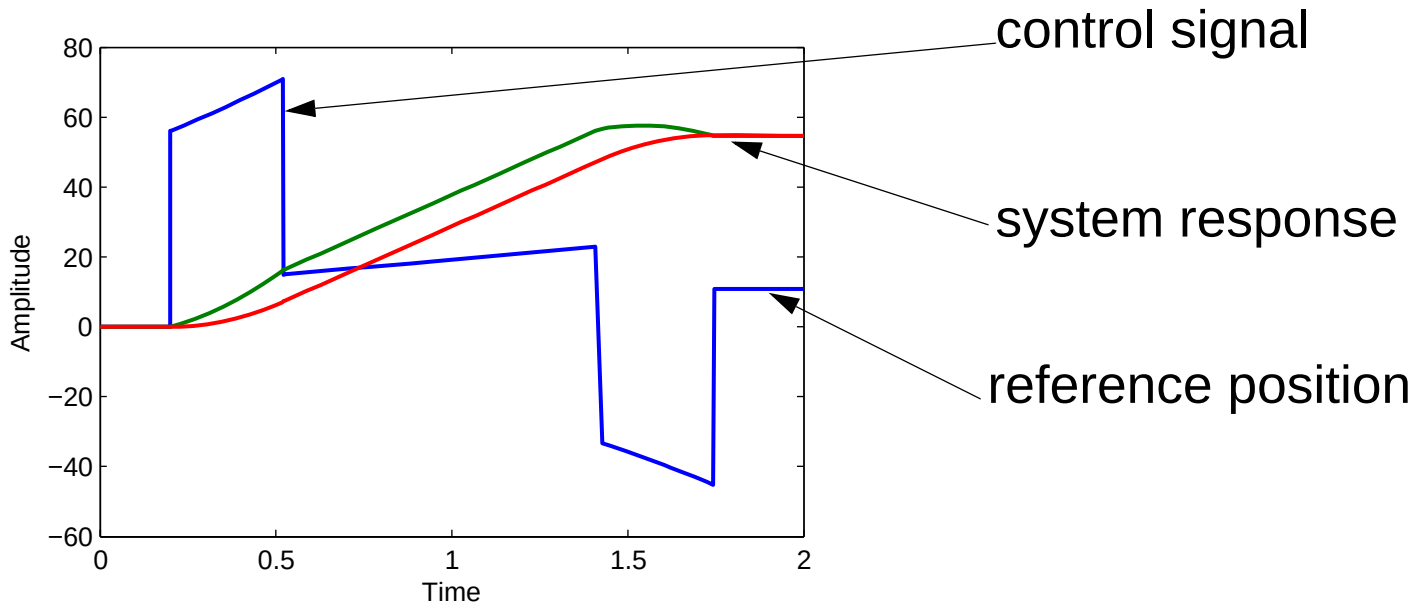
- Note again, that we assume exact knowledge of the model!

5.4.8. Adding a zero to the model



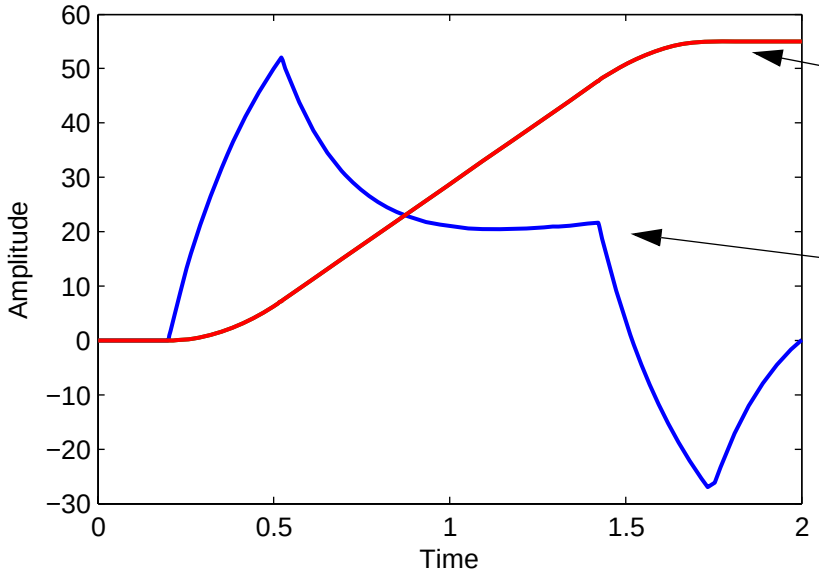
- The process is extended with LHP zero.
- Exact model following is still possible since the inverse system is stable.
- The static gain of the process is 5.

5.4.9. Feed forward without numerator inverse



- Let's first have look on the response with only changing the gain of the feed forward controller. $u_{ff} = \frac{2\ddot{r} + 15\dot{r} + r}{5}$.
- The added dynamics (model zeros) is clearly visible.

5.4.10. Exact model following for LHP zero



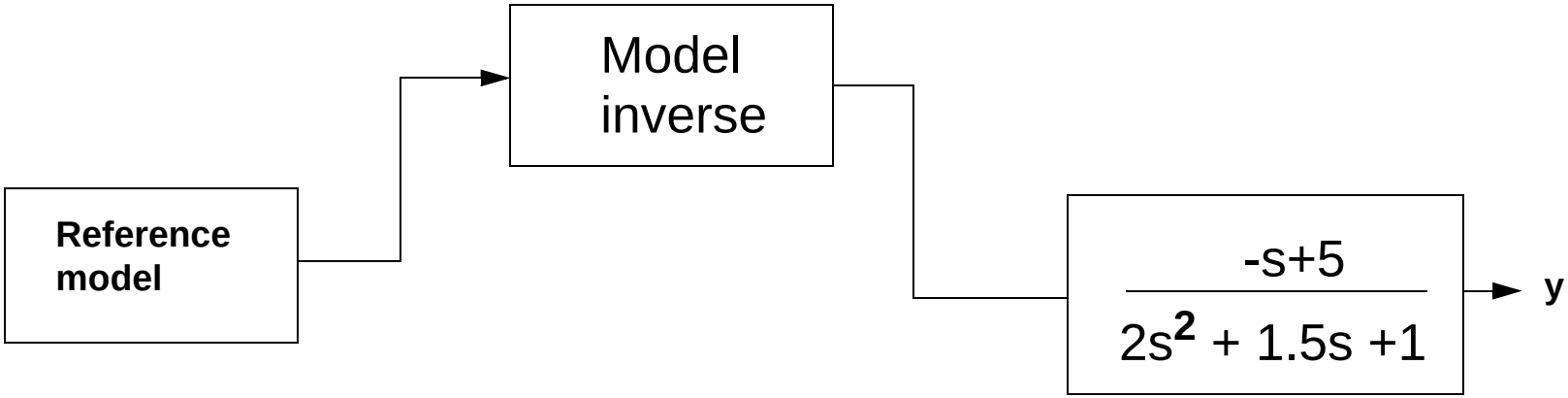
system response, *y* and reference

control signal, *u*

- Exact model following is implemented by including also the inverse of the numerator.

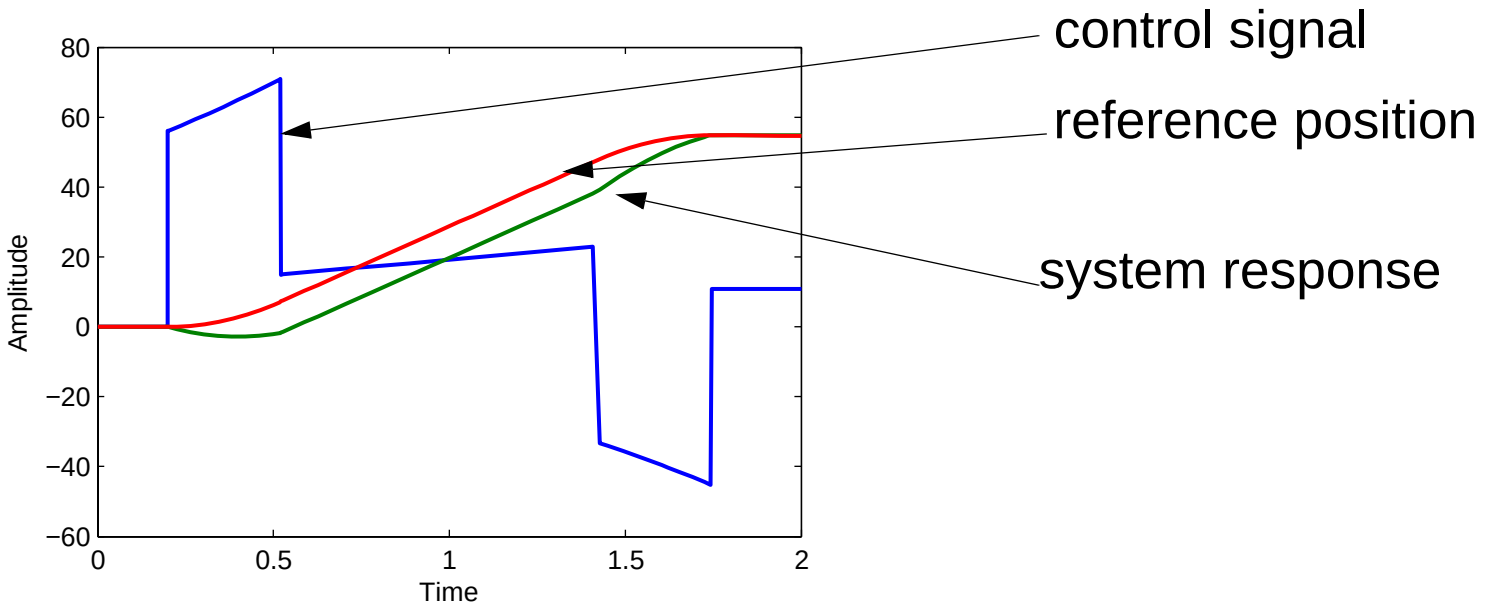
$$u_{ff} = (2\ddot{r} + 15\dot{r} + r)\frac{1}{s+5}$$

5.4.11. A non-minimum phase system



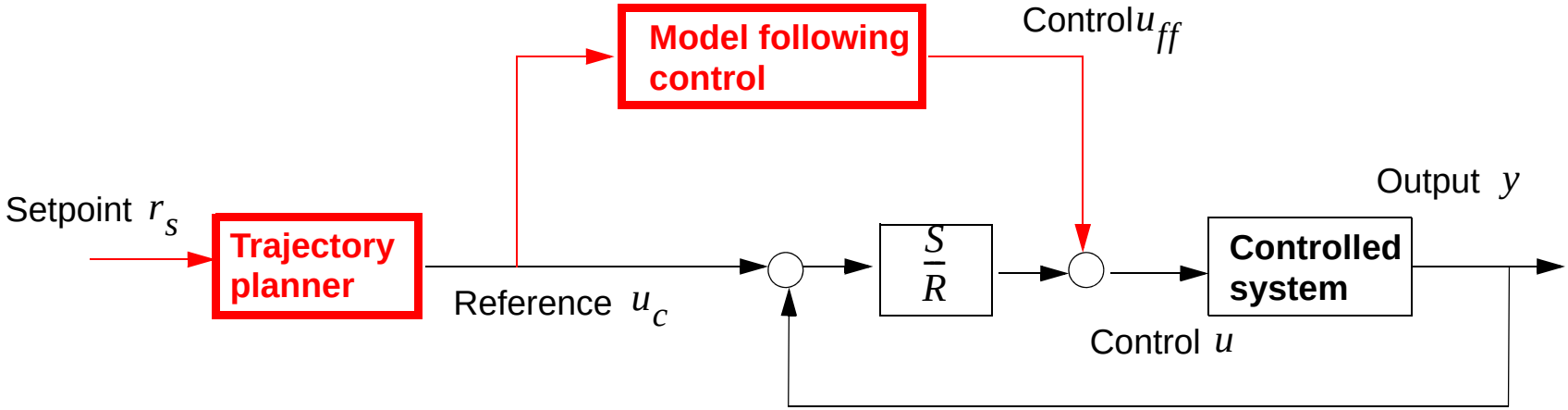
- Clearly, we have a RHP zero, making exact model following impossible. That is, the inverse of the process model constitute an unstable dynamic system.
- The static process gain is still 5.

5.4.12. Approximate model following (RHP zero)



- An approximate model following can be implemented by taking the inverse of the static portion of the numerator $u_{ff} = (2\ddot{r} + 15\dot{r} + r)\frac{1}{5}$.
- Observe the typical response starting in the wrong direction.

5.4.13. Summary



Design the trajectory planner based on the performance of the motor.

Design the model following control based on an inverse model of the process. Excellent for non-linear phenomena.

5.5. Lecture outline

- 1. Introduction
- 2. Transfer function based model following
- 3. Time domain based model following
- 4. An example
- **5. Matlab example for the dc motor**