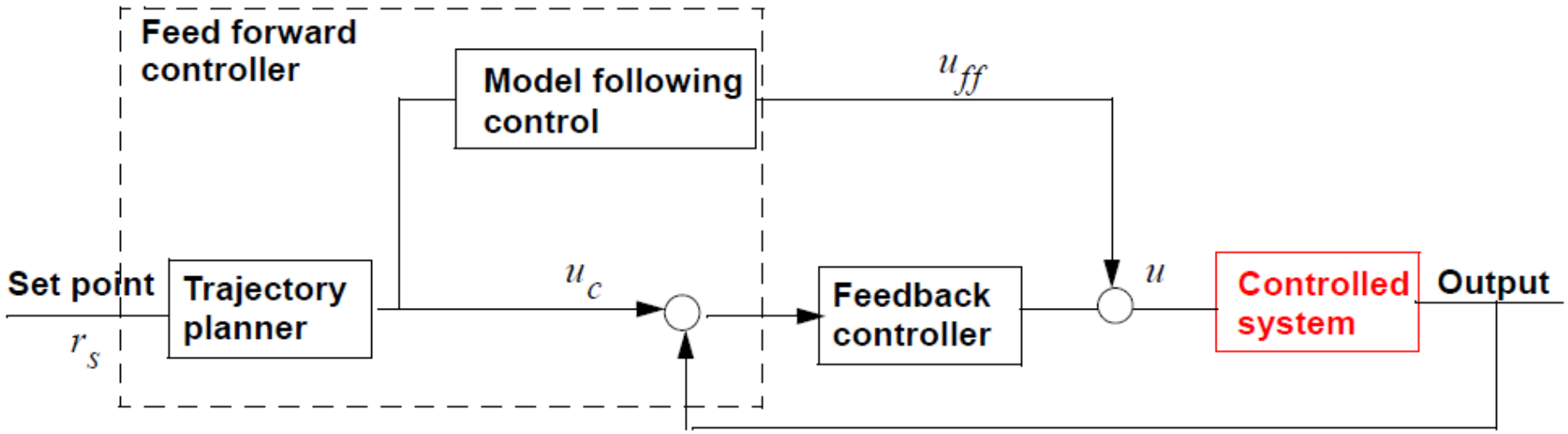
SERVO MOTORS AND MOTION CONTROL SYSTEMS

(Motion Control Systems: Theoretical Background: «model following control, real-time implementation, robustness»-Lecture 8)

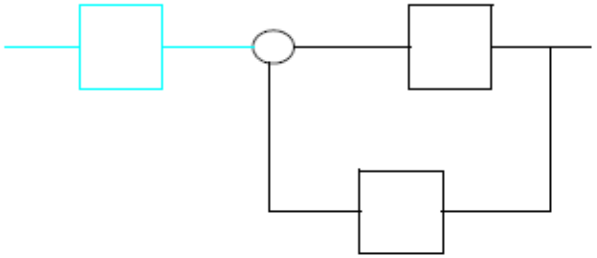
Dr. Caglar UYULAN

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

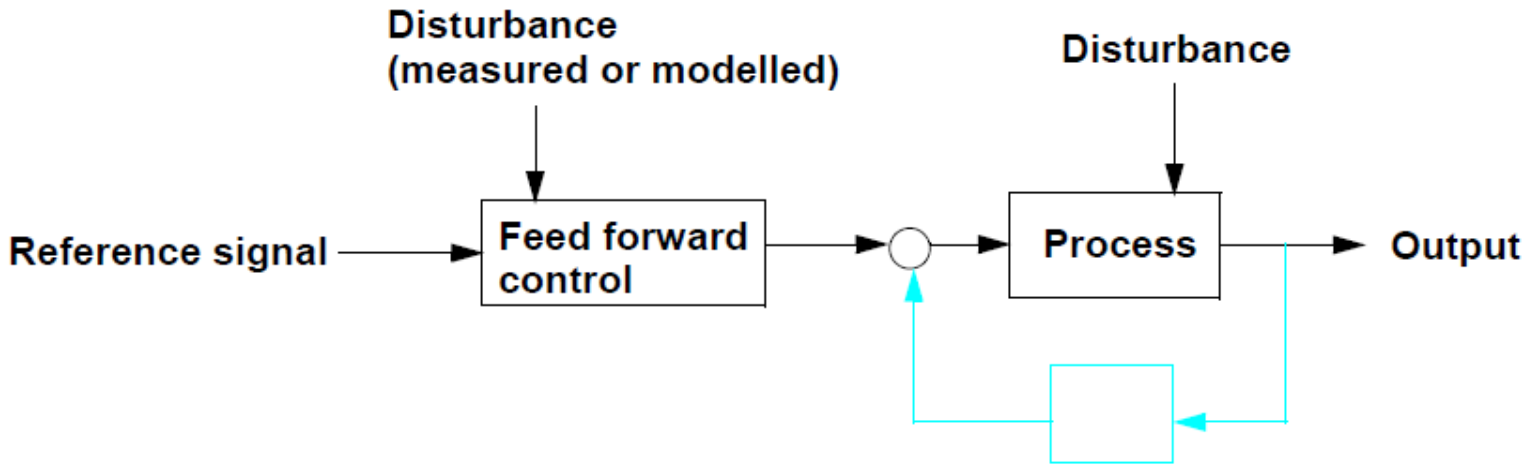


- Characterized by frequent changes in set point (e.g in an industrial robot)
- Control design mainly to follow (track) the reference (e.g. position or velocity, or position, velocity, acceleration and jerk).

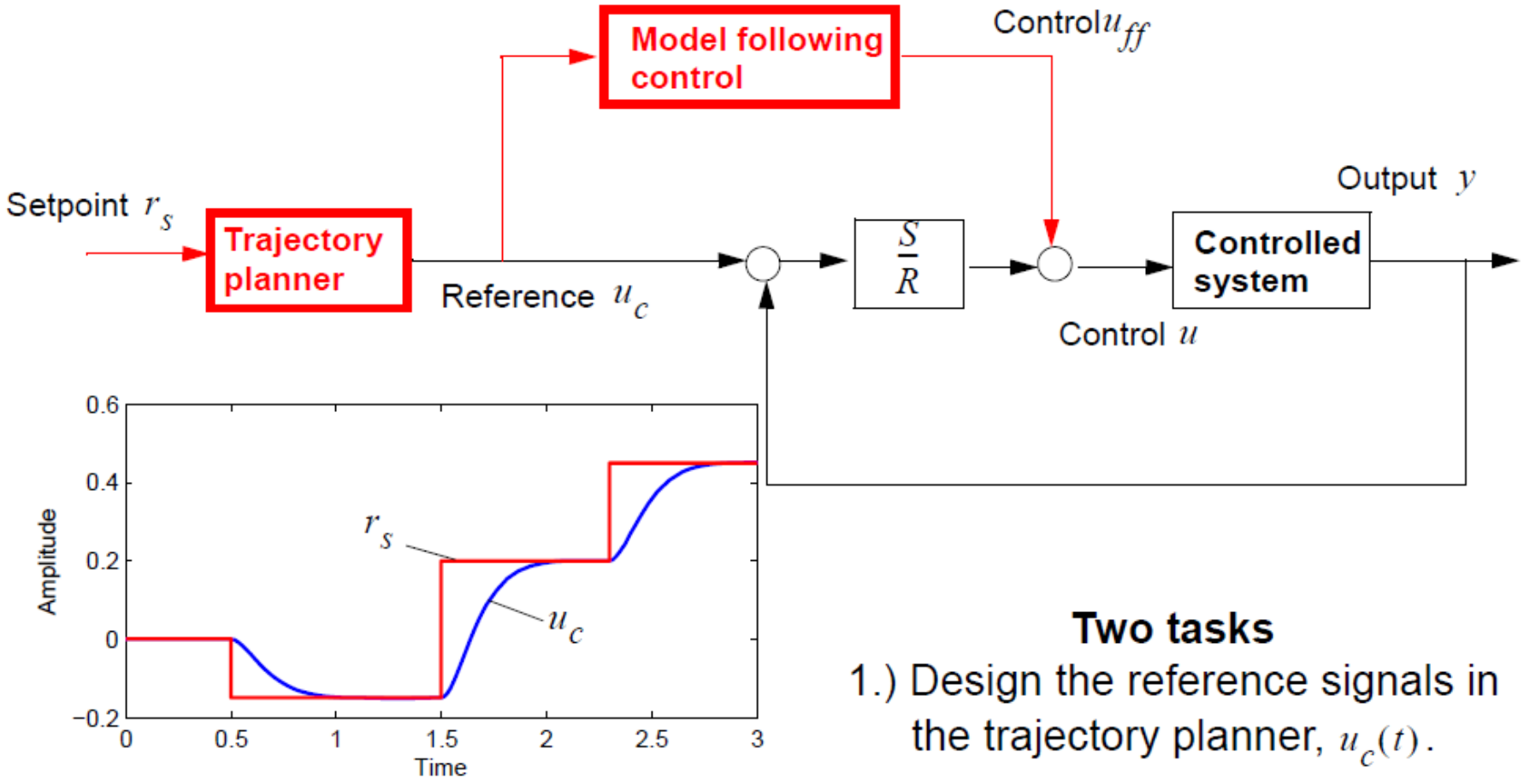
- The main principle in control engineering
- Typically model based (but not required to be)
- Produces control signals after an error has occurred



- 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



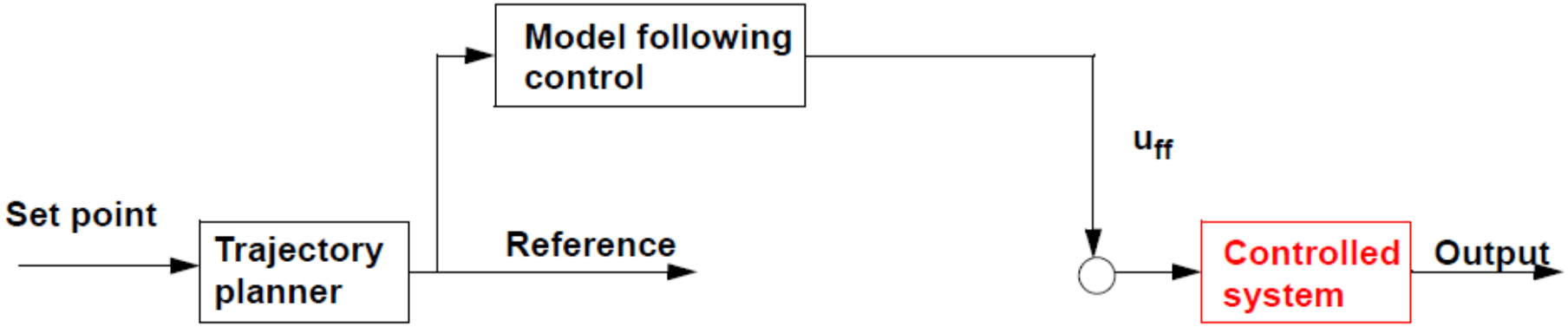
- Model based (but fairly simple models typically helps a lot)
- Produces control signals **before** error has occurred
- Uses measured or modelled disturbance and compensates for it
- Uses carefully designed reference signals to make the process follow the references “exactly” and without saturating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



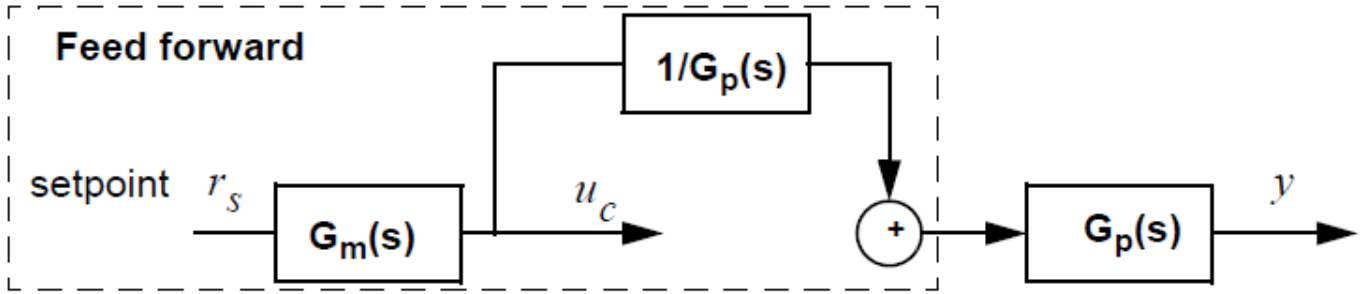
Two tasks

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

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



- 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.



- $G_m(s)$ 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.

- 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

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

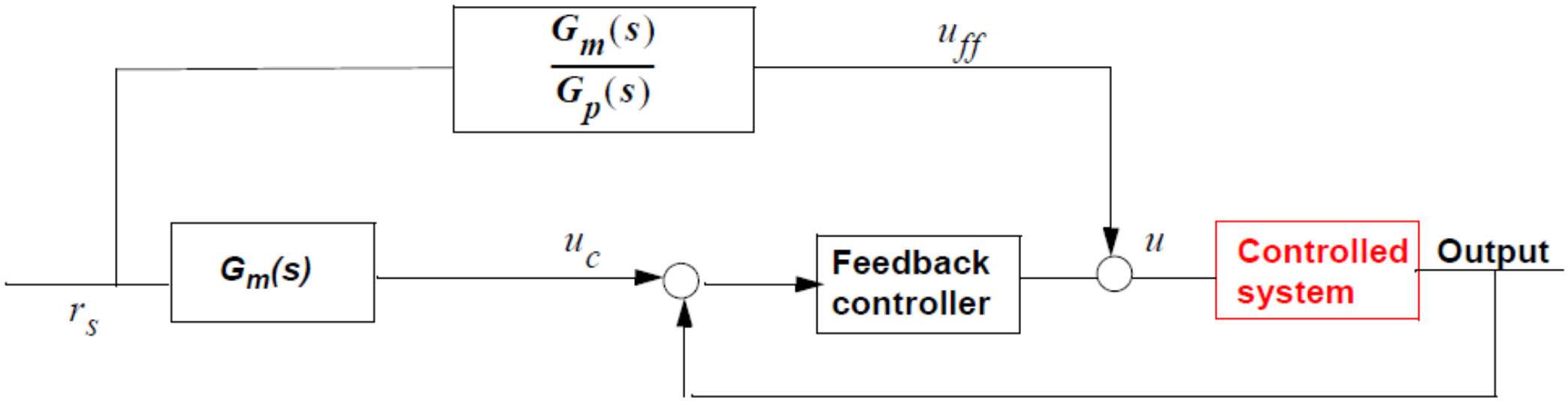
and finally the requirement

$$d_m \geq d_p$$

- 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.


$\frac{G_m(s)}{G_p(s)}$
u_{ff}
$G_m(s)$
r_s
u_c
Feedback controller
u
Controlled system
Output

- 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.

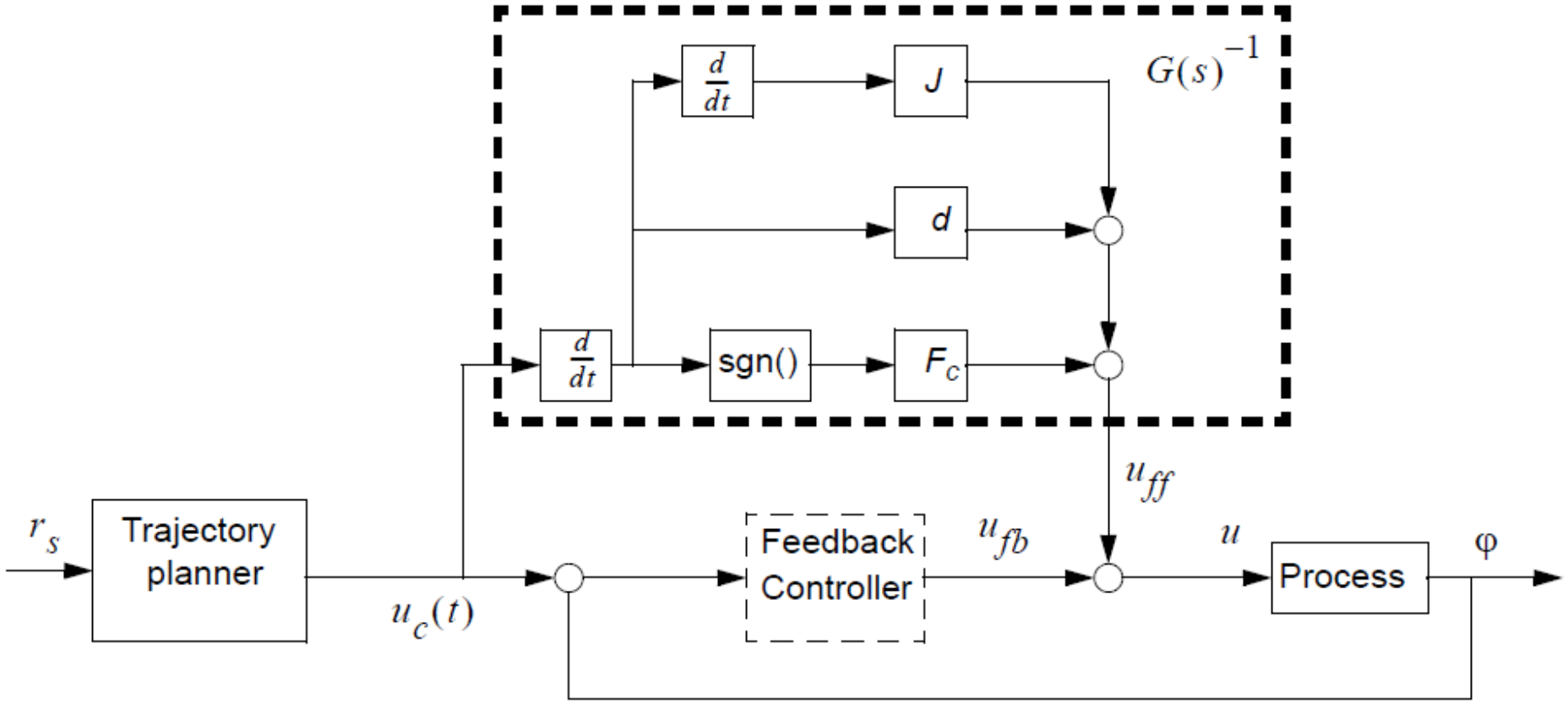
- Model following: If the inverse model (from *y* -> *u*) can be written as a function of the output and n-times the derivatives of the output (n:th order model).

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

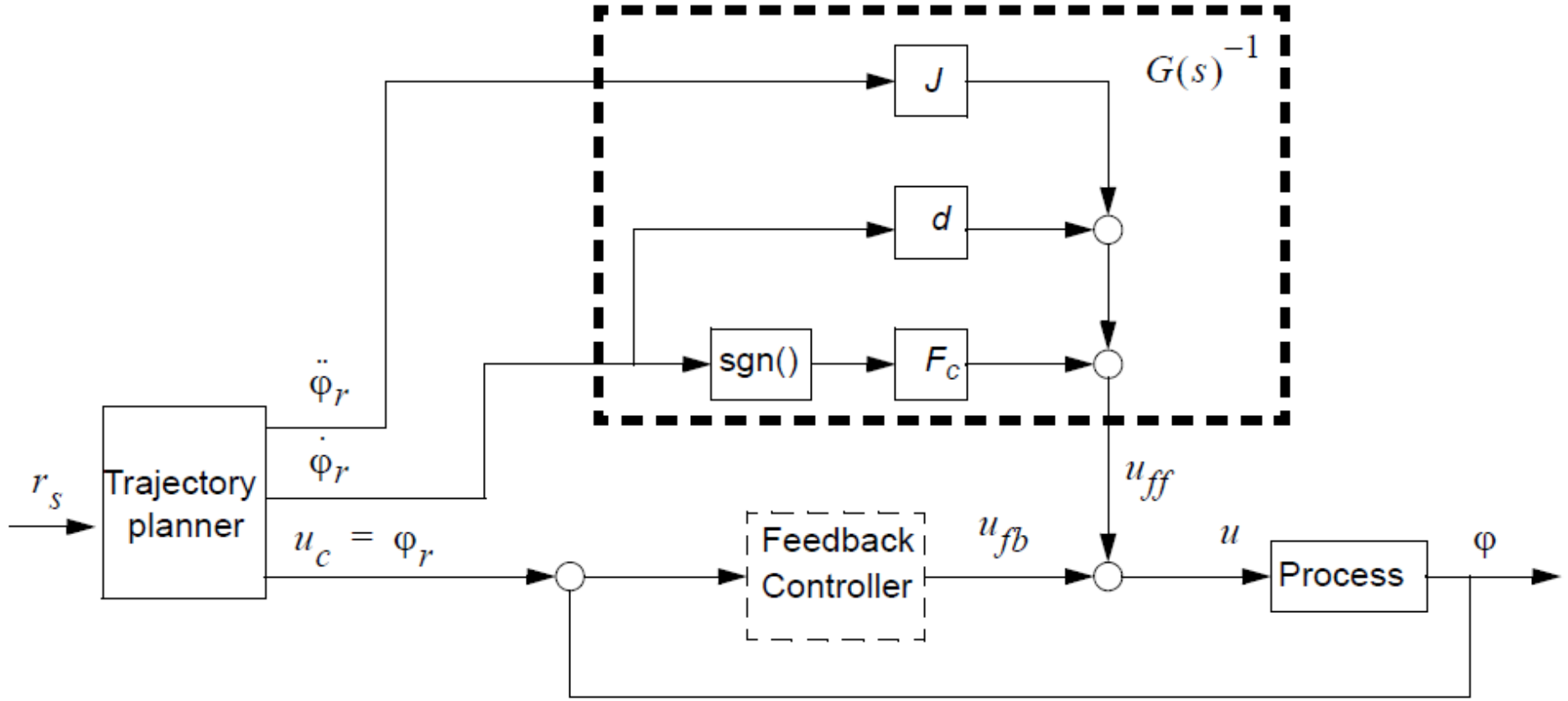
- Trajectory planner: Design $y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, \ldots, \frac{d^ny}{dt^n}$ based on closed loop specifications and process limitations, e.g., saturation.

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, $u(t) = J\ddot{\varphi}(t) + d\dot{\varphi} + F_c \mathrm{sgn}(\dot{\varphi})$.



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

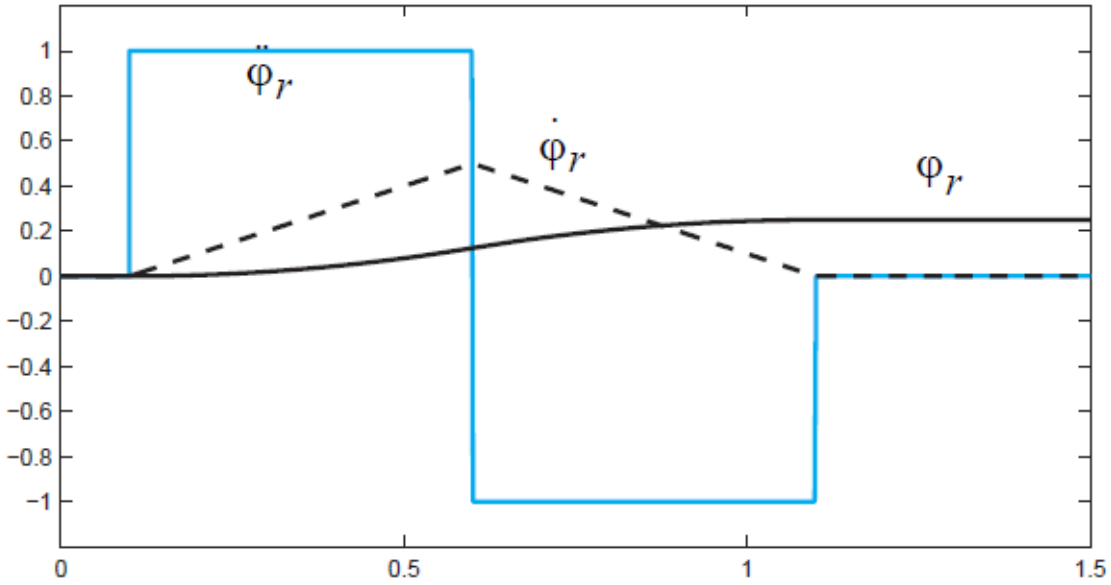


- **The DC-motor is a second order diff. eq. from input to position**

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

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

  ex.



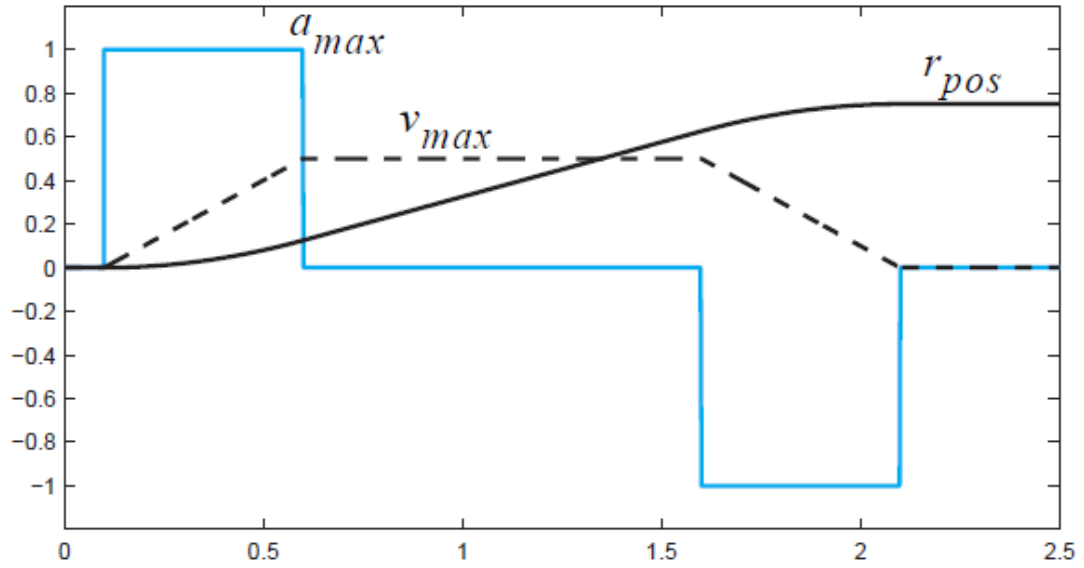
- **Fastest possible positioning (or specific time trajectories)**

  - 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

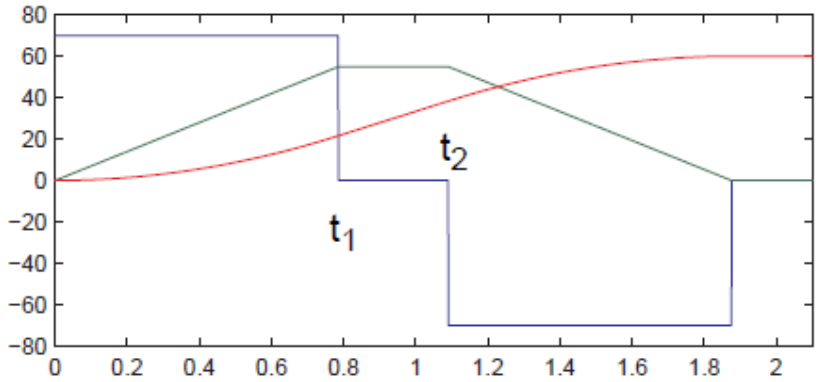


$v = at$, zero initial condition!

$t_1 = \frac{v_{max}}{a_{max}}$, 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::

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

- **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.

- 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

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

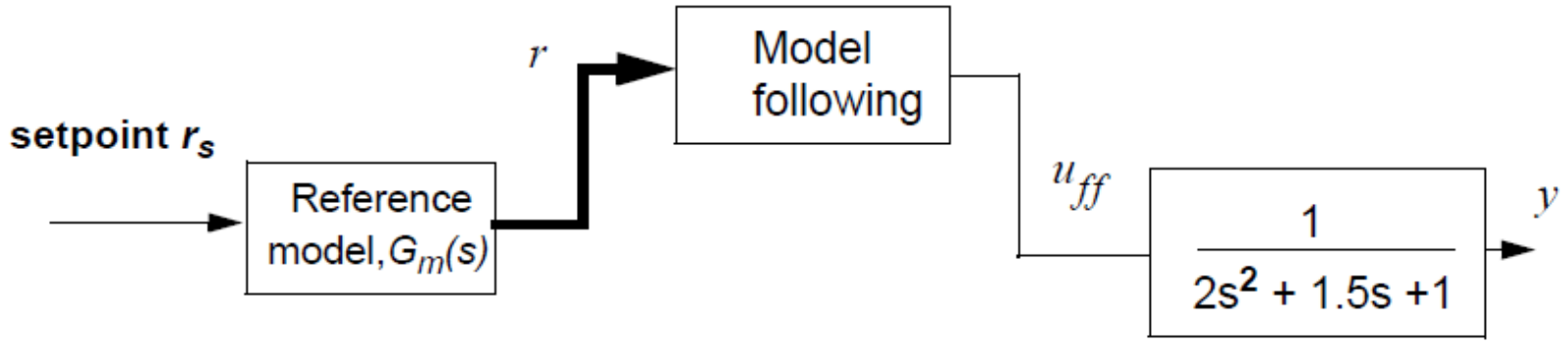
- Or in transfer function form

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

- 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

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

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

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

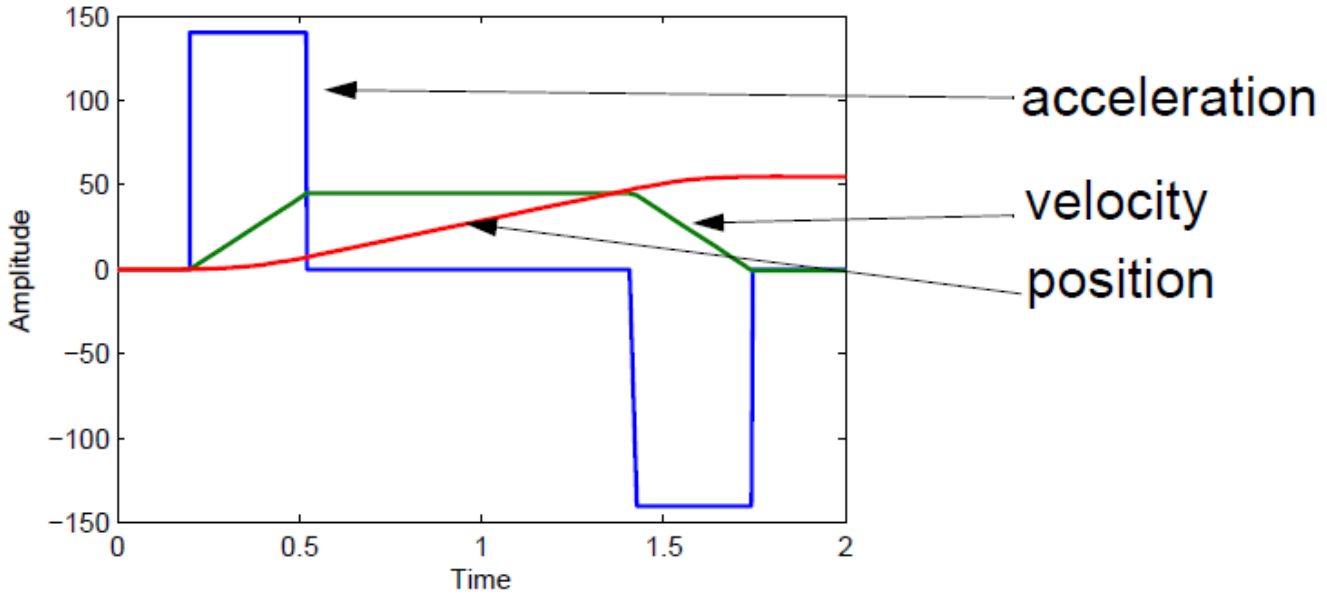
$$G_m(s) = \omega^2/(s^2 + 2\xi\omega s + \omega^2)$$

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

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

$$u_{ff}(s) = (2s^2 + 15s + 1)r_s(s)$$

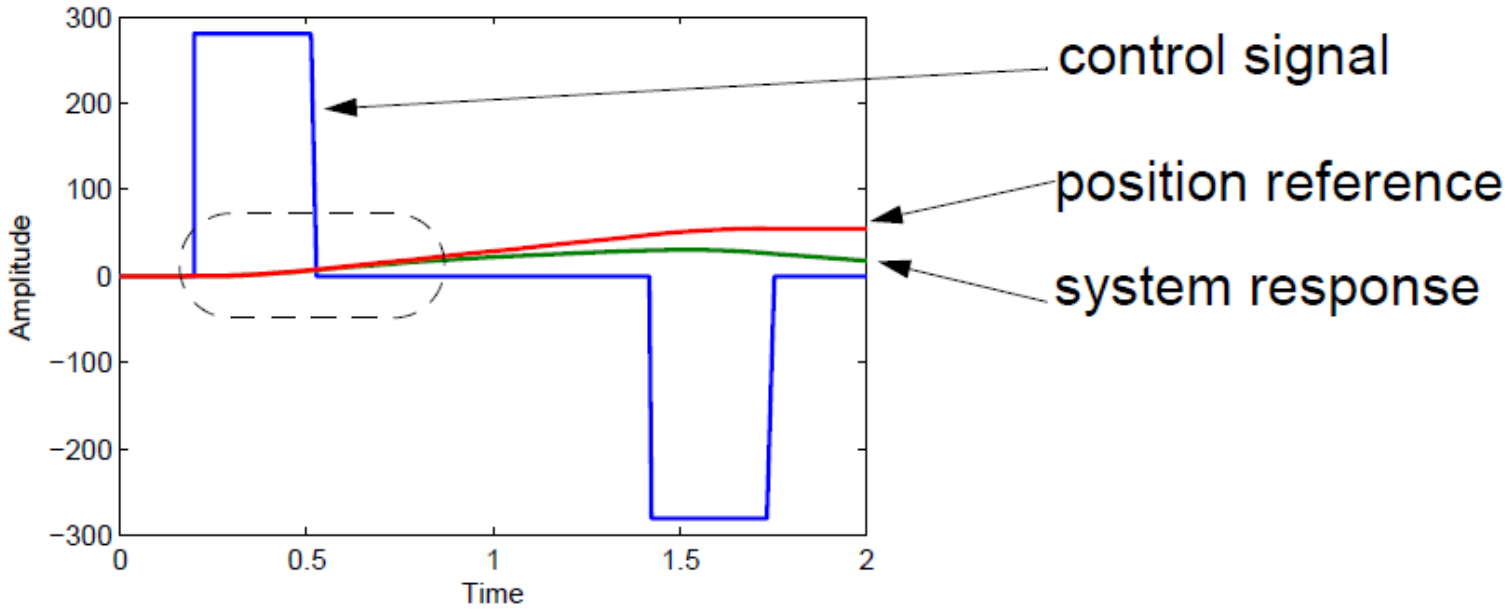
$$u_{ff}(t) = 2\ddot{r}_s(t) + 15\dot{r}_s(t) + r_s(t)$$



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

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

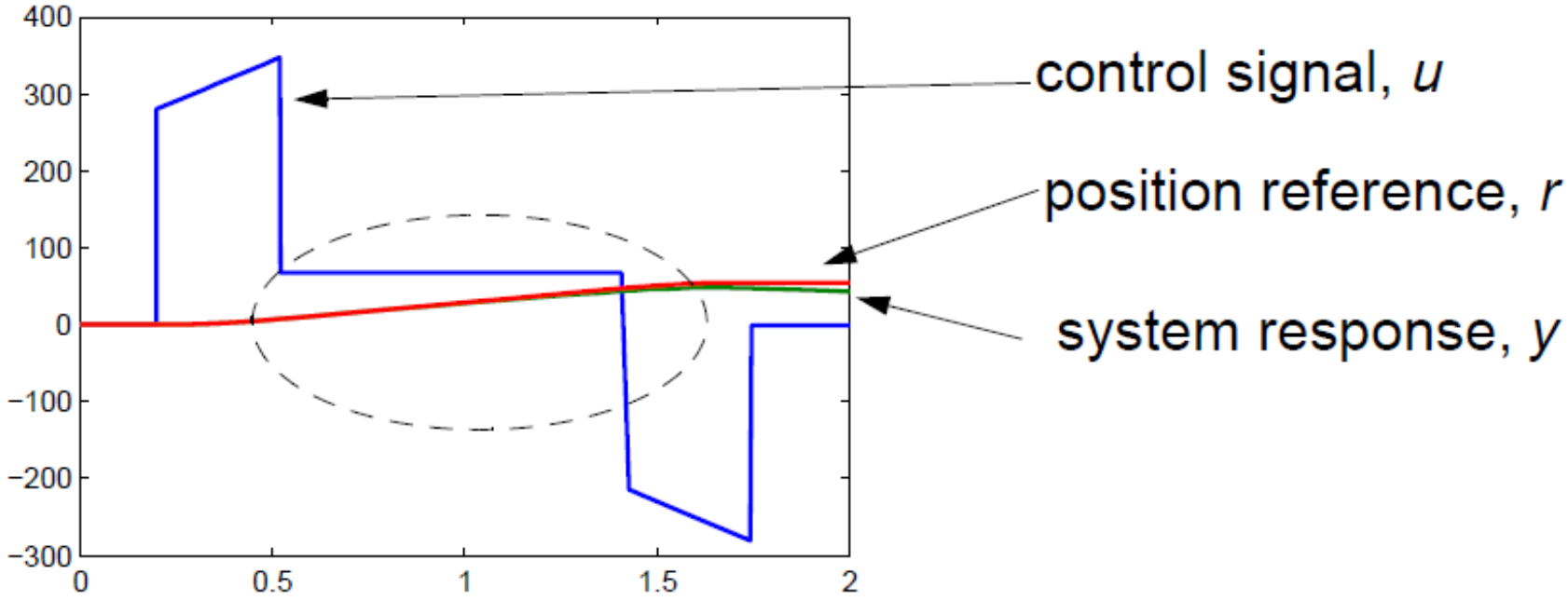
Observe, initial conditions



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

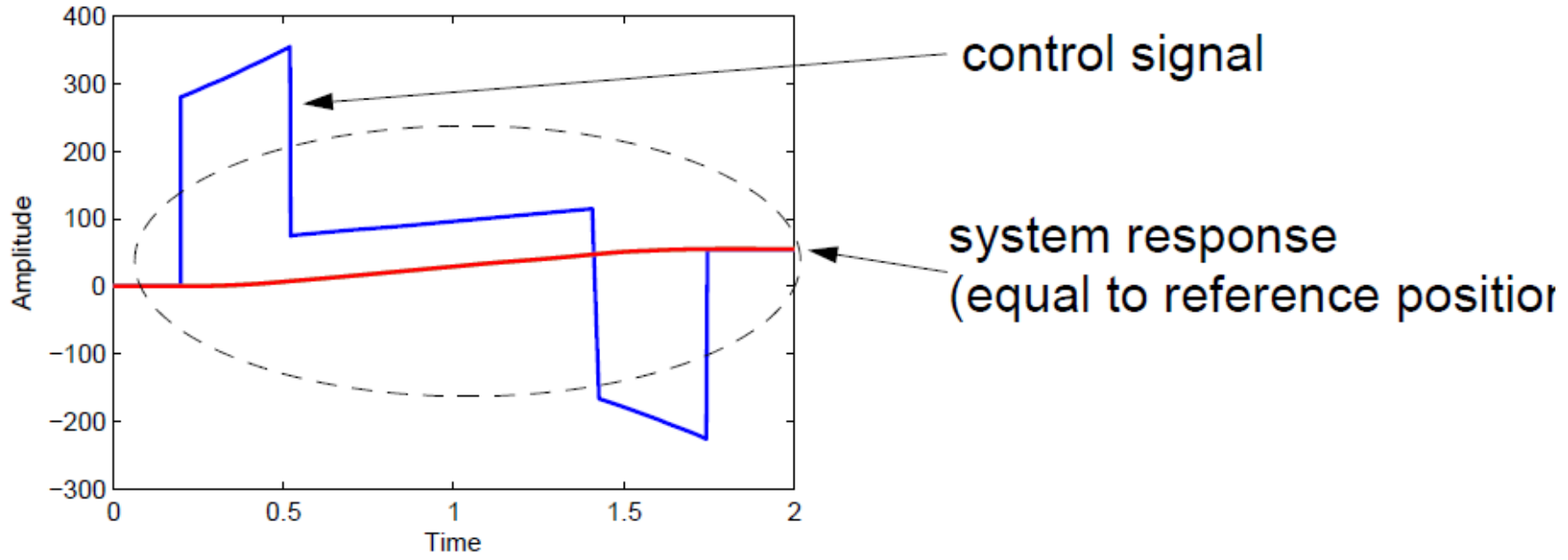
$$u_{ff} = 2\ddot{r}.$$

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



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

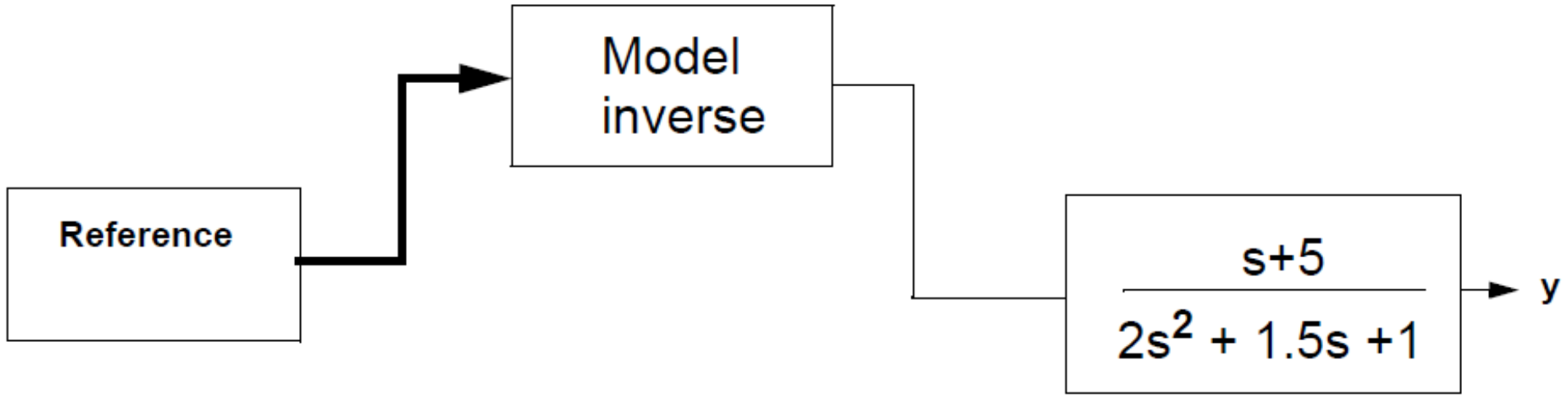
$$u_{ff} = 2\ddot{r} + 15\dot{r}$$



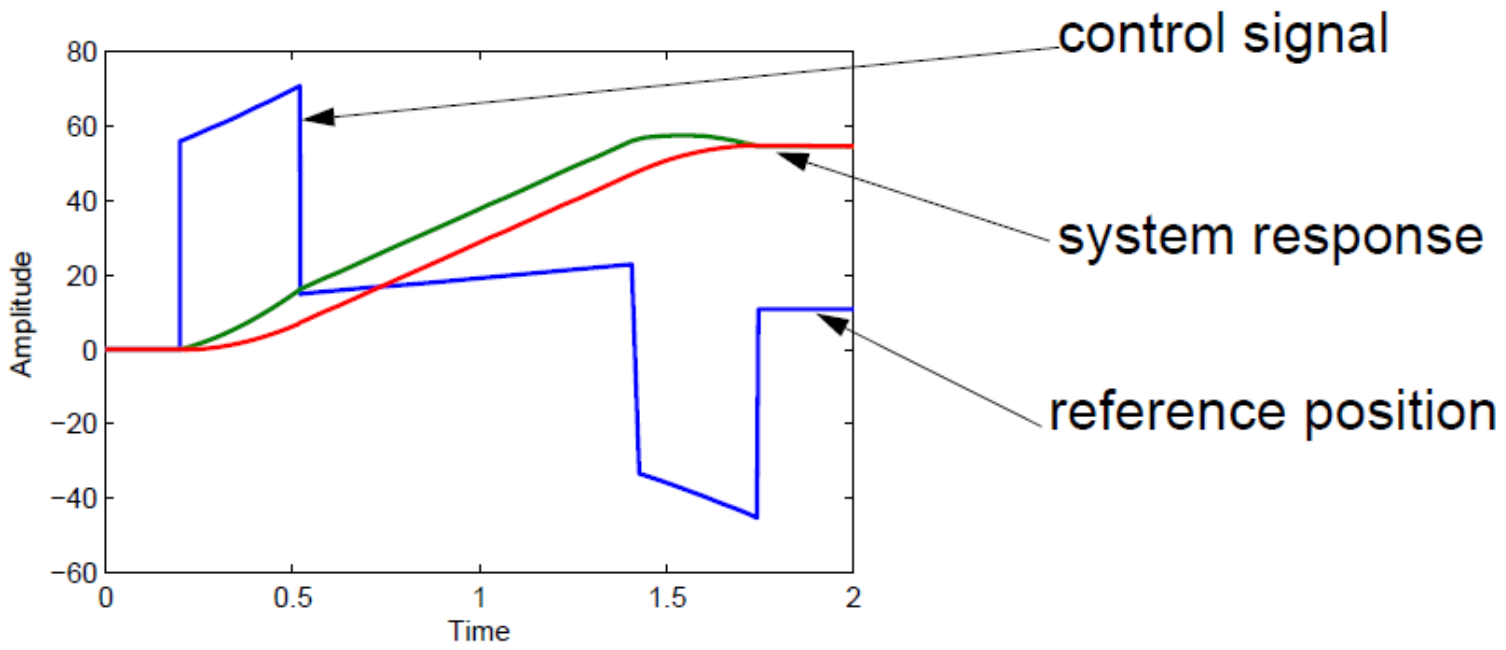
- Finally we add position feed forward which gives exact model following.

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

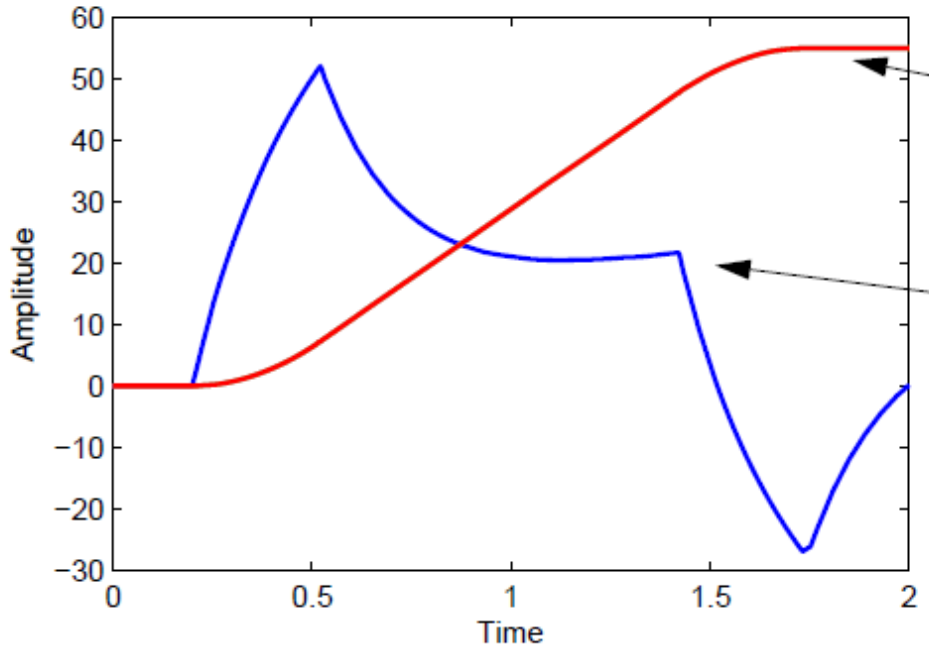
- Note again, that we assume exact knowledge of the model!



- 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.



- 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.

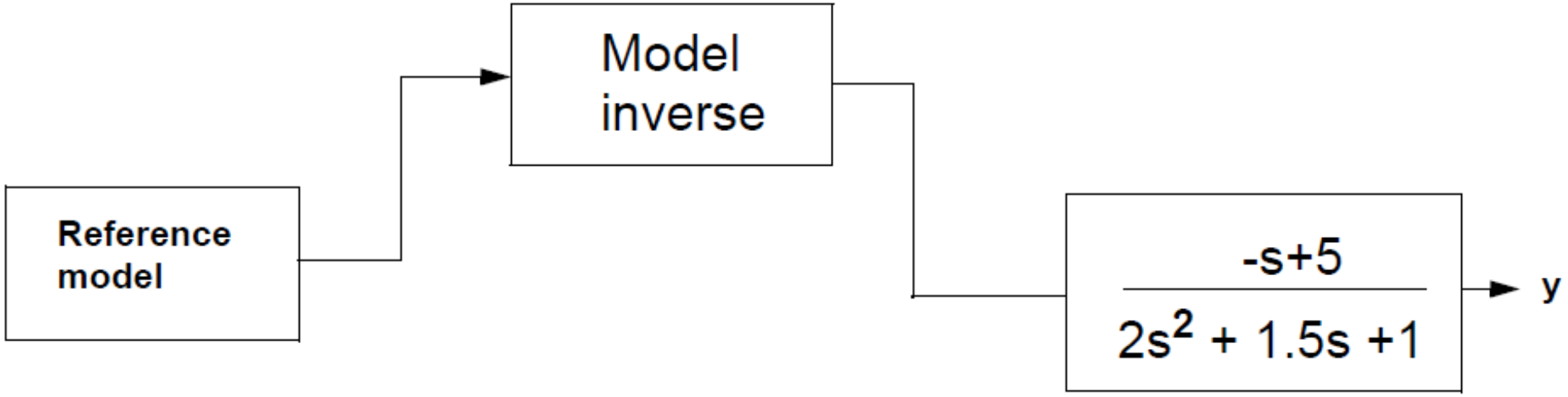


system response, *y* and reference

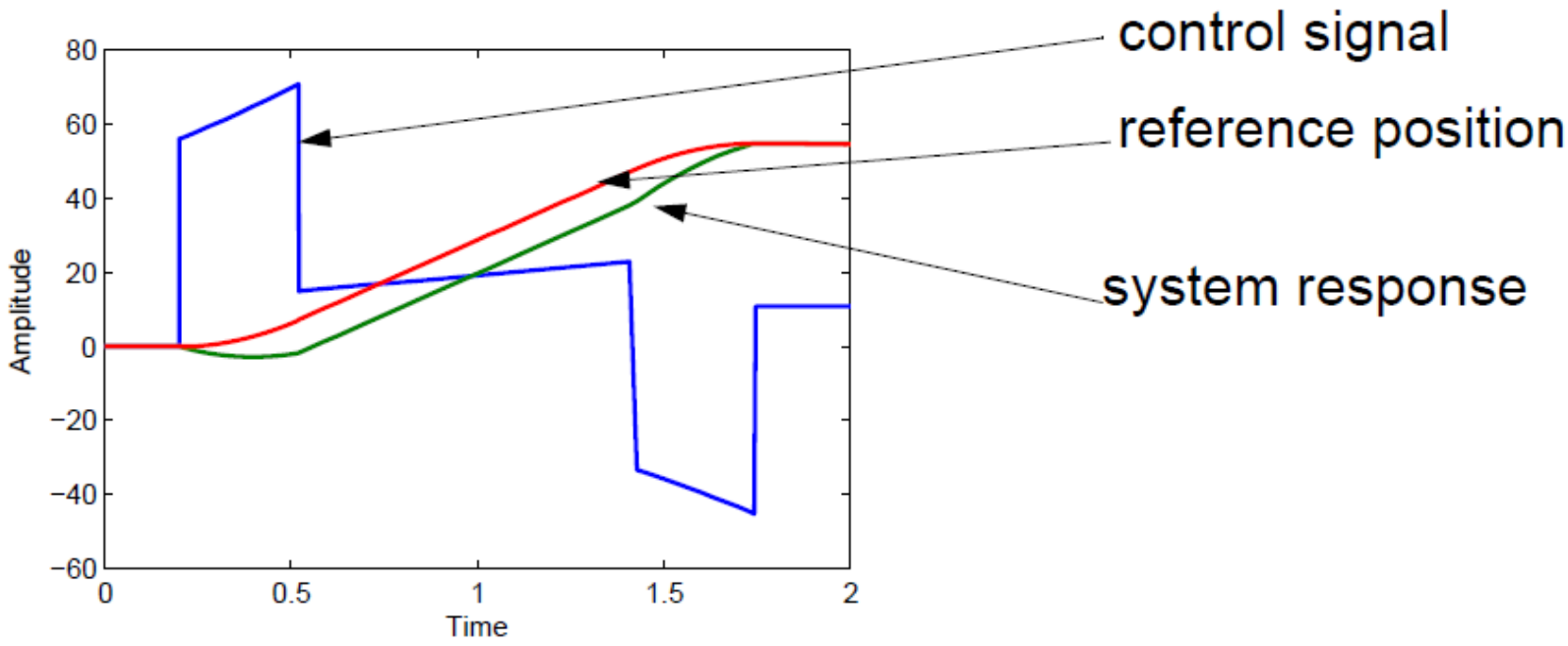
control signal, *u*

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

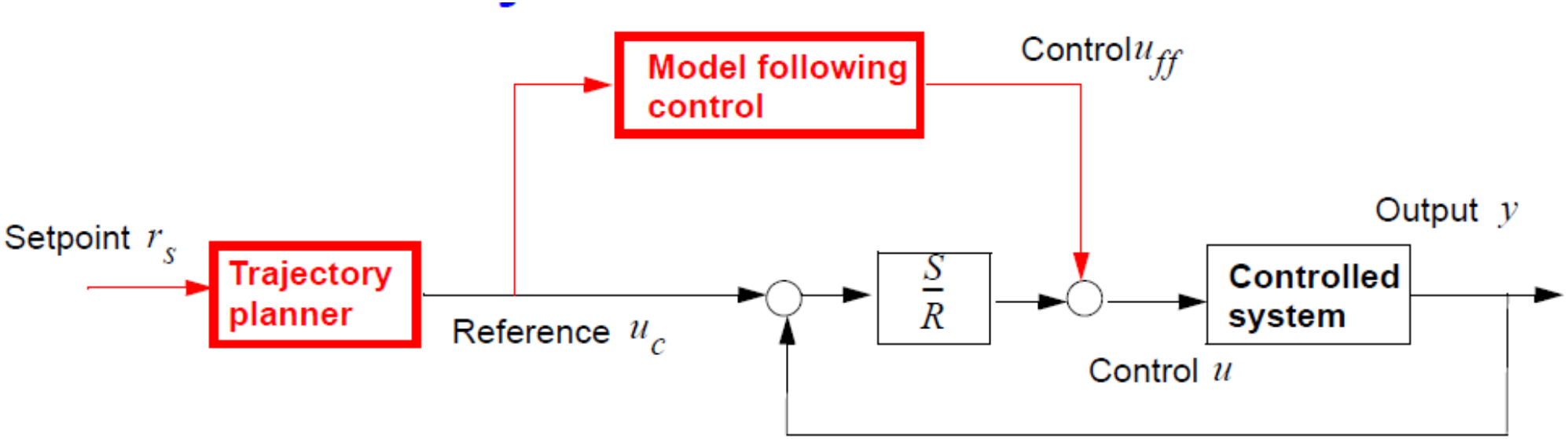
$$u_{ff} = (2\ddot{r} + 15\dot{r} + r)\frac{1}{s+5}$$



- 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.



- 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.



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.

From Control Design to Real-time Implementation

Contents

Lecture taster! and Recap

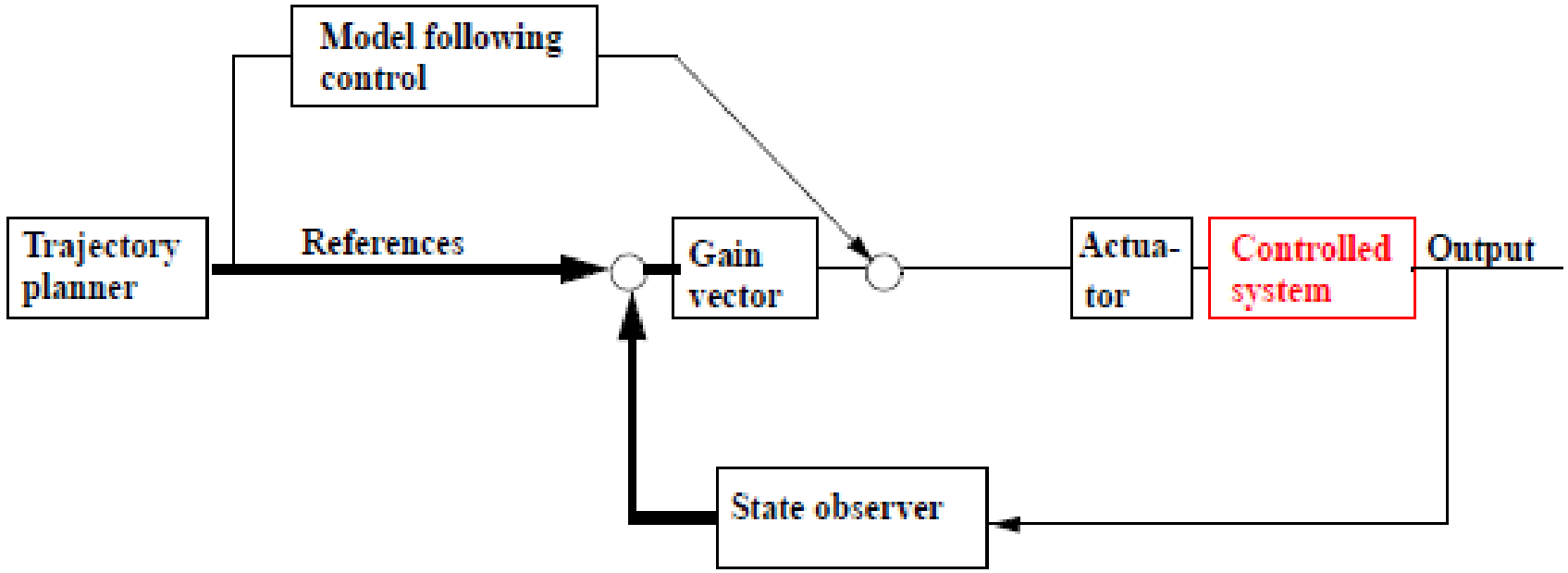
Control system implementation - overall perspective and issues

Controller implementation techniques and examples

- controller viewpoint
- implementation technology, trade-offs and optimization

More on control system implementation and tools

- Control algorithm implementation
- What can go wrong
- Real-time implementation and workshop E: Rubus and Tasking

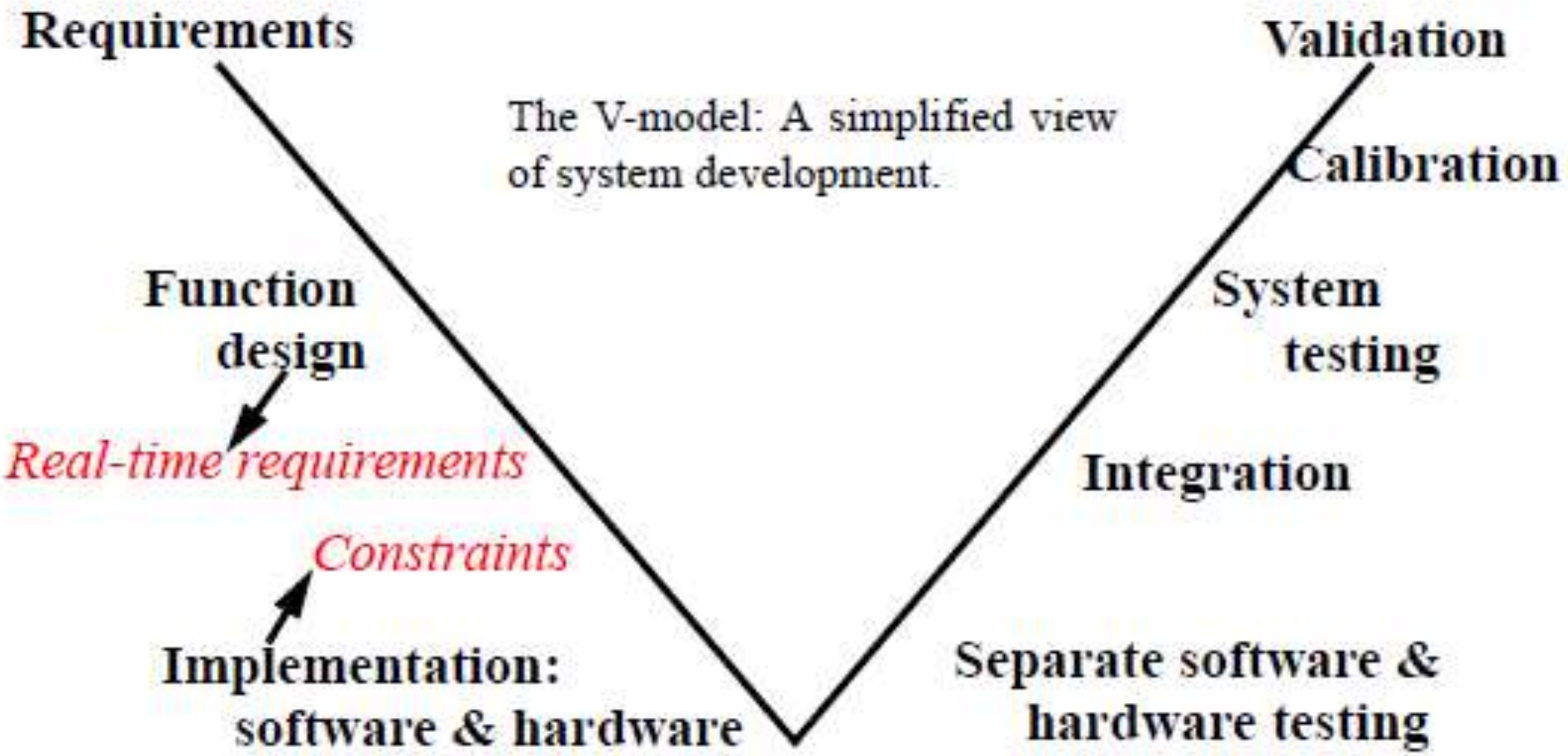


Going from a control design and algorithms described as differential equations to C-code.

- Modeling, Control design and Observers
- Discretization, choice of sampling interval

We will additionally in this lecture look at

- trade offs in selecting the sampling period
- how does the feedback delay affect performance?
- quantization (briefly)
- trade-offs in the implementation (cost, accuracy, speed and memory usage)
- translating the control system (equations/block diagram) to code (C-functions)
- common faults and how to avoid them


Requirements
Validation
The V-model: A simplified view of system development.
Calibration
Function design
System testing
Real-time requirements
Integration
Constraints
Implementation: software & hardware
Separate software & hardware testing

Control engineering part (project subtasks Pa, Pb, Pc, Pd, F1)

- Mechanical system analysis, modeling and model verification
- Control design, state feedback, sampled systems
- Model following
- Control algorithm discretisation
- Rapid Control Prototyping

Software engineering part (project subtasks Pe, Pf, F2)

- Embedded systems, Control algorithm implementation
- Concurrent programming and real-time operating systems
- Diagramming

- *A very broad range of applications!*

  "*Ubiquitous Computing*" nature of embedded systems
  (Mark Weiser)

- *From simple to complex automated control*

- *Very different requirements and characteristics regarding:*

  Environmental aspects, interfacing and performance
  Real-time and life time
  Safety, reliability, availability and security
  Size/complexity, flexibility
  Cost-sensitivity, users, legislation and certification

- *Harsh environment*
  - EMC, Vibrations, Temperature
- *Functionality:*
  - Fan, Alternator,
  - Cranking motor,
  - Engine brake
  - Turbo, EGR
  - Diagnostics, Logging
  - Communication
- *Other central issues:*
  - Safety, long life-time, maintenability

Scania: Engine *and* S6 engine control unit

Note for the example:

Different requirements on tasks;

e.g. speed driven injection control, sampled data control, and event-triggered engine brake control.

Human-machine interface

Time-triggered activities such as control loops

Event-triggered activities (communication, interrupts, etc.)

Common characteristics:

- A mixture of timing requirements
- Different modes of operation
- Diagnostics
- Configurability (on-line parameter change, software upgrade)
- Connectivity (for example diagnostics via GSM)

Goal from control viewpoint: Sufficiently fast with respect to the desirable dynamics of the closed loop system

$$T \ll t_{rise}$$

$$f_{bandwidth} \ll f$$

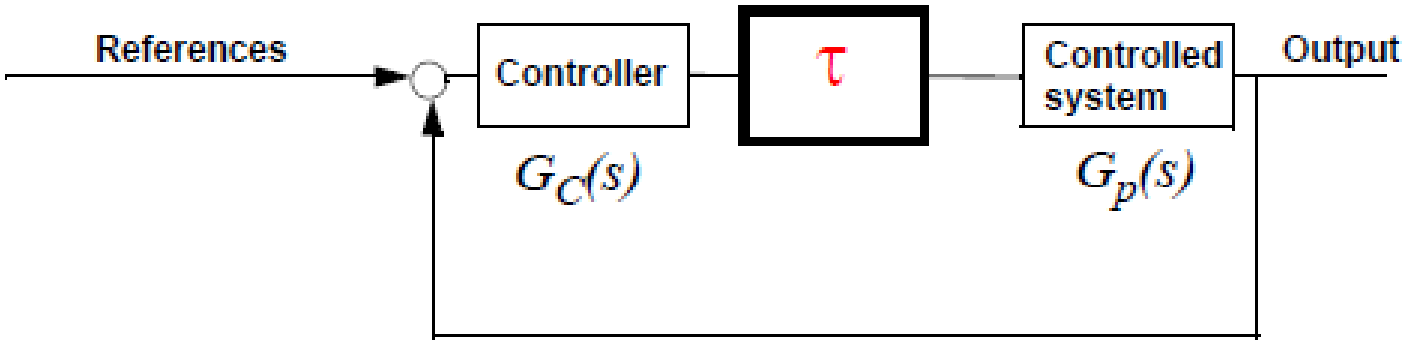
$T = 1/f$ = sampling period, f = sampling frequency

Closed loop system: t_{rise} = rise time, $f_{bandwidth}$ = bandwidth

⇨ How do you choose sampling period (recall earlier lectures)?

⇨ *What other issues affect the choice of the sampling period?*

- Main approach: Minimize the delay τ or make it constant

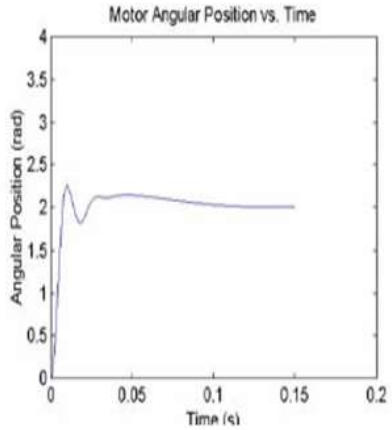


For most systems, the average delay is most important but exceptions do exist.

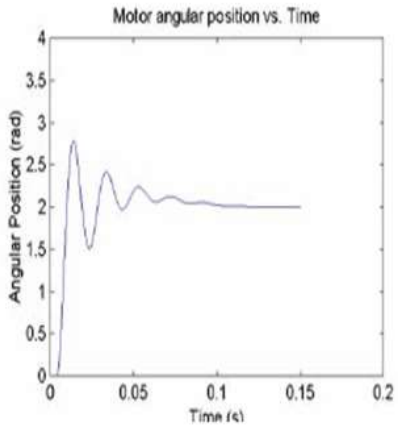
Compensation for a constant delay is straightforward, but requires a model of the controlled system.

Behavior of a second order system controlled by a PID controller:

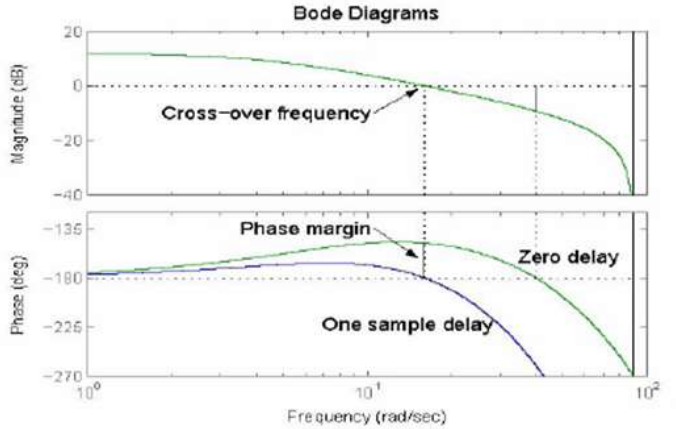
Simulation without any delay



Simulation with constant delay



Given: open loop system $G_C(s)G_P(s)$ with cross-over frequency, ω_c and phase margin, φ_m



Given: open loop system $G_C(s)G_P(s)$

- cross-over frequency, ω_c
- phase margin, φ_m

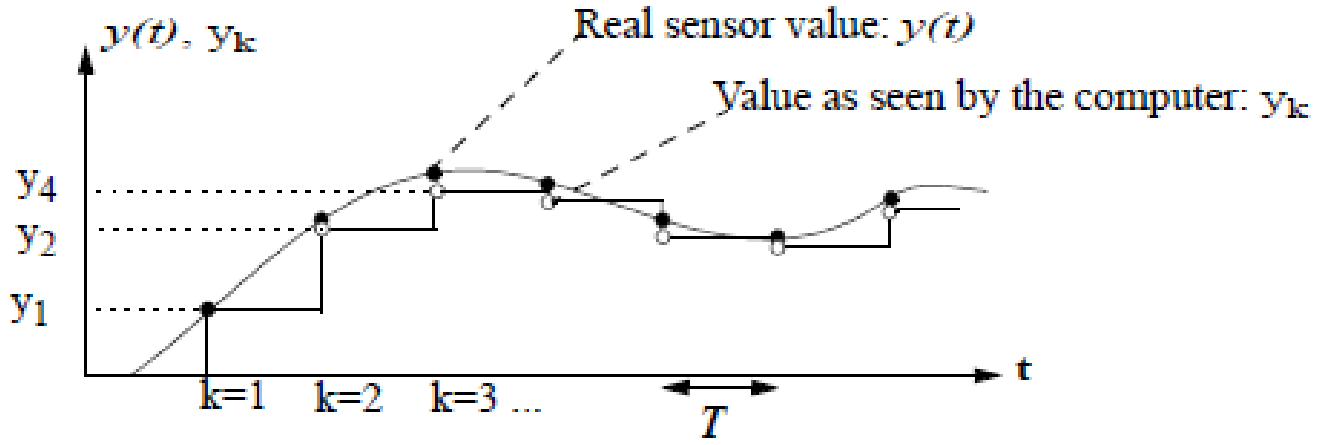
Laplace transformation of delay gives: $G_\tau(s)=e^{-s\tau}$

$$arg\ G_\tau(i\omega)= arg\ e^{-i\omega\tau}=-\omega\tau$$

At cross-over frequency: $arg\ G_\tau(i\omega_c) = -\omega_c\tau$

Thus stability requires that the delay fulfills: $\tau < \varphi_m/\omega_c$

since $\varphi_m>0$ is the condition for stability.



Roundoff, overflow, and underflow in operations (addition etc.)

The difference, [$y(t)$ - y_k], shown in the figure is due to limited resolution of the computer.

Quantization can occur due to (inherent limitations in resolution)

- sensors (e.g. encoders)
- sampling devices (e.g. analog to digital converter)
- limited resolution and range (overflow/underflow) in computations within the computer
- output devices (e.g. digital to analog converter)

Sampling period:

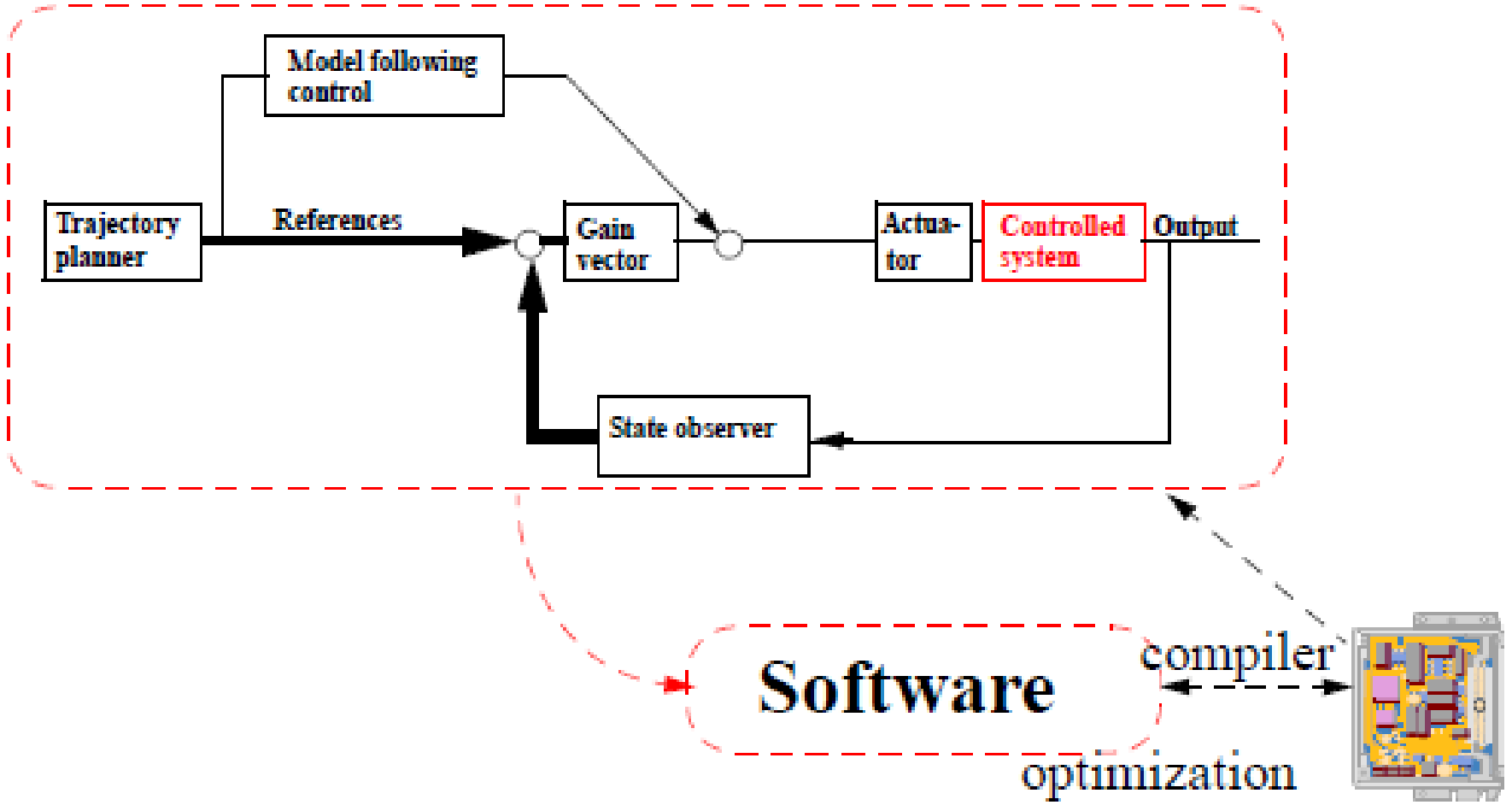
- Basic rules of thumbs
- BUT also needs to consider implementation constraints!

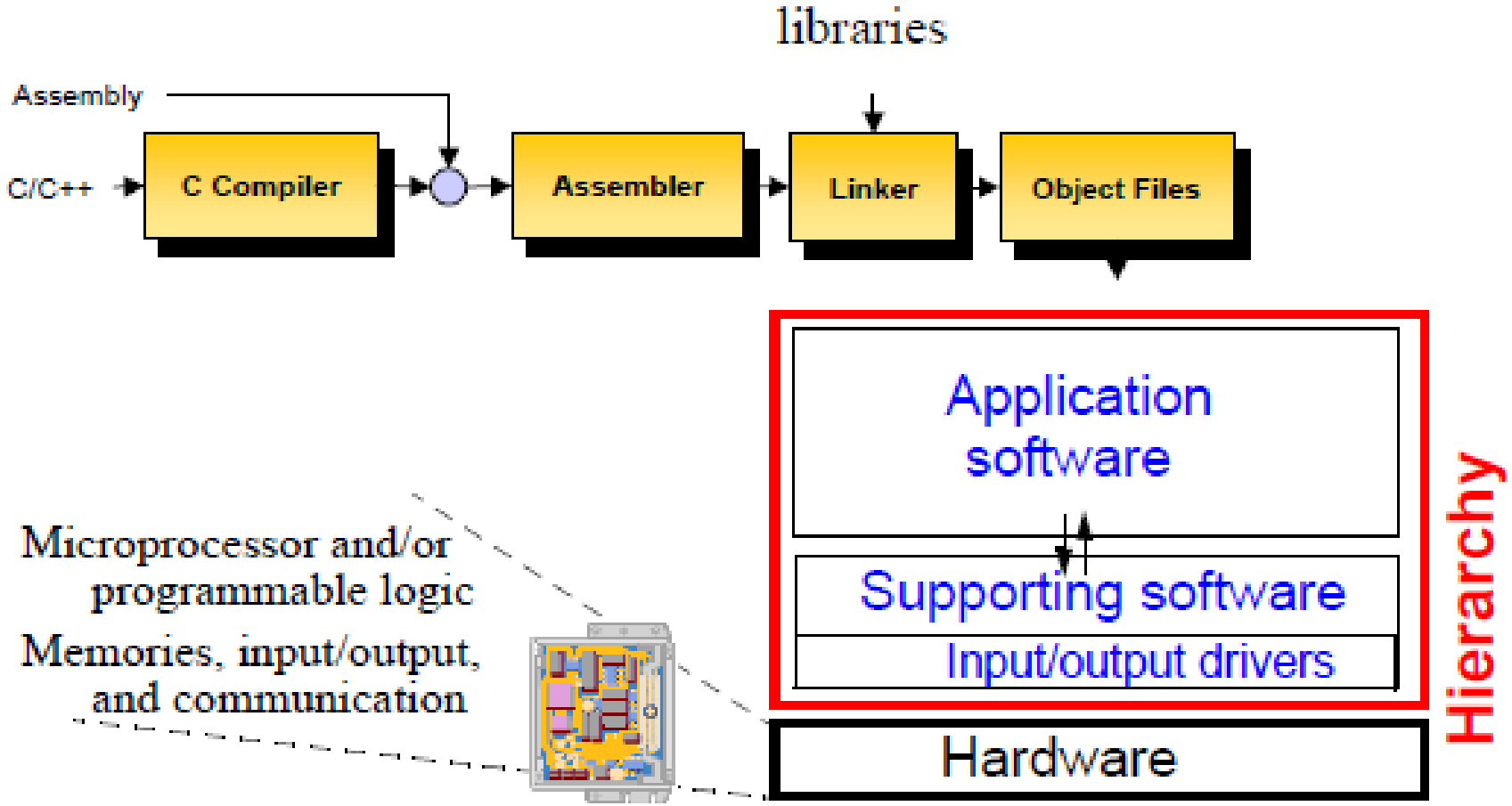
Feedback delay

- Minimize, or if this is not possible, keep constant
- Can compensate for constant delay
- Last resorts: Minimize jitter and use more elaborate compensation

Quantization

- Have to identify and analyse effects
- Floating vs. fixed point computation is a special issue


Model following control
Trajectory planner
References
Gain vector
Actua-tor
Controlled system
Output
State observer
Software
compiler
optimization


libraries
Assembly
C/C++
C Compiler
Assembler
Linker
Object Files
Application software
Supporting software
Input/output drivers
Hardware
Hierarchy
Microprocessor and/or programmable logic
Memories, input/output, and communication

- Purchasing an off-the shelf control system, for example, a
  - Programmable logic controller (PLC) or a
  - Motion controller

- Own design
  - Analog (low flexibility, only simple functionality)
  - Microprocessor based
  - Customized solution for large series:
    Application Specific Integrated Circuit (ASIC)
  - Flexible programmable hardware:
    Field Programmable Gate Array (FPGA)

- Choosing a microprocessor
  - Micro-controller / Digital signal processor / "General purpose"
  - Fixed point vs. floating point
  - I/O, memory and communication capabilities

The requirements are coupled and to some extent contradictory!

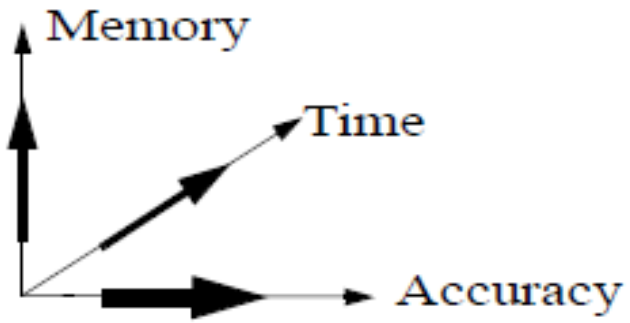
- Cost, reliability, flexibility, performance (speed, accuracy), ...

Potential implications of a "low cost" requirement:

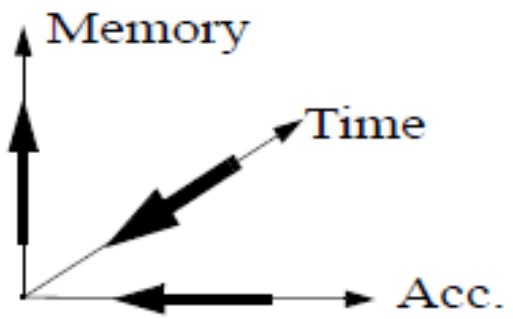
+ Cheap microcontroller
+ Use on-chip memory, no external circuits, to reduce cost (also robust)
- Scarce resources: Little room for later extensions
- Can performance requirements be met?
- Increased development (and possibly maintenance) cost

Choosing a highly performing processor:

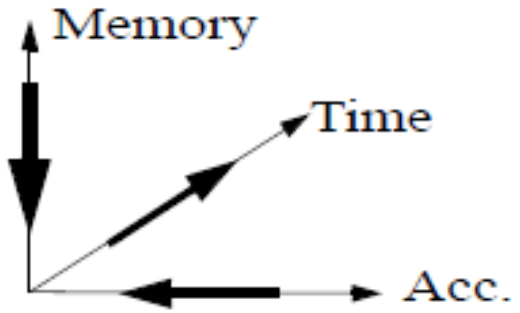
+ Easier to solve performance and flexibility requirements
- Increased production cost



(a) Increase in computational accuracy desired



(b) Increase in speed desired



(c) Scarce memory

Improved accuracy:

- better algorithm; typically requiring more execution time and memory

Improved timing:

- function in-lining; but requires more memory

Less memory usage:

- e.g. int16 rather then int32; reduced resolution, less accurate results

Extreme requirements may compromise good programming style;

Compare use of global varibles for speed-up rather than using call by value (benefits from tool support)

Factors affecting the execution time of a piece of code:

- The number of and types of instructions in the program.
  E.g. floating vs. fixed point computations.

- Data dependent selections and iterations in a program.
  -> varying execution time

- The compiler and linker libraries (other code) used

- The hardware speed (CPU itself and memory access)
  Specific hardware support, for example for computations
  Pipelines & caches increase average throughput but may cause a varying execution time - beware!

Thus in general: $C_{min} <= C <= C_{max}$

Goal: mix of desirable dynamics of the closed loop system and cost constraints.

$$C < T << t_{rise}$$

$$f_{bandwidth} << f < 1/C$$

C = execution time, $T = 1/f$ = sampling period, f = sampling frequency

Closed loop system: t_{rise} = rise time, $f_{bandwidth}$ = bandwidth

⇨ For a particular task, **the ratio C/T is defined as its CPU utilization.**

⇨ What will the feedback delay be of a control system implemented as one periodic activity on a microcontroller

Minimize computational delay by code structuring

- Task internal structuring
    1. *Sample*
    2. *Compute Control Output*
    3. *Actuate*
    4. *State update*

Recap

Control system implementation - overall perspective and issues

Controller implementation techniques and examples

- controller viewpoint
- implementation technology, trade-offs and optimization

More on control system implementation and tools

(1) S*ufficiently fast (processing, communication, ...)*

but this is not enough to guarantee real-time operation!

(2) Predictable and/or deterministic resource sharing and timing!

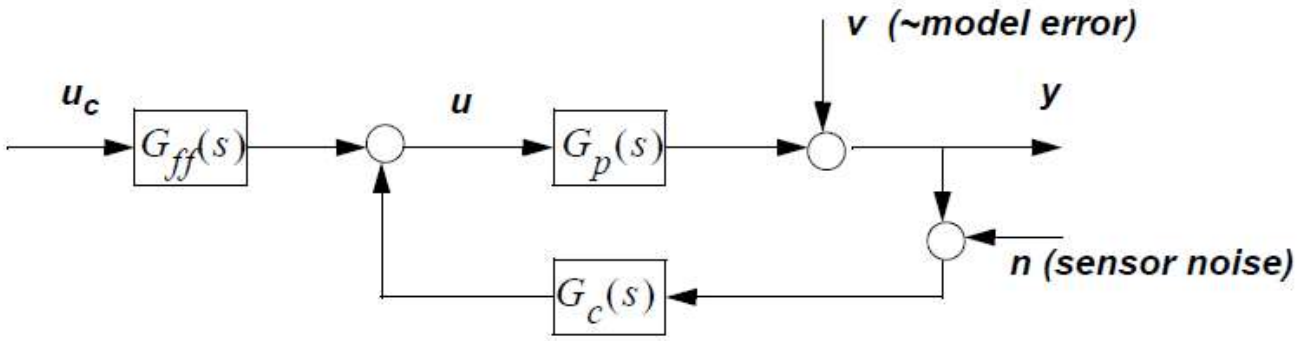
Requires well known and appropriate *mechanisms* and *policies* for *scheduling, communication, synchronization, error detection* and *error handling*

Requires bounded frequencies of external and internal events

- Characteristis of embedded (control) systems
  - ⇨ Terminology: distributed, real-time, and embedded systems
- From control design to real-time implementation:
  - ⇨ Sampling period selection
  - ⇨ Feedback delay
  - ⇨ Quantization
  - ⇨ Varying execution times, Processor utilization
  - ⇨ Dependencies between the parameters: C, T, τ
- Trade-offs!

- **The pole placement and model following design techniques do not take sensor noise and model parameter errors into account.**
- **Sensor noise is always a critical factor when analog sensors are used.**
- **The process model which is the base for the pole placement is always an approximation and has errors.**
  - Nonlinear effects
  - Varying parameters due to operation, e.g., temperature varying friction.
  - Varying operational conditions, e.g., a machine runs sometimes with and sometimes without payload.

- **Select $A_m(s)$ such that the c.l. response $y = \frac{Bt_0}{A_m} u_c$ has the desired properties of the c.l. response.**
- **Selecting the $A_o(s)$ part of the closed loop poles based on robustness and sensor noise.**
  - Introduction of the Sensitivity and Complementary sensitivity functions.
  - Comparison with state feedback.



- **Uncertainty modelling of the real process,** $G_p(s) = G_p^n(s)[1 + \Delta_p(s)]$

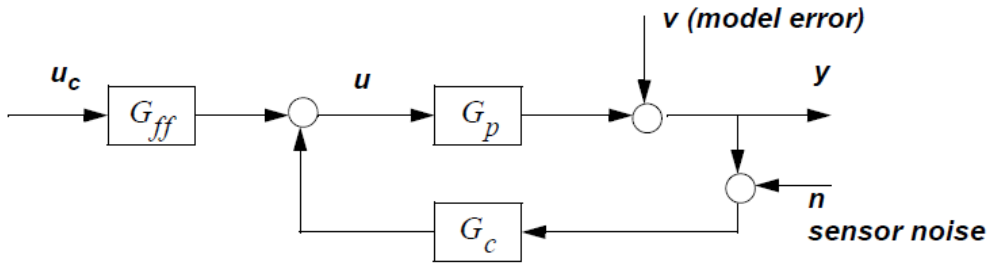
  $G_p^n(s)$ is the nominal model used in control design. $\Delta_p(s)$ the relative error.

- **Design goals**

  - low influence of $\Delta_p(s)$, (low gain from *v* -> *y*), $\Delta_p(s)$ is unknown
  - low high frequency gain from *n* -> *y*, sensor noise normally high frequency
  - high gain (bandwidth) from *r* -> *y*, servo performance

- What are the requirements on $G_c(s)$ to achieve the goals ?

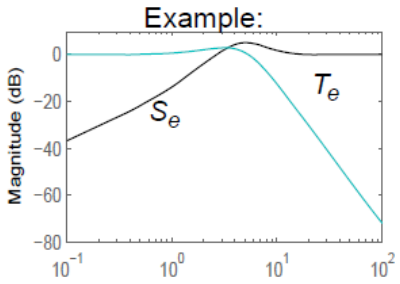
Sensitivity Function



$$y = \frac{G_p G_{ff}}{1 + G_p G_c} r + S_e v + (1 - S_e) n$$ where: $$S_e(s) = \frac{1}{1 + G_p G_c}$$

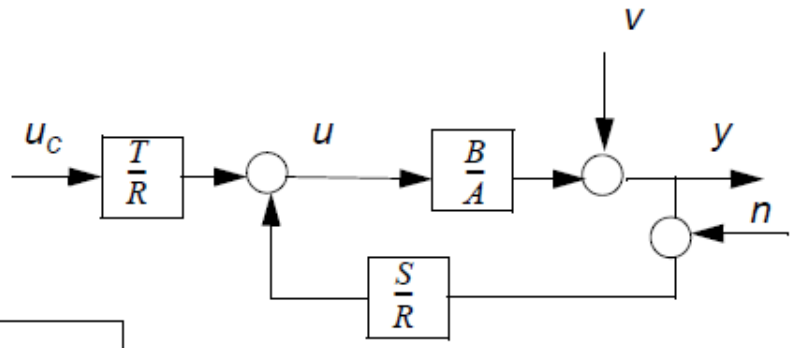
S_e is called the Sensitivity function and $T_e = 1 - S_e$ the complementary sensitivity function.

Regulator design is a compromise between S_e, model errors and T_e, sensor noise.



Control law

$$u(s) = \frac{T}{R}u_c - \frac{S}{R}y$$



Closed loop response

$$y = \frac{BT}{AR+BS}u_c + \frac{AR}{AR+BS}v - \frac{BS}{AR+BS}n$$

Pole placement gives

$$AR + BS = A_m A_o$$

Select $T = A_o t_0$

Then

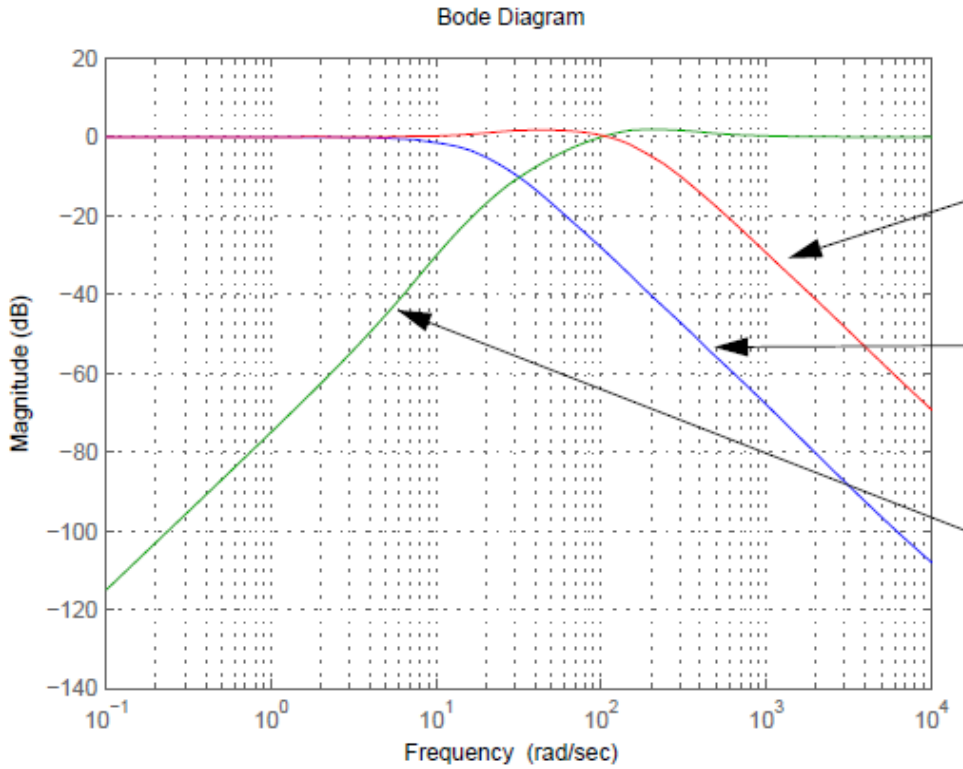
$$y = \frac{Bt_0}{A_m}u_c + \frac{AR}{A_m A_o}v - \frac{BS}{A_m A_o}n$$

$$y = \frac{Bt_0}{A_m}u_c + S_e v - T_e n$$

Select A_m and t_0 to get specified response from u_c.

Select A_o to get specified response from v and n.

The response from command signal u_c, and the response from noise n, and disturbance v, can be independently designed.



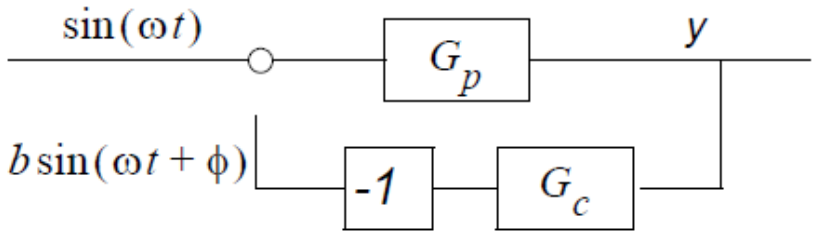
From n to y, $T_e(s)$

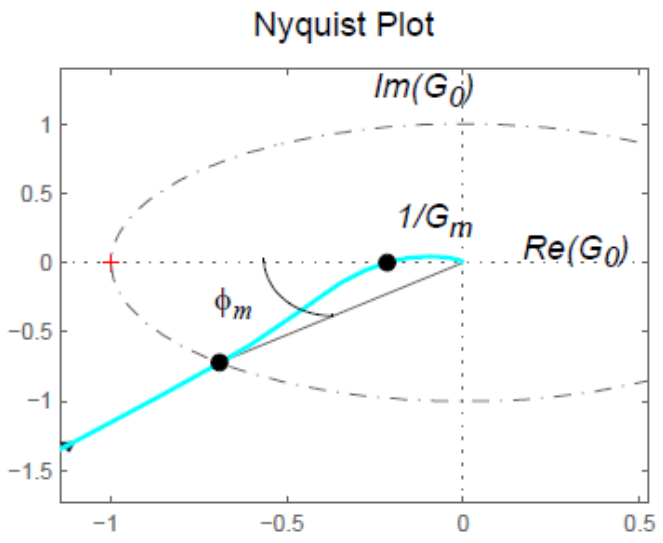
From u_c to y, $\frac{Bt_o}{A_m}$

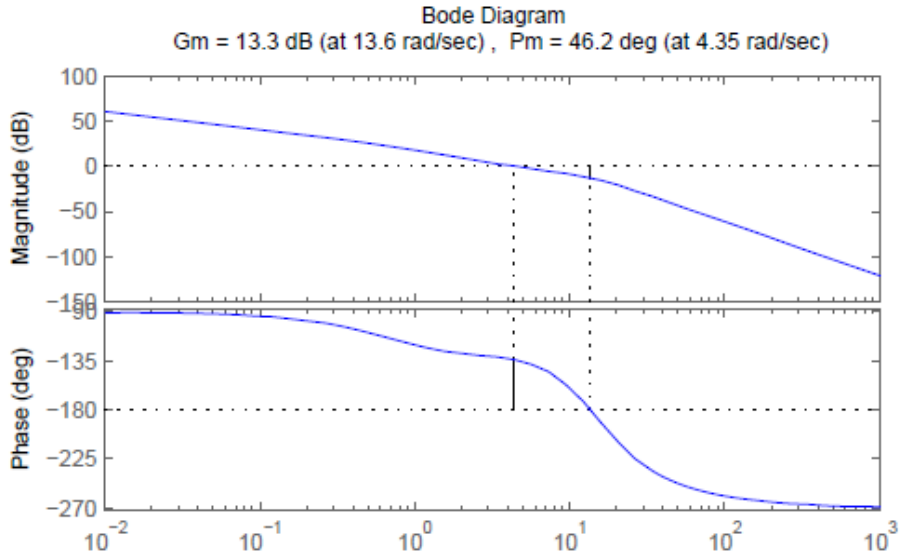
From v to y, $S_e(s)$

-The phase margin, ϕ_m and gain margin, G_m can be shown in the Bode and Nyquist plots.

$-G_0 = -G_p G_c$ can be ϕ_m and G_m wrong without instability.





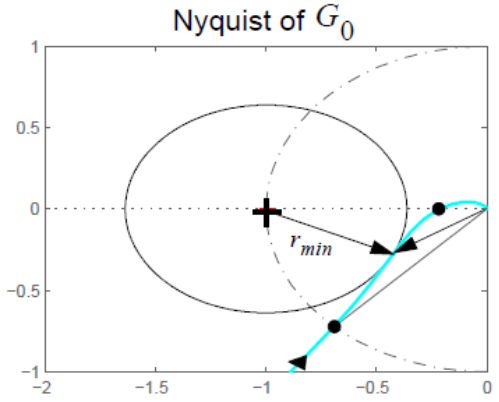


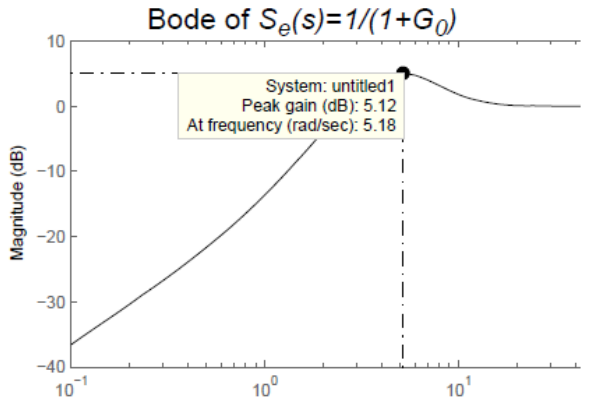
Sensitivity function as stability margin

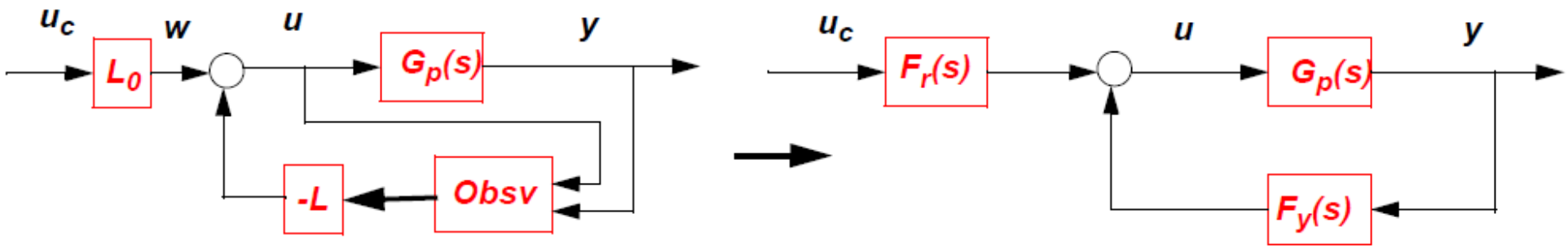
Radius of the circle with centre in (-1,0) that is tangent to the loop gain $|G_0|$ is $r = |1 + G_0|$, the minimum radius, $r_{min} = |1 + G_0|_{min}$ can be used as a measure on the stability margin against parameter errors in the design model.

$|S_e|_{max} = |1/(1 + G_0)|_{max} = |1/r_{min}|$, $|S_e|_{max}$ is the same as stability margin.

Typical specification $|S|_{max} < (2 - 5)\text{dB}$.







The state observer $\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x})$ with control law $u = -L\hat{x} + w$, gives

$$u = -\{L(sI - A + BL + KC)^{-1}K\}y + \{1 - L(sI - A + BL + KC)^{-1}B\}L_0 u_c = -F_y y + F_r u_c$$

see page 186 in Glad Ljung for a derivation.

AND: $S_e(s) = \dfrac{1}{1 + G_p F_y}$ and $T_e(s) = \dfrac{G_p F_y}{1 + G_p F_y}$

If the order of the polynomials $S(s)$, $T(s)$ and $R(s)$ are selected as, $\deg R = \deg G_p$ $\deg S = \deg T = \deg G_p - 1$.

Then:

1.) $F_y(s) = \frac{S(s)}{R(s)}$ and $F_r(s) = \frac{T(s)}{R(s)}$.

2.) The poles of the c.l. with state feedback $(A - BL)$ are the same as the c.l. polynomial $A_m(s)$, and the poles of the observer $(A - KC)$ are the same as for the polynomial $A_o(s)$.

Example: Selecting $\frac{S}{R} = \frac{s_1 s + s_0}{s^2 + r_1 s + r_0}$ and $T = A_o t_0$ for $G_p(s) = \frac{b}{s(s+a)}$, with

$A_m(s) = \det(sI - A + BL)^{-1}$ and $A_o(s) = \det(sI - A + KC)^{-1}$. Gives identical controllers.

However: selecting $R(s)$ one order higher than $S(s)$ i.e. designing a full state observer may give a discrete time controller with one sample delay.

Instead if, $\deg S = \deg R = \deg T = \deg G_p - 1$ then there is no delay in the controller

and if $A_m(s) = \det(sI - A + BL)^{-1}$ and $A_o(s) = \det(sI - A + KC)^{-1}$ where the observer is designed as a reduced order observer (Lueneburger observer) then again the two designs are identical. (see page 182 in Glad Ljung)

Advantage:

- no delay in discrete time
- one order lower controller gives less computations on the processor.
- simpler and more intuitive to include integral action in the controller.
- straight forward solution to the reduced observer comp. to s.s. design.

- Position controller using PID feedback $G_c(s) = \left(P + Ds + I\frac{1}{s}\right)$
- Process model $\frac{B}{A} = \frac{b}{s(s+a)}$
- Write it on polynomial form with a filter in denominator to get a proper feedback

  where $\deg S = \deg R$, $\frac{S(s)}{R(s)} = \frac{s_2 s^2 + s_1 s + s_0}{s(s+r_0)}$.
- The Integral term increases the order of the polynomials $S(s)$, $T(s)$ and $R(s)$ with one compared to the order without integral action.
- C.l. $AR + BS = s^4 + (r_0 + a)s^3 + ar_0 s^2$, 4:th order polynomial

- The c.l. transfer function is $y = \frac{BT}{AR+BS}u_c + \frac{AR}{AR+BS}v + \frac{BS}{AR+BS}n$.

- Select $AR+BS = A_m A_o$ and $T = A_o t_0$ where $A_m(s) = s^2 + 2\zeta_m \omega_m s + \omega_m^2$ and $A_o(s) = s^2 + 2\zeta_o \omega_o s + \omega_o^2$.

- Calculate t_0 by computing the dc-gain from u_c to y,

  $y = \frac{BT}{AR+BS}u_c = \frac{BA_o t_0}{A_m A_o}u_c = \frac{Bt_0}{A_m}u_c = \frac{Bt_0}{s^2 + 2\zeta_m \omega_m s + \omega_m^2}u_c$, selecting $t_0 = \frac{\omega_m^2}{B}$ gives

  $y = \frac{\omega_m^2}{s^2 + 2\zeta_m \omega_m s + \omega_m^2}u_c$ which has unit gain.

- The complete response becomes $y = \frac{\omega_m^2}{A_m} u_c + \frac{AR}{A_m A_o} v + \frac{BS}{A_m A_o} n$.

- Specifications:

  - *Rise time* = *0.2 s* (5% - 95% of final value) without saturation!

    -> $\omega_m = 30, \zeta_m = 0.9$

  - There is electric noise from the sensor at *50 Hz*.

    -> $T_e(2\pi 50j) < 20dB$

  - The inertia can change from a nominal value to twice this value due to changes in payload.

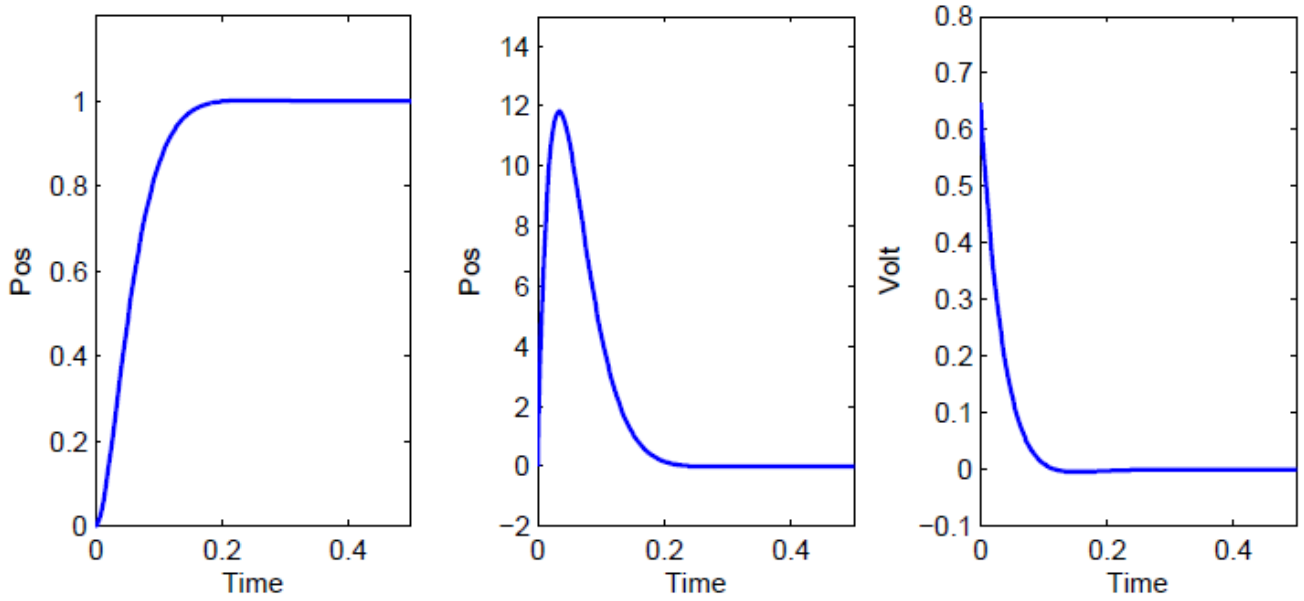
    -> Sufficiently phase and gain margins $|S_e|_{max} < Xdb$

- $A_m(s)$ is given from rise time.

- Test different observer polynomials $A_o(s) = s^2 + 2\zeta_o\omega_o s + \omega_o^2$ if it is possible to satisfy the other two specifications.

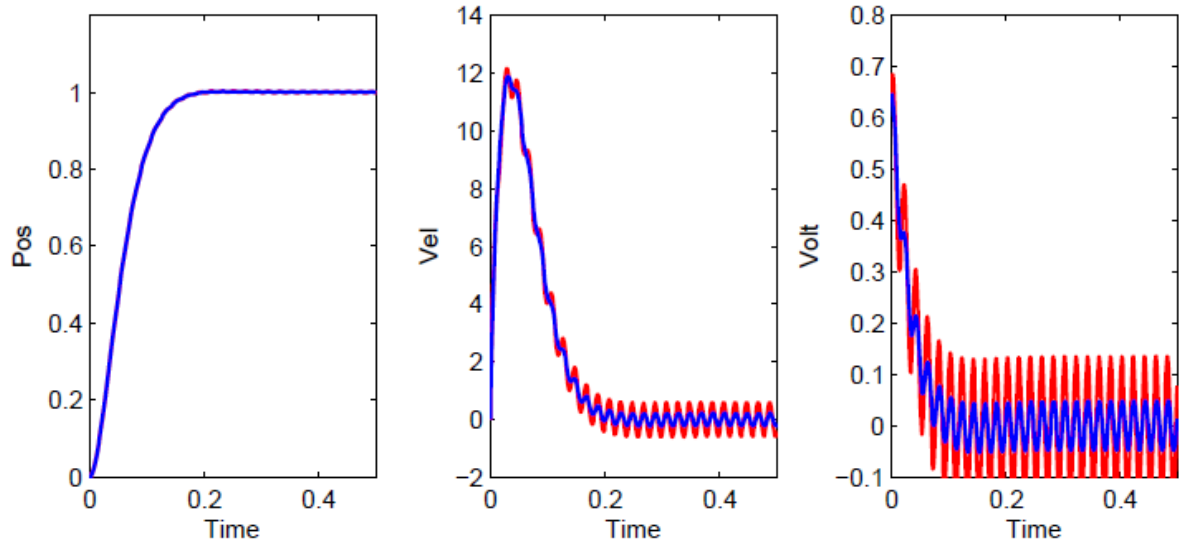
- The nominal step response without sensor noise and increased inertia does

not depend on $A_o(s)$! $y = \frac{BT}{AR + BS} = \frac{\omega_m^2}{s^2 + 2\zeta_m\omega_m s + \omega_m^2}$

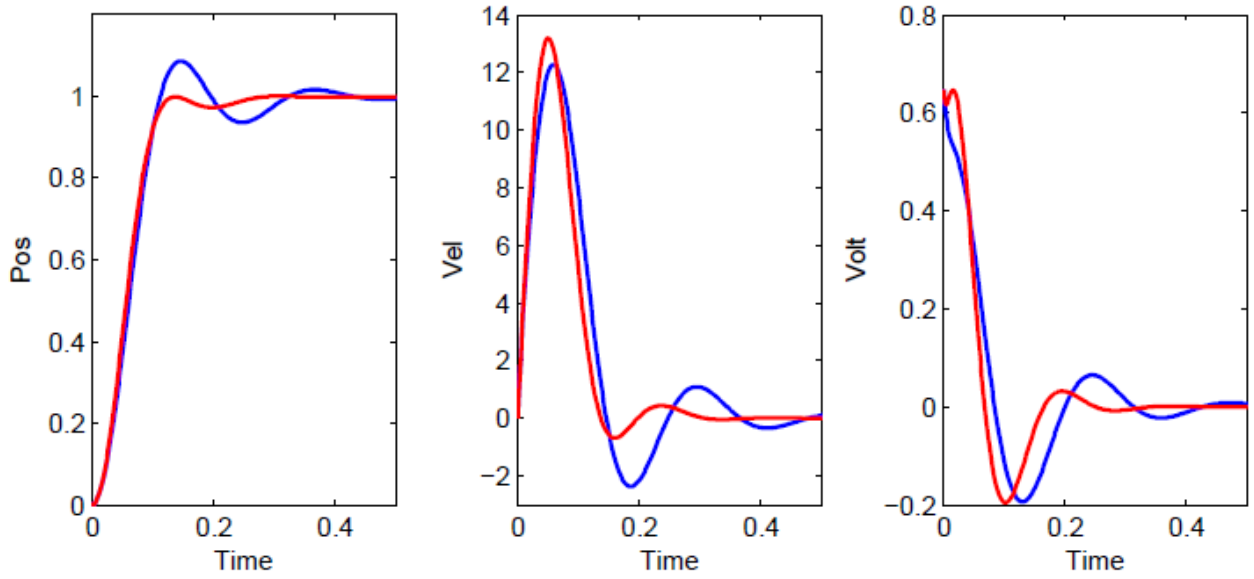


- Rise time OK!

- Design two different observer polynomial, $\omega_o = 2\omega_m$ and $\omega_o = 4\omega_m$.
- 1.) Inspect the step response with sensor noise, $n = 0.01\sin(2\pi 50t)$
- The red line is with the faster observer and the blue line with the slower observer.

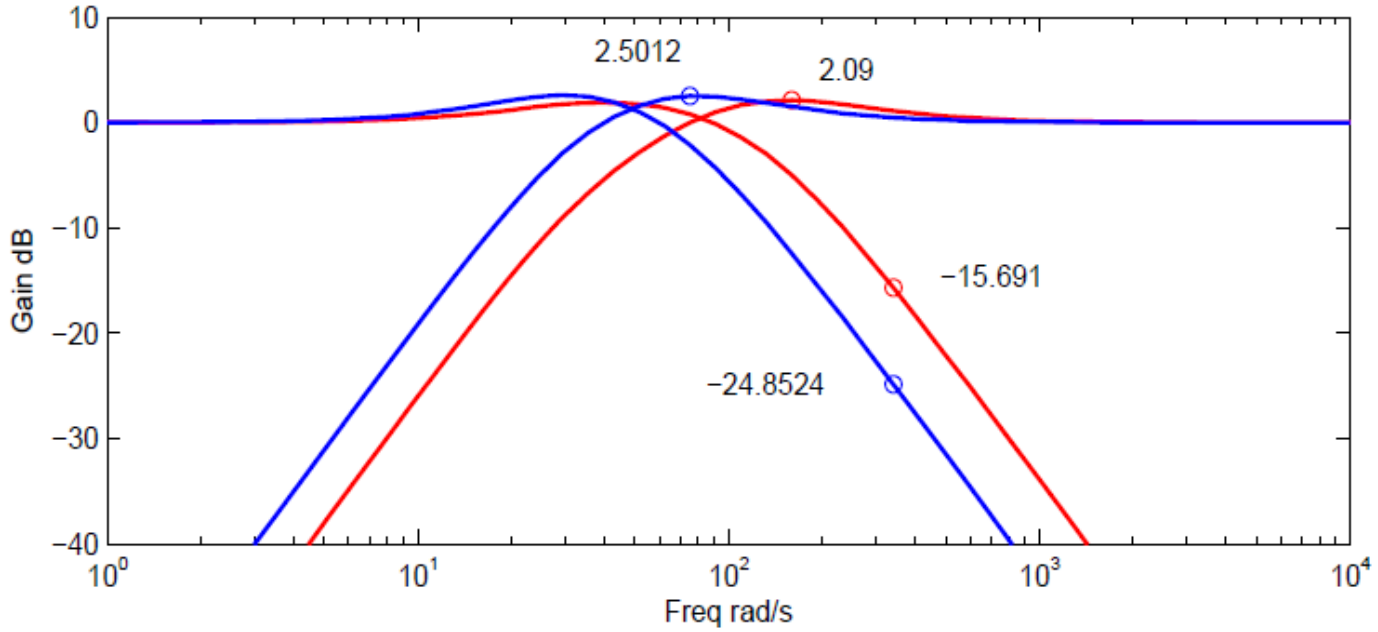


- 2.) Inspect the step response when the inertia is increased twice. This could e.g. happen when the machine picks up a payload.
- Red line is the fast observer polynomial. Blue line is the slow observer polynomial.



- Calculate the Sensitivity function $S_e(s) = \frac{AR}{AR + BS}$ and the Complementary Sensitivity function $T_e(s) = 1 - S_e(s) = \frac{BS}{AR + BS}$.

- Blue line is the slower and red line the faster observer. *(50 Hz = 314 rad/s)*



Robust control design methods

- **Find a fixed parameter controller that satisfies some c.l. specifications for a process with uncertain physical parameters.**
  - Example: $G_p(s, k, \tau) = \frac{k}{\tau s + 1}$ where $k \in [1, 5]$ and $\tau \in [0.01, 0.05]$.
- **Frequency plane, Quantified feedback theory, QFT.**

  A good reference is:
  Quantitative feedback design of linear and nonlinear control systems
  Oded Yaniv
  Kluwer Academic Publisher, ISBN 0-7923-8529-2
- **Complex plane, using pole region assignment**

  A good reference is:
  Robust Control -systems with uncertain physical parameters
  Jurgen Ackerman Springer-Verlag, ISBN 0-387-19843-1

- Regulator design is about giving and taking
  -better robustness against parameter variations gives higher sensitivity to noise, and the other way around.
- Tune $A_o(s)$ to achive an appropriate compromise.
- Pole placement can be done with state space models or with transfer functions. The result can depending on the choice of $S(s)$ and $R(s)$ be the same.
- Selecting $\deg S = \deg R = \deg G_p - 1$ gives something similar to a reduced order observer and less delay in the implementation.

References

Jan Wikander,Bengt Eriksson, KTH, Machine Design, Mechatronics Lab