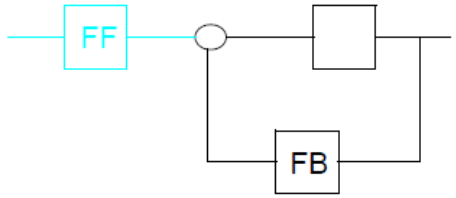
SERVO MOTORS AND MOTION CONTROL SYSTEMS

(Motion Control Systems: Theoretical Background: «Feedback Control»-Lecture 7)

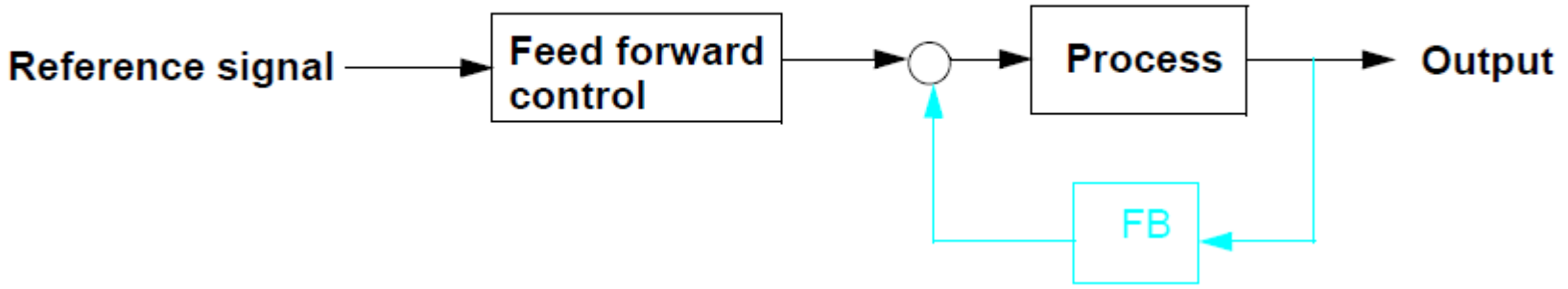
Dr. Caglar UYULAN

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
- May lead to instability if designed incorrectly
- Sensor noise may be amplified and deteriorate performance



Feedforward control properties



- Produces control signals prior to that an error has occurred
- Uses carefully designed reference signals to make the process follow the references “exactly”

- The **regulator** problem: -> FEEDBACK

  Find a feedback controller that satisfies the specifications on

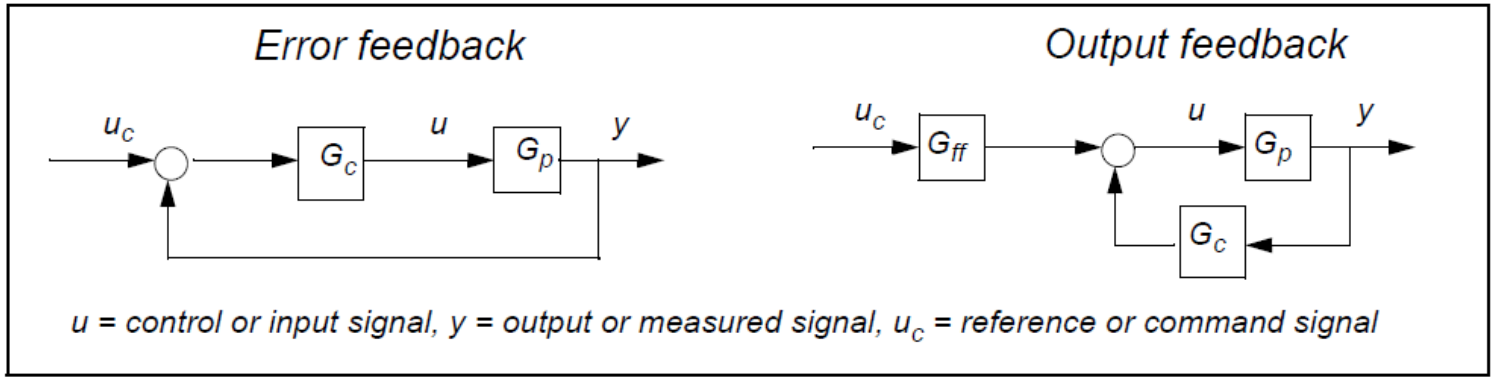
  -sensor noise, disturbance rejection and robustness to model and parameter uncertainties

- The **servo** problem: -> FEED FORWARD

  Find a feed forward controller that **tracks the references** according to specifications (a feedback must already exist)

  -Steady state accuracy, overshoot, tracking error, settling time

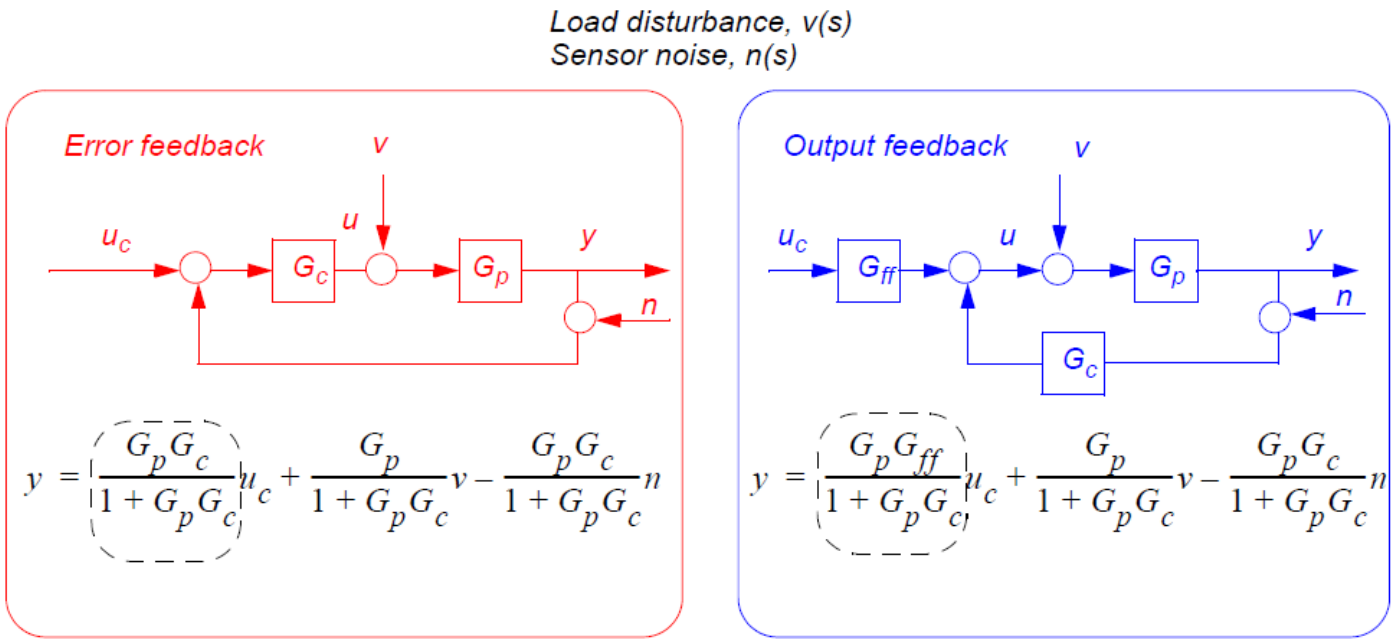
- More design freedom with Output feedback, also called 2DOF control



- Example PD-control

  If, $G_c(s) = G_{ff}(s) = P + Ds$, then both structures are equal

  Example. If we don't want derivative action on the reference signal we should instead choose $G_{ff}(s) = P$ and $G_c(s) = P + Ds$.



- **TF from reference, u_c to output *y* are different.**

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

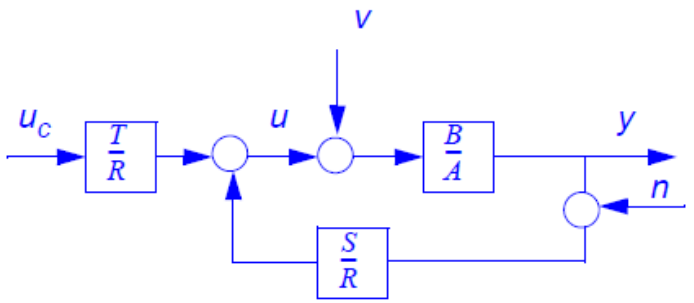
$G_c(s) = \frac{S(s)}{R(s)}$

$G_{ff}(s) = \frac{T(s)}{R(s)}$

Control law

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

Output feedback



Closed loop respones

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

For error feedback is, $T = S$

- **Model based T.F control design**

DC-motor model from lec. 2 $J\ddot{\varphi} = k_m i - d\dot{\varphi}$

set $y = \varphi$ and $u = i$ s.t. $G_p(s) = \frac{y(s)}{u(s)} = \frac{k_m}{Js^2 + ds}$

$J = 0.08$: rotor inertia
$k_m = 3.6$: torque constant
$d = 0.45$: friction coefficient
i: rotor current
φ: angular position

1.) Start simple, try position feedback $u = P(u_c - y)$

2.) Calculate c.l. poles

with: $G_p(s) = \frac{B(s)}{A(s)}$ and $G_c(s) = \frac{S(s)}{R(s)} = \frac{P}{1}$ and $G_{ff} = \frac{T(s)}{R(s)} = \frac{P}{1}$

Closed loop poles, $G_{yu_c} = \frac{BT}{AR + BS} = \frac{k_m P}{Js^2 + ds + Pk_m}$

-> 2:nd order poly. -> solve for $s_{1,2}$ from $Js^2 + ds + Pk_m = 0$.

- **Plot the c.l. poles as a function of one variable**

  the variable could be either a control parameter or a process parameter

  Here we choose P as the varying variable. **Matlab code for the calculations:**

```
m = 0.08; d = 0.45; km = 3.6;
Pvec = 0:.01:1;
for n = 1:length(Pvec), Poles(:,n) = roots([J d Pvec(n)*km]);end
plot(real(Poles(1,:)),imag(Poles(1,:)),'r.',real(Poles(2,:)),imag(Poles(2,:)),'b.')
sgrid
```

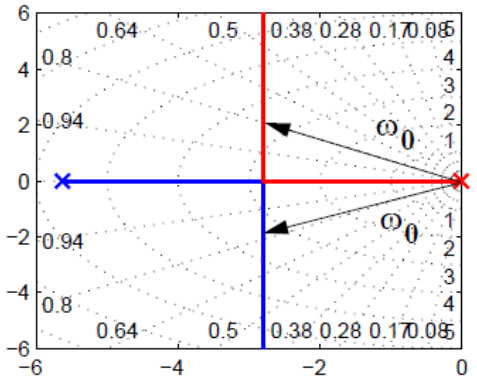
Normally we don't want to have a step response with an overshoot. A robot arm could collide with an object!

Choose fastest poles with $\zeta > 0.8$ -> $P = 0.27$

which gives $\omega_0 = 3.49$

Try:

```
G = tf(km,[J d 0]); rlocus(G,Pvec),P = rlocfind(G)
```

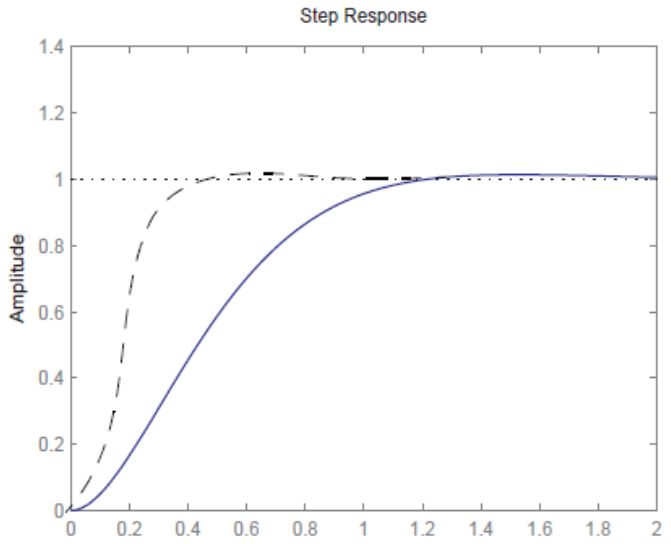


- **Closed loop** $G_{yu_c} = \dfrac{k_m P}{Js^2 + ds + k_m P} = \dfrac{(k_m P)/J}{s^2 + (d/J)s + (k_m P)/J} = \dfrac{\omega_0^2}{s^2 + 2\zeta\omega + \omega_0^2}$

  where $\omega_0 = 3.49$ and $\zeta = 0.806$

But what if we want a faster step response, higher ω_0 with the same $\zeta > 0.8$??

Try position and velocity feedback!



- New control law $u = -P(u_c - y) - D\frac{dy}{dt}$ -> $G_c(s) = \frac{S}{R} = \frac{Ds+P}{1}, G_{ff}(s) = \frac{T}{R} = \frac{P}{1}$

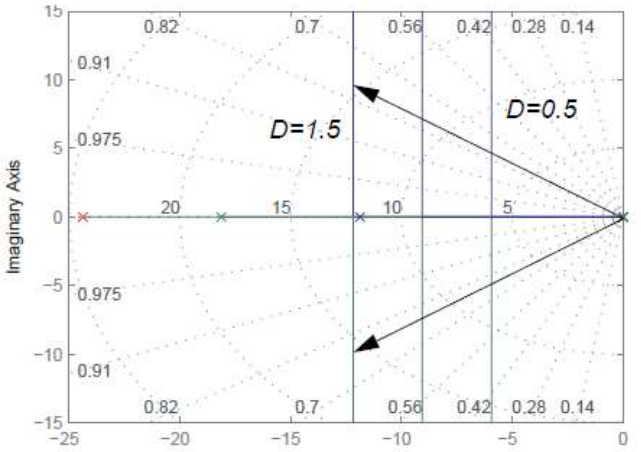
- How do we choose D?

$$G_{yu_c} = \frac{BT}{AR+BS} = \frac{k_m P}{Js^2 + (d + Dk_m)s + k_m P}$$

a root locus can not be done on both P and D at the same time lets try multiple root locuses on P with D ={0.5 1.0 1.5};

With $D = 1.5$ we can choose $(\omega_0, \zeta) = [15, 0.82]$ which is 5 times faster than without velocity feedback with the same ζ.

Is there a way to get any desired speed ω_0, and damping ζ ?



- **Solving for s in the c.l. denominator polynomial with position and velocity feedback gives** $A_{cl} = AR + BS = s^2 + \frac{(d + k_m D)}{J} s + \frac{k_m}{J} P = 0$

  - 2 control parameters and a second order polynomial, that is, we can choose **any** c.l. poles by selecting P & D in a proper way such that.

$$A_{cl}(s) = A_m(s)$$

where A_m is the desired closed loop polynomial e.g. $A_m = s^2 + 2\zeta_m \omega_m s + \omega_m^2$.

This gives

$$\frac{(d + k_m D)}{J} = 2\zeta\omega_m s \quad \text{->} \quad D = -d + 2\zeta\omega_m J$$

Use *solve* in Maple

$$\frac{k_m}{J} P = \omega_m^2 \qquad P = \frac{\omega_m^2 J}{k_m}$$

Pole Placement Design

Model: $\dot{x} = Ax + Bu$, where $x = [x_1, x_2, \ldots, x_n]^T$.

Control law: $u = -Lx + w$, where $L = [l_1, l_2, \ldots, l_n]$

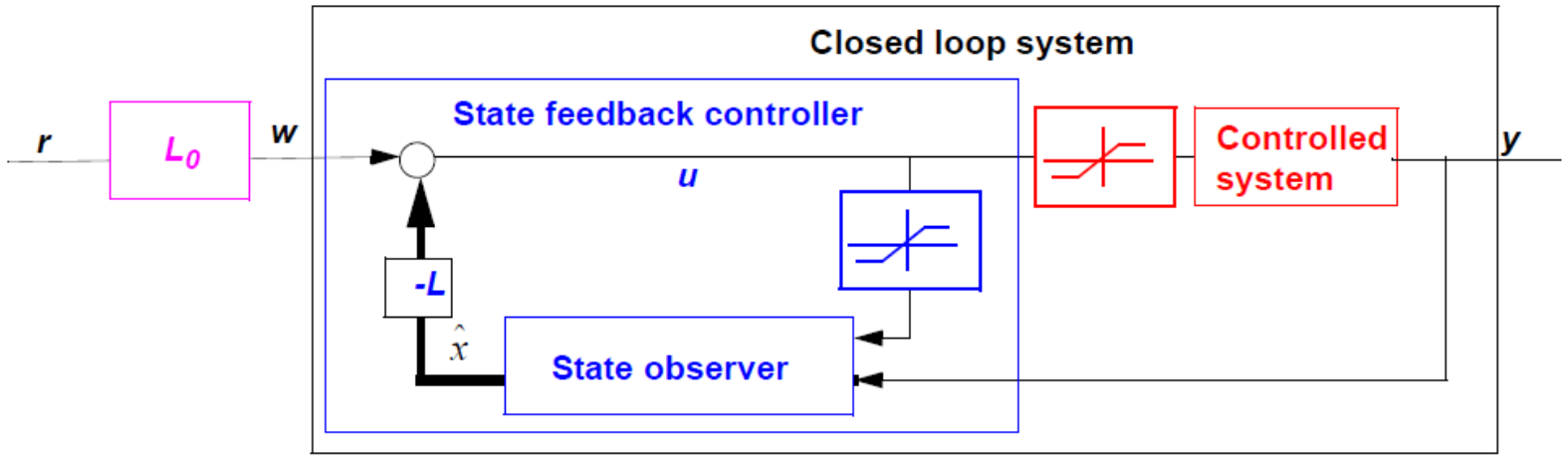
Closed loop: $\dot{x} = Ax + Bu = Ax - BLx + Bw = (A - BL)x + Bw$

The poles of the c.l. are totally defined by *L*, eigenvalues of the matrix, $(A - BL)$
L is easiest found numerically in Matlab using the 'acker' command.

Advantage, easy to calculate *L* for for any model, also high order.

The state vector *x* must be available from measurements or from designing a state observer.

$\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x}) = (A - KC)\hat{x} + Bu + Ky$, design *K* in the same way as *L*.



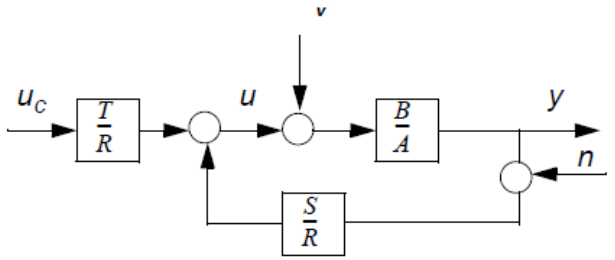
Calculate the state space model of the DC-motor, choose $x_1 = \varphi, x_2 = \dot{\varphi}$

which gives $A = \begin{bmatrix} 0 & 1 \\ 0 & -d/J \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ k_m/J \end{bmatrix}$ and $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

```
A = [0 1;0 -d/J]; B = [0;km/J]; C = [1 0]; D = 0; Gss = ss(A,B,C,D);
w0 = 20; zeta = 0.8;
poles = roots([1 2*zeta*w0 w0^2])
poles =
 -16.0000 +12.0000i
 -16.0000 -12.0000i
L = acker(Gss.a,Gss.b,poles)      % OBS L(1) =  P,  and L(2) = D
L =
    8.8889    0.5861
damp(A-B*L)
        Eigenvalue            Damping      Freq. (rad/s)

 -1.60e+001 + 1.20e+001i     8.00e-001       2.00e+001
 -1.60e+001 - 1.20e+001i     8.00e-001       2.00e+001
```

1.) Select control structure, $\frac{S(s)}{R(s)}$.



2.) Calculate the c.l. polynomilal
$A_{cl}(s) = AR + BS$ (characteristic equation).

3.) Select a desired c.l. polynomial,
$A_d(s) = A_m(s)A_o(s)$ where $\deg(A_d) = \deg(A_{cl})$

and $\deg(A_m) = \deg(A)$, which gives $\deg(A_o) = \deg(A_d) - \deg(A_m)$.

4.) Solve for the parameters in $R(s)$ and $S(s)$ in the so called Diophantine eq.
$A_{cl}(s) = A_m(s)A_o(s)$.

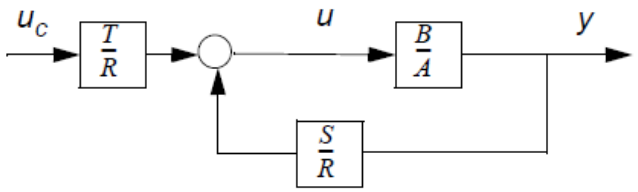
5.) Set the f.f. polynomial to $T(s) = t_0 A_o(s)$ where t_0 is a static gain that gives unit dc-gain in the c.l. T.F. from u_c to y.

$$G_{yu_c}(s) = \frac{BT}{AR + BS} = \frac{BT}{A_{cl}} = \frac{Bt_0 A_o}{A_m A_o} = \frac{Bt_0}{A_m}$$

, chose t_0 such that,

$$\frac{1}{t_0} = \frac{B(s)}{A_m(s)}\bigg|_{s=0} = \frac{B(0)}{A_m(0)} .$$

i.) the order of $G_{yu_c}(s)$ is the same as $A_m(s)$ and thereby also the order of the process $A(s)$, (see last slide).



ii.) The dc gain is one, $G_{yu_c}(0) = 1$.

$$G_p(s) = \frac{k_m}{Js^2 + ds} = \frac{3.6}{0.1s^2 + 0.45s} = \frac{36}{s^2 + 4.5s}$$

$J = 0.1$: rotor inertia

$k_m = 3.6$: torque constant

$d = 0.45$: friction coefficient

1.) PD-control with l.p. filter structure $\quad \frac{S(s)}{R(s)} = \frac{s_1 s + s_0}{s + r_0}$

2.) C.l polynomial $\quad A_{cl} = AR + BS = s^3 + s^2(4.5 + r_0) + s(4.5r_0 + 36s_1) + 36s_0$

3.) Select desired C.l polynomial

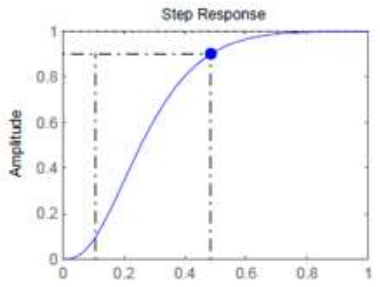
$$A_d = A_m A_o = (s^2 + 2\zeta\omega s + \omega^2)(s + \alpha)$$

From some specifications we want:

Rise time should be less than 0.5 s.

One possible selection of the

c.l. poles is $\quad \omega = 10, \zeta = 0.9 \;\; \alpha = 10$



4.) Diophantine equation $A_{cl} = A_d$

$$s^3 + s^2(4.5 + r_0) + s(4.5r_0 + 36s_1) + 36s_0 = s^3 + s^2(\alpha + 2\zeta\omega) + s(2\zeta\omega\alpha + \omega^2) + \omega^2\alpha$$

gives:

$$4.5 + r_0 = 28$$
$$4.5r_0 + 36s_1 = 280$$
$$36s_0 = 1000$$

$$r_0 = 23.5$$
$$s_0 = 27.8$$
$$s_1 = 4.8$$

$$\rightarrow \quad \frac{S}{R} = \frac{4.8s + 27.8}{s + 23.5}$$

Check! roots of :

$$AR + BS = 0$$

5.)

Feed forward part $T(s) = A_o t_0$

T.F from reference to output $G_{yu_c}(s) = \frac{BT}{AR + BS} = \frac{Bt_0}{A_m} = \frac{36t_0}{s^2 + 28s + 100}$

calculate $t_0 = \frac{100}{36}$ and $T(s) = 2.8s + 28$

gives $G_{yu_c}(s) = \frac{100}{s^2 + 28s + 100}$

with the control law $u(s) = \frac{T}{R}u_c - \frac{S}{R}y = \frac{2.8s + 28}{s + 23.5}u_c - \frac{4.8s + 27.8}{s + 23.5}y$

- Normally we need the order of $R(s)$ to be at least the same as for $S(s)$. This gives a proper t.f. $G_c(s) = \frac{S}{R}$. (the order of the numerator is not higher than that of the denominator).
- PD type controllers can however be used. (derivation of position to velocity)
- A time delay of at least one sample will be introduced if the order of $R(s)$ is higher then $S(s)$.
- A good choice is thereby to have the same order of $S(s)$ and $R(s)$, and if the order of $S(s)$ is one less than $A(s)$ then complete control in terms of poles and their c.l. locations is possible.
- Which order is then good to use? -depends on the control problem such as: Integral control, sensor noise, disturbances etc.

Process: $G_p = \frac{B}{A} = \frac{b}{s+a}$, and the Diophantine exp. $AR + BS = A_m A_o$

P-ctrl. $G_c = \frac{S}{R} = \frac{s_0}{1}$ Dio. $s + a + s_0 = s + \alpha \quad (A_o = 1)$

P-ctrl. with LP-filter. $G_c = \frac{S}{R} = \frac{s_0}{s + r_0}$ Dio. $s^2 + (r_0 + a)s + (bs_0 + ar_0) = (s + \alpha)(s + \beta)$

PI-ctrl. $G_c = \frac{S}{R} = \frac{s_1 s + s_0}{s}$, Dio. $s^2 + (a + bs_1)s + bs_0 = (s + \alpha)(s + \beta)$

PI-ctrl. with LP-filt.

$G_c = \frac{S}{R} = \frac{s_1 s + s_0}{s(s + r_0)}$, Dio. $s^3 + (a + r_0)s^2 + (bs_1 + ar_0)s + bs_0 = (s^2 + 2\zeta\omega s + \omega^2)(s + \alpha)$

Process: $G_p = \frac{B}{A} = \frac{b}{(s+a)s}$ c.l. dynamics $AR + BS$ and the Diophantine exp.

PD-ctrl. $G_c = \frac{S}{R} = \frac{s_1 s + s_0}{1}$ Dio. $s^2 + (bs_1 + a)s + bs_0 = s^2 + 2\zeta\omega s + \omega^2$

PD-ctrl. with LP-filter. $G_c = \frac{S}{R} = \frac{s_1 s + s_0}{s + r_0}$

Dio. $s^3 + (a + r_0)s^2 + (bs_1 + ar_0)s + bs_0 = (s^2 + 2\zeta\omega s + \omega^2)(s + \alpha)$

PID-ctrl. $G_c = \frac{S}{R} = \frac{s_2 s^2 + s_1 s + s_0}{s}$,

Dio. $s^3 + (a + bs_2)s^2 + bs_1 s + bs_0 = (s^2 + 2\zeta\omega s + \omega^2)(s + \alpha)$

PI-ctrl. $G_c = \frac{S}{R} = \frac{s_1 s + s_0}{s}$,

Dio. $s^3 + as^2 + bs_1 s + bs_0 = (s^2 + 2\zeta\omega s + \omega^2)(s + \alpha)$

(not solvable! 3:rd order Dio. with only two control parameters)

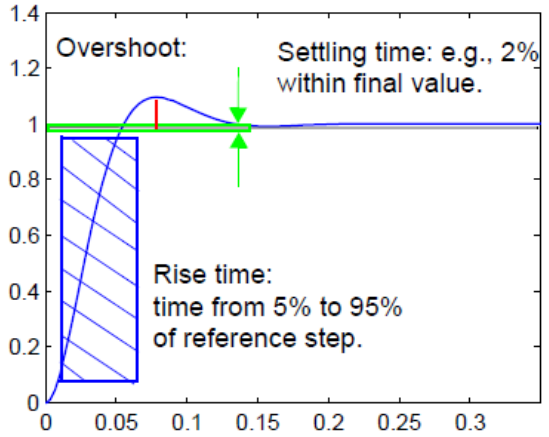
PI-ctrl. with LP-filt. $G_c = \frac{S}{R} = \frac{s_1 s + s_0}{s(s + r_0)}$, (observe $\deg R > \deg S$)

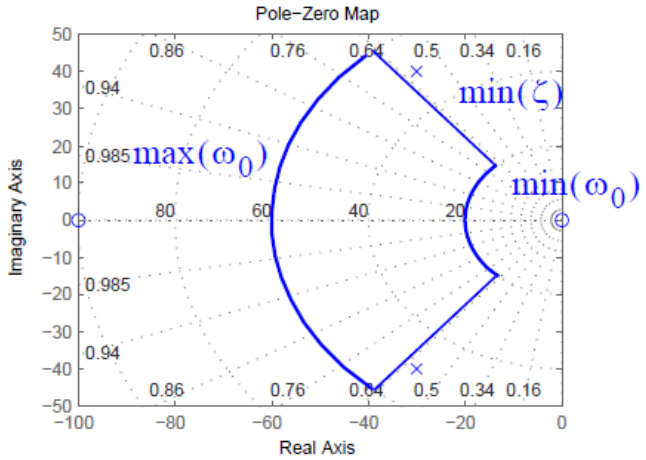
Dio. $s^3 + (a + r_0)s^2 + (bs_1 + ar_0)s + bs_0 = (s^2 + 2\zeta\omega s + \omega^2)(s + \alpha)$

PID-ctrl. with LP-filt. $G_c = \frac{S}{R} = \frac{s_2 s^2 + s_1 s + s_0}{s(s + r_0)}$

Dio. $s^4 + (r_0 + a)s^3 + (bs_2 + ar_0)s^2 + bs_1 s + bs_0 = (s^2 + 2\zeta_1\omega_1 s + \omega_1^2)(s^2 + 2\zeta_2\omega_2 s + \omega_2^2)$

- **Intuitively in time domain, but for design in complex plane.**
- **Need for translation between planes**





select c.l. poles:

uneven order process: $(s+\omega_0)$

even order process: $(s^2+2\zeta\omega_0 s+\omega_0^2)$

- The rise time for higer order systems will be slower (superposition)

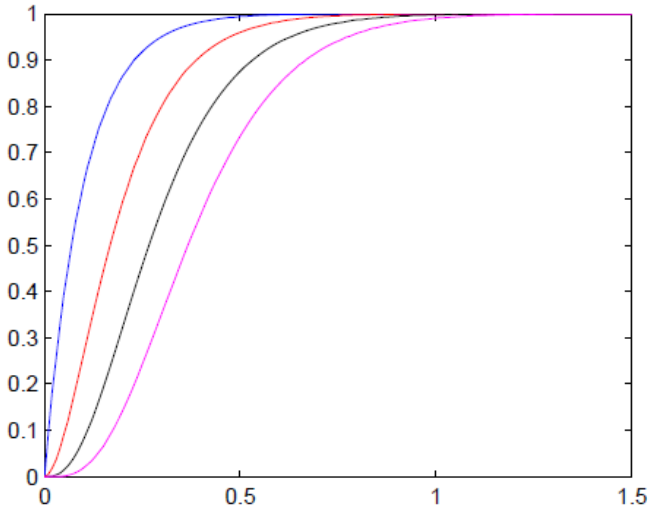
$$\frac{\omega}{s+\omega}$$

$$\frac{\omega^2}{s^2+2\zeta\omega s+\omega^2}$$

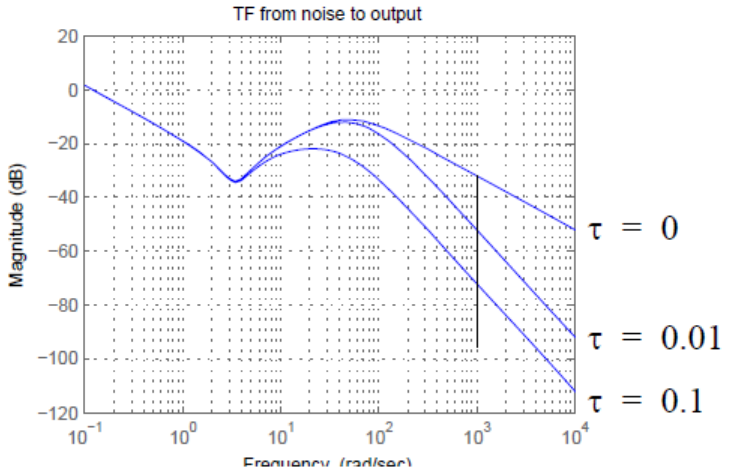
$$\frac{\omega^3}{(s^2+2\zeta\omega s+\omega^2)(s+\omega)}$$

$$\frac{\omega^4}{(s^2+2\zeta\omega s+\omega^2)(s^2+2\zeta\omega s+\omega^2)}$$

$\omega = 10, \zeta = 1$ for all models

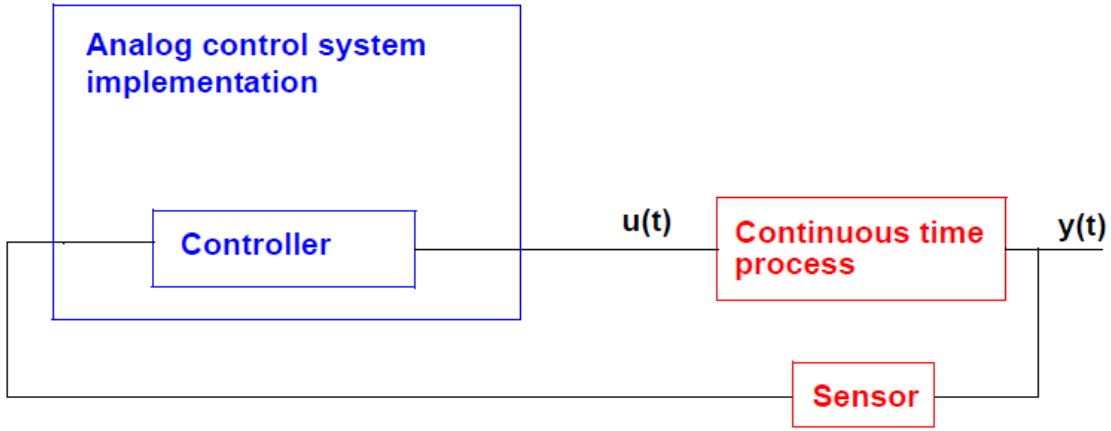


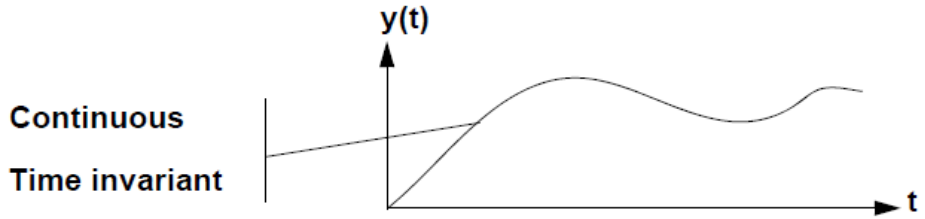
- Example, gain from sensor noise at 50 Hz to output must be less than ()

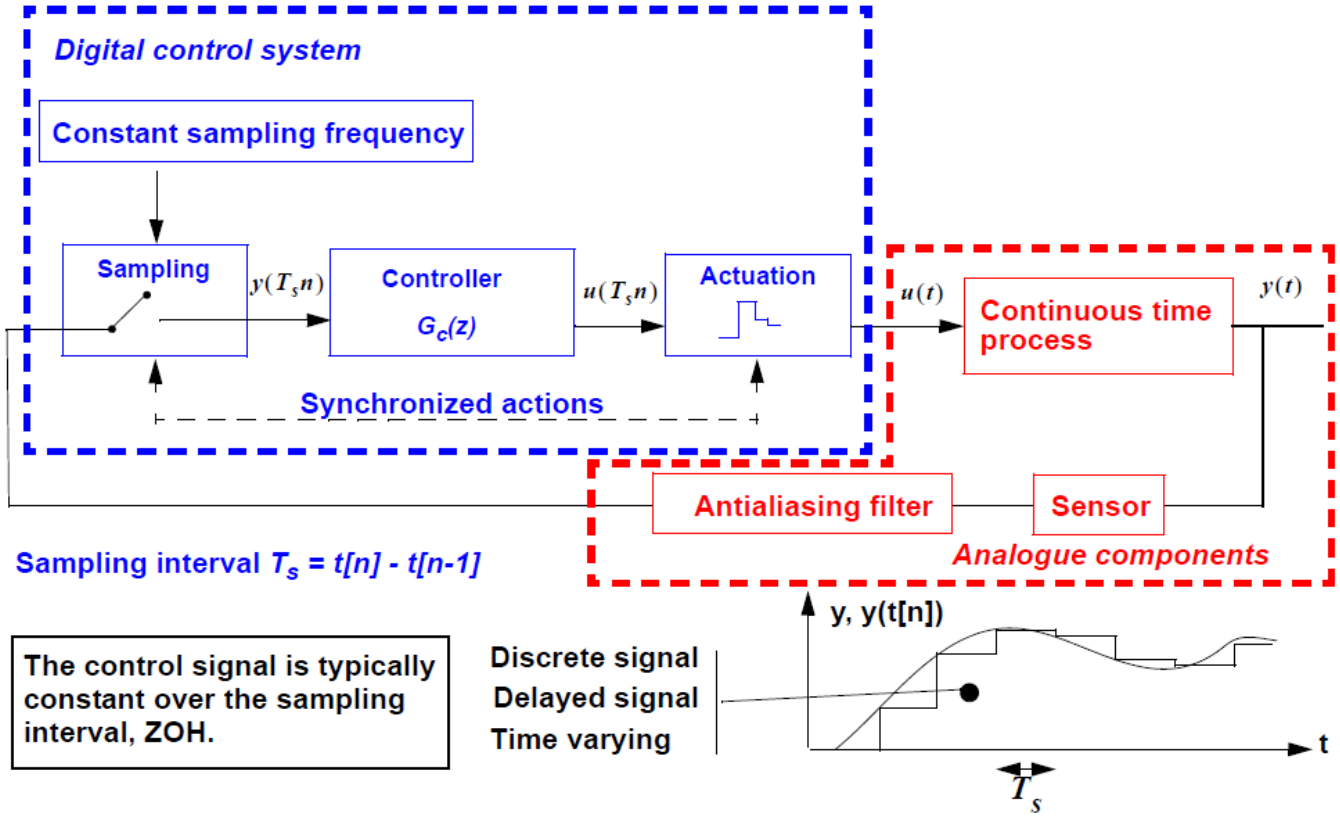


- **Example of a specification on t.f. from noise to output**

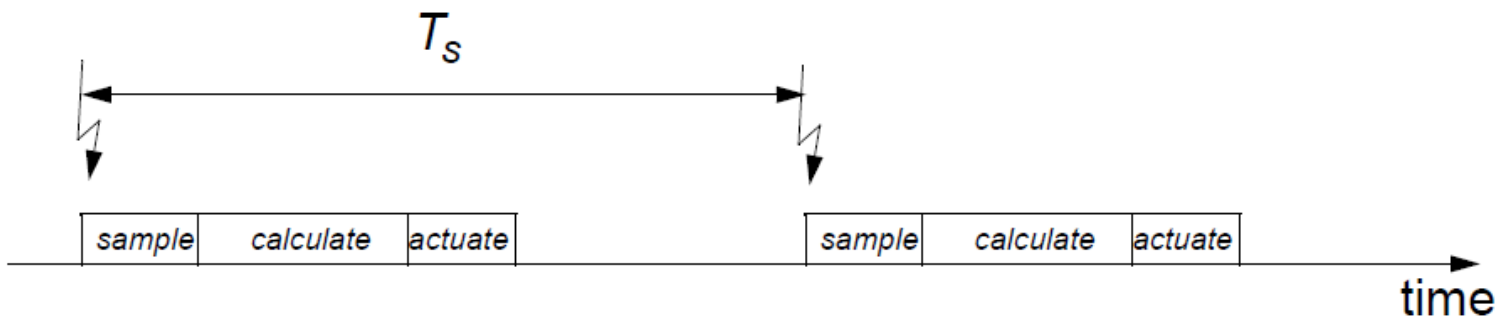
$$G_{yn}(j50) < -40\,\mathrm{db}$$


Analog control system implementation
Controller
u(t)
Continuous time process
y(t)
Sensor

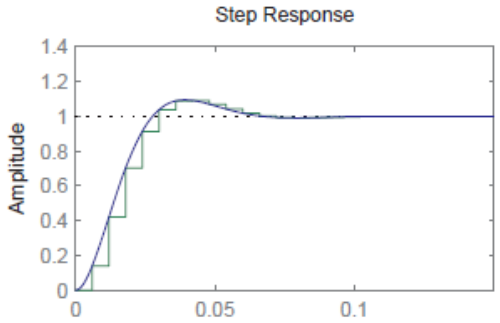

y(t)
Continuous
Time invariant
t

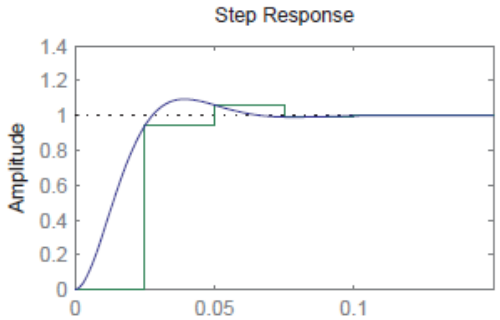

Digital control system
Constant sampling frequency
Sampling
$y(T_s n)$
Controller
$G_c(z)$
$u(T_s n)$
Actuation
Synchronized actions
$u(t)$
Continuous time process
$y(t)$
Antialiasing filter
Sensor
Analogue components
Sampling interval T_s = t[n] - t[n-1]
The control signal is typically constant over the sampling interval, ZOH.
Discrete signal
Delayed signal
Time varying
y, y(t[n])
t
T_s

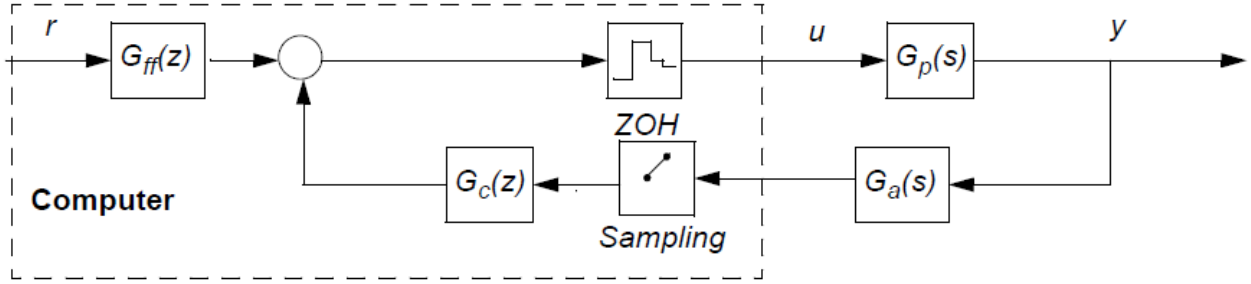
- sampling at constant frequency (constant sampling interval)
- synchronism between sampling and actuation
- zero delay between sampling and actuation (clearly we can not achieve this exactly, execution of the control algorithm takes time)



- single rate systems
  - high sampling rate is costly
  - the frequency should be set in relation to the fastest dynamics in the closed loop characteristics (i.e. bandwidth, rise-time) of the feedback, observer or model following.
  - or 4-10 samples per rise time







- **The sampling frequency must be faster then the fastest dynamic mode in the control system, which could be either:**
  - in the feedback $\frac{S}{R}$, in the feedforward $\frac{T}{R}$ or in the closed loop $AR + BS$. It can also be taken from the bandwidth or crossover frequency of the controller.
  - If the fastest pole is ω_b then the sampling frequency should be $\omega_s = [10 \ldots 30]\omega_b$ and thereby sampling time $T_s = \frac{2\pi}{\omega_s}$

Mapping s-plane to the z-plane

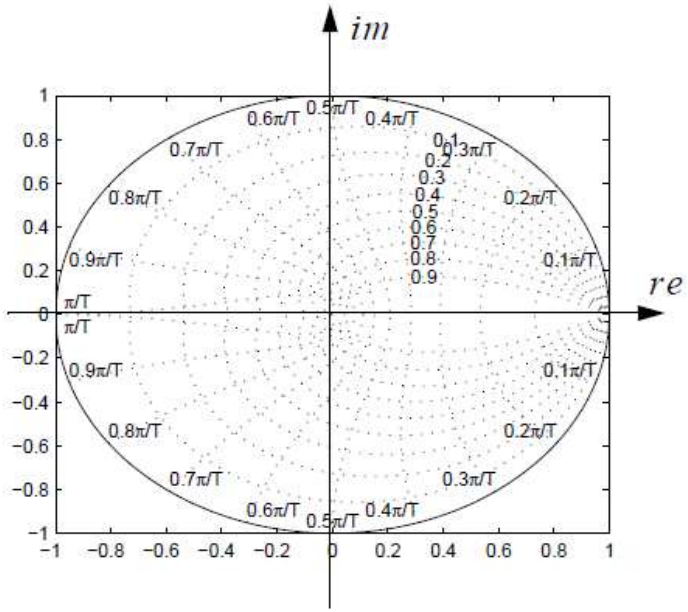
- **Poles**

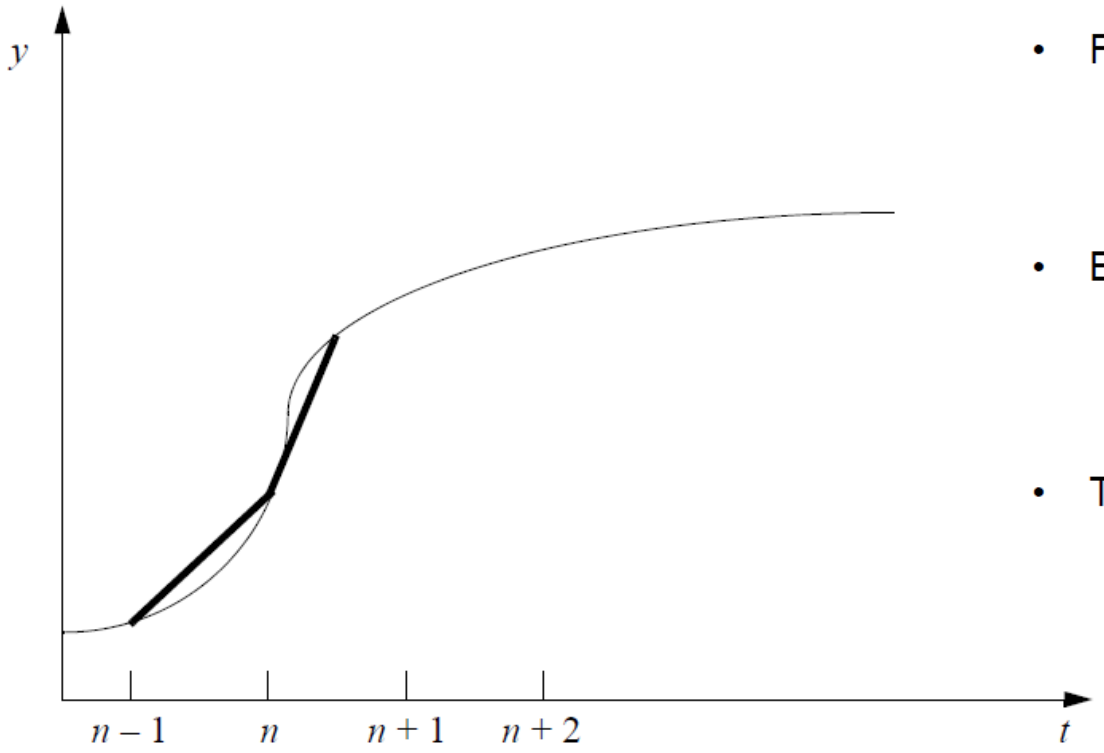
A continuous time pole $s = a + bi$ is mapped to a discrete time pole by $z = e^{sT_s}$ where T_s is the sampling period.
(From the definition of the z-transform)

The continuous time stability border $s = j\omega$, $\omega = [-\infty, \infty]$ is

$$z = e^{j\omega T_s} = \cos(\omega T_s) + i\sin(\omega T_s)$$

which is the unit circle.





- Forward difference approximation (Euler's method)

$$sx = \frac{dx(t)}{dt} \approx \frac{x(t+T_s)-x(t)}{T_s} = \frac{z-1}{T_s}x(t)$$

- Backward difference

$$sx = \frac{dx(t)}{dt} \approx \frac{x(t)-x(t-T_s)}{T_s} = \frac{1-z^{-1}}{T_s}x(z) = \frac{z-1}{zT_s}x(z)$$

- Tustins approximation, (bilinear transformation), (Trapezoidal method)

$$sx = \frac{dx(t)}{dt} \approx \frac{2}{T_s}\frac{x(t+T_s)-1}{x(t+T_s)+1} = \frac{2(z-1)}{T_s(z+1)}x(z)$$

Ex. for a PID controller with Euler forward

$$\frac{S(s)}{R(s)} = \frac{s_2 s^2 + s_1 s + s_0}{s(s + r_0)} \longrightarrow \frac{S(z)}{R(z)} \approx \frac{s_2\left(\frac{z-1}{T_s}\right)^2 + s_1\frac{z-1}{T_s} + s_0}{\frac{z-1}{T_s}\left(\frac{z-1}{T_s} + r_0\right)}$$

Use Maple!

Tustin is availabe for numeric approximation in Matlab, Control Toolbox

Euler forward

$$\dot{x} \approx \frac{x[n+1]-x[n]}{T_s} = Ax[n] + Bu[n]$$

$$y = Cx[n]$$

$$x[n+1] = x[n] + T_s Ax[n] + T_s Bu[n]$$

$$x[n] = (1 + T_s A)x[n-1] + T_s Bu[n-1]$$

$$y[n] = Cx[n]$$

Delay

Euler backward

$$\dot{x} \approx \frac{x[n]-x[n-1]}{T_s} = Ax[n] + Bu[n]$$

$$y = Cx[n]$$

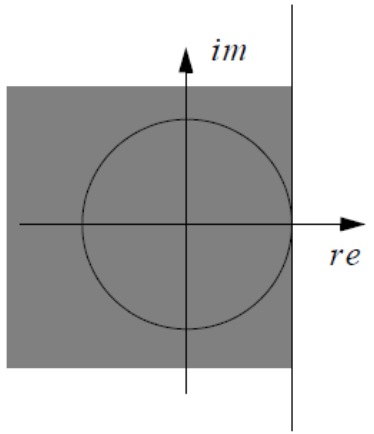
$$x[n] = x[n-1] + T_s Ax[n] + T_s Bu[n]$$

$$x[n] = (1 + T_s A)^{-1} x[n-1] + (1 + T_s A)^{-1} Bu[n]$$

No delay

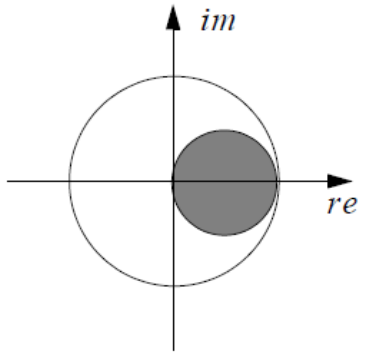
Mapping of poles

- The stability region in the continuous time case (left half plane) corresponds to the unit circle in the discrete time case.



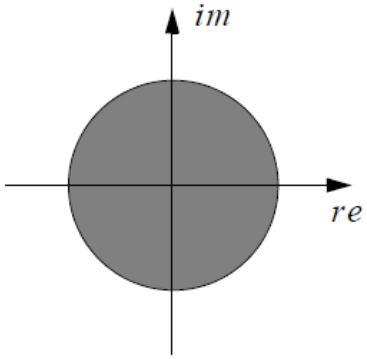
Forward difference

$$z = e^{sT_s} \approx 1 + sT_s$$



Backward difference

$$z = e^{sT_s} \approx \frac{1}{1 - sT_s}$$



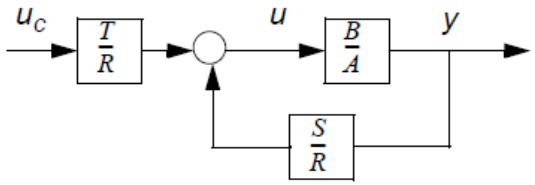
Tustin

$$z = e^{sT_s} \approx \frac{1 + (sT_s)/2}{1 - (sT_s)/2}$$

- **Compare simulated step response (in Simulink)**
  - 1.) With continuos prosess model and continuous controller
  - 2.) With continuous process model discrete time controller.
- **Compare the phase and amplitude margins**
  - 1.) With continuos prosess model and continuous controller
  - 2.) With a zoh model of the process and the discrete time controller
  - It is not possible to make a bode or nyquist plot in matlab for a combined continuous and discrete time model.

- The continuos time controller is

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



- After a discrete time approximation we have

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

- Select the sampling period $[10, 30]$ times faster than the c.l. poles. (Observe that the poles are in *rad/s*)

- Use Tustins approximation in Matlab, *i)* for the feedback part $G_c(s) = \frac{S(s)}{R(s)}$

  and, *ii)* for the feedforward part $G_{ff}(s) = \frac{T(s)}{R(s)}$ separatly.

Position control with PD controller

Process: $G_p(s) = \frac{B(s)}{A(s)} = \frac{K_t/J}{s(s+d/J)}$, with current as input!

PD-controller with L.P. filter $G_c(s) = \frac{S(s)}{R(s)} = \frac{s_1 s + s_0}{s + r0}$ -> $AR + BS$, third order

c.l. poles, specification $A_m(s)A_0(s) = (s^2 + 2\zeta\omega s + \omega^2)(s+\omega)$

Calculate $\{s_1, s_0, r_0\}$ by solving

$$s^3 + \left(\frac{d}{J} + r_0\right)s^2 + \left(\frac{dr_0}{J} + \frac{K_t s_1}{J}\right)s + \frac{K_t s_0}{J} = (s^2 + 2\zeta\omega_1 s + \omega^2)(s+\omega_2)$$, with $\omega_1 = \omega_2 = \omega$

$$\{s1 = \frac{\omega^2 J^2 + 2\zeta\omega^2 J^2 - 2d\zeta\omega J - d\omega J + d^2}{JKt}, r0 = \frac{2\zeta\omega J + \omega J - d}{J}, s0 = \frac{\omega^3 J}{Kt}\}$$

Based on specifications, choose $\omega = 50$ and $\zeta = 0.8$, which gives

$$\frac{S(s)}{R(s)} = \frac{138.8s + 2778}{s + 134.4} \text{ and } \frac{T(s)}{R(s)} = \frac{t_0 A_o}{R} = \frac{55.56s + 2778}{s + 134.4}$$

Approximate a discrete time implementation with e.g. Tustin, select the sampling period from rule of thumb $T_s = \frac{2\pi}{20\omega} \approx 0.006$.

$$\frac{S(z)}{R(z)} = \frac{104.8z - 92.9}{z - 0.43}, \quad \frac{T(z)}{R(z)} = \frac{45.5z - 33.7}{z - 0.43}$$

Control law: $R(z)u(z) = T(z)u_c(z) - S(z)y(z)$

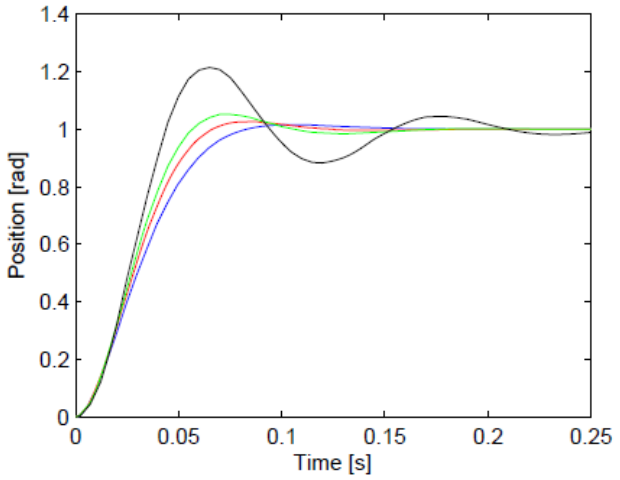
$(z - 0.43)u = (45.5z - 33.7)u_c - (104.8z - 92.9)y$ shift with z^{-1} gives the control,

$$u[n] = 0.43u[n-1] + 45.5u_c[n] - 33.7u_c[n-1] - 104.8y[n] + 92.9y[n-1]$$

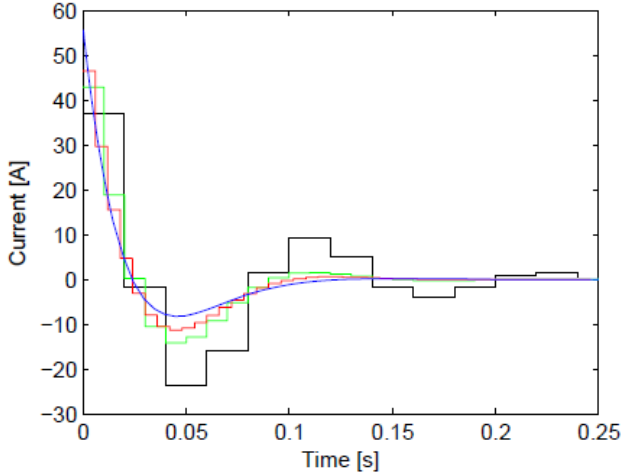
Simulated step response:

blue line, continuos time controller
red line, discrete time controller with Ts = 6 ms
green line, discrete time controller with Ts = 10 ms
red line, discrete time controller with Ts = 20 ms

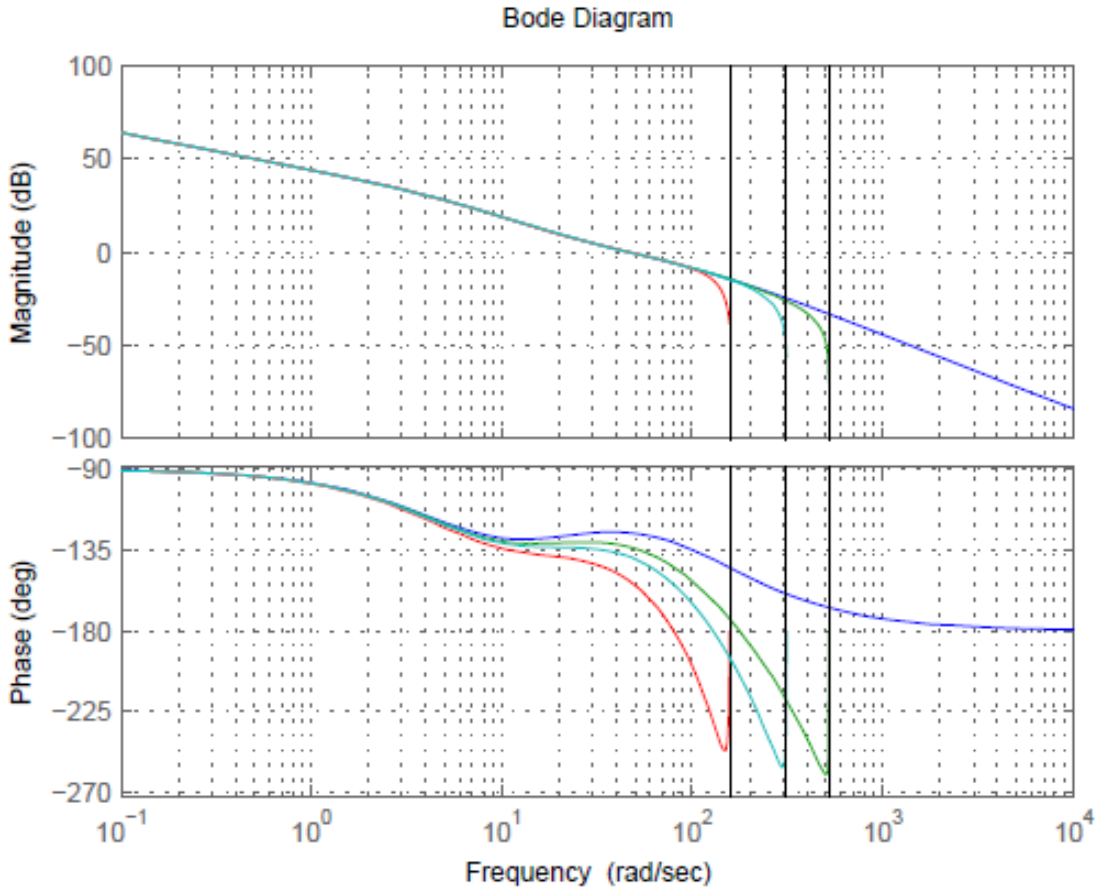
position signal



control signal



Model/Margin	Phase	Amplitude
Continuous	inf	54
Disc. 6 ms	46	16
Disc. 10 ms	41	12
Disc. 20 ms	27	5.6



- Polynomial approach to poleplacement feedback design
- Mapping of time domain specifications to pole location specifications.
- Selection of sample period for a discrete time implementation based on specifications.
- Approximation of the continuos time controller to a discrete time controller with e.g., Euler or Tustin.
- Evaluate the approximation in time and frequency planes.

- Modelling and design completely in discrete time from the start.

  "How does the computer see the process"

  A good reference is:
  Computer Controlled Systems - Theory and design, Third edition
  Åström and Wittenmark
  Prentice Hall, ISBN 0-13-314899-8

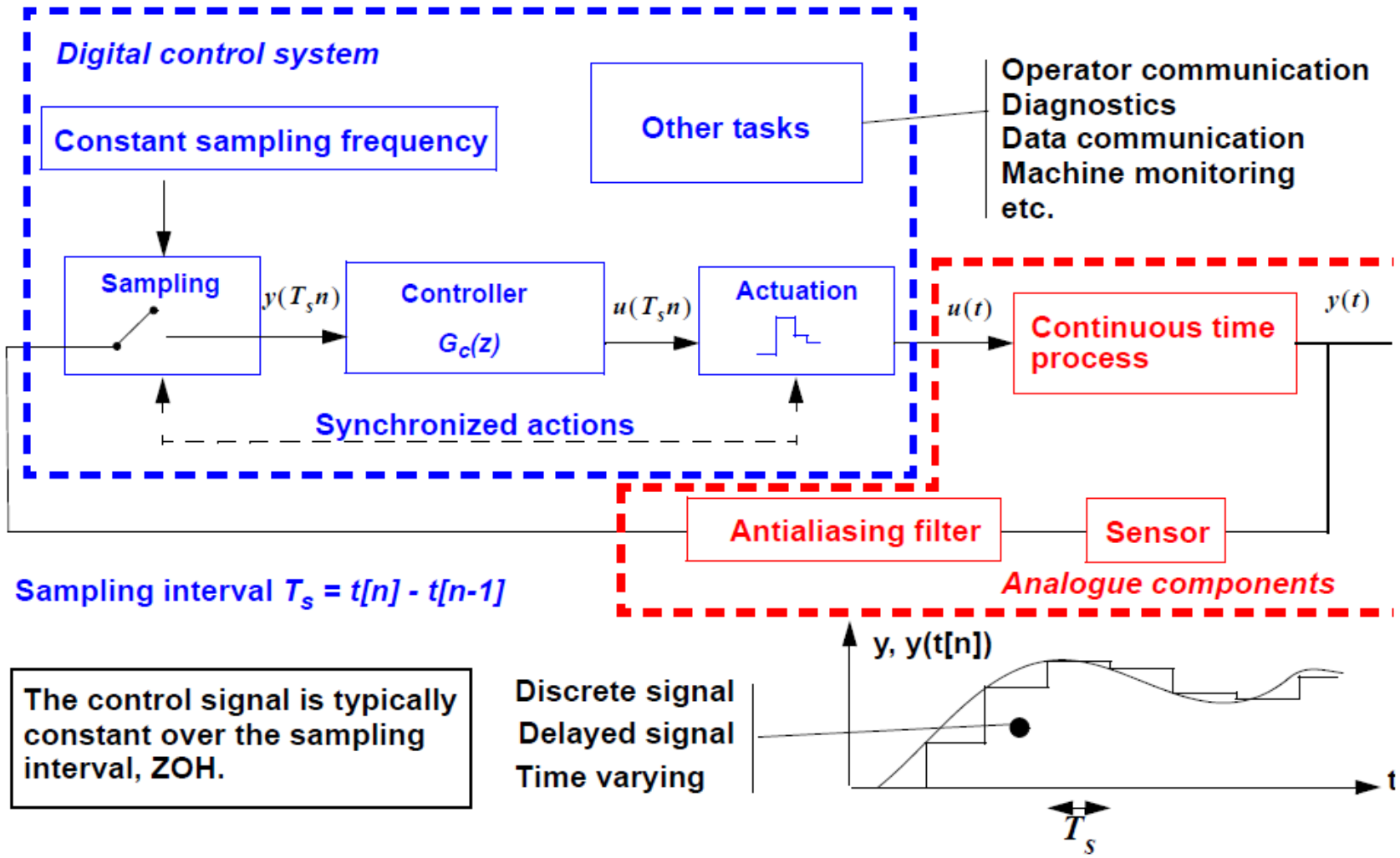
- Transformation of an existing continuous time design to discrete time.

  "How do we approximate a continuous time controllers diff. eq. as good as possible to a digital computer"

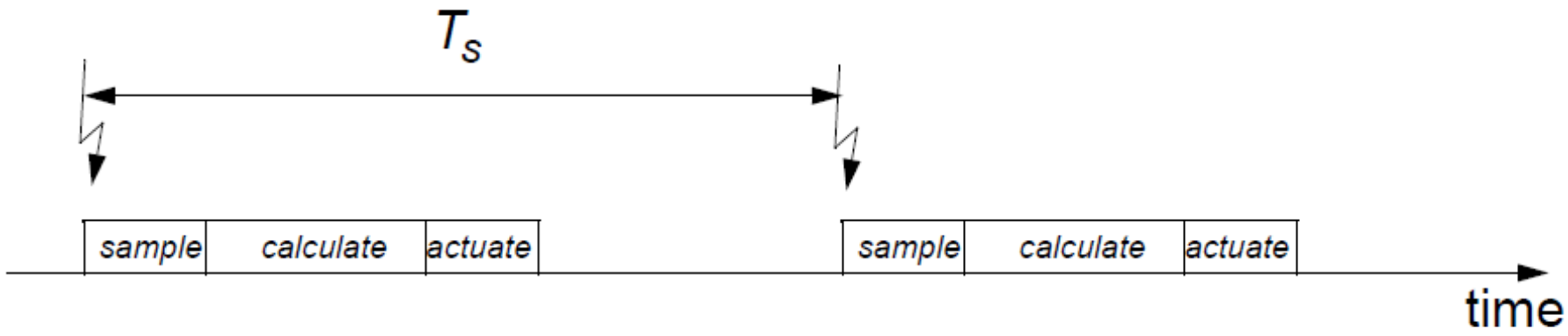
  - Continuous time design taking computer characteristics into account.

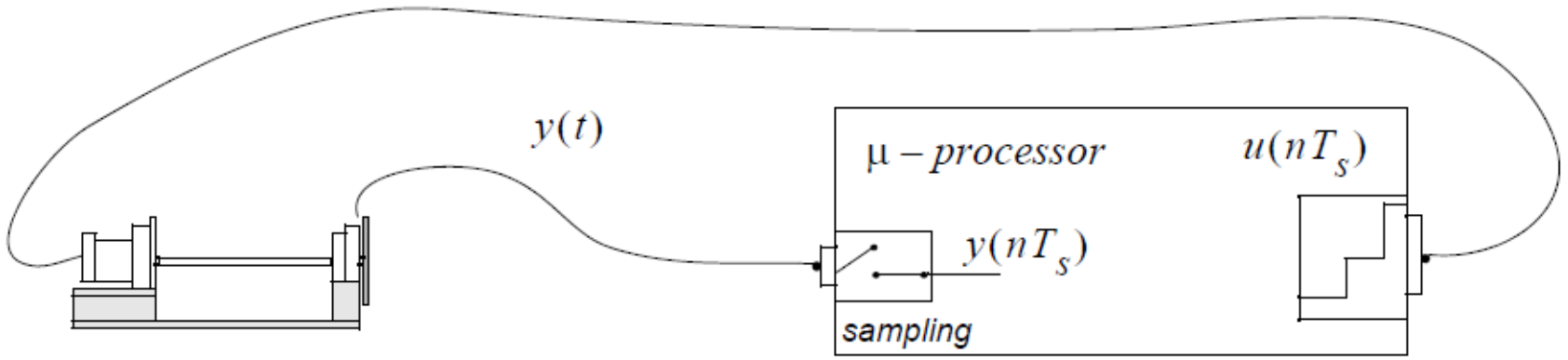
- The process to be controlled is first modelled (continuous time).
- The process model is then transformed into a discrete time model using zero order hold sampling (zoh).
- The control design (e.g. pole placement design) is performed completely in the discrete domain. (With new rules for pole placement).

- Advantage:
  Better performance when the sampling period is "to slow"
- Disadvantage:
  Some of the physical insight is lost when leaving the differential equations in favour of the difference equations.


Digital control system
Constant sampling frequency
Other tasks
Operator communication
Diagnostics
Data communication
Machine monitoring
etc.
Sampling
$y(T_s n)$
Controller
$G_c(z)$
$u(T_s n)$
Actuation
$u(t)$
Continuous time process
$y(t)$
Synchronized actions
Antialiasing filter
Sensor
Analogue components
Sampling interval T_s = t[n] - t[n-1]
The control signal is typically constant over the sampling interval, ZOH.
Discrete signal
Delayed signal
Time varying
y, y(t[n])
t
T_s

- sampling at constant frequency (constant sampling interval)
- synchronism between sampling and actuation
- zero delay between sampling and actuation (clearly we can not achieve this exactly, execution of the control algorithm takes time)





The signal $y(t)$ which is generated by the process with the model

$$y(s) = G_1(s)u(s):$$

is EXACTLY described at each sampling instance,

nT_s with $n = 1,2,3....,$ by the model

$$y(z) = G_2(z)u(z):$$

If $u(nT_s)$ is constant during the sampling period and

$G_2(z)$ is the zero order hold model of $G_1(s)$.

Consider the state space model

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t)$$

$$y(t) = C \cdot x(t)$$

The corresponding discrete time model describes the system only at the sampling instants $t[n]$. Given that we know the state vector at sample $t[n]$ we may integrate the continuous time system from $t[n]$ to $t[n+1]$. This gives the discrete time state space system

$$x[n+1] = \Phi \cdot x[n] + \Gamma \cdot u[n]$$

$$y[n] = C \cdot x[n]$$

The new system matrices Φ and Γ are given on the next slide.

Assuming that the system is sampled with sampling interval T_s, i.e. $t[n+1] - t[n] = T_s$ and that the input is constant during the sampling interval (referred to as zero order hold sampling), the system matrices A and B are replaced by Φ and Γ respectively

$$\Phi = e^{AT_s}$$

$$\Gamma = \int_0^{T_s} e^{As}\, ds \cdot B$$

- The Φ matrix defies the pole locations of the discrete time system. Note that the discrete time poles are different from the continuous time poles.
- The discrete model reflects the system behaviour at sampling instants only, i.e. system dynamics may be hidden by the sampling procedure.
- Φ and Γ are derived in *Computer control ch. 3*

2nd order continuous time system

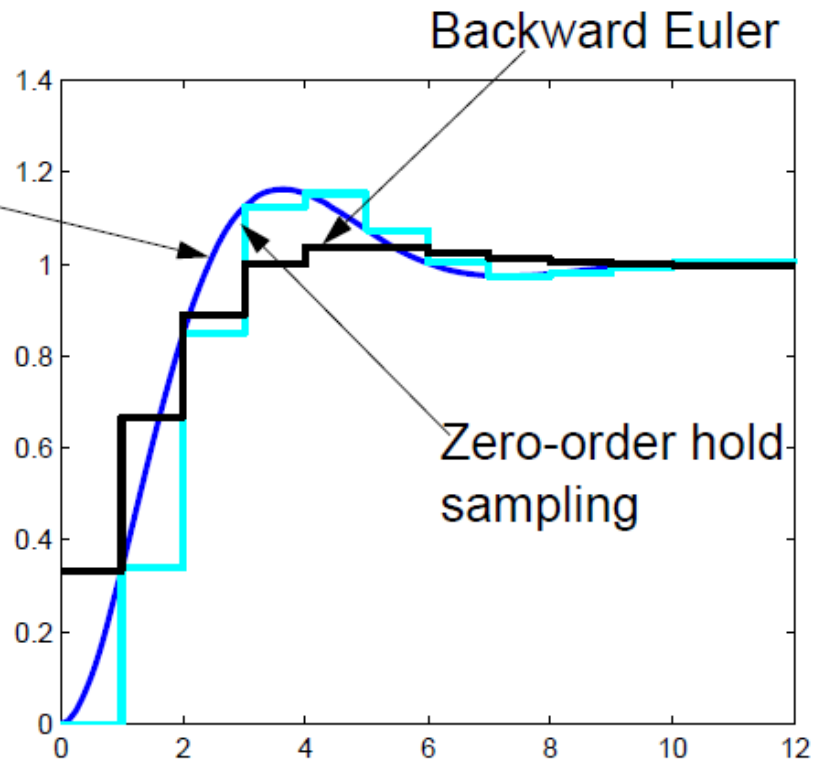
$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

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

$$\Phi = e^{AT_s} \qquad \Gamma = \int_0^{T_s} e^{As}\, ds \cdot B$$

$$x[n+1] = \begin{bmatrix} 0.66 & 0.53 \\ -0.53 & 0.13 \end{bmatrix} x[n] + \begin{bmatrix} 0.34 \\ 0.53 \end{bmatrix} u[n]$$

$$y[n] = \begin{bmatrix} 1 & 0 \end{bmatrix} x[n]$$



Ts = 1s

- **The discrete time transfer function is calculated from the discrete time state space model.**

s.s. model, $x[n+1] = \Phi x[n] + \Gamma u[n]$
$y[n] = Cx[n]$

the shift operator z, $x[n+1] = zx[n]$

hence, $(z-\Phi)x[n] = \Gamma u[n]$
$y[n] = Cx[n]$

$x[n] = (z-\Phi)^{-1}\Gamma u[n]$
$y[n] = Cx[n] = C(z-\Phi)^{-1}\Gamma u[n]$

The discrete time transfer function $G(z) = \frac{y(z)}{u(z)} = C(z-\Phi)^{-1}\Gamma$

- **ZOH is NOT an approximation as Euler or Tustin.**
- **ZOH should NOT be used to approximate continuous time controllers since the average delay will be a half sample period. (see slide 4.2.6.)**
- **The zoh model is calculated from the continuous time state space model and a sampling period.**
- **The zoh model is a discrete time state space model, but can be converted to a discrete time transfer function.**

- **Specifications are often given in the time domain, e.g., step response and response to external disturbances.**
  - The time domain specifications are converted to continuous poles.
  - The continuous poles are converted to discrete poles.
- **The control structure, i.e., *S(s)* and *R(s)* are selected.**
  - The control structure is converted to a discrete version, *S(z)* and *R(z)*.

- A continuous time pole $s = a + bi$ is mapped to a discrete time pole by $z = e^{sT_s}$ where T_s is the sampling period. (See lecture 2.1)

- A first order polynomial $P_c(s) = s + a$ has pole at $s = -a$

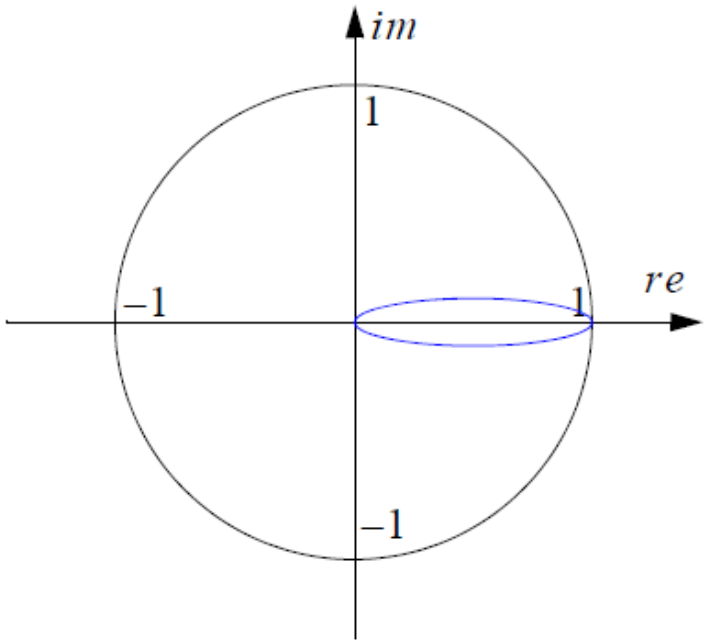
  which gives a discrete time pole at

  $z = e^{sT_s} = e^{-aT_s}$ and a discrete time polynomial $P_d(z) = z - e^{-aT_s}$

- $s = 0$, $P_d(z) = z - 1$ (integrator) -> $z = 1$

  $s = -\infty$, $P_d(z) = z$ (very fast!) -> $z = 0$

  $0 < s < -\infty$, $P_d(z) = z - e^{-aT_s}$ -> $1 > z > 0$



A second order polynomial

$P_{c_2}(s) = s^2 + 2\zeta\omega_0 s + \omega_0^2$ has poles at

$s_{1,2} = -\zeta\omega_0 \pm i\omega_0\sqrt{1-\zeta^2}$, calculating

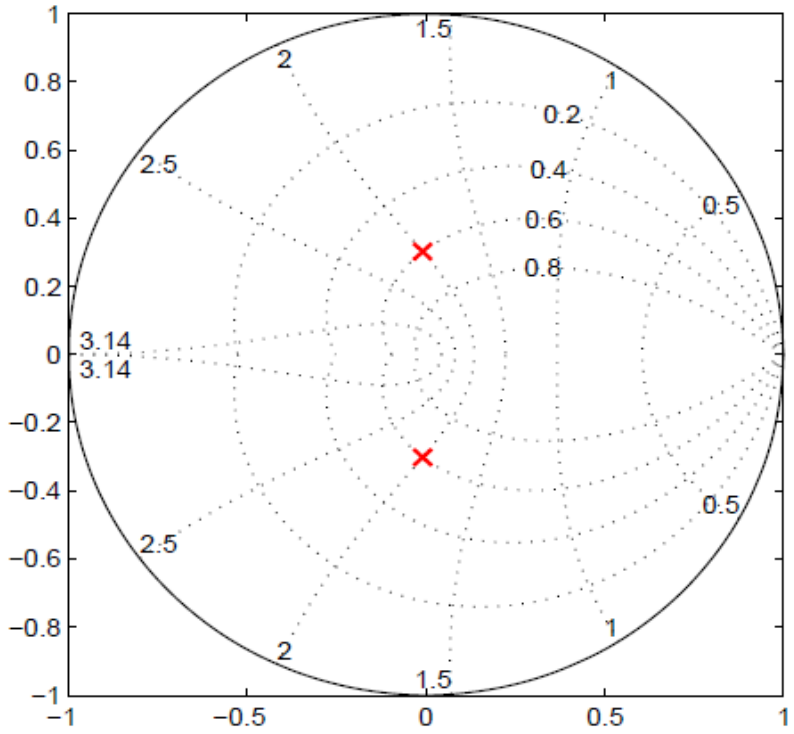
$z_{1,2} = e^{s_{1,2}T_s}$ gives the second order discrete time polynomial,

$P_{d_2} = z^2 + p_1 z + p_0$, with

$$p_1 = -2e^{-\zeta\omega_0 T_s}\cos\left(\omega_0 T_s\sqrt{1-\zeta^2}\right)$$

$$p_0 = e^{-2\zeta\omega_0 T_s}$$

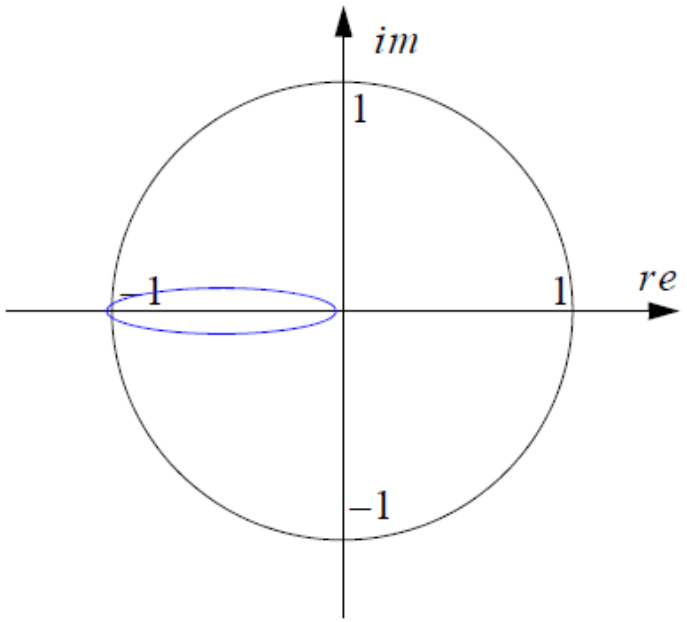
Matlab: >> poles_disc=exp(poles_cont*Ts)

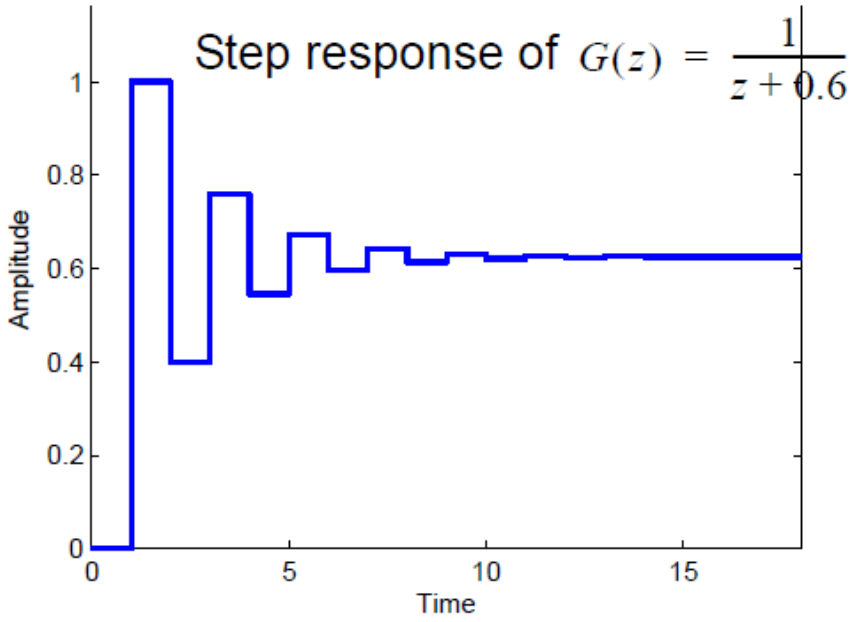


The frequencies are scaled with Ts.
And that π is the highest freq. (Nyquist freq.)

- The single discrete time pole $z = a$ with $-1 < a < 0$ can not be mapped to a single continuous time pole since $s = \frac{\ln(z)}{T_s}$ is a complex number.

- Example: $G(z) = \frac{1}{z + 0.6}$ with $T_s = 1$





- Calculate the process model, $G_p(z) = \frac{B(z)}{A(z)} = \frac{b_n z^n + \ldots + b_0}{z^m + a_{m-1} z^{m-1} + \ldots + a_0}$

- Chose control structure, P, PD, PI etc. which gives the structure of $S(z)$ and $R(z)$
  -more on next slide on the discrete time versions

- Determine the order of the closed loop, $AR + BS$

- Given the closed loop specifications define the desired closed loop polynomial $A_{cl}(z) = A_m(z)A_o(z)$, where A_m has the same order as A.

- Calculate the coefficients in S and R by solving the Diophantine equation $AR + BS = A_m A_o$.

- **To get integral action $\frac{1}{s}$ in a discrete time controller the factor $(z-1)$ should be included in R. ($e^{T_s a} = 1$ with $a = 0$)**

  ex. PI controller, $G_c(s) = \frac{Ps + I}{s}$,

  hence, $\frac{S(s)}{R(s)} = \frac{s_1 s + s_0}{s}$ gives $\frac{S(z)}{R(z)} = \frac{s_1 z + s_0}{z - 1}$

- **To get derivative action in the controller.**

  $G_c(s) = Ds + P$

  hence, $\frac{S(s)}{R(s)} = \frac{s_1 s + s_0}{1}$, gives $\frac{S(z)}{R(z)} = s_1 z + s_0$, however this is not proper since,

  $u[n] = -s_1 y[n+1] - s_0 y[n]$. (The notion of velocity in the control is lost!)

To get it proper include a time constant in R.

$$G_c(s) = \frac{Ds + P}{(\tau s + 1)},$$

hence, $\frac{S(s)}{R(s)} = \frac{s_1 s + s_0}{s + r_0}$, which gives $\frac{S(z)}{R(z)} = \frac{s_1 z + s_0}{z + r_0}$ with the implementation.

$(z + r_0)u = (s_1 z + s_0)y$ and $u[n] = r_0 u[n] - s_1 y[n] - s_0 y[n-1]$

- **PID control**

$G_c(s) = \frac{Ds^2 + Ps + I}{s}$ is not proper, include a time constant in R

$G_c(s) = \frac{Ds^2 + Ps + I}{s(\tau s + 1)}$ hence, $\frac{S(s)}{R(s)} = \frac{s_2 s^2 + s_1 s + s_0}{s(s + r_0)}$ which gives $\frac{S(z)}{R(z)} = \frac{s_2 z^2 + s_1 z + s_0}{(z-1)(z + r_0)}$

- **Time response. Simulation in Simulink with discrete time controller and continuous time process model. Not possible in Matlab with e.g. 'step'.**

  - Sensor simulation, sampling and noise.
  - Disturbance, external load...
  - Antialiasing filter

- **Frequency response. Must be done with discrete time controller and discrete time process model. Include a discrete time version of the antialiasing filter!**

  - phase and amplitude margin
  - load and sensor noise
  - robustness from Sensitivity function, (more in lecture 7).

- **The digital implementation of a controller on a microprocessor 'sees' the process model as the zoh model.**
- **Select a sample time and calculate the zoh model**
- **Design the controller based on the zoh model**
  - Convert the closed loop continuous poles from the specifications to discrete poles
  - Convert the control structure from continuous time to discrete time.
  - Calculate the control parameters exactly in the same way as in the continuous time case.

Advantage, very long sample periods are possible

Disadvantage, some of the physical insight is lost

- Motor from voltage to position $G_p(s) = \frac{b}{s(s+a)}$.
- From specifications we want c.l. poles at $\omega_0 = -50$ *rad/s* with minimum damping $\zeta = 0.9$.
- Sampling period from rule of thumb in the interval $T_s = \left[\frac{2\pi}{10\omega_0}, \frac{2\pi}{30\omega_0}\right]$.
- ZOH sampling of $G_p(s)$ gives $G_p(z) = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0}$ (second order)
- Design a PD controller of first order, -> $\frac{S(z)}{R(z)} = \frac{s_1 z + s_0}{z + r_0}$

- Compute c.l. polynomial, $A_{cl}(z) = A(z)R(z) + B(z)S(z)$

$$Acl := z^3 + (a1 + r0 + b1\ s1)\ z^2 + (a0 + a1\ r0 + b0\ s1 + b1\ s0)\ z + a0\ r0 + b0\ s0$$

- Select the desired c.l. polynomial as the third order polynomial $A_{cl}(z) = A_m(z)A_o(z)$ where A_m is second order (same as $A(z)$).

$$A_{cl}(z) = (z^2 + p_1 z + p_0)(z + p_0)$$

- Solve for the three unknown control parameters $\{s_1, s_0, r_0\}$.

$$\left\{ s1 = -\frac{b0\ a0 - b0\ a1^2 + b1\ a0\ a1 + b0\ a1\ p2 - b1\ a0\ p2 - b0\ p1 + b1\ p0}{-b0\ a1\ b1 + b0^2 + a0\ b1^2},\right.$$

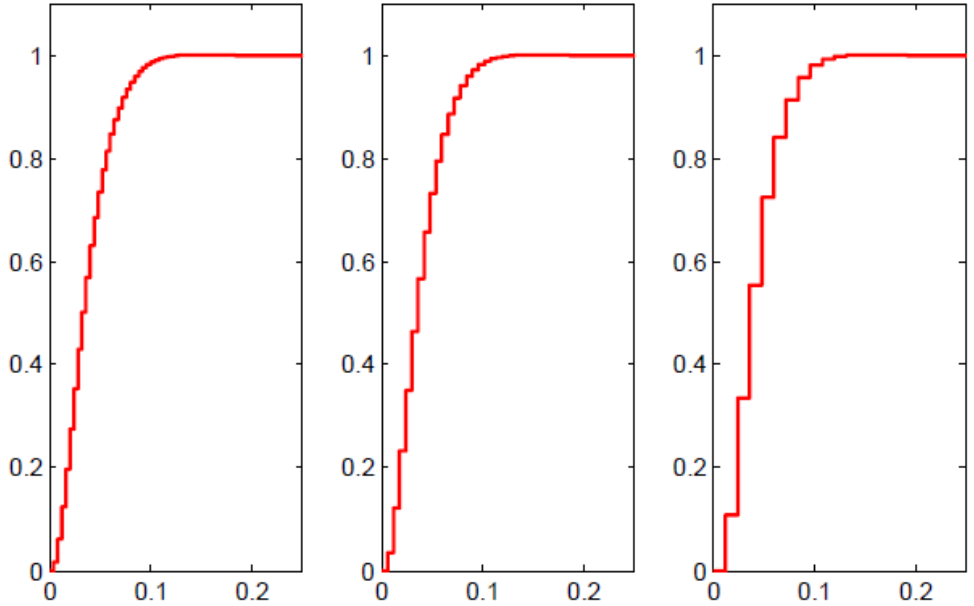
$$s0 = -\frac{-a0\ a1\ b0 + a0^2\ b1 - a0\ b1\ p1 + a0\ p2\ b0 + p0\ a1\ b1 - p0\ b0}{-b0\ a1\ b1 + b0^2 + a0\ b1^2},$$

$$\left. r0 = \frac{-a1\ b0^2 + b1\ b0\ a0 - b1\ b0\ p1 + b1^2\ p0 + p2\ b0^2}{-b0\ a1\ b1 + b0^2 + a0\ b1^2} \right\}$$

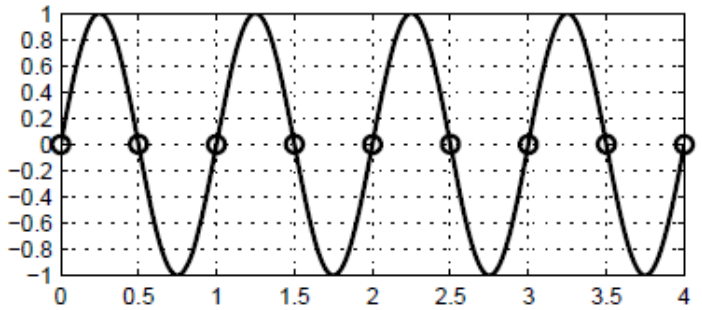
- Select the feed forward part, $T = t_0 A_o$, giving the c.l. $\frac{BT}{AR + BS} = \frac{BT_0 A_o}{A_m A_o} = \frac{BT_0}{A_m}$

- Try different sampling periods, $T_s = \frac{2\pi}{n\omega_0}$, $n = [30, 20, 10] = [4, 6, 12] ms$

Same behaviour for all versions!



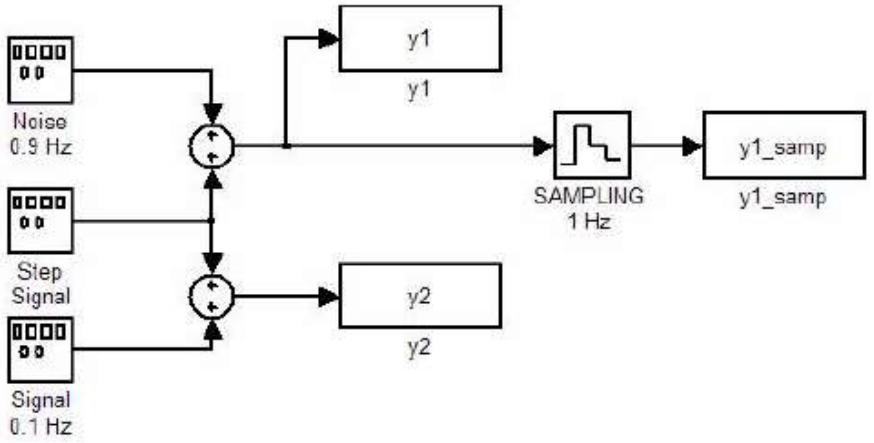
A sine wave signal with frequency ω sampled with freq. $\omega_s = 2\omega$ (circles)



It clearly appears that sampling at this frequency gives nothing!

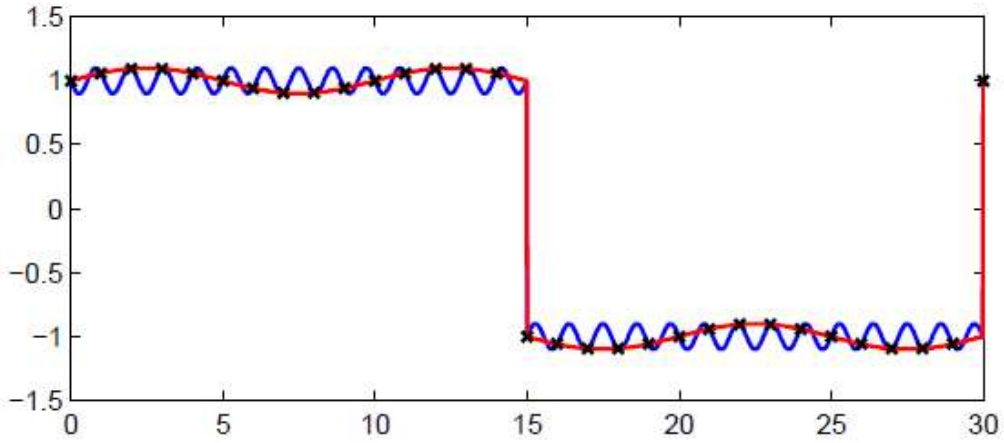
- **Shannons sampling theorem**: A continuous time signal with a fourier transform equal to zero outside the interval $(-\omega_0, \omega_0)$ is given uniquely by equidistant sampling with a frequency ω_s higher that $2\omega_0$.

- In a sampled system the important frequency $\omega_s/2$ is also referred to as the Nyquist frequency. Signals with frequency lower than the Nyquist frequency can be reconstructed after sampling.



A square signal with an overlaid noise signal, $\sin(0.9t)$. (Blue line y_1).

Sampling y_1 with 1 Hz, (black x) gives a new frequency with 0.1 Hz. (same as y_2 red line).



The signal $y_1(t) = \cos(2\pi f_1 t)$ with the frequency $f_1 = 0.9$ *Hz*, is sampled with the frequency $F_s = 1$ *Hz*.

The signal is sampled at the time instants $t = \frac{n}{F_s}$ where $n = 1 \ldots \infty$

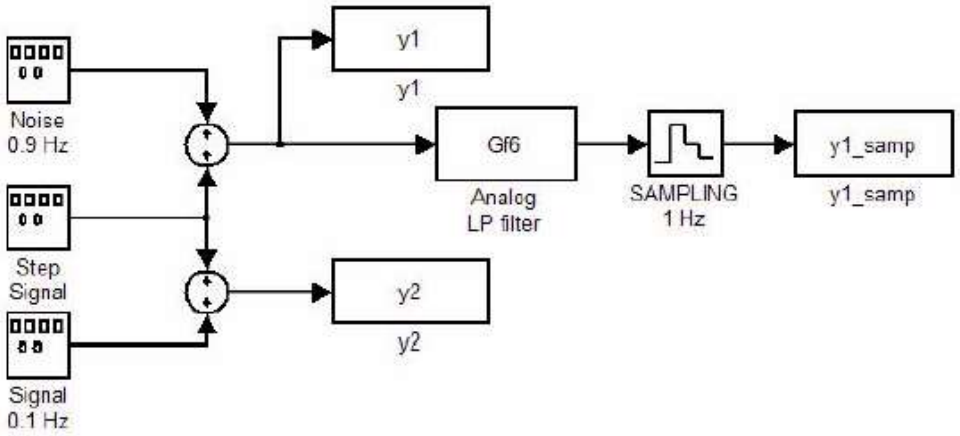
$$y_1(n) = \cos\left(2\pi f_1 \frac{n}{1}\right) = \cos(1.8\pi n)$$

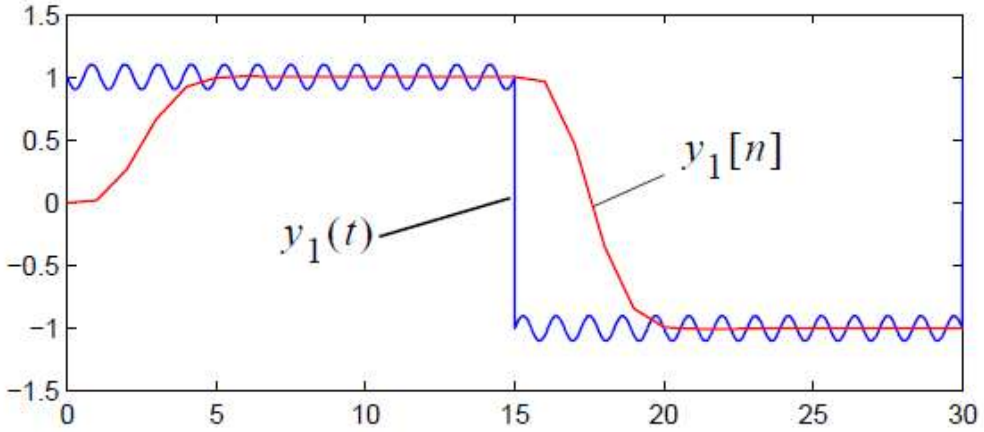
$$= \cos(2\pi n - 0.2\pi n)$$

$$= \cos(2\pi n - 2\pi 0.1 n)$$

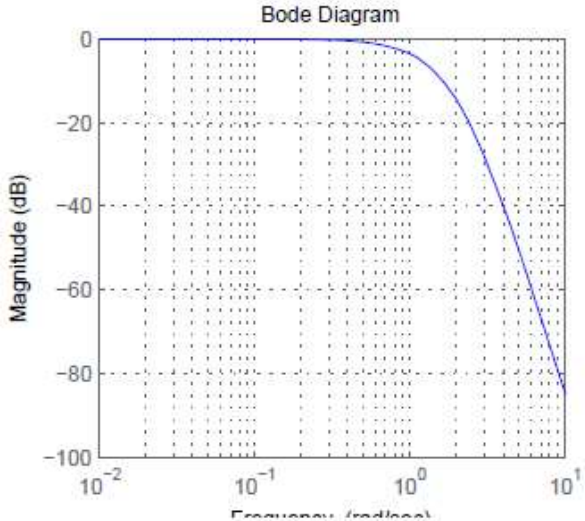
$$= \cos\left(2\pi 0.1 \frac{n}{1}\right) = \cos(2\pi f_2 t)$$

The frequency $f_1 = 0.9$ *Hz* is said to be an *alias* of the frequency $f_2 = 0.1$ *(Hz)* when it is sampled at 1 Hz.





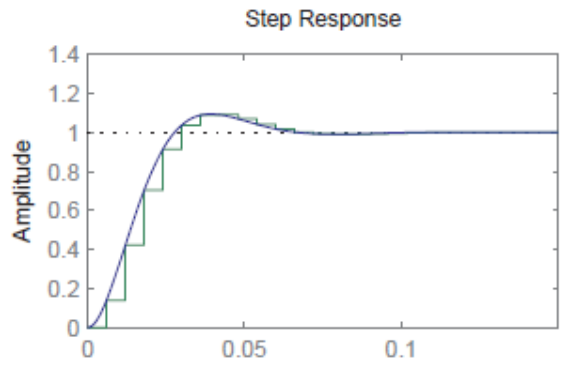
Analog pre filter

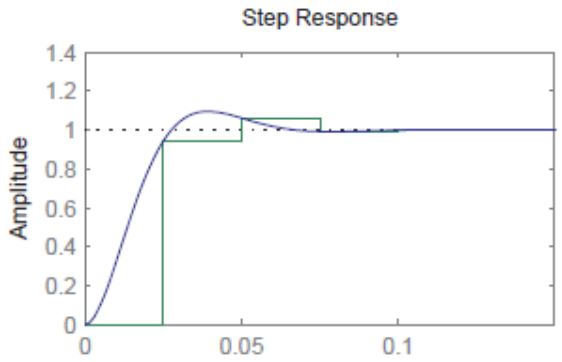


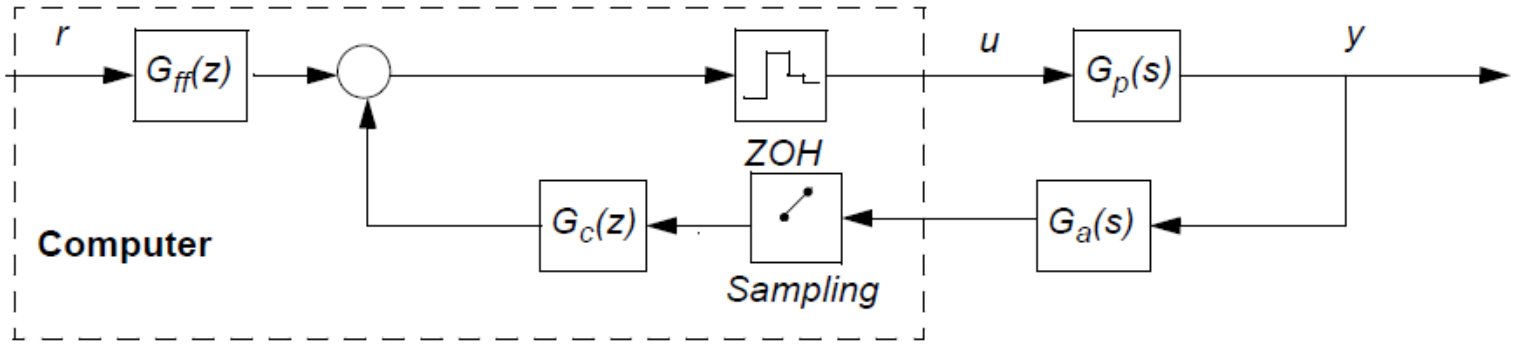
- Real signals are not band limited. High frequency components must be filtered away to avoid aliasing.
- For digital/discrete sensors the aliasing problem is less aggravating.
- The prefilter is typically implemented as an analog filter with resistors, capacitors and an operational amplifier.
- The filter should be taken into account in control design.

NEVER IMPLEMENT A FEEDBACK LOOP WITHOUT ANTIALIASING FILTER ON ALL AD-CONVERTERS

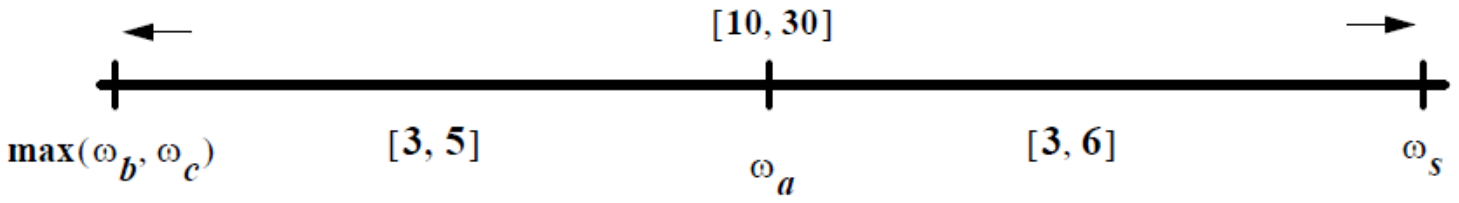
- single rate systems
  - high sampling rate is costly
  - the frequency should be set in relation to the fastest dynamics in the closed loop characteristics (i.e. bandwidth, rise-time) of the feedback, observer or model following.
  - or 4-10 samples per rise time







- Make a preliminary continuous time design of the controller $G_c(s)$ and $G_{ff}(s)$
- It gives the fastest closed loop bandwidths, ω_b and cross-over frequencies, ω_c for the various parts of the control system, normally the c.l. poles.
- Select Sampling frequency $\omega_s \in [10, 30]\max(\omega_b, \omega_c)$ and calculate $G_c(z)$ and $G_{ff}(z)$
- Design an anti-aliasing filter $G_a(s)$ with a bandwidth of $\omega_a \in [0.17, 0.33]\omega_s$



- **Often you will not have a free choice of selecting the sampling period. Based on implementation HW you will "get" a T_s.**

- **Choice of microprocessor for the implementation.**

  high cost processor:

  floating point arithmetic, 32/64 bit , 20-200 Mhz CPU clock
  -> high sampling frequency

  low cost processor:

  fixed point arithmetic, 8/16 bit, 4-30 Mhz CPU clock
  -> low sampling frequency

- **A lot of more functions have to run on the same processor.**

  less time for the controller code to execute.

- **The frequency response of the filter is known and should be realised as a electrical circuit.**
- **For example:**

  Lowpass filter for antialiasing

  Highpass filter for rejection of dc-level signals

  Bandpass filter for passing of specific frequencies

  Bandstop filter for blocking of specific frequencies

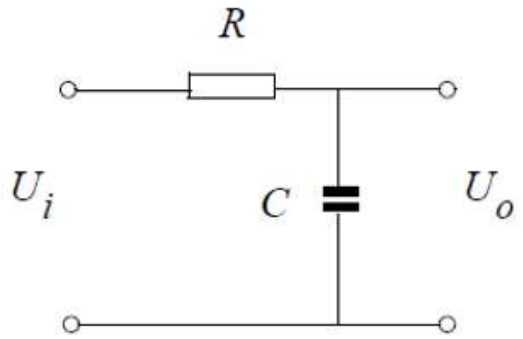
- **Passive circuits**

  Based only on resistors, capacitors and inductors

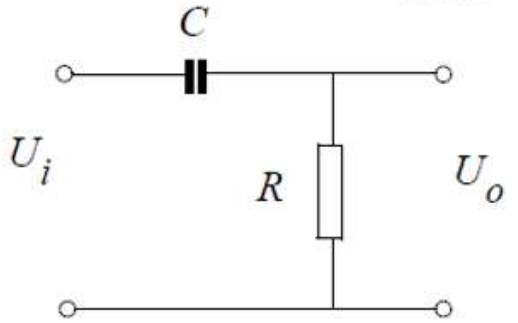
- **Active circuits**

  Also includes a active component such as a transistor or op amplifier.

Lowpass $U_o = \frac{1}{RCs+1} U_i$

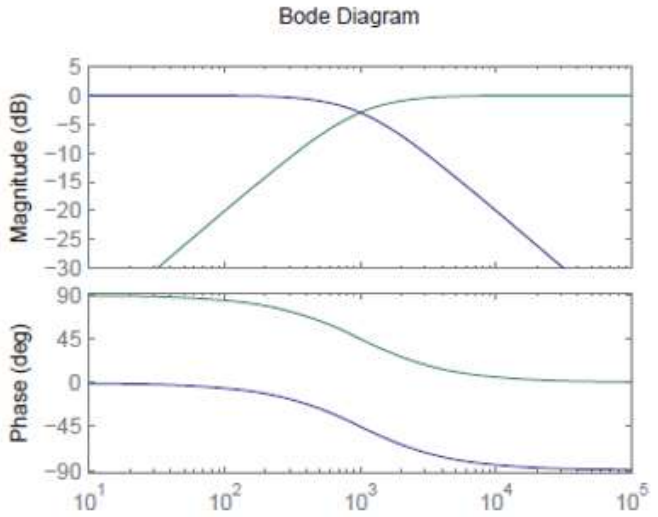


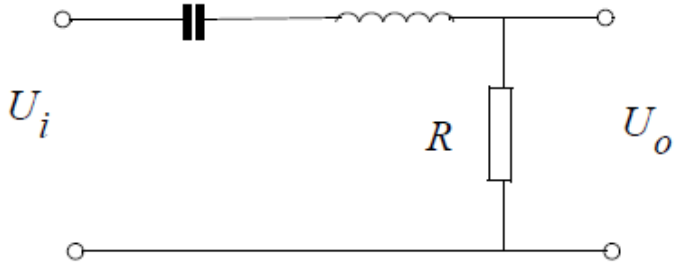
Highpass $U_o = \frac{RCs}{RCs+1} U_i$

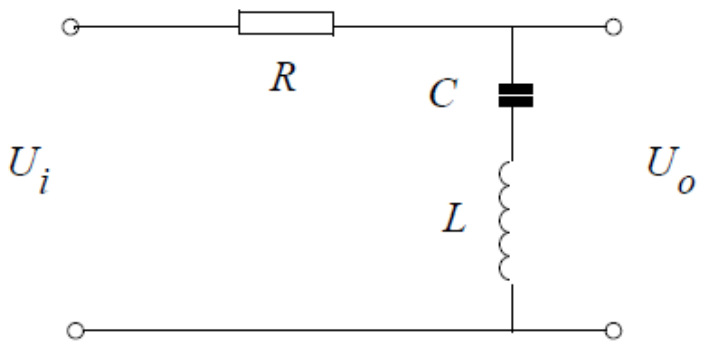


Example with
$R = 10$ kΩ and
$C = 100$ pF.

Also possible as *RL* circuits.

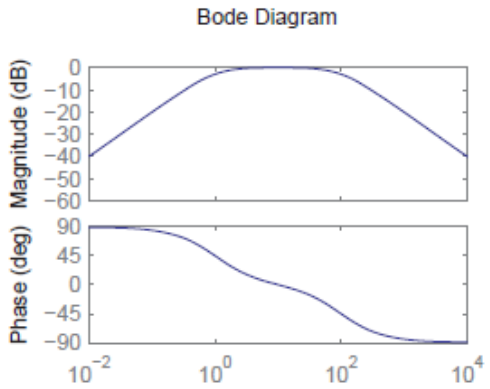






$$U_o = \frac{(R/L)s}{s^2 + (R/L)s + 1/(LC)} U_i$$

$$U_o = \frac{s^2 + 1/(LC)}{s^2 + (R/L)s + 1/(LC)} U_i$$



- Maximum gain of a passive filter is unity.
- The inductor in a passive filter may be large.
- Can require very small capacitors for lowpass filters with low cutoff frequency.
- Active filters are simple to connect in cascade for higher order filters.
- Active filters may become unstable.

- **If the sampling period is slow compared to the frequency of the closed loop poles. Then the cutoff frequency of the lowpass filter will be to close to the**

- **The antialiasing filter gives a undesired phase lag to the controller.**

  $y_a(s) = G_a(s)y(s)$, $y_a(t)$ will lag $y(t)$ in phase.

- **This can be avoided by designing the pole placement controller on $G_p(s)G_a(s)$ instead of only on $G_p(s)$.**

- **The order of the controller polynomial must be increased by the same order as the Low pass filter.**
  - Otherwise is it not possible to choose c.l. poles freely.

- **Using the same example as above with *Ts=12 ms*.**

  - which is 10 times faster than the c.l. poles. $\omega_s = \frac{2\pi}{T_s} = 524$ rad/s

  -chose a 1:st order L.P filter in between ω_s and ω_0. $\omega_a = 5\omega_0 = 250$ rad/s

  - Transfer function for 1:st order L.P. filter $G_a = \frac{1/(RC)}{s + 1/(RC)}$ where $1/(RC) = \omega_a$.

  - Transfer function of both dc motor and filter is 3:rd order

    $$G_{ap}(s) = \frac{b/(Rc)}{s^3 + (a + 1/(RC))s^2 + (a/(RC))s}$$

  - Zoh of $G_{ap}(s)$ is $G_{ap}(z) = \frac{b_2 z^2 + b_1 z + b_0}{z^3 + a_2 z^2 + a_1 z + a_0}$ also of third order

- Select a second order controller $\frac{S}{R} = \frac{s_2 z^2 + s_1 z + s_0}{z^2 + r_1 z + s_0}$

- 5:th order closed loop polynomial

$$Acl := z^5 + (a2 + r1 + b2\,s2)\,z^4 + (b1\,s2 + b2\,s1 + a1 + a2\,r1 + r0)\,z^3 + (b0\,s2 + b1\,s1 + b2\,s0 + a0 + a1\,r1 + a2\,r0)\,z^2 + (a0\,r1 + a1\,r0 + b0\,s1 + b1\,s0)\,z + a0\,r0 + b0\,s0$$

- Select the desired c.l. polynomial: $A_m(z)$ same order as $A(z) = 3$. Then $A_o(z)$ must be of second order.

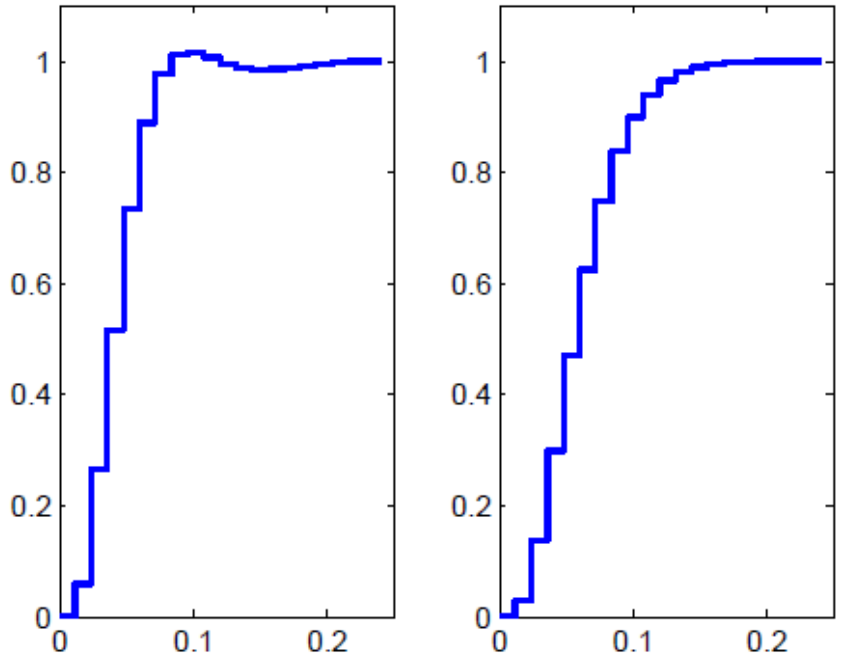
$$A_{cl} = A_m A_o = \underbrace{(z^2 + p_{m1} z + p_{m0})(z + p_m)}_{A_m} \underbrace{(z^2 + p_{o1} z + p_{o0})}_{A_o}$$

- Solve for the unknown control parameters, r_i and s_i as a function of the known process parameters, a_i and b_i and desired polynimial coefficients p_i.

- Calculate $T(z) = A_o(z) t_0$

With antialiasing filter,
but it is not included
in the design
(second order controller).

With antialiasing filter,
and it is included
in the design
(Third order controller).



- Two design concepts (continuous, discrete, combination of both)
- Rules of thumb for selection of sampling period.
- However, often the microprocessor gives the minimum sampling period.
- If the sampling period is to "close" to the c.l. poles is it better to design the controller in discrete time.
- Antialiasing filter is absolutely necessary when you are using an analog sensor as feedback signal.
- If the lowpass filter frequency is to "close" to the c.l. poles, include it in the controller design.
- Analysis of the effects of the transformation of the control design from continuous to discrete time e.g., phase and amplitude margins.

References

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