Dynamics and motion control

Lecture 4
Feedback control
-discrete time control design

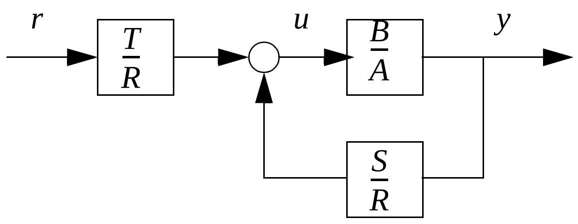
Bengt Eriksson, Jan Wikander
KTH, Machine Design
Mechatronics Lab
e-mail: benke@md.kth.se

4.1. Lecture outline

- **1. Introduction**
- 2. Sampling of continuous signals
- 3. Comparison of discrete and continuous time poles.
- 4. Pole placement control design in discrete time
- 5. Example
- 6. Selecting sampling period and the effect of aliasing
- 7. Designing antialiasing filters
- 8. Influence on performance from the antialiasing filters

4.1.1. From Last lecture

- **Polynomial approach to pole placement design.**



- **Translate specifications from time domain into c.l. poles.**
- **Choose the control structure, $\frac{S}{R}$.**
- **Solve the Diophantine equation $A_m A_o = AR + BS$**
- **Calculate the feed forward polynomial T**
- **Approximate the discrete time controller with e.g. Tustin.**

4.1.2. Two digital control design possibilities

- 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.

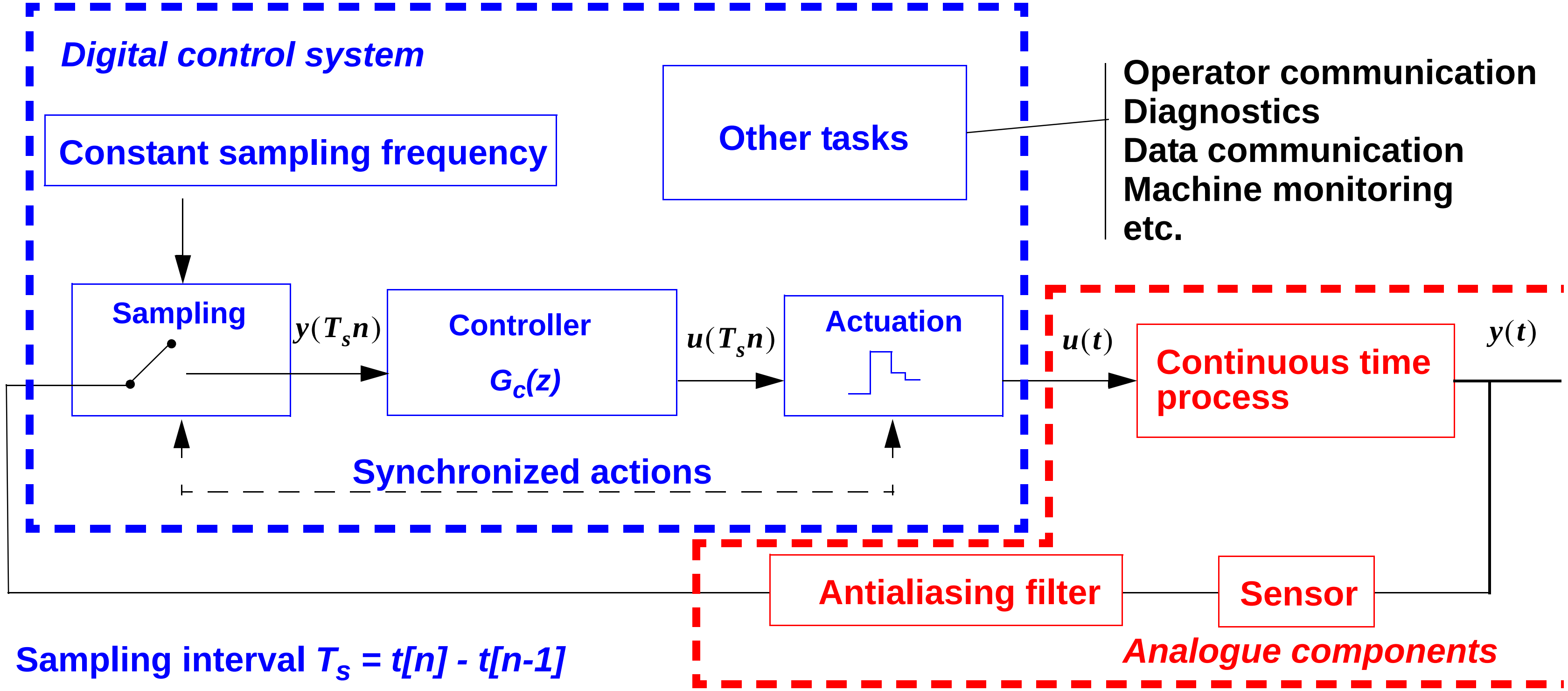
4.1.3. Control design in discrete time

- 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.

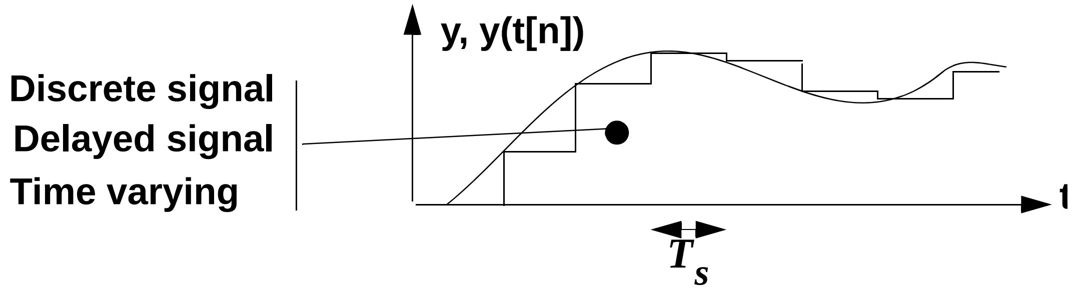
4.2. Lecture outline

4.2.1. Discrete time control



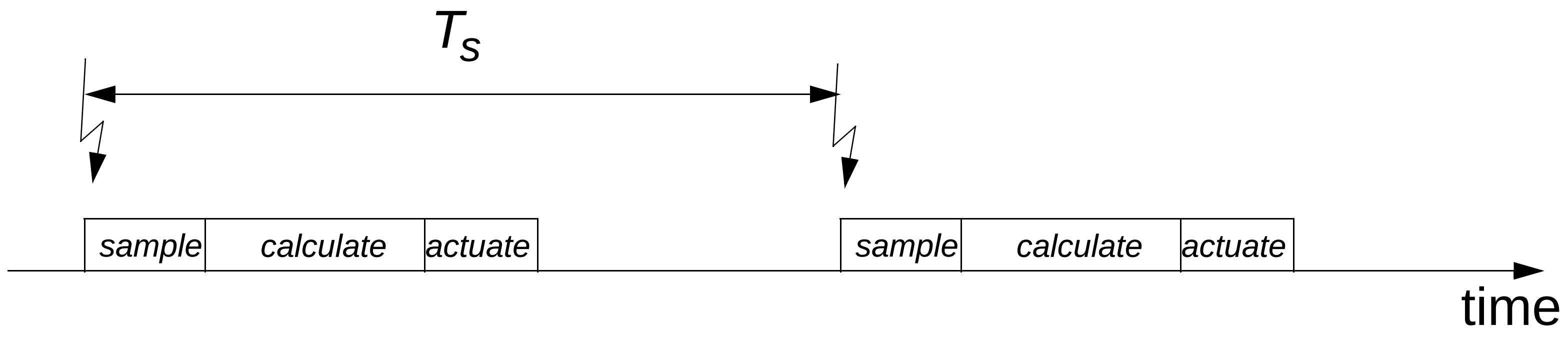
Sampling interval $T_s = t[n] - t[n-1]$

The control signal is typically constant over the sampling interval, ZOH.

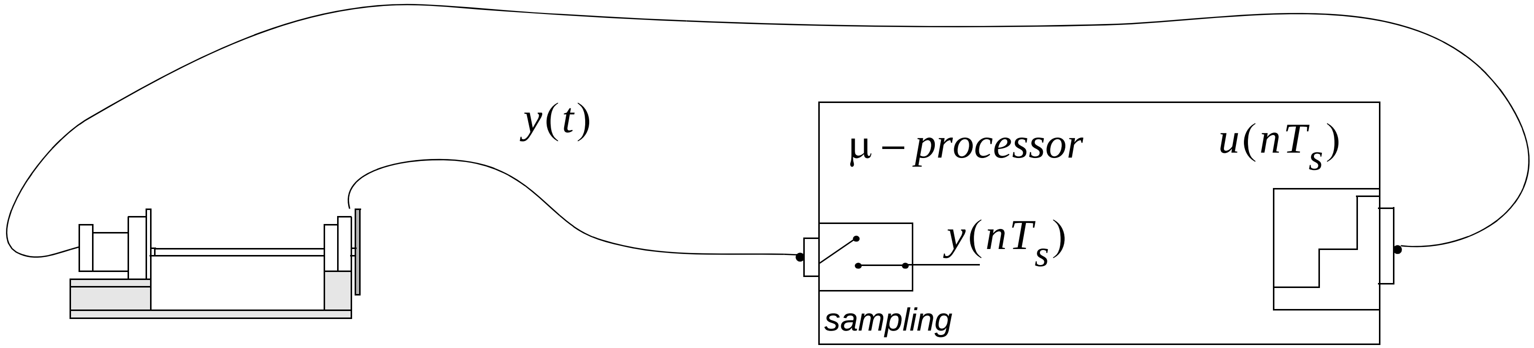


4.2.2. Assumptions/consequences

- 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)



4.2.3. The concept of ZOH -actuation



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 $G_1(s)$.

4.2.4. Calculate the zoh model

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 $\boldsymbol{t}[\boldsymbol{n}]$. Given that we know the state vector at sample $\boldsymbol{t}[\boldsymbol{n}]$ we may integrate the continuous time system from $\boldsymbol{t}[\boldsymbol{n}]$ to $\boldsymbol{t}[\boldsymbol{n}+\mathbf{1}]$. 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.

4.2.5. Continued.

Assuming that the system is sampled with sampling interval $\boldsymbol{T_s}$, i.e. $\boldsymbol{t}[\boldsymbol{n}+\mathbf{1}]-\boldsymbol{t}[\boldsymbol{n}] = \boldsymbol{T_s}$ 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*

4.2.6. Example of a zoh sampled model

2nd order continuous time system

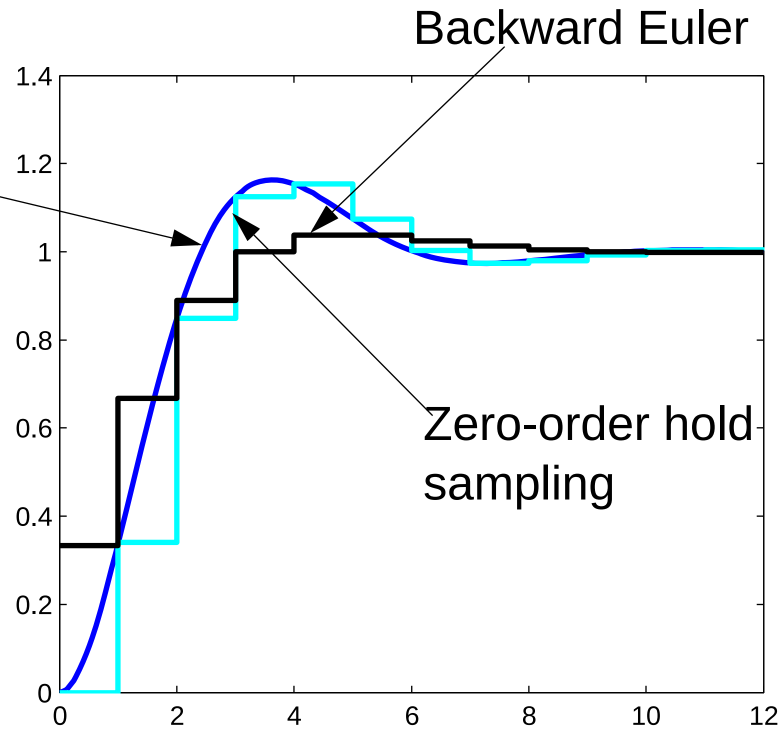
$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

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

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

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

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



Ts = 1s

4.2.7. Discrete time transfer function

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

s.s. model, $$\begin{aligned} x[n+1] &= \Phi x[n] + \Gamma u[n] \\ y[n] &= Cx[n] \end{aligned}$$

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

hence, $$\begin{aligned} (z-\Phi)x[n] &= \Gamma u[n] \\ y[n] &= Cx[n] \end{aligned}$$

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

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

4.2.8. Summary ZOH

- **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.**

4.3. Lecture outline

- 1. Introduction
- 2. Sampling of continuous signals
- **3. Comparison of discrete and continuous time poles.**
- 4. Pole placement control design in discrete time
- 5. Example
- 6. Selecting sampling period and the effect of aliasing
- 7. Designing antialiasing filters
- 8. Influence on performance from the antialiasing filters

4.3.1. Discrete time control design

- **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).*

4.3.2. First order discrete time 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. (See lecture 2.1)

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

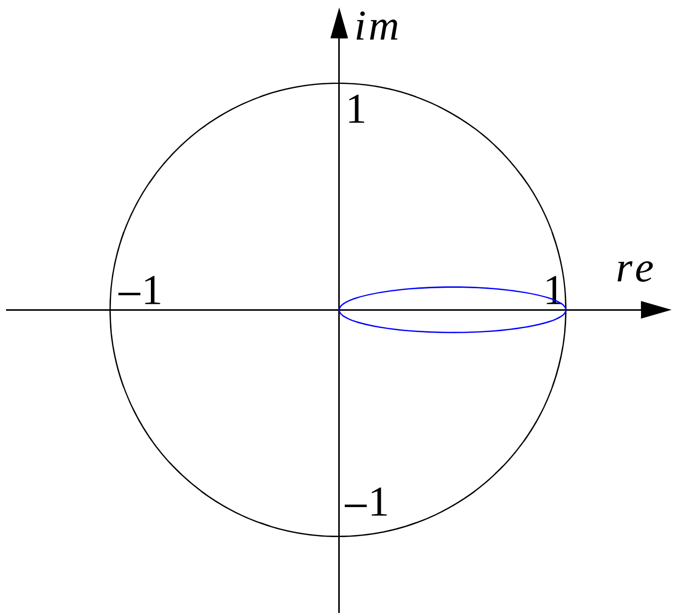
  which gives a discrete time pole at

  $z = e^{sT_s} = e^{-aT_s}$ 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!) -> $z = 0$

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



4.3.3. Second order discrete time poles

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

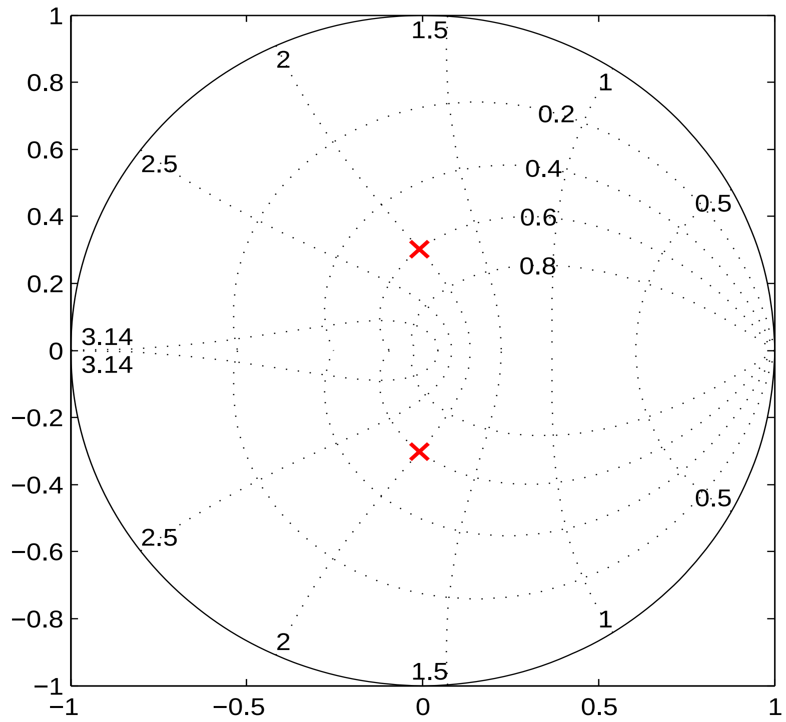
$z_{1,2} = e^{s_{1,2}T_s}$ gives the second order discrete time polynomial,

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

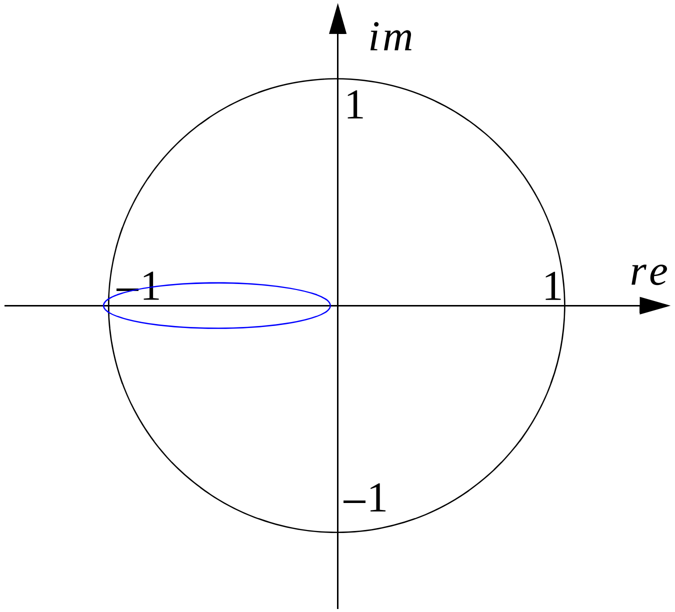


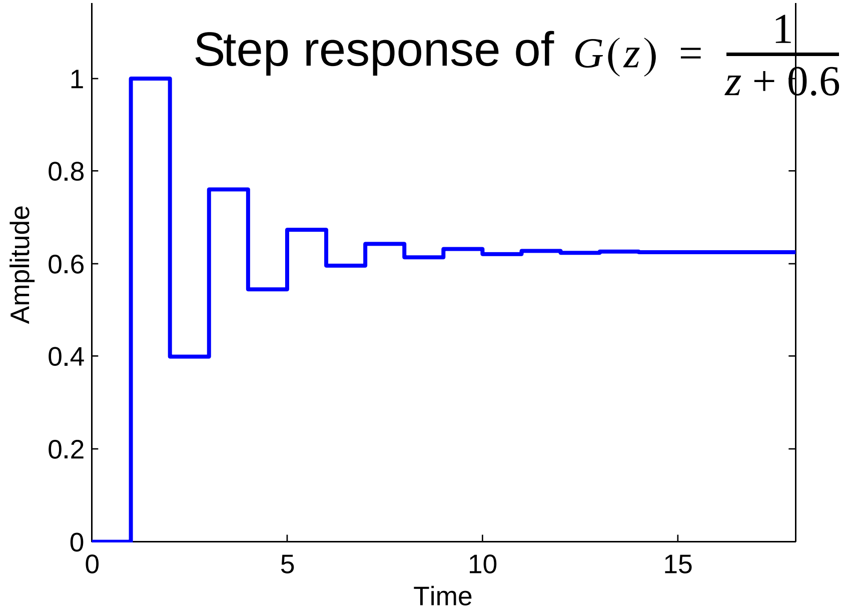
The frequencies are scaled with Ts.
And that π is the highest freq. (Nyquist freq.)

4.3.4. A special case

- 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 $T_s = 1$





4.4. Lecture outline

- 1. Introduction
- 2. Sampling of continuous signals
- 3. Comparison of discrete and continuous time poles.
- **4. Pole placement control design in discrete time**
- 5. Example
- 6. Selecting sampling period and the effect of aliasing
- 7. Designing antialiasing filters
- 8. Influence on performance from the antialiasing filters

4.4.1. Pole placement

- 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 $AR + BS = A_m A_o$.

4.4.2. Control structures

- **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 $u[n] = -r_0 u[n-1] - s_1 y[n] - s_0 y[n-1]$

- **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 $\frac{S(z)}{R(z)} = \frac{s_2 z^2 + s_1 z + s_0}{(z-1)(z + r_0)}$

4.4.3. Evaluation

- **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).

4.4.4. Summary

- **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

4.5. Lecture outline

4.5.1. Example

- 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 $\{s_1, s_0, r_0\}$.

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

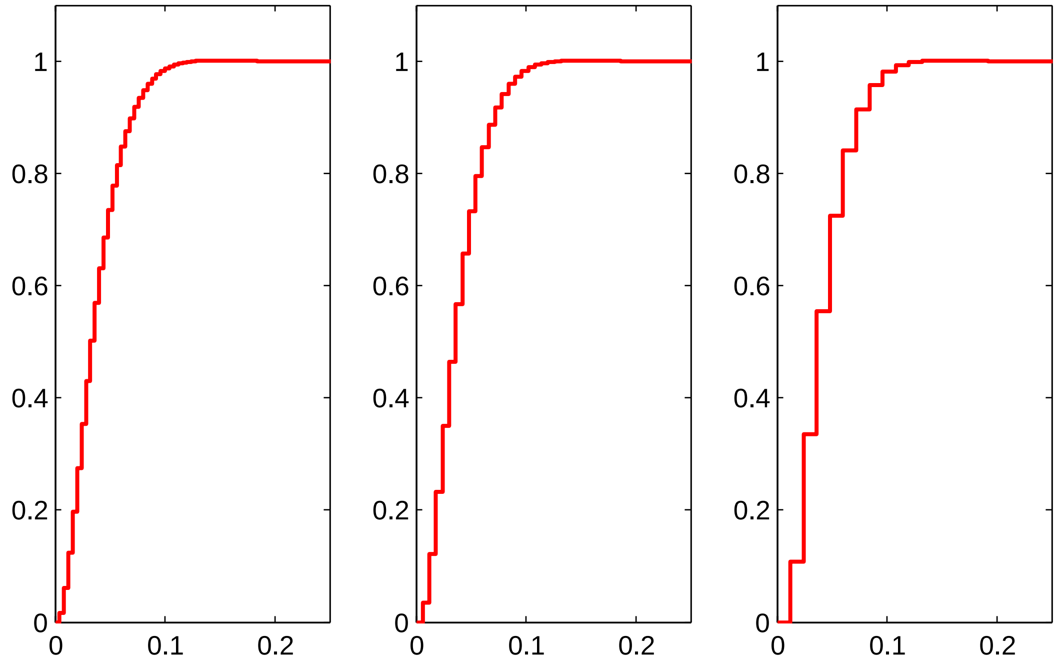
$$s0 = -\frac{-a0\ a1\ b0 + a0^2\ b1 - a0\ b1\ p1 + a0\ p2\ b0 + p0\ a1\ b1 - p0\ b0}{-b0\ a1\ b1 + b0^2 + a0\ b1^2},$$

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

- 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!

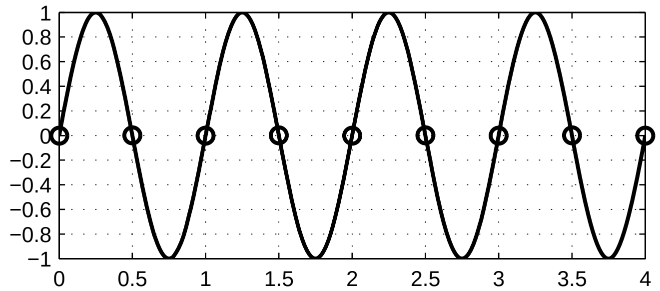


4.6. Lecture outline

- 1. Introduction
- 2. Sampling of continuous signals
- 3. Comparison of discrete and continuous time poles.
- 4. Pole placement control design in discrete time
- 5. Example
- **6. Selecting sampling period and the effect of aliasing**
- 7. Designing antialiasing filters
- 8. Influence on performance from the antialiasing filters

4.6.1. Sampling continuous time signals

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

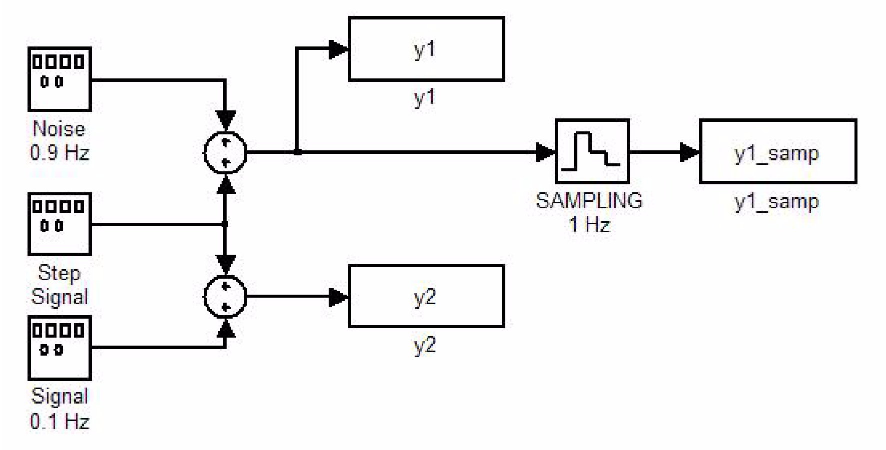


It clearly appears that sampling at this frequency gives nothing!

- **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 $2\omega_0$.

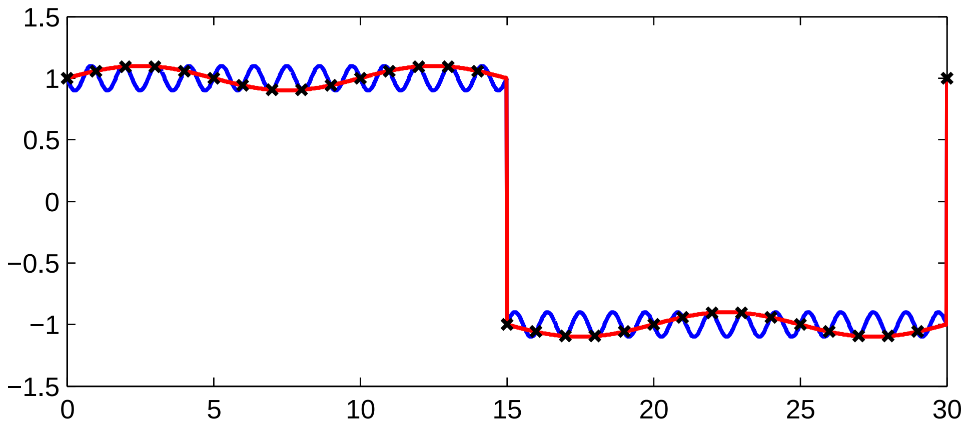
- In a sampled system the important frequency $\omega_s/2$ is also referred to as the Nyquist frequency. Signals with frequency lower than the Nyquist frequency can be reconstructed after sampling.

4.6.2. Aliasing and new frequencies



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).



4.6.3. Numeric example of aliasing

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 $n = 1\ldots\infty$

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

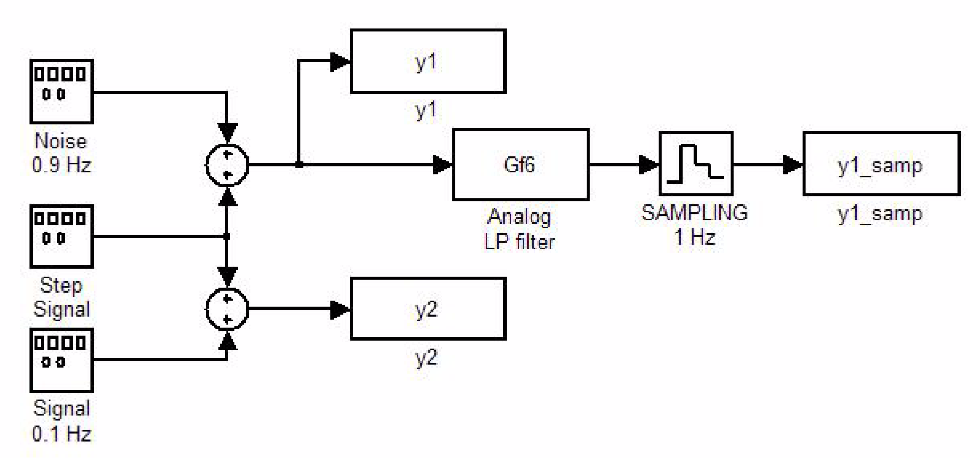
$$= \cos(2\pi n - 0.2\pi n)$$

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

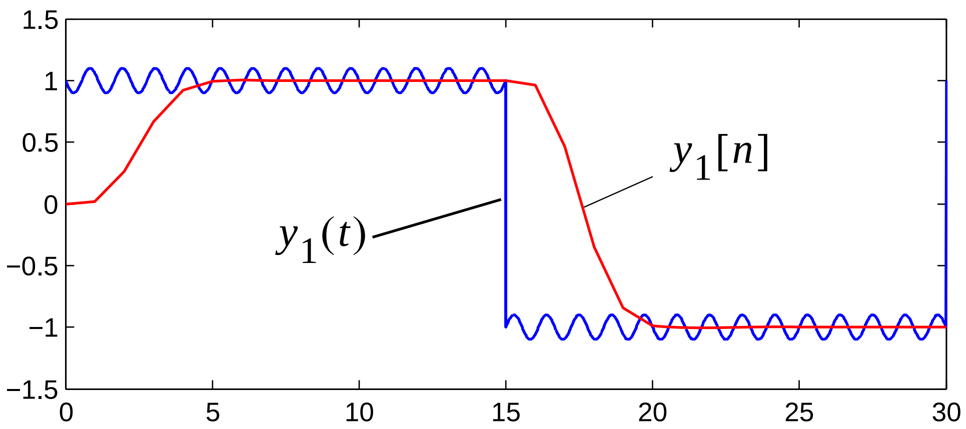
$$= \cos\left(2\pi 0.1 \frac{n}{1}\right) = \cos(2\pi f_2 t)$$

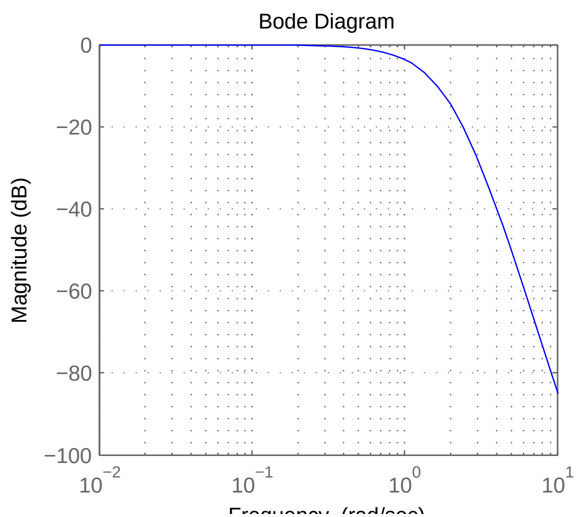
The frequency $f_1 = 0.9$ *Hz* is said to be an *alias* of the frequency $f_2 = 0.1$ *(Hz)* when it is sampled at 1 Hz.

4.6.4. Avoiding aliasing with analog LP filter



Analog pre filter





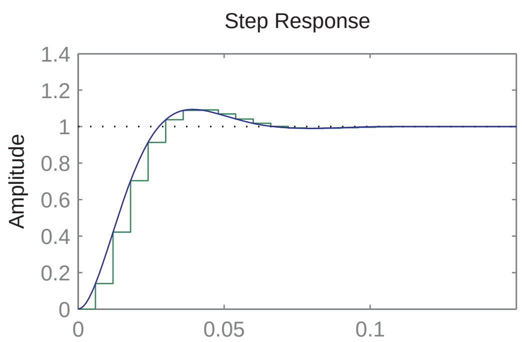
4.6.5. Pre- or antialiasing filtering

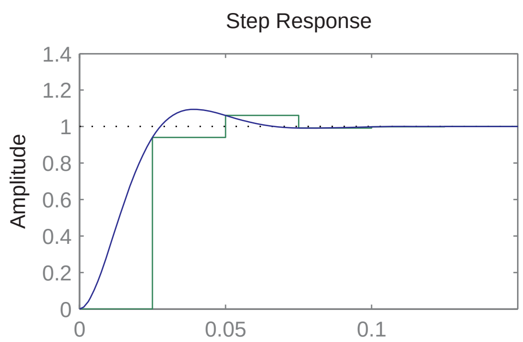
- Real signals are not band limited. High frequency components must be filtered away to avoid aliasing.
- For digital/discrete sensors the aliasing problem is less aggravating.
- The prefilter is typically implemented as an analog filter with resistors, capacitors and an operational amplifier.
- The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

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

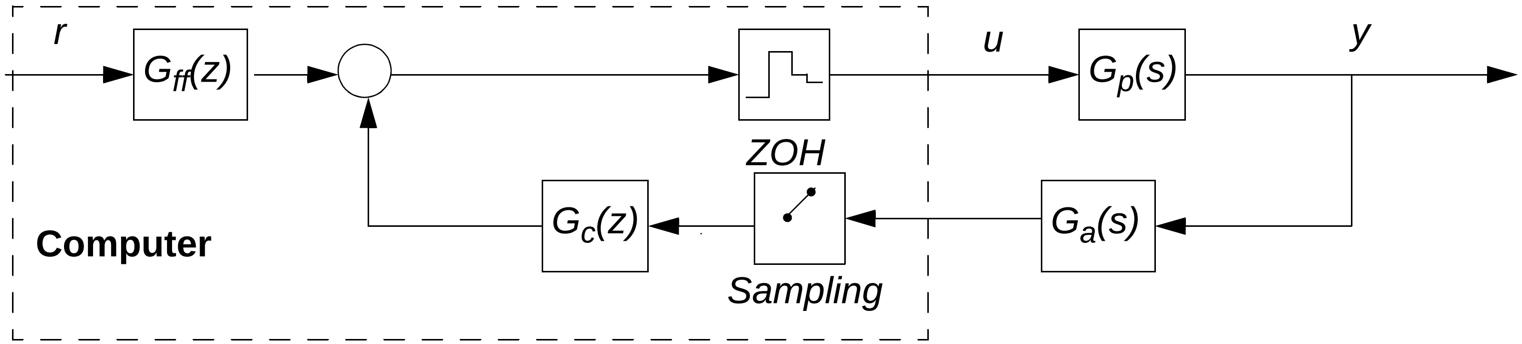
4.6.6. Choice of sampling rates

- 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

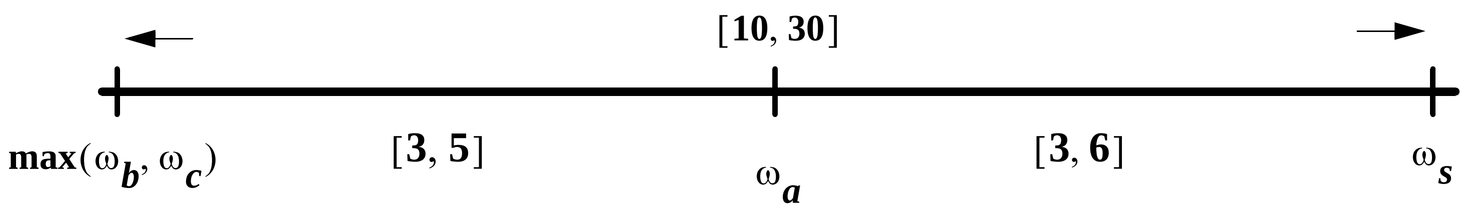




4.6.7. Sampling interval selection based on freq.



- **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 $\omega_a \in [0.17, 0.33]\omega_s$**



4.6.8. Sampling period based on reality!

- **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.

4.7. Lecture outline

4.7.1. Design of analog filters

- **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

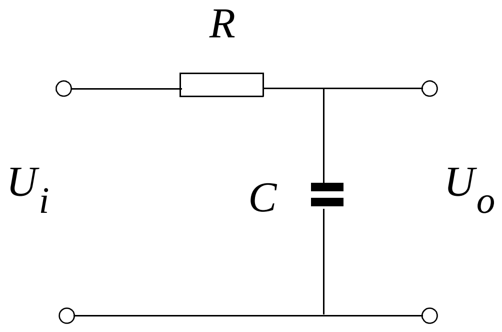
  Bandstop filter for blocking of specific frequencies
- **Passive circuits**

  Based only on resistors, capacitors and inductors
- **Active circuits**

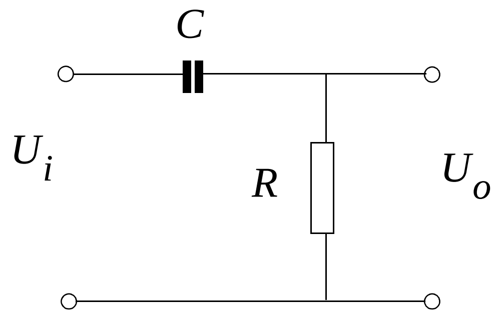
  Also includes a active component such as a transistor or op amplifier.

4.7.2. Passive filters

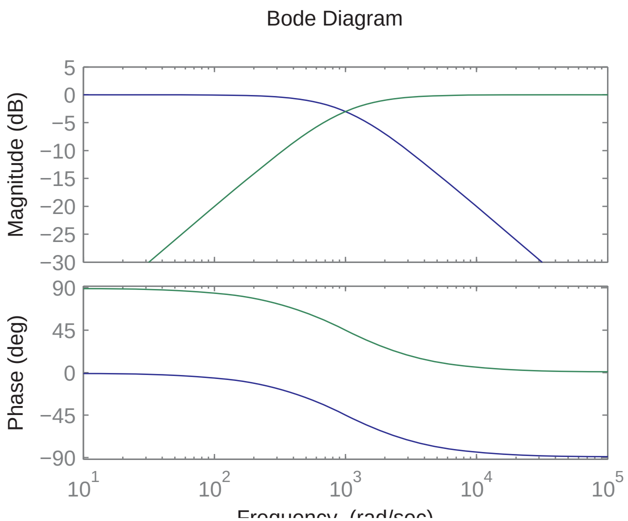
Lowpass $U_o = \frac{1}{RCs + 1} U_i$



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

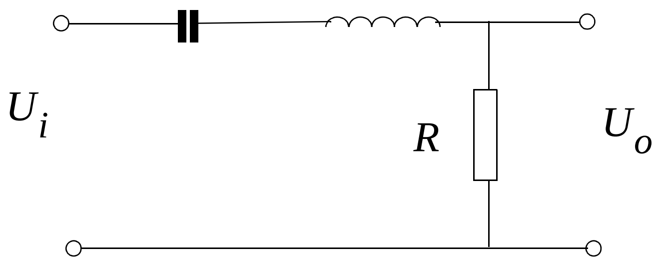


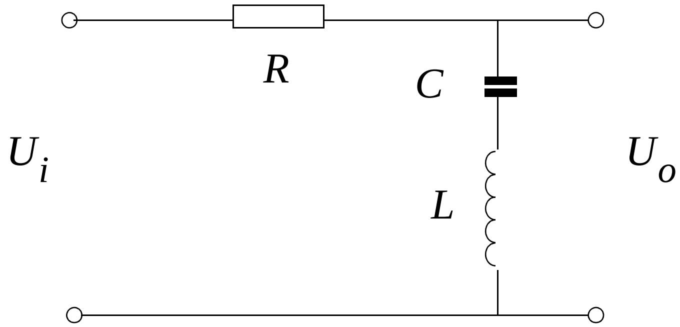
Example with
$R = 10$ kΩ and
$C = 100$ pF.



Also possible as *RL* circuits.

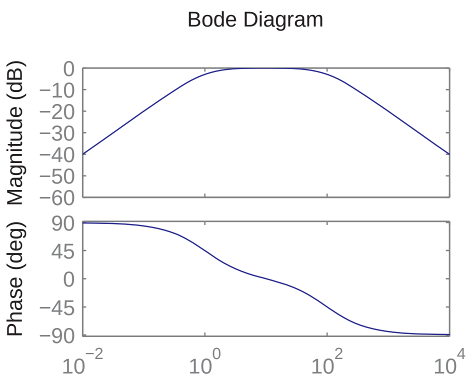
4.7.3. Bandpass





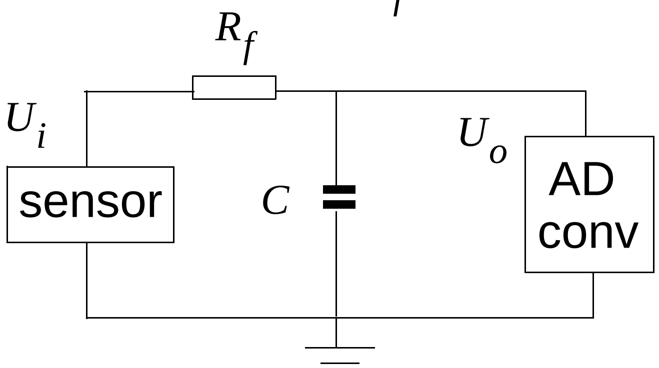
$$U_o = \frac{(R/L)s}{s^2 + (R/L)s + 1/(LC)} U_i$$

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

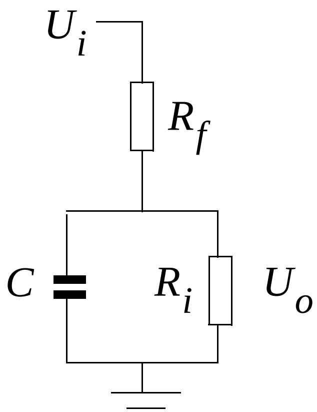


4.7.4. How to select the components

Lowpass $U_o = \frac{1}{R_f C s + 1} U_i$



R_i is the input resistance of the AD-conv. Typically around $1 M\Omega$



$$U_o = \frac{R_i}{R_i R_f C s + R_i + R_f} U_i$$

dc-gain $\frac{R_i}{R_i + R_f}$

time constant $\tau = \frac{R_i + R_f}{R_i R_f C}$

- Example with $R_i = 1 M\Omega$, $\tau = 1 ms$
- Select Capacitor -> $C = 100 pF$
- gives $R_f = 10 k\Omega$ and dc-gain = 0.99

4.7.5. Active filters with op-amp

- **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.**

4.8. Lecture Outline

- 1. Introduction
- 2. Sampling of continuous signals
- 3. Comparison of discrete and continuous time poles.
- 4. Pole placement control design in discrete time
- 5. Example
- 6. Selecting sampling period and the effect of aliasing
- 7. Designing antialiasing filters
- **8. Influence on performance from the antialiasing filters**

4.8.1. Incl. the antialiasing filter in the design

- **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.

4.8.2. Example with antialiasing filter

- **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.

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

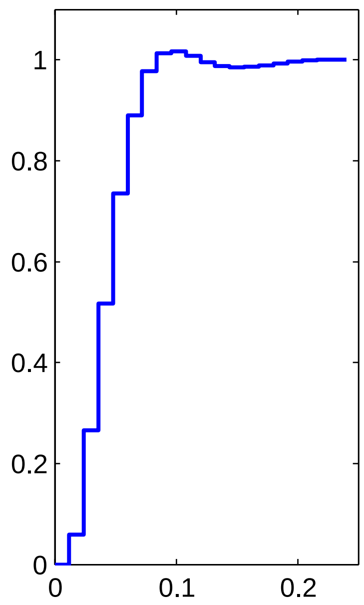
- Solve for the unknown control parameters, r_i and s_i as a function of the known process parameters, a_i and b_i and desired polynimial coefficients p_i.

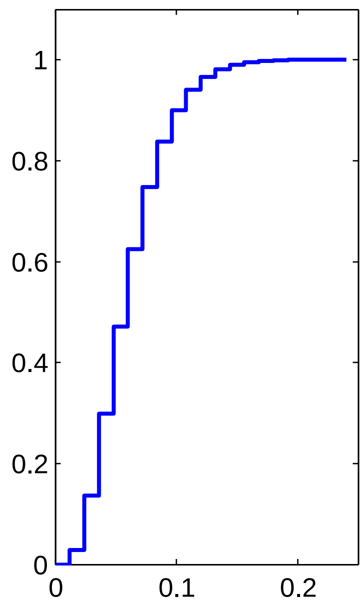
- Calculate $T(z) = A_o(z) t_0$

4.8.3. Results

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).





4.8.4. Summary

- 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.