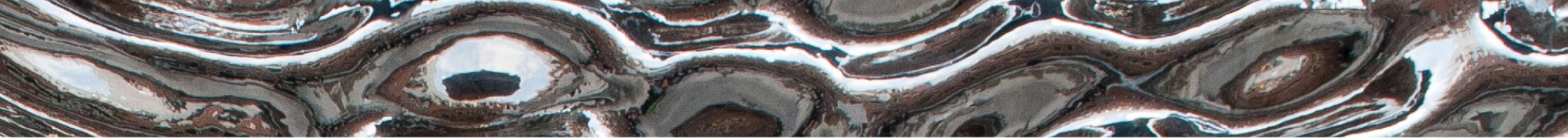


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN 1080

Kontrollsystemer og PID

Yngve Hafting, 2022

Beskjeder

- Videoer til denne forelesningen kan ses i canvas filer->Videoer

Hvor står vi og hvor går vi…

Kort om emnet

- *Grunnleggende analog elektronikk*, *sensorer og sensor grensesnitt, aktuatorer*. ***Programmering av mekatroniske systemer***.

Hva lærer du?

Etter å ha tatt IN1080 kan du:

- *forstå virkemåten til analoge kretser. Aktuelle begreper er: strøm, spenning, motstand, effekt, impedans, likestrøm, vekselstrøm, RCL, MOS, FET, OPAmp*
- *bruke klassiske analysemetoder basert på Kirchoff, Thevenin og Nortons teoremer*
- *forstå og anvende sensorer, signalkondisjonering og konvertering, samt noen komponent-komponent busser*
- *bygge og programmere enkle mekatroniske systemer med mikrokontroller, aktuatorer og sensorer*
- **forstå grunnleggende kontrollteori og virkemåte for PIDkontrollere**

Lab

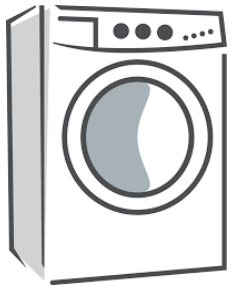
- Oblig 6: PWM og PID

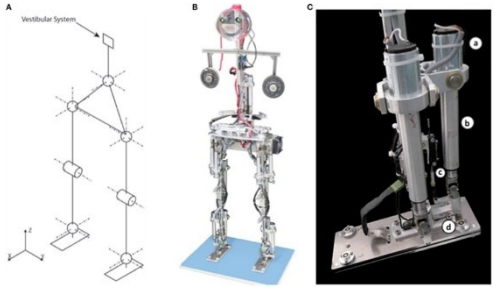
Forelesning

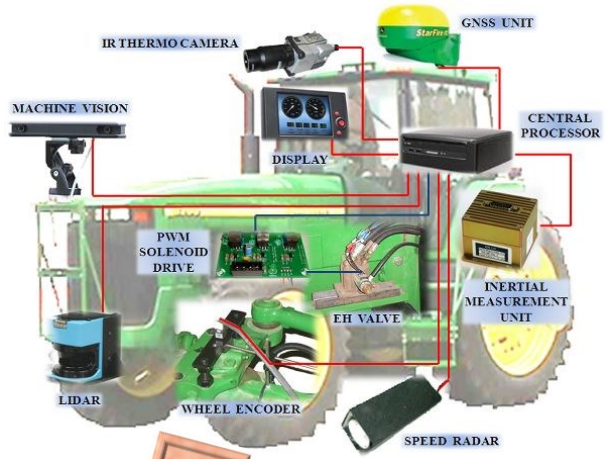
- Kunne forstå og redegjøre for grunnleggende kontrollteori
  - åpen sløyfe (open loop)
  - lukkede sløyfer (med feedback)
  - PID kontroll
    - tuning

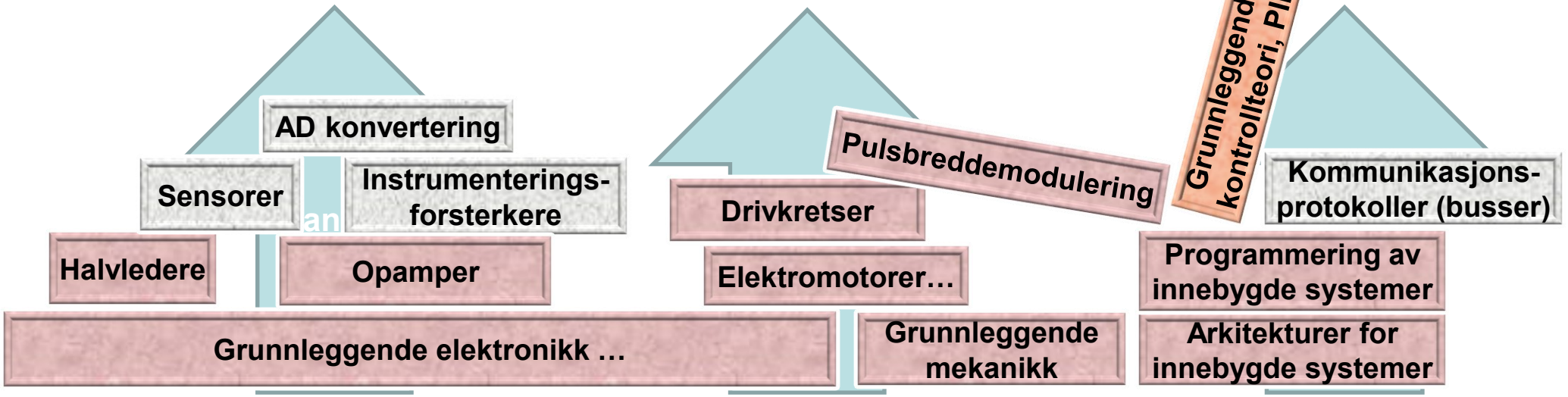
Hvor står vi – hvor går vi…

Formål: Å lage og programmere mekatroniske systemer

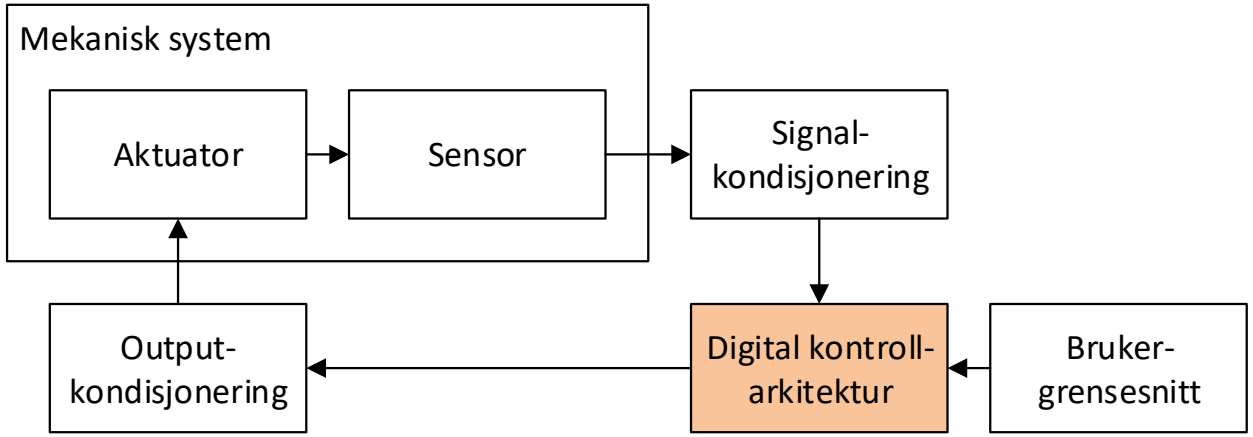








Systemperspektiv og oversikt



- Hvorfor kontrollsystemer
- Åpne og lukkede kontrollsløyfer
- Bang-bang kontroll
- Potensiometer
- Servoer
- PID
- tuning av PID

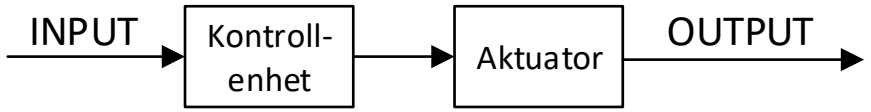
Hvorfor trenger vi et kontrollsystem?

- Eks: Elbil:
  - Vi ønsker at bilen skal kunne kjøre i 50 km/t.
  - Hva slags input har vi?
    - Kommutering, pulsbreddemodulering
    - Pulsbreddemodulering setter strøm
      - Strøm => kraftmoment => akselerasjon
      - Strøm minsker med hastighet (pga selvinduksjon, V_{EMF})
  - Varierende rullemotstand, luftmotstand, etc
    - => Hastigheten følger ikke strøminput alene.
  - Vi må ha en eller annen form for tilbakekobling (feedback) som forteller hvor fort vi kjører.

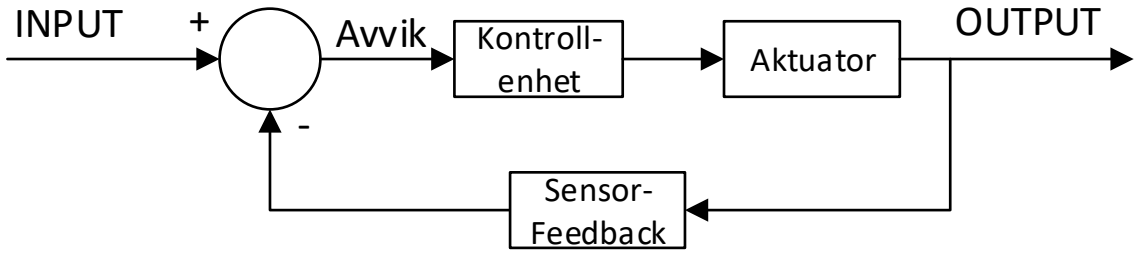
Kilde: Zooop, la voiture électrique de demain? (2006)

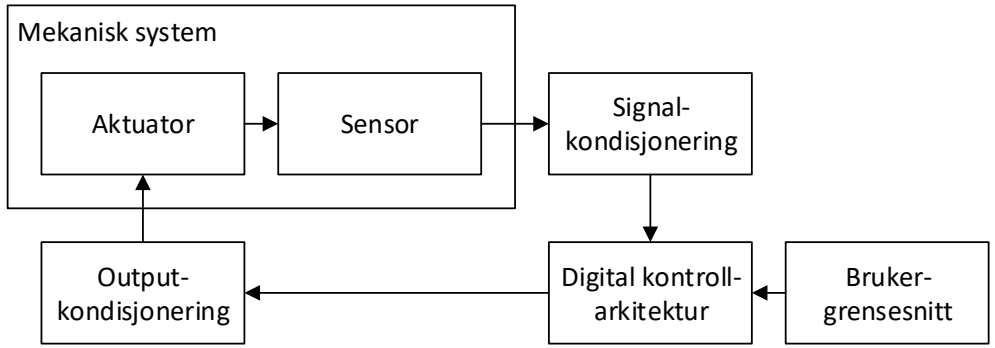
Åpne og lukkede kontrollsløyfer

- Åpen sløyfe (open loop)
  - $Ouput = f(input)$
  - Ingen sensorfeedback
  - Systemer med
    - kun mekaniske komponenter
      - Gass/ brems på eldre biler
    - steppermotorer
      - skrivere/printere
      - noen roboter
    - Menneskelig input
      - volumkontroll på stereoanlegg



- Lukket sløyfe (closed loop)
  - $Ouput = f(input, tilstand)$
  - Brukes der man trenger sensorfeedback
    - «Intelligente systemer» generelt
      - Servomotorer
      - ABS-brems
      - traction control
      - cruise control
      - selvbalanserende kjøretøy
      - Temperaturregulatorer
      - Vaskemaskiner
      - ...

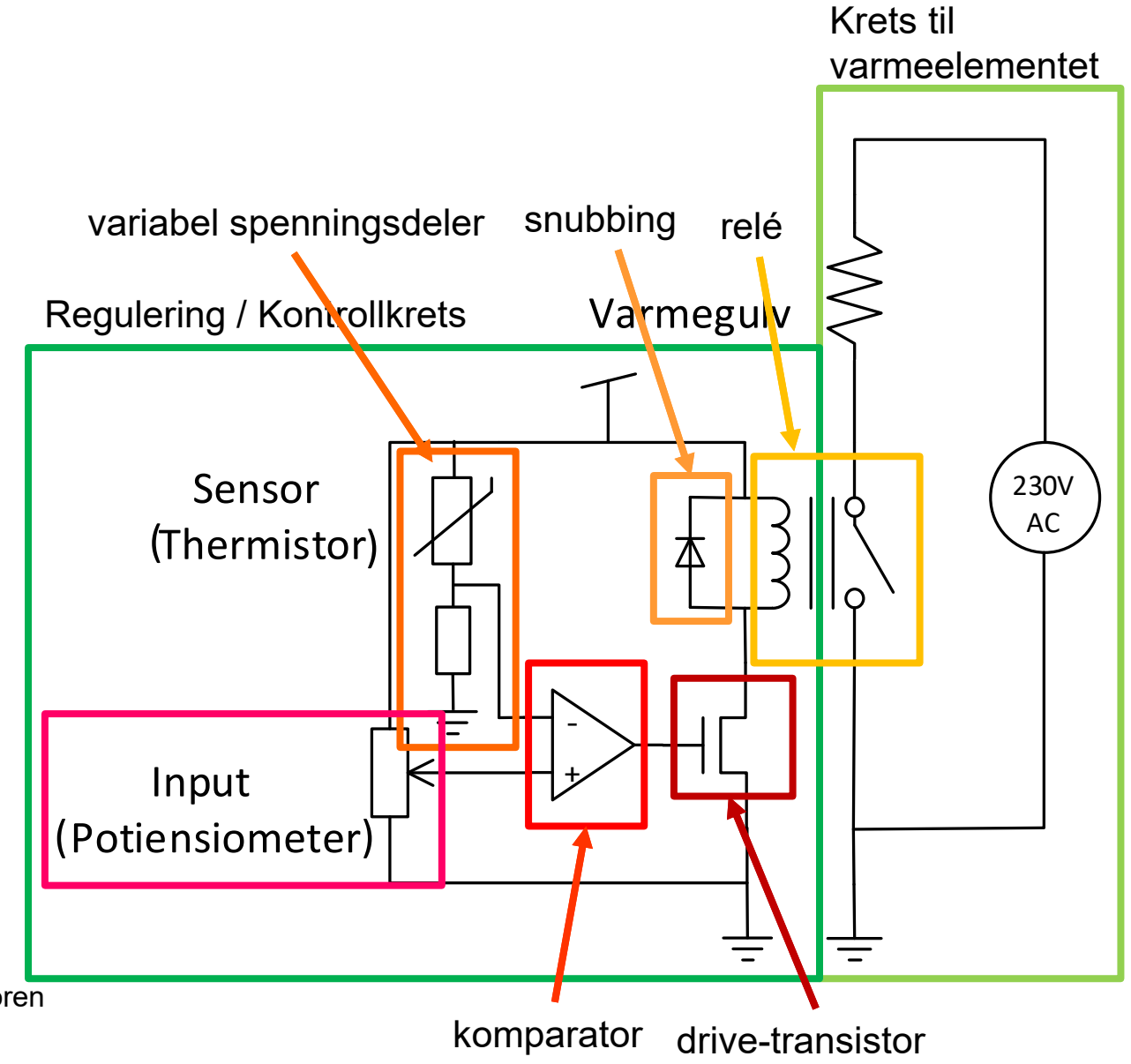




Bang-bang kontroll

- Binær kontroll
  - (System enten av eller på)

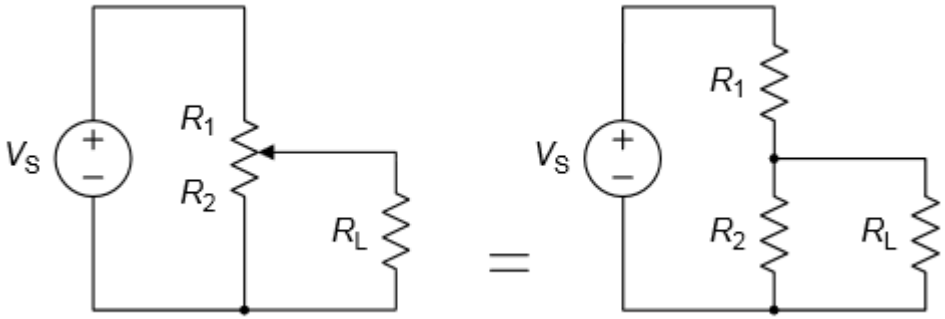
- Eks:
  - Varmegulv
    - Reléet
      - styrer strømmen i varmegulvet
      - gir galvanisk skille gulv ↔ styrekrets
    - Varmekretsen
      - er enten på eller av
    - Reguleringskretsen
      - Styrer releet ut i fra
        - » Spenningen i potensiometeret
        - » Spenningen i spenningsdeleren med thermistoren

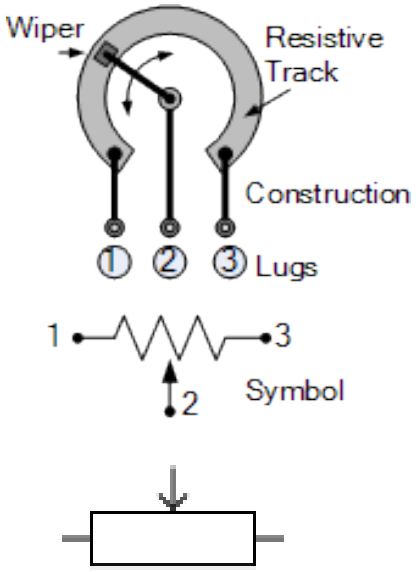


https://en.wikipedia.org/wiki/Potentiometer

Potentiometer

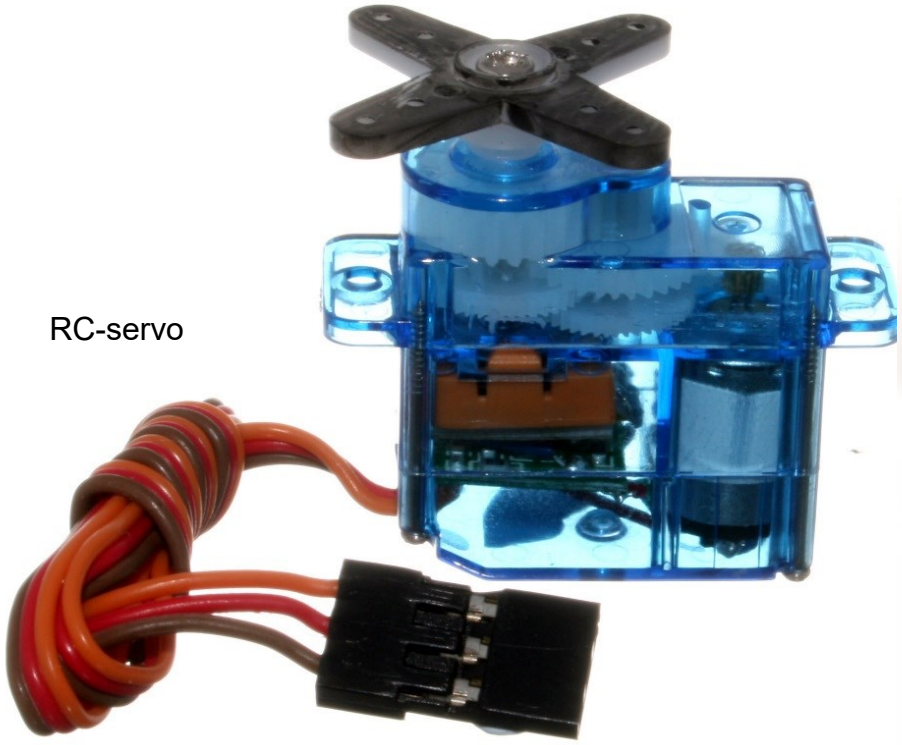
- Et potentiometer er en variabel spenningsdeler
  - Svært ofte er de dreibare, slik som vist på bilde/figur.
  - Kan også brukes som en variabel resistans
- Brukes f.eks i
  - servoer
  - skruknapper





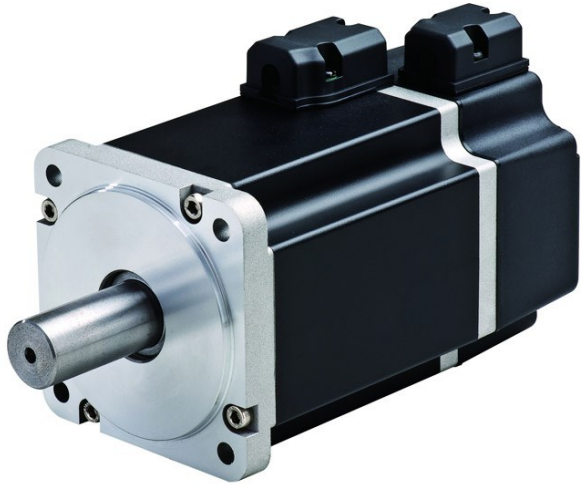
Servoer

- = en aktuator med et kontrollsystem i en lukket sløyfe (*closed loop*).
- Oftest elektrisk motor, trenger ikke være det.
  - Elektriske servomotorer er gjerne giret ned betraktelig for å oppnå høy presisjon og en passende hastighet.
- Kontrollsystemet gir tilbakemelding (Feedback) på hvor servoen befinner seg, og sørger for at motoren finner den posisjonen/vinkelen som er ønsket.
- Servo med potentiometer (7 min): ← *vise denne*
- *Servo koblet til arduino (10 min): https://www.youtube.com/watch?v=LXURLvga8bQ*



RC-servo

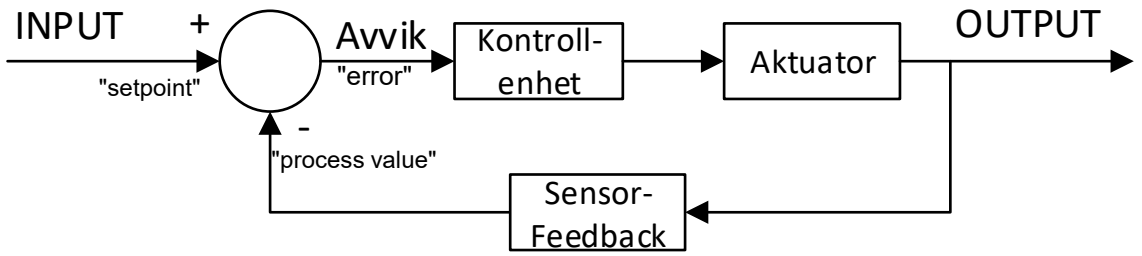
Dynamixel Servo



750W Industriell servo motor (Hiwin)

Kontroll og PID

Generelt kontrollsystem, lukket sløyfe:



- I en lukket sløyfe «*Closed loop*», bruker man sensordata sammen med input til å justere output.

- Avviket, (feil/«*error*»), avgjør hvilke signaler vi sender til aktuatoren.
  - **Målet til kontrollsystemet er å oppnå null avvik...**

- **Proporsjonal kontroll**
  - En servomotor i sin enkleste form har dette (*forrige slide, første video*)
  - K_p, proporsjonalitetskonstanten
    - bestemmer hvor kraftig styring vi gjør ved et gitt avvik
  - Kun gjeldende avvik bestemmer styrken på input
    - Til forskjell fra direkte kontroll...
    - Hva om lasten til servomotoren trekker den ut av posisjon?
      - *Vil systemet klare å oppnå null avvik?..*

$$P = K_p \cdot Avvik$$

Begrep: Stegrespons «step response»

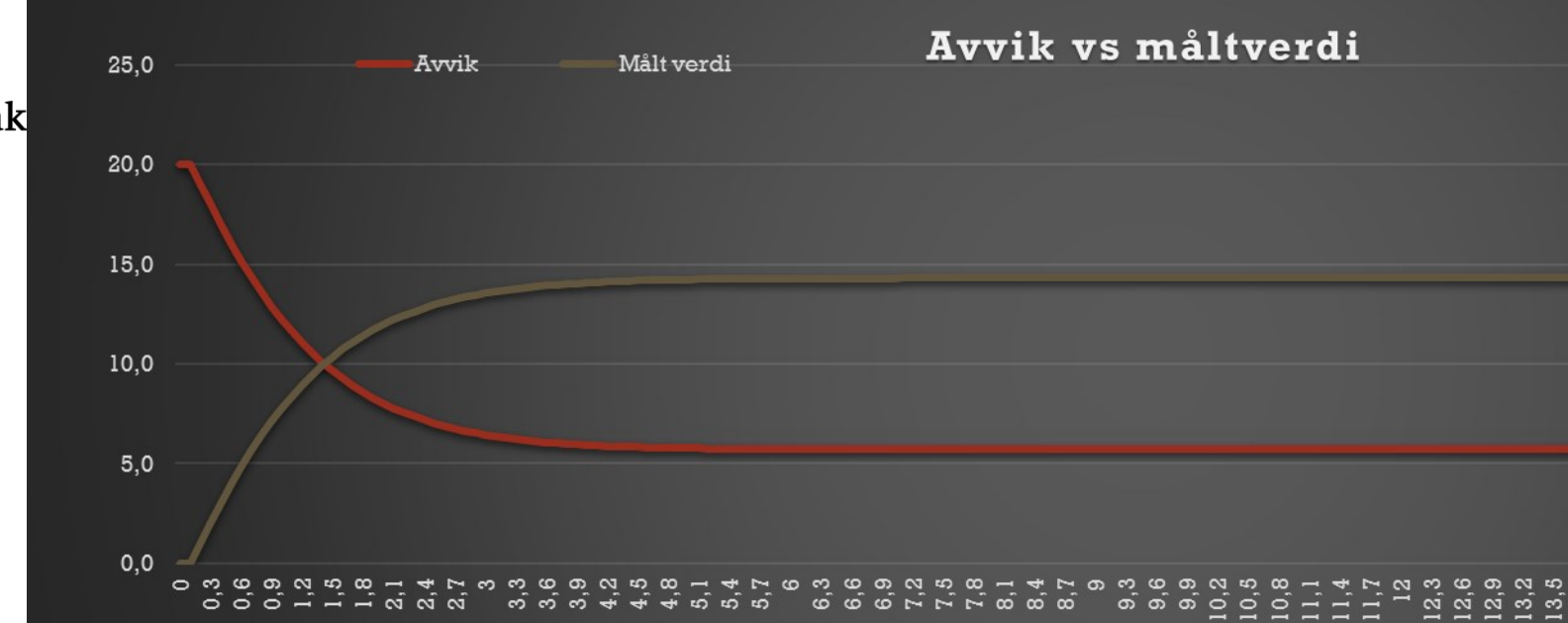
- Inngangssignalet gis én verdi
  - ved tiden t=0
  - Verdien holdes
- Stegresponsen til et systemet er forløpet til utgangen

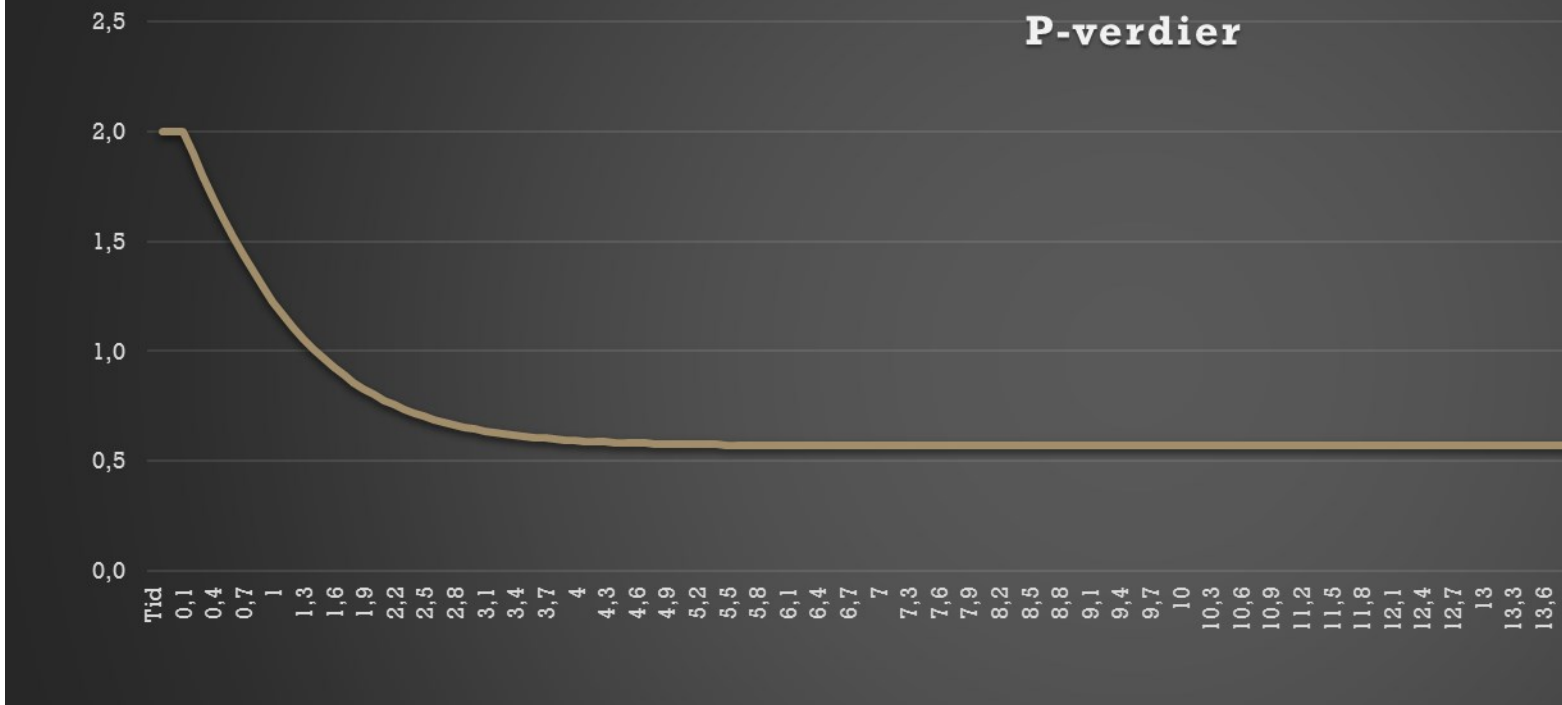
- Andre typer responser
  - Impulsrespons / enhets impulsrespons
    - Respons til et system når input kan summeres til 1 over et infinitesemalt kort intervall
  - Ramp respons
  - Etc. (disse er ikke eget tema i IN1080, så vi må vite at det er noe annet…)

Bare P

$$K_p \cdot Avvik$$

- Hvis Kp ikke er for stor, vil en proporsjonal kontroller stabilisere seg under mål/ønsket verdi (her: setpoint =20)
- Forløpet vi ser her er *overdempet* (over-damped)

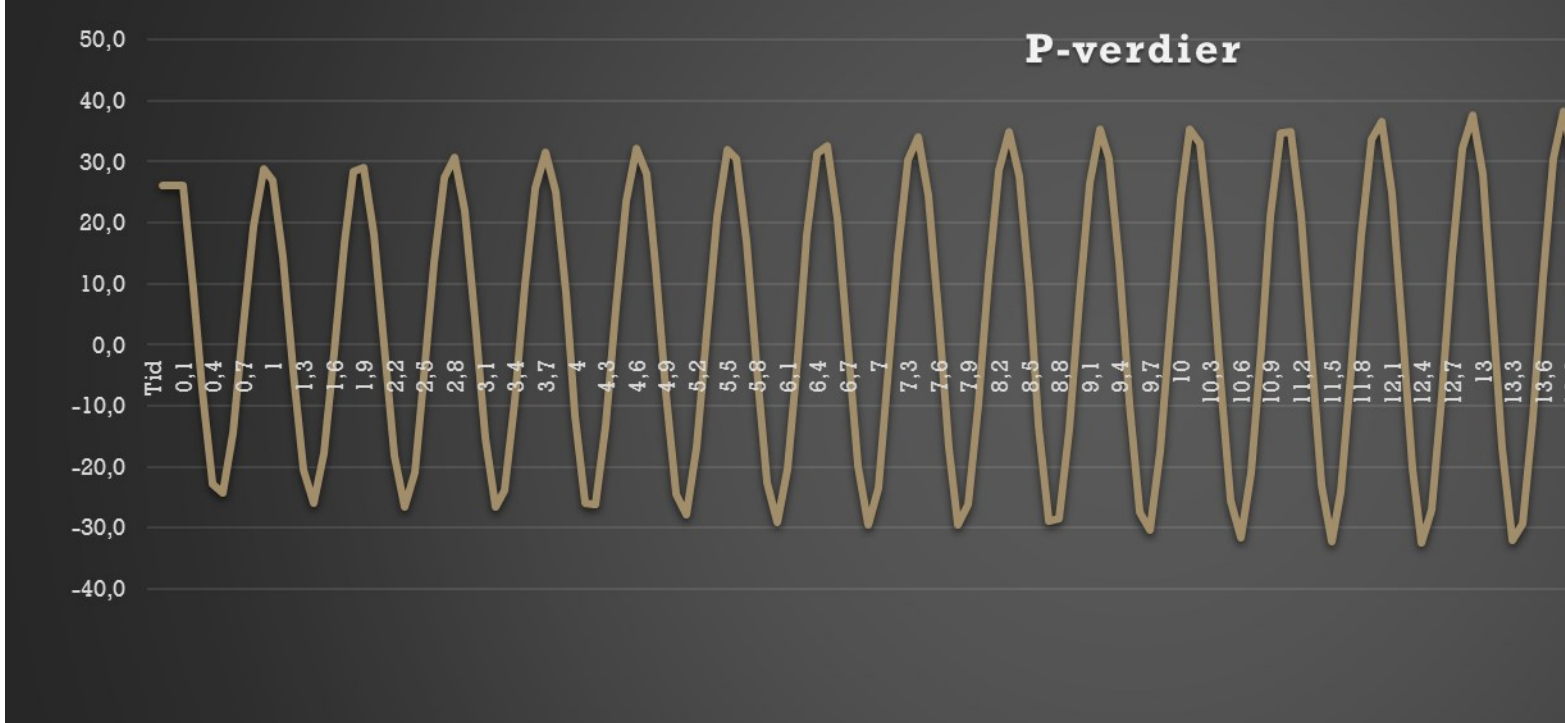




Bare P (forts)

$$K_p \cdot Avvik$$

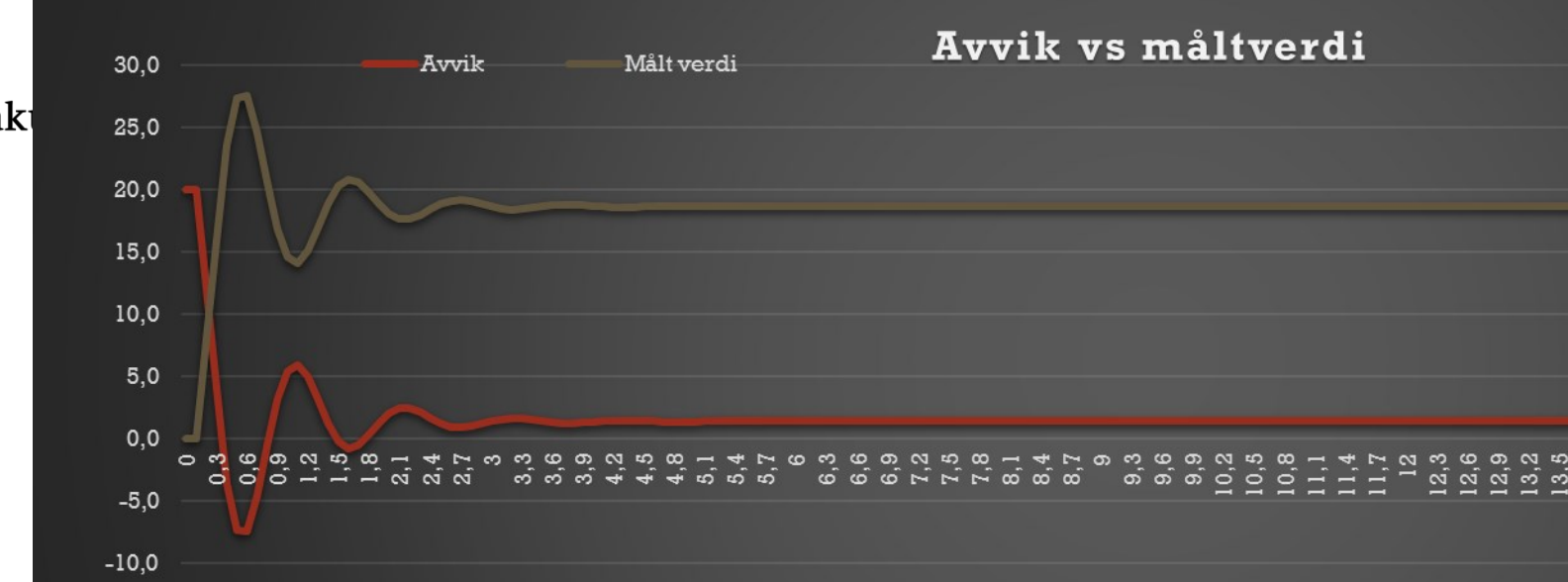
- Med for stor K_p vil systemet oscillere med økende utslag.
  - Vi får oscillasjoner fordi det tar tid før systemet responderer på outputen vi gir.
    - Dess lengre tid før vi ser responsen, dess lavere må K_p være

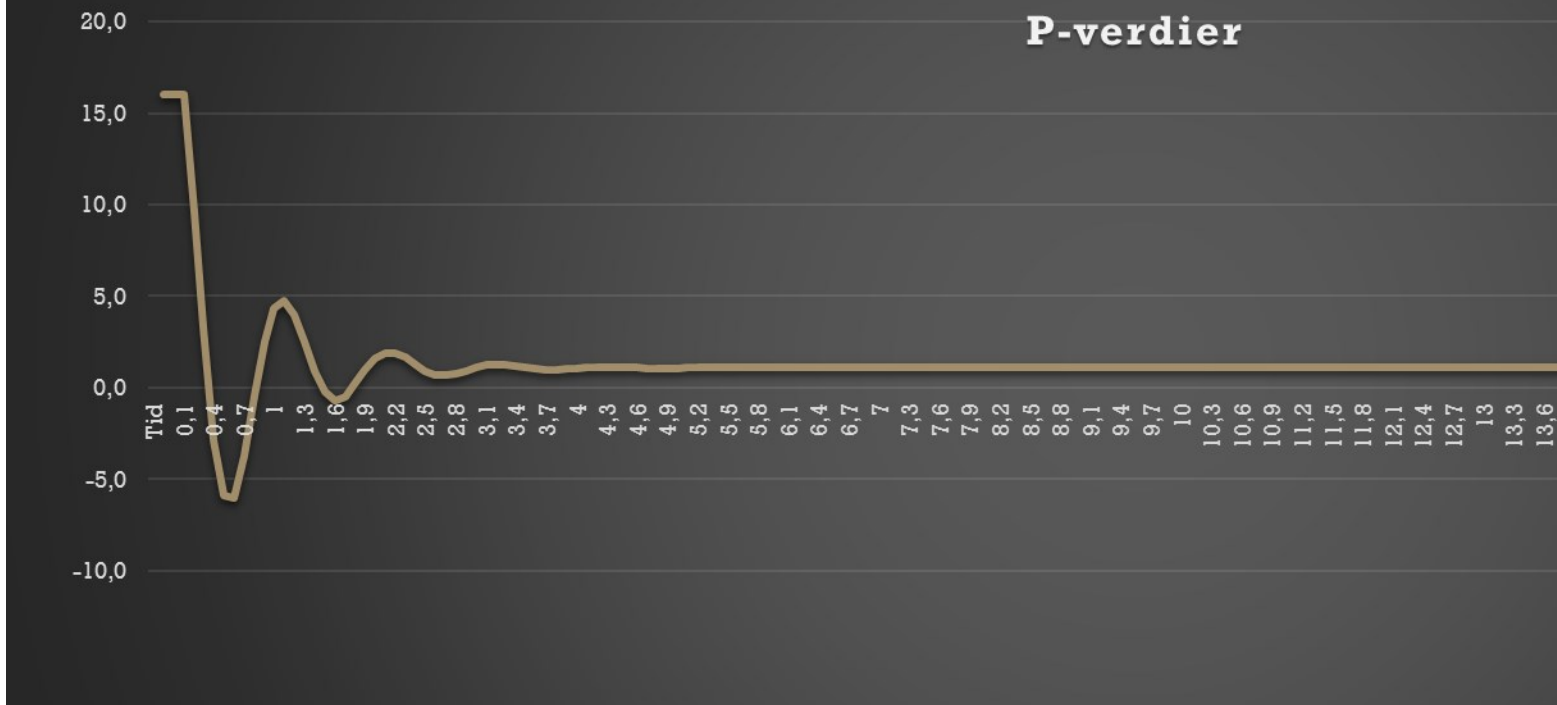


Bare P (forts)

$$K_p \cdot Avvik$$

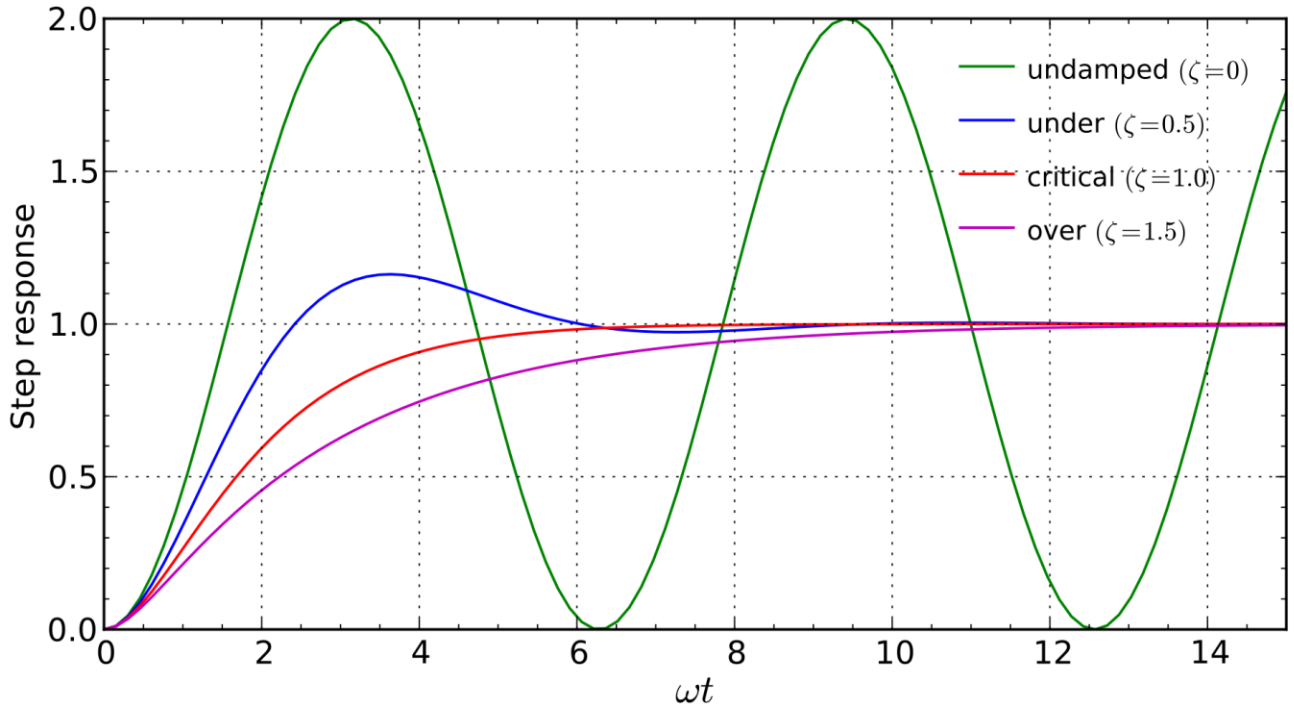
- Med et P-ledd (K_p*Avvik) som ikke er for stort, vil oscillasjonene kunne stabilisere seg.
- P alene stabiliserer seg ikke på målverdien (setpoint).
- Forløpet vi ser her kan beskrives som *underdempet* (under-damped)





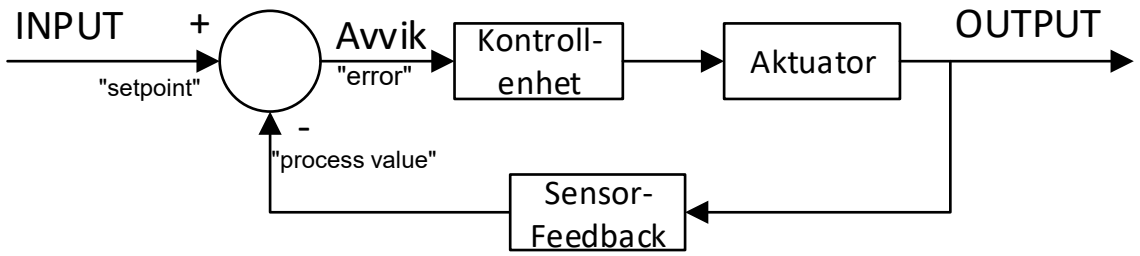
Demping - begreper

- Udempet «*undampet*»
  - Systemet kommer aldri til ro
- Underdempet «*under damped*»
  - Systemet kommer til ro med *ringing*
  - Bruker lengre tid enn kritisk demping
- Kritisk demping «*critical damping*»
  - Systemet kommer til ro (steady state) raskest mulig
    - Uten oscillasjoner (*ringing*)
- Overdempet «*over damped*»
  - Systemet kommer til ro uten ringing
  - Bruker lengre tid en kritisk demping.
-



Kontroll og PID

Generelt kontrollsystem, lukket sløyfe:



- Proporsjonal kontroll alene gir ikke null avvik (kun unntaksvis)
- Kan vi benytte tilstanden til systemet til å redusere avviket?

$$P = K_p \cdot Avvik$$

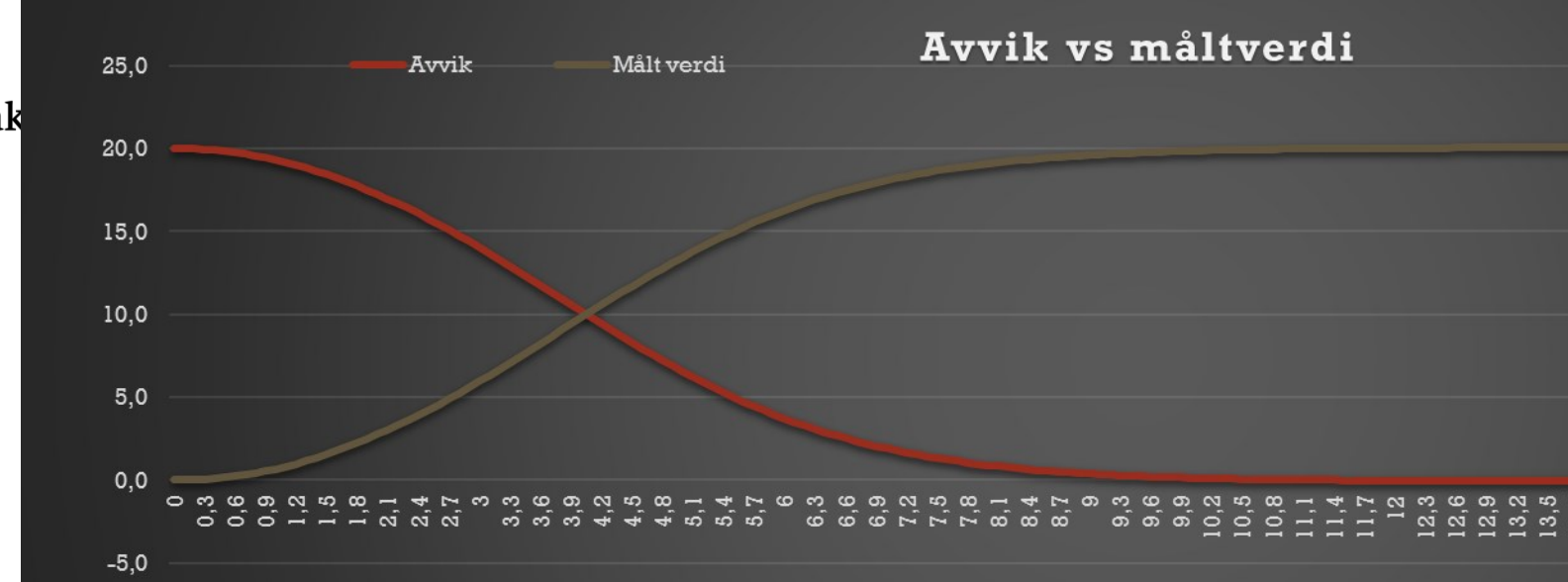
$$I = K_i \cdot \sum_n Avvik = K_i \cdot (\sum_{n-1} Avvik + Avvik_n)$$

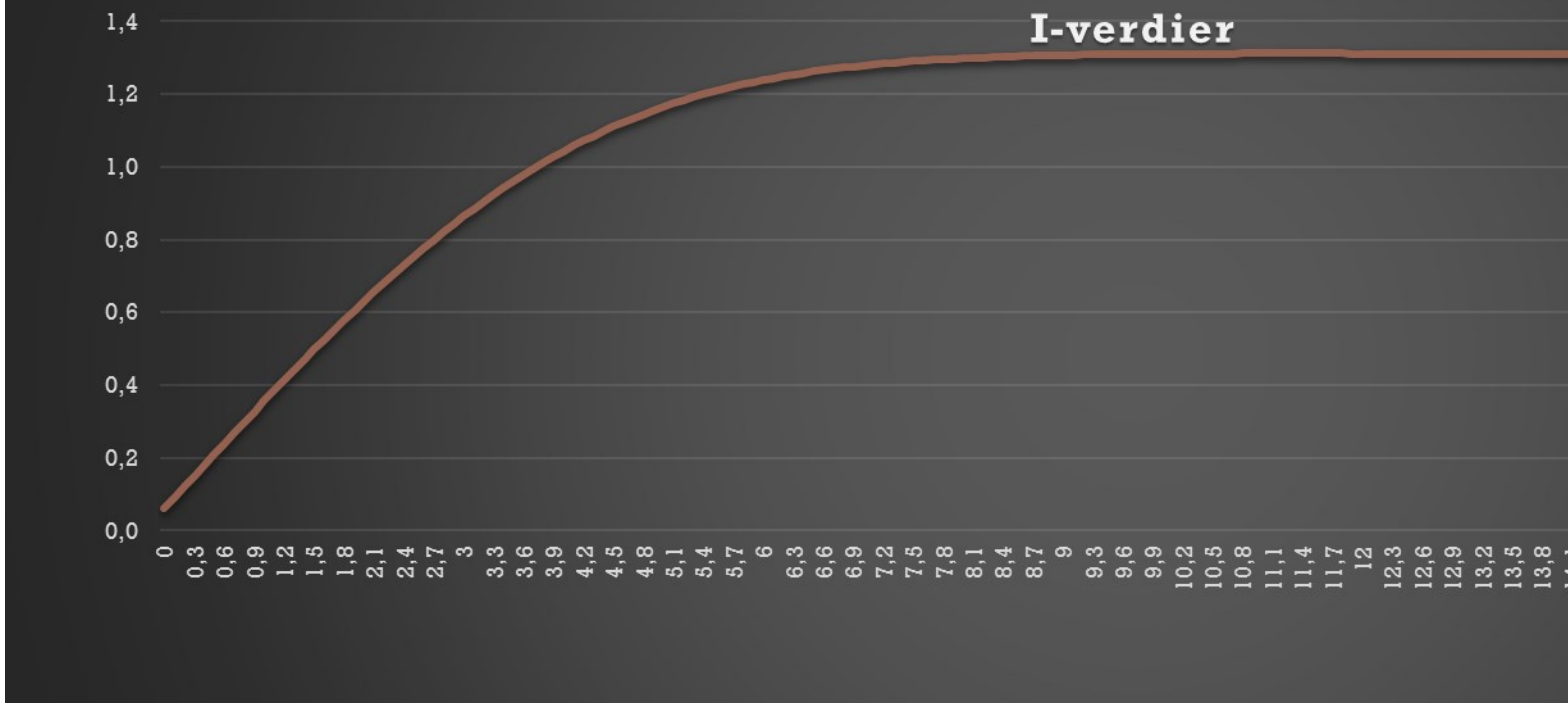
- **Integral kontroll**
  - Bruker det akkumulerte avviket til å beregne signalstyrke
    - *Kan* bringe avviket til null over tid...
      - *Til forskjell fra proporsjonal kontroll*
  - K_i, integralkonstanten
    - bestemmer hvor kraftig det summerte avviket skal virke på systemet

Bare I

$$K_i \cdot (\sum_{n-1} Avvik + Avvik_n)$$

- integral-ledd alene tar oss til målet, men det vil gå langsomt eller…

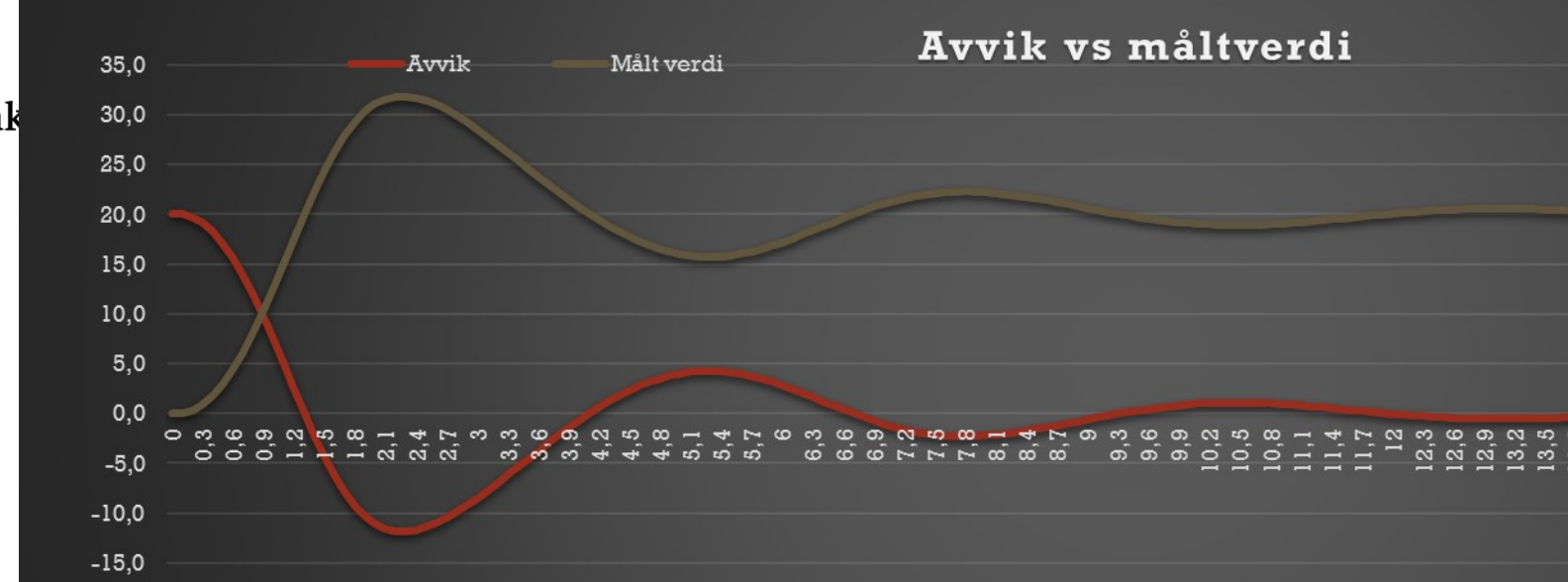


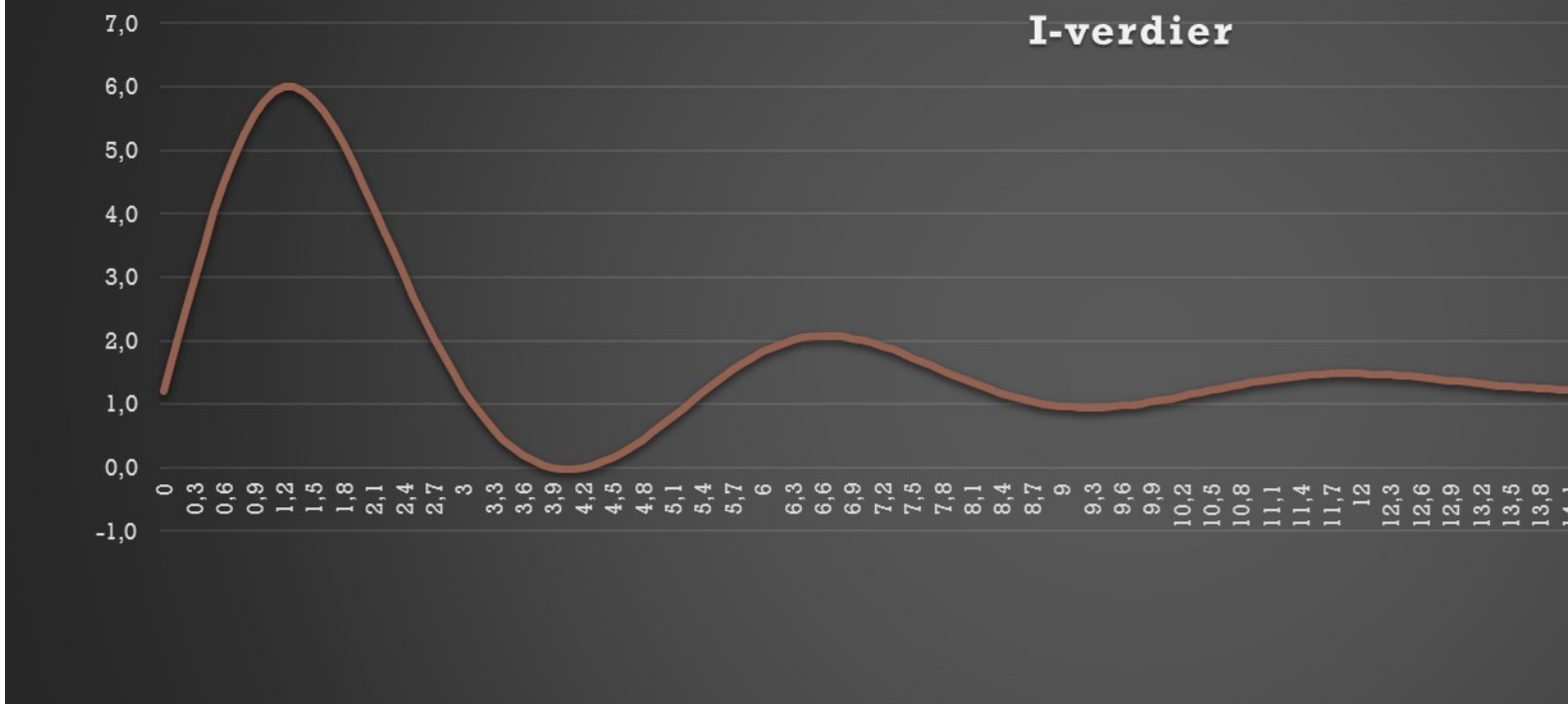


Bare I

$$K_i \cdot (\sum_{n-1} Avvik + Avvik_n)$$

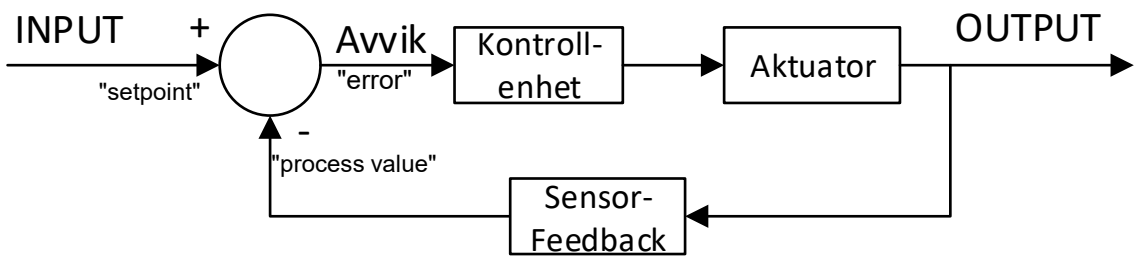
- Vi får oscillasjoner om K_i blir for stor eller systemet er udempet
- Dess lengre responstid, dess mindre må K_i være for å unngå ringing





Kontroll og PID

Generelt kontrollsystem, lukket sløyfe:



- Integral-ledd alene tar oss til målet...
  - men det tar tid
    - I noen tilfeller kan det være bra nok
- Kombinerer vi P og I ledd
  - Kommer vi raskere til målet enn I-ledd alene
  - Om K_p og K_i er for store fører det til ringing...
    - Kan vi gjøre noe for å reduserer ringing / oscillasjoner?
- ***Derivasjonskontroll..?***
  - Bruker vi endringen i avviket som parameter kan vi redusere oscillasjoner
  - K_d, derivasjonskonstanten
    - bestemmer hvor mye vi vil at forskjellen i avvik fra en måling til den neste skal virke inn
  - *Derivasjonsledd benyttes vanligvis ikke alene...*

$$P = K_p \cdot Avvik$$

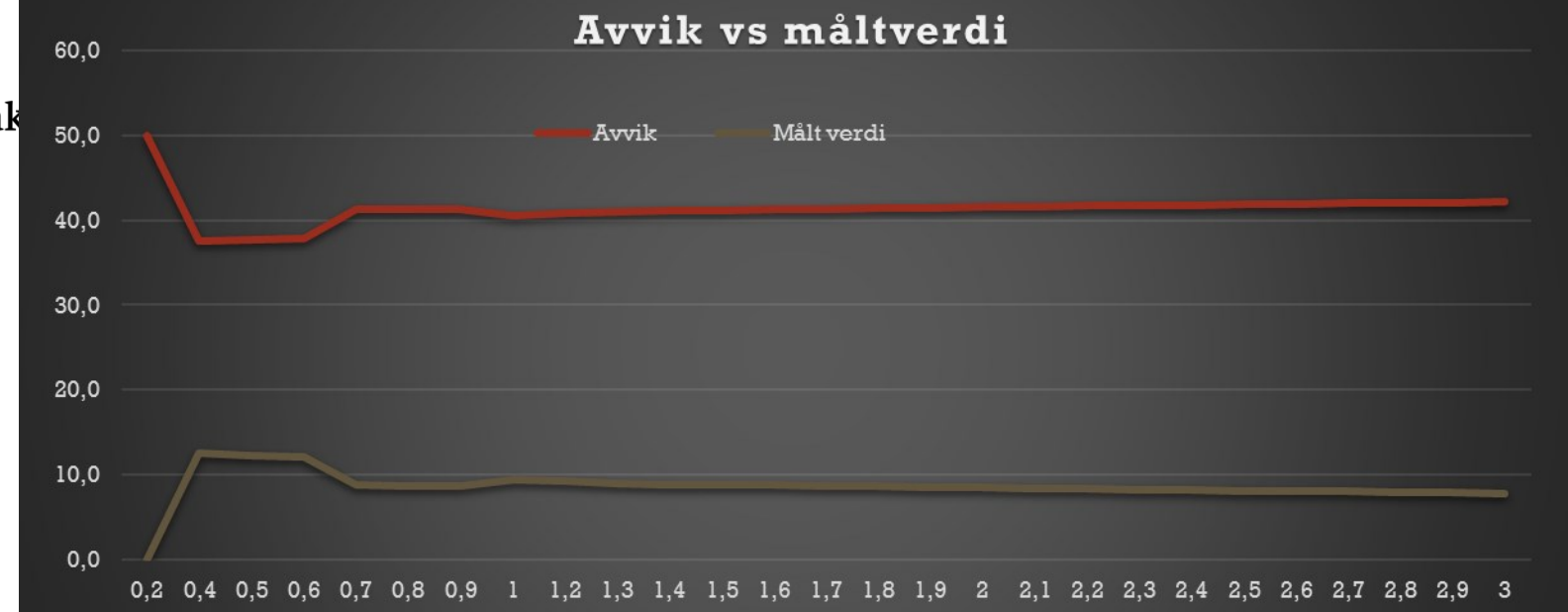
$$I = K_i \cdot \sum_n Avvik = K_i \cdot (\sum_{n-1} Avvik + Avvik_n)$$

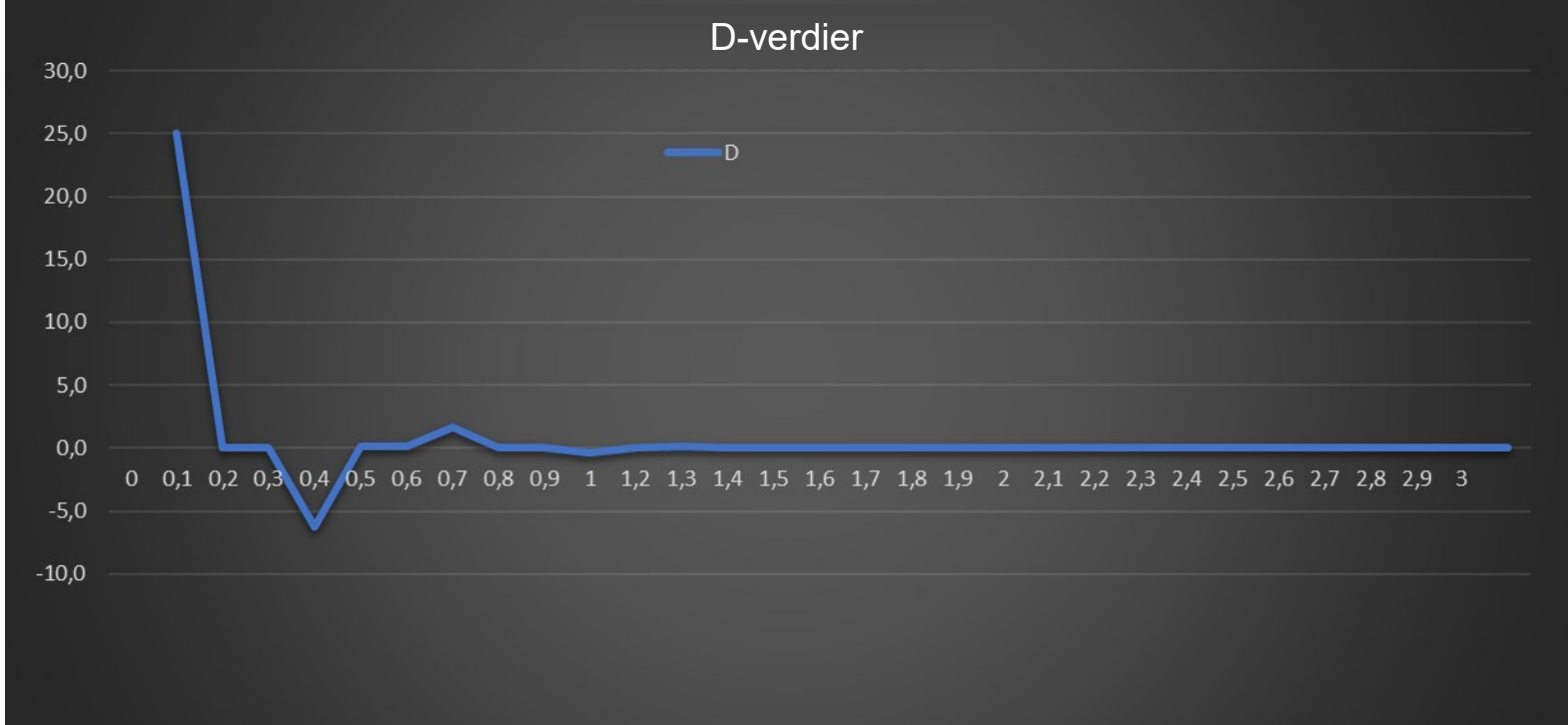
$$D = K_d \cdot \Delta Avvik = K_d(Avvik_n - Avvik_{n-1})$$

Bare D…

$$K_d(Avvik_n - Avvik_{n-1})$$

- D-ledd benyttes sjeldent alene, fordi det bare gjøre en forskjell så lenge avviket endrer seg.
- Så fort endringene går mot null, vil virkningen oppheves.
- For stort D-ledd vil skape oscillasjoner f.eks ved støy i måleverdiene.

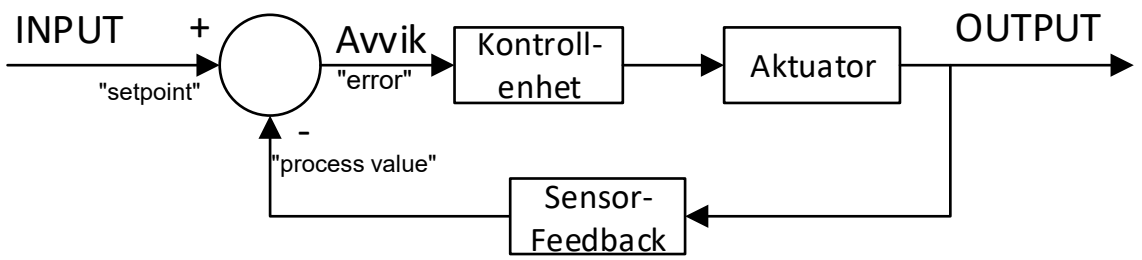




Kontroll og PID

- PID-(*Proporsjonal, Integrert, Derivert*) –regulering
  - beregner output basert på summen av
    - gjeldende avvik,
    - akkumulert (integrert) avvik og
    - forskjellen mellom forrige og gjeldende avvik.
  - kan gjøres både digitalt og analogt.
    - *Vi (IN1080) fokuserer på digitale løsninger*
  - Ved å stille inn forsterkning P, I og D ledd, kan vi nå målet vårt fort, uten unødvendige oscillasjoner.
    - For store konstanter (alle ledd) vil skape oscillasjoner
    - Konstantleddene må stilles inn «*tuning*» i forhold til
      - Kontrollsystemets forsinkelse (prosesseringstid + måletid)
        - » *Den bør være konstant...*
      - Systemets respons
        - » Et systems responstid kan avhenge av last
          - » Eks: bil med og uten tilhenger...
      - Tuning gjøres ofte manuelt, men også med automatisert vhja maskinlæring/AI el.
  - favner ikke alt, mer avanserte kontrollalgoritmer finnes
    - men PID brukes mye, *i seg selv og innenfor mer avanserte algoritmer*

Generelt kontrollsystem, lukket sløyfe:



P: $K_p \cdot Avvik$

I: $K_i \cdot \sum_n Avvik = K_i \cdot (\sum_{n-1} Avvik + Avvik_n)$

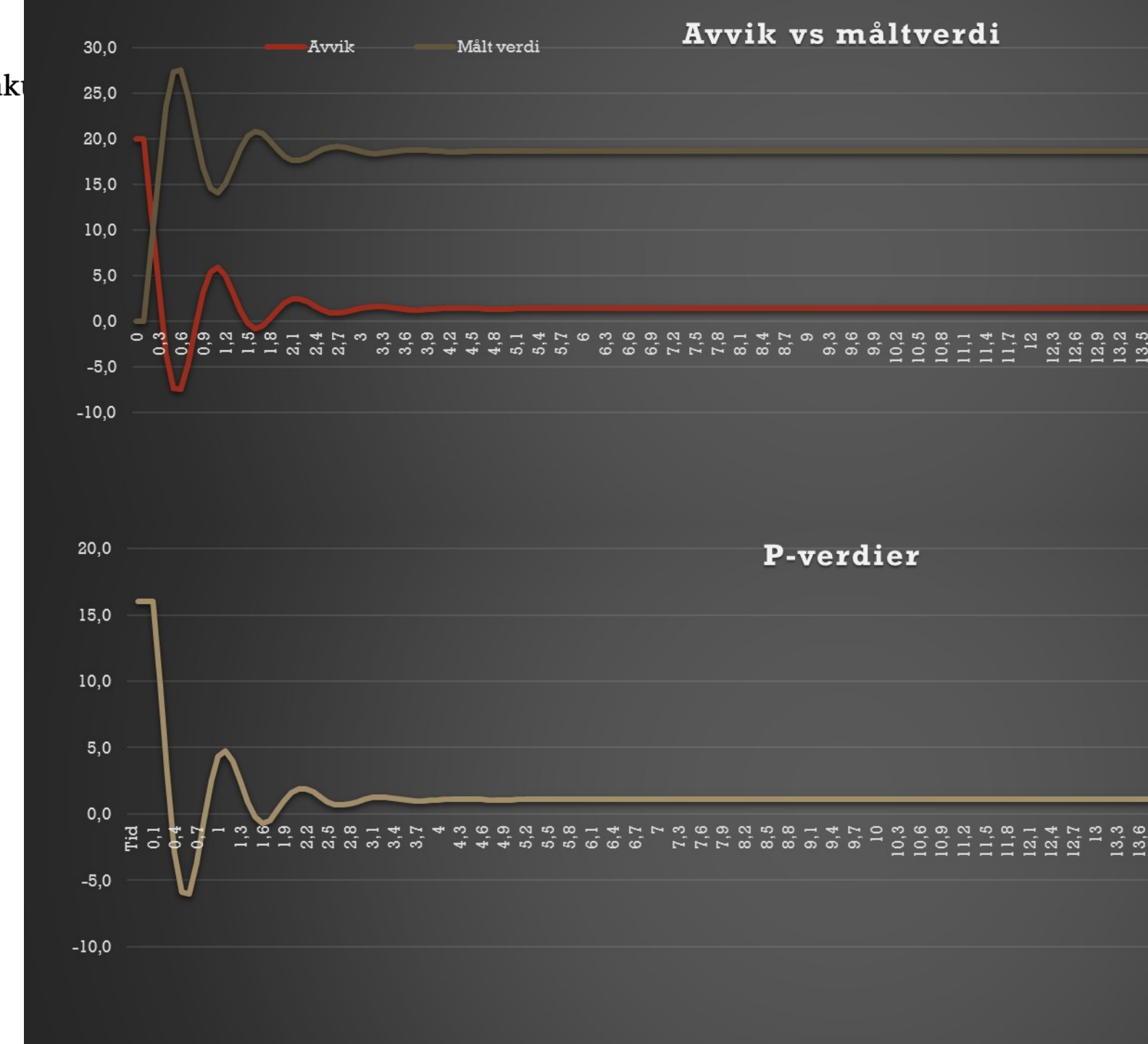
D: $K_d \cdot \Delta Avvik = K_d(Avvik_n - Avvik_{n-1})$

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$$

- Eksempel med PID-kontroll (2 min) https://www.youtube.com/watch?v=K-F_T59ZDPw
- PID-math demystified (14,5 min) https://www.youtube.com/watch?v=JEpWlTl95Tw
- Analog PID kontroll med operasjonsforsterkere (7 min) https://www.youtube.com/watch?v=YLGLrEwEiTQ

Bare Stor P (igjen)

- Stor *overshoot,* og
- kan ikke nå målet



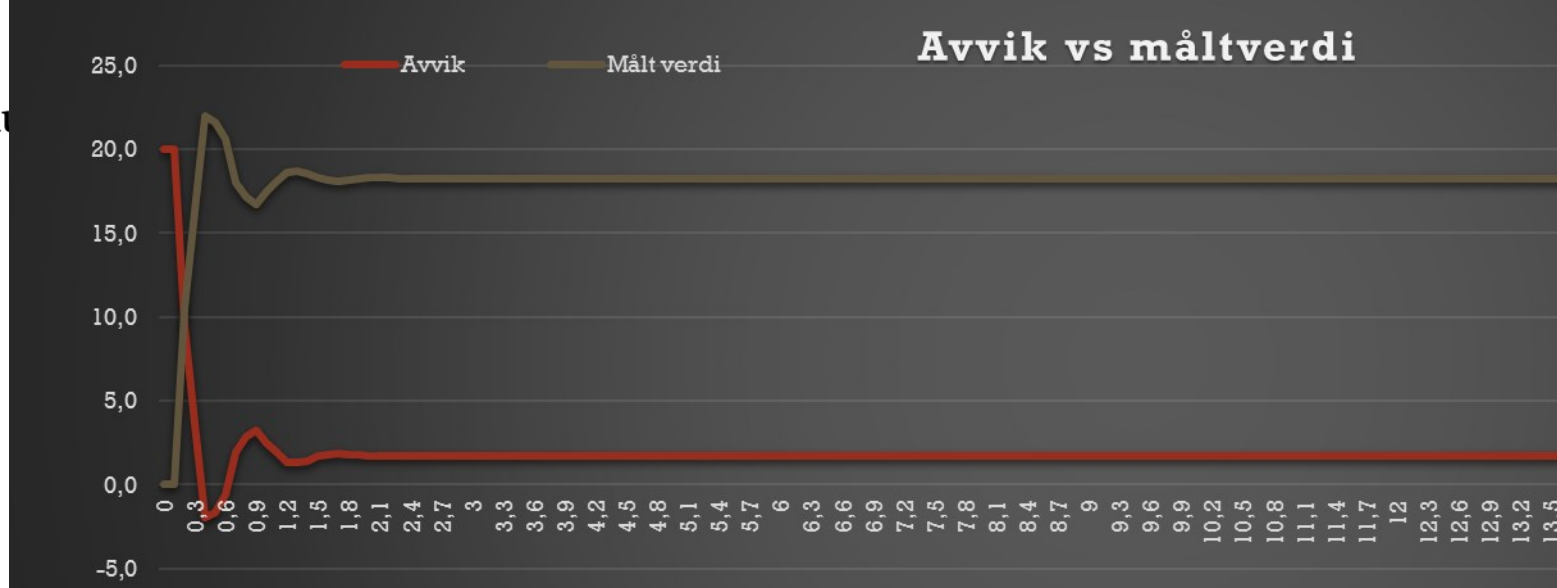
P og D-ledd:
P+D

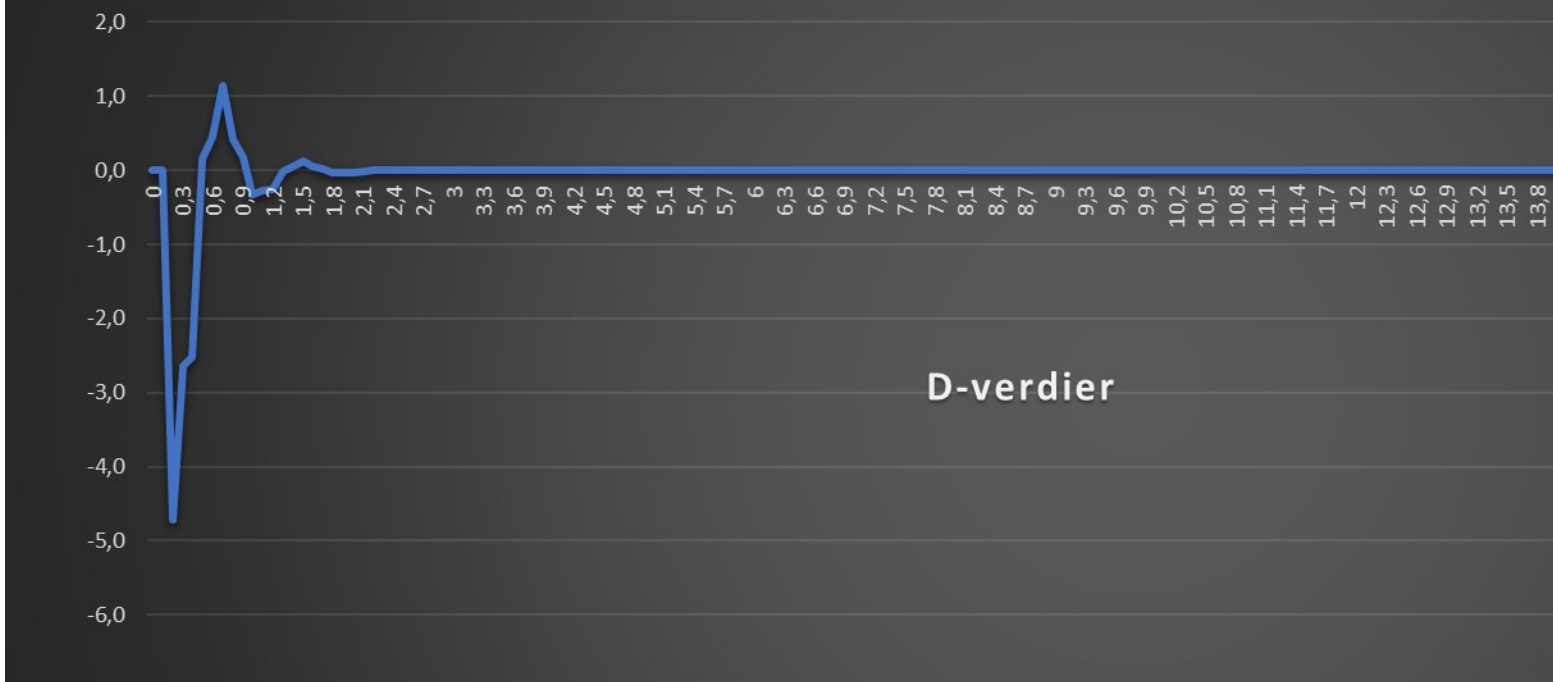
$$K_p \cdot Avvik + K_d(Avvik_n - Avvik_{n-1})$$

- D-leddet
  - kompenserer med forskjellen i mellom forrige og siste avvik.
  - har normalt en dempende effekt ved raske endringer.

- P+D-ledd
  - vil ikke stabilisere seg på ønsket verdi ettersom D leddet ikke legger noe til over lang tid.

- *D-ledd fungerer også dårlig om man har mye støy i systemet.*
  - *Men støy kan filtreres…*

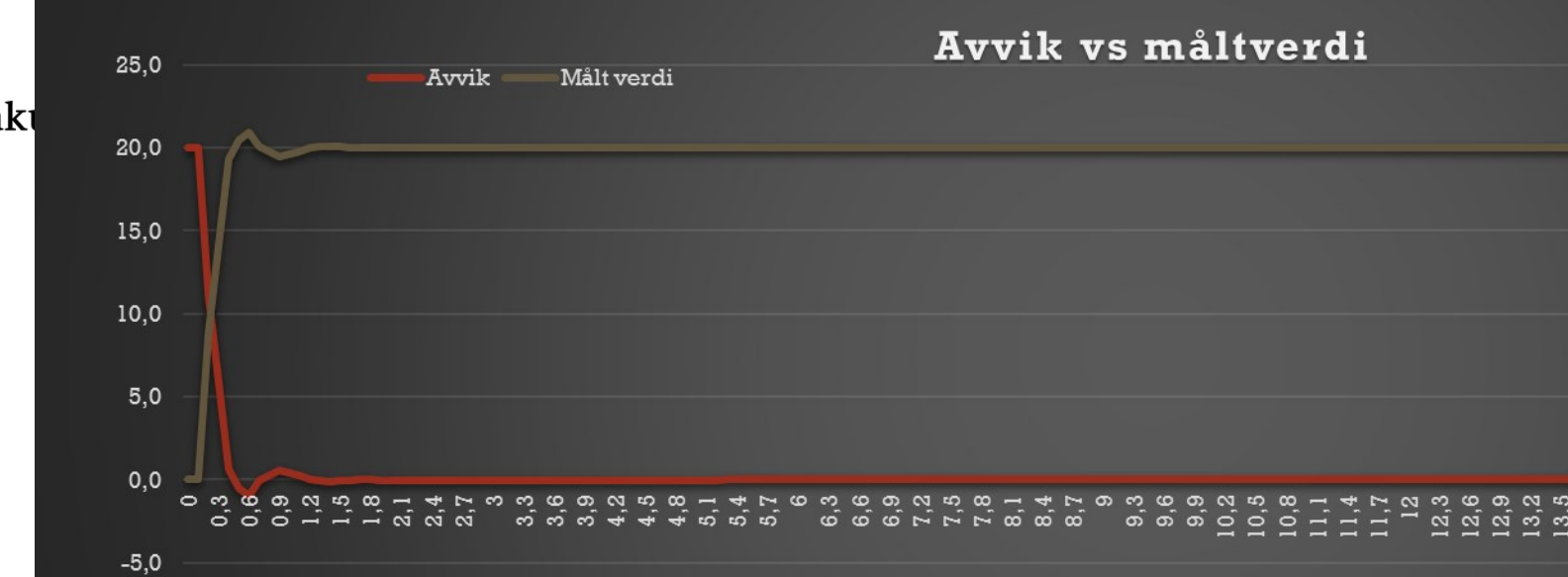


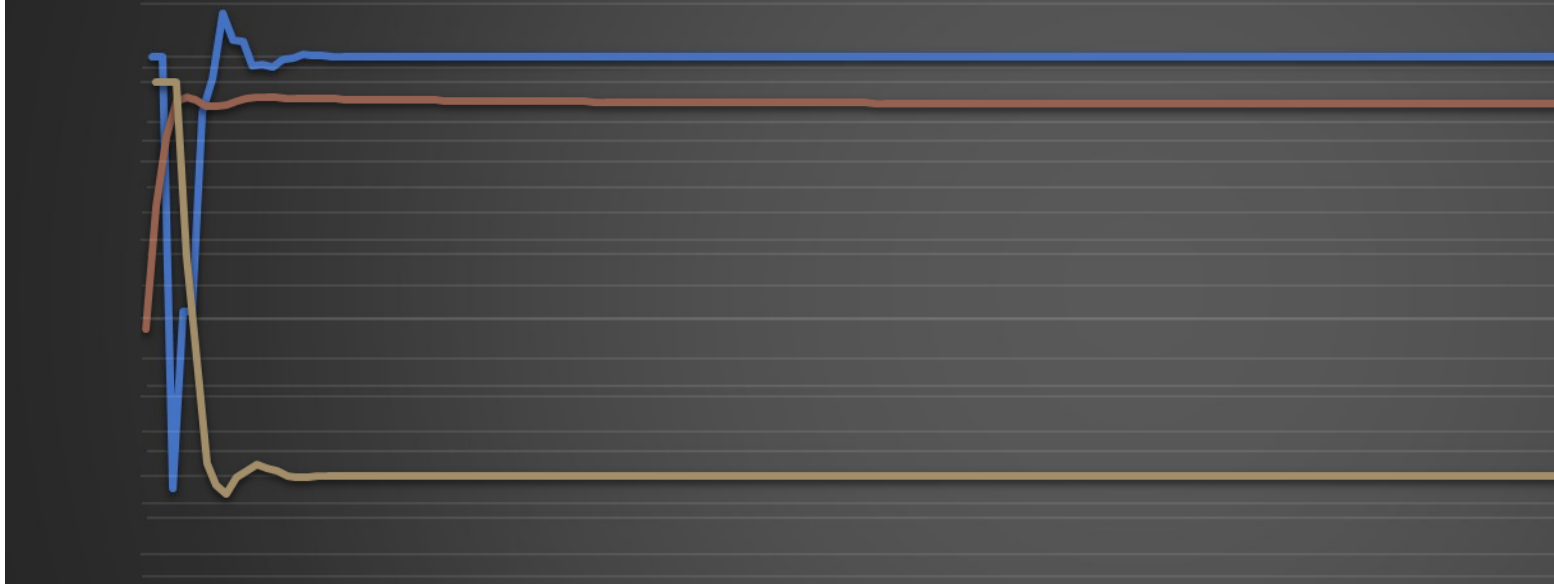


PID

$$K_p \cdot \text{Avvik} + K_i \cdot (\sum_{n-1} \text{Avvik} + \text{Avvik}_n) + K_d(\text{Avvik}_n - \text{Avvik}_{n-1})$$

- Med en riktig justert kombinasjon av konstanter vil en PID kontroller nå målet raskt og presist.
- PID-systemet kan justeres slik at man får en akseptabel *overshoot*, og lite *ringing*.
- Manuell tuning av PID kan ta tid!





PID: Proporsjonal, Integrert, Derivert Programmering

For å lage PID kontroll, må man gjøre følgende

Beregne avviket («error») mellom målverdi «setpoint» og målt (avlest) verdi.

$$avvik \quad = målverdi - avlest\_verdi$$

Beregne akkumulert avvik

$$akkumulert\_avvik \quad = akkumulert\_avvik_{n-1} + avvik$$

Beregne Proporsjonsledd:

$$P = K_p \cdot avvik$$

Beregne Integrasjonsledd:

$$\mathrm{I} = K_i \cdot akkumulert\_avvik$$

Beregne Derivasjonsledd:

$$\mathrm{D} = K_d(avvik - avvik_{n-1})$$

Beregne $\mathrm{output} = \mathrm{P} + \mathrm{I} + \mathrm{D}$

Størrelsen på konstantleddene må skaleres avhengige av hvor ofte PID-verdiene beregnes.

⇒ *Kjør rutinen med jevne mellomrom*

- ⇒ interrupt-styring eller jevnlig polling...
- ⇒ Alternativt kan man dele på tid siden forrige måling...

Arduino–PID-template:

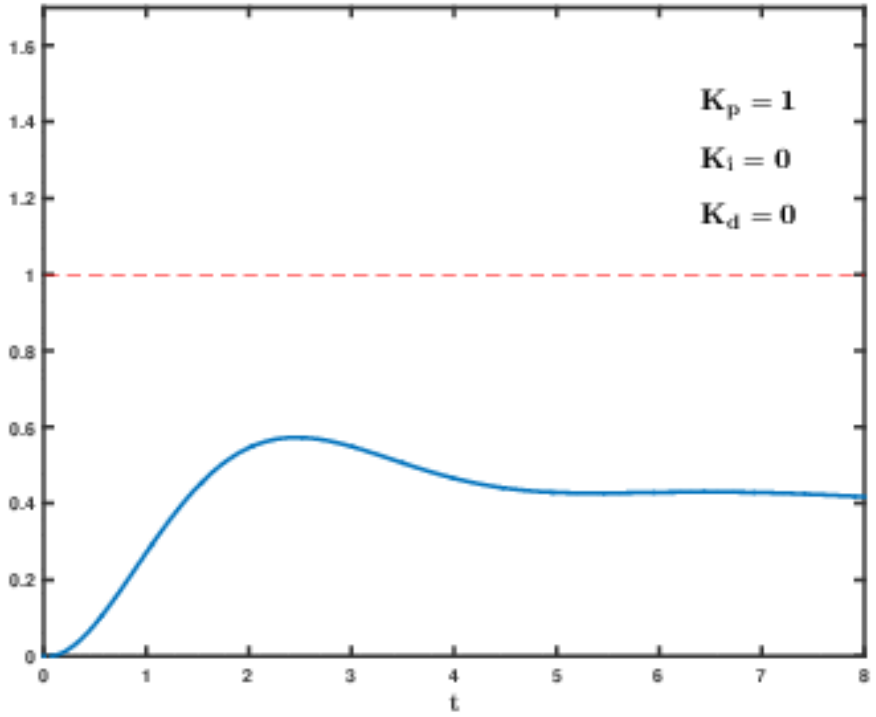
```
int Kp = <P-konstant>; // disse må ha en
int Ki = <I-konstant>; // konstant verdi
int Kd = <D-konstant>; // etter tuning

int myPID(int setpoint, int measured_value){

  static int previous_error, accumulated_error;
  int error, d_error, P, I, D;

  error = setpoint - measured_value;
  accumulated_error += error;
  d_error = error - previous_error;

  P = Kp*error;
  I = Ki*accumulated_error;
  D = Kd*d_error;

  previous_error = error; // lagre før neste kjøring

  return P+I+D;
}
```

Merk: Denne templaten kan bare brukes til én PID-kontroller siden den bruker static-int.
For å bruke samme prosedyre til flere PID-kontrollere må man ha inngangsparametere for akkumulerte- og tidligere avvik.

Tuning av PID, Eksempel

- I virkeligheten kan utprøving av parametere ta lang tid.
- Kjenner vi ligningen for systemet, kan vi simulere, og raskt stille inn optimale parametre.

- Vanlige metoder for tuning
  - Prøv og feil: *Trial and Error*
  - Ziegler-Nichols (Z-N) method
    - Benytter måling av step responsen til systemet til å sette utgangspunkt for parametre
    - Fungerer med førsteordens system...
    - Responstiden kan ikke være for stor



Kilde: Wikipedia https://en.wikipedia.org/wiki/PID_controller

PID DEMO (Prøv selv):
https://sites.google.com/site/fpgaandco/pid

Manuell tuning

Hvis vi øker...

Parameter (Konstant)	Stigetid (Rise time)	Oversving (Overshoot)	Stabiliseringstid (Settling time)	Stabilitetsavvik (Steady-State Error)
K_p	reduserer	øker	øker	reduserer [1]
K_i	reduserer	øker	reduserer, øker [2]	"eliminerer" [3]
K_d [4]	øker	reduserer	reduserer	uendret

1. Kp kan ikke eliminere avvik, men gjøre det mindre
2. Ki vil redusere stabiliseringstiden frem til man begynner å få overshoot, deretter øker den
3. Bruk av Ki vil eliminere avviket ved stabilitet, så lenge vi oppnår stabilitet. Med høyere Ki, skjer det raskere
4. Kd kompenserer for overshoot og ringing som forårsakes av Kp og Ki, slik at vi raskere kan oppnå kritisk demping.

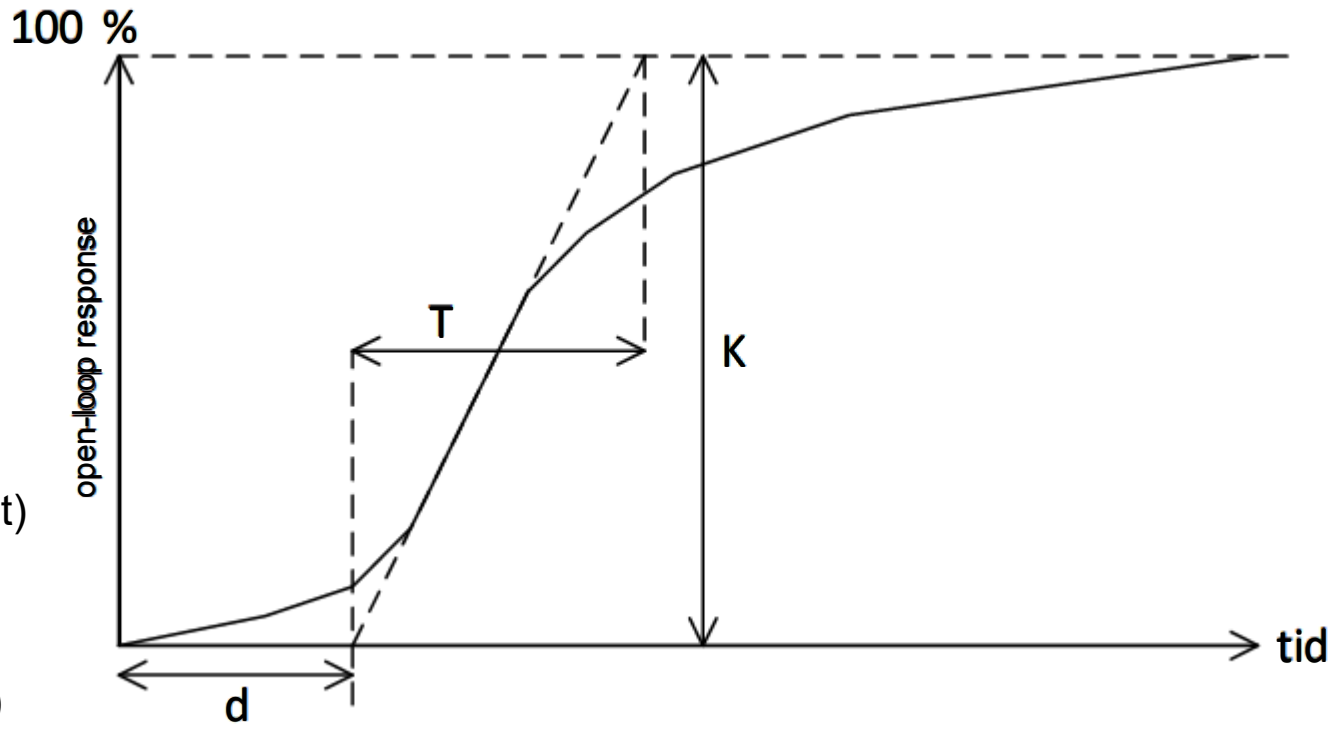
- *Merk: I Læreboka (COK) figur 28.1 og 28.2 (= denne) så sier læreboka at «innføring av D-ledd ikke øker stigetiden»,* (men økning i K_d *gir* økt stigetid). Dette *kan* være fordi :
  - Vi *vil kunne* øke K_p og/eller K_i ved innføring av K_d
    - Slik sett vil systemet kunne respondere raskere med ett D-ledd, *selv om D-ledded i seg selv sakker ned stigetid dess høyere K_d er*

Ziegler-Nichols metode

(heurestikk)

- Bruker step respons til å finne parametre (test eller beregn):

1. Finn «Process Gain» ved open loop,
   $G_p = \frac{Endring\ i\ output\ (K)}{endring\ i\ input}$
2. $K_p = 1{,}2\frac{T}{dG_p}$ (T er stigetid for tangenten)
3. $K_i = \frac{0{,}5}{d}$ (d er «pseudo-delay» før vi når tangent)
4. $K_d = 0{,}5d$



- To forutsetninger *(for at metoden skal være god...)*
  1. førsteordens system
     (= samme Gp uansett inputverdi)
  2. kontrollsløyfen svarer raskt (d < T/7)

Eks: Vi endrer input fra 2 til 4 og ouput går fra 1 til 2,
Pseudo-delay d er 0,1s, T er 1s =>
Endring i output = 2-1 = 1
endring i input = 4-2 = 2
Gp = 1/2

$$K_p = 1{,}2\frac{1}{0{,}1 * 0{,}5} = 24$$

$$K_i = \frac{0{,}5}{0{,}1} = 5$$

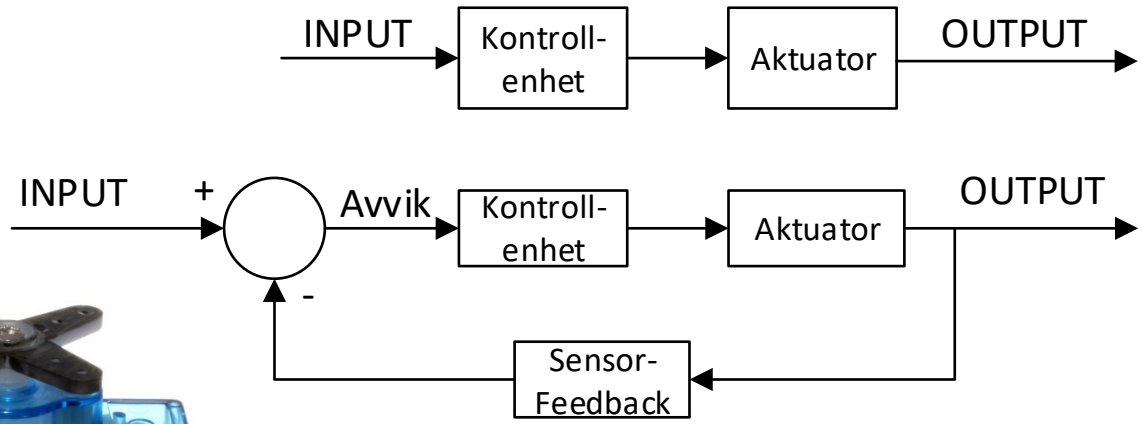
$$K_d = 0{,}5 * 0{,}1 = 0{,}05$$

NB: Ikke sikkert at denne metoden gir beste resultat Z-N beskrives i flere varianter

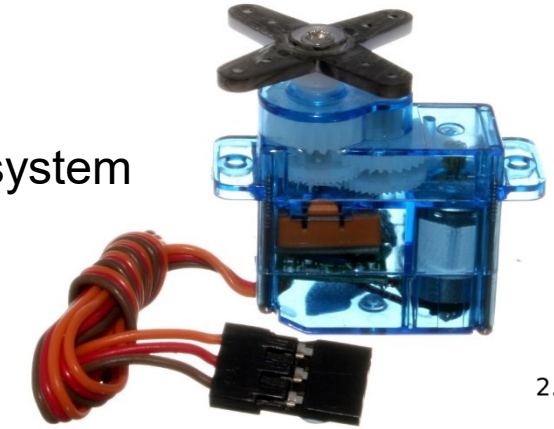
(Noen krever at man setter/beregner frekvens for oscilleringer, andre benytter step-respons).

Oppsummering

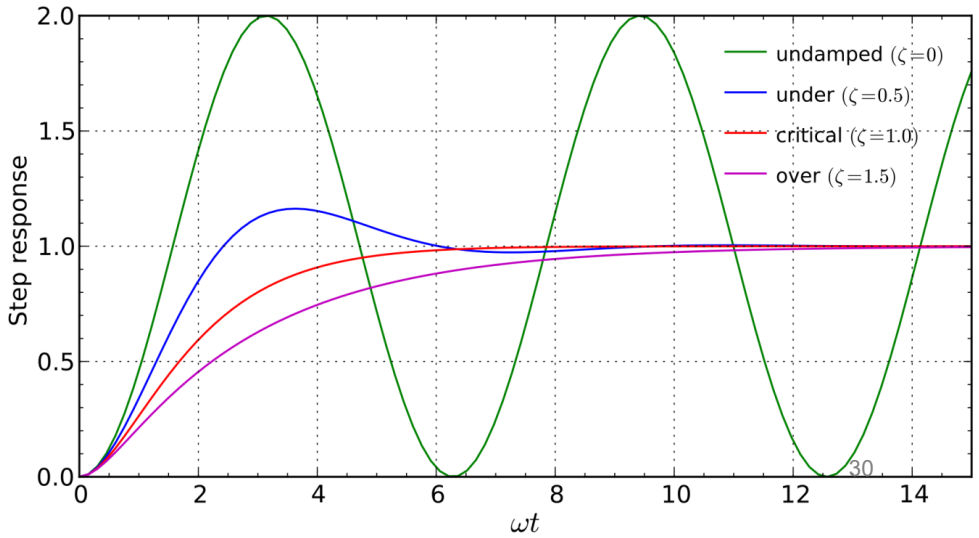
- Kontrollteori
  - Åpen sløyfe
  - Lukket sløyfe



- Motorer
  - Servo = aktuator med lukket kontrollsystem



- Kontrollsystemer
  - Bang-Bang
  - PID
    - Tuning: Manuelt eller Ziegler Nichols *(ifht step respons).*



- Begreper
  - Stegrespons (step response)
  - U-, under-, kritisk- og over- dempet

Anbefalt lesing og oppgaver

- Lese
  - COK: 28.1-28.8

- Oppgaver
  - COK: 28.1, 2, 3, 4, 5, 10, 11

  - *Merk for 28.1 fasit er ikke dekkende for alle muligheter...*