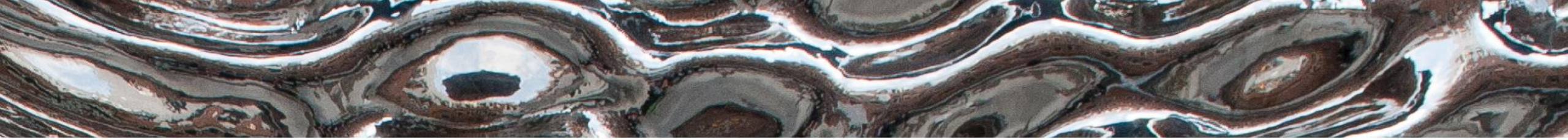


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN 1080

Kontrollsystemer og PID

Yngve Hafting, 2020

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**

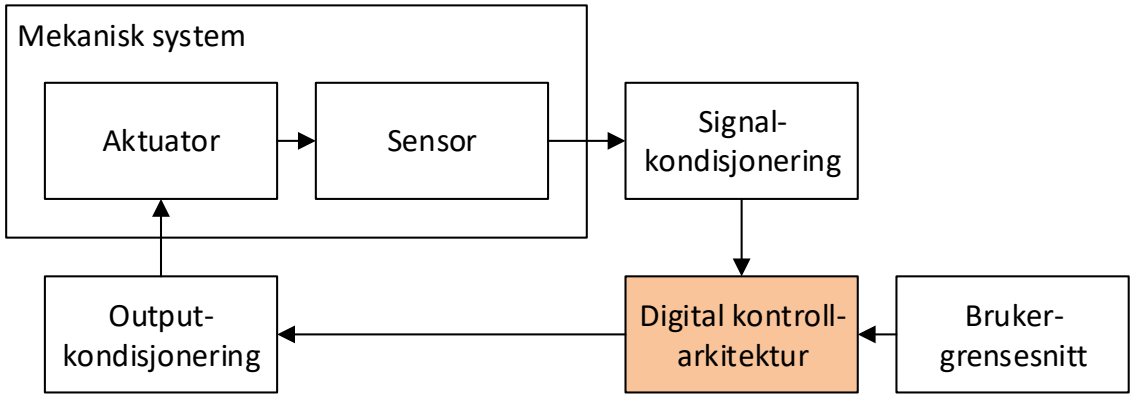
Lab

- Ikke lab pga koronavirus => jobb med oppgaver!

Forelesning

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

Systemperspektiv og oversikt



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

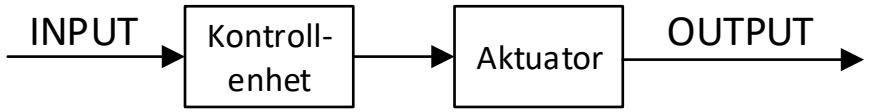
Hvorfor trenger vi et kontrollsystem?

- 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.

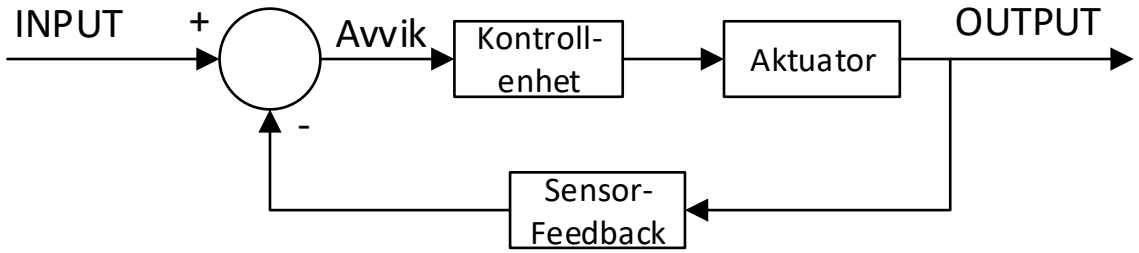
Kilde: Zooop, la voiture électrique de demain? (2006)

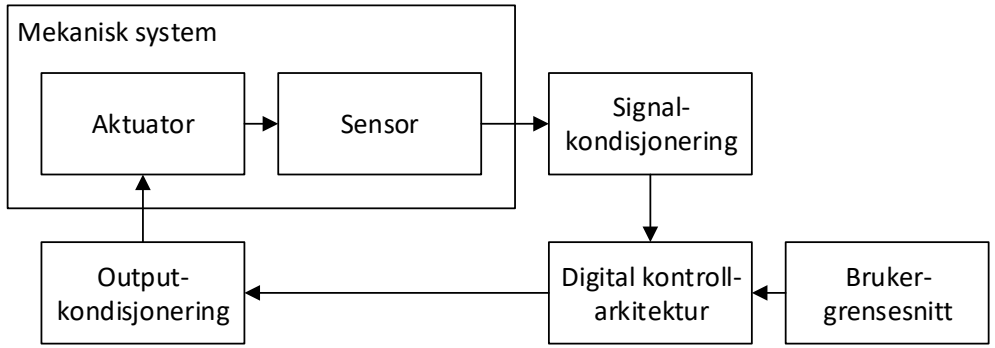
Åpne og lukkede kontrollsløyfer

- Åpen sløyfe (open loop)
  - Ingen sensorfeedback
  - Vi ser det ofte i systemer...
    - med kun mekaniske komponenter
      - Gass/ brems på eldre biler
    - med steppermotorer
      - skrivere/printere
    - Der mennesker står for input
      - volumkontroll på stereoanlegg



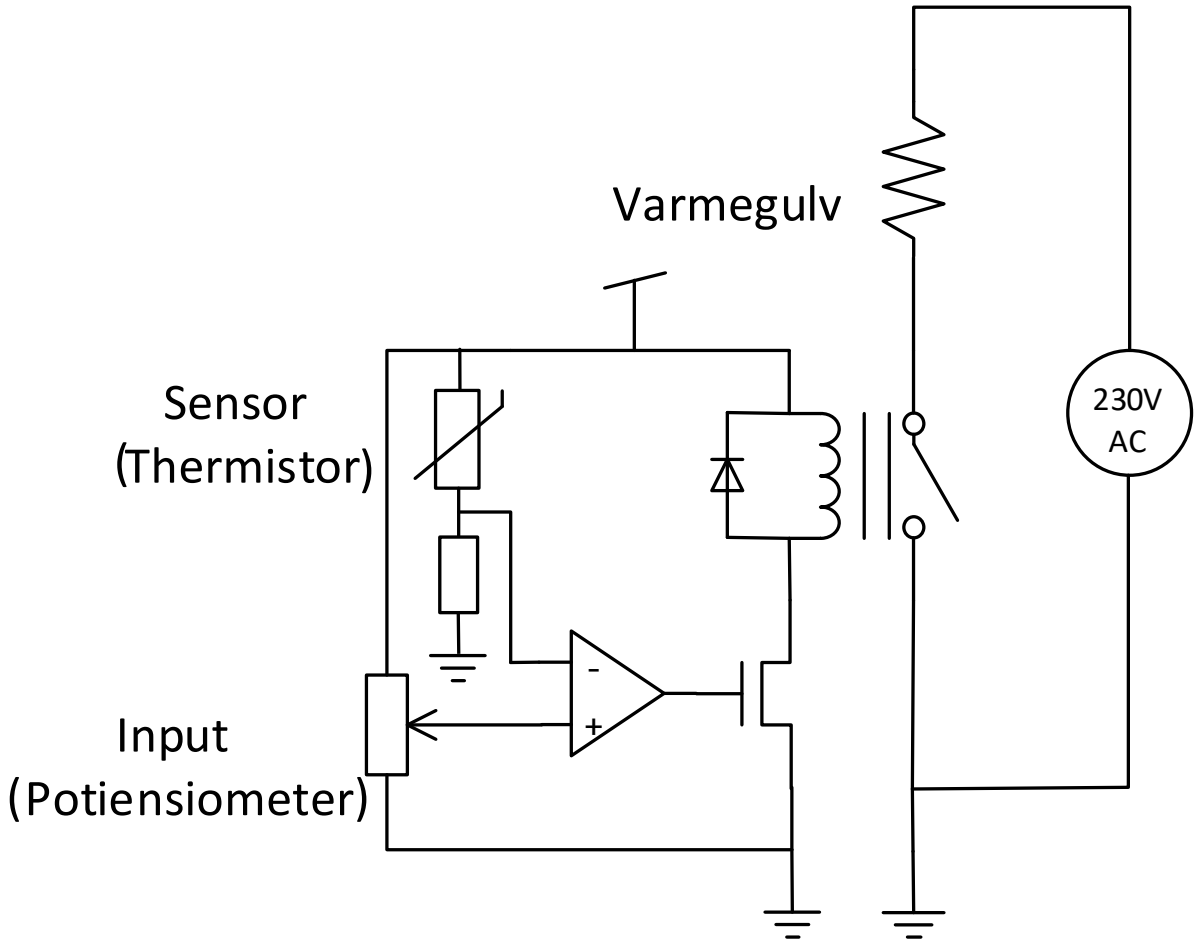
- Lukket sløyfe (closed loop)
  - Brukes der man trenger sensorfeedback
  - Intelligente robotikksystemer generelt
    - Servomotorer
    - ABS-brems
    - traction control
    - cruise control
    - selvbalanserende kjøretøy
    - Temperaturregulatorer
    - Vaskemaskiner
    - ...





Bang-bang kontroll

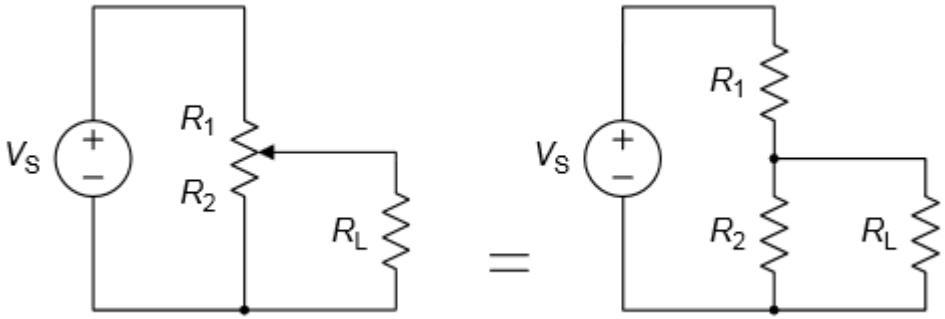
- Binær kontroll
  - (System enten av eller på)
- Eks:
  - Varmegulv styres med et relé

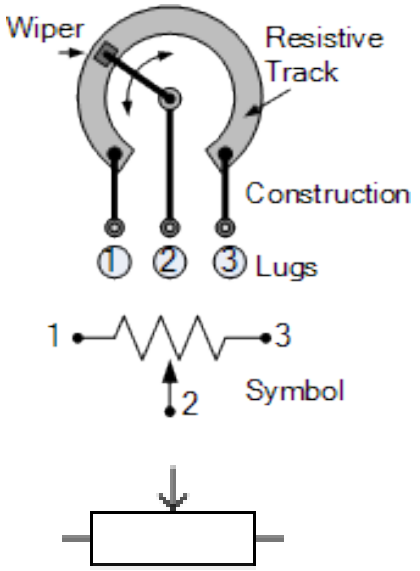


https://en.wikipedia.org/wiki/Potentiometer

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

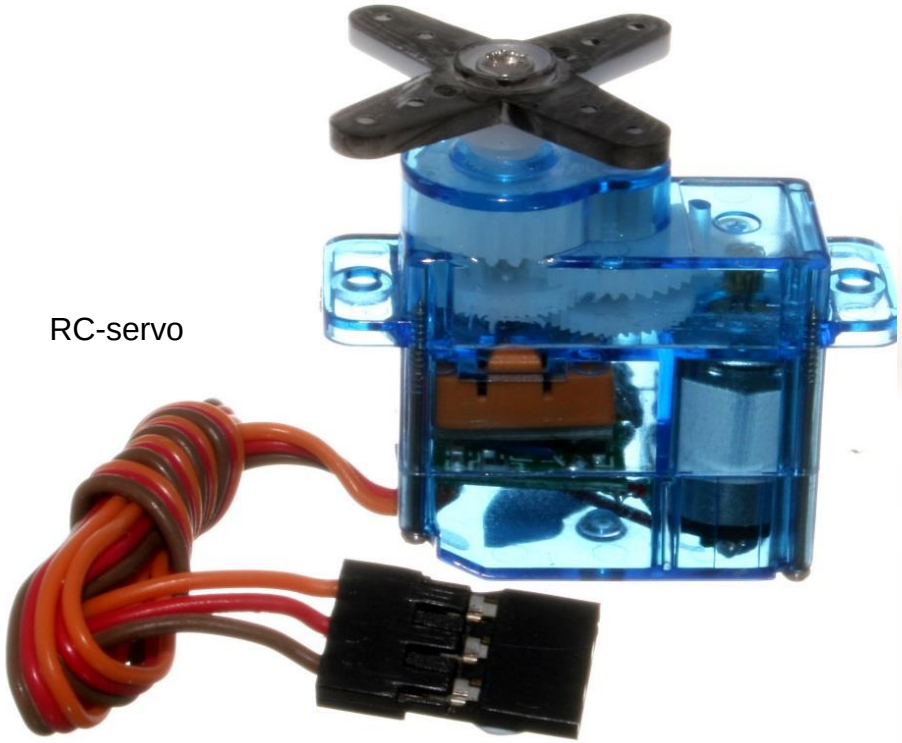
- En servo er i prinsippet en aktuator koblet sammen med et kontrollsystem i en lukket sløyfe (closed loop).

- 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.

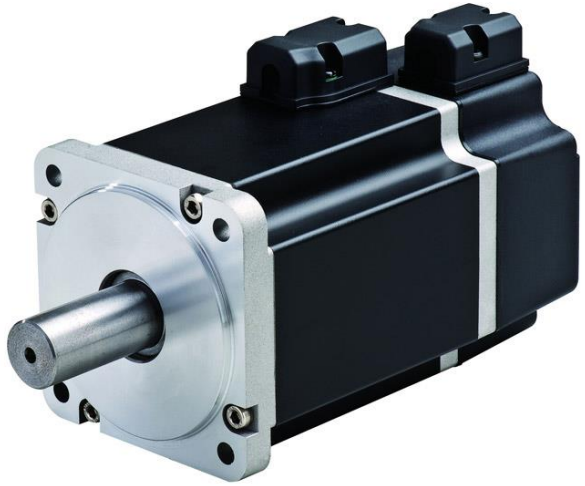
- Servo med potentiometer (7 min):

- *Servo koblet til arduino (10 min): https://www.youtube.com/watch?v=LXURLvga8bQ*

RC-servo



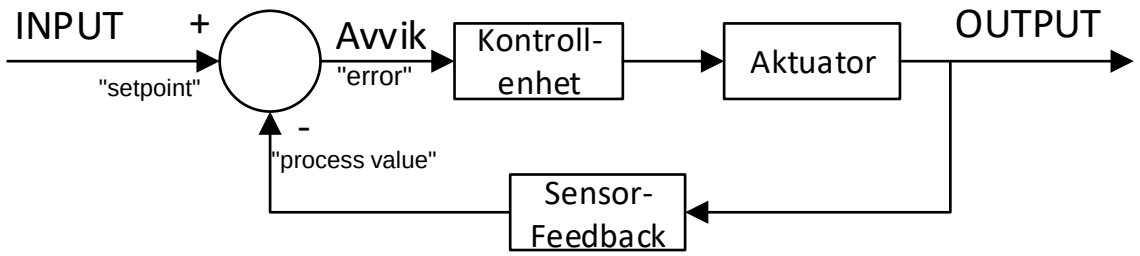
Dynamixel Servo



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, eller feil/«error», avgjør hvilke signaler vi sender til aktuatoren.
- PID-(*Proporsjonal, Integrert, Derivert*) –regulator gir god kontroll
  - PID regulatorer beregner output basert på summen av
    - gjeldende avvik,
    - akkumulert (integrert) avvik og
    - forskjellen mellom forrige og gjeldende avvik.
  - Ved å stille inn forsterkning P, I og D ledd, kan vi nå målet vårt fort, uten unødvendige oscillasjoner.
  - PID regulering kan gjøres både digitalt og analogt.
- I mer spesialiserte systemer kan man benytte (avanserte) matematiske modeller for å forutsi og styre bevegelse (ikke PID).
  - Man kan benytte maskinlæring til både å finne matematiske modeller basert på historikk, eller bare til å justere konstantene i en PID-sløyfe.

P: $K_p \cdot Avvik$

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

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

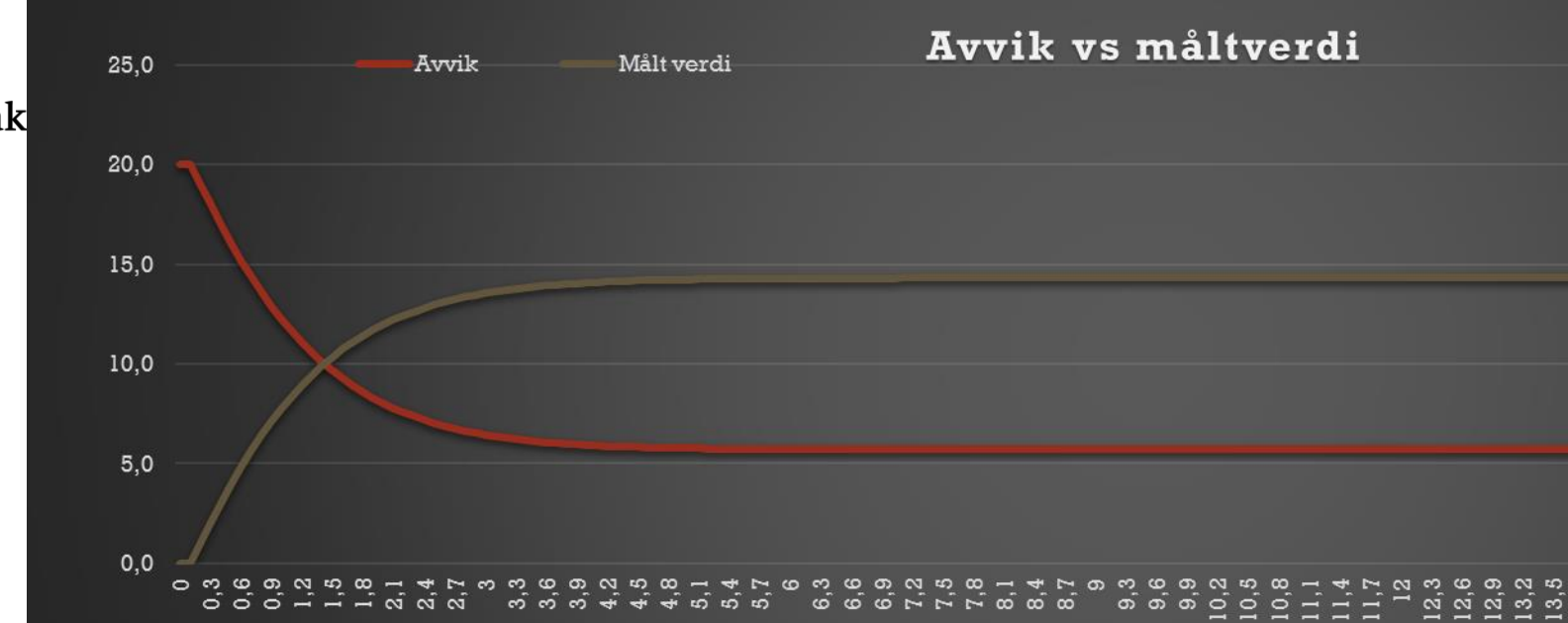
$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$$

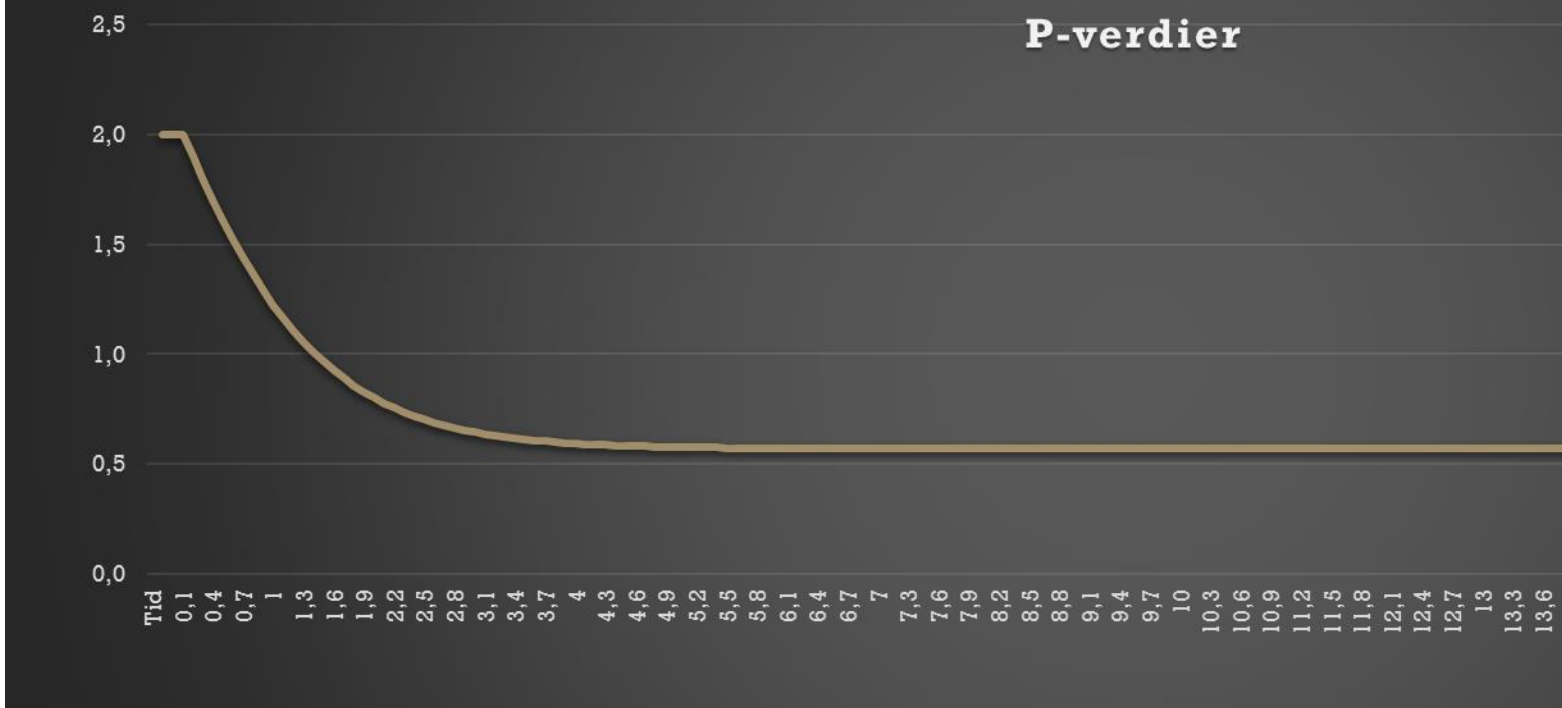
- Eksempel med PID-kontroll (2 min) https://www.youtube.com/watch?v=K-F_T59ZDPw
- PID-math demystified (14,5 min) https://www.youtube.com/watch?v=JEpWlTl95Tw
- Analog PID kontroll med operasjonsforsterkere (7 min) https://www.youtube.com/watch?v=YLGLrEwEiTQ

Bare 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)



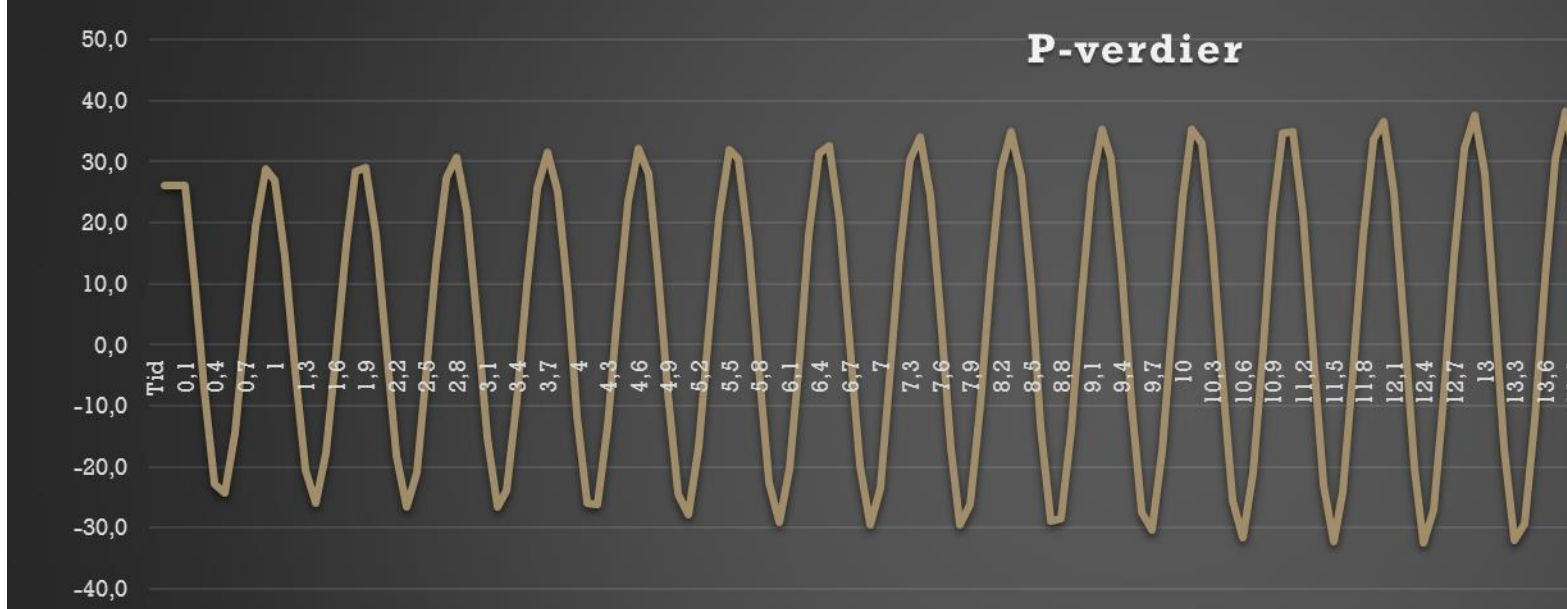


Bare P (forts)

$$K_p \cdot Avvik$$

- Dersom Kp er stor nok, vil systemet oscillere med økende utslag.

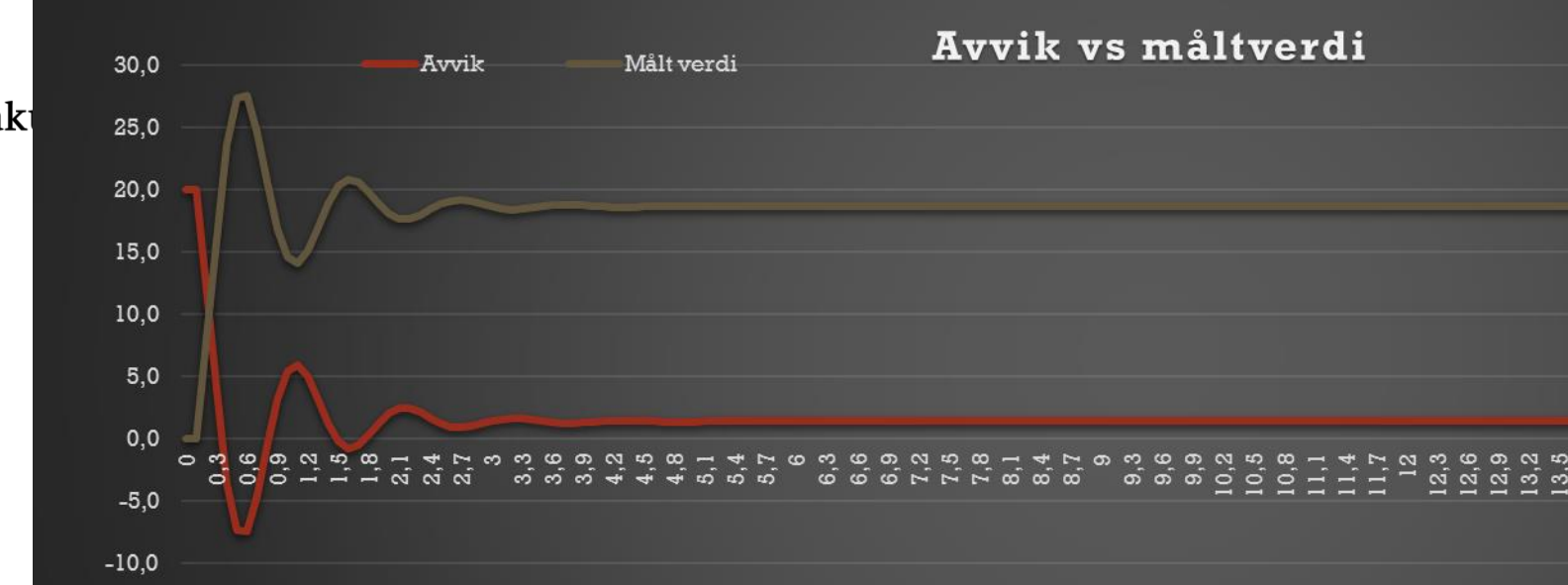
- Vi får oscillasjoner fordi det tar tid før systemet responderer på outputen vi gir.

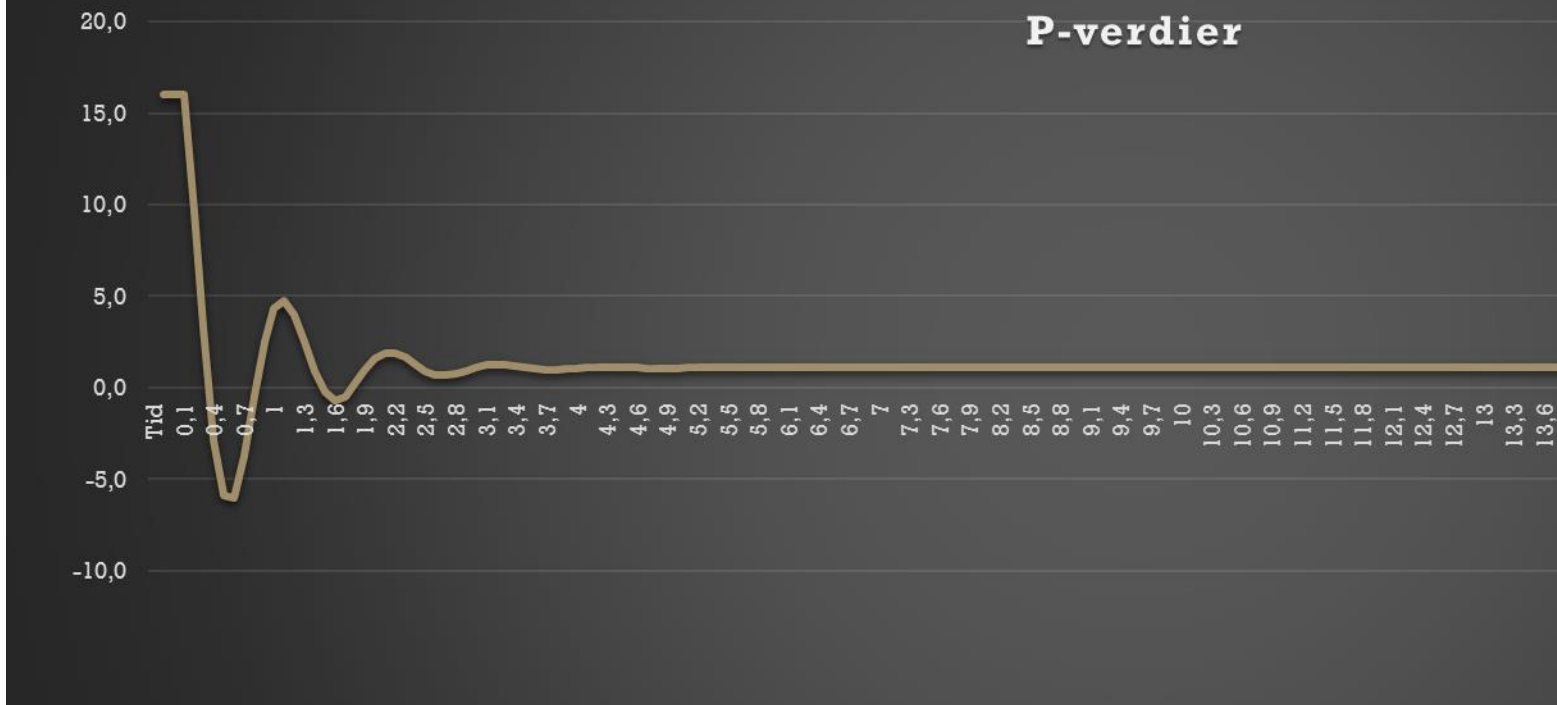


Bare P (forts)

$$K_p \cdot Avvik$$

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

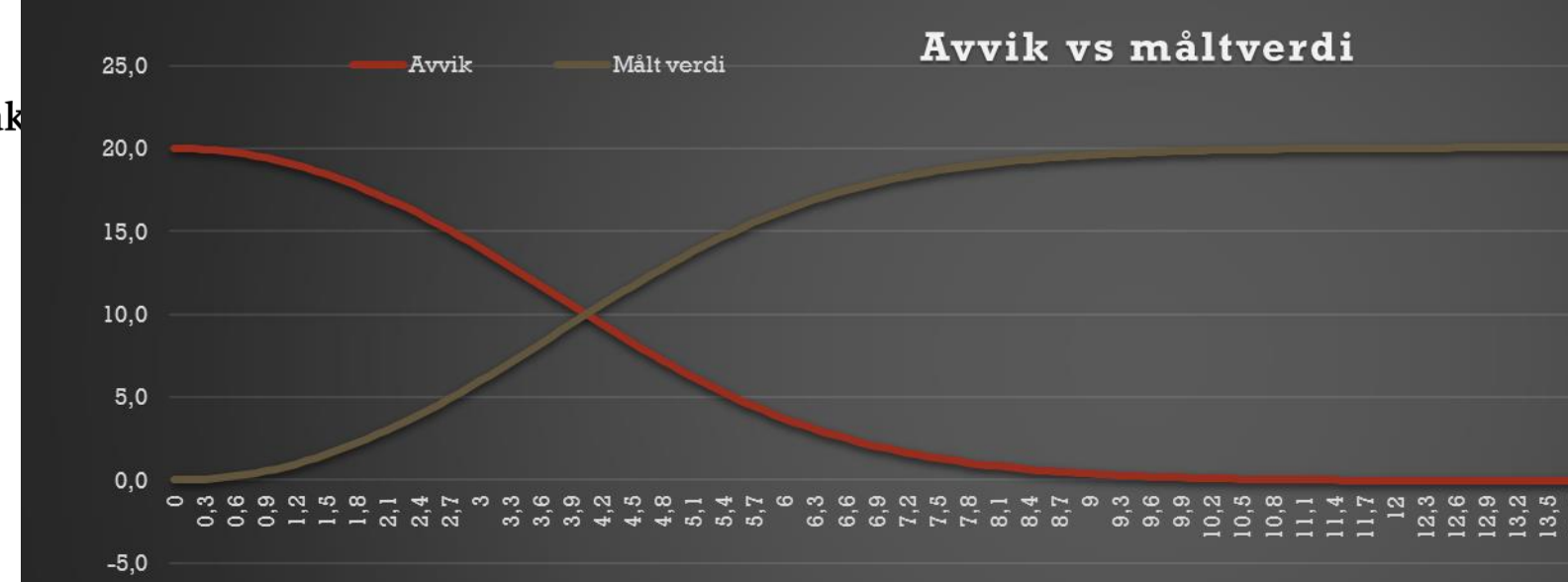


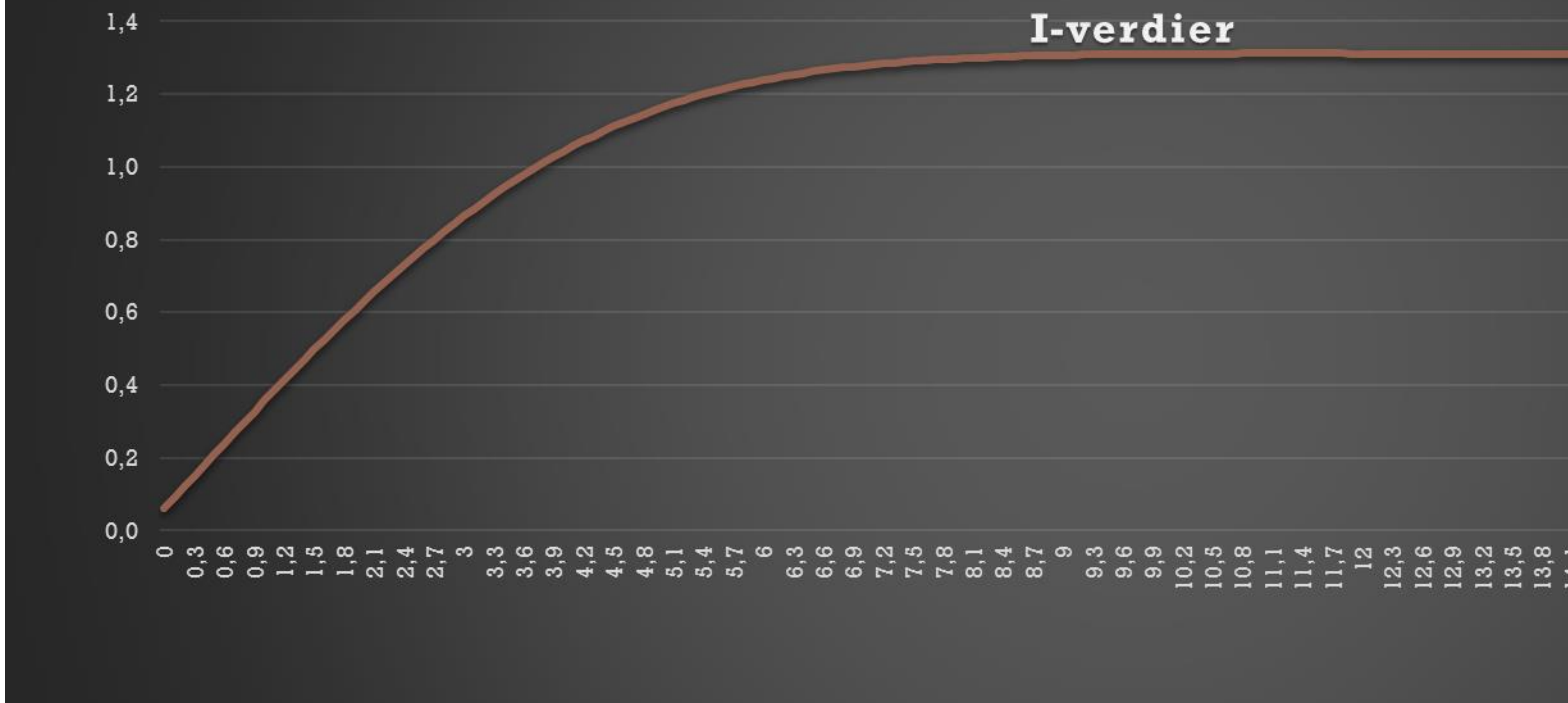


Bare I

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

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

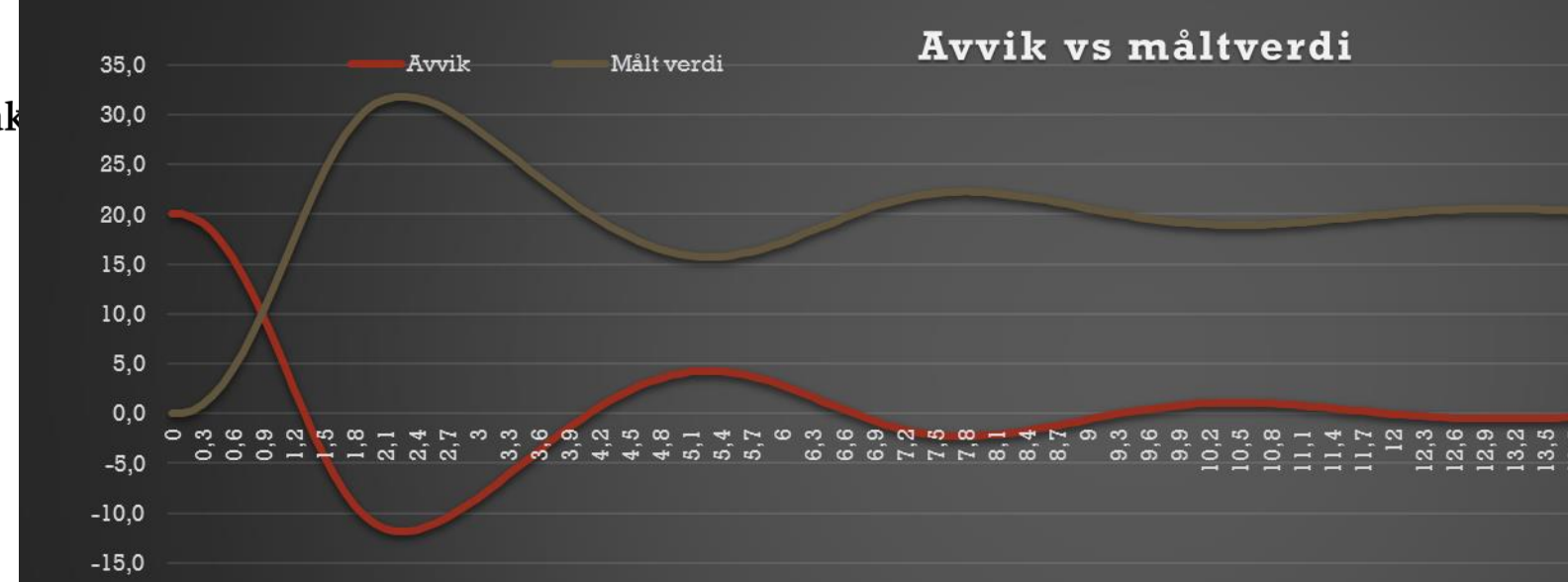


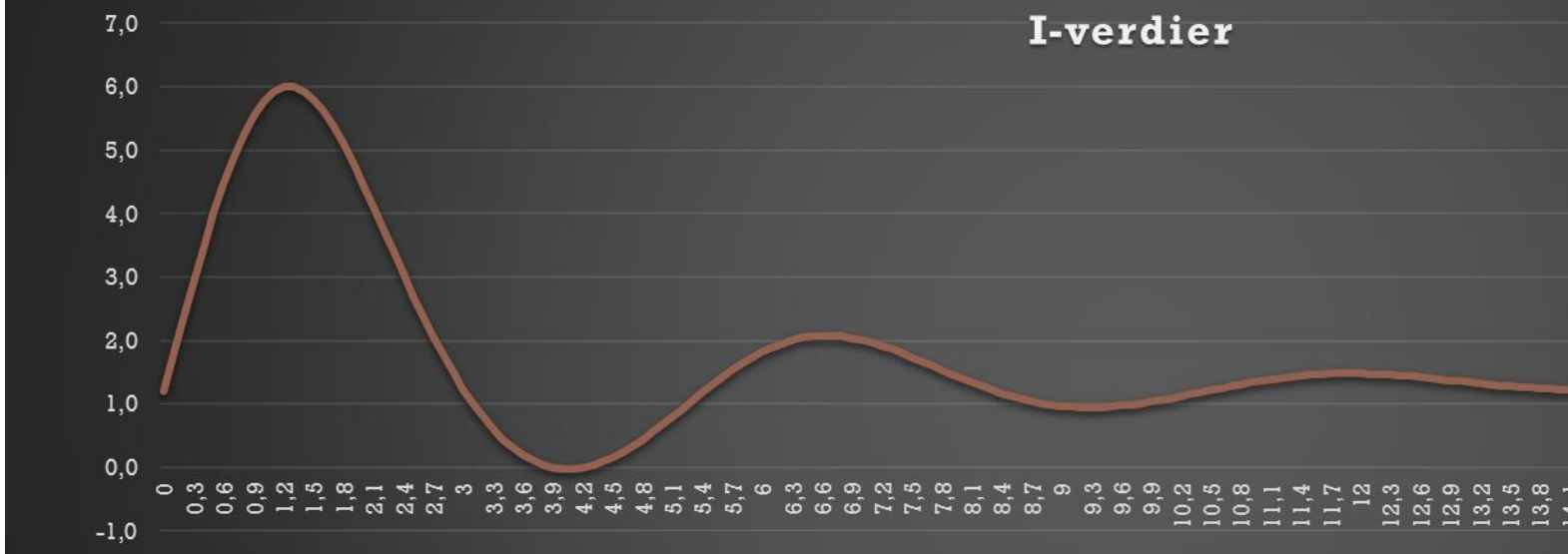


Bare I

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

- Vi får oscillasjoner om Ki blir for stor.

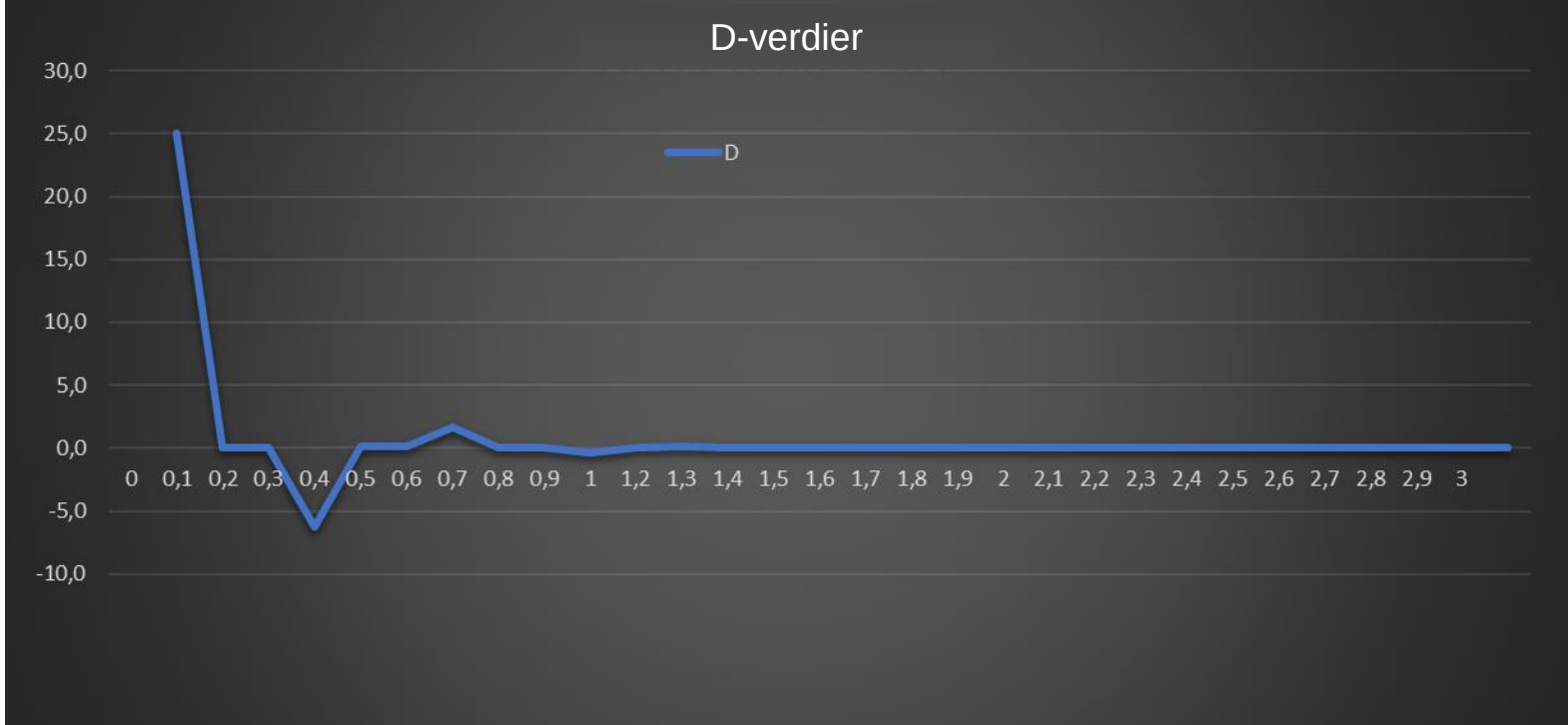




Bare D…

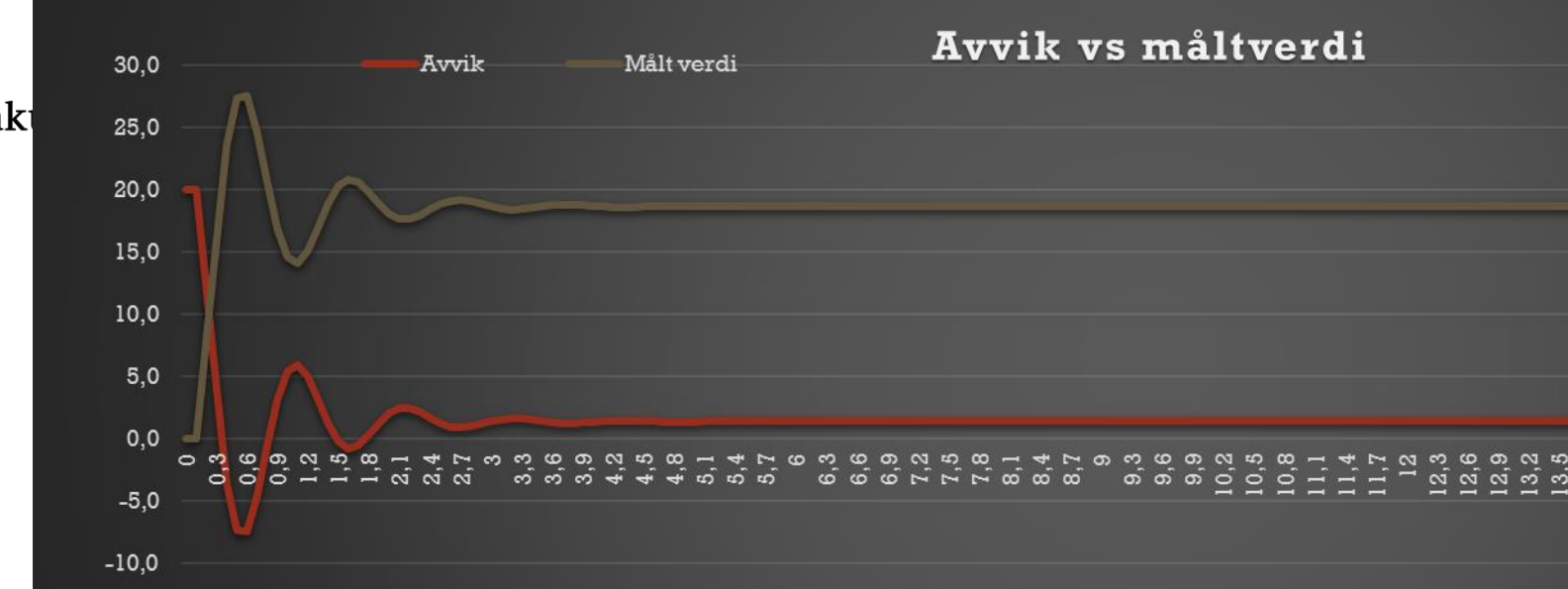
$$K_d(Avvik_n - Avvik_{n-1})$$

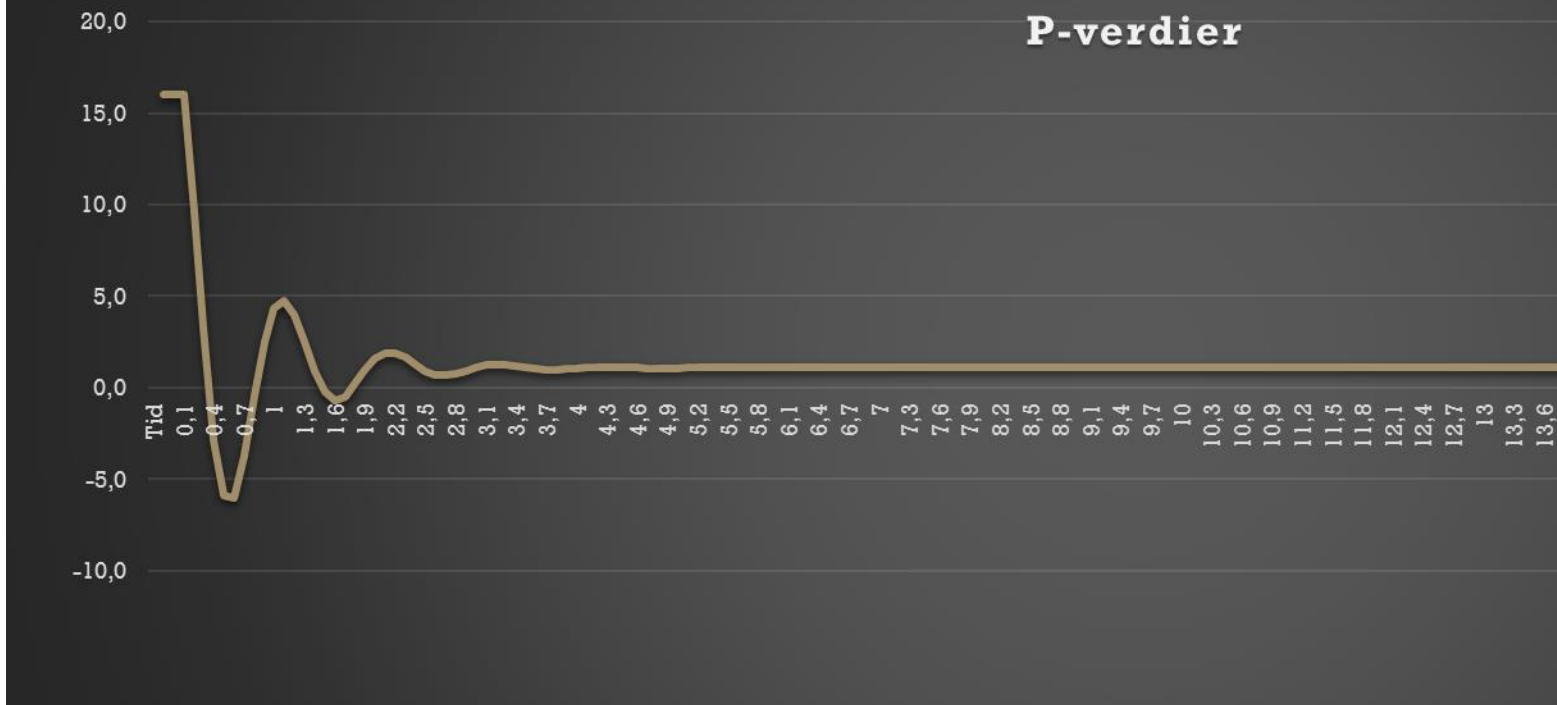
- D-ledd benyttes sjeldent alene, fordi det bare gjøre en forskjell så lenge avviket endrer seg.
- Så fort endringene går mot null, vil virkningen oppheves.
- For stort D-ledd vil skape oscillasjoner f.eks ved støy i måleverdiene.



Bare Stor P (igjen)

- Stor overshoot
- kan ikke nå målet





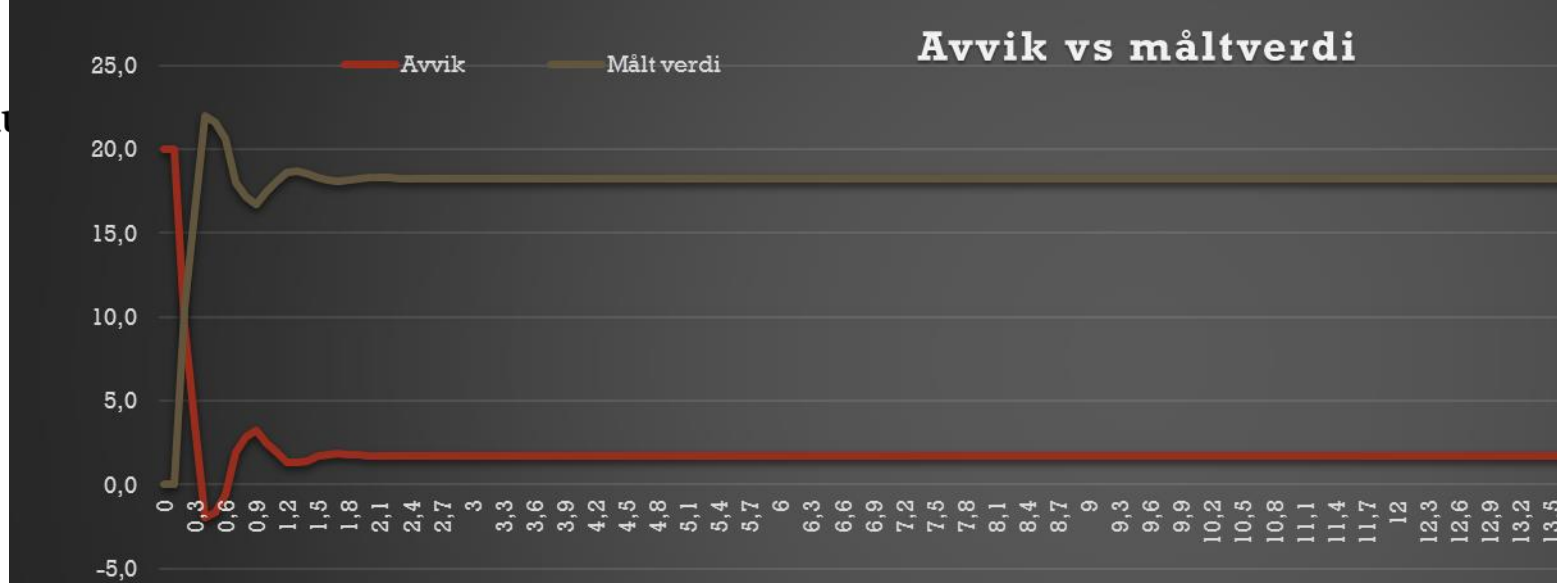
D-ledd: P+D

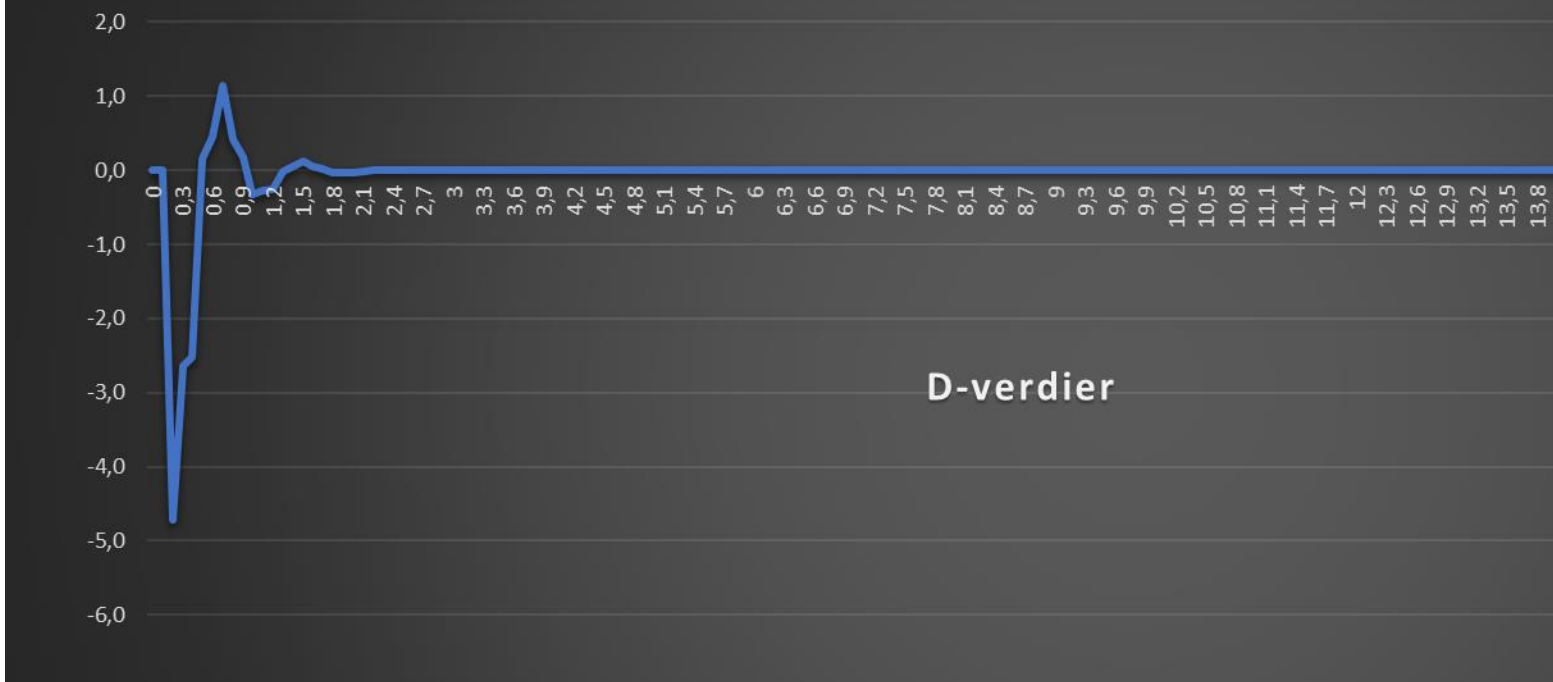
$$K_p \cdot Avvik + K_d(Avvik_n - Avvik_{n-1})$$

- D-leddet kompenserer med forskjellen i mellom forrige og siste avvik. Normalt har dette en dempende effekt på raske endringer.

- P+D-ledd vil heller 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…)

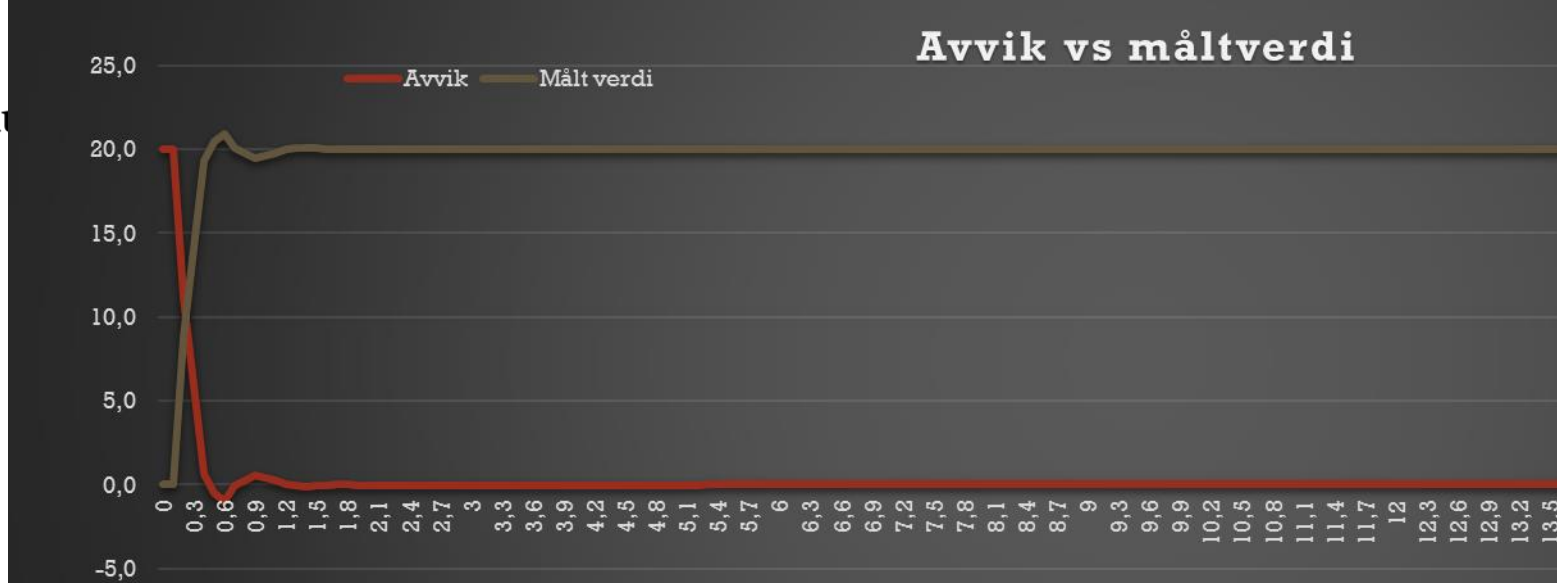


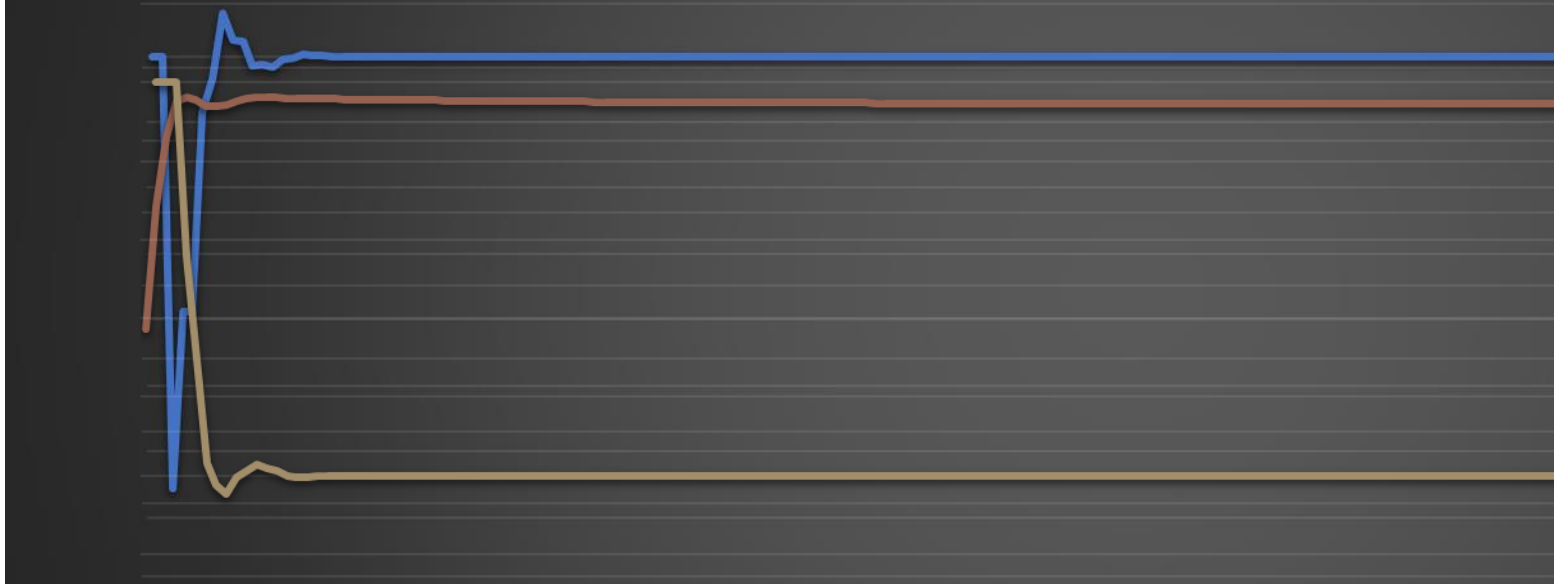


PID

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

- 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

For å lage PID kontroll, må man gjøre følgende

1. Beregne avviket (/feil, «error») mellom ønsket verdi «setpoint» og målt verdi.
   $$avvik_n = målverdi - avlest\ verdi$$
2. Beregne Proporsjonsledd:
   $$P = K_p \cdot avvik$$
3. Beregne Integrasjonsledd:
   $$\mathrm{I} = K_i \cdot avvik + I_{n-1}$$
4. Beregne Derivasjonsledd:
   $$\mathrm{D} = K_d(avvik - avvik_{n-1})$$
5. Beregne $\mathrm{output} = \mathrm{P} + \mathrm{I} + \mathrm{D} =$
   $K_p \cdot avvik + K_i \cdot avvik + I_{n-1} + K_d(avvik - avvik_{n-1})$

Merk: Størrelsen på konstantleddene i integrasjonsledd og derivasjonsledd må skaleres avhengige av hvor ofte PID-verdiene beregnes. (*Bruk interrupt-styring!*)

Python kode

```
Kp = <P-konstant>
KI = <I-konstant>
Kd = <D-konstant>

def PID(   previous_error,       # Forrige avvik
           integral,             # Forrige I-ledd
           setpoint,             # mål
           measured_value):      # avlest verdi

    error = setpoint-measured_value
    P = Kp * error
    I = Ki * error + integral
    D = Kd * (error-previous_error)

    return(P+I+D, error, I)
```

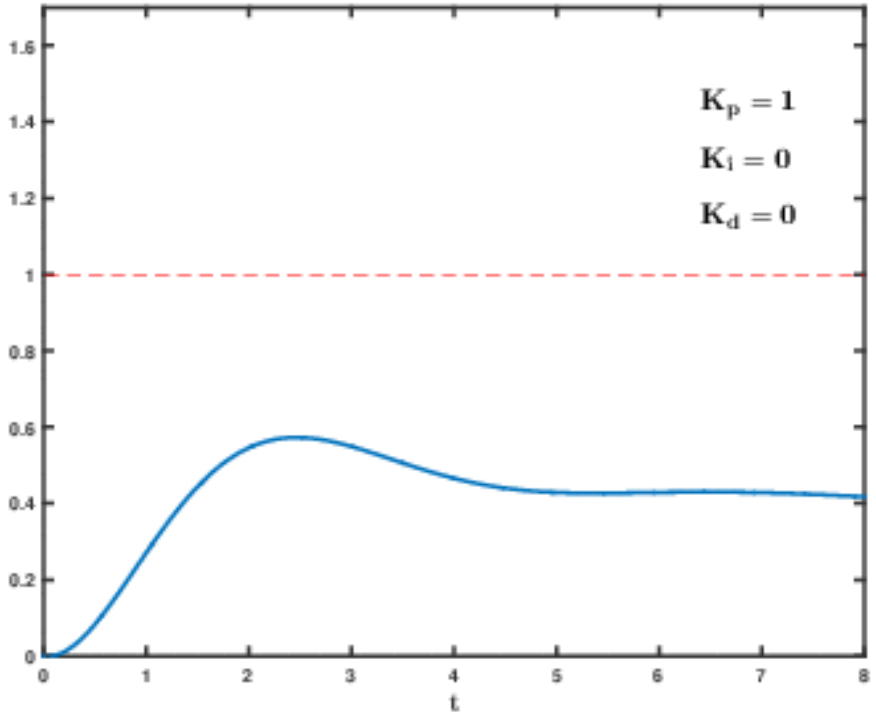
```
myPID  = PID(<forrige feil>,<forrige I>,<mål>,<avlest verdi>)
OUTPUT = myPID[0]
```

```
myPID2 = PID(myPID[1], myPID[2], <mål>, <avlest verdi>)
OUTPUT = myPID2[0]
```

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
    - Delay i respons 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)
Kp	reduserer	øker	øker	reduserer 1
Ki	reduserer	øker	reduserer, øker 2	"eliminerer" 3
Kd 4	øker	reduserer	reduserer	uendret

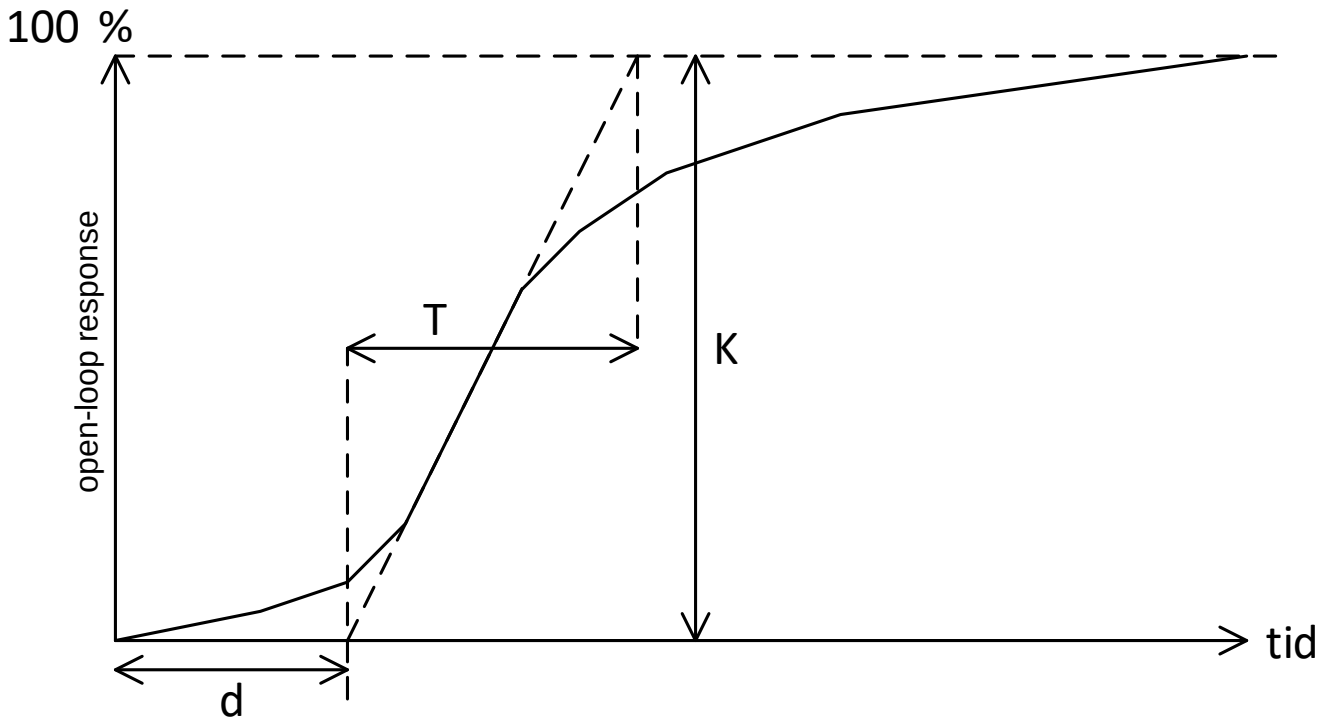
1. Kp kan ikke eliminere avvik, men gjøre det mindre
2. Ki vil redusere stabiliseringstiden frem til man begynner å få overshoot, deretter øker den
3. Bruk av Ki vil eliminere avviket ved stabilitet, så lenge vi oppnår stabilitet. Med høyere Ki, skjer det raskere
4. Kd kompenserer for overshoot og ringing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

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 «delay» før vi når Tangent)
4. $K_d = 0{,}5d$



Merk: Forutsetter at vi har førsteordens system
(=vi får samme svar Gp uansett hvor vi starter)
og kontrolsløyfen svarer raskt (< T/7)

Eks: Vi endrer input fra 2 til 4 og ouput går fra 1 til 2
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).

Anbefalt lesing og oppgaver

- Lese
  - 28.1-28.8 s 677-696

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