

UNIVERSITETET I STAVANGER

DET TEKNISK-NATURVITSKAPLIGE FAKULTET

ANVENDT MATEMATIKK OG FYSIKK I
ROBOTPROGRAMMERING

LEGO Mindstorms og Matlab i skjønn forening

GRUPPENUMMER 2341

Forfattere:

Christoffer Askeli ENOKSEN
Henry EIKELAND
Hermann Wathne TVEIT

Emneansvarlig:

Prof. Tormod
DRENGSTIG

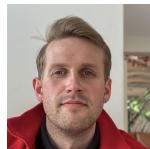
28. april 2023



Forord

Denne rapporten er skrevet for å dokumentere en rekke praktiske eksperiment utført med LEGO-EV3 robot. Rapporten er en del av vurderingsgrunnlaget i studieemnet *ELE130-1 23V Anvendt matematikk og fysikk i robotprogrammering*. Totalt ni uker er gått med til utførelse og dokumentasjon av de praktiske forsøkene. Medvirkende til rapporten er kreditert med bilde under.

Takk til emneansvarlig Prof. Tormod DRENGSTIG for en grundig introduksjon til emnet, og god oppfølging underveis i prosjektet. Også takk til flinke studentassisterenter som har bidratt med faglig veiledning og føret gruppemedlemmene med vafler hver tirsdag i stolpetimen.



Signatur: _____

Christoffer Askeli ENOKSEN
Automatisering og elektronikkdesign Y-vei



Signatur: _____

Henry EIKELAND
Automatisering og elektronikkdesign SOTS-vei



Signatur: _____

Hermann Wathne TVEIT
Automatisering og elektronikkdesign Y-vei

Innhold

1	Innledning	1
2	Numerisk integrasjon	2
2.1	Innledning	2
2.2	Problemstilling	3
2.3	Løsningsforslag	4
2.3.1	MATLAB kode	5
2.4	Verifisering av koden for numerisk integrasjon, del 1	6
2.4.1	Verifisering av koden for numerisk integrasjon, del 2	7
2.4.2	Verifisering av koden for numerisk integrasjon, del 3	11
2.4.3	Verifisering av koden for numerisk integrasjon, del 4	13
2.5	Integrasjon som funksjon	14
3	Filtrering	15
3.1	Innledning	15
3.2	Problemstilling	16
3.3	Matematisk Beskrivelse	16
3.4	MATLAB kode	20
3.5	Verifisering av beregningene gjort i FIR- og IIR-filter	22
3.5.1	Verifikasjon av FIR-filter	23
3.5.2	Verifikasjon av IIR-filter	23
3.6	Filtrere målt kaffetemperatur	24
3.7	Filtrering av målestøy i en industriell temperaturtransmitter	26
4	Numerisk derivasjon	28
4.1	Innledning	28
4.2	Problemstilling	29
4.3	Løsningsforslag	29
4.4	Matlab kode	30
4.5	Verifisering av koden for numerisk-derivasjon, del 1	31
4.6	Verifisering av koden for numerisk derivasjon, del 2	33
4.6.1	Eksperimenter	35
4.7	Derivasjon som funksjon	38
5	Manuell kjøring	39
5.1	Innledning	39
5.2	Problemstilling	39
5.3	Løsningsforslag	40
5.4	Kvalitetsmål	40
5.5	MATLAB kode	42
5.6	Resultat	44
6	Estimering av hastighet	47
6.1	Innledning	47
6.2	Problemstilling	48
6.3	Løsningsforslag	49

6.3.1	Estimering av hastighet fra ultralydmålinger	49
6.3.2	Estimering av fart fra pulsmåling	50
6.4	Resultat	52
7	Automatisk kjøring med PID-regulator	55
7.1	Innledning	55
7.2	Problemstilling	55
7.3	Løsningsforslag	56
7.4	MATLAB-kode	59
7.5	Resultat	61
7.5.1	P-regulator	61
7.5.2	PI-regulator	64
7.5.3	PID-regulator	66
8	Konklusjon	71
	Referanser	72
A	Timeliste	72

Forkortelser

FIR Finite Impulse Response

HMI Human Machine Interface

IAE Integral of Absolute Error

IIR Infinite Impulse Response

MA Moving Average

MAE Mean Absolute Error

PID Proporsjonal Integral Derivat

TV Total Variation

Sammendrag

Denne prosjektoppgaven utforsker hvordan ulike praktiske problemer kan løses ved hjelp av anvendt matematikk og fysikk i robotprogrammering. Problemstillinger som hastighetsberegning, bearbeiding av støybefengte signal og autonom kjøring er blant de som prøves ut og beskrives i detalj. Studiene viser at man ved hjelp av *numerisk integrasjon* er i stand til å dele en kurve opp i flere mindre søyler, og videre danne et bilde over akkumulert mengde.

Studiene viser også to ulike filtreringsmetoder *IIR-* og *FIR-filtrering*, som begge brukes til å håndtere støyende signal. Videre har studiene vist at hastighet kan estimeres ved hjelp av både ultralyd-avstandsmåling og lysmåling, og at lysmålingen kan brukes som hjelpemiddel for autonom kjøring.

Resultatene av denne studien viser at man ved hjelp av grunnleggende matematikk, fysikk og programmering er i stand til håndtere komplekse praktiske prosjekt.

1 Innledning

Motivasjonen bak de dokumenterte forsøkene i denne rapporten er å få inn-sikt i hvordan matematiske metoder som blant annet derivasjon og integrasjon blir brukt i praksis. Gjennom en rekke forsøk har prosjektgruppen tilegnet seg kunnskap om anvendt matematikk og fysikk. Prosjektet ble påbegynt 27. februar 2023 og hadde en tidsramme på seksti (60) dager. På disse seksti dagene har prosjektgruppen først individuelt gjennomført en obligatorisk del (kap. 2, 3, 4). Videre ble gruppen delt i par for å gjennomføre flere større prosjektoppgaver. Totalt er det dokumentert seks (6) delprosjekt.

2 Numerisk integrasjon

Deltakere:

Christoffer Askeli ENOKSEN

Henry EIKELAND

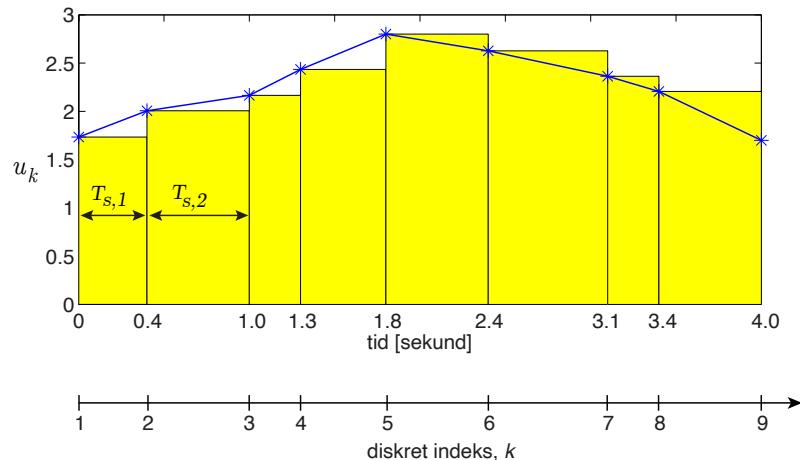
Hermann Wathne TVEIT

2.1 Innledning

I mange prosesser ønsker vi å vite noe om f.eks. mengden væske som har strømmet i et rørsystem, vi bruker da instrumenter som gir oss målinger om den øyeblikkelige strømningen ved gitte tidspunkt. For å finne den totale mengden væske som har strømmet i rørsystemet over tid, kan vi benytte oss av numerisk integrasjon.

Numerisk integrasjon er en matematisk teknikk som brukes for å finne arealet under en kurve eller en funksjon. For å finne arealet under en kurve deler vi opp området vi ønsker å finne arealet til i mindre segment. Hvert av disse segmentene har en bredde: avstanden mellom to punkt på x-aksen, og en høyde: den aktuelle funksjonsverdien relativ til den første funksjonsverdien. For å finne arealet av hvert segment multipliserer vi høyde med bredde og ender opp med et rektangel *søyle*.

For å finne det totale arealet under kurven summerer vi alle segmentarealetene (*med fortegn*). Figur 1 illustrerer en serie av målte verdier over en tidsperiode på 4 sekund. Ved å interpolere verdien mellom søylene avsløres funksjonskurven.



Figur 1: Prinsipp for numerisk integrasjon , T_s gitt som bredden til søylene, u_k som høyden av søylene og interpolering mellom søylene som gir den antatte funksjonskurven. Figur fra [1].

2.2 Problemstilling

Tilnærming av integral ved funksjonsverdiene.

Det finnes forskjellige metoder for denne prosessen, vi skal ta utgangspunkt i Eulers forovermetode:

$$y_k = y_{k-1} + T_s \cdot u_{k-1} \quad (1)$$

Vi ønsker å implementere numerisk integrasjon fra signalet til lyssensoren på LEGO EV3 roboten, lysmålingene skal representer et flowsignal. Lyssensoren avgir et konstant lys og gir en måling på hvor mye av lyset som blir reflektert tilbake. Målingene er gitt som reflektert[%] hvor fargen hvit tilsvarer 100 % reflektert og sort 0 % reflektert. Oppsett for lysmålinger gitt ved figur 2.



Figur 2: Lyssensor tilkoblet LEGO EV3 enhet. Lyssensor montert med avstandsklosser i front for få en konstant avstand mot underlaget (gråskala-arket).

Ved å bruke det matematiske uttrykket gitt ved likning (1) blir integralverdien fortløpende regnet ut.

Som vist i figur 1 så er størrelsen på T_s ikke konstant, lysmålingene fra LEGO-EV3 enheten og datainnsamlingen (PC/Matlab) har et variabelt oppdateringsintervall. En mulig grunn til at T_s ikke er konstant kan være at kommunikasjonen mellom EV3 enheten og datamaskinen ikke er konsistent, fordi andre prosesser som kjører på datamaskinen må fullføres. Dynamisk klokkefrekvens til prosessoren i datamaskinen for å redusere batteriforbruk kan også påvirke oppdateringsintervallet for målingene.

Ved å utføre Eulers forovermetode gitt av likning (1) på eksempladata i Figur 1, ser vi at tilnærming av integralet blir uriktig da Eulers forovermetode forutsetter konstant tidsskritt.

2.3 Løsningsforslag

Ved numerisk implementering av Eulers forovermetode gitt av likning (1), trenger vi å utforme en løsning på det variable tidsskrittet. Som vist med figur 1 dannes en kronologisk tidsvektor sortert etter indeks. Ved å bruke likning 2 kan tidsskrittet regnes ut.

$$Ts_{,k} = Tid_{,k} - Tid_{,k-1} \quad (2)$$

Gitt av figur 1 har vi vektoren med tid siden start:

$$\mathbf{Tid} = [0.0, \ 0.4, \ 1.0, \ 1.3, \ 1.8, \ 2.4, \ 3.1, \ 3.4] \quad (3)$$

For regne ut første tidskritt $Ts_{,1}$:

$$Ts_{,1} = Tid_{,1} - Tid_{,1-1} \quad (4)$$

Vi ser vektoren Tid ikke inneholder $Tid_{,0}$. Vi initiererer tidsskritt $Ts_{,1}$ til en fast verdi på forhånd, da denne verdien ikke har noe betydning for arealberegningen, men er allikevel nødvendig for at Tid- og Lys-vektoren har like mange element.

2.3.1 MATLAB kode

Vi ønsker å utføre numerisk integrasjon på signalet fra lyssensoren som et tolket flowsignal. Signalet fra lyssensoren er kun positivt, men vi ønsker også muligheten til å ha negativ flow. Man kan relatere til at du både kan drikke og fylle vann i et vannglass. Vi løser dette ved at første måling fra lyssensoren setter en referanse for flowsignalet. Det vil si at dersom første måling fra lyssensoren er 50% og neste måling er 100% så vil dette indikere positiv flow (påfylling). Med andre ord: *hvis en måling er større enn referanseverdien vil dette si at man fyller vann*; og motsatt for drikking.

Vi skriver om likningene for tidskritt (2) og Eulers forovermetode (1) slik at de passer til implementering som kode i MATLAB.

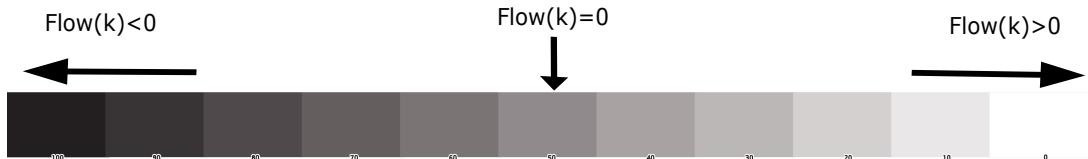
Kode 1: Kodeutdrag av numerisk integrasjon i MATLAB.

```
1 if k==1
2     % Inialverider første iterasjon av programmet
3     Ts(1) = 0.005;
4     volum(1) = 0;
5     Flow(1) = 0;
6     %Setter nullflow/referanseverdi lik første lysmåling
7     nullflow(1) = Lys(1);
8 else
9     %Beregner flowverdi i forhold til referanseflowen.
10    Flow(k) = Lys(k)-nullflow(1);
11    % Beregner Tidskritt
12    Ts(k)= Tid(k)-Tid(k-1);
13    % Beregner integralverdien
14    volum(k) = volum(k-1) + Ts(k)*Flow(k-1);
15 end
```

2.4 Verifisering av koden for numerisk integrasjon, del 1

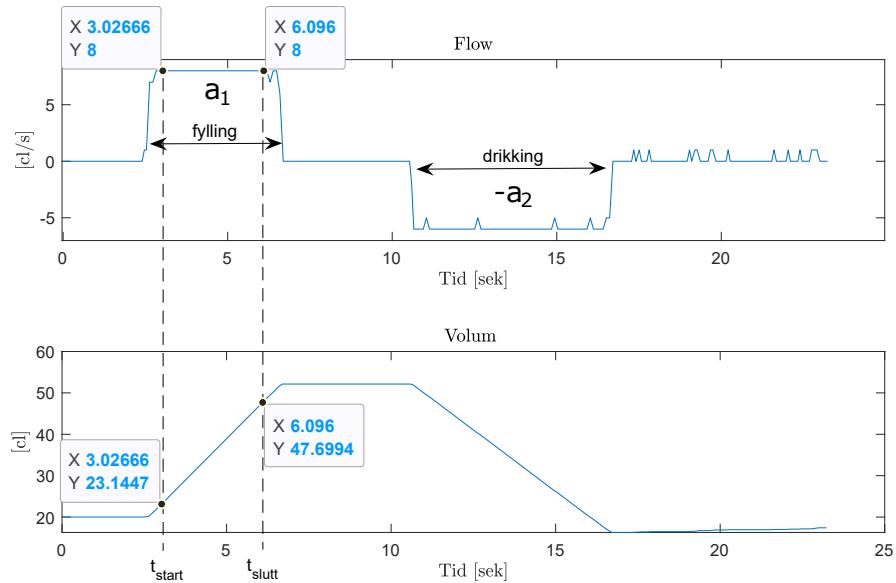
For å verifisere at koden for numerisk integrasjon er riktig implementert, utfører vi en test som tolkes som en konstant vannflow i et vannglass. Vi bruker lyssensoren til å generere flowsignalet, for så å verifisere den kalkulerte integralverdien med kontrollregning.

Lyssensoren plasseres i nøytral posisjon mellom sort og hvit på figur 3, dette blir første måling og derfor referanseverdien (*nullflow-variablene*) i MATLAB-koden. For å lage et konstant flowsignal til påfylling i glasset flyttes sensoren



Figur 3: Gråskala for tolking av flowsignal. [1]

hurtig to gråskalaverdier mot hvitt. Fyllingen stoppes ved å flytte lyssensoren tilbake til nullpunktet. Etter noen sekunder skal det tappes fra glasset. Sensoren flyttes da hurtig to gråskalaverdier mørkere, og etter noen sekunder stoppes tappingen ved å flytte lyssensoren tilbake til nullpunktet. Resultatet er vist i figur 4



Figur 4: Figur i topp ved $a_1 = 8 \text{ cl/s}$ som viser raten det fylles vann i glasset ved $t_{start} = 3.026\text{s}$ til $t_{slutt} = 6.096\text{s}$. Figur i bunn viser volum i glasset med initialverdi på 20 cl.

Signalet fra lyssensoren er ufiltrert og lett mottakelig for støy fra omgivelsene. Glasset får derfor en initialverdi på 20 cl for å unngå at volumet blir negativt. Figur 4 viser at det er sammenheng mellom strømningshastighet og stigende volum, og motsatt for negativ strømningshastighet. Det bestemte integralet til flowen mellom $t_{start} = 3.026$ og $t_{slutt} = 6.096$ beregnes for å kontrollere om den estimerte volumendringen i glasset etter påfyllingen er riktig.

$$\begin{aligned}
 \hat{A} &= \int_{t_{start}}^{t_{slutt}} a_1 d\tau \\
 &= \int_{3.026}^{6.096} 8 d\tau \\
 &= 8 \cdot \tau \Big|_{3.026}^{6.096} \\
 &= 8[\text{cl}/\text{s}] \cdot (6.096 - 3.026)[\text{s}] \\
 &= \underline{\underline{24.56[\text{cl}]}}
 \end{aligned} \tag{5}$$

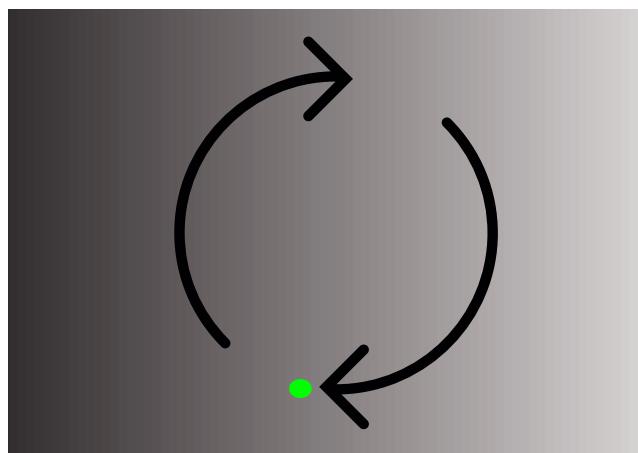
Gitt av likning 5 så har det i perioden for a_1 blitt fylt 24.56 cl med vann i glasset. Vi kan kontrollere dette ved å subtrahere volumet ved t_{slutt} fra t_{start} . Resultatet skal være \approx volumendringen i glasset på 24.56 cl.

$$\begin{aligned}
 \Delta \text{ Volum} &= \text{Volum}(t_{slutt}) - \text{Volum}(t_{start}) \\
 &= 47.699[\text{cl}] - 23.144[\text{cl}] \\
 &\approx \underline{\underline{24.55[\text{cl}]}}
 \end{aligned} \tag{6}$$

Resultatet fra verifikasjonen tilsier at koden er riktig implementert for integrasjon av en konstant.

2.4.1 Verifisering av koden for numerisk integrasjon, del 2

Ved å utsette lyssensoren for et roterende gråskalaark kan vi generere et periodisk signal, i dette tilfelle ønsker vi et sinussignal.



Figur 5: Prinsipp for generering av sinussignal med lyssensoren. Grønt punkt er startpunkt og nøytral verdi på gråskalaen.

Dersom et sinussignal har like mye positivt og negativt utslag omkring likevektslinjen *symmetrisk funksjon omkring x-aksen* vil integralet av dette signalet over en periode være lik null. Dette skyldes at de positive og negative delene av signalet vil kansellere hverandre ved integrasjon. Det å generere en symmetrisk sinusfunksjon for hånd med lyssensoren er en stor utfordring. Vist i figur 6 benyttes gråskalaarket som en LP-plate i platespilleren. Det vil da bli en mer kontrollert hastighet og amplitude på signalet.

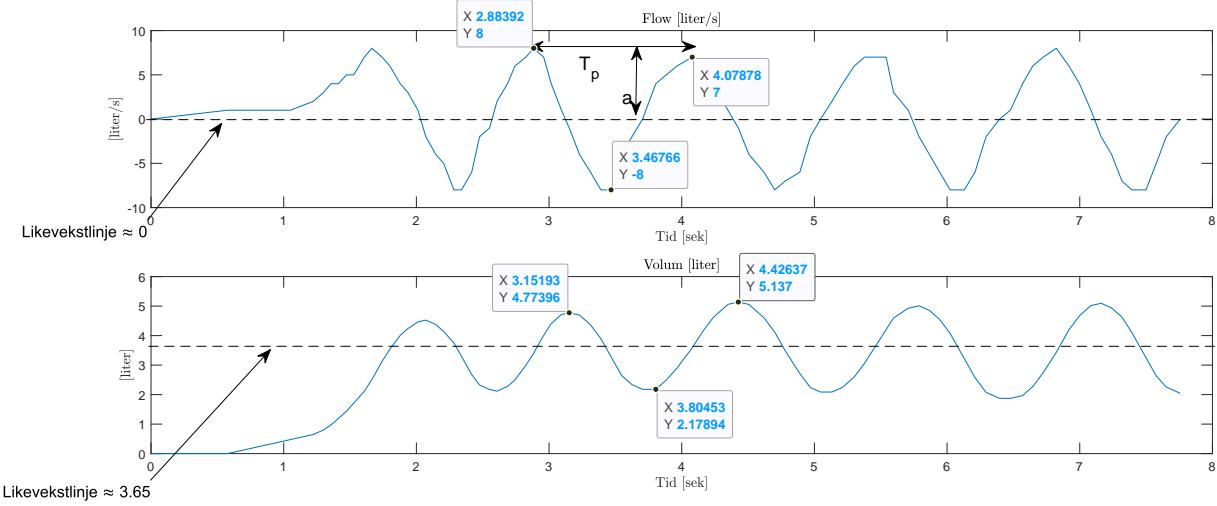


Figur 6: Platespiller oppsett for generering av sinus. A3-gråskalaark tilpasset med laserkutter.

Som vist i figur 5 starter målingen ved det grønne punktet for å sette referanseverdien (nullflow). Dermed blir dette en sinusformet drikke- og fyllefunksjon for vannglasset.

$$\begin{aligned} \text{Volum}(t) &= \int_0^t a \cdot \sin(\omega \cdot \tau) d\tau \\ &= -\frac{a}{\omega} \cdot \cos(\omega \cdot t) \end{aligned} \quad (7)$$

For å verifisere koden for numerisk integrasjon, utfører man verifikasjon av sinusflowsignalet. Gitt av likning (7) blir et integrert sinusflowsignal om til et cosinus volumsignal. Ved å se på at volumet i glasset er en cosinusfunksjon, kan det observeres at volumet vil ha faseforskyvning i forhold til strømningshastigheten på 90 grader.



Figur 7: Resultat av sinus flowsignal generert på platespiller innstilt til 4.71 rad/s. Øvre delfigur viser sinussignal som flow, nedre delfigur viser beregnet volum av sinussignalet

Ved å hente ut verdier fra grafen i figur 7. For øvre del av figuren som representerer sinussignalet for flow.

amplitude a for flowsignalet:

$$a = \frac{y_{max} - y_{min}}{2}$$

$$= \frac{8 - (-8)}{2} = 8 \text{ [l/s]} \quad (8)$$

Vinkelfrekvens ω for flowsignalet:

$$\omega = \frac{2\pi}{T_p}$$

$$= \frac{2\pi}{4.07 - 2.88} \approx 5.27 \text{ [rad/s]} \quad (9)$$

Fra likning (7), beregnes den teoretiske amplituden for volumsignalet

Teoretisk amplitude a for volumsignalet:

$$a = \frac{a}{\omega}$$

$$= \frac{8}{5.27} = 1.51 \text{ [l]} \quad (10)$$

Teoretisk faseforskyvning ϕ for volumsignalet:

$$\phi = 90^\circ \quad (11)$$

Verdiene fra flowsignalet er nå kjent, og verdiene for volumsignalet leses av figur (7). Ved å punktvis sammenligne avleste verdier fra figuren mot beregnede verdier, kan man verifisere koden for numerisk integrasjon av et sinussignal.

Amplitude a for volumsignalet:

$$\begin{aligned} a &= \frac{y_{max} - y_{min}}{2} \\ &= \frac{5.13 - 2.17}{2} = \underline{1.48 [l]} \end{aligned} \tag{12}$$

Vinkelfrekvens ω for flowsignalet:

$$\begin{aligned} \omega &= \frac{2\pi}{T_p} \\ &= \frac{2\pi}{4.42 - 3.15} \approx \underline{4.94 [rad/s]} \end{aligned} \tag{13}$$

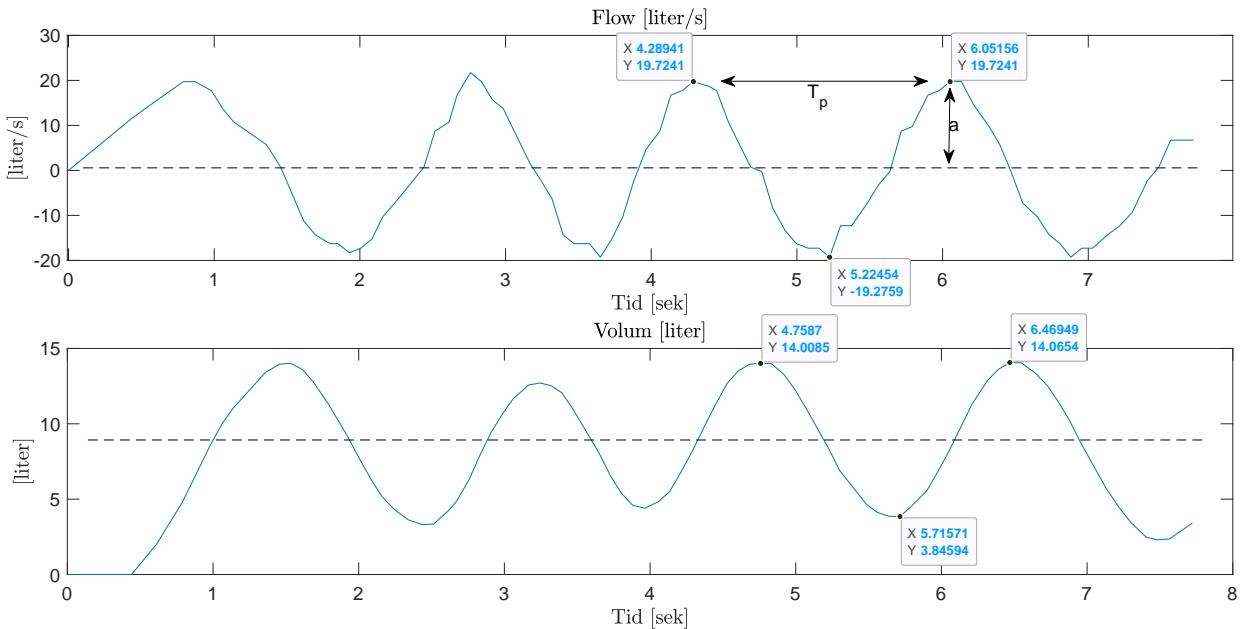
Faseforskyvning ϕ for volumsignalet:

$$\begin{aligned} \phi &= \frac{\Delta t}{T} \cdot 360^\circ \\ &= \frac{(3.15 - 2.88)}{4.42 - 3.15} \cdot 360^\circ = 76.5^\circ \end{aligned} \tag{14}$$

På grunn av tidsskrittet som er relativt stort i datasettet, er noe avvik forventet. Amplitude og faseforskyving har tilnærmet de teoretiske verdiene, disse er regnet ut med hensyn på likning 7. Vinkelfrekvensen som sinusgeneratoren(platespilleren) er innstilt til er også å finne igjen i signalene.

2.4.2 Verifisering av koden for numerisk integrasjon, del 3

Eksperimentet fra del én gjentas men denne gangen med en annen amplitud og vinkelfrekvens på flowsignalet. Det kontrolleres om sammenhengene i likning (7) stemmer. For å oppnå en større amplitude flyttes lyssensoren lengre ut fra senter på gråskalaarket. Platespilleren er innstilt på en lavere vinkelfrekvens. Resultatene er vist i figur 8.



Figur 8: Resultat av sinus flowsignal generert på platespiller innstilt til 3.45 rad/s. Øvre delfigur viser sinussignal som flow, nedre delfigur viser beregnet volum av sinussignalet.

Verdier fra grafen i figur 8 hentes ut.

Amplitude a for flowsignalet:

$$\begin{aligned} a &= \frac{y_{max} - y_{min}}{2} \\ &= \frac{19.72 - (-19.27)}{2} = 19.5 \text{ [l/s]} \end{aligned} \tag{15}$$

Vinkelfrekvens ω for flowsignalet:

$$\begin{aligned} \omega &= \frac{2\pi}{T_p} \\ &= \frac{2\pi}{6.05 - 4.28} \approx 3.54 \text{ [rad/s]} \end{aligned} \tag{16}$$

Fra likning (7), beregnes den teoretiske amplituden for volumsignalet.

Teoretisk amplitude a for volumsignalet:

$$\begin{aligned} a &= \frac{a}{\omega} \\ &= \frac{19.5}{3.54} = \underline{5.5[l]} \end{aligned} \tag{17}$$

Teoretisk faseforskyvning ϕ for volumsignalet:

$$\phi = 90^\circ \tag{18}$$

Verdiene fra flowsignalet er kjent og verdiene for volumsignalet leses av figur (7). De teoretiske verdiene beregnet med likning (7) sammenlignes med volumsignalet i figur (7).

Amplitude a for volumsignalet:

$$\begin{aligned} a &= \frac{y_{max} - y_{min}}{2} \\ &= \frac{14 - 3.84}{2} = \underline{5.1 [l]} \end{aligned} \tag{19}$$

Vinkelfrekvens ω for flowsignalet:

$$\begin{aligned} \omega &= \frac{2\pi}{T_p} \\ &= \frac{2\pi}{6.46 - 4.75} \approx \underline{3.67 [rad/s]} \end{aligned} \tag{20}$$

Faseforskyvning ϕ for volumsignalet:

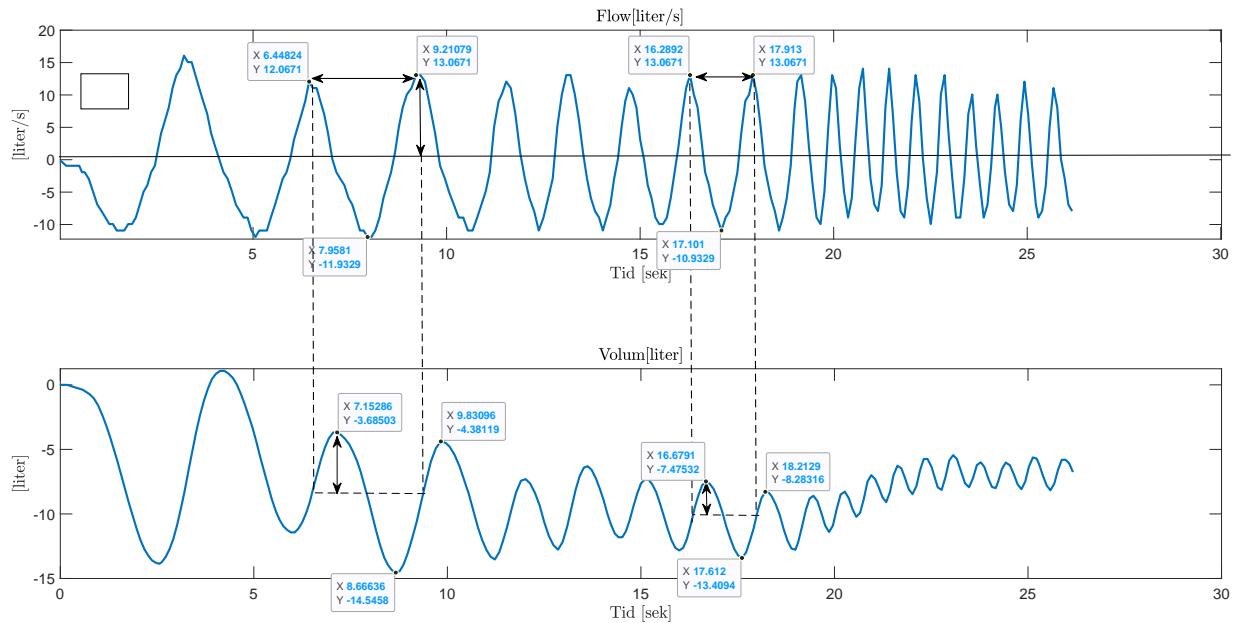
$$\begin{aligned} \phi &= \frac{\Delta t}{T} \cdot 360^\circ \\ &= \frac{(4.75 - 4.28)}{6.46 - 4.75} \cdot 360^\circ = 85^\circ \end{aligned} \tag{21}$$

I dette tilfellet er også amplitude og faseforskyving tilnærmet de teoretiske verdiene som er beregnet, med hensyn på likning (7). Vinkelfrekvensen til platespilleren også er å finne igjen i signalene. De avleste målingene hadde litt mindre avvik i forhold til de teoretiske beregningene i denne oppgaven. En mulig grunn til dette er at sinussignalet hadde en lavere vinkelfrekvens som gjør at man får flere målepunkter per periode. Noe som igjen gjør at mindre data går tapt på grunn av at det ikke blir tatt målinger ofte nok til å fange opp alle detaljene i signalet.

2.4.3 Verifisering av koden for numerisk integrasjon, del 4

For å observere hvordan ledetet $\frac{a}{\omega}$ i likning(7) påvirker amplituden til det bestemte integralet av sinus-flowsignalet, ønsker vi å generere et *chirp-signal* med økende vinkelfrekvens og fast amplitude. Dette vil gi oss mulighet til å se hvordan amplituden minsker når vinkelfrekvensen til flowsignalet øker.

For å generere chirp-signalet, roteres lyssensoren over gråskalaarket i en fast bane, men med økende hastighet. Dette vil skape et signal med økende vinkelfrekvens med en fast amplitud. Ved å visualisere flowsignalet og volumsignalet i figur 9, vil man kunne se hvordan ledetet $\frac{a}{\omega}$ påvirker amplituden til det bestemte integralet av strømsignalet.



Figur 9: Resultat av *chirp*-signal generert for hånd. Øvre delfigur viser sinus-signal som flow, nedre delfigur viser beregnet volum av *chirp*-signalet.

Vist i figur (9) kan man observere sammenhengen mellom vinkelfrekvensen og amplituden til volumsignalet. Når vinkelfrekvensen til flowsignalet øker, så minsker amplituden til volumsignalet proporsjonalt.

Resultatene fra verifikasjonen tilsier at koden for numerisk integrasjon er riktig. Hvor koden er verifisert med signaltypene: konstant, sinus og chirp.

2.5 Integrasjon som funksjon

Siden koden for Eulers forovermetode for integrasjon er verifisert kan det være praktisk å lage en funksjonsblokk i MATLAB som enkelt kan benyttes ved senere behov.

Kode 2: Funksjon for numerisk integrasjon i MATLAB

```
1 %%EulerForward(IntValueOld,FunctionValue,TimeStep)
2 %Funksjon for Eulers forovermetode
3 function[IntValueNew]=EulerForward(IntValueOld,FunctionValue,TimeStep)
4     IntValueNew=IntValueOld + (TimeStep*FunctionValue);
5 end
```

3 Filtrering

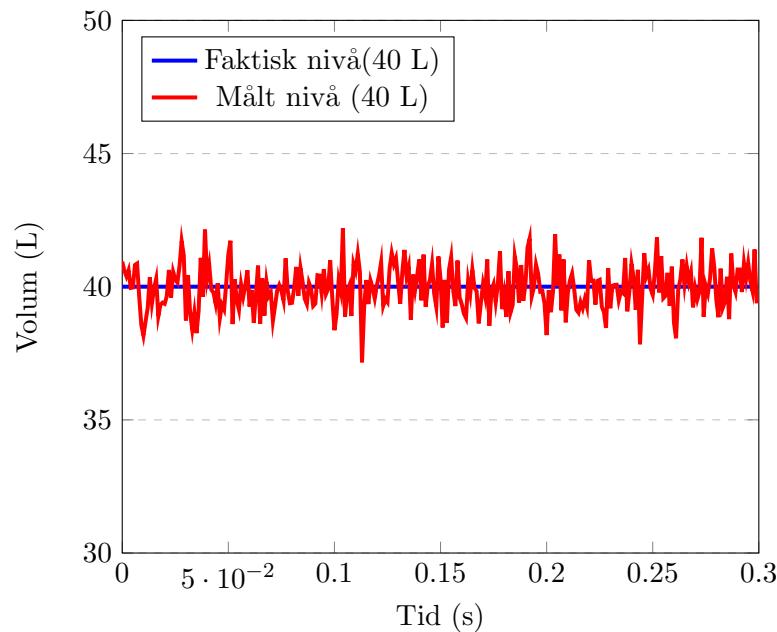
Deltakere:

Christoffer Askeli ENOKSEN
Henry EIKELAND
Hermann Wathne TVEIT

3.1 Innledning

Drivstoffmåleren i en bil er et godt eksempel på filtrering i praksis. Måleren viser et stabilt nivå som ikke er utsatt for momentan variasjon. Nivået i drivstofftanken blir oppfattet som stabilt, til tross for relativt store endringer i målesignalet. Målesignalet endres proporsjonalt med flottøren i drivstofftanken, som er utsatt for mekaniske forstyrrelser underveis i kjøringen.

For å oppnå denne stabiliteten har målesignalet gått gjennom en filtreringsprosess. Uten denne filtreringen ville operatøren av bilen blitt presentert et drivstoffnivå med betydelig større variasjon. Med andre ord: et ustabilt signal. Det er den praktiske løsningen av dette som er innhold i kapittel 3 - Filtrering.



Figur 10: Sammenligning mellom reelt nivå i en drivstofftank, mot det støybefengte målesignalet fra flottøren. 1000 Hz samplingsrate.

3.2 Problemstilling

Når en jobber med digitale signal vil disse alltid være begrenset til en viss oppløsning. I motsetning til et analogt signal hvor signalendringen skjer trinnløst, skjer endringen i digitale signal i binære steg (*0 og 1*).

En endring fra 0 til 1 skjer momentant. Eventuell bearbeiding av det digitale signalet kan bli sterkt påvirket av denne momentane endringen. For å minimere de momentane endringenes innvirkning på signalet, kan en ved hjelp av enkel matematikk *filtrere* signalet. (*sier at målestøyen filtreres bort*).

I dette kapittelet introduseres det to typer filter. Finite Impulse Response (FIR) og Infinite Impulse Response (IIR). De ulike filtrene respons skal undersøkes gjennom tre ulike eksperiment.¹

3.3 Matematisk Beskrivelse

Et Moving Average (MA) FIR filter er gitt av likning 22

$$y[k] = \frac{1}{M} \sum_{n=0}^{M-1} x[k-n] \quad (22)$$

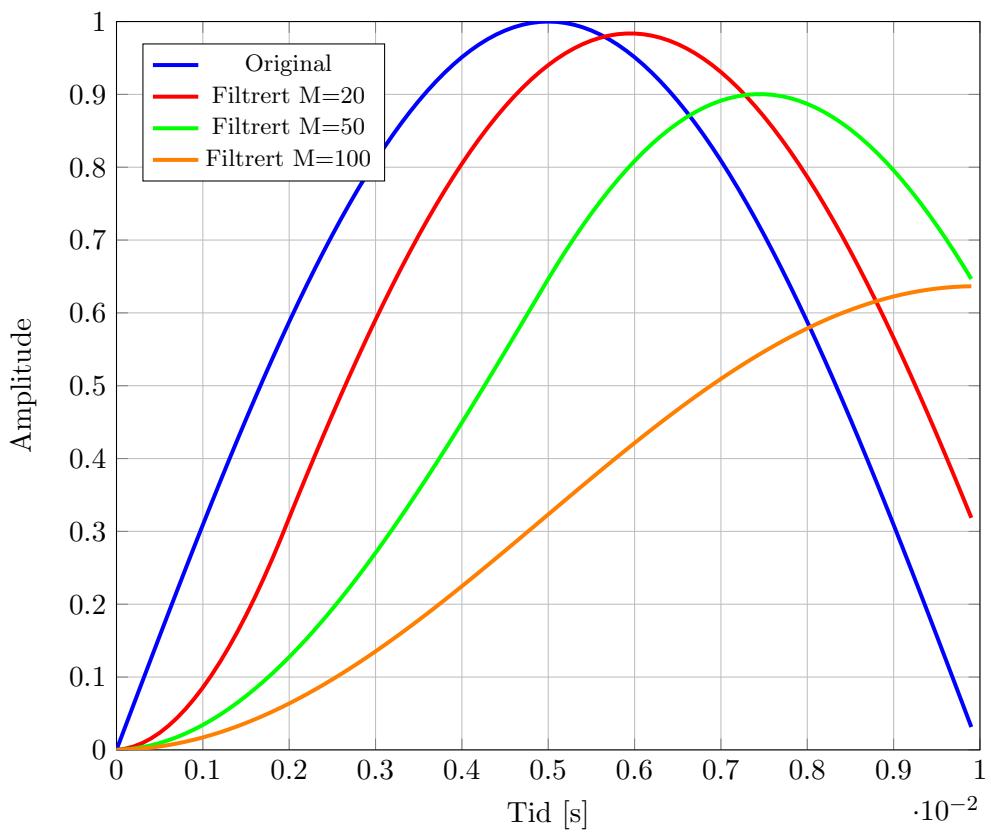
hvor $y[]$ er utgangssignalet, $x[]$ er inngangssignalet, M er antall verdier som blir brukt i gjennomsnittet, og k er antall iterasjoner.

I et 5-punkts MA FIR filter ($M = 5$) vil likningen for den sekstiende målingen se slik ut:

$$y[60] = \frac{x[60] + x[59] + x[58] + x[57] + x[56]}{5} \quad (23)$$

Likning 23 tydeliggjør funksjonen til konstanten M . Jo større M , desto flere målinger som påvirker gjennomsnittet. Siden n flere målinger påvirker $y[]$, vil responsen oppleves tregere jo høyere M -verdi. Se figur 11 for å se denne tregheten visualisert.

¹Eksperimentene er gjort med LEGO EV3 kontroller og tilhørende lyssensor. Lyssensoren har en samplingsrate på 1000 Hz, men antall målinger er begrenset av kommunikasjonen mellom PC og Kontroller. Typisk oppdateringsfrekvens er [50,100] Hz



Figur 11: Plott som viser dempingen som oppstår for ulike verdier av M. Samplingsrate 10kHz (100 målinger).

Et IIR filter er gitt av likning 24

$$y[k] = (1 - \alpha) \cdot y[k - 1] + \alpha \cdot x[k] \quad (24)$$

Hvor $y[]$ er utgangssignalet, $x[]$ er inngangssignalet og $\alpha [0, 1]$ er filtreringskonstanten (*vektingen*), som forteller oss hvor mye vekt det skal legges på inngangssignalet. α nær 1 vil gi en hurtig respons som etterligner responsen fra inngangssignalet. Motsatt vil α nær 0 gi en tungt filtrert, treg respons. Filtreringskonstantens innvirkning på utgangssignalet fremkommer av tabell 1 og 2.

Et IIR-filtrert signal med verdiene (α nær 1)

$$\begin{aligned} y[1 : 2] &= (5, 10) \\ x[3 : 5] &= (60, 60, 60) \\ \alpha &= 0.8 \end{aligned}$$

vil oppføre seg som i tabell 1:

Indeks	Filtrert verdi y	Råverdi x
1	5	60
2	10	60
3	50	60
4	58	60
5	59.6	60

Tabell 1: IIR-beregninger med stor α .

Et IIR-filtrert signal med verdiene (α nær 0)

$$\begin{aligned}y[1 : 2] &= (5, 10) \\x[3 : 5] &= (60, 60, 60) \\ \alpha &= 0.2\end{aligned}$$

vil oppføre seg som i tabell 2:

Indeks	Filtrert verdi y	Råverdi x
1	5	60
2	10	60
3	20	60
4	28	60
5	34.4	60
6	39.52	60
7	46.616	60
	...	
22	59.63	60

Tabell 2: IIR-beregninger med liten α . Store treheter i det filtrerte signalet.

3.4 MATLAB kode

For å implementere likningene i MATLAB må de skrives litt om. Hovedsaklig omhandler det håndtering av filterets oppførsel når det er lastet inn færre ufiltrerte målinger enn M-verdien til FIR-filteret ($k < M$), og når det er mindre enn én måling i IIR-filteret ($k < 1$). Denne problemstillingen er løst ved hjelp av en if-betingelse.

Kode 3: Kodeutdrag av FIR og IIR filtreringsberegnning.

```

1    % Konstanter
2    M = 3;           % Antall målinger i FIR-filteret
3    alpha_c = 0.6;   % Filterkonstant for IIR-filteret
4
5    % Oppstartsverdier %
6    Ts(1) = 0.005;
7    Temp(k) = Lys(k);
8    Temp_FIR(1) = Temp(1);
9    Temp_IIR(1) = Temp(1);
10
11   %% FIR-beregning %%
12   if k >= M
13       Temp_FIR(k) = (1/M) * sum(Temp(k-M+1:k));
14   else
15       M = k;
16   end
17
18   %% IIR-beregning %%
19
20   if k ~= 1
21       Temp_IIR(k) = (1 - alpha_c) * Temp_IIR(k - 1) + ...
22           alpha_c * Temp(k);

```

I linje 13 er det summen av M vektorelement i Temp vektoren som multipliseres med den inverse M-konstanten. Antall målinger som summeres er avhengig av M-konstanten.

Kode 4: Kodeutdrag av FIR-filter skrevet som Matlab funksjon

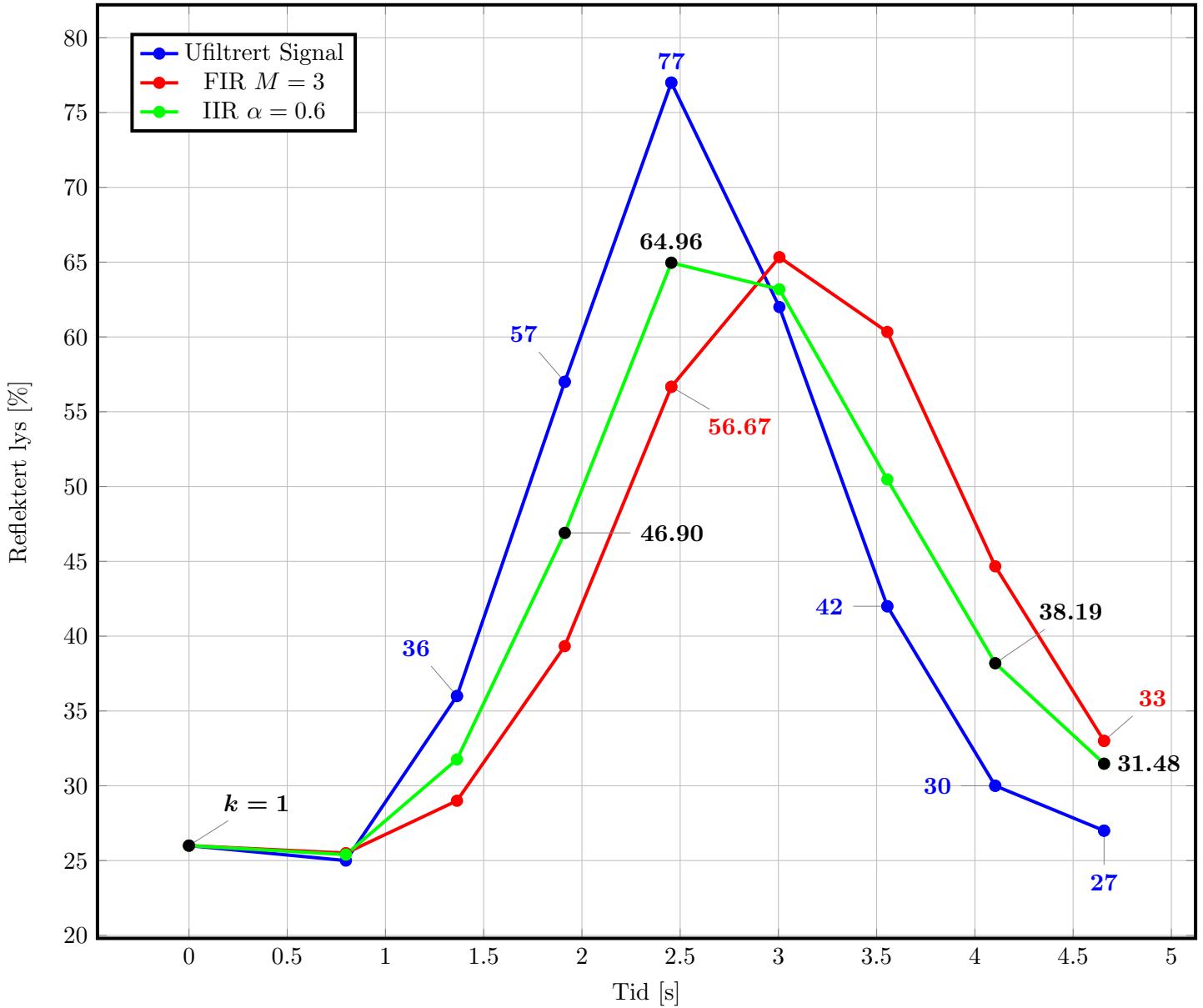
```
1 function [output_signal] = FIR_filter(input_signal, M)
2 k = length(input_signal); % Oppretter lokal tellevariabel
3 if k >= M
4     output_signal = (1/M) * sum(input_signal(k-M+1:k));
5 else
6     M = k;
7 end
```

Kode 5: Kodeutdrag av IIR-filter skrevet som Matlab funksjon

```
1 function [output_signal] = ...
    IIR_filter(OldFilteredValue,Measurements, alpha_c)
2 output_signal = (1-alpha_c)*OldFilteredValue+alpha_c*Measurements;
3 end
```

3.5 Verifisering av beregningene gjort i FIR- og IIR-filter

For å verifisere filterberegningene er det gjort et eksperiment med LEGO EV3 kontrolleren og lyssensoren tilkoplet. For å begrense målevektoren er det lagt inn brudd i koden når $k = 10$, derav 9 målinger. Vektor-rådata er hentet ut fra MATLAB, automatisk konvertert til (x,y) koordinatpar og tegnet med Tikz-pakken i figur 12.



Figur 12: Overlagret tidsgraf som fremstiller et ufiltrert-, FIR- og IIR-filtert signal med markerte aktuelle tallverdier brukte i verifikasjonsberegningene.

3.5.1 Verifikasjon av FIR-filter

$$\begin{aligned}
 y_{FIR}[5] &= \frac{x[5] + x[4] + x[3]}{3} \\
 &= \frac{77 + 57 + 36}{3} \\
 &= \underline{\mathbf{56.67}}
 \end{aligned} \tag{25}$$

$$\begin{aligned}
 y_{FIR}[9] &= \frac{x[9] + x[8] + x[7]}{3} \\
 &= \frac{27 + 30 + 42}{3} \\
 &= \underline{\mathbf{33}}
 \end{aligned} \tag{26}$$

FIR-filter beregninger med målinger hentet fra lyssensoren. Måling 5 og 9 ($k = 5, k = 9$)

3.5.2 Verifikasjon av IIR-filter

$$\begin{aligned}
 y_{IIR}[5] &= (1 - \alpha) \cdot y[5 - 1] + \alpha \cdot x[5] \\
 &= (1 - 0.6) \cdot y[4] + 0.6 \cdot x[5] \\
 &= 0.4 \cdot 46.9034 + 0.6 \cdot 77 \\
 &= \underline{\mathbf{64.96}}
 \end{aligned} \tag{27}$$

$$\begin{aligned}
 y_{IIR}[9] &= (1 - \alpha) \cdot y[9 - 1] + \alpha \cdot x[9] \\
 &= (1 - 0.6) \cdot y[8] + 0.6 \cdot x[9] \\
 &= 0.4 \cdot 38.1896 + 0.6 \cdot 27 \\
 &= \underline{\mathbf{31.48}}
 \end{aligned} \tag{28}$$

IIR-filter beregninger med målinger hentet fra lyssensoren.

De manuelle filterberegningsene utført i likning 25, 26, 27 og 28 er identiske med de kode-kalkulerte verdiene vist i figur 12, og gir derfor grunnlag til å påstå at koden er riktig implementert.

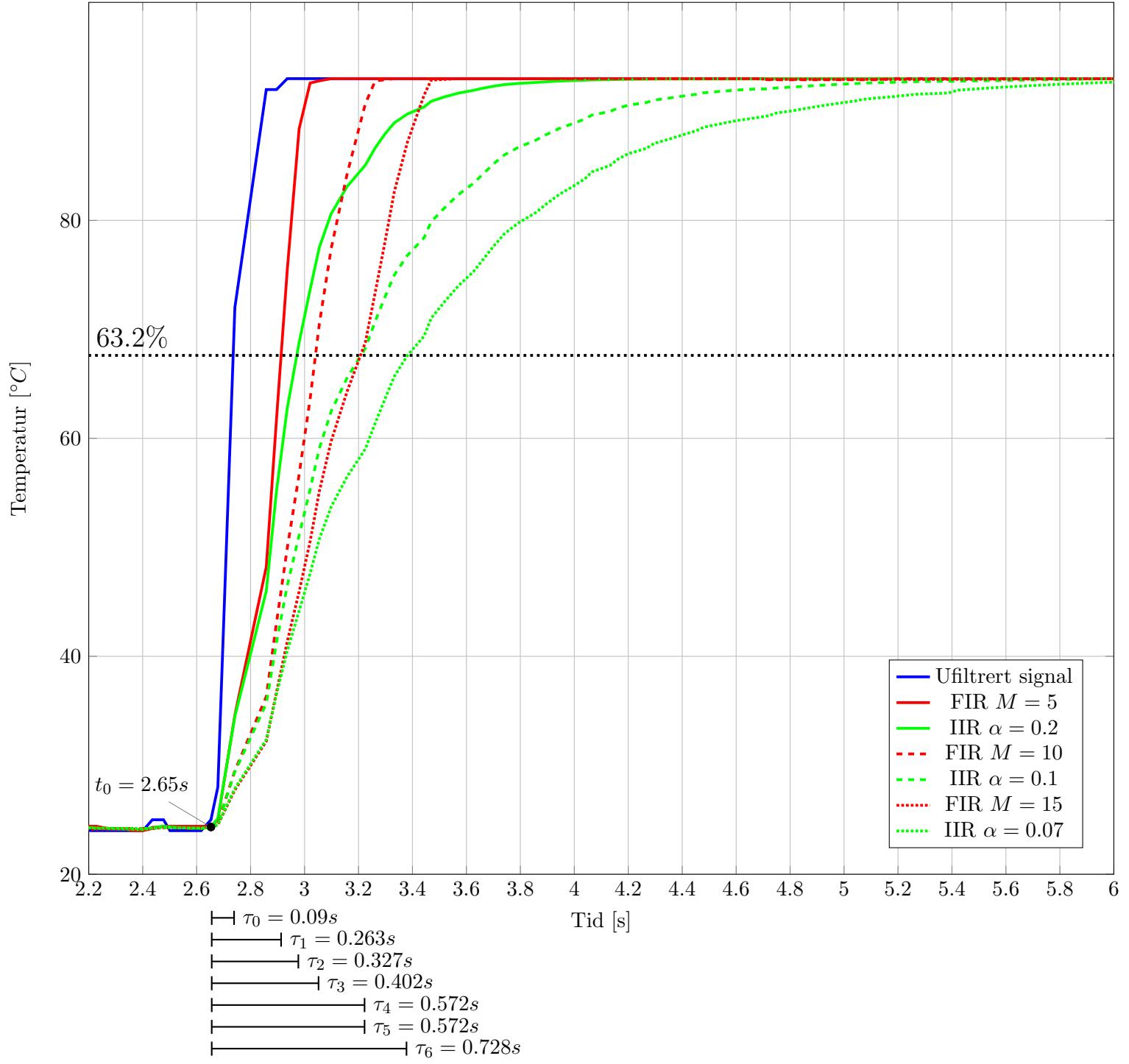
3.6 Filtrere målt kaffetemperatur

Lyssensoren, som sender verdier i området $[0,100]\%$ skal i dette forsøket tolkes som et termometer med måleområde $[0, 100]^\circ C$. Forsøket utføres for å tydelig gjøre filtrernes påvirkning på inngangssignalet.

Selv forsøket utføres ved at lyssensoren beveges fra et lite reflekterende felt (sort) over til et svært reflekterende felt (hvitt). Endringen i reflektert lys speiles over til en tolket endring i temperatur. På den måten er sluttresultatet enklere å relatere til.

Av figur 13 fremkommer det at responsen til de ulike filtrene blir betraktelig påvirket av filtreringskoeffisientene. Tidskonstanten τ tegner et bilde av responsen til det filtrerte signalet, men blir misvisende ved sammenlikning av de to filtertypene. Avlest i punkt $y = (1 - e^{-1}) \cdot 100 \Rightarrow 63.2\%$ ser man at det FIR-filtrerte signalet med $M = 15$ har identisk tidskonstant som det IIR-filtrerte signalet med filtreringskoeffisient $\alpha = 0.1$. Imidlertid bruker IIR-filteret 1.2 sekund lengre på å oppnå stasjonær temperatur.

Filtrert kaffetemperatur



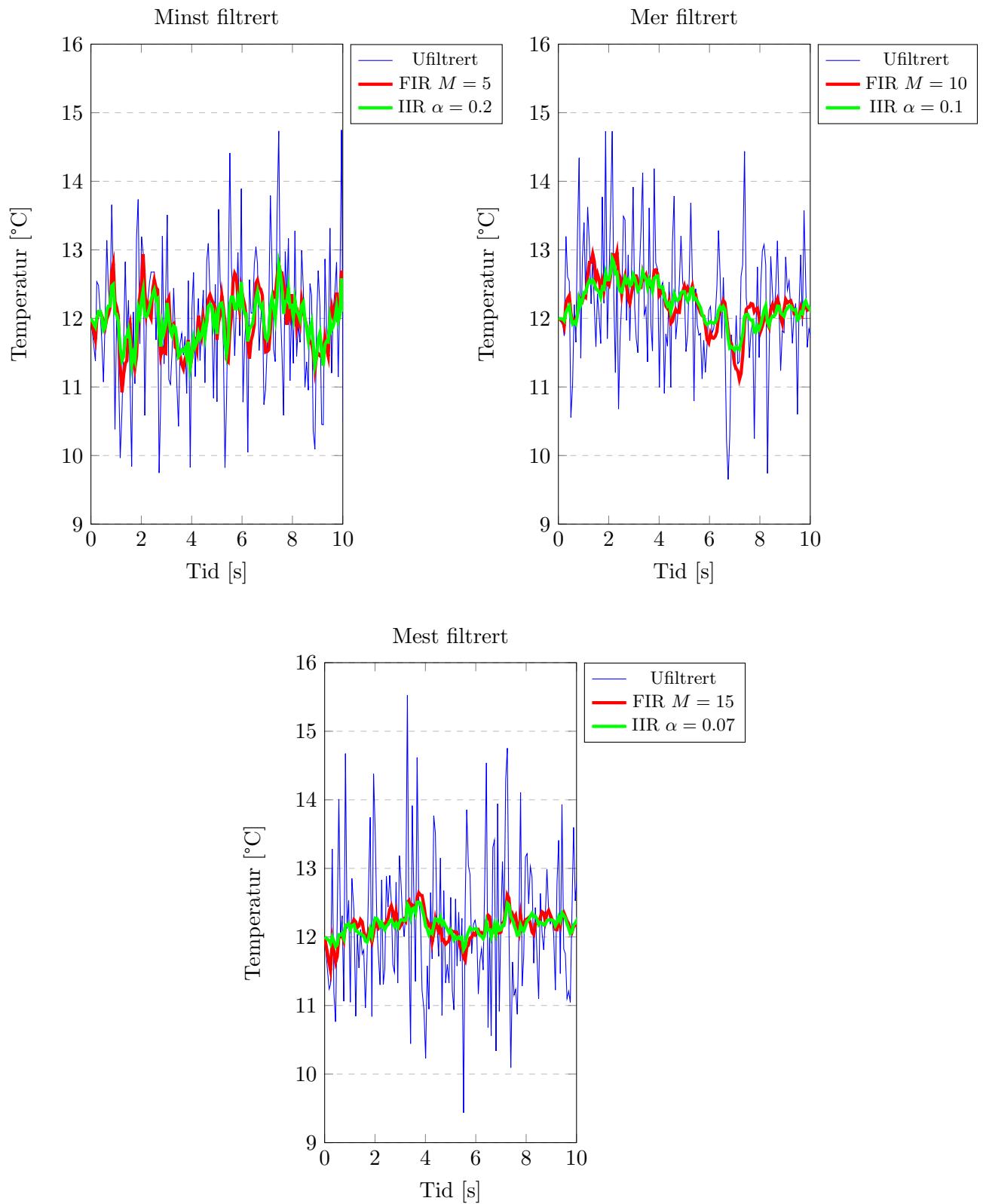
Figur 13: Overlagret tidsgraf med forskjellige filter-verdier. Ulike tidskonstanter τ presentert i sortert rekkefølge (iht. signaturforklaringen).

3.7 Filtrering av målestøy i en industriell temperaturtransmitter

Temperaturtransmittere kan være utsatt for momentane temperaturendringer som egentlig er neglisjerbare, men som allikevel kan virke forstyrrende på kode som benytter seg av målingen. I tillegg kan disse momentane endringene (*ofte kalt støy*) virke forstyrrende på operatøren, da gjerne når målingene er fremstilt i et grafisk grensesnitt Human Machine Interface (HMI). For å dempe effekten av støyen, kan målingene filtreres, på samme måte som vist tidligere i kapittel 3.

I dette forsøket legges det tilfeldig støy på lyssensor-signalet. Videre FIR- og IIR-filtreres signalet med 3 ulike verdier for M og α . Resultatene presenteres i figur 14.

Av figur 14 ser man nyttigheten av å filtrere støyutsatte temperaturmålinger. I den mest filtrerte figuren ($\alpha = 0.07, M = 15$) ser man den filtrerte temperaturen ligner mer på en gjennomsnittlig temperaturmåling over tid.



Figur 14: Grafer som viser IIR- og FIR-filterets påvirkning på et støyende temperaturtransmittersignal.

4 Numerisk derivasjon

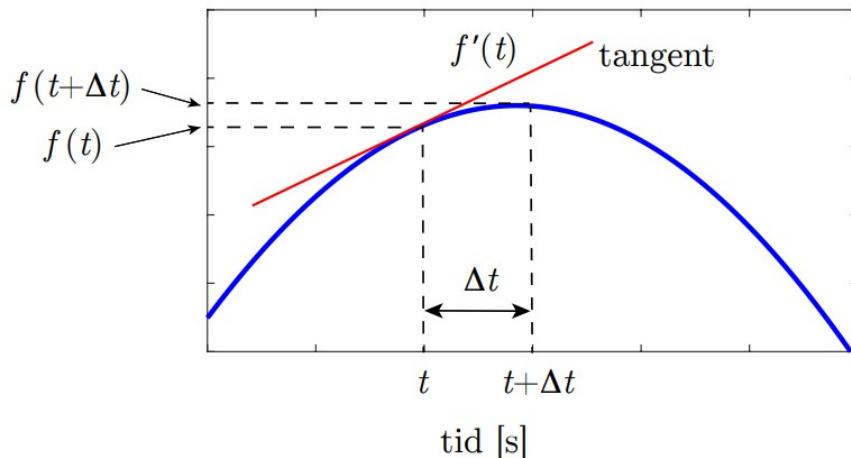
Deltakere:

Christoffer Askeli ENOKSEN
Henry EIKELAND
Hermann Wathne TVEIT

4.1 Innledning

For å vise til et derivasjons eksempel i hverdagen, kan man ta for seg hastighetsmåleren som politiet bruker i fartskontroller. Denne måleren gjør kalkulasjoner av den derivate i forhold til målt avstand. I prinsippet når denne måleren rettes mot en bil i fart, er det forskjellen på avstand over tid den registrerer. Høy endring i avstanden ift tid, gjør at den derivate øker som betyr en høyere fart.

Man er interessert i stigningstallet. Ved derivering av kontinuerlige funksjoner finner man stigningstallet i form av en tangenten til et punkt slik som figur 15 viser.



Figur 15: Den derivate av en kontinuerlig funksjon, hvor den rød tangenten viser stigningstallet. [1]

Den matematiske formelen for den derivate er

$$f'(t) = \lim_{\Delta t \rightarrow 0} \frac{f(t + \Delta t) - f(t)}{\Delta t} \quad (29)$$

Ved å la Δt gå så nær mot null uten at den går helt til null, finner man da tangenten i et punkt til den kontinuerlige funksjonen. Dette kapittelet går nærmere inn på hvordan man finner stigningstallet i praksis .

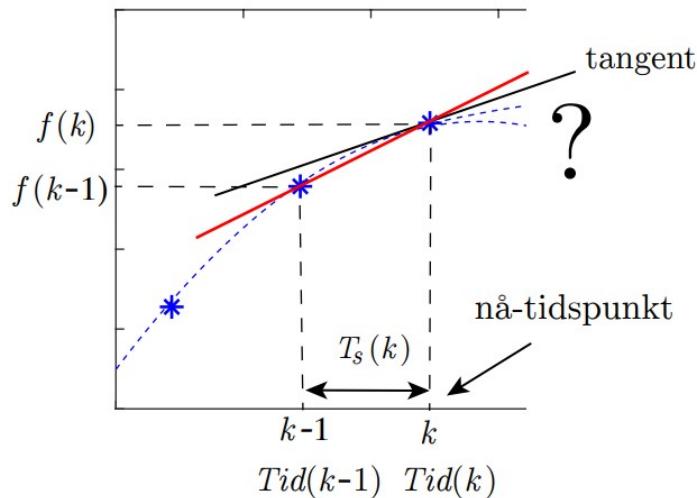
4.2 Problemstilling

Hensikten med dette prosjektet er å kunne bruke metoden for numerisk derivasjon og vise hvordan denne opptrer i praksis. Hvordan man samler inn målinger fra LEGO EV3-en, og hvordan man bruker disse dataene til å kontinuerlig regne ut stigningstallet når programmet kjører.

I matematiske funksjoner finner man stigningstallet i et bestemt punkt, dette er fordi funksjonen består av kontinuerlige variabler. Ved numerisk-derivasjon bruker man diskret variabler. Dette gjør at man ikke kan la Δt gå så nær som null slikt som formel 29 viser. Man må basere seg på de variablene som er tilgjengelige, og finne stigingstallet mellom disse.

4.3 Løsningsforslag

EV3-en registrerer verdier fra lyssensoren. De registrerte målingene har en verdi fra 0 til 100 etter hvor mye lys som blir reflektert tilbake til sensoren. I dette delforsøket tolkes disse verdiene som et mål på lest avstand. Målingene blir lagret i en datavektor for reflektert lys, og tidspunktet for målingen blir også lagret i en tidsvektor. Ved numerisk derivasjon finner man stigningstallet mellom disse diskret punktene. Figur 16 viser stigningstallet mellom målepunkter.



Figur 16: Rød sekant viser stigningstall mellom målepunkter. $Tid(k)$ viser tid i nåtid, og $Tid(k - 1)$ viser forrige verdi i tidsvektoren. Tidsskrittet, $T_s(k)$, er differansen imellom disse. [1]

Det finnes flere metoder for å numerisk derivere. I dette delforsøket brukes metoden bakoverderivasjon, og variabler brukt her kan beskrives som:

$$Fart(k) = \frac{Avstand(k) - Avstand(k - 1)}{T_s(k)} \quad (30)$$

Fra figur 16 er $f(k)$ verdien i avstandsvektoren ved gitt indeks k , som også defineres som siste måling. Differansen divideres med tidsskrittet (formel 31), mellom målingene. Resultatet av den numerisk deriverte legges inn i en ny vektor $fart(k)$.

For å beregne tidsskritt bruker man formelen:

$$T_s(k) = Tid(k) - Tid(k - 1) \quad (31)$$

Hvor tidsskrittet tilsvarer differansen mellom tidspunkt målingene ble registrert. Denne er beskrevet i kapittel 2 om numerisk integrasjon.

4.4 Matlab kode

I implementering av koden til numerisk-derivasjon tar man inn rådata-verdier fra lyssensoren. Disse verdiene legges inn i vektor $Avstand(k)$. Linje 7-13 definerer initialverdier til programmet. Her blir variablene for fart og filtert fart satt til null. Initialverdien til filtrert avstand tilsvarer første måling fra avstandsvektor. $Avstand(k)$ blir filtrert gjennom et IIR-filter i linje 18, som beskrevet i kapittel 3 Filtrering, hvor momentane endringer fra sensoren blir glattet ut. Tidsskrittet $Ts(k)$ regnes ut i linje 17. Den numeriske derivasjonen utført i linje 20 og 21. Her blir både rådata-verdiene og de filtrerte verdiene til avstand derivert ved bruk av formel 30, og tilordnet vektorene $Fart(k)$ og $Fart_IIR(k)$.

Kode 6: Kodeutdrag av Numerisk Derivasjon.

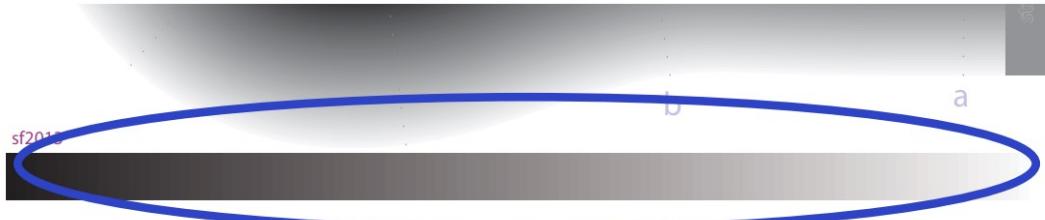
```

1 % Parametre
2 alfa = 0.03;
3
4 % Tilordne målinger til variabler
5 Avstand(k) = Lys(k);
6
7 % Spesifisering av initialverdier og beregninger
8 if k==1
9     % Initialverdier
10    Ts(k) = 0.05;
11    Fart(k) = 0;
12    Fart_IIR(k) = 0;
13    Avstand_IIR(k)=Avstand(k);
14
15 else
16     % Beregninger av Ts og variable som avhenger av ...
17     % initialverdi
18     Ts(k) = Tid(k) - Tid(k-1);
19     Avstand_IIR(k) = ...
20         (1-alfa)*Avstand_IIR(k-1)+alfa*Avstand(k);
21
22     Fart(k) = (Avstand(k) - Avstand(k-1))/Ts(k);
23     Fart_IIR(k) = (Avstand_IIR(k) - Avstand_IIR(k-1))/Ts(k);
24 end

```

4.5 Verifisering av koden for numerisk-derivasjon, del 1

Første verifisering av numerisk derivasjon går ut på å se om de beregnede verdiene samsvarer med hva som er forventet når det gjelder konstant hastighet. Ut fra eksempelet om hastighetsmålingen til politiet i innledningen, så skal reflektert lys fra lyssensoren til EV3-en tolkes som målt avstand fra bil til politiets hastighetsmåler. Figur 17 viser gråskala som ble brukt til denne testen. Her ble det laget et nokså lineært fallende datasett med verdier ved å bevege sensoren fra hvitt til mørkt område, hvor resultatet vises i figur 18.



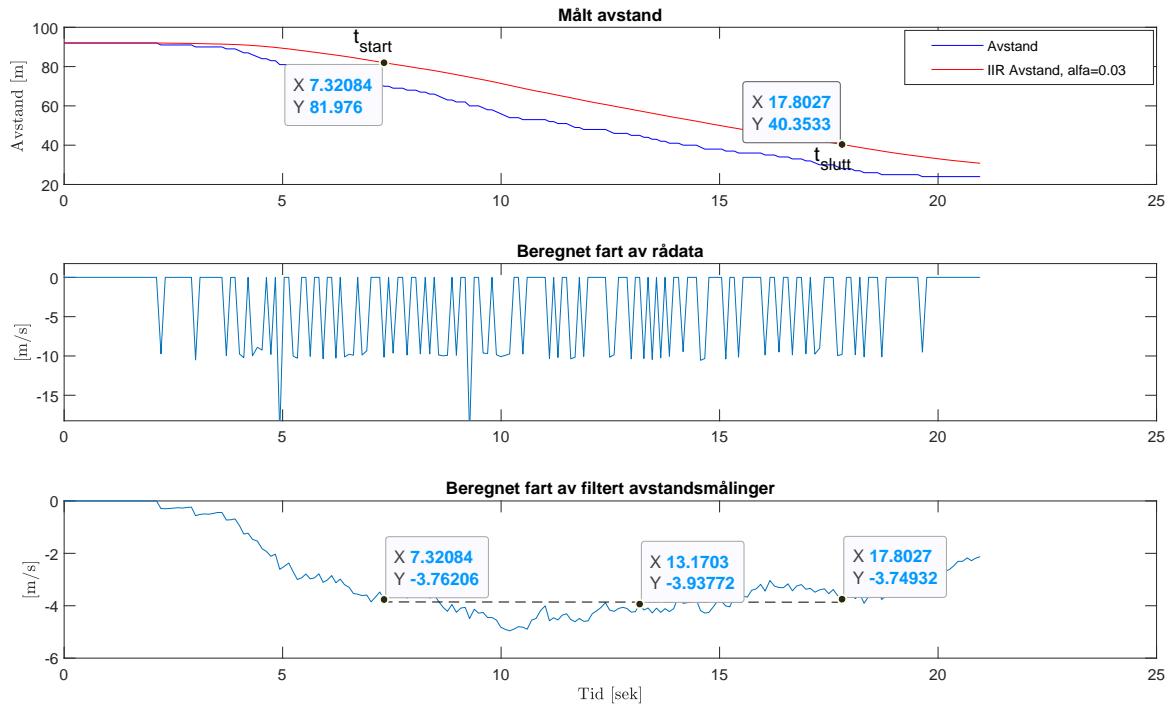
Figur 17: Gråskala til lyssensor for generering av avstandverdier. [1]

For å finne den øyeblikkelige farten må den registrerte lineære kurven deriveres ved bruk av formel 32, hvor farten er a i [m/s] i mot hastighetsmåleren. Denne formelen gjelder når avstandsendringen er konstant.

$$\frac{d}{dt}(a \cdot t) = a \quad (32)$$

For å få frem viktigheten av å filtrere, blir også avstandsdataene filtrert gjennom et IIR-filter, hvor alfa-parameteret er satt til 0.3. Dette vises i figur 18 i den øverste grafen, hvor begge datasett er plottet inn. Videre numerisk deriveres både rådata-avstand og filtrert avstand, som vist ved bruk av formel 30. Fra figuren kan man se forskjellig resultat hvor rådata-fart verdiene vises i midten og filtrert fart nederst.

Graf over fart beregnet fra rådata gir ikke noe klart bilde over farten til bilen. Den deriverete er følsom for plutselige endringer i målingen. Har man et støyfullt signal vil dette også påvirke den deriverete. Avstanden må filtreres først før den deriveres for å få en lesbar graf.



Figur 18: Resultat av hastighetsmåleren. Øverste graf viser rådata avstand fra lyssensor i blått, og den filtrerte IIR-avstanden i rødt. Midterste graf viser den deriverte av rådataene. Nederste graf viser den deriverte av den filtrerte IIR-avstandsdataen.

Beregning av gjennomsnittsfart ved konstant avstandsendring vises med formel 33. I denne beregningen brukes verdier fra den filtrerte avstandsmålingen i rødt fra øverste graf i figur 18.

$$\begin{aligned}
 Gjennomsnittsfart &= \frac{y_{slutt} - y_{start}}{x_{slutt} - x_{start}} \\
 Gjennomsnittsfart &= \frac{40.3533 - 81.976}{17.8027 - 7.32084} \quad (33) \\
 Gjennomsnittsfart &= \underline{-3.97} \text{ [m/s]}
 \end{aligned}$$

Fra nederste graf på figur 18 ser man flere punkter hvor farten indikert er ≈ -3.9 [m/s]. Til sammenligning viser den beregnede farten fra formel 33 en gjennomsnittsfart på -3.97 [m/s]. Denne viser som negativ siden avstanden minker mot hastighetsmåleren.

4.6 Verifisering av koden for numerisk derivasjon, del 2

Dette delkapittelet handler om derivasjon av en sinusfunksjon, gitt på formen(34):

$$y = b + a \cdot \sin(\omega \cdot t) \quad (34)$$

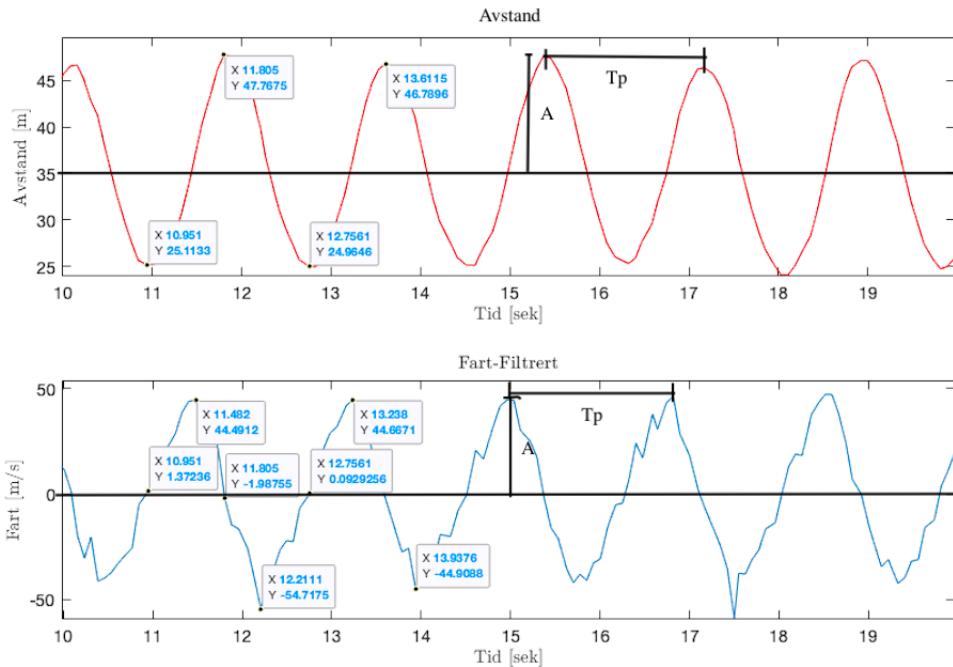
Deriveres uttrykket med hensyn på t ved hjelp av kjerneregelen, skrives det slik(35):

$$\frac{d}{dt}y = a\omega \cdot \cos(\omega \cdot t) \quad (35)$$

Verifisering av MATLAB-koden for numerisk derivasjon gjøres ved å utføre et eksperiment ved hjelp av LEGO EV3 og lyssensoren med støttekonstruksjoner satt opp på måten vist i figur 19. Målingene fra lyssensoren skal filtreres gjennom IIR-filteret fra kapittel 3. Koden verifiseres når det kan bekreftes om det er samsvar mellom resultatenes vinkelfart (ω) og periode (T_p), undersøkt ved likning 36 og 37 for hver delfigur i figur 20. For grundigere verifisering av koden skal det også utføres eksperimenter på samme måte som vist i kapittel 2, med ulike amplituder og vinkelfrekvenser, og til slutt et *chirp*-signal.



Figur 19: Oppsett av LEGO EV3 og lyssensor. Prinsippet bak oppsettet er avlesning av gråskalaverdier på en roterende plate.



Figur 20: Verifiseringsgrunnlag for koden i deloppgave 2, sinusfunksjon. Øverste delfigur viser den filtrerte målte avstanden(rød kurve), nederste delfigur viser den beregnede farten(blå kurve) ut i fra filtrerte målinger fra lyssensoren. T_p er perioden, A er amplituden.

Beregninger av "Avstand" (øverste delfigur, figur 20)

$$\begin{aligned}
 T_p &= \text{Toppunkt}(k) - \text{Toppunkt}(k-1) \\
 &= 13.61 - 11.81 \\
 &= 1.8[s]
 \end{aligned} \tag{36}$$

$$\begin{aligned}
 \omega &= \frac{2\pi}{T_p} \\
 &= 3.49[\text{rad}/\text{s}]
 \end{aligned} \tag{37}$$

Beregninger av Fart" (nederste delfigur, figur 20)

$$\begin{aligned}
 T_p &= \text{Toppunkt}(k) - \text{Toppunkt}(k-1) \\
 &= 13.24 - 11.48 \\
 &= 1.76[s]
 \end{aligned} \tag{38}$$

$$\begin{aligned}
 \omega &= \frac{2\pi}{T_p} \\
 &= 3.57[\text{rad}/\text{s}]
 \end{aligned} \tag{39}$$

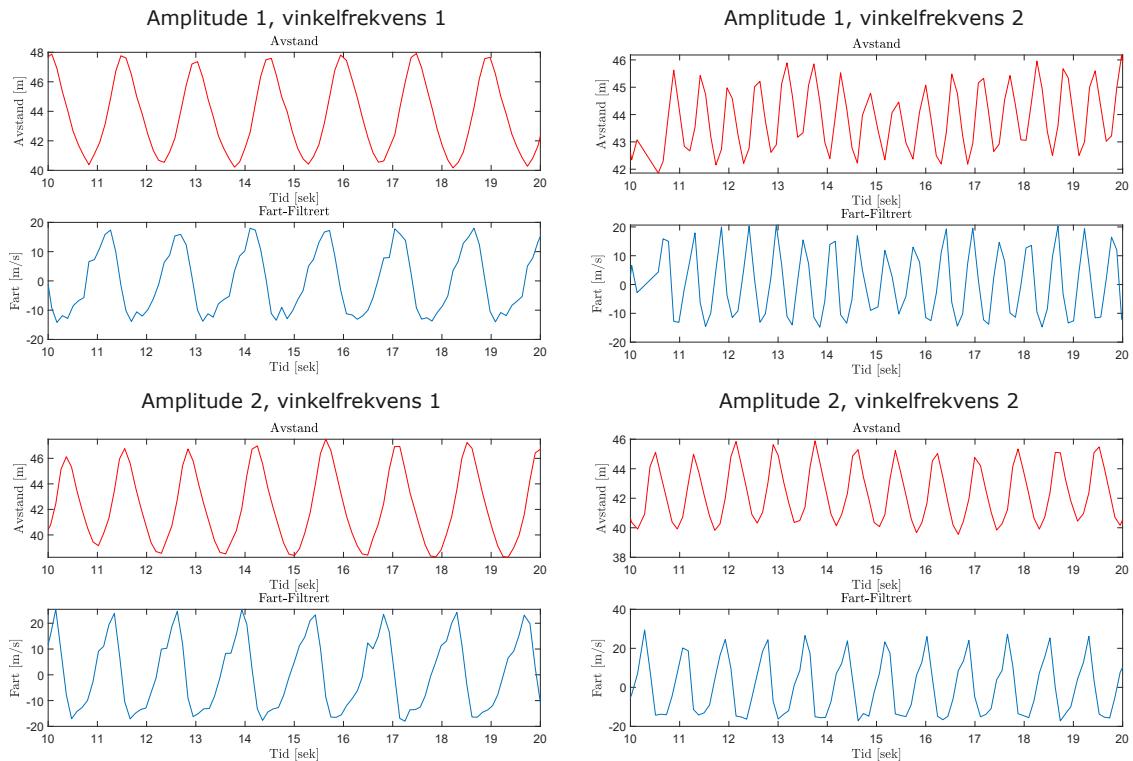
Basert på målingene fra lyssensoren og de beregnede verdiene for fart, vist i figur 20, samt beregninger for periode og vinkelfart ved likning (36,37) riktig

implementert. Én ser da etter samstemhet i kurvene for avstand og fart i figur 20 iht. periode og vinkelfart. Beregningene i likning (36, 37) ser ut til å samsvare relativt godt ($T_p = 1.8 \approx 1.76[s]$ og $\omega = 3.49 \approx 3.57[rad/s]$)

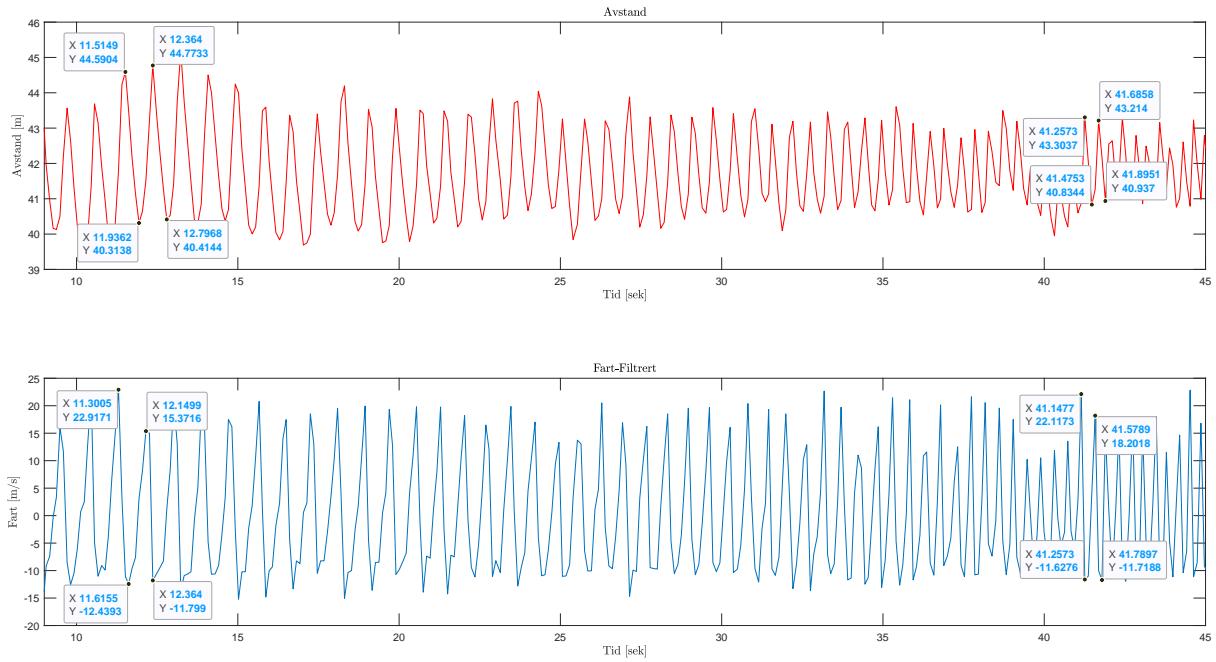
Legg også merke til hvordan tidspunktene for topp- og bunnpunktene i øverste delfigur samsvarer med tidspunktene i nederste delfigur hvor farten er ≈ 0 . I tillegg er det verdt å legge merke til at i figur 20 er også faseforskyvningen resultatet av derivasjon av \sin i likning 35.

4.6.1 Eksperimenter

Som i kapittel 2, utføres det flere ulike eksperimenter for å sikre et godt verifiseringsgrunnlag for den implementerte koden i MATLAB. Utførelsen av nevnte eksperimenter gjøres ved samme oppsett som vist i figur 19. Det skal undersøkes på hvor stor betydning amplitude og vinkelfrekvens har for den beregnede farten ut i fra IIR-filtrerte målinger av rådata.



Figur 21: Verifiseringsgrunnlag for kodens deloppgave 2, sinusfunksjon ved eksperiment. Samling av 4 eksperimenter med varierende amplitude og vinkelfrekvens. Øverste delfigur viser den filtrerte målte avstanden(rød kurve), nederste delfigur viser den beregnede farten(blå kurve) ut i fra filtrerte målinger fra lysensoren. Notasjonstallene 1 og 2 indikerer hvorvidt verdien er lav/høy(1=lav, 2=høy)



Figur 22: Verifiseringsgrunnlag for koden i deloppgave 2, sinusfunksjon ved eksperiment. Øverste delfigur viser den filtrerte målte avstanden(rød kurve) av et *chirp*-signal, nederste delfigur viser den beregnede fartens(blå kurve) ut fra filtrerte målinger fra lyssensoren av et *chirp*-signal.

Beregninger for alle eksperimenter (figur 21 og 22 er basis for tallverdier):

Periode T_p for avstandssignalet:

$$\begin{aligned} T_p &= \text{Toppunkt}(k) - \text{Toppunkt}(k-1) \\ &= 14.53 - 13.03 \\ &= 1.5[s] \end{aligned} \quad (40)$$

Amplitude A_{avlest} for avstandssignalet:

$$\begin{aligned} A_{avlest} &= \frac{y_{max} - y_{min}}{2} \\ &= \frac{47.3 - 40.2}{2} = \underline{3.55 [m]} \end{aligned} \quad (41)$$

Vinkelfrekvens ω for avstandssignalet:

$$\begin{aligned} \omega &= \frac{2\pi}{T_p} \\ &= \frac{2\pi}{1.5} \approx \underline{4.19 [rad/s]} \end{aligned} \quad (42)$$

Amplitude $A_{estimert}$ for estimert fart:

$$\begin{aligned} A_{estimert} &= A_{avlest} \cdot \omega \\ &= 3.55 \cdot 4.19 = \underline{14.87 [m/s]} \end{aligned} \quad (43)$$

Amplitude A_{avlest} for estimert fart:

$$\begin{aligned} A_{avlest} &= \frac{y_{max} - y_{min}}{2} \\ &= \frac{18.1 - (-14.2)}{2} = \underline{16.15 [m/s]} \end{aligned} \quad (44)$$

Faseforskyvning ϕ for avstandssignalet:

$$\begin{aligned} \phi &= \frac{\Delta t}{T_p} \cdot 360^\circ \\ &= \frac{(14.1 - 13.03)}{1.5} \cdot 360^\circ = 256.8^\circ \\ &= 256.8^\circ - 360^\circ = -103.2^\circ \end{aligned} \quad (45)$$

Avviksberegning:

$$\begin{aligned} Avvik &= \frac{A_{avlest} - A_{estimert}}{A_{estimert}} \cdot 100 \\ &= \frac{16.15 - 14.87}{14.87} \cdot 100 \\ &= 8.6[\%] \end{aligned} \quad (46)$$

Tabell 3: Tabellen viser alle resultatene som følge av beregninger gjort ved likning(40 til 47). Indeksene 1 og 2 svarer til lav/høy(verdi) respektivt. Fargekodene svarer til utslaget på avviket, hvor grønn er innenfor 10%, oransje og rødt er utenfor. Eksperimentnavnene svarer til hver enkelt delfigur i figur 21

Eksperiment	T_p [s]	ω [rad/s]	A_{avlest}	$A_{estimert}$	Avvik[%]	ϕ
A1 ω 1	1.5	4.19	16.15	14.87	8.6	-103.2
A1 ω 2	0.54	11.6	16.26	20.07	-19	-66.7
A2 ω 1	1.29	4.87	20.45	19.97	2.4	-61.4
A2 ω 2	0.85	7.39	21.85	19.21	13.7	-93.2
A2-chirp(<i>tidlig</i>)	0.85	7.39	17.65	15.81	11.6	-88.9
A2-chirp(<i>sen</i>)	0.43	14.61	16.88	18.04	-6.4	-83.7

Faseforskyvningen har negativt fortegn, noe som indikerer at forskyvningen går mot høyre, hvilket stemmer overens med kurven. Basert på resultatene i tabell 3 kan en observere at avlest og estimert amplitude for den derivate(farten) av avstanden samsvarer godt nok til å kunne verifisere koden i MATLAB. Avvik som kom frem under gjennomføringen av eksperimentene kan mest sannsynlig kobles til alfa-verdien til IIR-filteret og tidsskritt, da en kan observere at målingene blir mindre kontinuerlige ved høyere vinkelfrekvens. Eksperimentet med *chirp*-signal(vist i figur 22) leder en til å konkludere at amplituden til farten holder seg uendret. Det er kun observerbar endring i frekvensen til oscillasjonene som følge av økende vinkelfrekvens.

Alle eksperimenter tyder på at koden er verifisert.

4.7 Derivasjon som funksjon

For å minske kodebruken ved senere prosjekter, opprettes det en funksjon i MATLAB for derivasjon vist i kode 7:

Kode 7: Kodeutdrag av Derivasjon som funksjon.

```

1 function [Secant] = Derivation(FunctionValues , TimeStep)
2     Secant = (FunctionValues(end) - FunctionValues(end-1)) ...
2         / TimeStep(end);
3 end

```

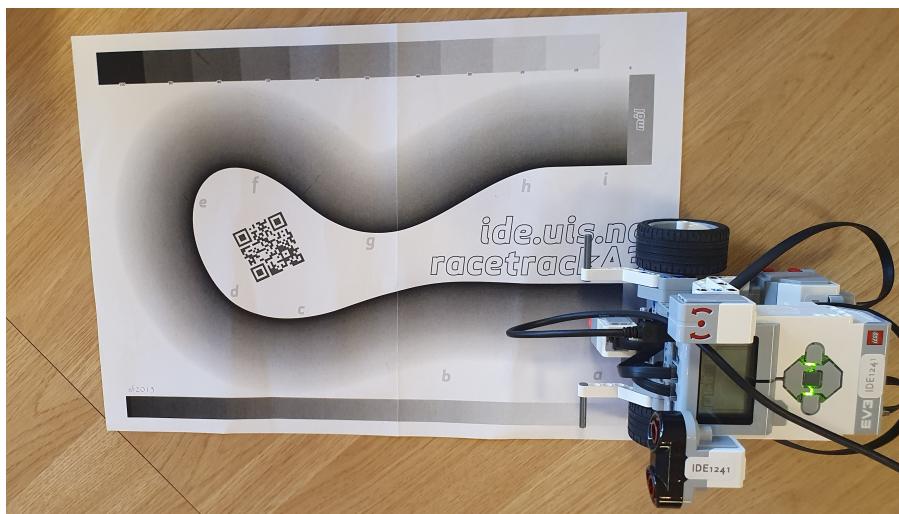
5 Manuell kjøring

Deltakere:

Christoffer Askeli ENOKSEN
Henry EIKELAND
Hermann Wathne TVEIT

5.1 Innledning

Dette delkapittelet betrakter bygging og kjøring av selve LEGO-roboten. Det ble lagt opp til en testkjøring hvor samtlige i prosjektgruppa har utført et forsøk på manuell kjøring gjennom banen som vist på figur 23. I tillegg til å programmere roboten for manuell styring, ble det også sett på hvordan man fra kjøringen kunne registrere ulike kvalitetsdata, og ut fra disse datane bestemme hvem som hadde den beste utførelsen.



Figur 23: LEGO-robot plassert i starten på gråskalabane. Målet er å manuelt kjøre gjennom banen med de beste kvalitetsdataene.

5.2 Problemstilling

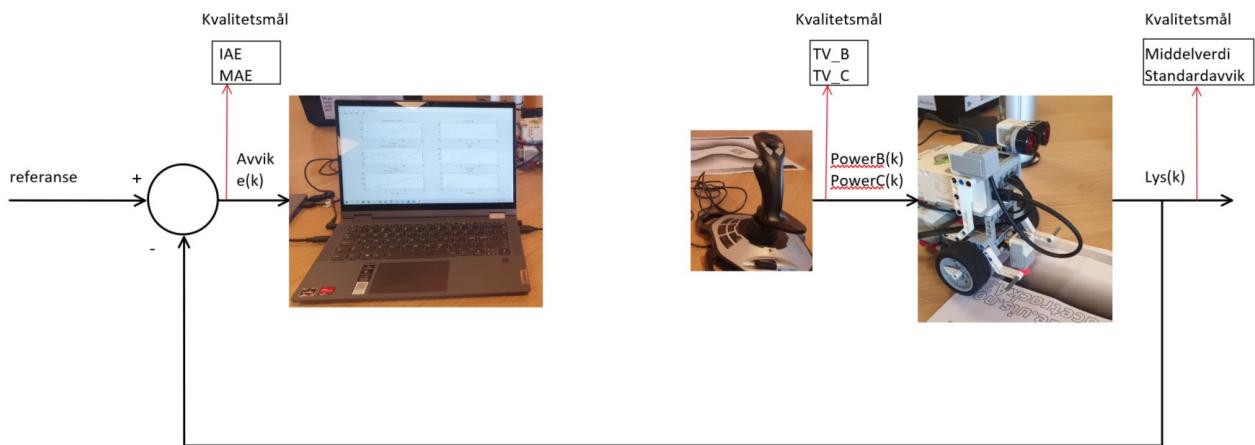
LEGO-roboten kjøres manuelt gjennom banen styrt ved hjelp av en styrespak. Roboten starter i midten av gråskala-feltet, som da blir referanseverdien. Målet er å kjøre langs banen med minst mulig avvik fra referanseverdien. Lyssensoren brukes til å måle reflektert lys i gråskalafeltet underveis i kjøringen. Data fra lyssensor og styrespak brukes så til beregninger av ulike kvalitetsmål for å videre presentere beste resultat. I gruppen var det om å gjøre å få de beste samlede resultatene.

5.3 Løsningsforslag

For å bestemme hvem som kjørte best, er det en rekke kvalitetsmål som beregnes. Fra disse kan man for eksempel se hvem som har truffet referansepunktet best mulig, eller hvem som har hatt den mest vinglete kjøringen. Avviket fra referanseverdien kan beskrives matematisk som:

$$e(t) = Referanse - Lys(k) \quad (47)$$

I tillegg til dette avviket, bruker man pådraget til begge motorene og lyssensordata for å beregne de forskjellige kvalitetsmålene. Figur 24 viser en skisse av hvilke disse er og hvor i prosessen det blir hentet ut data. Hvert kvalitetsmål blir nærmere beskrevet i neste delkapittel.



Figur 24: Prinsippskisse for prosess over manuell styring av LEGO-robot. Her vises det også hvor det blir hentet ut data for de ulike kvalitetsmålene.

5.4 Kvalitetsmål

Vi bruker en rekke begrep for å vurdere kvaliteten til kjøringen, og beregningen skjer både underveis og etter endt kjøring. kvalitetsmålene som blir vurdert i dette delforsøket er:

- **Integral of Absolute Error (IAE)** er en metode som regner integralet av absoluttverdien til avviket $e(t)$, ved bruk av numerisk integrasjon. Det er en måte å se hvor mye feil som har oppstått i prosessen, og den beskrives med formel:

$$IAE = \int_0^t |e(\tau)| d\tau \quad (48)$$

Fordi avviket, $e(t)$, kan være positivt eller negativt, avhengig av verdien på lyssensoren i forhold til referansepunktet, vil man få feil resultat ved å bare integrere denne. De negative avvikene vil nulle ut de positive, og det kan i verste fall ende med at en perfekt kjøring egentlig er en vinglete

kjøring hvor verdiene har nullet hverandre ut. Derfor brukes absoluttverdien når man beregner IAE. Beregninger skjer underveis i kjøringen.

- **Mean Absolute Error (MAE)** er en annen metode å se hvor mye feil som har oppstått i prosessen, og har formelen:

$$MAE = \frac{1}{k} \sum_{n=1}^k |e(n)| \quad (49)$$

Også denne bruker absoluttverdien til avviket, $e(t)$. Og metoden for denne er summasjon, som betyr at den summerer opp avviket over tid og deler på antall målinger. Den leverer dermed det gjennomsnittlige avviket. Beregninger skjer underveis i kjøringen. Til forskjell fra IAE som baserer seg på numerisk integrasjon, vil ikke MAE bli ikke påvirket av tidsskrittet i prosessen.

- **Total Variation (TV)** er en metode å se hvor mye variasjon som har oppstått underveis i pådragene til prosessen og beskrives med den generelle formelen:

$$TV = \sum_{n=1}^k |u(n) - u(n-1)| \quad (50)$$

Her vises u som pådrag. I dette forsøket brukes to pådrag, *PowerB* og *PowerC*. TV viser hvor mye motorene har endret hastighet i form av fremdrift og styring.

- **Middelverdi** μ er et begrep for å definere kvalitetsmål av sensordata i et datasett. Her fra lysmålingene.

$$\mu = \frac{1}{k} \sum_{n=1}^k Lys(n) \quad (51)$$

Beregningen skjer etter endt kjøring, og denne summerer opp alle verdiene fra lysmålingen og deler på antall målinger.

- **Standardavvik** σ er et begrep for å definere spredingen i et datasett, med formel:

$$\sigma = \sqrt{\frac{1}{k} \sum_{n=1}^k (Lys(n) - \mu)^2} \quad (52)$$

Også standardavviket beregnes etter endt kjøring. Ved høy verdi, kan dette betraktes ved at prosessen har hatt store variasjoner målt fra sensoren. I dette prosjektet vil det si en som har hatt en vinglete kjøring.

5.5 MATLAB kode

Kode 8: Kodeutdrag av data fra sensor og styrespak.

```
1 % sensorer
2 Lys(k) = ...
3     double(readLightIntensity(myColorSensor, 'reflected'));
4
5 % Data fra styrestikke.
6 [JoyAxes,JoyButtons] = HentJoystickVerdier(joystick);
7 JoyMainSwitch = JoyButtons(1);
8 JoyForover(k) = JoyAxes(2);
9 Joyside(k) = JoyAxes(1);
```

Programmet tar inn data fra styrespak og lyssensor. Her vil styrespak gi ut akseverdier om den blir beveget for/bakover eller til siden. Lyssensoren vil måle reflektert lys slik det er beskrevet i de tidligere kapittelene.

Kode 9: Kodeutdrag av pådragsberegninger

```
1 % pådragsparametre
2 a=0.7;
3 b=0.3;
4 % Pådragsberegninger, PowerB=venstre hjul, PowerC = høyre ...
5 % hjul.
6 if JoyForover(k) >0
7     % fremdrift
8     PowerB(k) = a*JoyForover(k)+b*(Joyside(k));
9     PowerC(k) = a*JoyForover(k)-b*(Joyside(k));
10 else
11     % revers
12     PowerB(k) = a*JoyForover(k)-b*(Joyside(k));
13     PowerC(k) = a*JoyForover(k)+b*(Joyside(k));
14 end
```

For å gjøre styringen fra styrespak mindre følsom, er det lagt inn pådragsparametre, a og b . Disse påvirker hvor mye kraft som gis til motorene under kjøring. $PowerB$ er drift på venstre hjul og $PowerC$ høyre hjul.

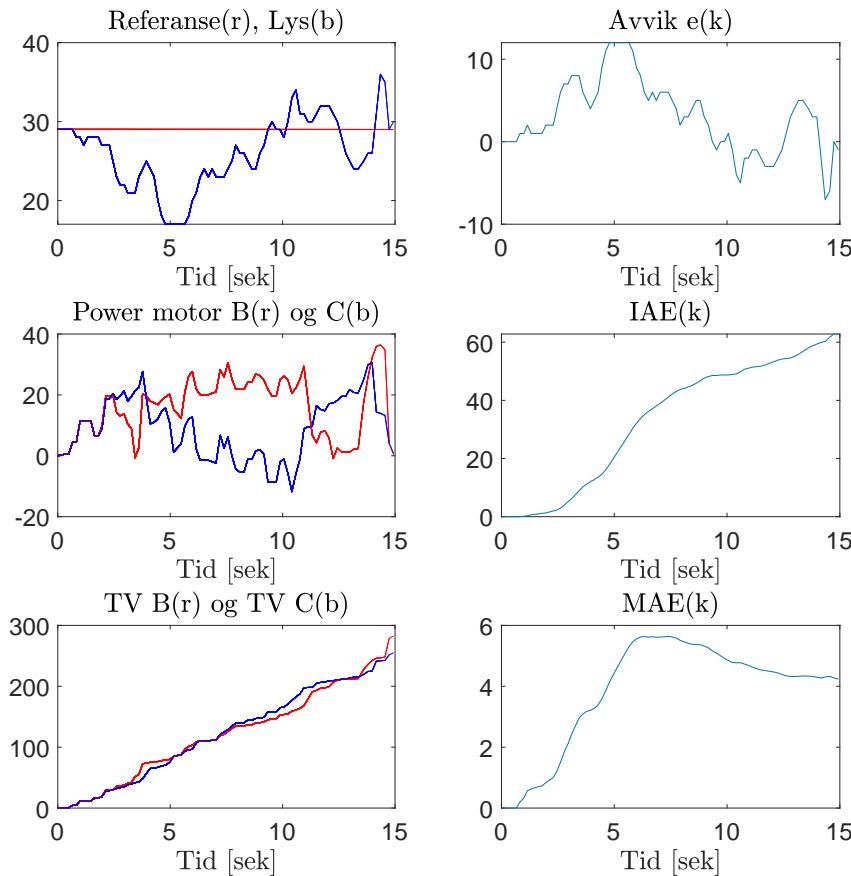
Kode 10: Kodeutdrag av beregninger til kvalitetsmål

```
1 % IAE og MAE kvalitetsberegnning
2 IAE(k) = EulerForward(IAE(k-1),abs(e(k-1)),Ts(k));
3 MAE(k) = (1/k)*sum(abs(e));
4
5 % mellomberegning variasjon motorB og MotorC
6 V_B(k) = abs(PowerB(k)-PowerB(k-1));
7 V_C(k) = abs(PowerC(k)-PowerC(k-1));
8
9 % Total Variasjon
10 TV_B(k) = sum(abs(V_B));
11 TV_C(k) = sum(abs(V_C));
12
13 % kvalitetsberegnning etter endt kjøring
14 middelverdi = (1/k)*sum(Lys); % Middelverdi Lyssensor
15 standardavvik = std(Lys); % Standardavvik Lyssensor
```

Beregningene IAE, MAE og TV blir gjort underveis. IAE(k) kaller på Euler-forward funksjonen i linje 2 som numerisk integrerer absoluttverdien av avviket $e(k)$. MAE(k) summerer underveis opp avviket $e(k)$ og deler denne på antall k . Middelverdi og standardavvik beregnes også underveis, men disse gir bare ut siste verdi i kjøringen siden disse ikke er satt opp som vektorer.

5.6 Resultat

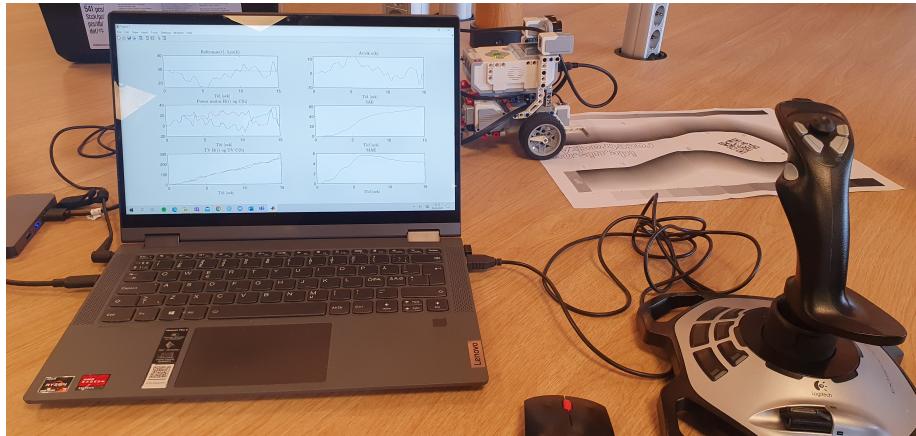
Etter endt kjøring samles alle dataene og beregningene for de ulike kvalitetsmålene. På figur 25 er det fremvist ulike grafer for mål av referanse, avvik, IAE, MAE og TV. Også pådraget til hver motor lagt ved. Dette er et utførelsen til én på gruppas.



Figur 25: Resultat av kjøring til én på gruppas. Fra toppen på venstre side vises; referanseverdien + lysverdi, pådrag til motorene i midten og TV nederst. Fra toppen på høyre side vises; Avvik(k), IAE(k) i midten og MAE(k) nederst.

Avvik $e(k)$ ser man følger formen til det omvendte av lysmålingen, hvor det har vært et større avvik rundt 5 sek, i motsetning til etter 10 sek. Som en konsekvens av det, er IAE brattere på tiden ved 5 sek enn etter 10 sekunder. MEA avtar etter rundt 10 sekunder, fordi avviket har en mindre verdi og det gjør at summasjon delt på antall målinger også avtar slikt som grafen viser. Den totale variasjonen er nokså likt økende på begge motorene, dette viser at det har vært et nok så stabil kjøring gjennom banen.

Grafene som viser pådrag til motorene, kan man se at varierer nokså likt fra intervallet [5,10], men at motorB stiger og motorC minker. Siden motorB er pådrag til venstre hjul, som beskrevet i kodeutdrag om pådragsberegninger, vil LEGO-roboten svinge motsatt til høyre i dette intervallet. Som betyr at roboten i dette intervallet mest sannsynlig befinner seg i den store svingen på banen. Av figur 26 ser man oppsettet til kjøringen og PC-skjermen som viser logging av data og beregning av kvalitetsdataene. Disse blir fremstilt i tabell 4.

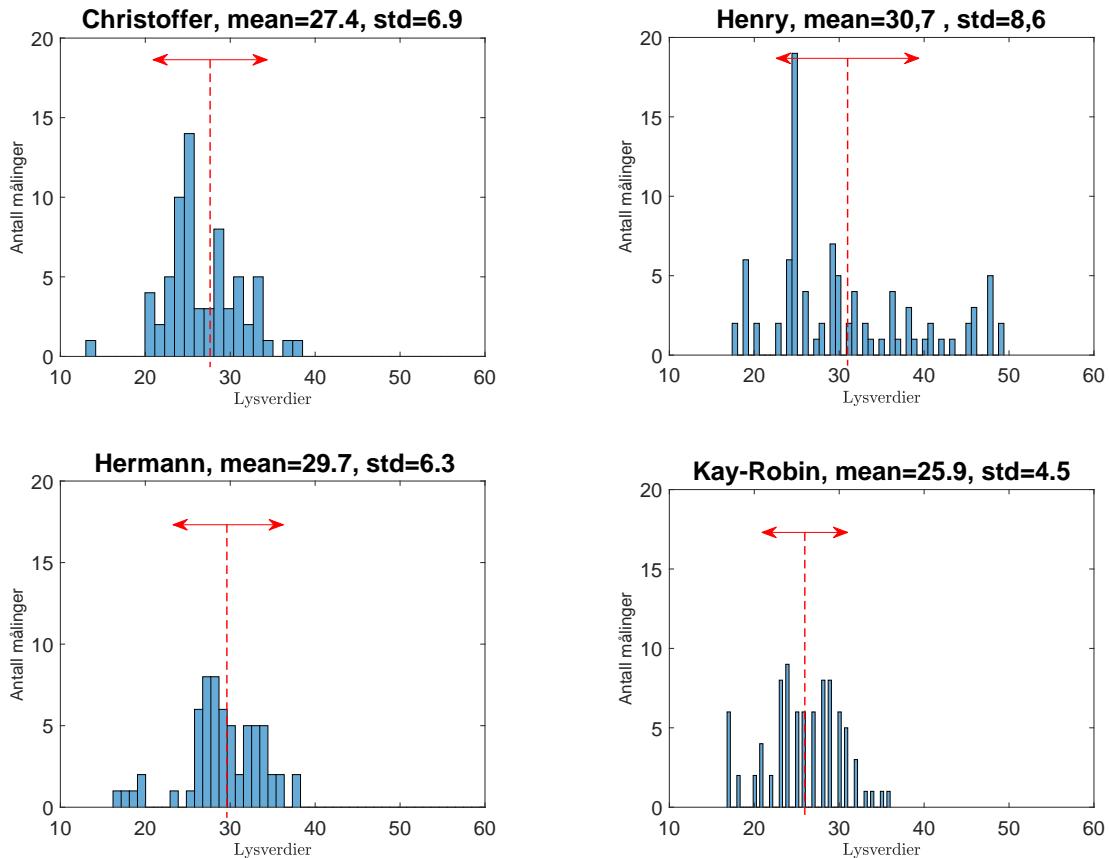


Figur 26: LEGO-robot plassert i starten på gråskala bane. Bilde viser oppsettet til den manuelle kjøringen med PC og styrespak.

	Christoffer	Henry	Hermann	Kay-Robin
Plattform	pc	pc	pc	pc
Strømkilde	230v	230v	230v	230v
Samtidig plotting	ja	ja	ja	ja
Referanse	25	25	28	29
Middelverdi μ	27,4	30,7	29,7	25,9
$ referanse - \mu $	2,4	5,7	1,7	3,1
Standardavvik σ	6,9	8,6	6,3	4,5
Kjøretid [s]	11,9	16,3	11,3	14,9
IAE	41,6	121,3	39,7	62,8
MAE	4,01	7,25	4,02	4,24
TV B	92	260	329	283
TV C	167	333	340	255
Middelverdi av Ts [s]	0,173	0,177	0,177	0,176
Antall målinger(k)	70	93	65	86

Tabell 4: Resultat manuell kjøring gjennom gråskalabane fra hver gruppemedlem. Beste verdier er markert i fet skrift. Alle har kjørt gjennom banen med samme forutsetninger for å ha best mulig grunnlag for sammenligning. IAE- og MAE-verdiene til Christoffer og Hermann nokså like lave. Når det gjelder TV, så har Christoffer lavere verdier, noe som tyder på en roligere kjøring.

Videre så viser figur 27 hvordan hvert gruppemedlem utførte forsøket av manuell kjøring gjennom banen. Disse dataene tar utgangspunkt i hvordan samlingen av lysensordataene har vært. Diagrammene her er satt opp med like akser for sammenligning. Som det kommer frem av de ulike fordelingene, er det ulike samlinger av lysverdier og standardavvik etter kjøringen. Ut ifra disse dataene så har fordelingen ned til høyre den med minst mulig standardavvik, som tyder på liten variasjon rundt referansepunktet.



Figur 27: Fordelingen viser resultatet til hvert gruppemedlem. Det vises her antall lysmålinger registrert under forsøket. For hver fordeling er det lagt inn middelverdi og standardavvik øverst i delfiguren. Middelverdi vises i tillegg som stiplet linje og standardavvik som pil.

Av de ulike kvalitetsmålene, er det vanskelig å score bra på alle sammen. F.eks så virker IAE og TV proporsjonalt av hverandre. For å bedømme hvilke kvalitetsmål man tar mer hensyn til så ser man på systemets helhet. Målet med dette forsøket var å ha minst mulig avvik, dermed verdsettes IAE og MAE høyere. Beste kjøring ble dermed utført av Hermann.

6 Estimering av hastighet

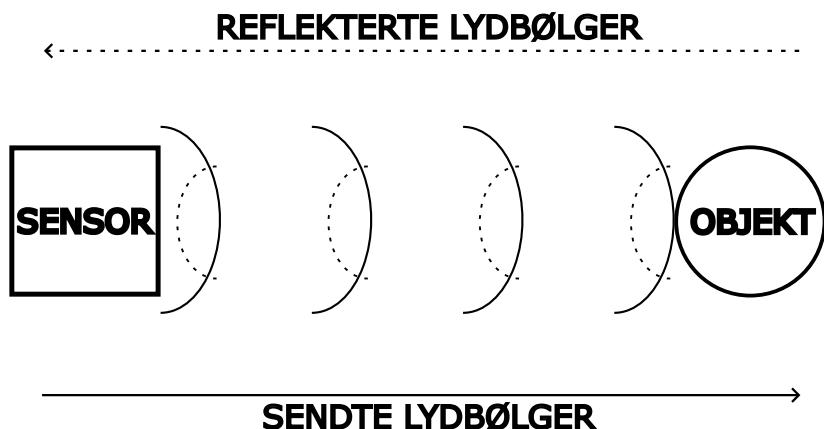
Deltakere:

Christoffer Askeli ENOKSEN
Henry EIKELAND
Hermann Wathne TVEIT

6.1 Innledning

Ultralydgivere er en type måleinstrument som ved hjelp av ultralydbølger er i stand til si noe om avstanden frem til et objekt. Ultralyd er definert som en frekvens høyere enn den mennesket kan oppfatte, nominelt 20 kHz. [2] Ved å sende ut en rekke lydbølger - og måle tiden det tar fra lydpulsen sendes, reflekteres og mottas, kan avstanden fra det reflekterende objektet presist kalkuleres. Figur 28 viser en forenklet prinsippskisse over driften til en ultralydgiver.

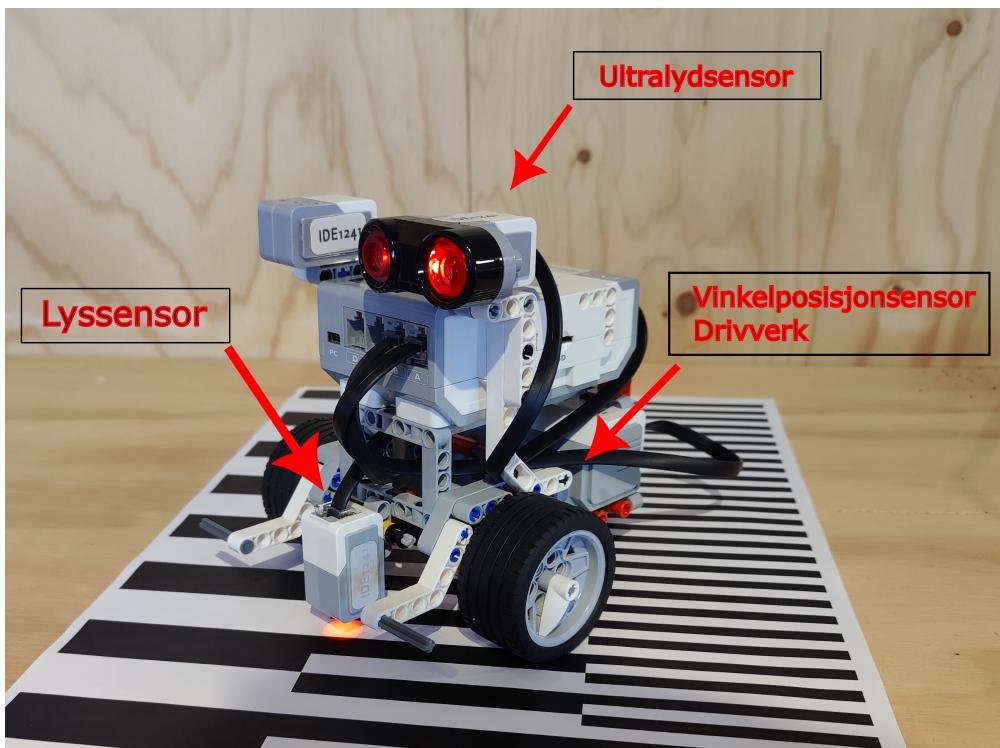
I industrien blir ultralydgivere hyppig brukt til å måle endring i væskenivå, endring i volumstrøm og andre avstandsmålinger (*f.eks ekkolodd*). Siden ultralydgiveren sier noe om avstanden til et objekt til en hver tid (*begrenset av samplingsraten*), kan måleverdien matematisk deriveres, og hastighet estimeres. Det er den praktiske løsningen av dette som er innhold i kapittel 6 - Estimering av hastighet.



Figur 28: Prinsippskisse for ultralydgiver. Sendte lydbølger i helstrukken linje, og mottatte (*reflekterte*) lydbølger i mindre, stiplet linje. Sensoren inneholder både sender og mottaker.

6.2 Problemstilling

LEGO-roboten er utrustet med servomotorer som gir tilbakemelding om vinkelposisjon, ved å derivere trilleavstanden $x(k)$ i likning 53 får man den reelle hastigheten som roboten beveger seg. Som vist i figur 29 er roboten i tillegg utrustet med en ultralydgiver og en lyssensor. Hastigheten til roboten skal individuelt estimeres ved hjelp av målinger hentet ut fra disse to sensorene. Ved å grafisk sammenligne den estimerte hastigheten mot den reelle hastigheten, kan vi verifisere om koden er riktig implementert.



Figur 29: Oversiktsfotografi av LEGO-robot med tilhørende sensorer. Det stripte underlaget er brukt til å estimere fart ved hjelp av lyssensoren.

6.3 Løsningsforslag

Vi skal i denne delen presentere hvordan vi med utgangspunkt i vinkelposisjonen til hjulene på EV3 roboten beregnet forflyttet strekning.

Reell hastighet kan da kalkuleres som:

$$x(k) = \frac{0.176m \cdot \theta(t)}{360}$$

$$v(k) = \frac{x_k - x_{k-1}}{t_k - t_{k-1}} \quad (53)$$

hvor hjulet til roboten har omkrets $0.176m$, θ er vinkelposisjonen i grader, og en omdreining av hjulet svarer til 360° .

Kode 11: MATLAB-kodeutdrag for beregning av strekning og hastighet ved vinkelposisjon

```
1 % Parametre
2 hjulomkrets = 0.176;
3
4 % Vinkelposisjonberegning
5 trip_teller(k) = 1/360 * hjulomkrets * VinkelPosMotorA(k);
6
7 % Derivasjon av vinkelposisjonen
8 deriv_vinkelpos(k) = Derivation(trip_teller(k-1:k), Ts(k));
```

Hovedelement fra kodeutdraget:

- **trip_teller(k)** i linje 5 er beveget strekning.
- **deriv_vinkelpos(k)** i linje 8 er beregnet hastighet
- **Derivation** i linje 8 er derivasjonsfunksjon fra kodeutdrag 7

6.3.1 Estimering av hastighet fra ultralydmålinger

Estimert hastighet ved derivert IIR-filtrert avstandsmåling kalkuleres som:

$$v(k) = \frac{x_k - x_{k-1}}{t_k - t_{k-1}} \quad (54)$$

hvor $x(k)$ er det IIR-filtrerte ultralydsignalet i meter.

Kode 12: MATLAB-kodeutdrag for ultralyd-estimert hastighet med derivasjons funksjon fra Kode 7

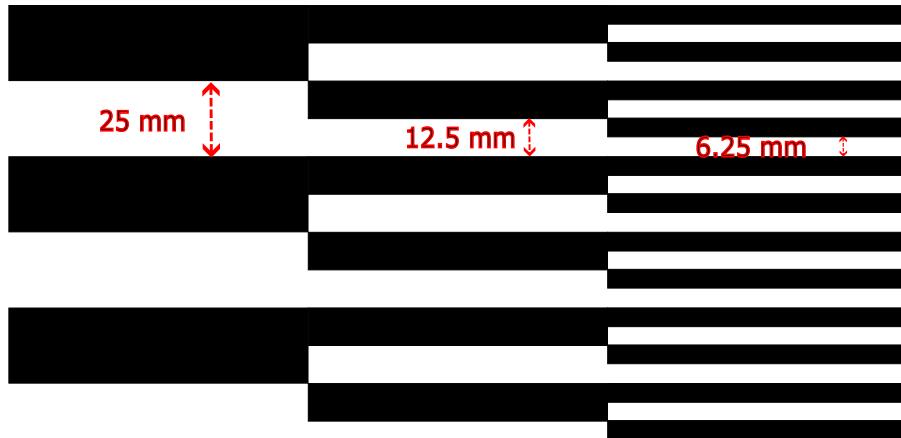
```
1 x_dot_iir(k) = Derivation((Avstand_IIR(k-1:k)*-1), Ts(k));
```

Siden den ultralyd-målte avstanden synker når LEGO-roboten kjører forover, inverteres avstandsmålingen før de blir derivert. På den måten gjenspeiler utgangsverdien $x_dot_iir(k)$ riktig hastighetsvektor.

6.3.2 Estimering av fart fra pulsmåling

Ved å la roboten kjøre over vekselvis sorte og hvite stripene, og underveis måle andel reflektert lys kan man generere et fartsestimat. Figur 30 viser et utklipp av underlaget som ble brukt under forsøket. Siden avstanden mellom stripene er kjent kan målingene brukes til å finne avlagt strekning ved multiplisere antall ganger roboten har registrert fargeendring med stripeavstanden.

Vi kan ikke si noe om posisjonen mellom pulsene. Roboten kan ha forflyttet seg, selv om det ikke er registrert en ny puls. Det er fordi lyssensoren fortsatt er i samme felt, og det ikke er detektert en fargeendring. Oppløsningen til avstandsmålingen er derfor gitt av tettheten og bredden på stripene. Farten har derfor heller ingen retningsvektor, det er ikke mulig å si noe om hvilken retning roboten beveger seg i.



Figur 30: Underlag for pulsgenerering av lysmålinger for strekningsmåling, med tre ulike intervall/oppløsning. Kun 25mm og 12.5 mm er brukt i forsøkene.

Estimert fart ved pulsmåling kalkuleres som:

$$v(k) = \frac{w}{t_2 - t_1} \quad (55)$$

hvor w er bredden på stripene i meter og $t_2 - t_1$ er den forløpte tiden fra forrige puls i sekunder.

Kode 13: MATLAB-kodeutdrag for puls-estimert fart

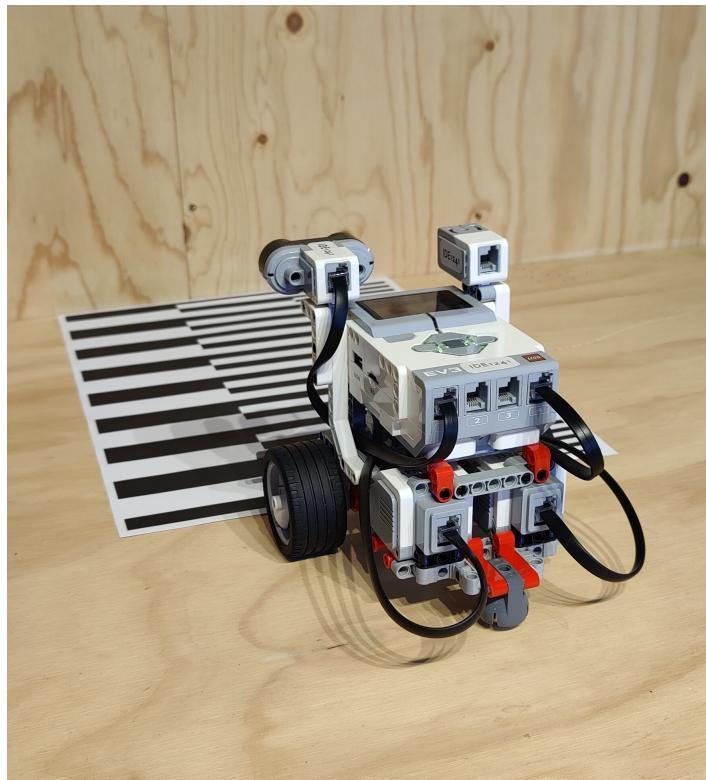
```
1 lys_endring = diff(Lys >= 50);
2 start_indeks = find(lys_endring == 1)+1;
3 slutt_indeks = find(lys_endring == -1);
4 if length(start_indeks) > length(slutt_indeks)
5     start_indeks = start_indeks(1:end-1);
6 end
7 puls_tid = Tid(slutt_indeks) - Tid(start_indeks);
8 fart_vektor = stripe_bredde ./ puls_tid;
```

Hovedelementene fra kodeutdraget:

- **lys_endring = diff(Lys >= 50)** i linje 1 registres endring fra forrige lysverdi med en terskelverdi på 50, oppretter en lokal vektor der alle elementene i Lys-vektoren med verdi større eller lik 50 erstattes med 1 og verdiene mindre enn 50 erstattes med 0. Videre går denne lokale vektoren gjennom diff-funksjonen som detekterer endring fra $0 \rightarrow 1$ og $1 \rightarrow 0$ (*hvit → sort, sort → hvit*). Diff-funksjonen skriver så henholdsvis 1 og -1 der hvor det er detektert endring.
- **start_indeks = find(lys_endring == 1)+1**, i linje 2 finner indeksen til elementene som er lik 1. Siden **lys_endring** vektoren er ett element kortere enn tidsvektoren, adderes 1.
- **start_indeks = find(lys_endring == 1)+1**, i linje 3 finner indeksen til elementene som er lik -1.
- **if length(start_indeks) >length(slutt_indeks)**, linje 4 til 6 kontrollerer at antall elementer i start og sluttindeks vektoren er lik. Hvis et puls signal blir registrert men ikke avsluttet, fjernes siste verdi fra startindeks.
- **puls_tid = Tid(slutt_indeks) - Tid(start_indeks)** i linje 7, pulstiden beregnes ved tidsdifferansen mellom start og slutten på en registrert puls.
- **fart_vektor = stripe_bredde ./ puls_tid**, i linje 8, beregner fartene.

6.4 Resultat

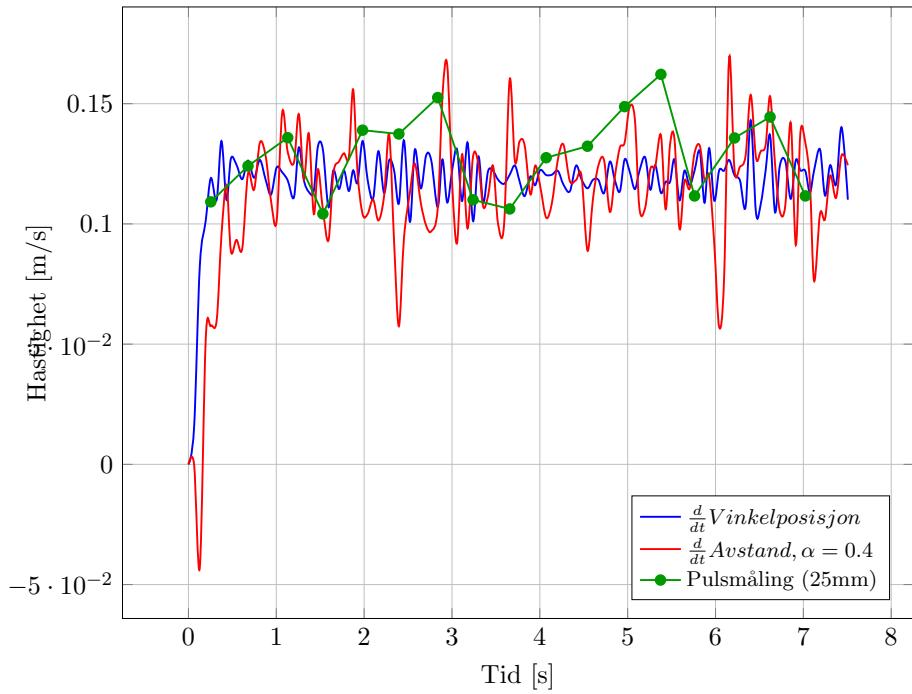
I denne delen presenteres forsøksresultatene av hastighetsberegning med EV3 roboten, ved målinger fra vinkelposisjonsensoren, ultralydsensoren og pulsmålinger. Figur 31 viser oppstilling av roboten for forsøket.



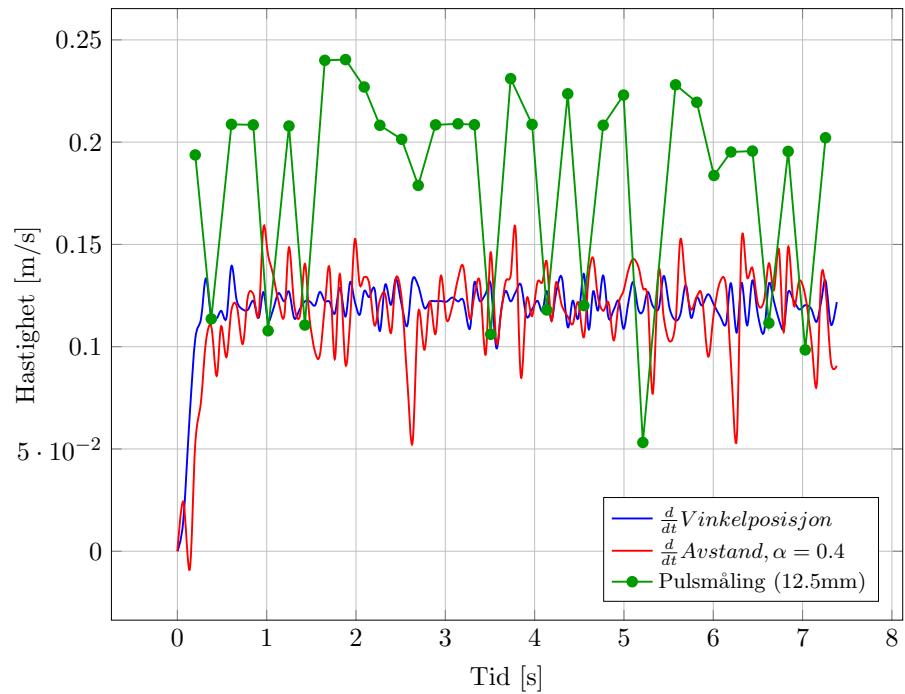
Figur 31: Oppsett for forsøk, ultralydsensor måler avstand fra vegg

For å få repeterbare forsøk kjører roboten med samme motorpådrag og stopper etter en strekning på omrent én meter. Ved å flytte roboten til de ulike oppløsningene ved underlaget for pulsgenerering, kan man sammenligne resultatene med tilnærmet identiske kjøregenskaper i hvert forsøk. Det blir ikke inkludert resultater fra underlaget med 6.25 mm oppløsing, dette er fordi registreringsområdet til sensoren er større enn strekene som gjør at det ikke registrerer pulsene med et konsistent resultat.

Forsøksresultatene er vist i figur 32 og 33.



Figur 32: Sammenligning av faktisk hastighet, derivert avstand (ultralyd) og pulsmåling med bredde 25.0mm.



Figur 33: Sammenligning av faktisk hastighet, derivert avstand (ultralyd) og pulsmåling med bredde 12.5mm.

Vi ser av grafene (figur 32 og 33) at hastigheten av vinkelposisjonen og ultralydmålingen har en høy grad av korrelasjon med noen mindre avvik.

Av likning 55 for beregning av fart med pulser fra lyssensoren, er det ingen informasjon om farten mellom pulsene. Farten kan derfor ha endret seg uten å gi utslag. Ved å redusere intervallet mellom stripene vil man få en hyppigere fartsmåling. I praksis fungerer det ikke så lett med LEGO-EV3 lyssensoren. I figur 33 ser man at den puls-estimerte farten er ustabil. Det skyldes for tett intervall mellom stripene. Lyssensoren rekker ikke å detektere stor nok endring i reflektert lys, man går glipp av en del målinger. Motsetningen er figur 32 hvor bredden på stripene er dobbelt så stor, som gir stort nok utslag til å registrere endring.

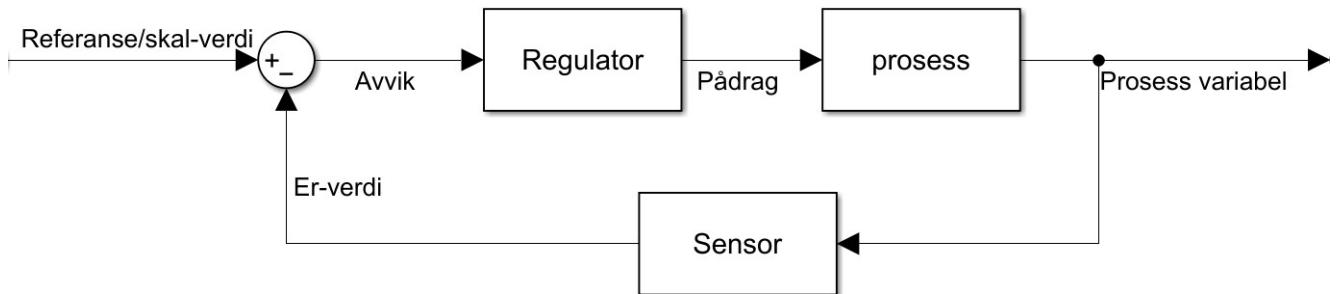
7 Automatisk kjøring med PID-regulator

Deltakere:

Christoffer Askeli ENOKSEN
Henry EIKELAND
Hermann Wathne TVEIT

7.1 Innledning

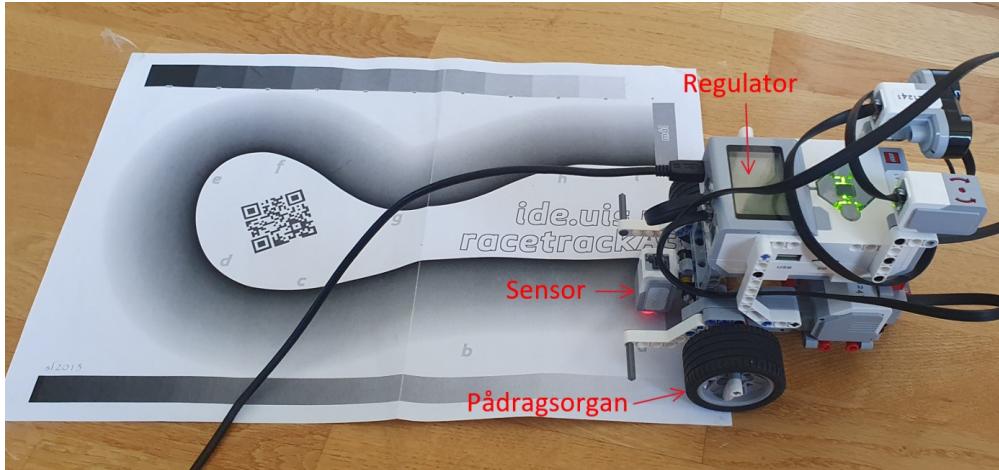
Mange prosesser i dag er automatiserte uten bruk av menneskelig påvirkning. Regulatorer styrer selv pådragsorganer som f.eks motorer, ventiler eller varmeovner. Illustrasjonen i figur 34 viser prinsippet av en reguleringsløyfe i blokkskjematiske form. Noen reguleringsprosesser krever en viss mengde nøyaktighet for å tilfredsstille de gitte kravene til prosessen. Dette er oppnåelig ved å stille inn en rekke forskjellige parameterer i regulatoren, som deretter bestemmer hvordan prosessen skal opptre.



Figur 34: Blokkskjema over en reguleringsløyfe med negativ tilbakekopling. Dette betyr at regulatoren styrer pådragsorganet ift referanseverdi.

7.2 Problemstilling

I dette kapittelet har målet vært å få til automatisk kjøring av LEGO-roboten gjennom gråskala-banen slik som det ble utført i kapittelet om manuell kjøring. Roboten skal selv styre hvordan den skal kjøre gjennom banen (figur 35). Etter endt kjøring ved mållinjen eller ved kjøring utenfor banen skal roboten selv stanse. Forskjellige parameterer i regulatoren skal testes for å bestemme hvilken påvirkning de har til styringen. I tillegg til kjøring gjennom banen skal det bevises hvorfor noen parametere er bedre enn andre.

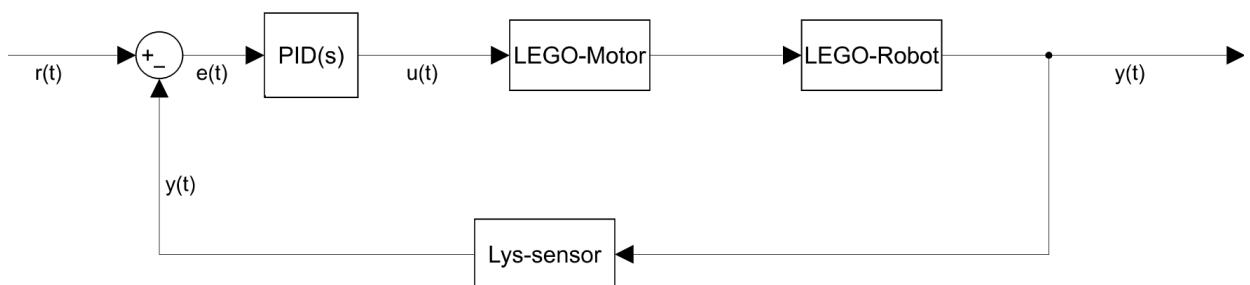


Figur 35: Tegning over gråskala-bane hvor LEGO-robot selv skal kjøre gjennom banen. Sensor i front registrer lysrefleksjon. Roboten justerer selv hvordan den skal kjøre og svinge for å komme til mål.

7.3 Løsningsforslag

Det blir introdusert en Proporsjonal Integral Derivat (PID)-regulator som skal styre motorene til LEGO-roboten. Motorene er pådragsorganene i denne prosessen, og regulatoren innføres ved kode i programmet. Lyssensoren brukes på lik linje som i manuell kjøring til å mÅle reflektert lys fra banen. Figuren 36 viser blokkskjematiske hvordan EV3-en opptrer som en reguleringsprosess.

Referanse/Settpunkt



Figur 36: Blokkskjema over reguleringssløyfa til LEGO-roboten. PID viser til regulatoren i EV3-en. Pådragsorganene i denne prosessen er begge motorene til hjulene, hvor selve roboten er prosessen.

Avviket beregnes ut som summen mellom referanseverdien og lyssensorverdien som vist med formel:

$$e(t) = r(t) - y(t) \quad (56)$$

Regulatoren bruker avviket til å beregne hva pådraget ut til motorene skal være. En PID-regulator består i prinsippet av 3 forskjellige ledd, hvor hvert ledd gir et bidrag til det totale pådraget.

De er:

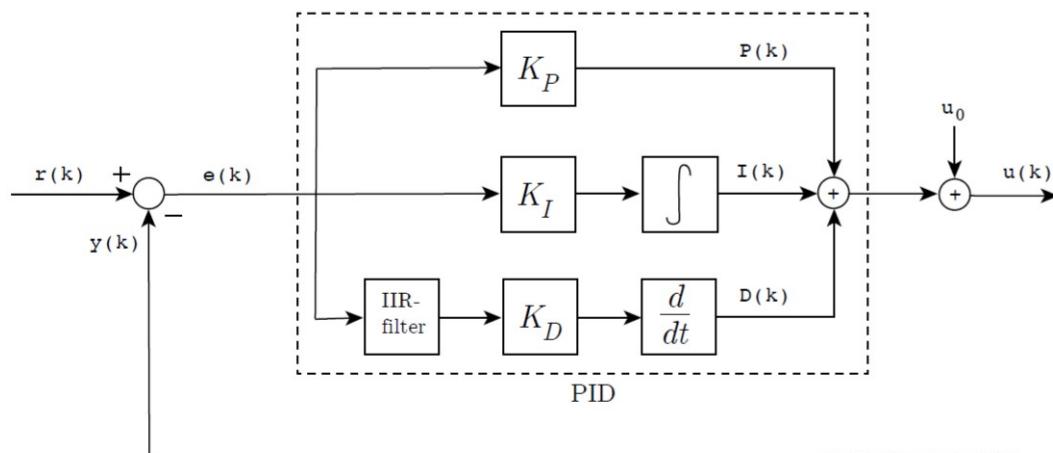
- P – Proporsjonalandel
- I – Integralandel
- D - Derivatdel

Det totale pådraget beskrives med formel:

$$u(t) = u_0 + \underbrace{K_P \cdot e(t)}_{P} + \underbrace{K_I \int_0^t e(\tau) d\tau}_{I} + \underbrace{K_D \cdot \frac{d}{dt} e_f(t)}_{D} \quad (57)$$

med innhold:

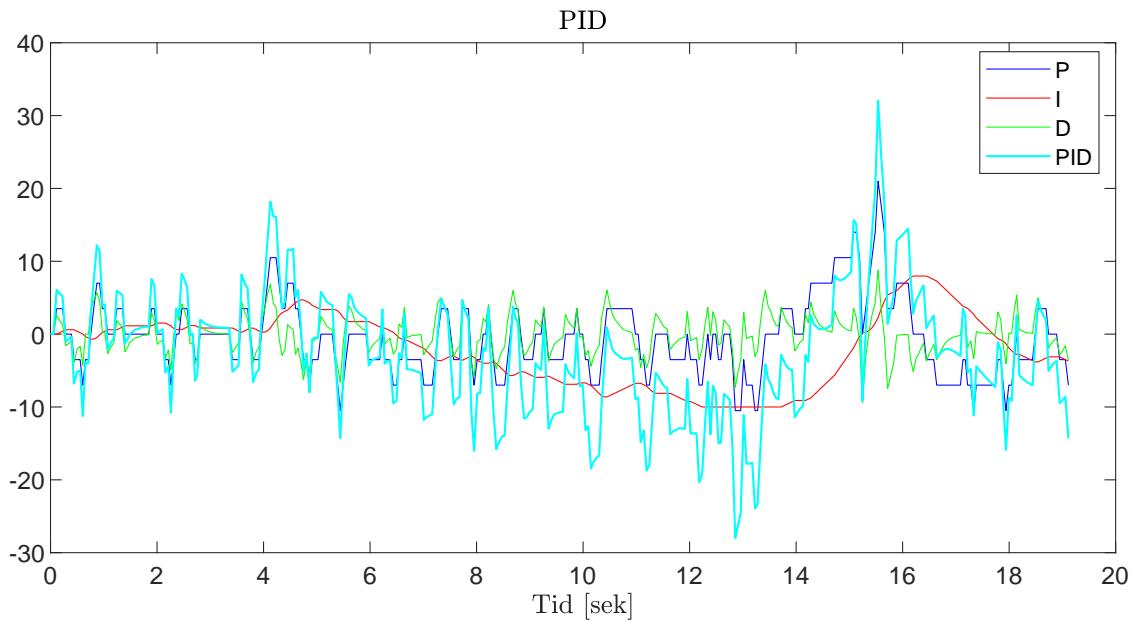
- $u(t)$ - er det totale pådraget
- u_0 - er basispådraget
- $e(t)$ - avviket i prosessen
- $e_f(t)$ - filtrert avviksverdi
- K_P - regulatorparameter, proporsjonalforsterkning
- K_I - regulatorparameter, integralforsterkning
- K_D - regulatorparameter, derivasjonsforsterkning



Figur 37: Det totale pådraget her presentert i blokkskjematiske form. Hver regulatorparameter ser avviket og gir ut et bidrag til det totale pådraget som beskrevet i formel 57. $P(k)$, $I(k)$ og $D(k)$ tilsvarer her P, I og D i formelen. [1]

Basispådraget er et fast pådrag til motorene for fremdrift. Dette fordi det er ønskelig å regulere fremdrift og svinging med å enten legge til eller trekke fra litt på basispådraget. Ved å velge $u_0 = 0$ vil roboten bare rotere uten fremdrift. I dette prosjektet er det brukt henholdsvis et basispådrag på 10.

Det beregnede totale pådragssignalet svinger mellom positive og negative verdier ift. avviket, som dermed vil avgjøre hvilken vei roboten ender med å svinge. Figur 38 viser et eksempel på hvordan de forskjellige parametriene oppfører seg, og hvordan det totale pådragssignalet til slutt ser ut.



Figur 38: Graf over bidragene til regulatorparameter P (blå), I (rød) og D (grønn). Det totale pådragssignalet er summen av alle disse i lyseblått.

7.4 MATLAB-kode

Koden til automatisk kjøring med PID-regulator er bygget videre fra koden til manuell kjøring, med tillegg implementering av en regulator som tar seg av pådragssignalet til motorene. Kvalitetsberegningen skjer på lik linje som i manuell kjøring(Kodeutdrag 10). Kodeutdrag 14 nedenfor viser den delen i programmet hvor man stiller inn regulatorparametrene. Linje 2 viser filtreringskonstanten α , og linje 3, innstillingene for valg av referansepunkt. Denne har blitt satt til første sensorverdi.

Kode 14: Kodeutdrag av Innstillinger til parametre

```

1 % Parametre
2 alfa = 0.3;
3 setpkt = Lys(1);
4
5 % Parameterinnstillinger
6 u0 = 10;
7 Kp = 3.5;
8 Ki = 3.5;
9 Kd = 0.1;

```

Kodeutdrag 15 viser utregninger av hvert bidrag til det totale pådraget som beskrevet i formel 57. P-bidraget viser her i linje 2, I-bidraget i linje 4 og D-bidraget i linje 7. Som beskrevet i tidligere kapittel om numerisk integrasjon, så brukes samme metode til beregning av I-bidraget. Det gjelder også D-bidraget som kaller på funksjonen til numerisk derivasjon. Legg merke til at avviket filtreres i linje 6 før funksjonen kalles, dette er fordi den deriverte er følsom for store endringer som beskrevet i kapittel om numerisk derivasjon.

Kode 15: Kodeutdrag av PID beregninger

```

1 %P
2 P(k) = Kp*e(k);
3 %I
4 I(k) = EulerForward(I(k-1),Ki*e(k-1),Ts(k));
5 %D
6 e_f(k) = IIR_filter(e_f(k-1),e(k),alfa);
7 D(k) = Derivation(Kd*e_f(k-1:k),Ts(k));

```

Det totale pådragsignalet beregnes i linje 2 i Kodeutdrag 16. Legg merke til at basispådraget, u_0 , ikke ligger i linje 2 som det er beskrevet i formel 57. Dette er fordi roboten har 2 pådragsorgan, nemlig hjulene på hver side som skal oppstre motsatt av hverandre ift. pådragssignalet. u_0 legges da heller inn på linje 5 og 6, som er signal til motorene, for å sørge for at roboten har en konstant fremdrift.

Kode 16: Kodeutdrag av total pådragssignal

```

1      % Det totale pådraget
2      u(k) = P(k)+I(k)+D(k);
3
4      % Signal til pådragsorgan
5      PowerB(k) = u0-u(k);
6      PowerC(k) = u0+u(k);

```

I-bidraget integrerer seg oppover når den ser et avvik. Ved vedvarende avvik vil dette vokse seg større og større og ikke minke før prosessen snur og avviket endrer fortegn. I-leddet kan da plutselig henge etter, og bidraget fører til en ustabil regulering. Roboten kjører dermed av banen. I kodeutdrag 17 er det satt inn en integratorbegrensning i form av if-betingelser, som setter øvre og nedre grenseverdier. I koden her er det brukt 10 og -10 som grenser.

Kode 17: Kodeutdrag av integratorberegsning

```

1      % Integratorbegrensing
2      if I(k)>10
3          I(k) = 10;
4      elseif I(k) < -10
5          I(k) = -10;
6      end

```

7.5 Resultat

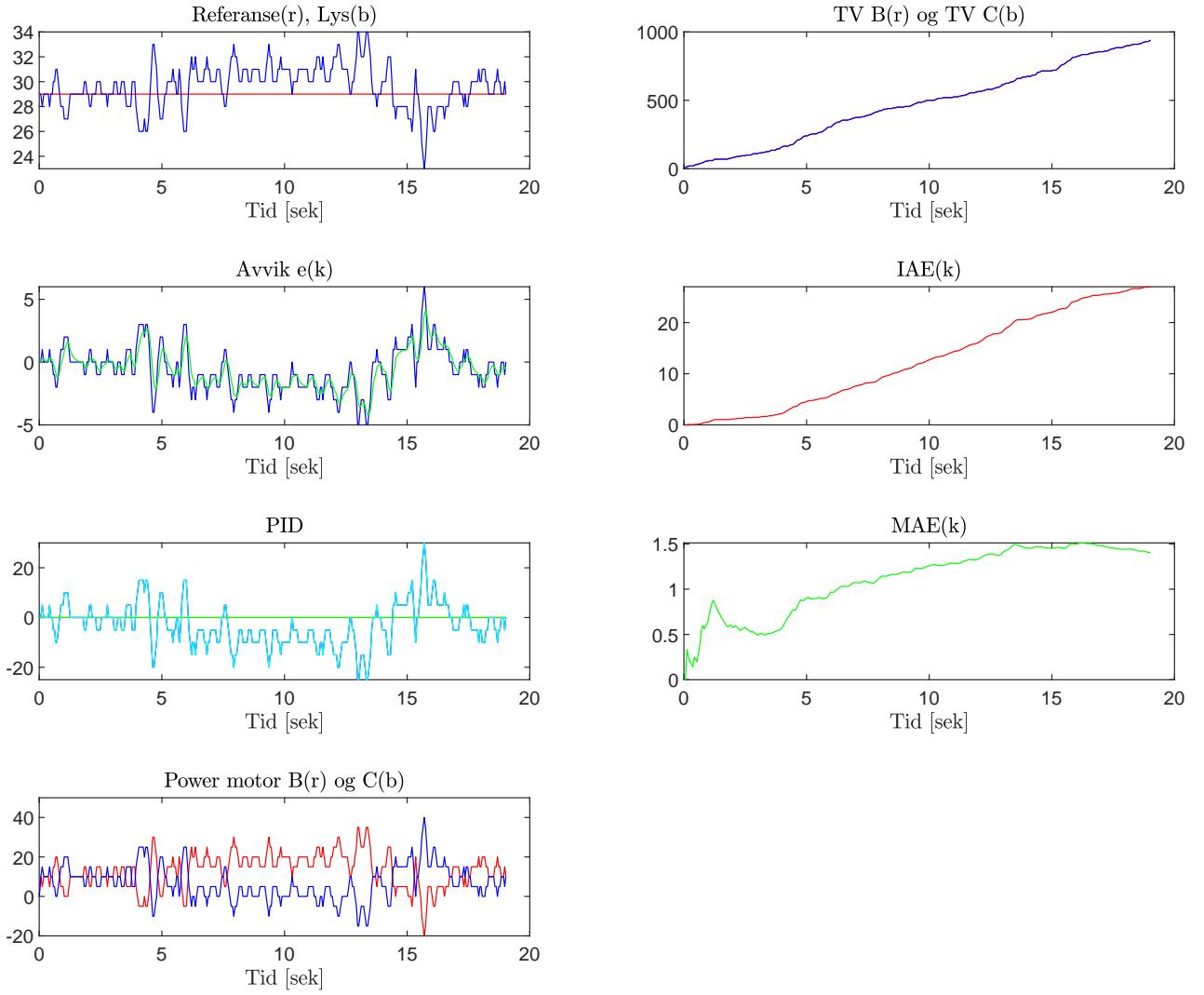
I dette kapittelet presenteres det ulike brukte kombinasjoner av regulator parametre K_p , K_i og K_d . Det finnes ulike metoder å optimalisere en prosess på. Metoden som er brukt her er *prøve og feile* metoden. De ulike kombinasjonene blir så sammenlignet ved ulike kvalitetsdata.

7.5.1 P-regulator

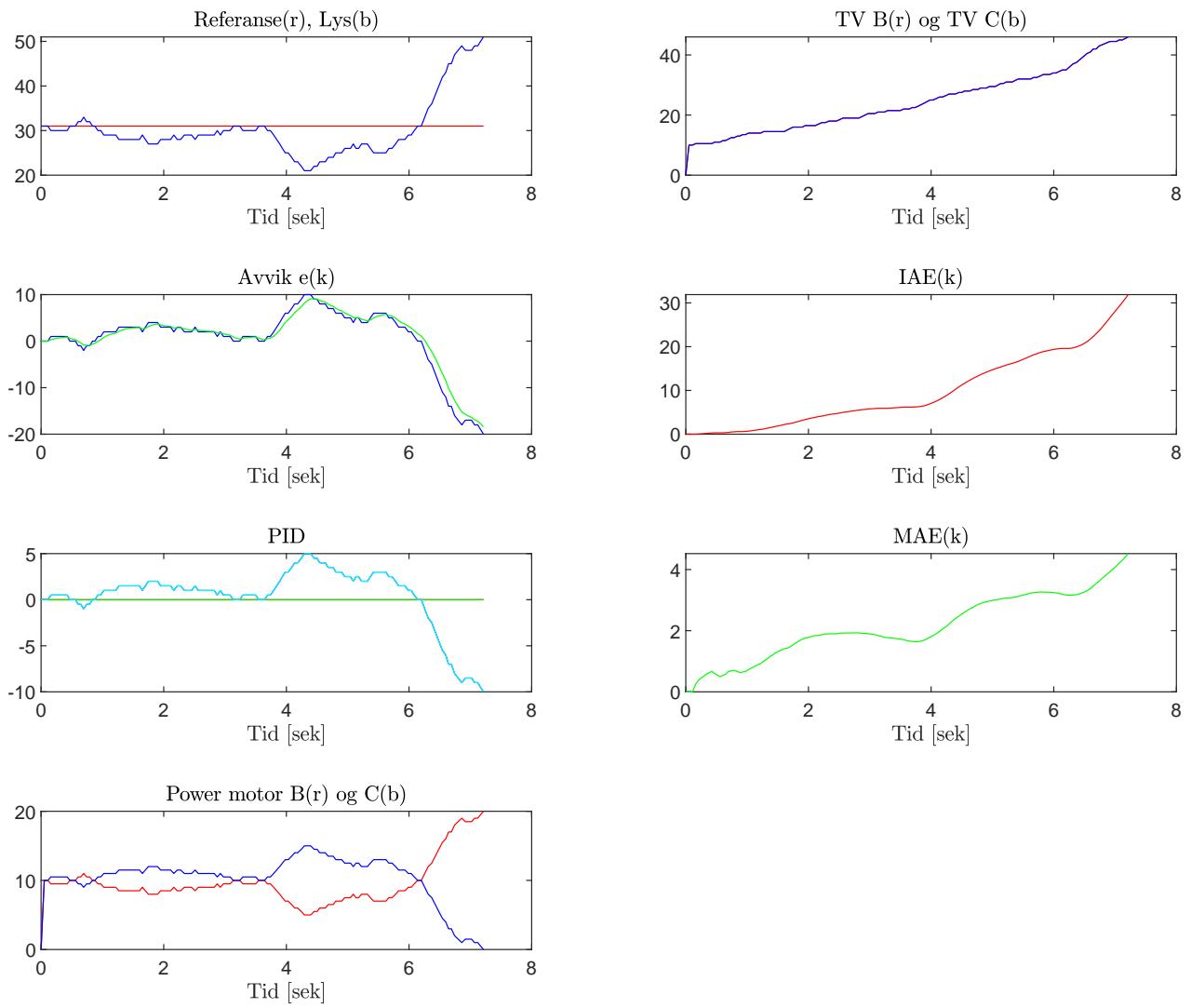
Til *prøve-og-feile* metoden, så starter man først med P-leddet. I-, og D-bidraget skrus av ved å sette K_i og K_d til null. K_p blir først satt til en lav verdi. Deretter blir nye gjennomføringer utført med økt K_p . Når verdien så høy at man opplever prosessen som litt hakkete, reduseres denne med 30%, og man har funnet en K_p som kan brukes til videre optimalisering. Tabell 5 viser tester utført med ulik K_p . Figurene 39 og 40 viser resultat av utførelse med $K_p = 5$ og $K_p = 0.5$.

Plattform	pc	pc	pc	pc	pc
Strømkilde	230v	230v	230v	230v	230v
Samtidig plotting	nei	nei	nei	nei	nei
u0	10	10	10	10	10
Kp	0,5	1	2	5	10
Ki	0	0	0	0	0
Kd	0	0	0	0	0
referanse	X	32	32	29	X
Middelverdi μ	X	34,9	33,5	29,5	X
$ referanse - \mu $	X	2,88	1,53	0,53	X
Standardavvik σ	X	7,2	3,5	1,7	X
Kjøretid [s]	X	19,9	19,5	19	X
IAE	X	127,7	60	27	X
MAE	X	6,5	3,1	1,4	X
TV B	X	207	334	940	X
TV C	X	207	334	940	X
Middelverdi av Ts [s]	X	0,051	0,051	0,052	X
Antall målinger(k)	X	389	384	369	X

Tabell 5: Utførelser av automatisk kjøring med P-regulator med ulike K_p -verdier. Den beste innstillingen her er ved $K_p = 5$. Legg merke til at for lave eller for høye verdier av K_p fører til at roboten ikke fullfører, som betyr at regulatoren blir for treg eller for kjapp. Beste verdier er merket med **fet skrift**. X betyr at roboten ikke kom i mål.



Figur 39: P-regulator med innstilling: $u_0 = 10$, $k_p = 5$. Fra tabell 5 var disse innstillingene de med best overordnet resultat. Legg merke til at pådragssignalet, PID, har samme form som avviksignalet. Dette fordi K_i og K_d er skrudd av, og P-leddet er proporsjonalt med avviket. Legg også merke til den hakkete grafen til motorB og motorC, dette resulterer i en høyere TV(total variasjon). Likevel scorer denne parameterinnstillingen bra på de andre kvalitetsmålene.



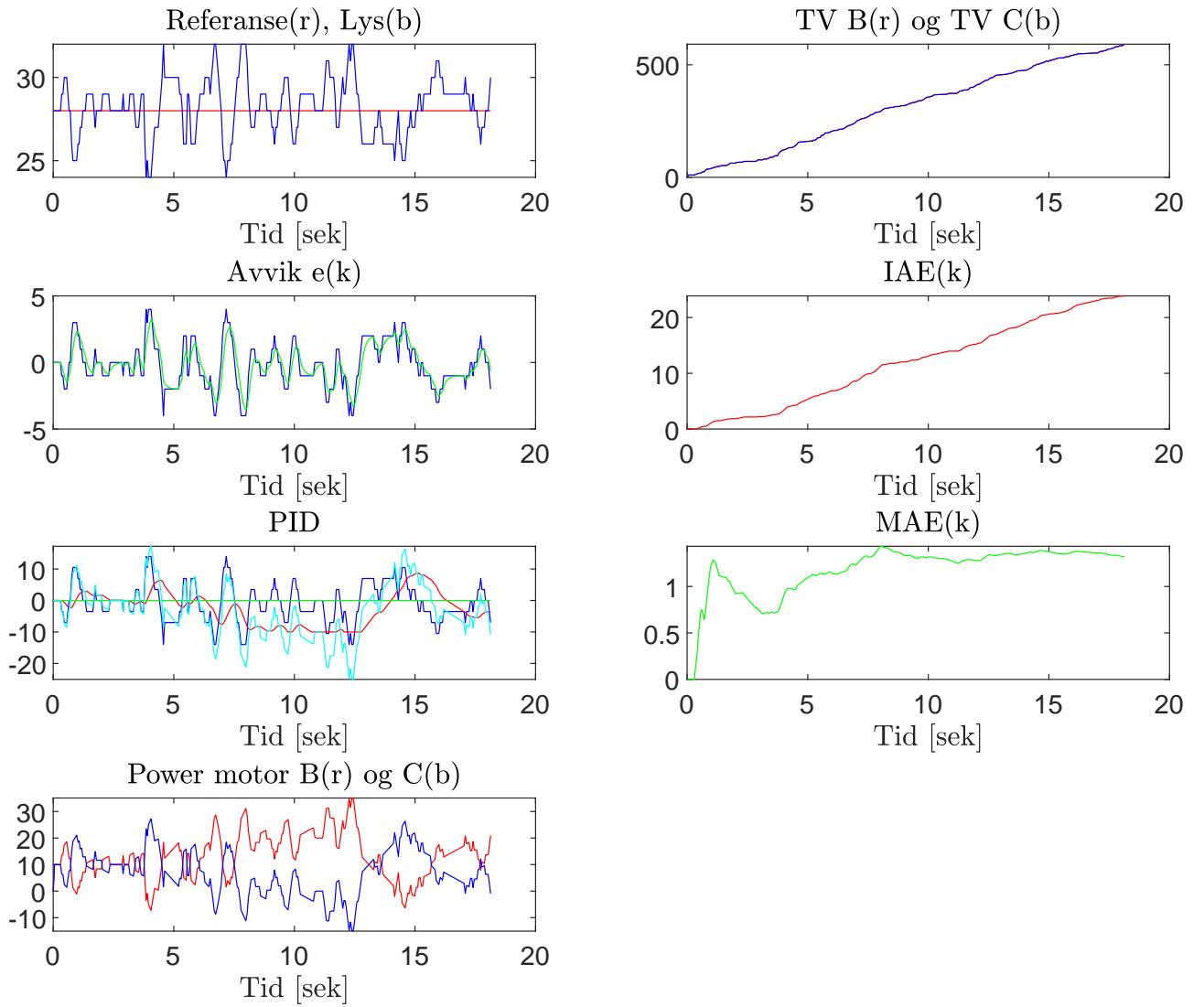
Figur 40: P-regulator med inntilling: $u_0 = 10$, $k_p = 0.5$. Disse parameter innstillingene gjør at roboten kjører ut av banen ved $Tid = 7$ sek. Det er altså for svak parameterforsterning til P-leddet. EV3'en henger ikke med i prosessendringene.

7.5.2 PI-regulator

Videre i *prøve-og-feile* metoden skal også I-bidraget legges til i pådragssignalet. Når man har funnet en tilfredstillende K_p , så er fremgangsmetoden lik for I-leddet. Man starter med lav verdi, for så å øke for hver utførelse. Man forsetter helt til systemet reagerer med for mye svingninger. Den verdien man da har på K_i reduserer man med 30%, og dermed har man funnet regulatorparameter til K_i . Tabell 6 viser de forskjellige kombinasjonene som ble testet. Verdien på K_p er satt til 3,5, etter testen som ble utført med P-regulator. Figur 41 viser resultatet med parameterinnstilling $K_p = 3,5$ og $K_i = 5$.

Plattform	pc	pc	pc	pc
Strømkilde	230v	230v	230v	230v
Samtidig plotting	nei	nei	nei	nei
u0	10	10	10	10
Kp	3,5	3,5	3,5	3,5
Ki	0,5	1	2	5
Kd	0	0	0	0
referanse	33	33	29	28
Middelverdi μ	33,1	33	29,1	28,1
$ referanse - \mu $	0,15	0,04	0,08	0,12
Standardavvik σ	2	2,1	1,8	1,7
Kjøretid [s]	19,4	18,9	18	18,1
IAE	30,7	31,5	25	23,9
MAE	1,6	1,7	1,4	1,3
TV B	629	625	545	592
TV C	629	632	538	592
Middelverdi av Ts [s]	0,051	0,052	0,052	0,052
Antall målinger(k)	379	367	350	348

Tabell 6: Utførelser av automatisk kjøring med PI-regulator med ulike K_i -verdier. K_p er satt til 3,5 på alle utførelsene. Beste verdier er merket med **fet skrift**. Fra tabellen har parameterinnstilling med $K_i = 5$ den med best score når det gjelder IEA, MAE og standardavvik.



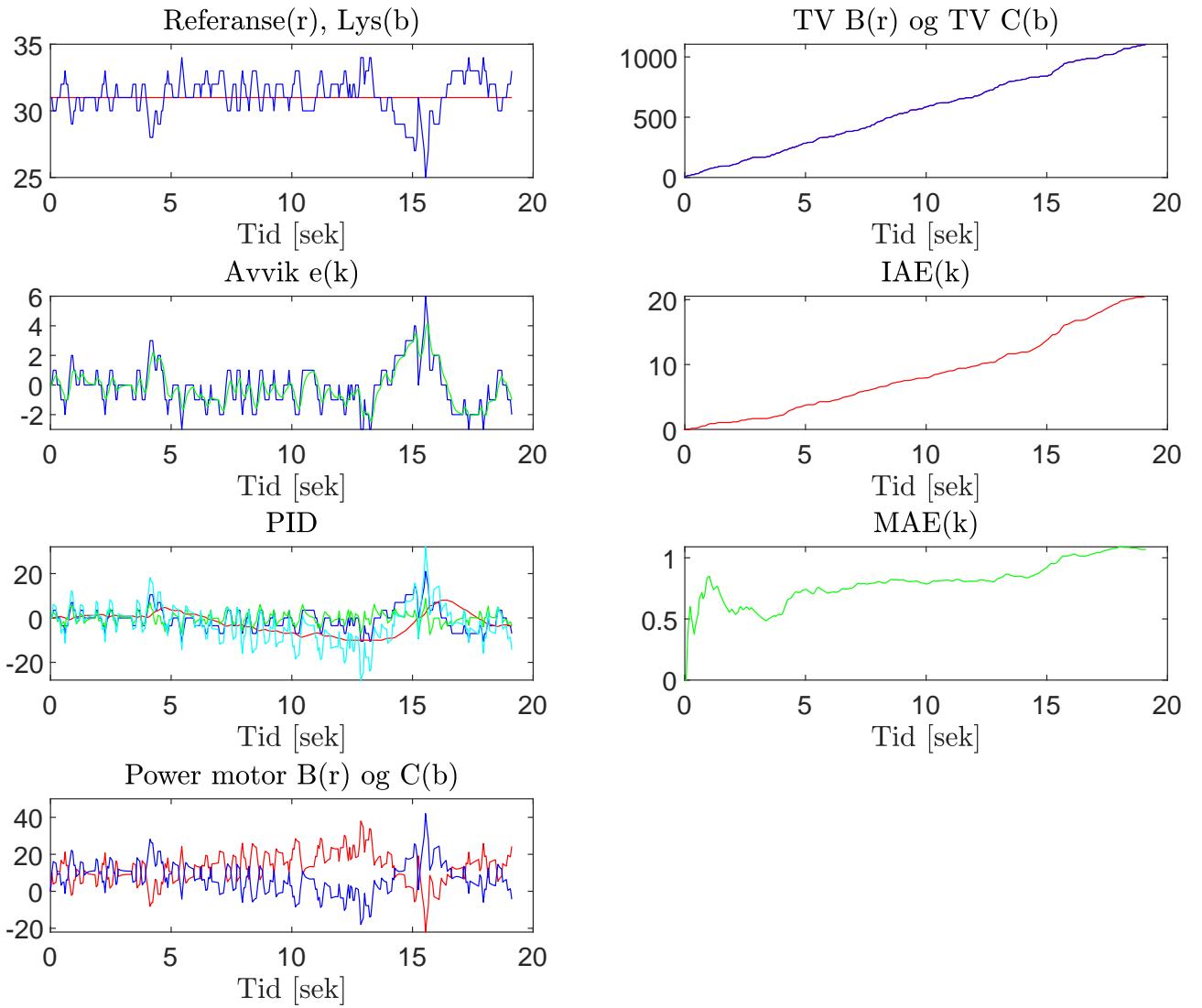
Figur 41: PI-regulator med inntilling: $u_0 = 10$, $k_p = 3.5$ og $K_i = 5$. Fra tabell 6 var disse de inntillingene med best overordnet resultat. Legg merke til graf av PID, der kan man se at I-leddet(i rødt) nå leverer et bidrag til pådragssignalet (lyseblå). D-leddet ligger konstant på null som forventet siden den er skrudd av. Legg også merke til summen av TV er lavere her ift. P-regulatoren(figur 39).

7.5.3 PID-regulator

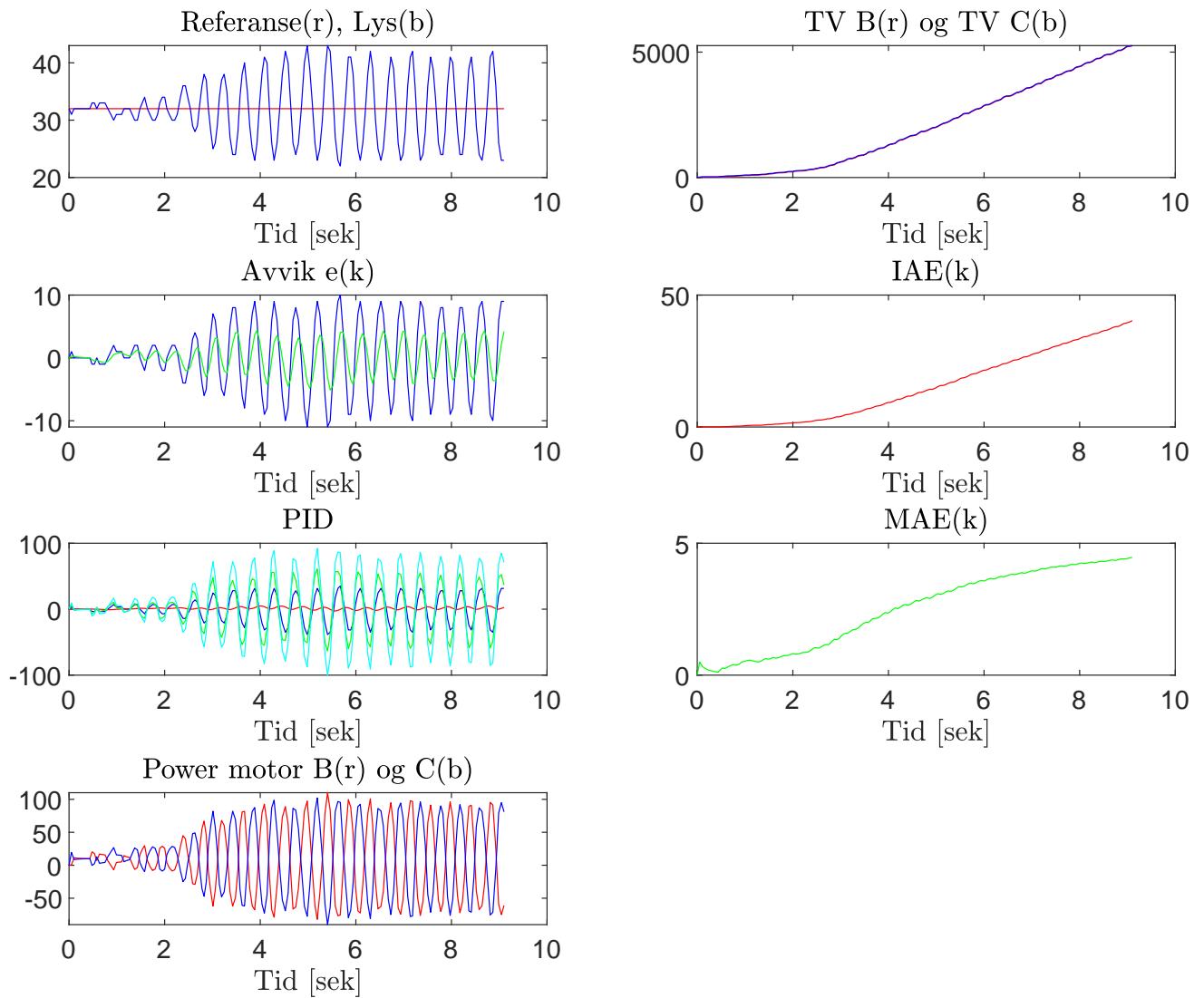
Til slutt legger man til D-bidraget i *prøve og feile* metoden. Denne baseser seg på numerisk derivasjon som beskrevet i formel 57. Fremgangsmåten for denne er lik som for de andre parametrene, hvor man begynner med lave verdier for så å øke. Når systemet reagerer med for hurtige svingninger, så reduseres verdien med 30%, og man har funnet en kandidat til Kd -parametren. Tabell 7 viser kombinasjonene som ble testet. Kp og Ki har faste verdier som er funnet fra tidligere steg, og er henholdsvis satt til 3.5 i dette tilfellet. Figur 42 viser resultatet med parameterinnstilling $Kp = 3.5$, $Ki = 3.5$ og $Kd = 0.5$. Den deriverte reagerer på endringer i systemet, og hvis Kd -paramateren blir satt for høy vil prosessen begynne å svinge ukontrollert, dette vises i figur 43 som har en $Kd = 1$.

Plattform	pc	pc	pc	pc	pc
Strømkilde	230v	230v	230v	230v	230v
Samtidig plotting	nei	nei	nei	nei	nei
u0	10	10	10	10	10
Kp	3,5	3,5	3,5	3,5	3,5
Ki	3,5	3,5	3,5	3,5	2,1
Kd	0,1	0,3	0,5	1	0,3
referanse	25	33	31	X	30
Middelverdi μ	25,2	33	31,1	X	30
$ referanse - \mu $	0,16	0,04	0,08	X	0,04
Standardavvik σ	1,8	1,7	1,4	X	1,8
Kjøretid [s]	18,8	19,7	19,1	X	19,1
IAE	24,6	25,3	20,5	X	25,7
MAE	1,3	1,3	1,1	X	1,3
TV B	674	1149	1105	X	950
TV C	674	1149	1105	X	940
Middelverdi av Ts [s]	0,053	0,052	0,052	X	0,051
Antall målinger(k)	357	381	368	X	373

Tabell 7: Utførelser av automatisk kjøring med PID-regulator med ulike Kd -verdier. Kp er satt til 3.5 på alle utførelsene i denne gjennomføringen. $Kd = 1$ resulterte i for hurtige svingninger. Kombinasjonen med best score har $Kd = 0.5$. Beste verdier er merket med **fet skrift**. Legg merke til at bedre resultat på IAE og MAE, gir høyere verdier til TV(total variasjon). X betyr at roboten ikke kom i mål.



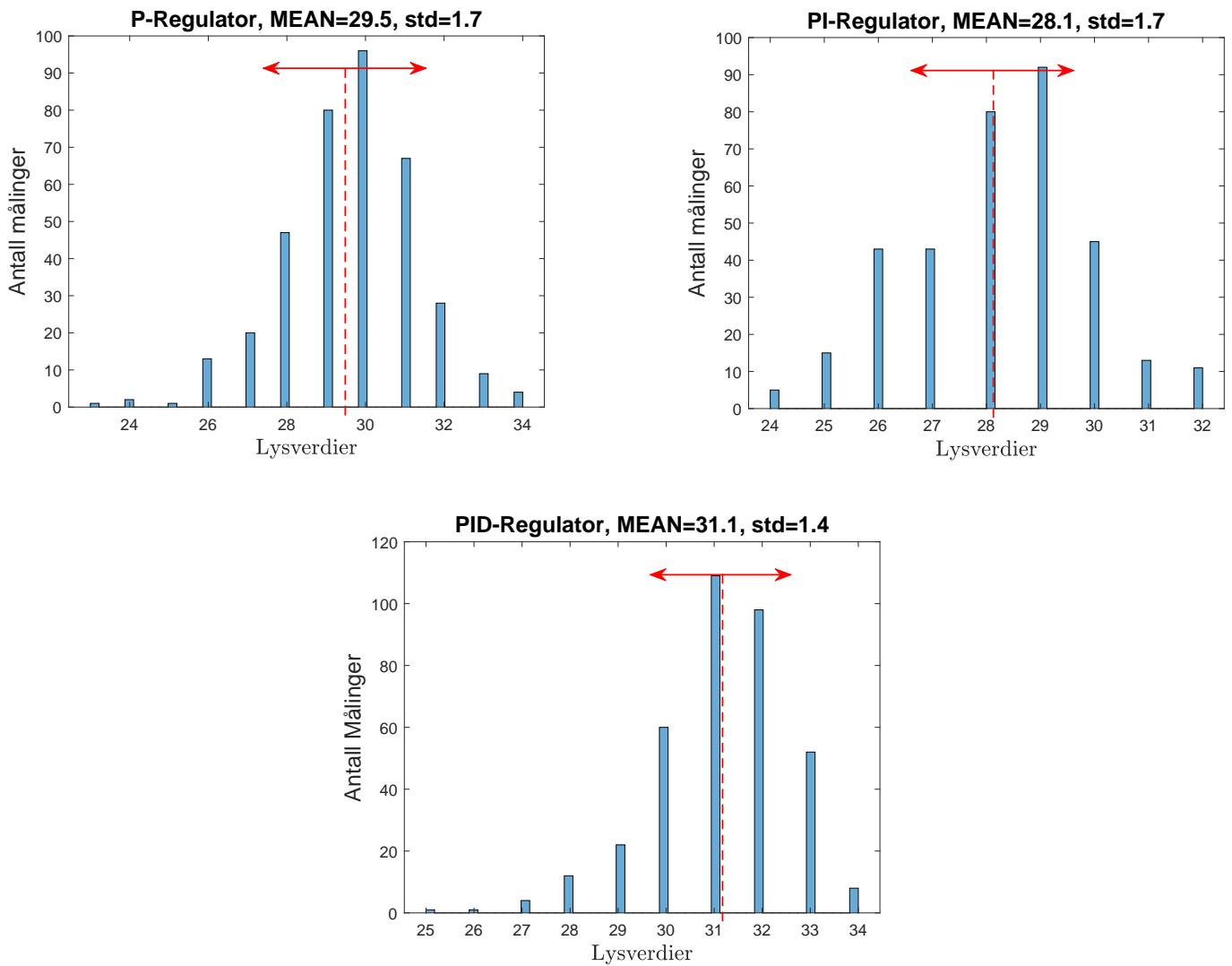
Figur 42: PID-regulator med innstilling: $u_0 = 10$, $k_p = 3.5$, $K_i = 3.5$ og $K_d = 0.5$. Disse innstillingene gav best score fra tabell 7. Grafene i PID vinduet viser nå at alle PID leddene nå leverer et bidrag til det totale pådraget. Avviket over kjøreperioden ligger her nokså nær null, men dette fører og til høyere TV(total variasjon). Det viser altså at TV og IAE/MAE virker proporsjonalt motsatt av hverandre.



Figur 43: PID-regulator med innstilling: $u_0 = 10$, $k_p = 3.5$, $K_i = 3.5$ og $K_d = 1$. Disse innstillingene tilsvarer kjøring fra tabell 7 merket med X. K_d har for stor forsterkning som fører til at D-bidraget blir for høyt og prosessen begynner å svinge ukontrollert. Her begynte LEGO-roboten å svinge rakst att og frem på gråskala-banen. Svingningene blir for høye og basispådraget får ingen virkning. Det blir ingen fremdrift.

Fra tabellene til automatisk kjøring med P-, PI-, og PID-regulator sammenlignes nå de kombinasjonene som ga best score for hver regulator. I fordelingene under (figur 44) blir det sett på middelverdi og standardavviket til lysmålingene. Parameter innstillingene er:

- P-regulator: $u_0 = 10, K_p = 5$
- PI-regulator: $u_0 = 10, K_p = 3.5, K_i = 5$
- PID-regulator: $u_0 = 10, K_p = 3.5, K_i = 3.5, K_d = 0.5$



Figur 44: Fordelingene viser middelverdi som stiplet rød linje, og standardavvik som rød pil til hver regulator. PID-regulator viser seg å ha de beste verdiene, der $std = 1.4$. Likevel er det lite så skiller de ulike regulatorene. Derimot tar disse dataene kun i betrakning registrert lysmåling gjennom banen.

Selv om PID-regulatoren har best overordnet resultat, vil det ikke alltid bety at dette er den foretrukne å bruke i prosessen. Verdiene til TV er nokså høye, og noen ganger er det foretrukket å tolerere en større avviksfeil(IAE) til gevinst for lengre levetid på utstyrret. Dermed kan PI-regulatoren være den foretrukne å bruke her siden den har lavere verdier på TV og likevel gode resultater på resterende kvalitetsmål.

8 Konklusjon

Hovedmålet med dette prosjektet har vært å kunne løse ulike oppgaver ved hjelp av anvendt matematikk og fysikk i programmering av LEGO-robot. I oppgavene om numerisk integrasjon, filtrering og numerisk derivasjon har vi sett at teorien bak stoffet bekreftes gjennom de praktiske oppgavene vi har utført. På de større prosjektene har vi fått bruk for flere av disse elementene for å løse oppgavene. Gjerne i kombinasjon med bruk av flere sensorer for å bekrefte teorien, som for eksempel i oppgaven om estimering av hastighet.

Når det gjelder manuell kjøring og automatisk kjøring, har vi sett at den automatiske kjøringen ofte produserer et bedre resultat, forutsatt brukbare regulatorparametre. Dette får vi bekreftet gjennom kvalitetsmålene. Vi har også sett at sammenhengen mellom proporsjonaliteten på IAE og TV viser igjen resultatene. Gjennomførelse av disse prosjektene har ført til økt kunnskap av både teori og programvare (MATLAB). Her har vi blitt tryggere på å hente ut data til analysering i form av grafer, diagrammer og tabeller.

Referanser

- [1] T. Drengstig. Prosjektdel av ELE130 Anvendt matematikk og fysikk i robot-programmering. Lego Mindstorms og MATLAB/Python i skjønn forening. Technical report, Universitet i Stavanger, 2023. Utdelt materiale.
- [2] J.C. LACEFIELD. Diagnostic radiology physics: A handbook for teachers and students. Technical report, University of Western Ontario, 2014. ”12: Physics of Ultrasound”.

Tillegg A Timeliste

Uke	Henry	Hermann	Christoffer
9	2	2	2
10	25	13	13
11	16	23	24
12	0	13	12
13	16	24	11
14	7	16.5	0
15	37.5	26	29
16	30	24	38
17	38	37	41
Sum	171.5	178.5	170

Tabell 8: Timeliste oppdatert 28.04.2023 14:11