DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering:

Automatisering og elektronikkdesign
Y-vei

Vårsemesteret, 2022

Åpen

Forfatter: Stig Haaland,
Simon Rygg Meland

Fagansvarlig: Tormod Drengstig

Veileder(e): Tormod Drengstig

Tittel på bacheloroppgaven:
Momentregulering ved bruk av Q2A frekvensomformer.

Engelsk tittel:
Torque control using Q2A variable frequency drive

Studiepoeng: 20

Emneord:

Vinsj
Posisjonsregulering
Hastighetsregulering
Momentregulering
PLS
Frekvensomformer

Sidetall: 107

+ vedlegg/annet: 2 Vedlegg

Stavanger 15. Mai 2022

Sammendrag

Bacheloroppgaven tar for seg forskjellige måter å regulere et vinsjsystem. Det blir først gjennomgått hvordan systemet er bygget opp, de sentrale komponentene og programvarene som ble brukt i oppgaven. Kodene for å kalibrere og regulere systemet ble implementert som strukturert tekst og ladderkode i programvaren Sysmac Studio.

Vinsjsystemet består av frekvensomformer, PLS og motor med enkoder. På akslingen til motoren er det festet en vaiertrommel med hengende last. For at brukeren skal ha tilgang på nyttig og relevant informasjon ble det utviklet HMI skjermer.

For å finne høydeområdet vinsjsystemet kan kjøre mellom, har vi utviklet en kalibreringsprosess. Den vil ved aktivering beregne et høydeområde, som videre settes som begrensing for hvor mye vaier som kan spoles inn eller ut i programmene for regulering av systemet.

Videre tar oppgaven trinnvis for seg de ulike reguleringene av vinsjsystemet. Det presenteres teori, løsning og resultater for posisjon-, hastighet- og momentregulering. Det ble brukt 'prøv og feil'- og skogestads metode for å finne reguleringsparameterene til posisjon- og hastighetsregulatoren.

Det ble utviklet en PID-regulator for posisjonsjustering av systemet. Her setter brukeren en ønsket høyde innenfor det kalibrert området og regulatoren vil justere posisjonen til denne høyden.

For å hastighetsregulere vinsjsystemet ble det utviklet en feed-forward PI-regulator. Her kan brukeren sette en ønsket innspolingshastighet, regulatoren vil justere pådrag til motoren for å oppnå denne hastigheten.

For å unngå slakk vaier når lasten løftes opp i en bølgebevegelse ble det implementert momentregulering. Brukeren setter et ønsket moment på motoren og regulatoren justerer pådraget til motoren dersom det oppstår avvik fra denne verdien. Det blir dermed foretatt hurtig inn- og utspoling av vaier når lasten løftes.

Til slutt tar oppgaven for seg en konklusjon sammen med forslag til forbedringer og videreutvikling av vinsjsystemet.

Innhold

Kapittel 1

Innledning

1.1 Oppgavebeskrivelse

Under er oppgaveteksten gjengitt:

Ved låring av livbåt fra plattform/skip i høy sjø er det kritisk at livbåtene senkes kontrollert. Strekk-kreftene i livbåtvaieren må holdes konstant etter første kontakt med bølgene ved at det foretas hurtig inn- og utspoling av vaier for å unngå rykk og slakk i store bølgehøyder. For å regulere kreftene i vaieren når livbåten treffer vannflaten samtidig med at det er bevegelser i skipet, må det anvendes momentregulering.

Ut ifra denne oppgaveteksten baserte denne bacheloroppgaven seg på demomodellen fra Omron som vist i figur 1.1. Den består av en PLS med HMI-skjerm og en frekvensomformer som er tilkoblet en asynkron AC-motor med enkoder. På akslingen til denne motoren er det montert en vaiertrommel med tau som fungerer som et vinsjsystem med hengende last.

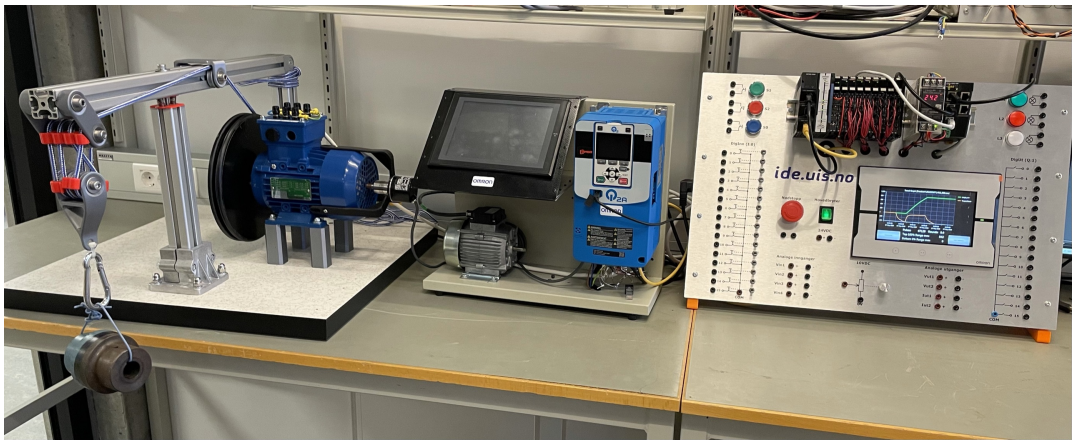


Figure 1.1: I figuren vises hele vinsjsystemet. Fra venstre: Vinsjmodell, motor med enkoder, frekvensomformer og PLS med HMI skjerm.

Oppgaven gikk ut på å utvikle kode som gjorde det mulig å posisjon-, hastighet- og momentregulere vinsjen. I PLS koden til systemet er det i tillegg implementert en funksjon som kalibrerer vinsjsystemet. Når denne funksjonen aktiveres vil styresystemet finne området vinsjen kan kjøre mellom. Videre er det ønsket av Omron å bruke vinsjsystemet til demonstrasjoner og opplæring.

1.2 Mål for oppgaven

Ut ifra oppgaveteksten ble følgende punkter utarbeidet under prosjektet:

- Sette oss inn i problemstillingen og forstå ønsket funksjonalitet.
- Lage en forbedret vaiertrommel med et tau uten strekk.
- Sette oss inn i frekvensomformeren og medfølgende programvare.
- Programmere og teste de ulike innebygde reguleringsfunksjonene i frekvensomformeren (hastighets-, posisjons- og momentregulering).
- Finne reguleringsparametre for posisjon- og hastighetsregulator gjennom ulike metoder.
- Koble bryterpanelet i modellen for ulike måter å operere frekvensomformeren.
- Kople frekvensomformeren mot PLS og skjerm for presentasjon av data.

1.3 Overordnet virkemåte for systemet

Blokkdiagrammet i figur 1.2 viser hvordan demomodellen er bygget opp. Den består av fire forskjellige hovedkomponenter; brukergrensesnitt gjennom PLS, frekvensomformer, motor og tilbakekobling via enkoder.

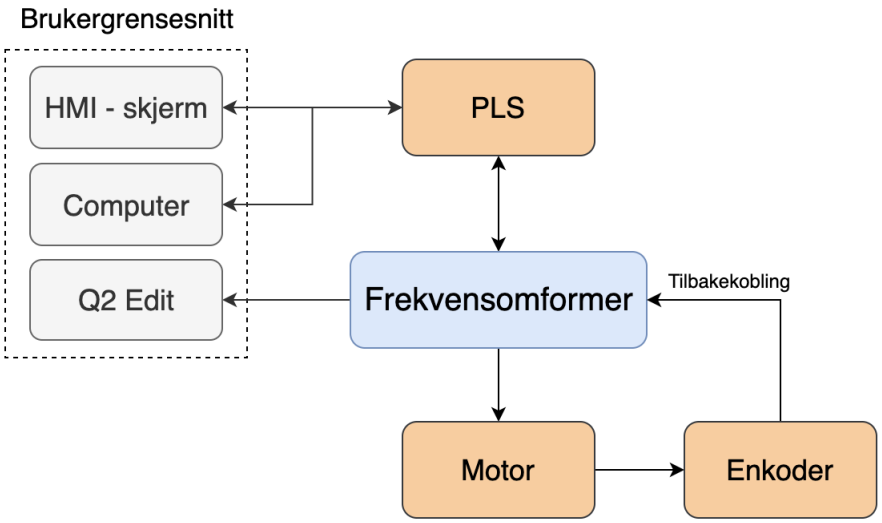


Figure 1.2: Overordnet virkemåte for systemet.

En strømforsyning mater den nødvendige strømmen til frekvensomformeren. Frekvensomformeren endrer frekvensen til motorspenningen for å regulere motorens hastighet. Denne prosessen skjer når det mottas signaler fra prosessen eller via brukeren gjennom brukergrensesnittet.

Når motoren roterer vil frekvensomformeren telle pulser fra enkoderen, denne tilbakekoblingen brukes videre i PLS programmene til å styre hastigheten og posisjonen til motoren.

Brukergrensesnittet gjør det mulig å overvåke driften av systemet, i tillegg til å endre enkelte variabler som videre integreres i programmer for systemet. Ut fra frekvensomformeren hentes det flere variabler som f.eks motorhastighet, strømtrekk og moment. Disse og flere andre variabler styres i programvaren Sysmac Studio fra Omron. Denne programvaren ble brukt til å lage koder for kalibrering og til å regulere vinsjsystemet. I tillegg har frekvensomformeren en egen programvare: Q2 Edit. Q2 Edit danner kommunikasjon mellom PC og Q2A frek. omf. via USB-kabel. Vi kan også hente ut (lese/overvåke) parametre og endre (skrive) parametre til frekvensomformeren fra Q2 Edit. Autotuning ble også gjort via denne programvaren.

1.4 Rapportens struktur

Rapporten er strukturert på følgende måte:

1. I kapittel 2 beskrives sentrale komponenter og programvare som ble brukt i oppgaven.
2. I kapittel 3 beskrives teori, løsning og resultat for kalibreringsprosessen.
3. I kapittel 4, 5 og 6 beskrives teori, løsning og resultater for de ulike reguleringene av systemet.
4. I kapittel 7 diskuteres utfordringer som forekom under arbeidet med oppgaven. I tillegg belyses forslag til forbedringer og videreutvikling av systemet, før bacheloroppgaven til slutt konkluderes.
5. Vedlegg A inneholder info om kommunikasjon mellom frekvensomformer og PLS.
6. Vedlegg B inneholder bruksanvisning for autotuning av frekvensomformeren.

Kapittel 2

Utstyr og programvare

Dette kapittelet gir en kort introduksjon til sentrale komponenter og programvare som danner grunnlaget for prosjektet. I tabell 2.1 gis en oversikt over hovedkomponentene til demomodellen.

Tabell 2.1: Oversikt over hovedkomponentene i demomodellen.

Komponent	**Type**	**Produsent**
PLS	NX102-1000	Omron
HMI skjerm	NA5-7W001S-V1	Omron
Frekvensomformer	Q2A-A2012-AAA	Omron
Motor	4AK 712-4 4-pole B14	Bevi
Enkoder	E6C2-CWZ1X	Omron

2.1 PLS

I figur 2.1 vises PLS-stasjonen som ble brukt i oppgaven. Denne består av PLS-en NX102-1000 fra Omron, som benyttes til å regulere vinsjsystemet. I tillegg til PLS er det en HMI-skjerm for visning av resultater og for å gi brukeren mulighet til å endre enkelte variabler gjennom dette brukergrensesnittet.

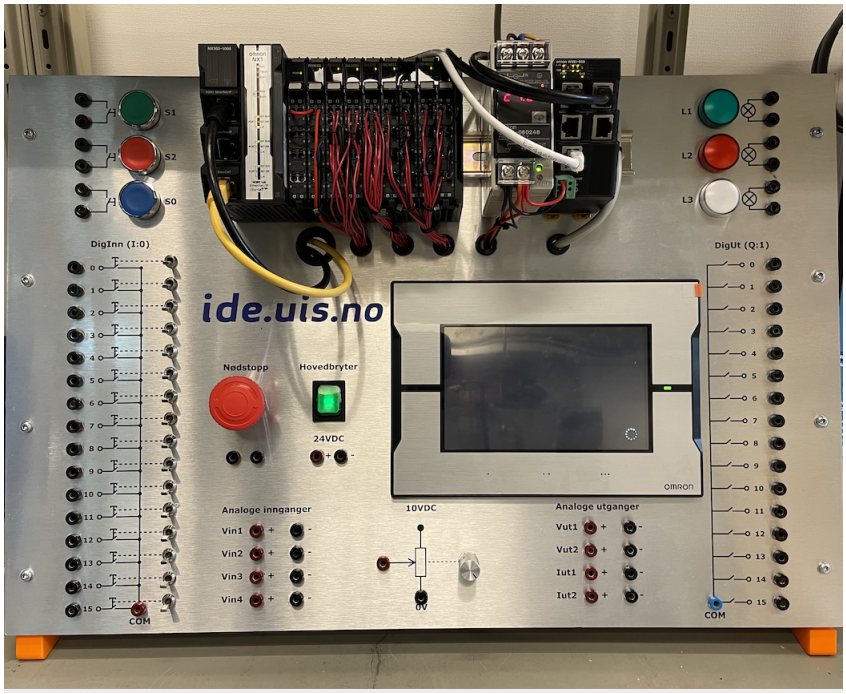


Figure 2.1: Oversiktsbilde av PLS og HMI-skjerm.

For å oppnå kommunikasjon mellom PLS-en og frekvensomformeren må EtherCAT benyttes. Q2A frekvensomformeren er ikke inkludert i standard Sysmac studio pakken. Det må legges til en ekstra pakke som innholder de nødvendige filene som gjør det mulig å implementere omformeren i programvaren. Dette gjøres som vist i vedlegg A.1. Når programvaren gjenkjenner frekvensomformeren kan den legges til i EtherCAT seksjonen i Sysmac studio og det kan hentes ut ønskede variabler som vist i figur 2.2.

Port	Des	R/W	Data Type	Variable
EtherCAT Network Configuration				
Q2A series				
6th receive PDO Mapping_Operation command_2000_01		W	UINT	operation_command
6th receive PDO Mapping_Speed reference/limit_2010_01		W	UINT	speed_ref_wright
38th receive PDO Mapping_Torque reference/limit_2020_01		W	INT	torq_ref_wright
6th transmit PDO Mapping_Drive status_2100_01		R	UINT	drive_status_read
6th transmit PDO Mapping_Output frequency_2110_01		R	UINT	output_frequency
38th transmit PDO Mapping_Output current_2120_01		R	UINT	output_current
38th transmit PDO Mapping_Motor speed_2200_01		R	UINT	motor_speed
39th transmit PDO Mapping_Output torque reference_2130_01		R	INT	torque_ref
39th transmit PDO Mapping_U6-18_2656_12		R	UINT	encoder_pulses

Figure 2.2: PDO (Process data object) mapping. I figuren vises et eksempel på variabler hentet ut fra frekvensomformeren. Kolonnen `Port` beskriver hvilken port variabelen hentes ut fra og variabelnavnet. Kolonnen `R/W` beskriver om det er en variabel det kan skrives til (W: Wright) eller om verdien til variabelen kun kan avleses (R: Read). I `Variable` kolonnen kan man navngi variablene for videre bruk i kode.

For å lettere kunne styre frekvensomformeren har Omron laget to funksjonsblokker `Q2x_MappingStd` og `Q2x_Control` som vist i figur 2.3.

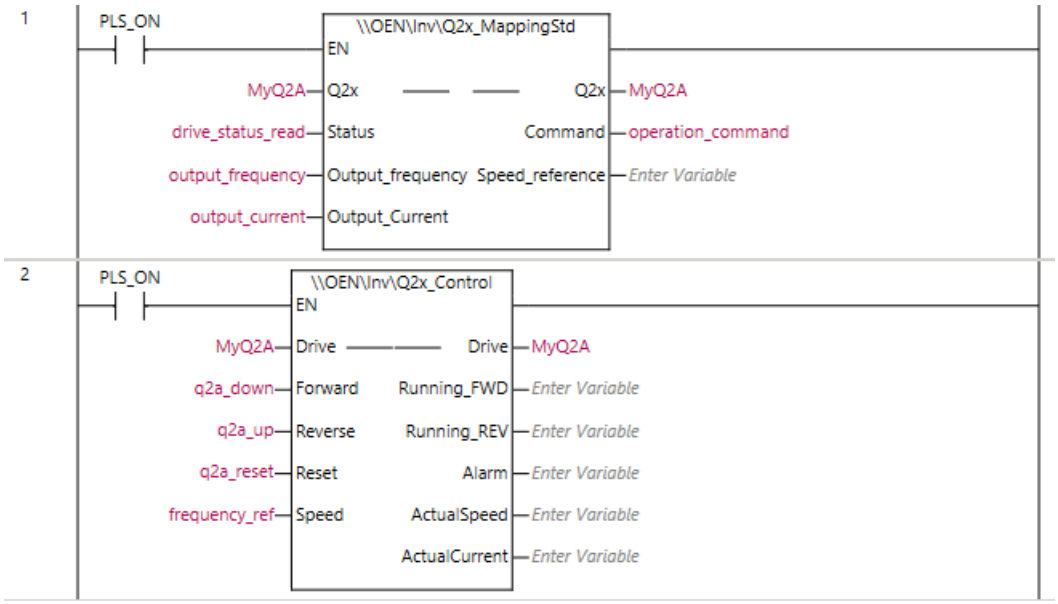


Figure 2.3: Ferdigprogrammerte funksjonsblokker for styring av Q2A frekvensomformer. Fra funksjonsblokken `Q2x_MappingStd` kan man lese av enkelte variabler fra frekvensomformeren. Gjennom funksjonsblokken `Q2x_Control` settes orientering og pådrag til motoren.

Når funksjonen `q2a_up` blir aktivert vil motoren kjøre i den retningen som fører til at vinsjsystemet heiser opp. Funksjonen `q2a_down` vil kjøre motoren motsatt vei i den retningen som fører til at vinsjsystemet senker ned / lårer. Ved justering av frekvensreferansen `frequency_ref` vil pådraget til motoren endres og disse må verdiene må være positive.

2.2 Q2A Frekvensomformer

Frekvensomformeren vist i figur 2.4 er produsert av Omron og modellen heter Q2A. Den er en type motorregulator som driver en motor ved å variere frekvensen og spenningen som blir levert til motoren. Under start og stopp av motoren kan også frekvensomformeren styre opp- og nedrampingen.

Q2-serie frekvensomformere har mulighet til å kontrollere forskjellige motor typer i både open og closed loop hastighet eller momentkontroll, dette vil forklares ytterligere i kap. 2.2.1. De har også kapasitet til å styre elektriske motortyper opp til 590 Hz. Kommunikasjonsmulighetene til frekvensomformeren er EtherCAT, EtherNet/IP, PROFINET eller POWERLINK. Q2A 200v klassen har et bredt effektområdet fra 0,55 til 110 kW, i tillegg til flere nyttige funksjoner som sanntidsklokke og mulighet for tilkobling til mobile enheter (via USB eller Bluetooth) [6].

Figure 2.4: Q2A Frekvensomformer som er brukt i oppgaven.

Frekvensomformersystemet består av en elektrisk forsyning, frekvensomformer og en vekselstrømsmotor som vist i figur 2.5. På innsiden av frekvensomformeren er det en likeretter, kondensator og vekselretter. Likeretteren vil konvertere vekselstrømmen (AC) til likestrøm (DC), signalet vil deretter filtreres av kondensatoren. Frekvensomformeren vil konvertere likestrømmen tilbake til vekselstrøm ved bruk av transistorer. Disse transistorene vil bruke pulsbreddemodulering (PWM) til å variere utgangsfrekvensen, noe som vil kontrollere hastigheten til motoren [1].

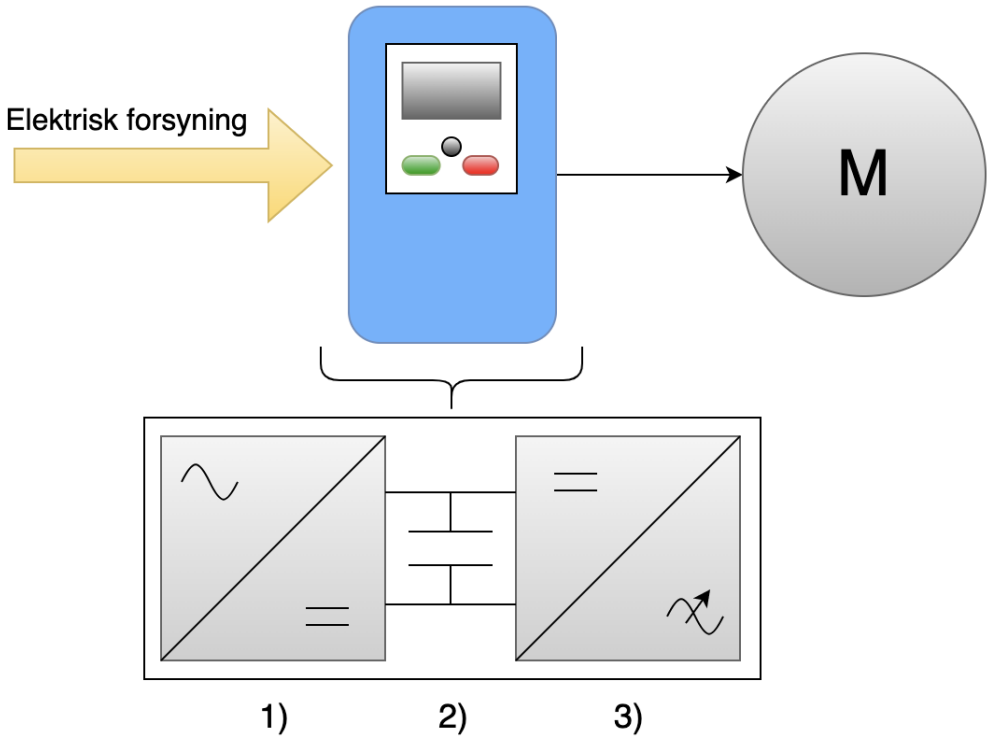


Figure 2.5: Frekvensomformeren består av 1) likeretter, 2) kondensator og 3) vekselretter.

Ved styring av frekvensen og spenningen som tilføres motoren har man en hastighetsregulator. Dette betyr at motorhastigheten er direkte relatert til frekvensen. Dersom det settes høyere frekvens vil motorens hastighet øke og dersom frekvensen synker vil også motorhastigheten bli lavere.

2.2.1 Vektorkontroll

Det er to typer vektorkontroll som brukes for styring av frekvensomformere; open loop og closed loop. Når en enkoder brukes til å gi tilbakemelding om akselposisjonen benyttes closed loop vektorkontroll. Ved vektorkontroll uten enkoder vil det ikke gis tilbakemelding om akselposisjonen, men i stedet brukes en matematisk modell utledet av motorens parametere, og dette kalles open loop vektorkontroll [3].

Heiser og kraner er ofte utstyrt med enkodere og derfor er closed loop vektorkontroll godt egnet. Det vil i denne type vektorkontroll benyttes enkoder og dens pulser til å gi en indikasjon om posisjonen til lasten.

Blokkskjema over closed loop vektorkontroll er vist i figur 2.6. Closed loop vektorkontroll fungerer på den måten at pulsene fra enkoderen (`encoder_pulses`) blir overvåket av frekvensomformeren via tilbakekoblingen. Disse pulsene brukes i en matematisk modell for utregning av motorhastigheten (`motor_speed`). Hastighetskontrollen vil sammenligne frekvensreferansen (`frequency_ref`) med motorens hastighet. Avviket mellom disse (`e_freq`) benyttes av en intern PI-regulator til å beregne et pådrag (`torque_ref`) som fungerer som en momentreferanse. Denne strukturen tilsvarer en kaskadestruktur.

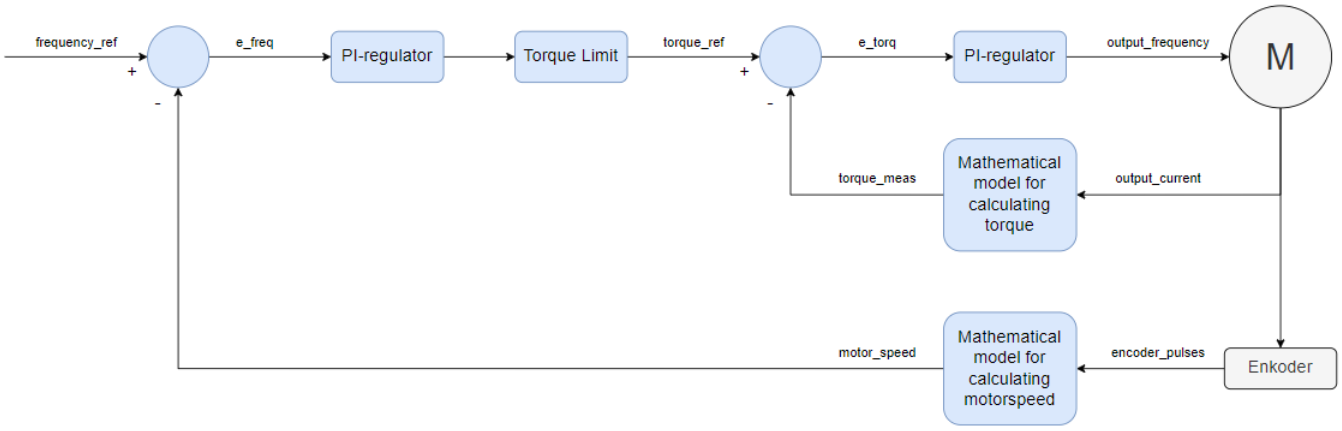


Figure 2.6: Generellt blokkskjema for closed loop vektorkontroll hvor blå er i frekvensomformer og grå er fysiske komponenter.

Videre vil momentreferansen sjekkes av funksjonen `Torque Limit`, som vil kutte signalet om det overstiger en maksverdi.

Målingen av strømforbruket gjennom motoren (`output_current`) brukes i en matematisk modell for utregning av moment (`torque_measured`). Denne målte momentverdien vil sammenlignes med momentreferansen og lage et reguleringsavvik (`e_torque`). Basert på dette avviket vil en innebygget PI-regulator i frekvensomformeren beregne et pådrag til motoren (`output_frequency`).

2.3 Motor

I figur 2.7 vises den asynkrone trefase motoren som ble brukt i prosjektet. En asynkron trefase-motor er en type vekselmotor som består av stasjonære og roterende komponenter (statorer og rotorer). Motoren har en effekt på 0,37-0,44kW, alt etter koblingsmetode på viklingene [2].

Figure 2.7: Motoren 4AK 712-4 4-pole B14 fra Bevi.

Viklingene i statoren er utført på en slik måte at det blir et dreiefelt i motoren når viklingene tilkobles en vekselstømskilde. Dette dreiefeltet vil indusere vekselspenninger i rotorviklingene. Disse gir rotorstrømmer som sammen med dreiefeltet gir krefter og dreiemoment slik at rotoren roterer sammen med magnetfeltet.

For at dreiefeltet skal kunne indusere vekselspenninger og dermed strøm og moment må det være en hastighetsforskjell mellom statorfeltet og rotor. Rotoren vil da alltid gå asynkront med statorens dreiefelt [8].

Motorens oppbygning er vist i figur 2.8.

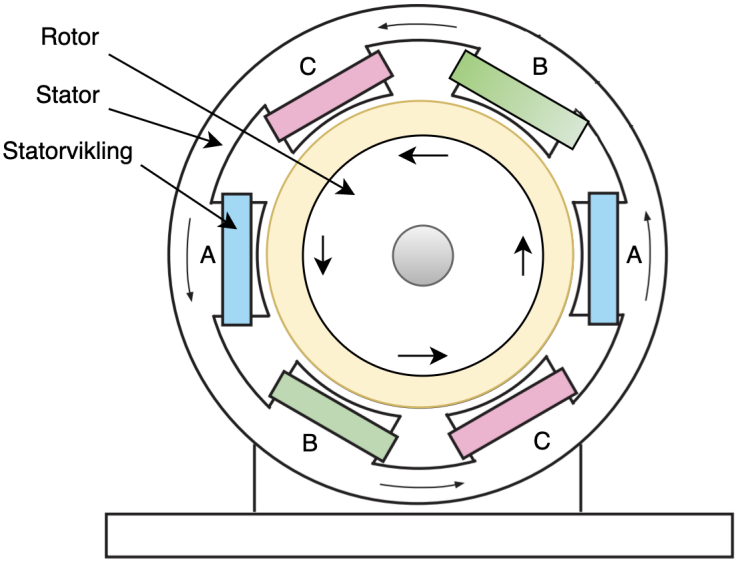


Figure 2.8: Rotor og stator i en asynkron motor.

Asynkronmotorer drives rundt av et roterende magnetisk felt, men ved høyere belastning på motoren får rotoren en langsommere hastighet enn det roterende feltet har, dette kalles sakking. Dreiemomentet vil da falle mot null og det vil dermed aldri oppnås synkront turtall, dette er grunnen til at denne type motor kalles asynkron.

2.4 Enkoder

Prinsippet til enkodere er vist i figur 2.9. Det genereres pulser ved rotasjon som gjør det mulig å utlede den totale vinkelrotasjonen ved å telle disse pulsene.

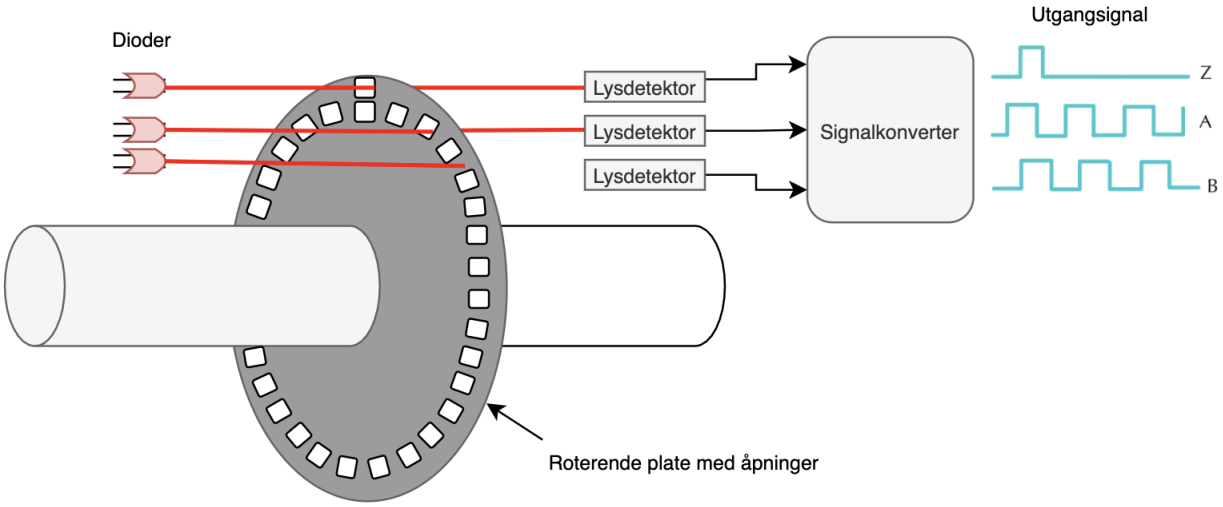


Figure 2.9: Prinsipp over hvordan det genereres pulser.

En inkrementell enkoder er en linær eller roterende elektromagnetisk enhet som har to utgangsignaler, A og B, disse vil avgi pulser ved rotering. Sammen kan A- og B-signalene indikere avstanden og retningen til bevegelsen. Mange inkrementelle enkodere har i tillegg et ekstra utgangsignal; typisk betegnet Z, som indikerer en viss referanseposisjon til enkoderen. Dette signalet vil kunne gjøre det mulig å indikere runder under rotasjon.

En inkrementell enkoder har tre utgangssignal som standard:

- Signal A: Pulser pr omdreining
- Signal B: Forskjøvet 90 grader i forhold til signal A.
- Signal Z: Initialsignal som signaliserer en omdreining

Enkoderen vist i figur 2.10 er produsert av Omron, og er brukt i oppgaven. Den benytter en høy-ytelses LED-indikator sammen med et mottakerelement til å avgi pulser. Det er oppgitt i databladet at enkoderen har en oppløsning på 2000 ppr (pulses pr revolution), men én omdreining er faktisk 8000 pulser [5].

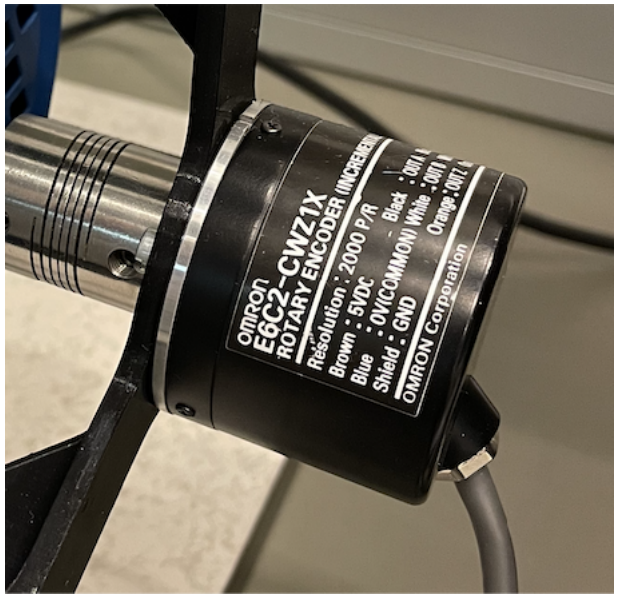


Figure 2.10: Enkoder: E6C2-CWZ1X fra Omron

Enkoderen er en inkrementell roterende enkoder som er festet mekanisk til akslingen på motoren. Den er koblet til frekvensomformeren som registrerer pulsene og rotasjonsendringen til motoren.

2.4.1 Q2A Signalbehandling av enkoder pulser

Pulsene fra enkoderen ble brukt til kalibrering og regulering av vinsjsystemet. Frekvensomformeren har en variabel som fungerer som en teller, den vil telle verdier opp til et ord på 16 bit. Det betyr at maksverdien til telleren er $2^{16} - 1 = 65535$, etter telleren når denne verdien vil den resette til null. Figur 2.11 viser oppførselen til parameteren når vi overvåker den i programvaren Q2 Edit.

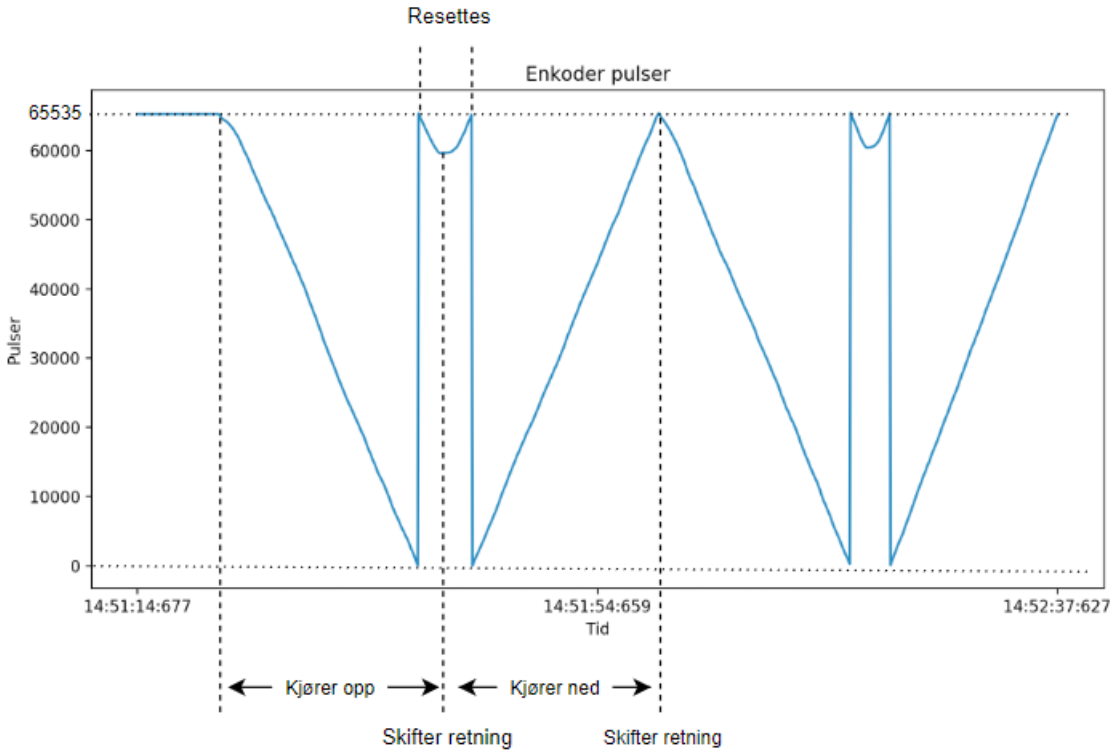


Figure 2.11: Oppførselen til enkoder puls telleren når vinsjsystemet kjøres opp og ned. Resettingen kan skje når motoren kjører i begge retninger.

Fra figur 2.11 ser man at når vinsjsystemet kjører opp vil telleren synke mot 0. I det pulstelleren blir 0 vil den resette til 65535 og fortsette å synke inntil det oppstår en retningsendring på motoren. Når det skjer en retningsendring vil pulstelleren telle i motsatt retning. Når pulstelleren når 65535 vil den resette til 0 og fortsette å telle oppover inntil det oppstår en retningsendring.

Pulsene fra enkoderen blir lagret i telleren i frekvensomformeren. Videre ble tellerverdien brukt til å få tilbakemelding om retning, hastighet og posisjon til motoren. I kodene til systemet tas det hensyn til at 8000 pulser representerer én omdreining.

2.5 Vinsjmodell

Vinsjmodellen vist i figur 2.12 ble gjort tilgjengelig fra instituttet. Den består av en vaiertrommel med tau som er festet til akslingen på motoren beskrevet i kap. 2.3.

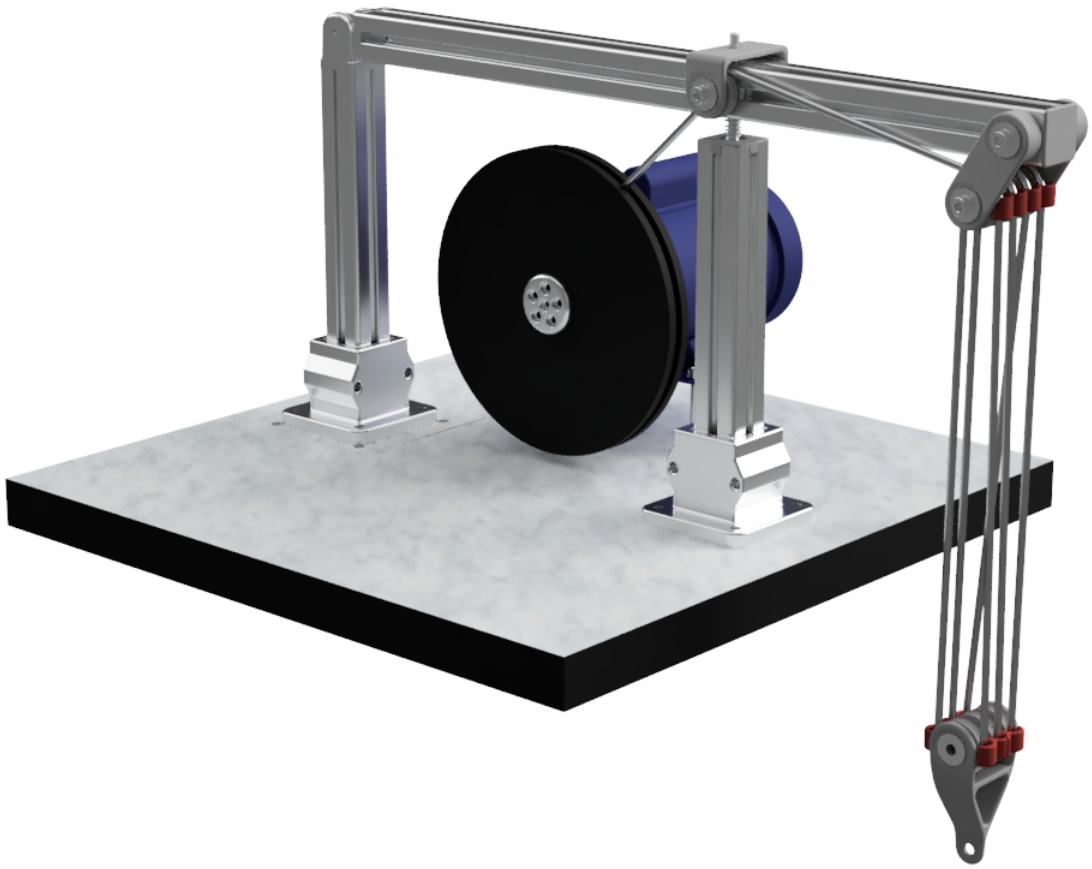


Figure 2.12: Vinsjmodell gjort tilgjengelig fra instituttet.

Den opprinnelige vinsjmodellen for heising og låring av en last besto av en klessnor festet over et ventilasjonsrør i taket [4].

Konstruksjonen ble forbedret med følgende endringer:

1. Motor ble byttet ut med ny hvor datablad var tilgjengelig.
2. Det ble laget en solid krankonstruksjon.
3. Ny 3d printet trommel og feste for tau.
4. Klessnor ble byttet med et 6mm elastisk tau.
5. Gamle trinser ble erstattet med seks 3d printede trinser.
6. Nytt 3d printet feste for lodd.

I figur 2.13 vises endringene. Markeringene i figuren tilsvarer de i listen over.

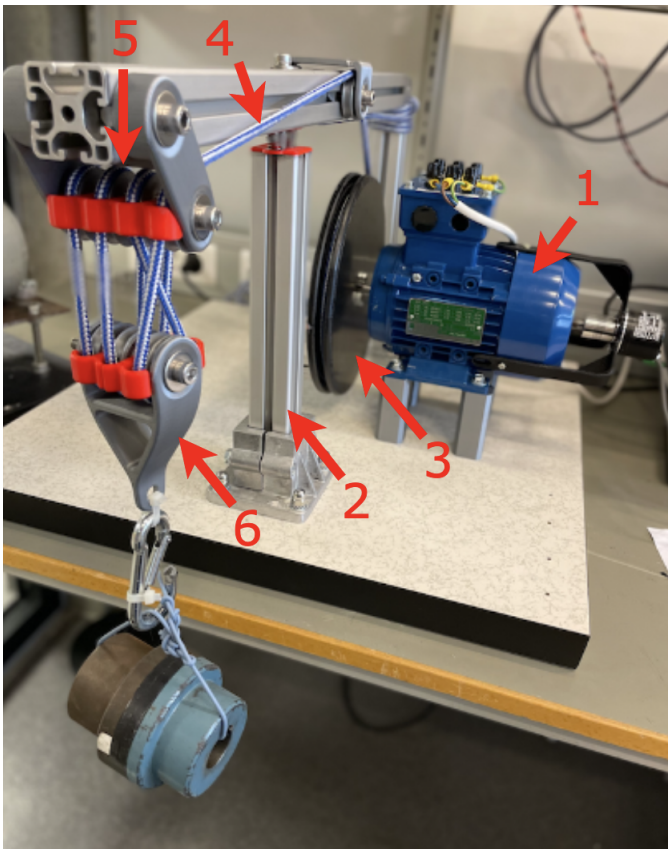


Figure 2.13: Figuren viser vinsjmodellens endringer. De nummererte pilene refererer til listen.

2.5.1 Trinser

På vinsjen er det montert 6 trinser. Trinsene blir brukt for å redusere mengden kraft som er nødvendig for å løfte en last ved å fordele vekten til denne lasten. Samtidig blir antall omdreininger for å flytte lasten 6 ganger så lang, som betyr at det kreves 6 ganger mer vaier på trommelen.

Det kompenseres for trinsene i utregninger i kodene. Hvis det ikke blir kompensert for trinsene vil utregningsverdien og den reelle verdien være ulik.

2.6 Programmvare

I denne bacheloroppgaven er det i hovedsak brukt to programvarer; Q2 Edit for overvåking av variabler knyttet til frekvensomformeren, og Sysmac Studio til å lage koder for kalibrering og regulering av systemet.

2.6.1 Q2 Edit

Q2 Edit er en programvare som lar brukeren overvåke variabler og endre parametere knyttet til frekvensomformeren Q2A. For å opprette kommunikasjon mellom programvaren og frekvensomformeren brukes USB-kabel. I figur 2.14 opprettes forbindelse mellom Q2A og Q2 edit.

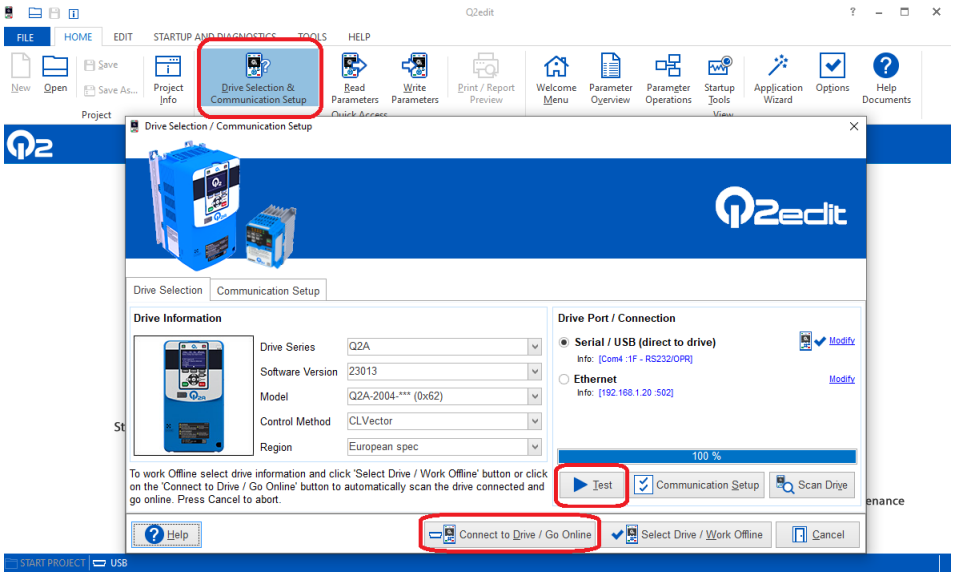


Figure 2.14: Oppretting av kommunikasjon mellom Q2A og Q2 edit; I de røde utringningene i figuren vises hva som trykkes på for å opprette kommunikasjon.

Tabell 2.2 viser innstillinger i frekvensomformeren som er brukt i oppgaven, og en beskrivelse av innstillingene er gjort under tabellen. Gjennom disse parameterene kan frekvensomformeren konfigureres etter ønsket behov, dette kan gjøres både direkte fra frekvensomformerens display eller gjennom programvaren Q2 Edit.

Tabell 2.2: Innstillinger i frekvensomformeren.

Nummer	**Parameter**	**Valgt innstilling**
A1-02	Control Method	3: CLVector
b1-01	Freq. Ref. Sel 1	3: Option PCB
b1-02	Run. Comm. Sel 1	3: Option PCB
C1-01	Accel Time 1	2.00 sec
C1-02	Decel Time 1	2.00 sec
F6-06	Trq Ref/Lim Comms	1
H1-01	DI1 Function Selection	13: Spd/Trq Switch

Nummerene i tabell 2.2 tilsvarer nummeringen på de forskjellige parameterene i frekvensomformeren. Videre forklares de forskjellige parameterene.

- A1-02 : Driftsmodus.

  Parameteren setter kontrollmetoden for systemet. Det er 8 mulige innstillinger for kontrollmetoden. Vi brukte closed-loop vektorkontroll og valgte dermed innstilling 3 (CLVector).

- b1-01 : Frekvensreferanse metode.

  Parameteren brukes til å velge metoden for setting av frekvensreferansen. Det er 4 forskjellige metoder som kan velges: 0: Keypad, frekvensreferansen kan kun settes gjennom frekvensomformerens display. 1: Analog input, frekvensreferansen kan settes gjennom analoge terminaler på frekvensomformeren. 2: Modbus, frekvensreferansen kan settes gjennom Modbus kommunikasjon. 3: Option PCB, frekvensreferansen kan settes via EtherCAT.

  Vi har EtherCAT basert kommunikasjon og derfor velges innstilling 3 (Option PCB).

- b1-02 : Run-signal metode.

  Parameteren setter metoden for setting av run-signalet. Det er 4 forskjellige metoder som kan velges. Disse metodene er de samme som for b1-01.

  Vi har EtherCAT basert kommunikasjon og derfor velges innstilling 3 (Option PCB).

- C1-01 : Rampetid ved start.

  Parameteren setter akselerasjonstid i sekunder. Det kan settes en verdi mellom 0 - 6000s. Vi satt innstillingen på 2 sekunder for å unngå bråstart (som ved 0 sek).

- C2-02 : Rampetid ved stopp.

  Parameteren setter retardasjonstid i sekunder. Det kan settes en verdi mellom 0-6000s. Vi satt innstillingen på 2 sekunder for å unngå bråstopp.

- F6-06 : Momentreferanse kommunikasjon.

  Parameteren setter funksjonen som aktiverer og deaktiverer muligheten til å endre momentreferansen. Innstillingen ble satt til 1 (enable).

- H1-01 : DI1 funksjon.

  Parameteren setter funksjonen for digital terminal 1 (DI1). Det er 65 forskjellige mulige innstillingsvalg, alt etter hva en ønsker den digitale terminalen skal foreta. Vi satt innstillingen til 13 (Spd/Trq Switch) som videre ble brukt til å momentregulere systemet.

Tabell 2.3 viser motorvariabler som er brukt i oppgaven. Disse ble avlest og endret etter behov i kode for reguleringsprosessene.

Tabell 2.3: Motorvariabler fra frekvensomformeren.

Nummer	**Variabel**	**Variabelnavn i programmer**
U1-01	Frequency Reference	`frequency_ref`
U1-02	Output Frequency	`output_frequency`
U1-03	Output Current	`output_current`
U1-05	Motor Speed	`motor_speed`
U1-09	Torque Reference	`torque_ref`
U4-52	Torque Ref from Comm	`torq_ref_wright`
U6-18	Encoder Pulse Counter	`encoder_pulses`

Nummerene i tabell 2.3 tilsvarer nummereringen i frekvensomformeren. Videre forklares de forskjellige motorvariablene.

- U1-01 : Frekvens referanse. Viser den satte frekvens referansen (Hz).
- U1-02 : Utgangsfrekvens. Viser den faktiske utgangsfrekvensen (Hz).
- U1-03 : Utgangstrøm. Viser den faktiske utgangstrømmen (A).
- U1-05 : Motorhastighet. Viser den faktiske hastigheten til motoren (Hz).
- U1-09 : Momentreferanse. Viser moment referansen (%).
- U4-52 : Satt momentreferanse. Viser momentreferansen skrevet til frekvensomformeren.
- U6-18 : Pulsteller. Viser antall telte pulser fra enkoderen.

2.6.2 Autotuning

For at frekvensomformeren skal vite hvilken motor den er koblet til kjøres det en autotuning. En riktig autotunet frekvensomformer vil være mer effektiv (lavere strøm for samme dreiemoment) og gi bedre ytelse (mer lineær og stabil drift). Autotuningen går ut på at frekvensomformeren kjører gjennom en rekke frekvens og strømverdier. Autotuningen kan utføres direkte via displayet til frekvensomformeren eller gjennom programvaren Q2 Edit. Resultatet fra autotuning av frekvensomformeren er vist i figur 2.15.

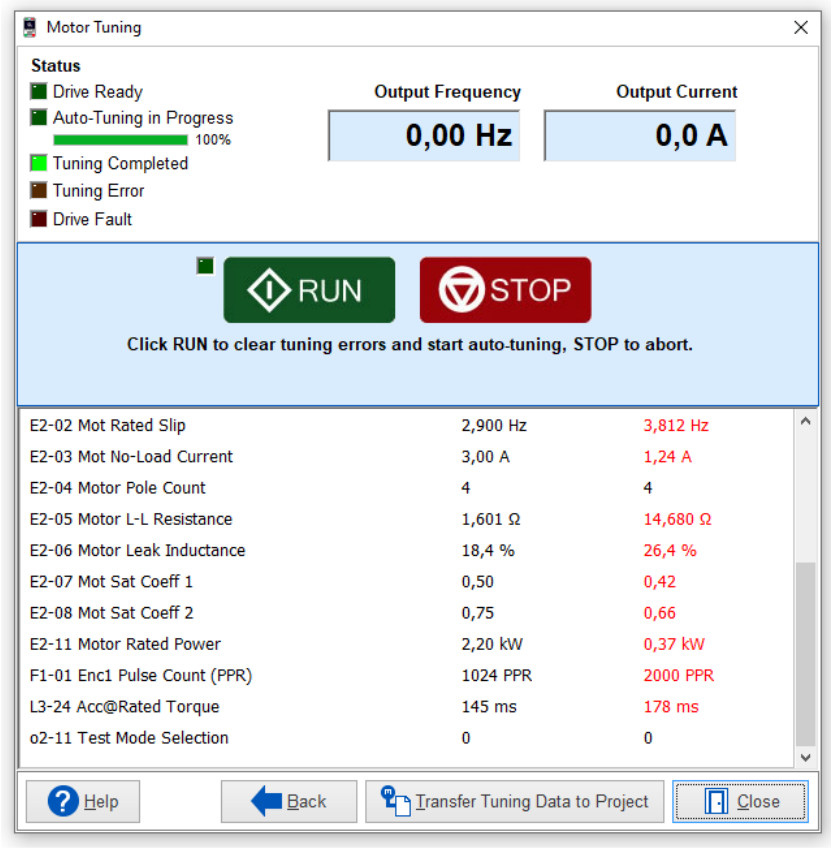


Figure 2.15: Resultater fra autotuning. Svart: default-verdier for motorparametere, Rød: nye verdier for motorparametere etter autotuning.

2.6.3 Sysmac Studio

Programmvaren Sysmac Studio fra Omron ble benyttet til å programmere PLS-koder for å kalibrere og regulere systemet. Det ble programmert kode i ladder og strukturert tekst og laget skjermer til HMI skjermen gjennom denne programmvaren.

Kapittel 3

Kalibrering

I dette kapittelet presenteres teori, løsning og resultater for kalibrering av vinsjsystemet.

Før det kan posisjonsjusteres eller spoles inn fra en posisjon må systemet kalibreres. Det ble utviklet en kalibreringsfunksjon, som ved aktivering kjører fra start i topp og ned til bunn for å beregne et høydeområde vinsjen kan kjøre mellom. Dette kalibrerte området fra topp til bunn settes som en begrensning for hvor mye vaier som kan spoles inn eller ut i programmene.

Hvordan kalibreringsprosessen er bygget opp er vist i pseudokoden under.

Kodeutdrag 3.1: Pseudokode kalibreirng

```
// 1. Venter til bruker trykker inn kalibrering knapp
// 2. Knapp trykket inn -> Kjører loddet til topposisjon, stoppes av høyt moment
// 3. Kjører ned til bunn, stoppes av lavt moment
// 4. Teller antall pulser avgitt fra enkoderen på vei ned
// 5. Kjører opp igjen, stoppes ved topp av det kalibrerte området
// 6. Konverterer de telte pulsene til lengde i mm.
// 7. Henger i vente tilstand i topp ved fullført kalibrering
```

3.1 Teori

For å beregne lengden med vaier som er spolt ut fra trommelen må formelen til omkretsen av en sirkel brukes, vist i likning 3.1.

$$Omkrets(n) = 2\pi \cdot r_{total}(n) \tag{3.1}$$

Den totale radiusen vil variere etter hvor mange runder (n) med vaier det er på trommelen som vist i likning 3.2. Diameteren til vaieren må brukes i utregningen av den totale radiusen.

$$r_{total}(n) = r_{trommel} + (d_{vaier} \cdot n) \tag{3.2}$$

Som illustrert i figur 3.1 vil den totale radiusen variere som en funksjon av antall runder med vaier på trommelen.

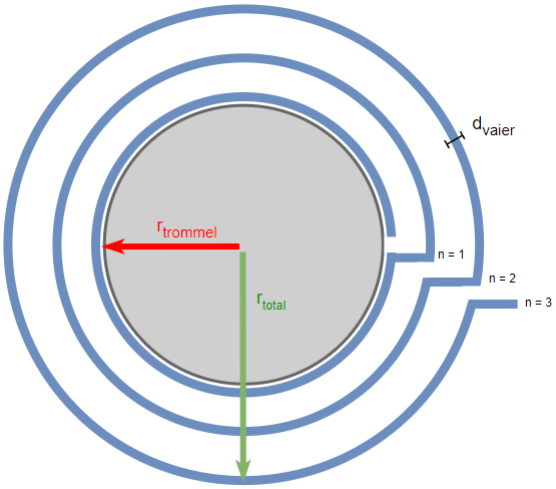


Figure 3.1: Total radius (r_{total}) endres som funksjon av runder (n) med vaier. Hvor $r_{trommel}$ er radiusen til trommelen uten vaier og d_{vaier} er diameteren til vaieren.

Trommelen til demomodellen er like bred som diameteren på vaieren, det fører til at hver runde med vaier som spoles inn vil legge seg over den forrige og ikke ved siden av. Bredden til trommelen trenger ikke bli tatt hensyn til i utregningen for dette systemet.

Omkretsen blir en funksjon av antall runder med vaier på trommelen som vist i likning 3.3.

$$Omkrets(n) = 2\pi \cdot (r_{trommel} + (d_{vaier} \cdot n)) \tag{3.3}$$

Ved installasjon av trommel ble det funnet ut at trommelen holder 16 runder med vaier når lasten henger i topposisjon ($n_{max} = 16$). Når systemet starter å spole ut vil omkretsen gradvis reduseres på grunn av den varierende totale radiusen. Vaierlengde som blir spolt ut fra trommelen tilsvarer den reduserte omkretsen fra topposisjonen, som vist i likning 3.4.

$$Vaierlengde(x) = 2\pi \cdot (r_{trommel} + (d_{vaier} \cdot (n_{max} - x))) \tag{3.4}$$

Hvor det er kjent at vaier diameteren er 6mm og radiusen til trommelen uten vaier er 25mm.

$$Vaierlengde(x) = 2\pi \cdot (25 + (6 \cdot (16 - x))) \tag{3.5}$$

Funksjonen til likning 3.5 vises i figur 3.2.

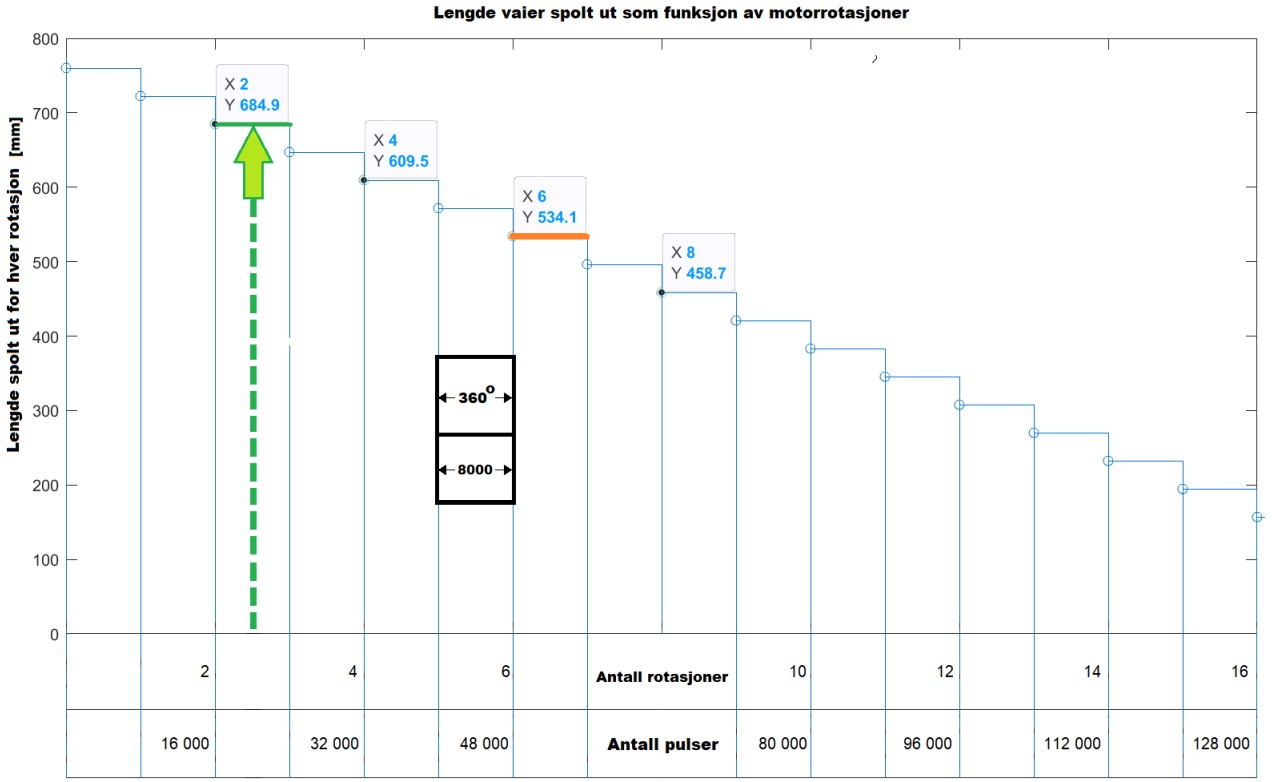


Figure 3.2: x-aksen viser antall rotasjoner motoren har spolt ut og antall pulser avgitt fra enkoderen. Vaierlengde som spoles ut reduseres ved antall runder som spoles ut. x-aksen vil øke med en runde når motoren har rotert 360 grader, som tilsvarer 8000 pulser avgitt fra enkoderen som vist i den markerte svarte boksen.

Fra figur 3.2 ser vi at motoren spoler ut mest vaierlengde når x=0 (lasten henger i topposisjon). Når motoren har spolt ut en runde vil rundene med vaier som er på trommelen reduseres med én ($n_{max} - x \rightarrow 16 - 1 = 15$), og ved neste omdreining vil mindre lengde med vaier spoles ut. Variabelen "x" (antall rotasjoner spolt ut) vil øke eller minke når det er telt 8000 pulser fra enkoderen (i samme retning).

I figur 3.3 illustreres lengden fra figur 3.2 når x=6 (oransje farge) som omkrets på trommelen.

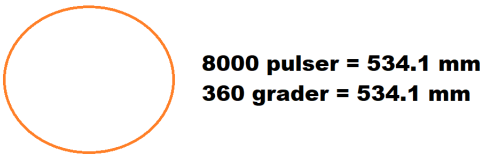


Figure 3.3: Vaierlengde(x) kan representeres som pulser avgitt fra enkoderen eller i grader. Når x=6 i figur 3.2 vil 8000 pulser (1 omdreining) tilsvare en lengde som spoles ut på 534,1 mm.

En puls endring fra enkoderens telleverdi er den minste endringen vi kan detektere som dermed er oppløsningen til systemet. Lengden i mm som denne endringen representerer er avhengig av hvor mange runder som er spolt ut av fra motoren.

Fra figur 3.3 får vi sammenhengen:

$$8000 \ \ pulser = 534,1 \ \ mm = 360^{ø}$$

Det betyr at hver puls fra enkoderen tilsvarer $\frac{360 grader}{8000 pulser} = 0,045$ grader og $\frac{534,1 mm}{8000 pulser} = 0,0066$ mm når det er spolt ut 6 runder med vaier fra trommelen (x=6).

Likning 3.4 gir en vaierlengde som spoles ut eller inn for den aktuelle runden. Distansen lasten faktisk beveger seg er 6 ganger mindre pga. trinsene i systemet. Det kompenseres for trinsene og distansen lastens faktisk beveges blir som vist i likning 3.6.

$$Lastdistanse(x) = \frac{Vaierlengde(x)}{6} = \frac{2\pi \cdot (25 + (6 \cdot (16 - x))}{6} \tag{3.6}$$

Funksjonen til likning 3.6 vises i figur 3.4.

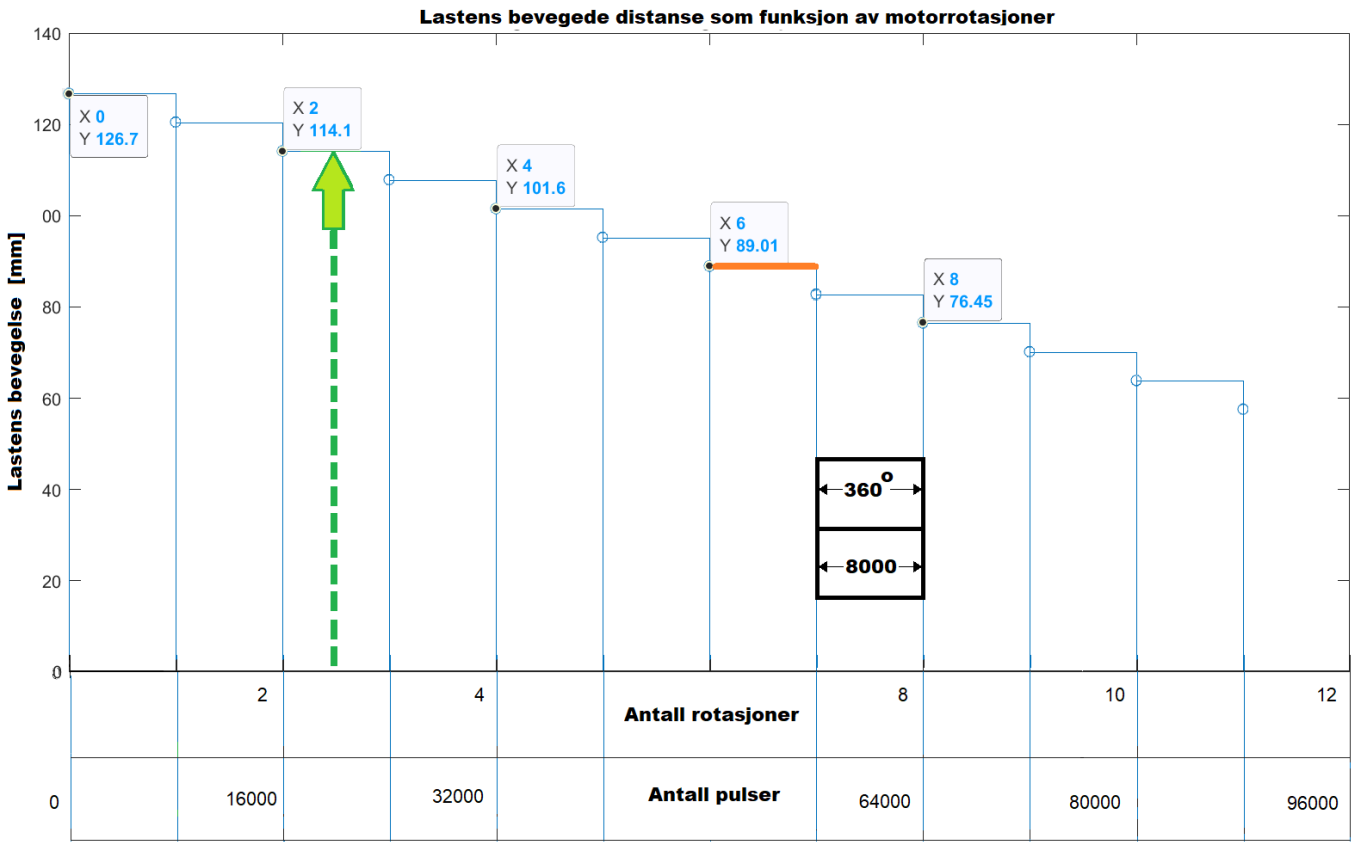


Figure 3.4: Lastens bevegede distanse som funksjon av motorrotasjoner. Den oransje lengden fra figur 3.2 er her vist som den distansen lasten beveger seg .

Ved den grønne pilen i figur 3.2 og 3.4 er motoren dreid 2,5 runder fra topposisjonen, som tilsvarer totalt 20 000 pulser avgitt fra enkoderen. Trekkes det ifra antall pulser når x=2 (16 000 pulser) får vi 4000 pulser.

I figur 3.5 vises det hva som skal til for at rundene som er spolt ut (x) skal reduseres eller økes med én ut ifra motorens posisjon ved den grønne pilen.

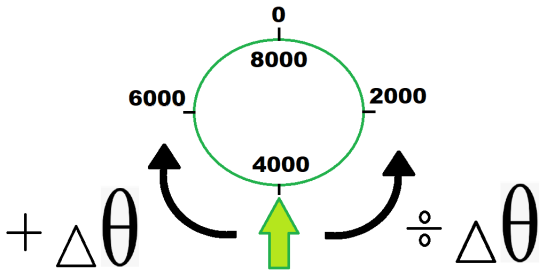


Figure 3.5: Grønn pil fra figur 3.2 indikerer hvor mye motoren har dreid fra topposisjon. $\Delta\theta$ er vinkelendringen til motoren i antall pulser fra enkoderen eller i grader (8000 pulser = 360 grader).

Fra figur 3.5 vises det at vinkelendringen ($\Delta\theta = \Delta pulser$) som må til på motoren for at antall runder spolt ut fra trommelen (x) skal oppdateres, er 4000 pulser fra enkoderen. Enten i positiv retning som medfører at det har blitt spolt ut en runde mer med vaier (x økes med en), eller i negativ retning som medfører at det er spolt inn en runde med vaier (x reduseres med en).

Det totale distansen som lasten har beveget seg kan skrives som summen av antallet rotasjoner (x) motoren gjør for å senke lasten ned til bunnposisjonen. Den totale distansen blir som vist i likning 3.7.

$$Total\ lastdistanse(x) = \sum_{x=0}^{16} Lastdistanse(x) = \sum_{x=0}^{16} 2\pi \cdot (r_{trommel} + [d_{vaier} \cdot (16 - x)]) \quad (3.7)$$

Funksjonen til likning 3.7 er vist i figur 3.6.

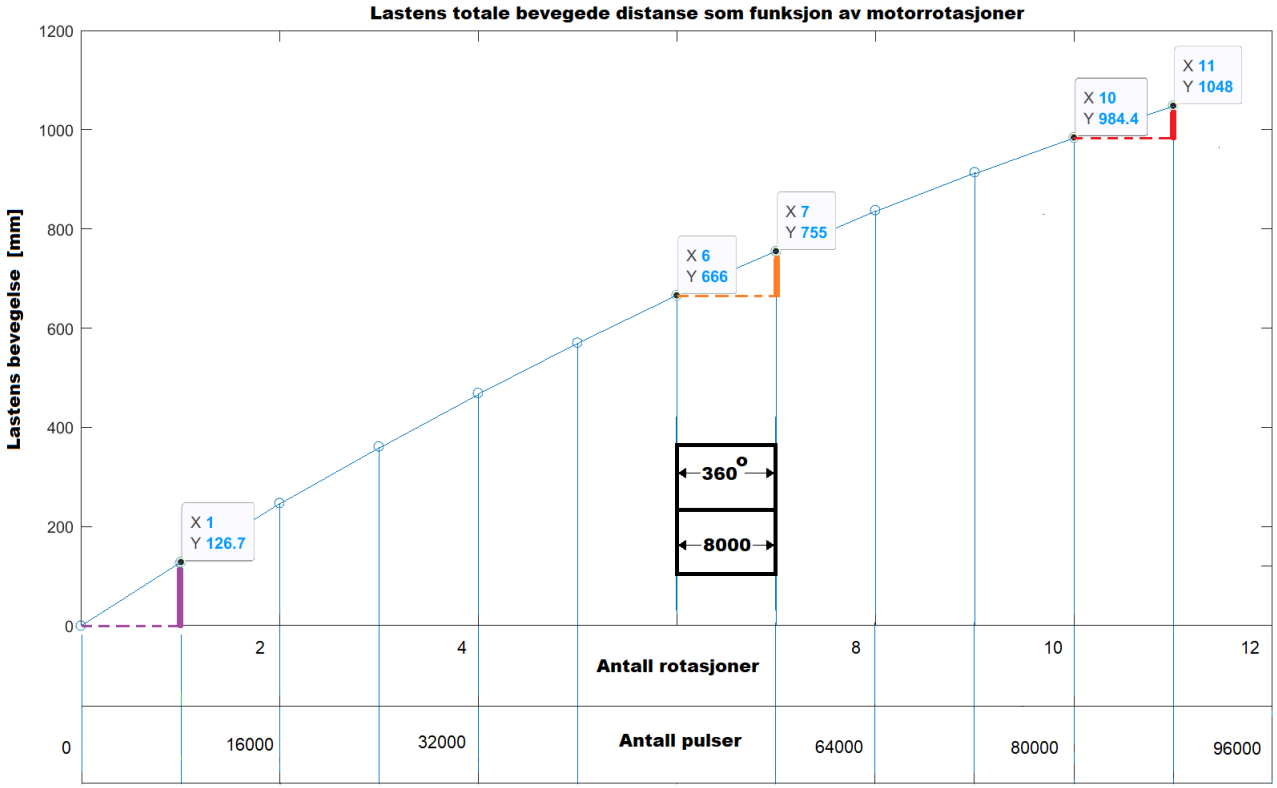


Figure 3.6: Lasten endring i posisjon som funksjon av motorrotasjoner. Den oransje lengden fra figur 3.2 og figur 3.4 her representert med en oransje vertikal linje. Det er distansen som lasten beveger seg.

Fra figur 3.4 vises det at distansen lasten beveges pr. omdreining på motoren er størst når lasten er i topp (x=0) og blir redusert for hver runde med vaier som spoles ut. Lasten blir senket ned 126,7mm når motoren gjør første rotasjon ved den lilla vertikale linjen (x=0 → x=1), mens ved den røde vertikale linjen (x=10 → x=11) senkes den bare 63,6mm.

Plottene 3.2 og 3.4 kan verifiseres ved å bruke dataene fra dem i likningene. Den oransje lengden fra 3.2 brukt i likning 3.5:

$$Lastdistanse(x=6) = \frac{Vaierlengde(x=6)}{6} = \frac{534,1}{6} = 89,01mm \tag{3.8}$$

Vi sjekker denne verdien med verdiene oppgitt for den totale distansen som lasten beveger seg i figur 3.6 (stiplede oransje linje).

$$755 - 666 = 89mm$$

Med dette vet vi at likningen for distansen som lasten beveger seg kan knyttes til antall motor-rotasjoner gjort fra topposisjonen.

3.2 Kalibreringsprosess

Vi har et trommel system der det er kjent at den totale radiusen på trommelen vil endres som funksjon av hvor mange runder med vaier som er spolt inn eller ut. I figur 3.7 illustreres det kalibrerte området fra topposisjonen (h0) til bunnposisjonen (h100). Fra figuren ser man at det er en ulineær endring i runder over området.

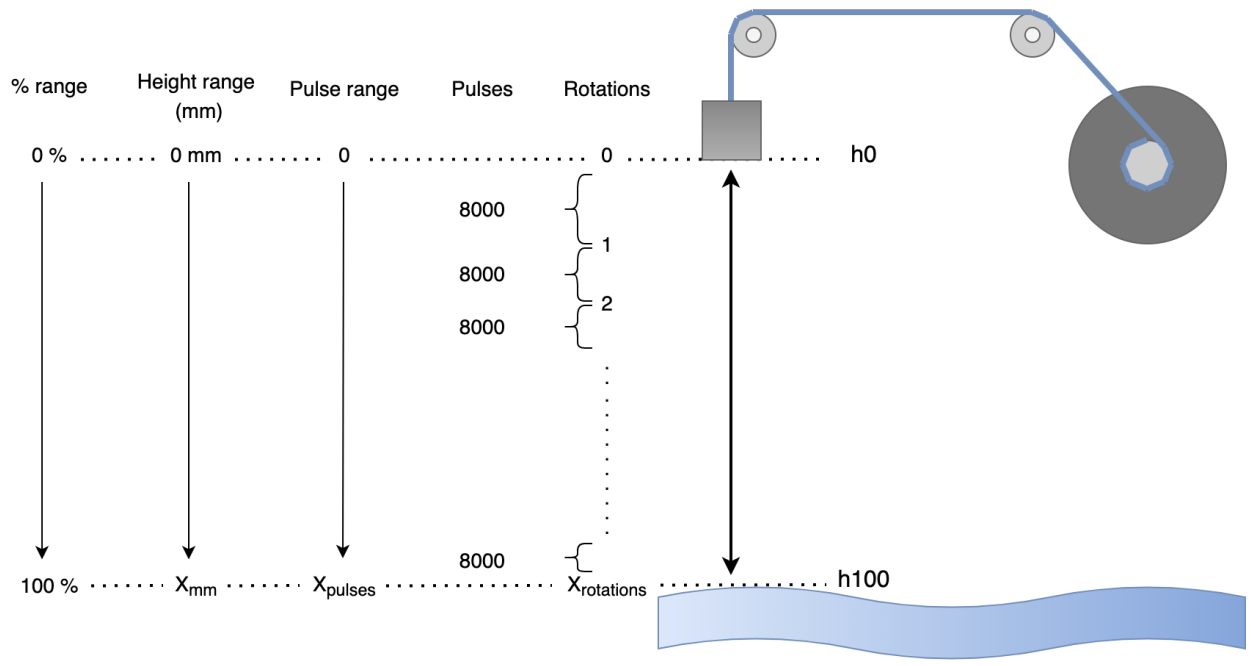


Figure 3.7: I figuren vises det kalibrerte området mellom topposisjon (h0) og bunnposisjon (h100). Det lages områder for lengde (mm), antall pulser avgitt fra enkoderen og antall runder.

Når motoren kjører nedover vil frekvensomformeren telle pulser fra enkoderen. Differansen mellom nåværende og forrige tellerverdi gir vinkel endringen til motoren.

Alle differanseverdiene adderes og gir et område i antall pulser avgitt fra enkoderen. Pulsene brukes videre til å finne et område for lengden i millimeter og området for antall runder som spoles ut. Området for rundene beregnes ved å dele tellerverdiområdet på 8000 pulser (1 omdreining).

Ved å bruke likningen 3.7 fra tidlige, kan lengden i millimeter fra topposisjonen h(0) og ned til bunnposisjonen h(100) beregnes.

3.2.1 Beregning av rotasjonsendringen til motoren

Pulsene avgitt fra enkoderen som blir telt av variabelen `encoder_pulses` i frekvensomformeren brukes til å beregne rotasjonsendringen til motoren. Differansen mellom nåværende tellerverdi og forrige tellerverdi gir rotasjonsendringen til motoren.

Rotasjonsendringen ble beregnet i programmene som vist i kodeutdraget 3.2. Det blir hentet en tellerverdi som representerer den nåværende tellerverdien (`encoder_value_now`), deretter regnes det ut en differanse (`encoder_difference_value`) mellom denne tellerverdien og den forrige. Den forrige tellerverdien settes til slutt lik den nåværende (`encoder_value_old`).

Kodeutdrag 3.2: Utdrag fra kode.

```
encoder_value_now = encoder_pulses
encoder_difference_value = encoder_value_now - encoder_value_old
encoder_value_old = encoder_value_now
```

Det blir generert en postiv (+) differanseverdi når motoren kjører nedover og negativ (-) differanseverdi når motoren kjører oppover. Det vil oppstå ett tilfelle for differanseverdien som må tas hensyn til når frekvensomformerens teller blir resatt. Som nevnt i kapittel 2.4.1 resettes telleren når den overskrider en verdi på 65535 når den kjører oppover og den resettes ved 0 når den kjører nedover.

I figur 3.8 illustreres tilfellet når motoren kjører oppover. Vinsjsystemet kjører nedover og differansen mellom tellerverdiene er positiv på hele strekket mellom bitverdi 0 og bitverdi 65535. I det øyeblikket det regnes ut en differanseverdi mellom siste tellerverdi før og tellerverdien etter resetting blir utregningen feil.

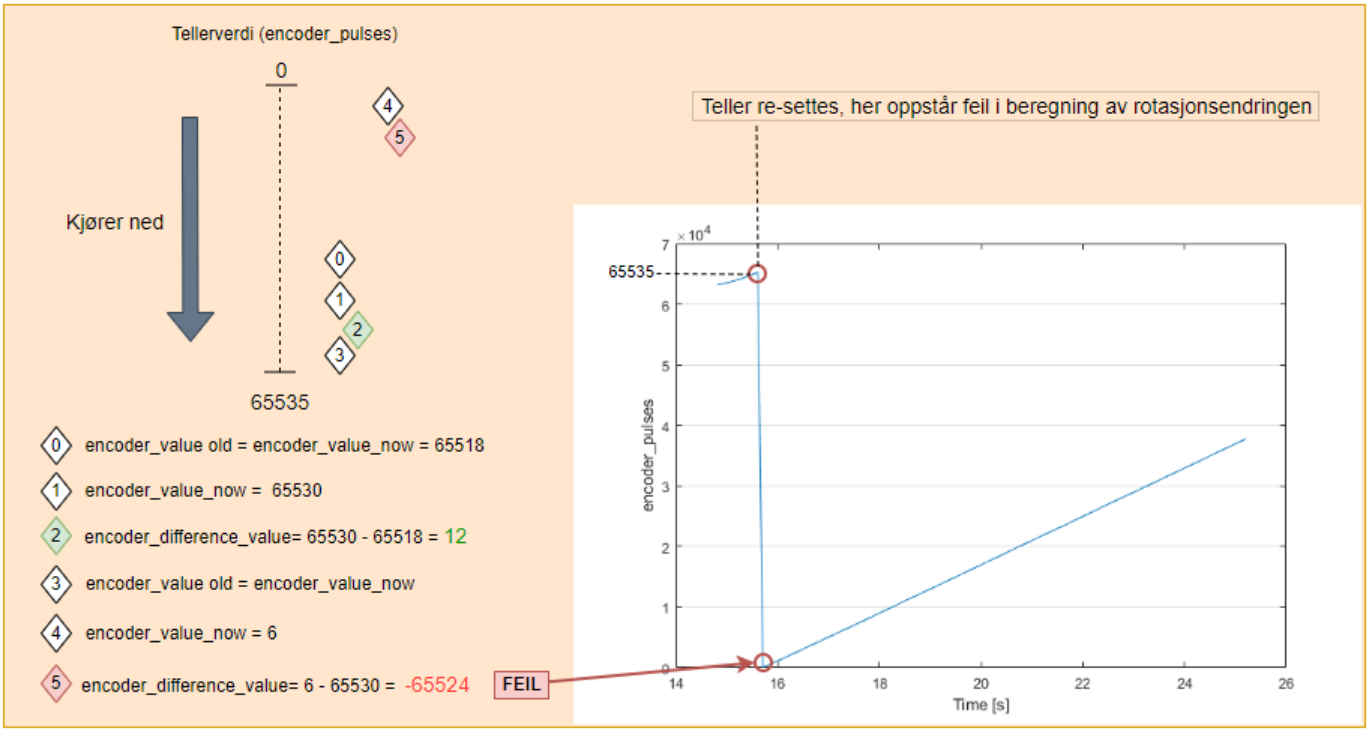


Figure 3.8: Figuren viser hvordan det oppstår feil i utregning av differansen mellom tellerverdiene. Feilen oppstår i det punktet når telleren resettes (markert med røde ringer).

Vi ender opp med en differanseverdi som har feil fortegn og en verdi som er veldig høy (markert med rød skrift i figuren). Ettersom pådraget og motororienteringen er uendret skulle differansen mellom tellerverdiene vært lik det den var før resetting (= 12).

Tilfellet løses i programmene som vist i kodeutdraget 3.3. Programmet spør om differanseverdien (`encoder_difference_value`) er mindre enn 0 når motoren kjører vinsjsystemet nedover. Hvis ja; verdien ganges med -1 for å endre fortegnet. I tillegg spør programmet om differanseverdien er større enn 1000 som vil indikere en urealistisk høy verdi. Hvis ja; det trekkes det fra 65535 (maksverdien til telleren) for å ordne den høye verdien.

Kodeutdrag 3.3: Utdrag fra kode

```
// Difference value logic if Q2A resets while driving down
IF encoder_difference_value < 0 THEN
// Handles the orientation fault
   encoder_difference_value := encoder_difference_value * (-1);
// Handles the very large magnitude
   IF ABS(encoder_difference_value) > 1000 THEN
      encoder_difference_value := 65535 - encoder_difference_value;
   END_IF;
END_IF;
```

Denne situasjonen vil oppstå når motoren kjører vinsjsystemet i begge retninger. Logikken må også ta hensyn til resetting når vinsjsystemet kjøres opp. Det løses i programmene som vist i kodeutdraget 3.4. Programmet spør om differanseverdien (`encoder_difference_value`) er større enn 0 når det kjøres oppover. Hvis ja; verdien ganges med -1 for å endre fortegnet. I tillegg spør programmet om differanseverdien er større enn 1000. Hvis ja; det legges til 65535 (maksverdien til telleren) for å ordne den høye verdien.

Kodeutdrag 3.4: Utdrag fra kode

```
// Difference value logic if Q2A resets while driving up
IF encoder_difference_value > 0 THEN
// Handles the orientation fault
   encoder_difference_value := encoder_difference_value * (-1);
// Handles the very large magnitude
   IF ABS(encoder_difference_value) > 1000 THEN
       encoder_difference_value := 65535 + encoder_difference_value;
   END_IF;
END_IF;
```

3.3 Kode for kalibrering

For å vise hvordan kalibreringsprossesen er bygget opp har vi i figur 3.3 laget et tilstandsdiagram som viser hovedstrukturen til dette programmet.

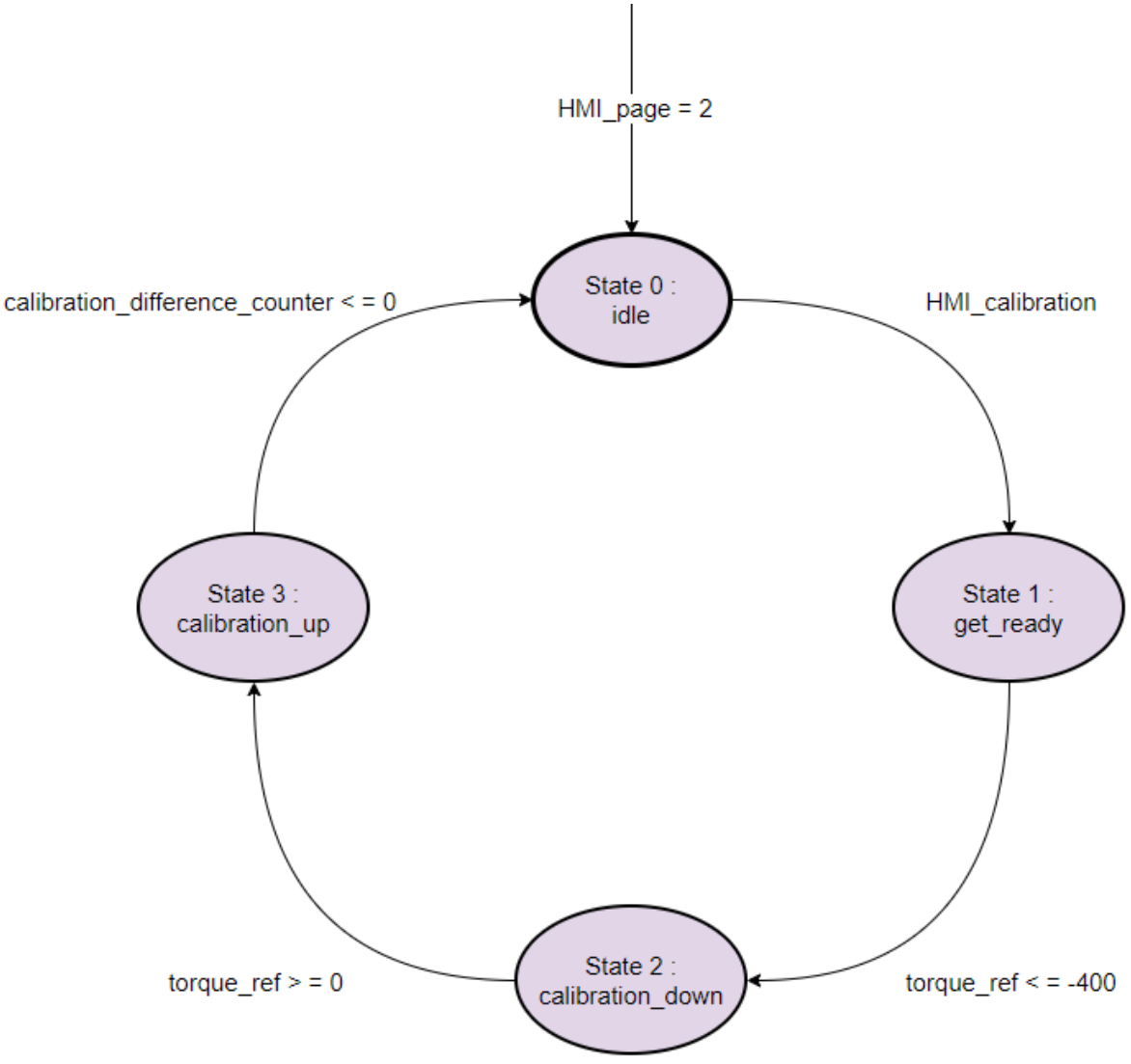


Figure 3.9: Tilstandsdiagram for programmet. Beskrivelse av tilstandene blir gitt videre i delkapittelet.

Videre forklares hva som utføres i de ulike tilstandene og betingelsene for flytting mellom disse. I tillegg vises utdrag fra kodene til tilstandene.

Tilstand 0 : Dette er ventetilstanden til programmet. Når brukeren trykker seg inn på kalibreringssiden på HMI skjermen (`HMI_page = 2`) vil programmet gå til tilstand 0 (`idle`). I linje 3-5 i kodeutdrag 3.5 settes initial verdiene, som består av vaier diameteren (`wire_diameter`), trommelradius (`drum_radius`) og antall trinser (`number_of_pulleys`) i systemet.

I linje 13 spør programmet om kalibrering er gjennomført (`done`); ved JA skal lasten bli hengende i ro med null pådrag (`frequency_ref`) som vist i linje 14, ved NEI skal det være mulig å starte kalibrering. `done` settes til `TRUE` ved fullført kalibrering i tilstand 3.

For at programmet skal flytte seg videre til tilstand 1 (`get_ready`) er det satt et krav om at brukeren må ha aktivert knappen `HMI_calibration` for å starte kalibrering i linje 17.

Kodeutdrag 3.5: Utdrag fra kode; tilstand 0

```
1  IF HMI_page = 2 THEN
2  // Sets initial values
3     wire_diameter:= 6;
4     drum_radius:= 25;
5     number_of_pulleys = 6;
6
7  CASE state OF
8  // State 0; idle
9     idle:
10
11 // System already calibrated?
12 // Yes -> Standby
13       IF done = TRUE THEN
14          frequency_ref := 0;
15 // NO -> Possible to start calibration by activating HMI button
16       ELIF done = FALSE THEN
17          IF HMI_calibration = TRUE THEN
18             state := get_ready;
19          END_IF;
20       END_IF;
```

Tilstand 1 : I denne tilstanden vil programmet gjøre seg klar til kalibrering ved å kjøre loddet til startposisjonen i topp. I linje 29-30 i kodeutdrag 3.6 settes pådrag (`frequency_ref`) og motororientering (`q2a_up`) slik at motoren heiser opp lasten. Når det merkes et lavere moment (`torque_ref`) enn -400 i linje 33 vil programmet hente en tellerverdi fra frekvensomformeren og gå videre til neste tilstand (`calibration_down`).

Et moment på -400 vil indikere at loddet har nådd topposisjonen. Et høyere moment som virker på motorens momentarm gir mer negativ moment referanse.

Kodeutdrag 3.6: Utdrag fra kode; tilstand 1

```
25 // State 1; get_ready
26    get_ready:
27
28 // Sets speed refererence and motor orientation
29      frequency_ref := 1;
30      q2a_up := TRUE;
31
32 // When torque is high (top position) -> move to next state
33      IF torque_ref <= -400 THEN
34         q2a_up := FALSE;
35 // Get a value form the pulsecounter and move to next case
36         encoder_value_old := encoder_pulses;
37         state := calibration_down;
38      END_IF;
```

Tilstand 2 : I denne tilstanden vil kalibreringen starte. I linje 49 i kodeutdraget 3.7 endres motororienteringen (`q2a_down`) slik at lasten blir senket ned. Videre vil det i linje 52-54 hentes ny tellerverdi fra frekvensomformeren og regnes ut en differanse (`encoder_difference_value`) mellom den nåværende (`encoder_value_now`) og forrige tellerverdi (`encoder_value_old`). Dette blir gjort for å oppdatere rotasjonsendringen til motoren som beskrevet tidligere i delkapittel 3.2.1.

Kodeutdrag 3.7: Utdrag fra kode; tilstand 2

```
45 // State 2; calibration_down
46    calibration_down:
47
48 // Sets motor orientation
49       q2a_down := TRUE;
50
51 // Difference between two pulses from the encoder = rotation change of the motor
52      encoder_value_now := encoder_pulses;
53      encoder_difference_value := encoder_value_now - encoder_value_old;
54      encoder_value_old := encoder_value_now;
```

I linje 56 i kodeutdrag 3.8 telles alle differanseverdiene for å utlede et området for tellerverdiene. Alle disse differansene blir summert i variabelen `calib_difference_counter`. På linje 58 vil pulsene (`calib_difference_counter`) bli gjort om til rotasjoner som er spolt ut fra trommelen (`calib_round_counter`) ved å dele alle de telte differanseverdiene på 8000 (1 rotasjon).

Heltallsvariabelen (`whole_round_counter`) representerer antall hele runder som er spolt ut fra trommelen. På linje 62 spør programmet hvor langt motoren har rotert fra variabelen `calib_difference_counter`. Om motoren har rotert 360 grader vil antall helrunder spolt ut øke med 1. Etter økningen blir det satt et minne på linje 64 slik at dette bare blir gjort en gang mellom intervallene på 8000 pulser.

Videre vil det i linje 67-69 kontinuerlig bergens en totallengde (i mm) ved bruk av likningen 3.7 som beskrevet tidligere i kapittelet. For å få en kontinuerlig utregning av lengden når vinsjsystemet kjører, ganges likningen med et forholdstall som finnes ved å ta vinkelrotasjonen og dele på 8000 pulser (1 omdreining). Forholdstallet gir hvor mye vinkelrotasjonen representerer av en omdreining.

Når det merkes et moment som er høyere enn eller lik 0 i linje 72 vil programmet gå videre til neste tilstand (`calibration_up`). Den siste lengden i millimeter som er målt og siste tellerverdi blir lagret i linje 73-74. Et moment høyere enn 0 vil indikere slakk vaier og nådd bunnposisjon. Et lavere moment som virker på motorens momentarm gir mer positiv momentreferanse.

Kodeutdrag 3.8: Utdrag fra kode; tilstand 2

```
55 // Update total numer of pulses
56      calib_difference_counter := calib_difference_counter + encoder_difference_value;
57 // Update number of rounds
58      calib_round_counter := (calib_difference_counter/8000);
59
60 // Logic for calculating whole rounds
61     FOR count:=1 TO 16 DO
62         IF calib_difference_counter > (count*8000) AND round_memory[count] = FALSE THEN
63          whole_round_counter := whole_round_counter + 1;
64           round_memory[count] := TRUE;
65
66 // Calculation of length [mm] in real time
67      calib_mm_range := calib_mm_range
            +((encoder_difference_value/8000)*LREAL_TO_REAL((2*3.14))
            *(drum_radius+((16-whole_round_counter)*wire_diameter)));
68 // Compensate for number of pulleys in the system
69        calib_mm_range := calib_mm_range/number_of_pulleys;
70
71 // When torque is low (bottom position) -> move to next state
72        IF  torque_ref >= 0 THEN
73            calib_h100_mm := calib_mm_range;
74            calib_h100_difference_counter:= calib_difference_counter;
75            q2a_down := FALSE;
76            state := calibration_up;
77        END_IF;
```

Tilstand 3 : Det vil i denne tilstanden kjøres tilbake til startposisjonen i topp. I linje 84 endres motororienteringen (`q2a_up`) slik at lasten blir heiset opp. Videre vil det i linje 87-89 hentes ut tellerverdi fra frekvensomformeren og regnes ut en differanse mellom den nåværende og forrige tellerverdien. I linje 92 og 94 oppdateres antall telte pulser fra enkoderen og antall runder spolt ut fra trommel.

Når det merkes at de telte differanseverdiene (`calib_difference_counter`) er mindre eller lik 0 på linje 97, vil programmet gå videre til neste tilstand (`idle`). Kalibreringsprosessen er ferdig.

Kodeutdrag 3.9: Utdrag fra kode; tilstand 3

```
80  // State 3; calibration_up
81  calibration_up:
82
83  // Sets motor orientation
84     q2a_up := TRUE;
85
86  // Difference between two pulses from the encoder = rotation change of the motor
87     encoder_value_now := encoder_pulses;
88     encoder_difference_value := encoder_value_now- encoder_value_old;
89     encoder_value_old := encoder_value_now;
90
91  // Update total numer of pulses
92     calib_difference_counter := calib_difference_counter + encoder_difference_value;
93  // Update number of rounds
94     calib_round_counter := (calib_difference_counter/8000);
95
96  //When at countervalue 0 -> Calibration DONE, move back to idle state
97     IF  calib_difference_counter <= 0 THEN
98        HMI_calibration := FALSE;
99        done := TRUE;
100       state := idle;
101    END_IF;
```

3.4 Resultat kalibrering

Det ble utført en test av kalibreringsprosessen. Vi ønsket å få verifisert det kalibrerte området ved å sammenligne høyden fra programmet med en målt høyde fra en tommestokk. Den faktiske høyden ble målt fra bakken til undersiden av loddet. Dette ble gjort fordi det er lettere å måle nøyaktig ved å plassere en tommestokk på bakken. Siden målingen i programmet skjer fra topposisjonen, må den målte høyden trekkes fra området for å finne høyden fra topp.

$$Høyde\ fra\ topp = område - høyde\ fra\ bunn$$

Den faktiske verdien ble målt med en tommestokk og sammenlignet med høyden fra programmet som vist i bildet 3.10.

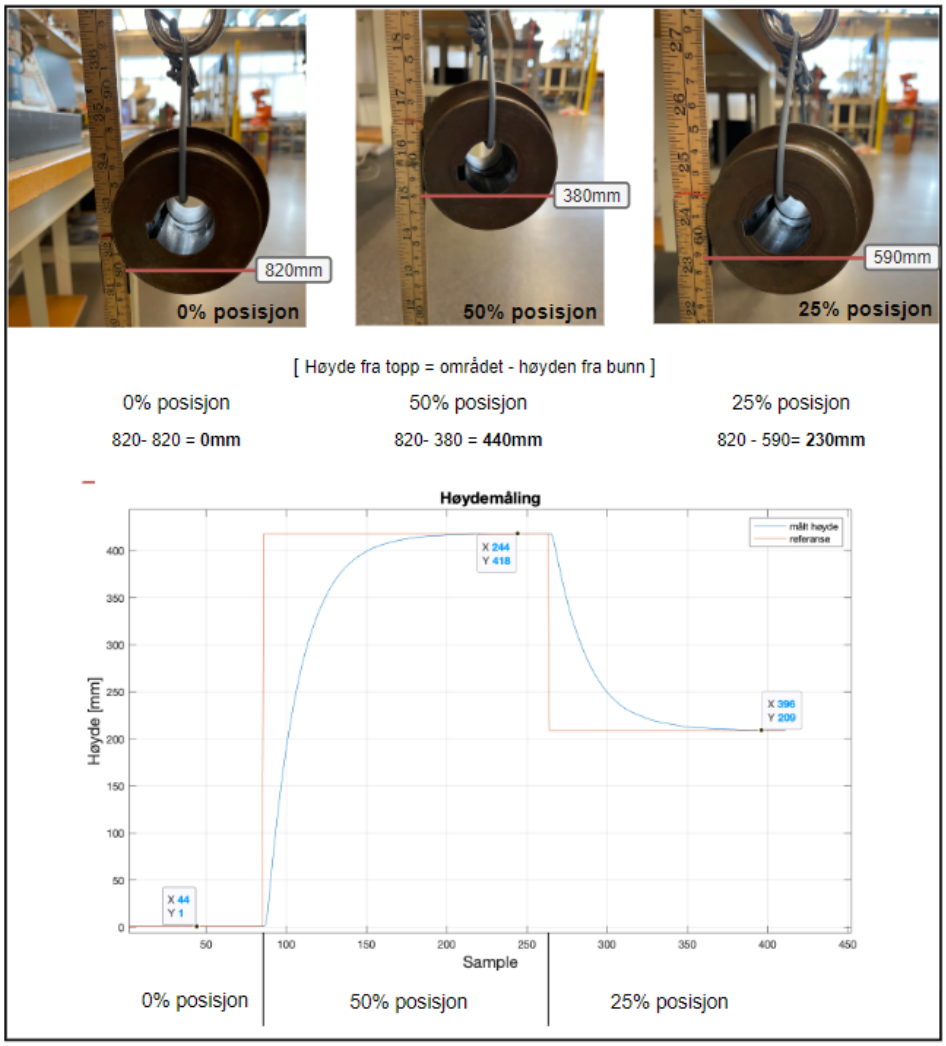


Figure 3.10: Målt verdi fra tommestokk sammenlignet med målt verdi fra programmet.

Lasten ble kjørt opp og ned i intervaller på 25% via posisjonsregulering. Høydereferansen fra programmet ble sammenlignet med den målte høyden og det ble utledet en avvikstabell:

Avvikstabell					
% av området	**0%**	**25%**	**50%**	**75%**	**100%**
Høyde referanse	0mm	209,5mm	418mm	628,5mm	838mm
Målt høyde	0mm	230mm	440mm	648mm	820mm
Avvik	0mm	20,5mm	22mm	19,5mm	18mm
Sample	1	2	3	4	5

Tabell 3.1: Avvik mellom målt høyde og høyde fra program.

De målte høydene i mm ved hvert intervall er de samme i begge kjøreretninger og det blir derfor bare vist en retning i tabellen. Differansen mellom målt høyde og høyden fra prosessen er vist i figur 3.11.

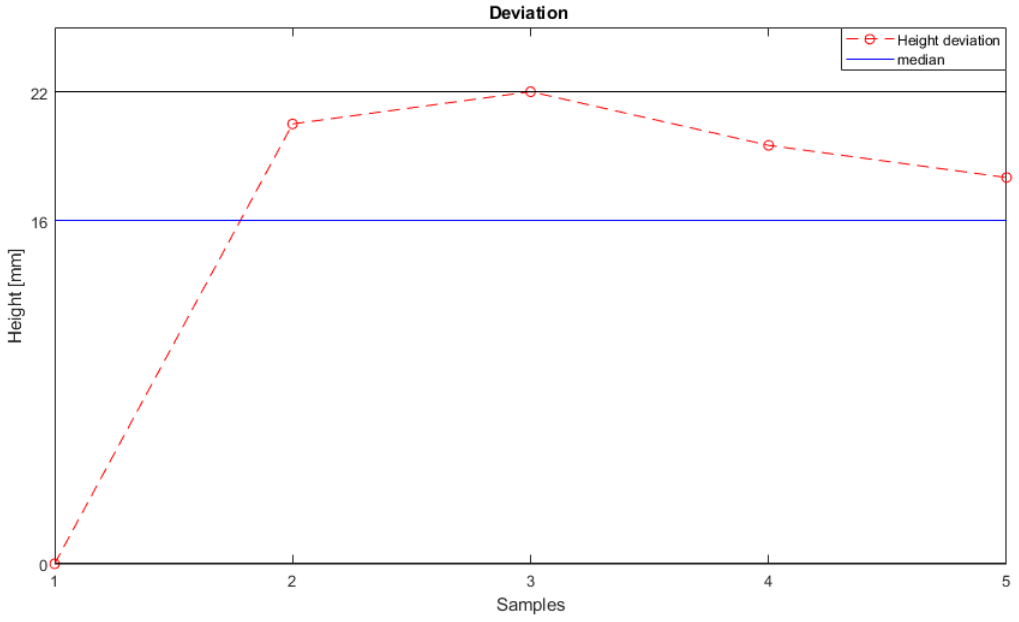


Figure 3.11: Grafen viser differansen mellom målt og utregnet høyde

Noe avvik i høydemålingene er forventet; når området kaliberes blir lasten spolet ned til det er merket lavt moment ved bunnposisjonen. Ut-spolingen stopper ikke momentant, men det spoles ut litt mer vaier ($\approx 1,5cm$, vist i figur 3.12, oransje boks) som ender opp som avvik ved målingen på 100%. Innstillingen for rampetid vi har satt i frekvensomformeren skaper noe avvik også fordi den forskyver alle punktene utenom bunnpunktet på bakken. For å minimere avviket må rampetiden settes til $\approx$ 0s, men dette er ikke ønskelig da man får brå stopp ved

endring i retning eller ved stopp. Dette avviket skapt av rampetiden og den lille utspolingen i bunn ved kalibering ble funnet til å samlet sett utgjøre rundt 18mm. For å få de relle avvikene som oppstår på 25%, 50% og 75% ka man trekker ifra 18mm på de målte verdiene der for å kompensere for dette, men vi valgte å la det stå slik som det er. Situasjonen er illustrert i figuren under 3.12:

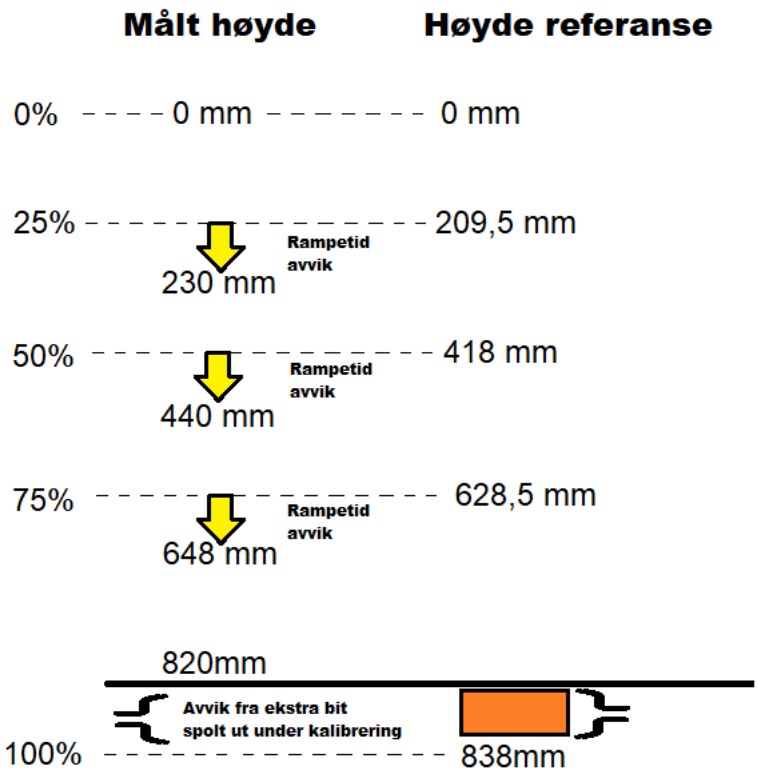


Figure 3.12: De gule pilene viser hvordan rampetiden offsetter lasten fra referansen slik at avviket dannes. Rampetide vil naturligvis ikke klare å offsettet lasten igjennom bakken ettersom det er fysisk barriere. Den orange boksen i bunn viser situasjonen fra kalibrering som nevnt med at det spoles ut en distanse ettersom det ikke er momentan stopp på utspoling når det merkes lavt moment ved sjøen.

Vi kan konkludere med at kalibreringsprosessen som er utviklet fungerer som forventet. Det er lite avvik mellom den målte verdien og høydeverdien fra prosessen. Medianen til avviket ligger rund 19-20mm.

Det ble utført flere forsøk, og avvikstabellen er reprensentativ for alle kalibreringer.

I figur 3.13 vises data som blir gjort tilgjengelig for brukeren på HMI skjermen under kalibrering.

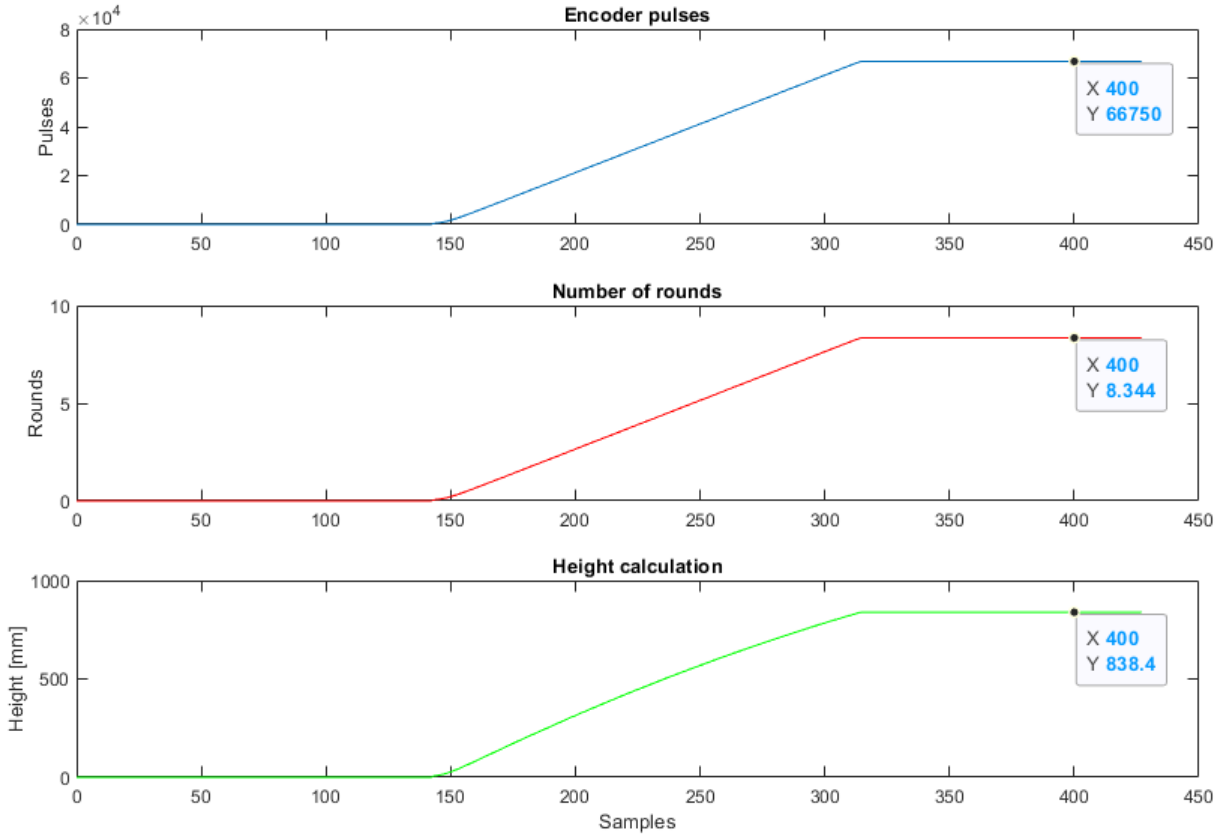


Figure 3.13: Graf 1: Viser antall pulser avgitt fra enkoderen som representerer den totale vinkelforskyvningen til motoren helt ned til bunnposisjonen. Graf 2: Viser antall runder regnet ut fra den totale vinkelforskyvningen for å senke lasten ned til bunn. Graf 3: Viser kalibrert område i mm fra topposisjon og ned til bunnposisjonen.

Kapittel 4

Posisjonsregulering med PID-regulator

I dette kapittelet presenteres teori, løsning og resultater for posisjonsregulering av vinsjsystemet.

Det ble implementert posisjonsregulering av vinsjsystemet. Posisjonsregulatoren vil regulere pådraget til motoren for å oppnå ønsket høyde innenfor det kalibrerte området. Ved lastendring vil regulatoren øke pådraget for å vedlikeholde denne posisjonen.

Pseudokoden under viser hvordan posisjonsregulatoren er bygget opp.

Kodeutdrag 4.1: Pseudokode posisjsonsregulering

```
// 1. Brukeren taster inn ønsket posisjon i millimeter på HMI skjermen.
// 2. Det genereres et avvik mellom den målte posisjonen til lasten og den ønskede
    referanseverdien for posisjonen.
// 3. Avviket reguleres av en PID-regulator som beregner pådrag til
    frekvensomformeren.
// 4. Frekvensomformeren, via sin interne regulator beregner pådrag ut på motoren for
    åjustere posisjonen.
// 5. Lastens endrede posisjon regnes ut ved differansen mellom nåværende og forrige
    tellerverdi.
```

Figur 4.1 viser blokkskjema av prosessen ved posisjonsregulering.

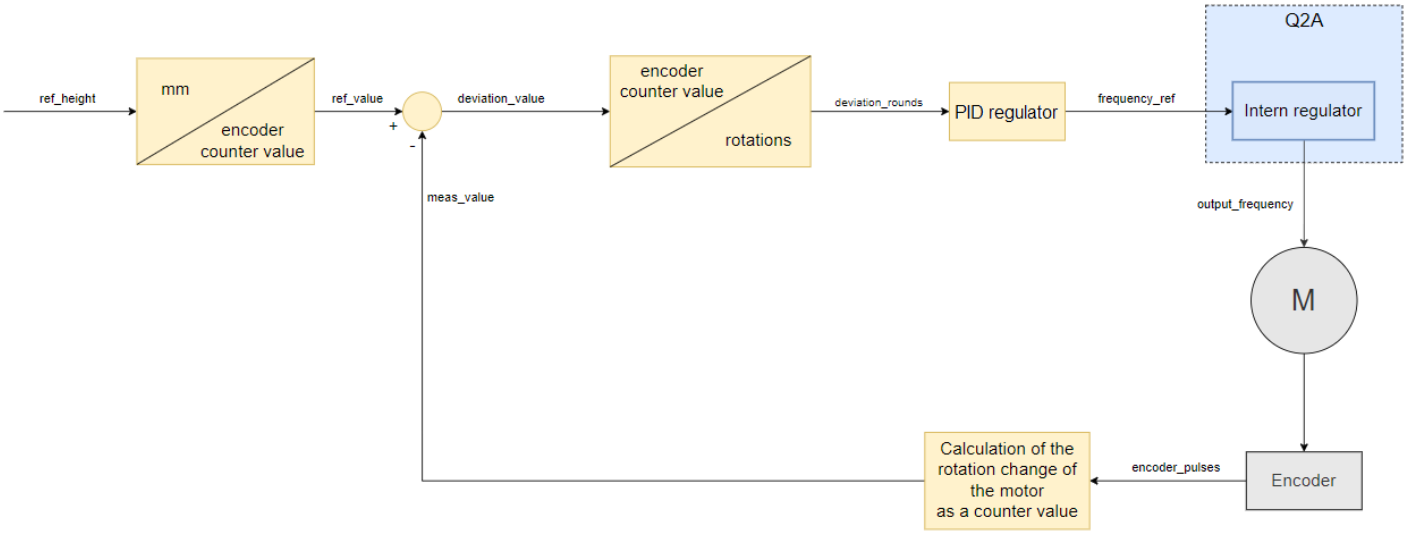


Figure 4.1: Blokkskjema av prosessen ved posisjonregulering. Gul del er i PLS program, blå del er internt i frekvensomformeren (vist tidligere i fig. 2.6) og grå er fysiske komponenter.

Pulsene fra enkoderen som er telt av variabelen (`encoder_pulses`) i frekvensomformeren blir anvendt i en matematisk utregning for å beregne den målte høyden (`meas_value`) i form av en tellerverdi i det kalibrerte området. Referanseposisjonen (`ref_height`) er den posisjonen brukeren har tastet inn på HMI skjemen som ønsket høyde [mm]. Referanseposisjonen konverteres til en tellerverdi (`ref_value`) i det kalibrerte området. Avviket (`deviation_value`) mellom referanse og målt høyde vil konverteres til et avvik representert som runder (`deviation_rounds`). Basert på denne verdien vil en PID-regulator beregne en frekvensreferanse (`frequency_ref`) som sendes til den interne reguleringsløyfen i frekvensomformeren.

Den interne sløyfen er vist tidligere i figur 2.6, hvor det beregnes et pådrag (`output_frequency`) ut til motoren for å justere posisjonsavviket.

Fra kapittel 4.4.5 vet vi at systememt er kalibrert for å lage en distanse fra høyden h(0) i topp ned til h(100) ved sjøen. Dette medfører at en possisjonsreferanse (`ref_value`) ved 838mm vil være nede ved sjøen og en referanse ved 0mm vil være oppe i toppen nær trinsene.

Figur 4.2 illustrer posisjonen til lasten i virkeligheten og den målte høyden (`meas_value`) for posisisjon ifht referansen (`ref_value`). Dette er basert på avviket (`deviation_value`):

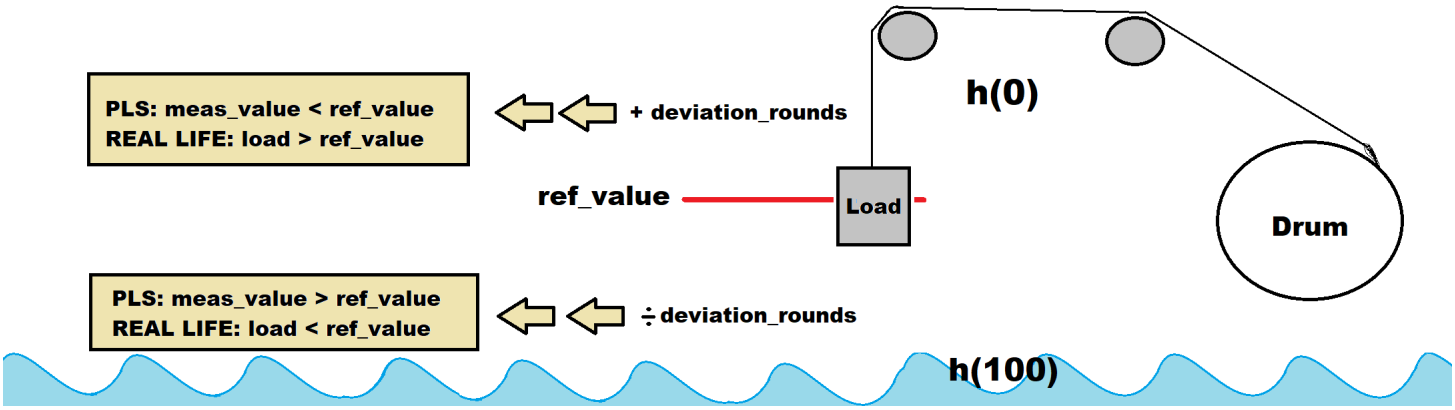


Figure 4.2:
Ved positivt (+) avvik (`deviation_value`):
lasten er over referansehøyden (`ref_value`) i virkeligheten, men den målte distansen (`meas_value`) i PLSen er lavere enn referansen(`ref_value`) ettersom systemet jobber med 0-punktet sitt i toppen h(0).
Ved negativt(-) avvik (`deviation_value`):
lasten er under referanse høyde (`ref_value`) i virkeligheten, men den målte høyden (`meas_value`) i PLSen er høyere enn referansen (`ref_value` ettersom systemet jobber med 100-punktet sitt i toppen h(100).).

Fra figur 4.2 vises sammenhengen mellom hvor lasten befinner seg i det virkeligheten ifht. hvilken målt verdi man leser av i PLS programmet.

4.1 PID-regulator

Avviket i posisjon mellom referanseverdien og den faktiske posisjonen til lasten blir regulert av en PID-regulator. Regulatoren beregner et pådrag fra avviket som settes ut til motoren. Motoren vil kjøre og avviket justeres.

Pådraggssignalet som beregnes av PID-regulatoren vises i likning 4.1.

$$u(t) = Kp \cdot e(t) + Ki \cdot \int_0^t e(t)\,dt \;+\; Kd \cdot \frac{d\,e(t)}{dt} \tag{4.1}$$

P-leddet til regulatoren vil påføre et bidrag som er proporsjonalt med avviket, mens I-leddet vil fungerer som et minne av alle de målte avvikene som integreres opp over tid. D-leddet gir bidrag basert på raske endringer i avviket.

I likningen 4.1 brukes det en justeringsfaktor K_p for P-leddet, justeringsfaktor K_i for I-leddet og justeringsfaktor K_d for D-leddet. Disse faktorene ganges inn i bidragene. Størrelsen på disse faktorene bestemmes senere via 'prøv og feil'-tuning metoden og 'Skogestads'-tuning metode.

Den numerisk integrasjonen basert på 'Eulers forovermetode' som ble brukt for å lage I-leddet er vist i likning 4.3:

$$u\_I(t) = u\_I(t-1) + e(t) \cdot Ts \tag{4.2}$$

Hvor variablene i liking 4.3 er:

- u_I(t) - Det nye I-leddet
- u_I(t-1) - Det gamle I-leddet
- e(t) - Avviket i posisjon
- Ts - Tidssteget i sekund

Den numerisk derivasjonen som ble brukt for å lage D-leddet er vist i likning 4.3:

$$u\_D(t) = \frac{e(t)}{Ts} \tag{4.3}$$

Hvor variablene i liking 4.3 er:

- u_D(t) - D-leddet
- e(t) - Avviket i posissjon
- Ts - Tidssteget i sekund

4.2 Kode for posisjonsregulator

I dette delkapittelet vil koden for posisjonsregulatoren presenteres. For å kunne posisjonsregulere ble det utviklet en HMI skjerm som vist i figur 4.3. På denne skjermen kan brukeren skrive inn ønsket høyde (referanseverdi) og verdier for K_p, K_i og K_d. Posisjonsjusteringen starter når brukeren trykker inn knappen `Position adjustment`.

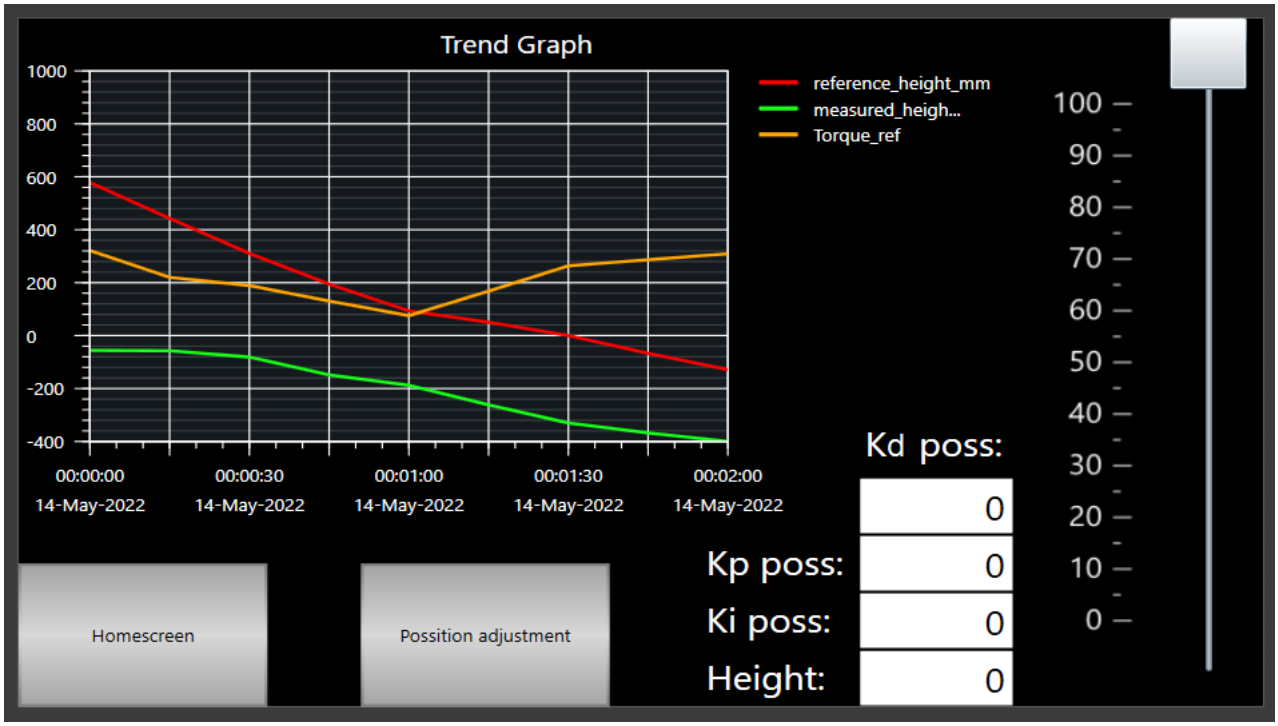


Figure 4.3: I figuren vises HMI skjerm for posisjonsregulering. Brukeren kan på denne skjermen posisjonsjustere gjennom knappen `Possition adjustment`. `Homescreen` knappen vil føre brukeren tilbake til hjemskjermen. Ønsket posisjonen settes gjennom scrollbaren eller numerisk (via `Height`). Det kan også settes verdier for K_p, K_i og K_d . I grafen vises den målte høyden i mm sammen med referanseverdien, i tillegg vises momentreferansen. Verdiene i grafen er ikke reelle; bildet er tatt ved simulering i Sysmac studio.

Det første programmet gjør er å sjekke programminnene for å vite hvilket program som sist ble kjørt. Dette gjøres for å overføre høydemåling fra andre programmer som vist i kodeutdrag 4.2. I linje 12 og 16 spør programmet om det forrige program som ble kjørt var hastighetskontroll (`HMI_page_speedCTL_memory`) eller manuell kjøring (`HMI_page_manual_memory`). Deretter blir det i linje 13-14 eller 17-18 hentet verdier for antall pulser avgitt fra enkoderen og runder spolt ut fra trommel.

Det forrige programminne settes til `FALSE` og programminnet for posisjonsjustering til `TRUE` i linjene 21-23.

Kodeutdrag 4.2: Utdrag fra kode; Position adjustment

```
10 IF HMI_page_position_memory = FALSE THEN
11 // Gets current position and rounds from manual driving
12    IF HMI_page_speedCTL_memory = TRUE THEN
13       position_difference_counter := speedCTL_difference_counter;
14       position_round_counter := speedCTL_round_counter;
15 // Gets current position and rounds from speed control
16    ELSIF HMI_page_manual_memory = TRUE THEN
17       position_difference_counter := manual_difference_counter;
18       position_round_counter := manual_round_counter;
19    END_IF;
20 // Sets position page as last page
21    HMI_page_speedCTL_memory := FALSE;
22    HMI_page_manual_memory := FALSE;
23    HMI_page_position_memory := TRUE;
24 END_IF;
```

For å oppdatere rotasjonsendringen til motoren brukes differansen mellom to pulser avgitt fra enkoderen som vist i kodeutdrag 4.3. Logikken ble forklart tidligere i delkapittel 3.2.1.

Kodeutdrag 4.3: Utdrag fra kode; Position adjustment

```
26 // Difference between two pulses from the encoder = rotation change of the motor
27 encoder_value_now := encoder_pulses;
28 encoder_difference_value := encoder_value_now - encoder_value_old;
29 encoder_value_old := encoder_value_now;
```

Den målte verdien for høyde (`meas_value`) oppdateres kontinuerlig ved bruk av rotasjonsendringen til motoren som vist i kodeutdrag 4.4. Differansen (`encoder_difference_value`) mellom nåværende og forrige tellerverdi gir rotasjonsendringen. Ved å telle alle differanseverdiene har man endringen i posisjon (`position_difference_counter`). Antall runder spolt ut fra trommelen (`position_round_counter`) kan finnes ved å dele alle de telte differanseverdiene på 8000. I delkapittel 3.1 ble det beskrevet at 8000 pulser avgitt fra enkoderen tilsvarer 1 omdreining på motoren.

Kodeutdrag 4.4: Utdrag fra kode; Position adjustment

```
31 // Update total number of pulses
32 position_difference_counter := position_difference_counter + encoder_difference_value;
33 // Update number of rounds
34 position_round_counter := position_difference_counter/8000;
35 // Sets the measured height value
36 meas_value := position_difference_counter;
```

Ønsket høyde er definert av variabelen `ref_display` gjennom det numeriske displayet, og `ref_scrollbar` gjennom scrollbaren til HMI skjermen. I kodeutdraget 4.5 sjekkes det i linje 45 og 47 hvilken av metodene brukeren har valgt å bruke. Referanseverdien blir skrevet videre til variabelen `ref_height`.

Referanseverdien til høydeposisjonen settes i millimeter, men blir konvertert til en tellerverdi som representerer denne høyden (`ref_value`) i linje 52. Tellerverdien beregnes ved å dele den ønskede høyden på hele høydeområdet i millimeter fra kalibreringen (`calib_h100_mm`) og gange med hele tellerverdiområdet fra kalibreringen (`calib_h100_difference_counter`).

Kodeutdrag 4.5: Utdrag fra kode; Position adjustment

```
40 // Referance length in mm/sec from either numeric display or scrollbar
41 ref_display := HMI_userinput_ref_height_display;
42 ref_scrollbar := HMI_userinput_ref_height_scrollbar;
43
44 //  Logic for input method: numeric display or scrollbar
45 IF ref_display > 0 and ref_scrollbar = 0 THEN
46     ref_height = ref_display
47 ELSIF ref_scrollbar > 0 AND ref_display = 0 THEN
48     ref_height = ref_scrollbar
49 END_IF
50
51 // Convert to number of pulses
52 ref_value = (ref_height/calib_h100_mm) * calib_h100_difference_counter;
```

Ved å sammenligne referansen med den målte høydeverdien oppstår et avvik (`deviation_value`) mellom disse som vist i linje 164 i kodeutdraget 4.6. Avviket konverteres videre fra en tellerverdi til runder (`deviation_rounds`) ved å dele på 8000 (1 omdreining).

Kodeutdrag 4.6: Utdrag fra kode; Position adjustment

```
163 // Deviation
164  deviation_value = ref_value - meas_value;
165  deviation_rounds = deivation_height/8000
```

Kodeutdrag 4.7 viser beregningen av integralfunksjonen til I-leddet til regulatoren. I linje 168-172 defineres tidsskrittet og i linje 175-176 beregnes integralfunksjonen. På linje 179 regnes derivatfunksjonen som bare ble brukt for 'Skogestad'-metoden. Ved linje 182 filtreres D-leddet.

Kodeutdrag 4.7: Utdrag fra kode; Position adjustment

```
167 // Timestep caluclation
168 time_new := GetTime();
169 time_step:= SUB_DT_DT(time_new, time_old);
170 time_old := time_new;
171 time_step_nano := TimeToNanoSec(time_step);
172 time_step_nano_real := LINT_TO_REAL(time_step_nano) / 1000000000;
173
174 // Integral function of the deviation
175 I_value_new := I_value_old + (deviation_rounds * time_step_nano_real);
176 I_value_old := I_value_new;
177
178 // Derivation function of the deviation (ONLY IN USE FOR SKOGESTAD)
179 D_value_new := deviation_rounds / time_step_nano_real;
180
181 // Filerede derivation function (ONLY IN USE FOR SKOGESTAD)
182 filtered_D_value := (filter_factor * D_value_new) + (1-filter_factor * D_value_old);
183 D_value_old := filtered_D_value;
```

I kodeutdraget 4.8 vises PID-regulatoren ved posisjonsjustering. Verdier for K_p, K_i og K_d kan settes av brukeren gjennom HMI skjermen (fig. 4.3). Justeringsparameterne brukes sammen med avviket (`deviation_rounds`), I-verdien (`I_value_new`) og D-verdien (`filtered_D_value`) til å beregne et reguleringspådrag (`regulator`).

Kodeutdrag 4.8: Utdrag fra kode; Position adjustment

```
182 regulator = HMI_userinput_kp * deviation_rounds + HMI_userinput_ki * I_value_new +
        HMI_userinput_kid * filtered_D_value;
```

Det sjekkes om regulatorsignalet er positivt eller negativt i linje 191 og 195 i kodeutdrag 4.9. Pådragsorientering er styrt utifra regulatorsignalets fortegn.

Frekvensomformeren kan ikke ta imot negativ frekvensreferanse, det er nødvendig å bruke absoluttverdien av regulatorsignalet som vist i linje 200.

Kodeutdrag 4.9: Utdrag fra kode; Position adjustment

```
190 // Motor orientation and frequency reference adjustment based on the controller signal
191 IF regulator > 0 THEN
192    q2a_up := FALSE;
193    q2a_down := TRUE;
194 
195 ELSIF regulator < 0 THEN
196    q2a_up := TRUE;
197    q2a_down := FALSE;
198 END_IF;
199 
200 frequency_ref := ABS(regulator);
```

4.3 Skogestad tuning metode for prosesser uten tidsforsinkelse

I dette delkapittelet vil vi analysere prosessene vi regulerer og knytte en sammenheng mellom dem. Vi finner transferfunksjonene deres og knytter denne opp til en tabell fra 'Skogestads'-metoden som gir oss retningslinjer for å valg av reguleringsparametrene. Vi finner også τ og prosessens 'gain' (K).

Posisjon er intergrert av vinkelhastighet. Det ser vi fra likningen 4.4:

$$Posisjon(t) = \theta(t) = \int_0^t \omega(t)\, dt \tag{4.4}$$

Som i Laplace domenet har en 2. orden transferfunksjon som vist i likning 4.5:

$$H_{poss}(s) = \frac{K}{s(\tau \cdot s + 1)} \tag{4.5}$$

Vinkelhastighet er derivert av posisjon. Dette er vist i likning 4.6:

$$Vinkelhastighet(t) = \omega(t) = \frac{d}{dt}\theta(t) \tag{4.6}$$

Som i Laplace domenet har en 1. orden transferfunksjon som vist i likning 4.7:

$$H_{vinkelhast}(s) = \frac{K}{\tau \cdot s + 1} \tag{4.7}$$

Sammenhengen mellom vinkelhastighet og posisjon er vist i figur 4.4.

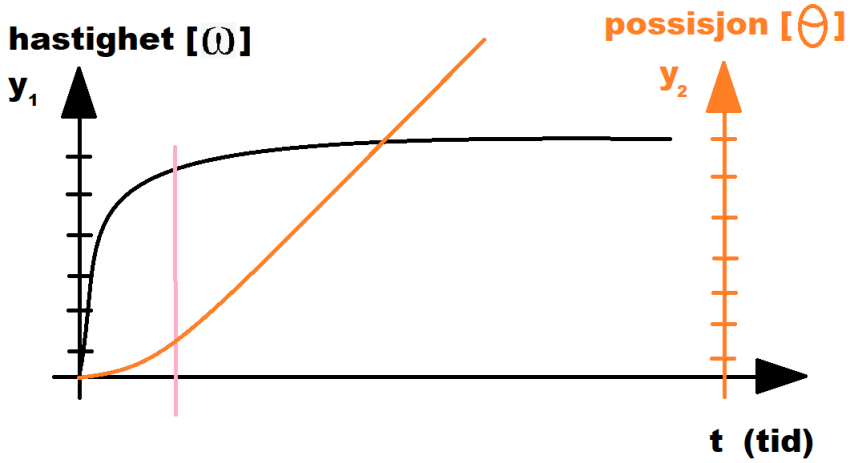


Figure 4.4: Sammenhengen mellom vinkelhastighet(t) i oransje og posisjon(t) svart. Den rosa streken markerer når posisjonen går over til å være en ren intregrerende funksjon.

Tabellen 4.1 viser hvordan reguleringsparametrene skal settes ifra prosessens transferfunksjon:

Skogestads metode for prosesser uten tidsforsinkelse				
Index	**Hp(s) Prosess**	**Kp**	**Ti**	**Td**
1	$\frac{K}{s}$	$\frac{1}{KTc}$	k1Tc	0
2	$\frac{K}{\tau s+1}$	$\frac{\tau}{KTc}$	$\min[\tau, k1Tc]$	0
3	$\frac{K}{(\tau s+1)s}$	$\frac{1}{KTc}$	k1Tc	T
4	$\frac{K}{(\tau 1 s+1)(\tau 2 s+1)}$	$\frac{\tau 1}{KTc}$	$\min[\tau 1, k1Tc]$	T2

Tabell 4.1: Prosessene som reguleres er vist i Index3(posisjon) og Index 2(fart).

Hvor konstantene i tabellen over er som følger:

- K - "gain" $= \frac{y(2)-y(1)}{u(2)-u(1)} = \frac{\delta målt}{\delta referanse}$
- τ - er tiden det tar før den målte verdien er nådd 63,2% av sprangets magnitude.
- K1 - Kompenseringsfaktor (Standardvalg for k1 = 4 eller k1 = 1,44 for hurtigere forstyrrelsekompensering)
- Tc - For prosesser uten tidsforsinkelse velges $Tc = \tau$, men man kan også sette Tc mellom 0.2-0.5.

Fra likningen 4.5 knyttes posisjon opp til "Index 3" i tabell 4.1. Vinkelhastighet fra liking 4.7 knyttes til 'Index 2' i tabell 4.1.

4.3.1 Åpen krets respons:

For å finne de nødvendige verdiene fra Skogestads metode ble prosessen satt i et arbeidspunkt i åpen krets, for å deretter sette et sprang i hastighetsreferansen. Fra denne responsen finner vi τ og "gainen" K til prosessen. En oversikt over responsen er vist i figur 4.5.

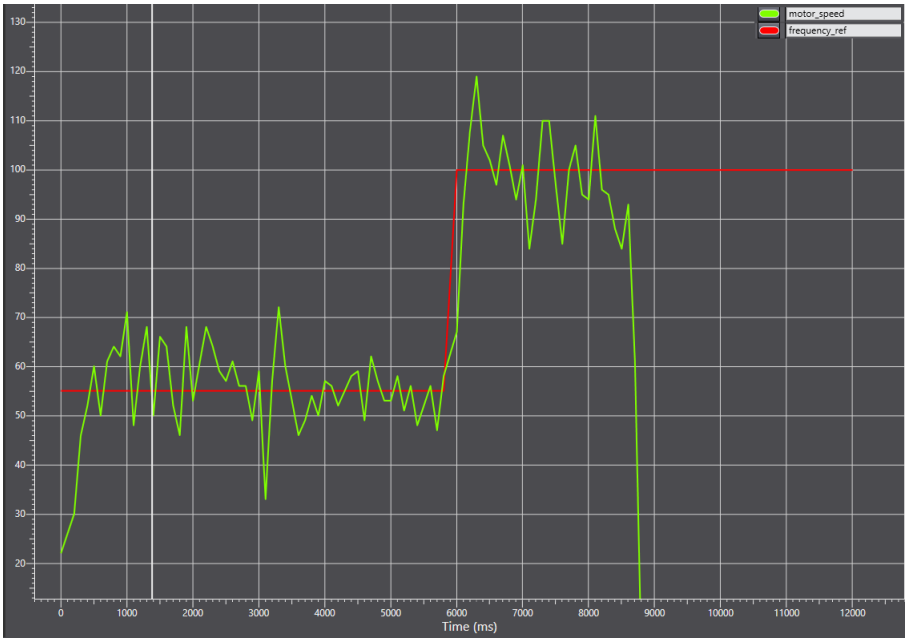


Figure 4.5: Oversikt over responsen i hastighet ved endring av frekvensreferansen.

Den stiplete oransje streken i figur 4.6 viser når den målte farten er ved τ verdien til spranget.

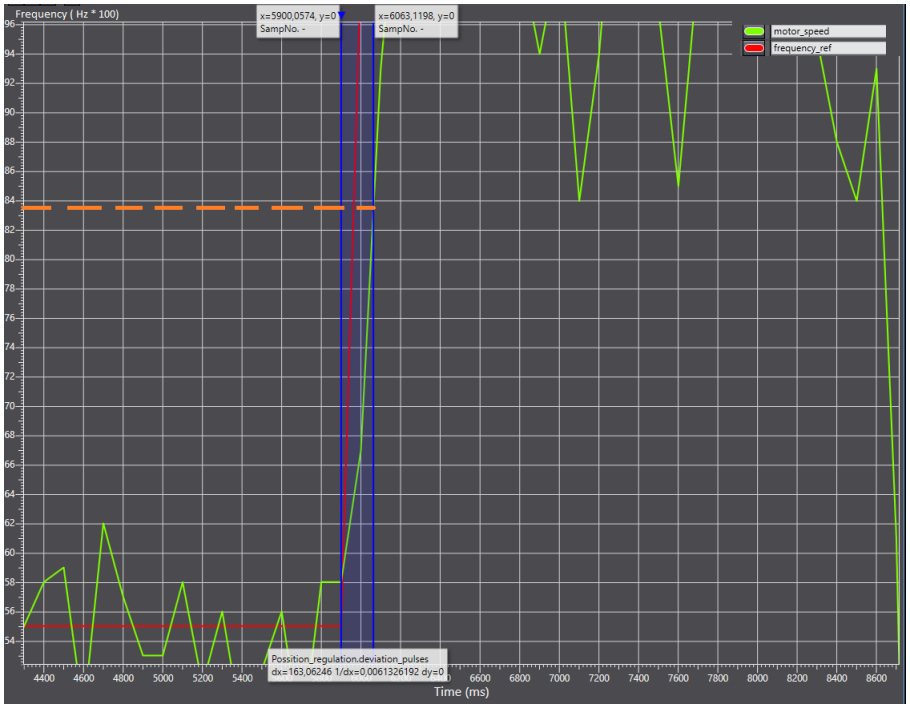


Figure 4.6: Endring av frekvensreferanse med frekvensomformeren satt i 'åpen krets'. Rød strek representerer frekvens referansen. Grønn strek; målt fart. Det blå tidsintervallet markerer tiden det tar å oppnå τ. Oransje stiplet strek markerer den målte hastigheten ved τ. X-aksen er i millisekund og Y-aksen i skalert verdi for Hz (100 = 1Hz).

Alle verdiene for fart i plottet er skalert med 100 ganger den reelle farten.

Den første rekvensreferanse i figur 4.6 er 55. Etter spranget er frekvensreferansen 100. Størrelsen på spranget er 45. Farten har en del svinginger over og under referanseverdien, gjennomsnittet av målingene ved den første frekvensreferansen er lik 54.27. Farten etter spranget er 99.35.

63, 2 % av magnituden til sprangets verdi er 28.44. Den målte farten ved τ er 83,44 som vist av den oransje stiplede linjen i figur 4.6. Tiden når hastigheten er målt er 6063ms og tiden når spranget ble introdusert er 5900ms. Dette tidsintervallet er markert i blått i figur 4.6. Det gir en verdi for τ som er 163ms.

Regner ut "gain" til prosessen:

$$K = \frac{\delta y(t)}{\delta u(t)} = \frac{99,35 - 54,27}{100 - 55} = 1,0018 \approx 1 \tag{4.8}$$

4.4 Resultater av posisjonsregulering

I dette delkapittelet vil resultatene fra ulike tester ved posisjonsregulering av systemet beskrives. Vi finner reguleringsparameterene til regulatoren gjennom 'prøv og feil'- og 'skogestads'-metode. Regulatoren blir testet i ulike tilfeller med disse parametrene. Regulatoren som presterte best ifht. til kravene (vist under) har flere tester som er vist mer i detalj.

Vi satt følgende krav for prosessen:

- Ingen oversving
- Rask respons
- Håndterer forstyrrelser

Ved posisjonsjustering av en last er det ikke ønskelig med for mye oversving, men samtidig er det ønsket en rask respons. Ved lastendring under posisjonsjustering er det ønsket at regulatoren kompenserer for den ekstra lasten.

4.4.1 'Prøv og feil'-tuning metode for PI-regulator

Det ble utført 'prøv og feil'-metoden som går ut på å teste flere ulike parameterverdier for K_p og K_i. Ved å observere responsen til den målte posisjonen opp mot referansen finner vi de parameterverdiene som tilfredsstiller kravene vi har satt for prosessen best.

Valg av K_p:

For å finne den beste K_p verdien til posisjonsregulatoren ble verdiene i tabell 4.2 testet. I dette forsøket er I-leddet ikke i bruk ($K_i = 0$).

Reguleringsparametrene Kp test					
Kp	1	2,5	5	7,5	10
Ki	0	0	0	0	0

Tabell 4.2: Verdier for K_p som ble testet i forsøket.

Testen startet i en posisjon på 400mm fra startposisjonen i topp. Lasten ble posisjonsjustert med et sprang på 10%, som vist i figur 4.7.

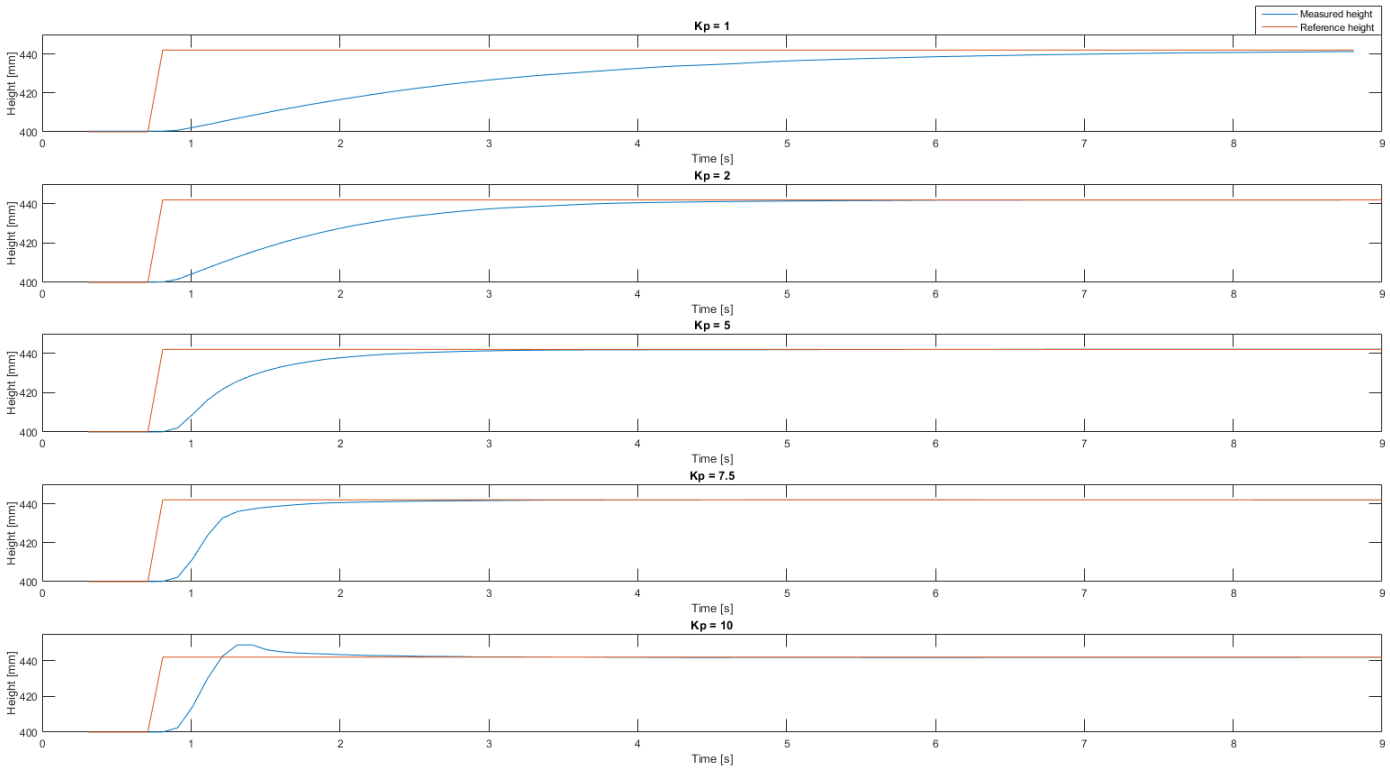


Figure 4.7: Responsen til prosessen ved ulike K_p verdier. Rød linje er høyde referansen og blå er den målte høyden.

Fra figur 4.7 ser vi at regulering med $K_p = 1$, har en treg respons som bruker omtrent 8.5 sekund på å justere posisjonen. Regulering med $K_p = 10$ bruker 1.2 sekund på å nå den nye poisjonsreferansen, men det skapes oversving.

Fra dette plottet ser $K_p = 5$ ut som den beste parameteren, men det vil bli oversving hvis vi introduserer større sprang. Vi ender dermed opp med $K_p = 2$ som den beste reguleringsparameteren.

Valg av K_i:

For å finne den beste verdien for K_i ble ulike verdier testet sammen med $K_p = 2$. Verdiene som ble testet er vist i tabell 4.3.

Reguleringsparametre Ki test				
Kp	2	2	2	2
Ki	1	0,5	0,1	0,01

Tabell 4.3: Verdier som ble testet i forsøket.

Sprangtesten er lik som den vi brukte for å finne Kp. Resultatet for forsøket vises i figur 4.8.

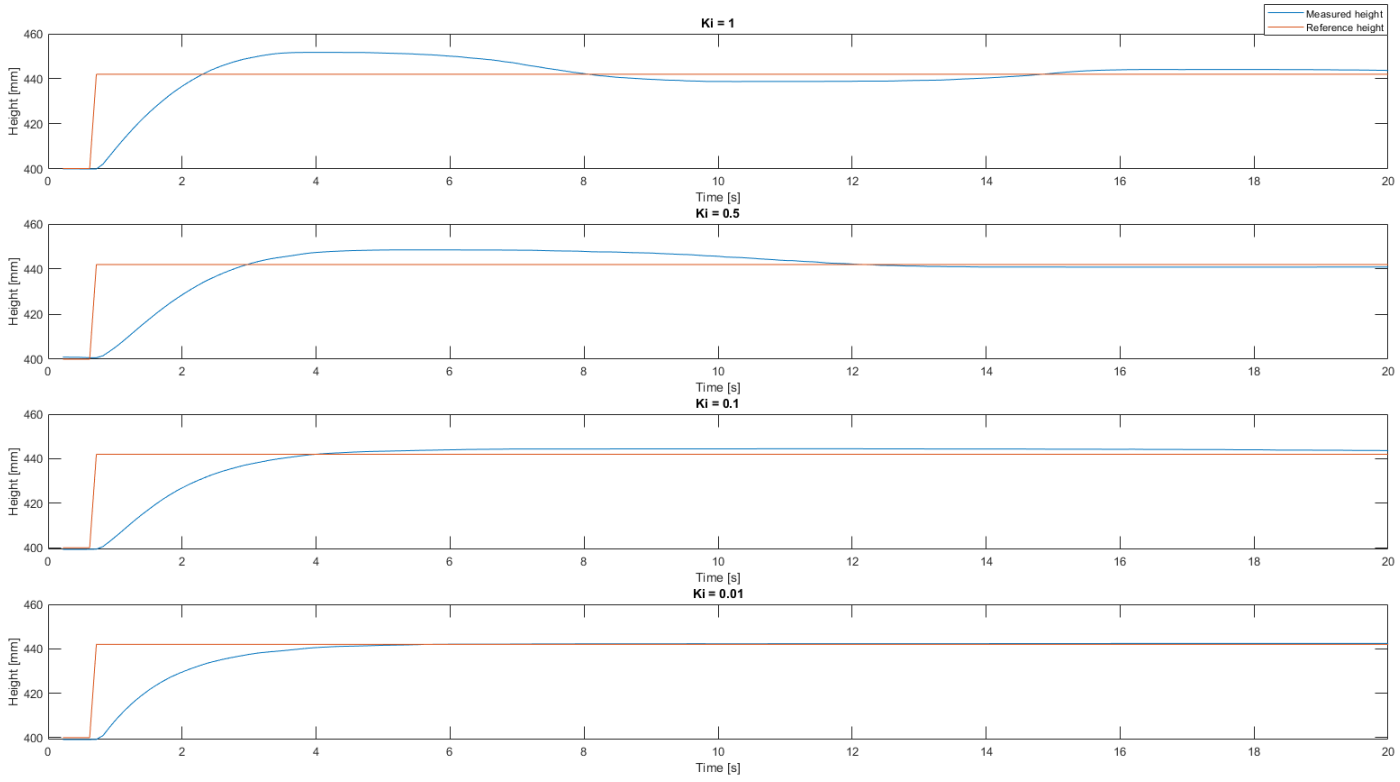


Figure 4.8: Responsen til prosessen med $K_p = 2$ og de ulike K_i verdier. Rød linje er høyde referansen og blå er den målte høyden.

Fra figuren ser man at høyere verdi for K_i gir betydelig oversving. Vi kan dermed konkludere med at $K_i = 0.01$ er den beste for regulatoren, ettersom den ikke gir oversving.

4.4.2 Tester 'prøv og feil'-metode for PI-regulator (detaljert)

Alle testene under blir utført med verdiene $K_p = 2$ og $K_i = 0.01$ fra 'prøv og feil'-metoden.

Store referanse sprang:

Det ble introdusert større sprang (25%, 50%, 75%, 100%). Dette ble gjort for å se at responsen på store sprang er like tilfredsstilende som ved små sprang. Resultatet vises i figur 4.9.

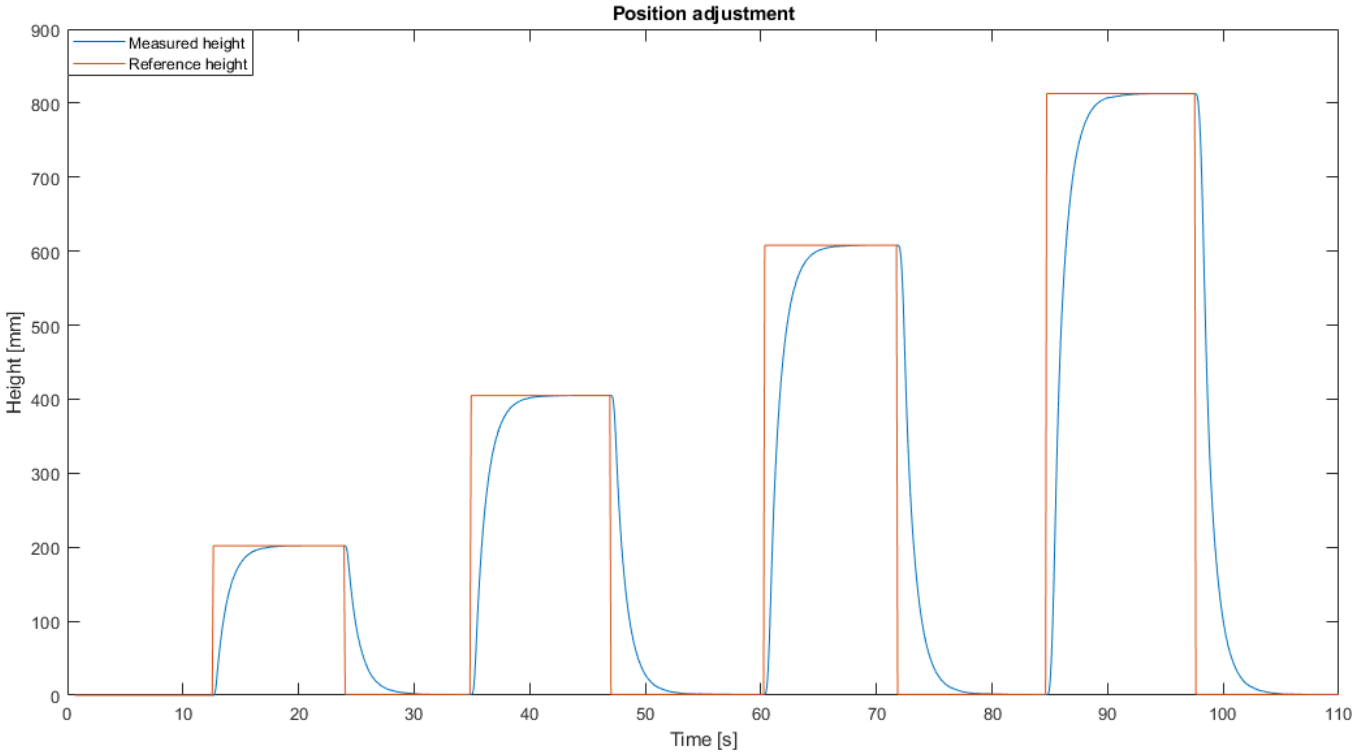


Figure 4.9: Ulike sprang i høydereferansen med $K_p = 2$ og $K_i = 0.01$. Rød linje er høyde referansen og blå er den målte høyden.

Fra figur 4.9 ser vi at ved større sprang er responsen fortsatt rask, i tillegg til at det ikke skapes oversving.

Kaotiske referanse sprang:

For å se hvordan systemet taklet veldig kaotiske endringer i referansen ble det forsøkt å sette mange tilfeldige sprang før systemet hadde posisjonsjustert ferdig inn til denne høyden. Det ble gjort for å prøve å sette prosessen i ustabil selvsving. Oppførselen til systemet er vist i figur 4.10.

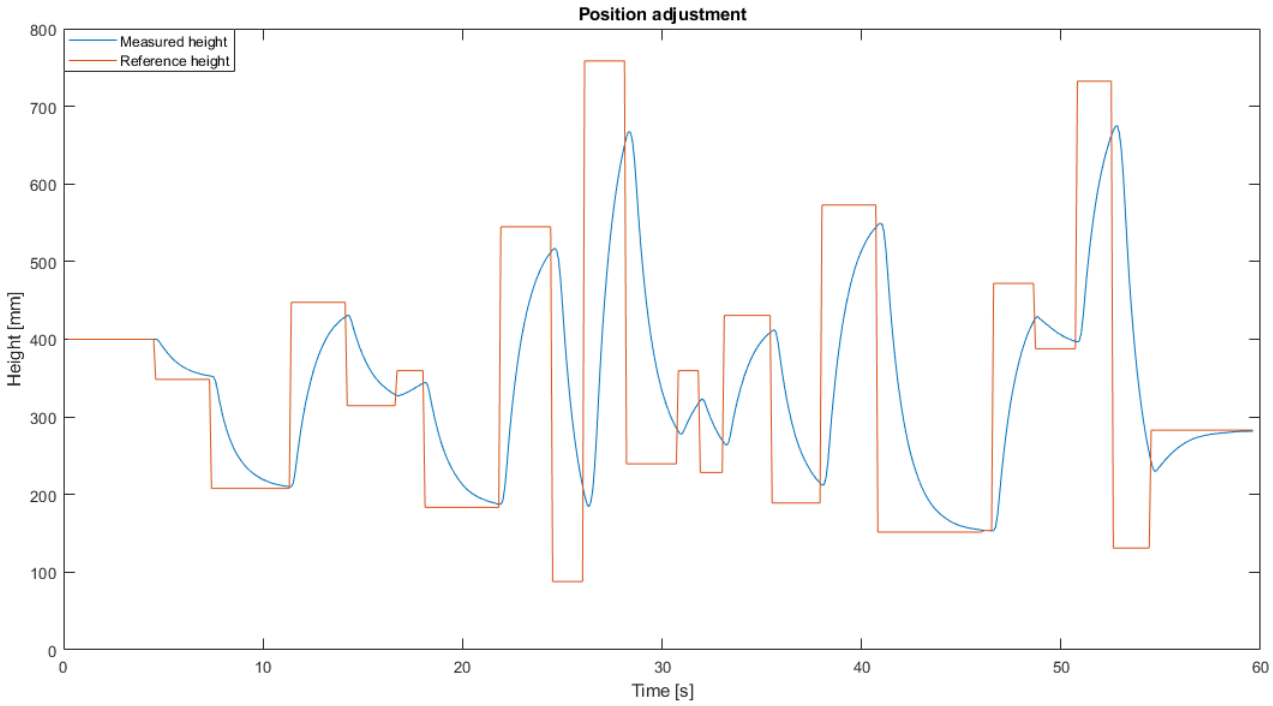


Figure 4.10: Kaotisk sprang respons. Rød linje er høyde referansen og blå er den målte høyden.

Fra grafen ser vi at veldig raske endringer i referansen ikke fører til ustabilt pådrag til motoren. Den takler fint å håndtere slike kaotiske tilfeller.

Lastkompensasjon (detaljert):

Det ble utført en test av kompenseringsegenskapene. Testen gikk ut på å la posisjonen justeres inn til høydereferansen, for å deretter legge til 7,5 kg ekstra vekt. Testen er vist i figur 4.11.

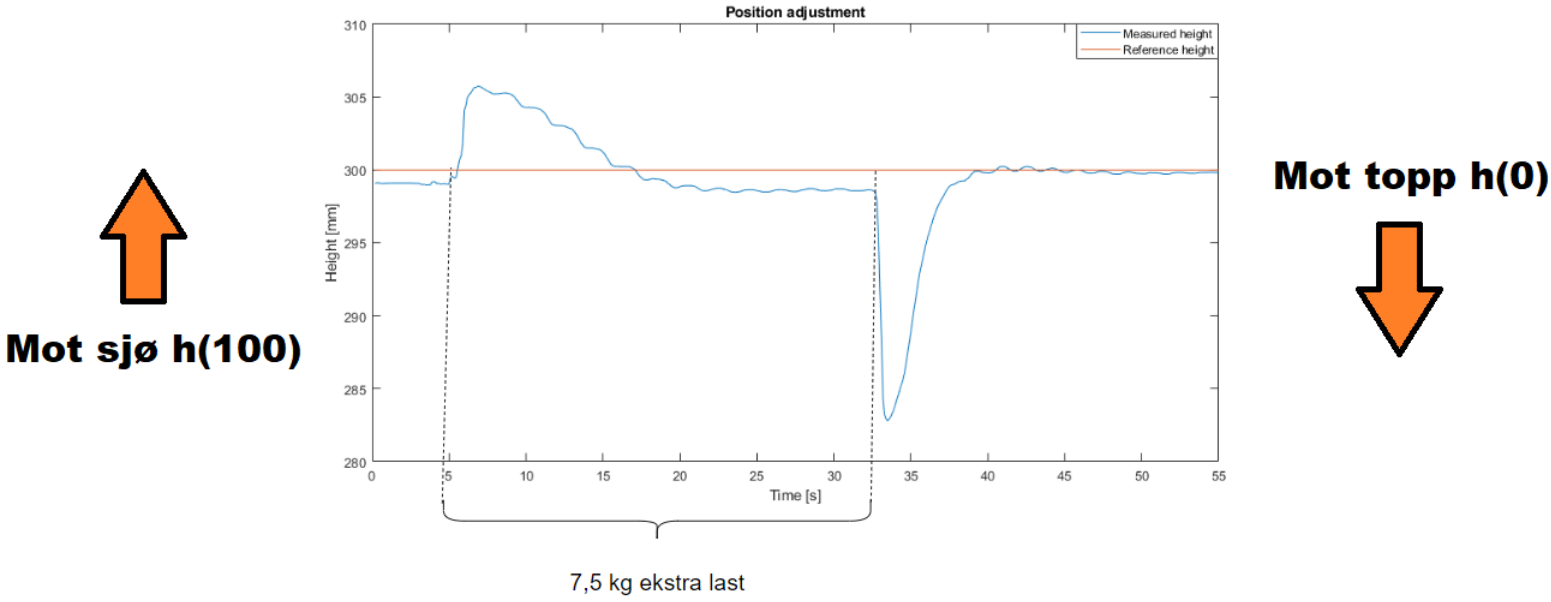


Figure 4.11: Lastkompensasjon: Figuren viser hvordan regulatoren kompenserer for en lastendring på 7,5kg. Lastendringen skjer ved 5sek og fjernes ved 32sek. $K_p = 2$ og $K_i = 0.01$. Rød linje er høyde referansen og blå er den målte høyden.

Når det blir lagt til last under regulering av posisjon vil den målte høyden stige (forklart tidligere fra figur 4.2) og det oppstår et negativt avvik. Regulatorsignalet blir negativt etter en tid som funksjon av avviket og dermed blir motororienteringen ordnet slik at lasten spoles opp.

Når lasten blir tatt av vil den målte høyden synke (fig. 4.2) og det oppstår et positivt avvik. Regulatorsignalet blir positivt etter en tid som funksjon av avviket og dermed blir motororienteringen ordnet slik at lasten heises opp.

Figur 4.12 viser mer detaljert informasjon rundt testen.

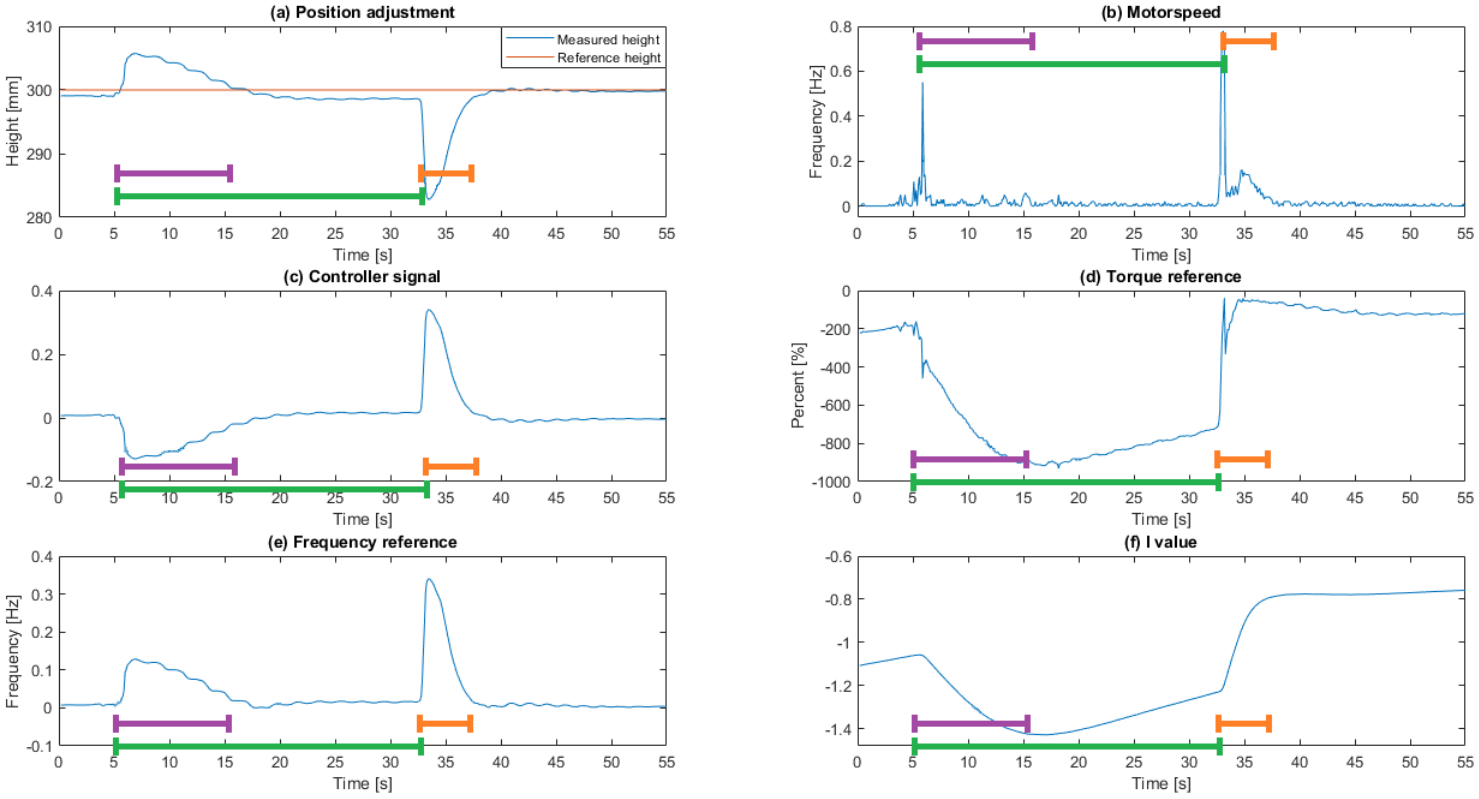


Figure 4.12: Lastkompensasjon med mer informasjon fra testen i figur 4.11. Grønt tidsintervall er med ekstra last. Lilla og oransje tidsintervall er kompenseringstiden etter forstyrrelsen har intruffet.

Videre forklares hva som skjer i hvert intervall i figur 4.12.

Før grønt og lilla intervall[0-5sek]:
Den målte høyden er veldig nær posisjonsreferansen (plott A).

Frekvensreferansen (plott E) er tilnærmet 0 ettersom det ikke er særlig avvik mellom den målte posisjonen og referansen. Motorhastigheten (plott B) er tilnærmet 0. Momentet (plott D) ligger rundt -200.

Lilla intervall [5-15sek] → ekstra last lagt til:
Den målte høyden øker (lasten synker mot sjøen). Dette danner et negativt avvik som gjør at regulator signalet reduseres (plott C). Regulator signalet endrer motororientering og frekvensreferansen. Motorhastigheten følger frekvensreferansen og øker for å spole lasten opp til høydereferansen. Momentet øker betraktelig som følge av den ekstra lasten, hvor det høyeste momentet som blir målt er -900. I-leddet (plott F) blir mer negativ som følge av det negative avviket.

Det tar omtrent 10 sekunder før systemet har kompensert for lastøkningen. Det største avviket generert i målt høyde av denne forstyrrelsen er 6mm.

Grønt intervall uten overlapp av lilla [15-32sek]:
Systemet holder høydereferansen med den ekstra lasten. Momentet ligger mye høyere fordi det er mye mer vekt på motorens momentarm. Dynamikken mellom avvik og referansen endrer I-leddet.

Oransje intervall [32-37sek] → ekstra last fjernet:
Når lasten reduseres vil den målte høyden synke (lasten heves mot trinsene). Avviket blir veldig positivt som endrer regulatorsignalet. Motorhastigheten følger frekvens referansen som igjen følger regulatoren. Momentet synker betraktelig som følge av at lastens vekt blir redusert. I-leddet øker som følge av det positive avviket.

Det tar $\approx$ 5 sekunder før systemet har kompensert for lastreduksjonen. Det største avviket i målt høyde av denne forstyrrelsen er 16mm.

Etter oransje intervall [37-55sek]:
Situasjonen her er lik som før grønt og lilla intervall. Det eneste som har blitt endret er I-leddet som har blitt mer positivt som følge av det store positive avviket når den ekstra lasten ble tatt av i oransje intervall.

Vi kan gjennom denne testen se at posisjonsregulatoren har gode kompenseringsegenskaper. Den vil ved forstyrrelse skapt av lastøkning og lastreduksjon klare å regulere posisjonen tilbake til referansehøyden.

4.4.3 'Skogestads'-tuning metode for PID-regulator

Det ble utført skogestads metode for å finne gode verdier for K_p, T_i og T_d.

Finner reguleringsparametre:

Vi bruker τ og K som ble funnet i delkapittel 4.3.1.

Ved utregning av reguleringsparametrene settes Tc lik 0.5 og K1 lik 4 for å kunne respondere bedre på sprang enn forstyrrelser. Dette fordi den interne regulatoren i frekvensomformeren vil ta seg av forstyrrelseskompenseringen raskere. Utregningen av K_p, T_i og T_d for posisjonsreguleringen blir gjort etter *Index 3* i tabell 4.1.

$$Kp = \frac{1}{K \cdot Tc} = \frac{1}{1 \cdot 0,5} = 2 \tag{4.9}$$

Ti er gitt av:

$$Ti = K1 \cdot Tc = 4 \cdot 0,5 = 2 \tag{4.10}$$

Finner K_i:

$$Ti = \frac{Kp}{Ki} \rightarrow Ki = \frac{2}{2} = 1 \tag{4.11}$$

Td er gitt av:

$$Td = \tau = 0,163s \tag{4.12}$$

Kd finner vi av:

$$Kd = Kp \cdot Td = 2 \cdot 0,163 = 0,326 \tag{4.13}$$

4.4.4 Tester 'Skogestads'-metode for PID-regulator

Alle testene under blir utført med verdiene $K_p = 2$, $K_i = 1$ og $K_d = 0.326$ funnet fra Skogestads metode.

Store referanse sprang:

Det ble introdusert større sprang. Dette ble gjort for å se om responsen på store sprang ville gi oversving. Resultatet vises i figur 4.13.

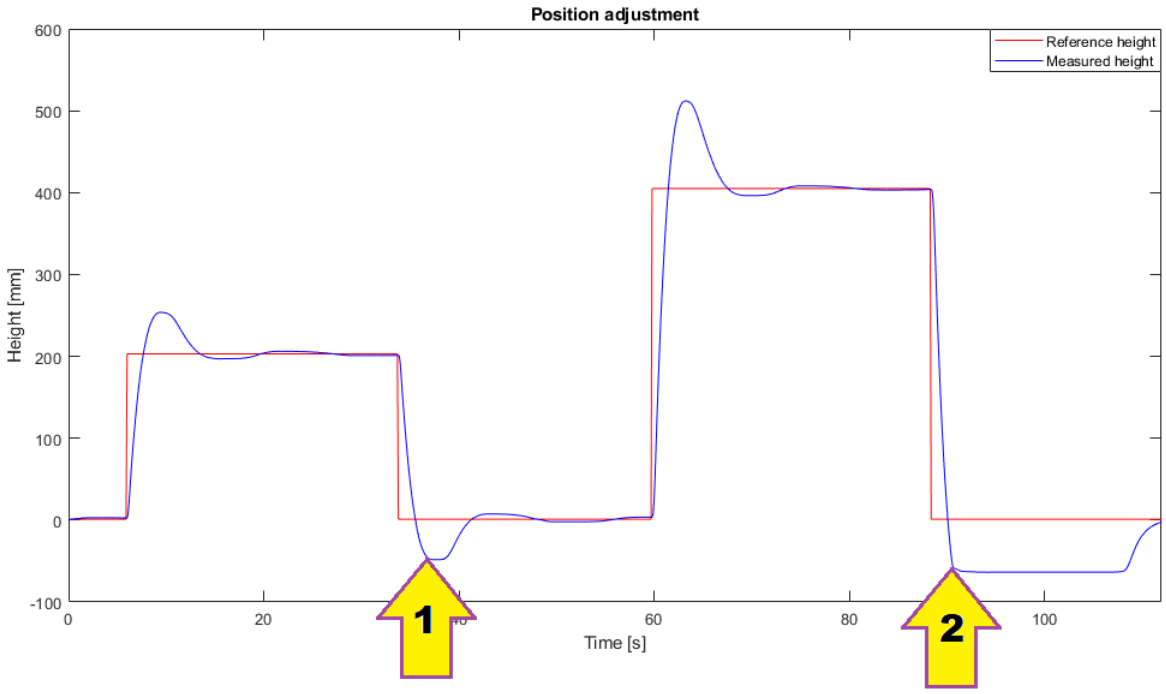


Figure 4.13: Store sprang. $K_p = 2$,$K_i = 1$ og $K_d = 0,326$. Rød linje er høyde referansen og blå er den målte høyden.

Vi ser av den gule pilen markert 1 i figur 4.13 at det blir oversving. Lasten kjøres nesten inn i trinsene. Ved den gule pilen markert 2 kjøres lasten inn i trinsene og blir hengende fast.

Fra dette konkluderte vi at det ikke er nødvendig å teste Skogestad på forstyrrelser ettersom metoden ikke tilfredsstilte ønsket krav om null oversving.

4.4.5 Oppsummering

Det ble utført tester av responsen til en PI-regulator hvor parametrene ble funnet gjennom 'prøv og feil' metoden. Det ble også gjort test av responsen til en PID-regulator hvor parametrene ble funnet gjennom 'Skogestads'-metode. En oppsummering av regulatorene som ble sammenlignet og deres parametre er vist i tabellen 4.4.5:

Sammenligning			
Metode og regulator	**Kp**	**Ki**	**Kd**
'Prøv og feil' PI-regulator	2	0.01	-
Skogestad PID-regulator	2	1	0,326

Responsen til regulatorene ble sammenlignet. Den som tilfredsstilte kravene satt fra delkapittel 4.4 ble valgt. Den valgte metoden ble 'prøv og feil' metoden og de respektive parametrene.

Kapittel 5

Hastighetsregulering med feed forward PI-regulator

I dette kapittelet presenteres teori, løsning og resultater for hastighetsregulering av vinsjsystemet.

Det ble implementert hastighetsregulering av vinsjsystemet. Hastighetsregulatoren vil regulere pådraget til motoren for å oppnå ønsket innspolingshastighet. Ved lastøkning eller annen forstyrrelse under innpoling vil regulatoren øke pådraget til motoren for å vedlikeholde hastigheten.

Pseudokode under viser hvordan hastighetsregulatoren er bygget opp.

Kodeutdrag 5.1: Pseudokode hastighetsregulering

```
// 1. Bruker setter ønsket innspolingshastighet til lasten i mm/s via HMI skjermen.
// 2. Den ønskede verdien i mm/s brukes i en matematisk utregning for åutelede en
    varierende motor referansehastighet i Hz som produserer den konstanten
    lasthastigheten i mm/s.
// 3. Den varierende motor referansehastighet i Hz blir konvertert til en varierende
    motor referansehastighet i pulser.
//4. Den målte motorhastigheten blir konvertert til motorhastighet i pulser.
// 5. Motor hastighetsreferansen i pulser sammenlignes med den målte verdien for
    hastighet i pulser og det utledes et avvik.
// 6. Avviket vil bli regulert av en feedforward PI-regulator som setter pådrag ut
    til motoren.
// 7. Etterhvert som motoren spoler inn vil den endrede radiusen føre til en redusert
    motor hastighetsreferanse i Hz (punkt 2)
```

Figur 5.1 viser blokkskjema av prosessen ved hastighetsregulering.

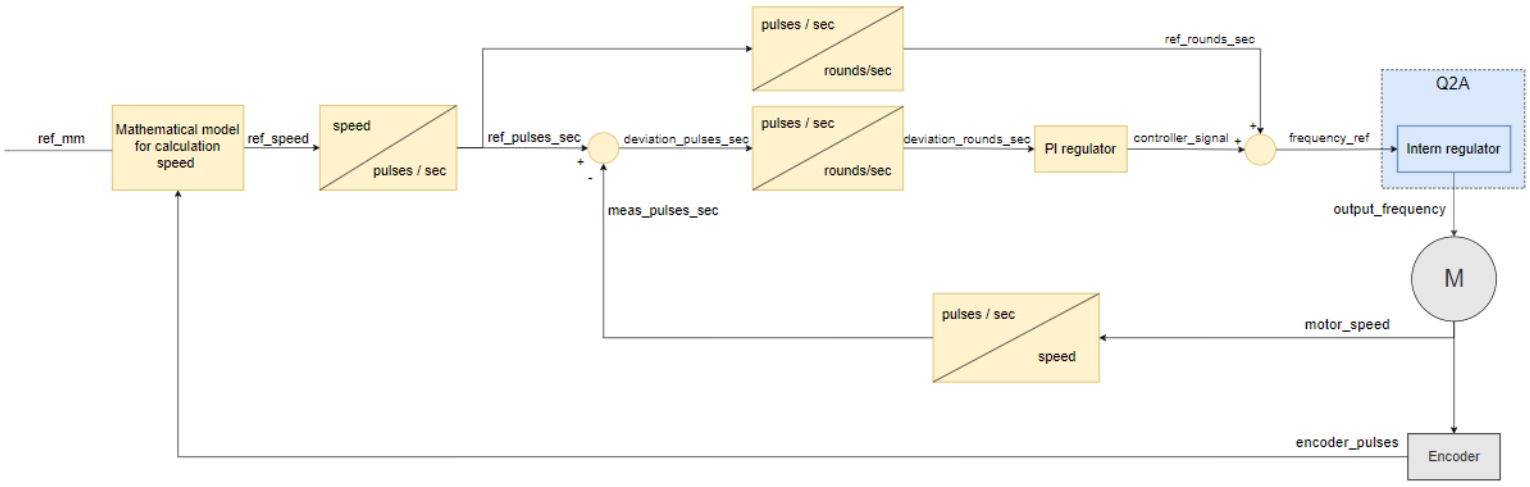


Figure 5.1: Blokkskjema av prosessen ved hastighetsregulering. Gul del er i PLS program, blå del er internt i frekvensomformeren (vist tidligere i fig. 2.6) og grå er fysiske komponenter.

Brukeren setter en hastighetsreferanse (`ref_mm`) for lasten i mm/s fra HMI panelet.

Pulsene fra enkoderen som er telt av variabelen (`encoder_pulses`) i frekvensomformeren blir anvendt til å oppdatere trommelradiusen. Denne varierende radiusen som funksjon av innspoling vil i den mattematiske modellen bli brukt sammen med en konstant last hastighetsreferansen (`ref_mm`) til å lage en varierende motor hastighetsreferanse (`ref_speed`). Denne hastigheten må motoren gå med for å flytte lasten ønsket lengde pr sek.

Last referansehastigheten (`ref_speed`) blir konvertert til en representasjon i pulser per sekund (`ref_pulses_sec`). Motorhastigheten (`motor_speed`) blir også konvertert til antall pulser avgitt fra enkoderen per sekund (`meas_pulses_sec`) og sammenlignes med referansen (`ref_pulses_sec`). Avviket (`deviation_pulses_sec`) mellom referansen og målt hastighet blir videre konvertert fra pulser til runder per sekund (`deviation_rounds_sec`). Basert på denne avviksverdien vil PI-regulatoren beregne pådragsignalet (`controller_signal`). I en feed forward PI-reguleringsløyfe vil reguleringsignalet (`controller_signal`) adderes med referanseverdien (`ref_rounds_sec`), som her er representert i rotasjoner per sekund. Resultatet etter adderingen settes som frekvensreferanse (`frequency_ref`) inn til den interne reguleringsløyfen i frekvensomformeren. Den interne sløyfen (fig. 2.6) beregner pådragsignalet ut til motoren (`output_frequency`), for å justere hastighetsavviket.

5.1 Teori

For å få et utrykk for vaierens bevegelse i lengde per sekund, kombineres vaierlengden som spoles ut for en gitt runde fra likning 3.4 med en hastighetet [Hz] som motoren går med. Utrykket blir som vist i likning 5.1.

$$Vaierlengde\ mm/s = motorhastighet \cdot Vaierlengde(x)$$

$$Vaierlengde\ mm/s = motorhastighet \cdot 2\pi \cdot (r_{trommel} + [d_{vaier} \cdot (16 - x)]) \tag{5.1}$$

Ettersom vi er interessert i lastens bevegelse, kombineres lastdistansen fra likning 3.6 med en hastighet motoren må gå med. Utrykket for lastens bevegelse i lengde per sek blir som vist i likning 5.2.

$$Lasthastighet\ mm/s = motorhastighet \cdot Lastdistanse(x)$$

$$Lasthastighet\ mm/s = motorhastighet \cdot \frac{2\pi \cdot (r_{trommel} + [d_{vaier} \cdot (16 - x)])}{6} \tag{5.2}$$

Det ble vist i figur 3.2 i kap. 3.1 at motoren spoler ut mest vaier per rotasjon i topp og mindre vaier desto lenger ned lasten senkes. Ved innspoling vil det dermed spoles inn minst vaier i starten av innspolingsfasen og mer vaier etter hver rotasjon som spoles inn.

For å få lasten til å bevege seg med en konstant lengde per sekund må farten reduseres som en funksjon av økende runder med vaier som spoles inn på trommelen. Ønsket innpolingshastighet (i lengde per sekund) er den verdien som settes av brukeren via HMI skjermen. Likning 5.2 kan endres slik at utregningen vil gi den ukjente motor hastigheten som må til for å produsere den ønskede lasthastigheten. Utregningen blir som vist i likning 5.3.

$$motorhastighet = \frac{Lasthastighet\ mm/s \cdot 6}{2\pi \cdot (r_{trommel} + [d_{vaier} \cdot (16 - x)])} \tag{5.3}$$

Figur 5.2 illustrerer likning 5.3.

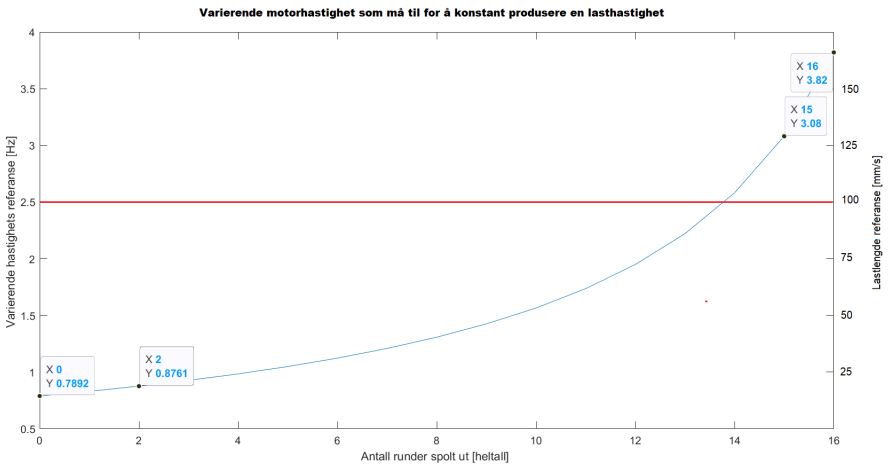


Figure 5.2: Figuren viser den varierende motorhastigheten i Hz som må til for å oppnå ønsket konstant lasthastighet (100 mm/s). Motorhastigheten er en funksjon av varierende radius på trommel. Variabelen 'x' representerer antall runder spolt ut fra trommelen ifra topposisjon.

Fra figuren 5.2 er det mulig å se at for å endre lastenhastigheten med konstant 100mm/s er motor hastigheten som kreves ved x=16 (mye utspolt vaier, liten total radius på trommel) mye større enn ved x=2 (lite utspolt vaier, stor total radius på trommel).

5.1.1 Feedforward PI-Regulator

Feedforward PI-regulatoren består av et proporsjonal ledd (P-ledd) som bruker avviket i kombinasjon med justeringsfaktoren K_p og et integrerende ledd (I-leddet) kombinert med justeringsfaktoren K_i. I-leddet fungerer som et minne over alle avvikene som har oppstått. Summen av PI-leddet legges i sammen med referanseverdien for motorhastighet.

Feedforward PI-regulatoren vises i likningen 5.4 :

$$u(t) = Kp \cdot e(t) + Ki \cdot \int_0^t e(t)\,dt \; + motorhastighet \tag{5.4}$$

Den numerisk integrasjonen basert på 'Eulers forovermetode' som ble brukt for å lage I-leddet er vist i likning 5.5:

$$u\_I(t) = u\_I(t-1) + e(t) \cdot Ts \tag{5.5}$$

Hvor variablene i liking 5.5 er:

- u_I(t) - Det nye I-leddet
- u_I(t-1) - Det gamle I-leddet
- e(t) - Avvik i hastighet
- Ts - Tidssteget i sekund

I arbeidet med oppgaven ble det i utgangspunktet valgt å hastighetsregulere med PI-regulator. Den fungerte tilstrekkelig til å følge en hastighetsreferanse greit, men viste seg å ikke tilfredsstille forstyrrelseskompensering. Derfor ble det brukt utviklet en feedforward PI-regulator som fungerte til både å følge en hastighetsreferanse og justere for forstyrrelser.

I figurene 5.3 og 5.4 blir det vist forskjellen mellom vanlig PI-regulator og feed forward PI-regulator på en forenkelt og lett forståelig måte.

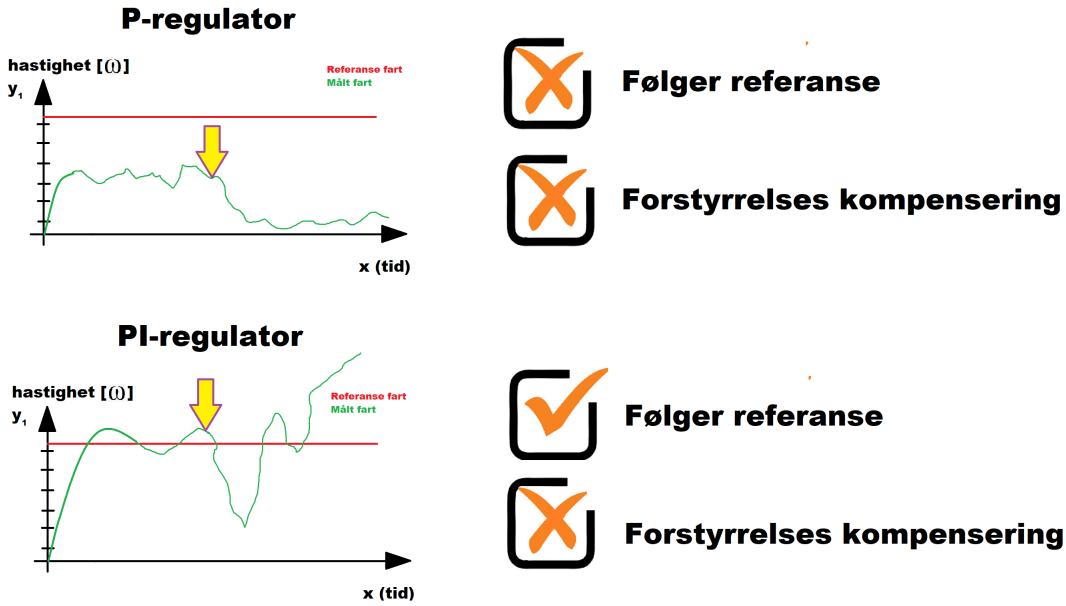


Figure 5.3: Hastighetsregulering: P-regulator og PI-regulator. Gul pil er forstyrrelsen. Rød linje er referanse hastighet. Grønn linje er målt hastighet.

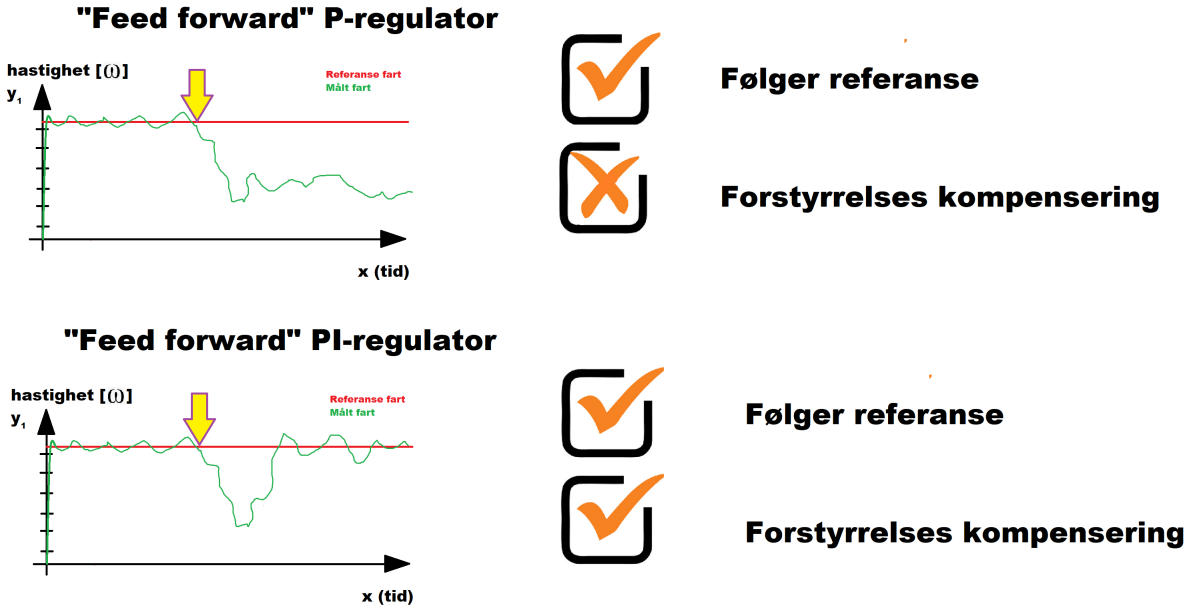


Figure 5.4: 'Feed forward' Hastighetsregulering: P-regulator og PI-regulator. Gul pil er forstyrrelsen. Rød linje er referanse hastighet. Grønn linje er målt hastighet.

Fra figurene vises hvorfor feed forward PI-regulator gir best regulering av hastighet; den vil kunne følge referansen uavhengig av forstyrrelser. Det blir gjort sammenligninger ved forskjellige tester med vanlig PI-regulator og feed forward PI-regulator senere i delkapittel 5.3.4.

5.2 Kode for hastighetsregulator

I dette delkapittelet vil koden for hastighetsregulatoren presenteres. For å kunne hastighetsregulere ble det utviklet en HMI skjerm som vist i figur 5.5. På denne skjermen kan brukeren skrive inn ønsket innspolingshastighet (i mm pr sekund) og verdier for K_p og K_i. Hastighetsregulatoren vil starte når brukeren trykker inn knappen `Speed control`.

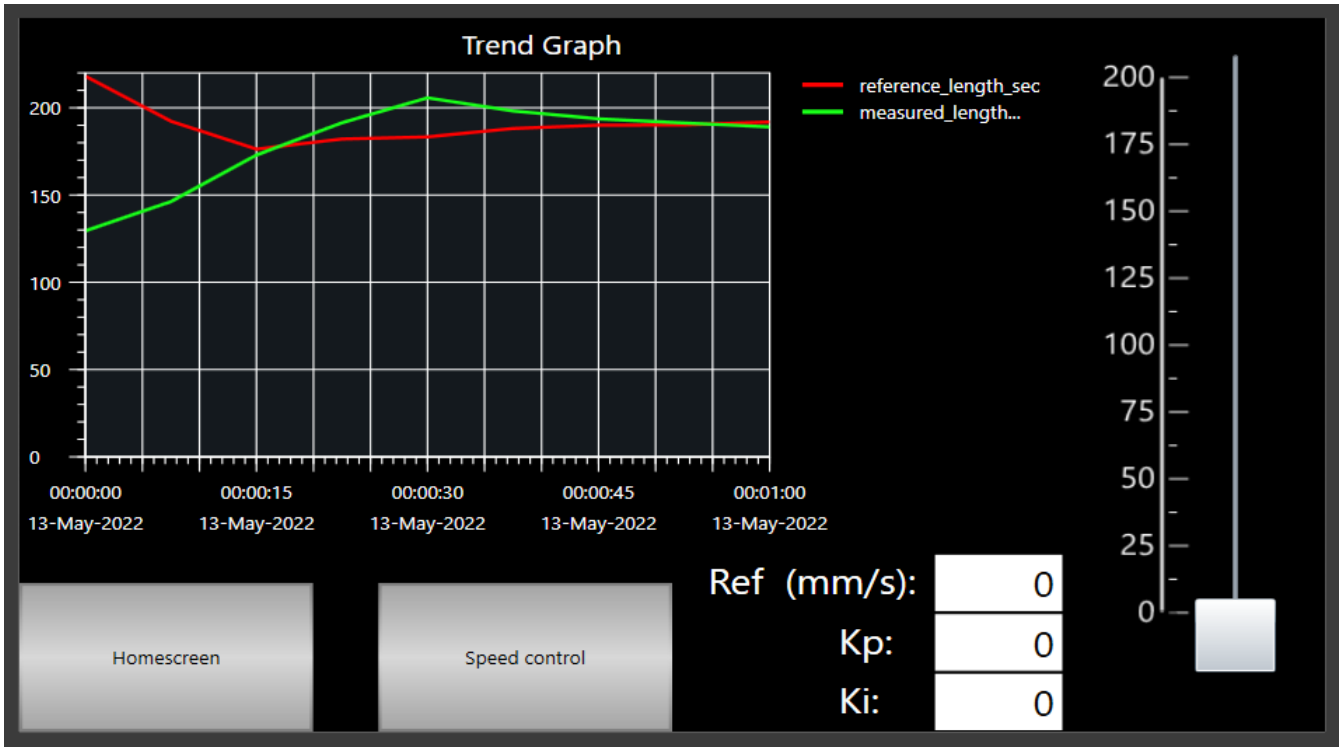


Figure 5.5: I figuren vises HMI skjermen for hastighetsregulering. Brukeren kan på denne skjermen starte innspoling ved å trykke på knappen `Speed control`. `Homescreen` knappen vil føre brukeren tilbake til hjemskjermen. Ønsket innspolingshastighet for lasten i mm/s settes gjennom scrollbaren (som går fra 0-200 mm/s) eller numerisk (`Ref (mm/s)`). Brukeren kan i tillegg sette verdier for K_p og K_i. I grafen vises referanse og målt innspolingshastighet. Verdiene i grafen er ikke reelle; bildet tatt ved simulering i Sysmac studio.

Det første programmet gjør ved start er å sjekke programminnene for å overføre verdi for antall pulser og rundemåling som vist i kodeutdrag 5.2. I linje 32 og 36 spør programmet om det forrige programmet som ble kjørt var posisjonsregulering (`HMI_page_position_memory`) eller manuell kjøring (`HMI_page_manual_memory`). Deretter blir det i linje 33-34 eller 37-38 overført antall pulser avgitt fra enkoderen og rundene spolt ut fra trommelen. Det forrige programminnet settes til `FALSE` og programminnet til hastighetskontroll settes til `TRUE` i linjene 40-42.

Kodeutdrag 5.2: Utdrag fra kode; Speed control

```
30 IF HMI_page_speedCTL_memory = FALSE THEN
31 // Gets current number of pulses and rounds from position adjustment
32    IF HMI_page_position_memory = TRUE THEN
33       speedCTL_difference_counter := position_difference_counter;
34       speedCTL_round_counter := position_round_counter;
35 // Gets current number of pulses and rounds from manual driving
36    ELSIF HMI_page_manual_memory = TRUE THEN
37       speedCTL_difference_counter := manual_difference_counter;
38       speedCTL_round_counter := manual_round_counter;
39 // Sets speed control as last page
40    HMI_page_position_memory := FALSE;
41    HMI_page_manual_memory := FALSE;
42    HMI_page_speedCTL_memory := TRUE;
43    END_IF;
44 END_IF;
```

For å oppdatere rotasjonsendringen til motoren brukes differansen mellom to pulser avgitt fra enkoderen som vist i kodeutdrag 5.3. Logikken ble forklart tidligere i delkapittelet 3.2.1.

Kodeutdrag 5.3: Utdrag fra kode; Speed control

```
70 // Difference between two pulses from the encoder = rotation change of the motor
71 encoder_value_now := encoder_pulses;
72 encoder_difference_value := encoder_value_now - encoder_value_old;
73 encoder_value_old := encoder_value_now;
```

For å oppdatere posisjonen til lasten telles alle differanseverdiene (`encoder_difference_counter`) som vist i kodeutdrag 5.4. Antall runder spolt ut fra trommelen (`speedCTL_round_counter`) kan finnes ved å dele alle de talte differanseverdiene på 8000. I delkapittel 3.1 ble det beskrevet at 8000 pulser avgitt fra enkoderen tilsvarer 1 omdreining på motoren.

Kodeutdrag 5.4: Utdrag fra kode; Speed control

```
75 // Update total number of pulses
76 speedCTL_difference_counter := speedCTL_difference_counter + encoder_difference_value;
77 // Update number of rounds
78 speedCTL_round_counter := speedCTL_difference_counter/8000;
```

Heltallsvariabelen `whole_round_counter` i kodeutdrag 5.5 representerer antall hele runder spolt ut fra trommelen (x) som forklart tidligere i delkapittel 3.1.

For å finne antall hele runder som er spolt ut fra trommelen avrundes variabelen `speedCTL_round_counter` i linje 76. Skulle rundene som er spolt ut være over halvveis til neste helrunde vil variabelen for antall hele runder (`whole_round_counter`) bli rundet opp. Det blir derfor i linje 122 spørt om den avrundede variabelen er større enn antall runder spolt ut fra trommelen; Ved ja vil det i linje 123 trekkes fra 1 for å ha korrekt antall helrunder med vaier som er spolet ut.

Kodeutdrag 5.5: Utdrag fra kode; Speed control

```
120 // Logic for calculating whole rounds
121 whole_round_counter := Round(speedCTL_round_counter);
122 IF LINT_TO_REAL(whole_round_counter) > speedCTL_round_counter THEN
123    whole_round_counter := whole_round_counter - 1;
124 END_IF;
```

Ønsket lengdeendring for lasten i mm/s er definert av variabelen `ref_display` gjennom det numeriske displayet, og `ref_scrollbar` gjennom scrollbaren til HMI skjermen. I linje 135-139 i kodeutdrag 5.6 sjekker programmet om brukeren har valgt det numeriske displayet eller scrollbaren til å sette referanseverdien, verdien blir skrevet videre til variabelen `ref_mm`.

Kodeutdrag 5.6: Utdrag fra kode; Speed control

```
130 // Referance length in mm/sec from either numeric display or scrollbar
131 ref_display := HMI_userinput_ref_speed_display;
132 ref_scrollbar := HMI_userinput_ref_speed_scrollbar;
133
134 //  Logic for input method: numeric display or scrollbar
135 IF ref_display > 0 AND ref_scrollbar = 0 THEN
136    ref_mm := ref_display;
137 ELSIF ref_scrollbar > 0 AND ref_display = 0 THEN
138    ref_mm := ref_scrollbar;
139 END_IF;
```

Videre vil programmet i kodeutdrag 5.7 konvertere referanseverdien (`ref_mm`) til en motorhastighet referanse via likning 5.3 som beskrevet tidligere. Utregningen skjer i linje 146, hvor `drum_radius` (25mm) og `wire_diameter` (6mm) er satt som initial verdier ved start av programmet. Det kompenseres for trinsene i systemet i linje 148.

Utregningene for heltallsvariabelen "x" (`whole_round_counter`) ble gjort på samme måte som i kalibrering nevnt i kodeutdrag 3.8, linje 66 og 67.

Referanseverdien i motorhastighet konverteres videre til en verdi i antall pulser avgitt fra enkoderen pr sekund på linje 151.

Kodeutdrag 5.7: Utdrag fra kode; Speed control

```
145 // Wanted motor turns pr sec
146 ref_speed := ref_mm / ((2*3.14)
        *(drum_radius+(wire_diameter*(16-whole_round_counter)));
147 // Castor compensation
148 ref_speed := ref_speed*number_of_pulleys;
149
150 // Converts the referance speed to pulses pr sec
151 ref_pulses_sec := ref_speed * 8000;
```

På linje 156 i kodeutdrag 5.9 hentes motorhastigheten ut fra frekvensomformeren og konverteres til hastighet i form av pulser avgitt fra enkoderen pr sek.

Kodeutdrag 5.8: Utdrag fra kode; Speed control

```
155 // Motor speed in pulses pr sec
156 meas_pulses_sec := motor_speed*8000;
```

Ved å sammenligne motorhastighets referansen med den målte motorhastigheten oppstår et avvik (`deviation_speed_pulses`) i pulser mellom disse som vist i linje 158 i kodeutdraget 5.9. Avviket konverteres videre fra pulser pr sekund til runder per sekund (`deviation_rounds_sec`) ved å dele på 8000 (1 omdreining).

Kodeutdrag 5.9: Utdrag fra kode; Speed control

```
157 // Deviation
158 deviation_pulses_sec := (ref_pulses_sec - meas_pulses_sec);
159 deviation_rounds_sec := deviation_pulses_sec/8000;
```

Kodeutdrag 5.10 viser beregningen av integralfunksjonen til I-leddet til hastighetsregulatoren. I linje 169-173 defineres tidsskrittet og i linje 177 beregnes integralfunksjonen.

Kodeutdrag 5.10: Utdrag fra kode; Speed control

```
167
168 // Timestep caluclation
169 time_new := GetTime();
170 time_step:= SUB_DT_DT(time_new, time_old);
171 time_old := time_new;
172 time_step_nano := TimeToNanoSec(time_step);
173 time_step_nano_real := LINT_TO_REAL(time_step_nano) / 1000000000;
174
175 // Integral function of the deviation
176 I_value_new := I_value_old + (deviation_rounds_sec * time_step_nano_real);
177 I_value_old := I_value_new;
```

I kodeutdrag 5.11 beregner 'feed forward' PI-regulatoren et pådragsignal. Verdier for K_p og K_i settes av brukeren gjennom HMI skjermen (fig. 5.5). Justeringsparameterene brukes sammen med avviket (`deviation_rounds_sec`), utregnet I-verdi (`I_value`) og referanseverdien (`ref_rounds_sec`) til å beregne et reguleringspådrag (`controller_signal`).

Kodeutdrag 5.11: Utdrag fra kode; Speed control

```
175 // Feedforward PI controller
176 controller_signal := ref_rounds_sec +
    (HMI_userinput_kp_speedCTL*deviation_rounds_sec)+(HMI_userinput_ki_speedCTL *
    I_value);
```

Regulator signalet fra 'feed forward' PI-regulatoren settes som frekvensreferanse (`frequency_ref`) videre til den interne sløyfen i frekvensomformeren, som vil regulere pådraget til motoren. Avviket vil dermed justeres.

Kodeutdrag 5.12: Utdrag fra kode; Speed control

```
187 frequency_ref := ABS(controller_signal);
```

5.3 Resultater av hastighetsregulering

I dette delkapittelet vil resultatene fra ulike tester ved hastighetsregulering av systemet presenteres. Først sammenlignes en ordinær PI-regulator med en feed forward PI-regulator. Reguleringsparametrene for disse blir funnet via 'prøv og feil'-metoden. Videre blir disse regulatorene sammenlignet opp mot hverandre i ulike tester og den responsen som tilfredsstiller kravene (vist under) best blir valgt.

Deretter vil beste regulatortype sammenlignes opp mot responsen til reguleringsparametre funnet fra 'Skogestads'-metode.

Vi har satt følgende krav for prosessen:

- Litt oversving.
- Rask respons.
- Håndterer forstyrrelser.

Regulatoren som presterte best ifht. til kravene er vist mer i detalj under delkapitlene.

Ved hastighetsregulering er ikke oversving like problematisk som ved possisjonsregulering, men vi ønsker fortsatt å ha minst mulig for å forhindre for brå akselerasjoner.

5.3.1 'Prøv og feil'-tuning metode for PI-regulator

Det ble utført "prøv og feil" metoden for PI-regulatoren, som går ut på å teste flere ulike parameterverdier for K_p og K_i. Vi observerer responsen til den målte verdien opp mot referansen og finner de parameterverdiene som tilfredsstiller kravene vi har satt for prosessen.

Valg av Kp:

For å finne prosessens respons på bare P-leddet ble I-leddet satt ut av drift ved å sette ($K_i = 0$). Verdiene som ble testet er vist i tabell 5.1:

Kp test				
Kp	1	2	3	5
Ki	0	0	0	0

Tabell 5.1: Justeringsfaktorer som ble testet i forsøket.

Testene startet i bunnposisjonen til det kalibrerte området (h100). Lasthastigheten ble satt til 100 mm per sekund. Resultatene for forsøket vises på grafene i figur 5.6.

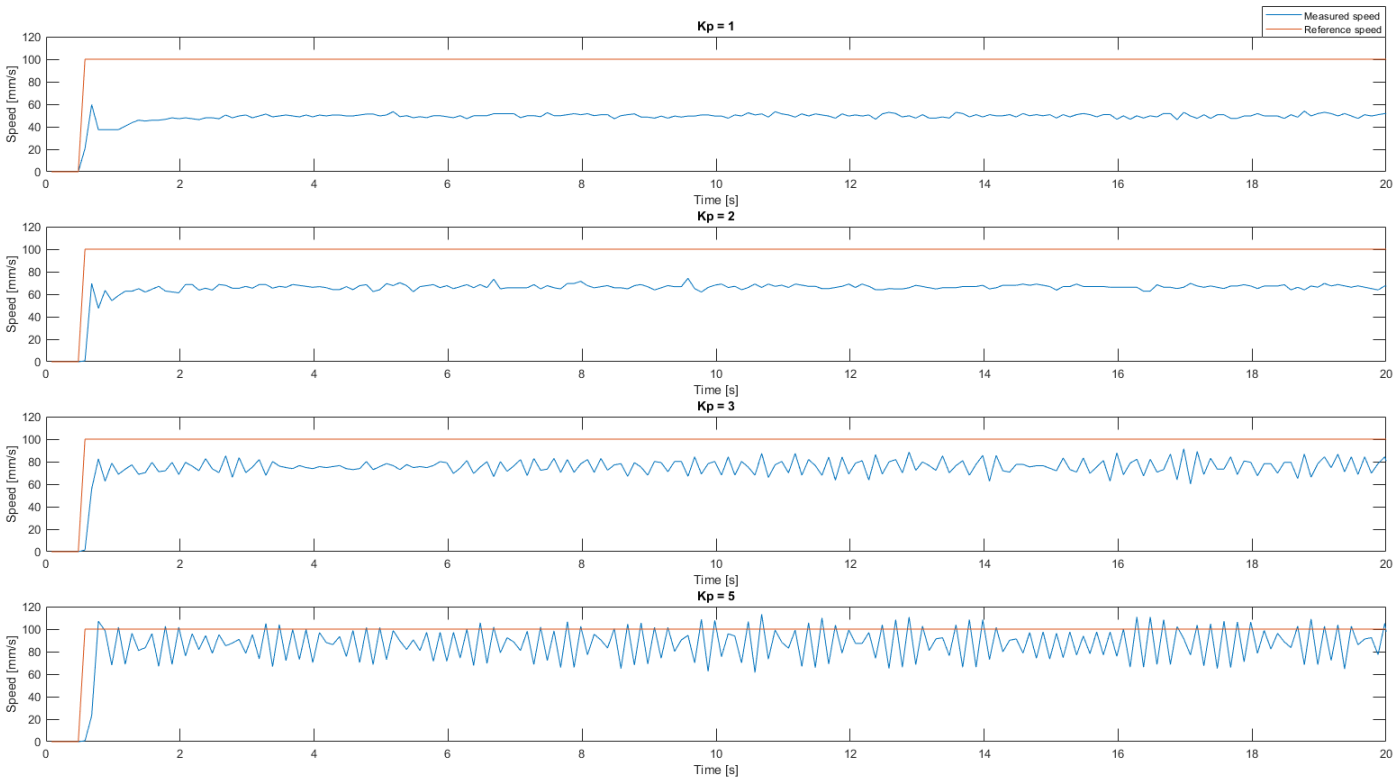


Figure 5.6: PI regulator: 'Prøv og feil' metode - Forskjellige K_p verdier for hastighetsregulering. Rød linje er referanse mm/s og blå linje er målt mm/s.

Fra figur 5.6 ser vi at regulering med $K_p = 1$ ikke klarer oppnå ønsket innspolingshastighet. Regulering med $K_p = 5$ bruker 0.3 sekund på å nå referansen, men gjennonsnittet av alle målingene ligger under referansen og det dannes mye svingninger i den målte hastigheten.

Oppjustering av K_p gir raskere respons, men gir større svingninger i den målte verdien. Fra dette plottet velges $K_p = 2$, selv om den ikke når helt opp. Dette fordi vi vet at K_i leddet vil ta seg av det resterende avviket.

Valg av Ki:

For å finne den beste K_i verdien som fungerer sammen med den funnede K_p verdien ble det testet ulike verdier som vist i tabell 5.3.1.

Ki test						
Kp	2	2	2	2	2	2
Ki	1	2	3	4	5	10

Tabell 5.2: Justeringsfaktorer som ble testet i forsøket.

Prosedyren er lik som når vi fant K_p verdien. Resultatene fra forsøket er vist i figur 5.7.

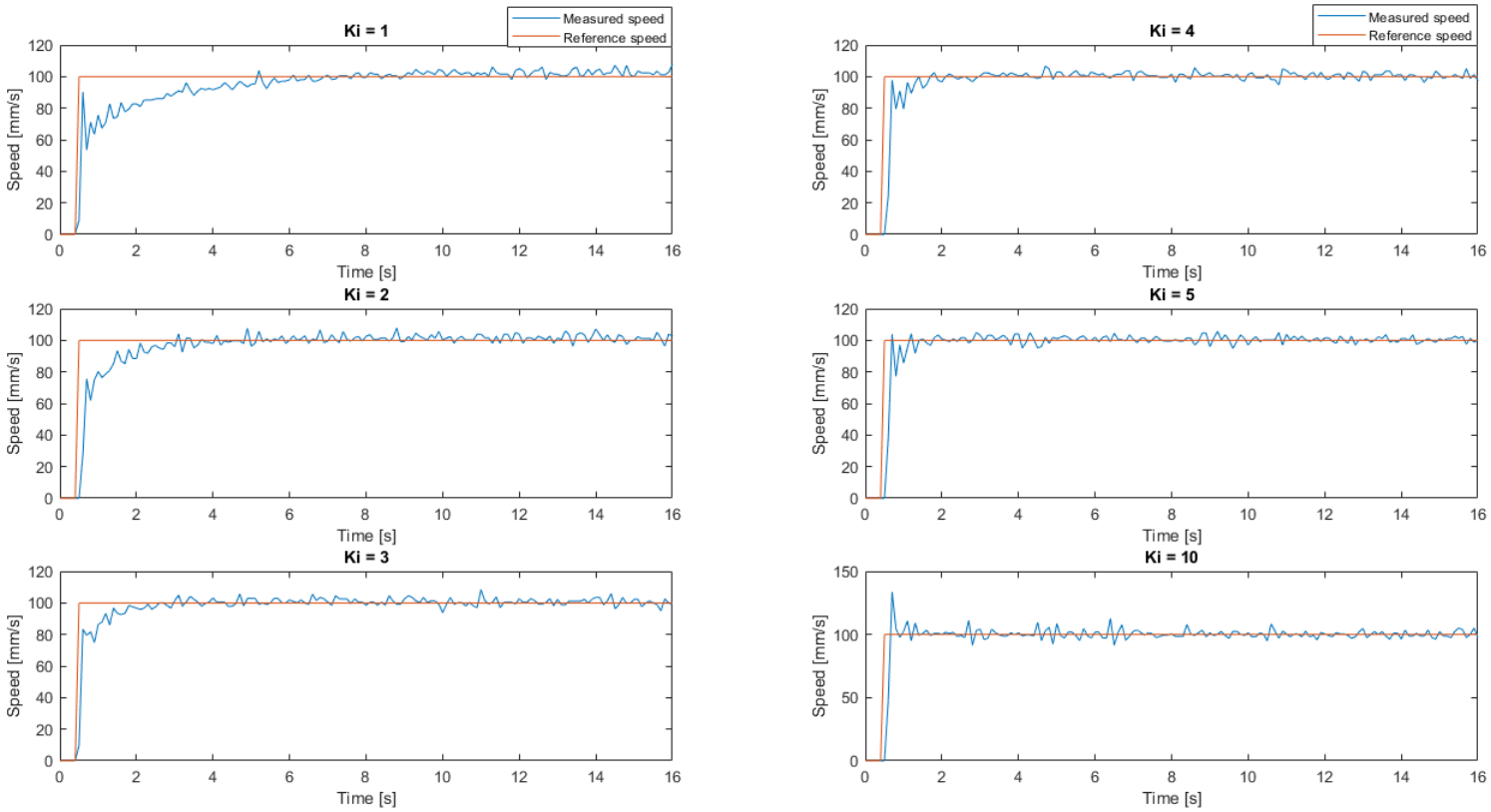


Figure 5.7: PI regulator: 'Prøv og feil' metode. Forskjellige K_i verdier for hastighetsregulering med $K_p = 2$. Rød linje er referanse mm/s og blå linje er målt mm/s

Fra figur 5.6 ser vi at regulering med $K_1 = 1$ klarer å fjerne avviket og nå referansen innen 5 sek. Regulering med $K_i = 10$ bruker 0.2 sekund på å nå referansen, men skaper oversving og mer støyete svingninger i den målte hastigheten.

Oppjustering av K_i gir oss raskere respons, men gir oversving og større svingninger i den målteverdien. Fra dette plottet ser det ut som at $K_i = 5$ var det beste valget siden denne verdien tilfredsstilte kravene om rask respons og lite oversving. Ved kjøring i tregere hastigheter ble det produsert et mer støyete signal. Vi valgte derfor $K_i = 3$ som ga rask respons og lite svingninger.

5.3.2 Tester 'prøv og feil'-metode PI regulator

Alle testene under for PI-regulatoren ble utført med med verdiene $K_p = 2$ og $K_i = 3$ fra 'prøv og feil'-metoden.

Tre forskjellige last hastigheter:

Det ble introdusert større og mindre sprang (50 mm/s, 100mm/s, 150mm/s). Dette ble gjort for å se at responsen ved andre sprang også er like tilfredsstillende. Resultatet vises i figur 5.8.

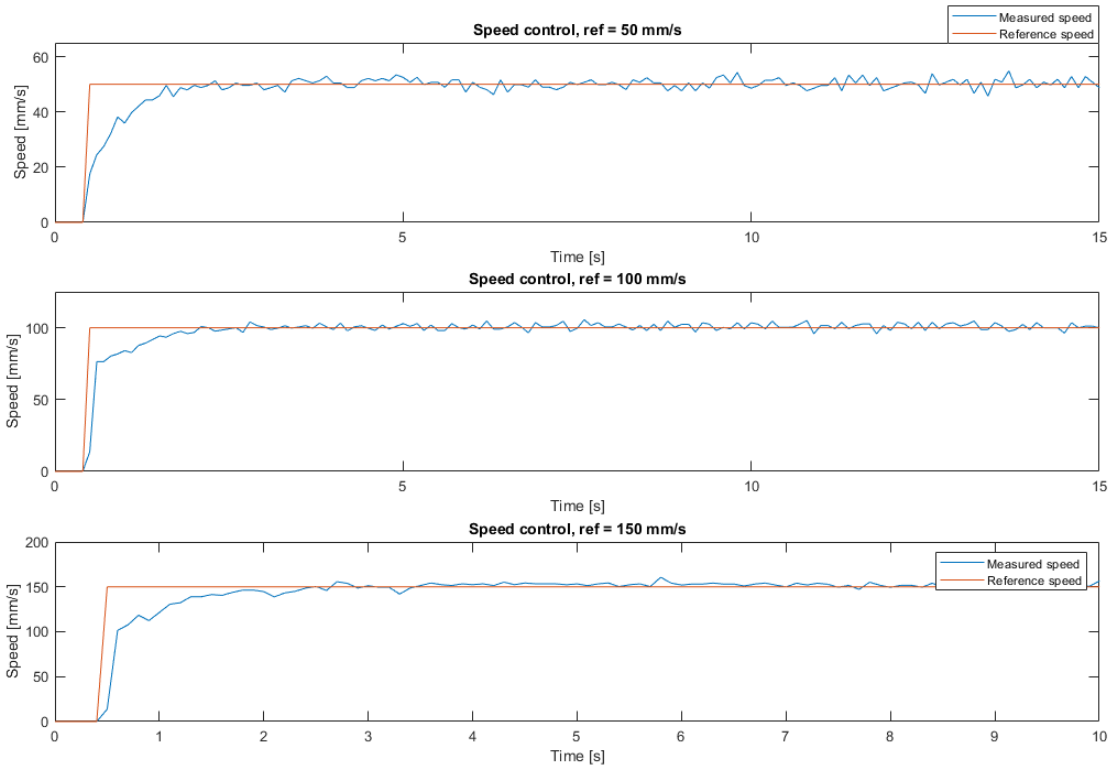


Figure 5.8: PI-regulator: 3 forskjellige hastighetsreferanser - $K_p = 2$ og $K_i = 3$. Rød linje er referanse i mm/s og blå linje er målt mm/s

Fra figur 5.8 ser vi at det tar 1.8 sekund å nå referansen på 50 mm/s. Det tar 2 sekund å nå referansen 100mm/s og 2.5 sekund for å nå referansen 150mm/s.

Lastkompensasjon:

Det ble utført en test av kompenseringsegenskapene til regulatoren. Testen gikk ut på å la hastigheten justeres inn til last hastighetsreferansen, for å deretter legge til 7,5 kg ekstra vekt. Testen er vist i figur 5.9.

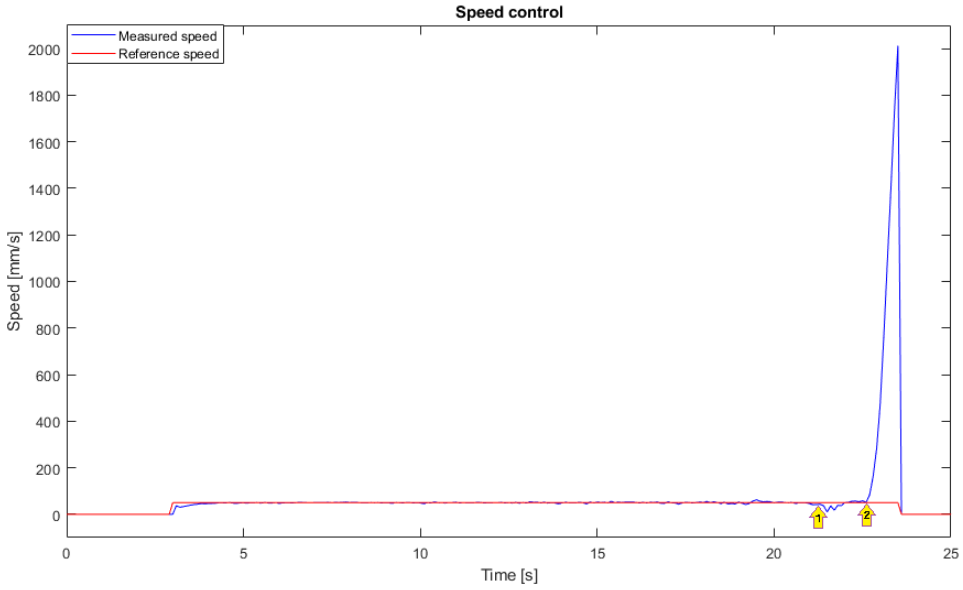


Figure 5.9: PI-regulator: Lastkompensajons test, $K_p = 2$ og $K_i = 3$. Rød linje er referanse mm/s og blå linje er målt mm/s

Vi ser fra figuren at ved gule pilen (markert 1) blir lasten lagt til. Farten synker i en liten periode og regulatoren justerer hastigheten slik at farten når referansen igjen.

Ved den gule pilen (markert 2) fjernes lasten og hastigheten øker betraktelig. Her klarer ikke regulatoren å fjerne avviket. Det er nok mulig å finne en i-leddsbegrensning som gjør det mulig å håndtere denne situasjonen, men vi testet en feed forward PI-regulator istedenfor og sammenlignet denne med PI-regulatoren.

5.3.3 'Prøv og feil'-tuning metode for 'feed forward' PI-regulator

Det ble utført "prøv og feil" metoden for 'feed forward' PI-regulatoren, hvor det testes flere ulike parameterverdier for K_p og K_i.

Valg av Kp:

For å finne prosessens respons på bare P-leddet ble I-leddet satt ut av drift ved å sette ($K_i = 0$). Verdiene som ble testet er vist i tabell 5.3:

Kp test					
Kp	0.1	0.5	1	2	3
Ki	0	0	0	0	0

Tabell 5.3: Justeringsfaktorer som ble testet i forsøket.

Prosedyren for å finne Kp verdien ble utført likt som i delkapittel 5.3.1. Resultatet for forsøket vises i figur 5.6.

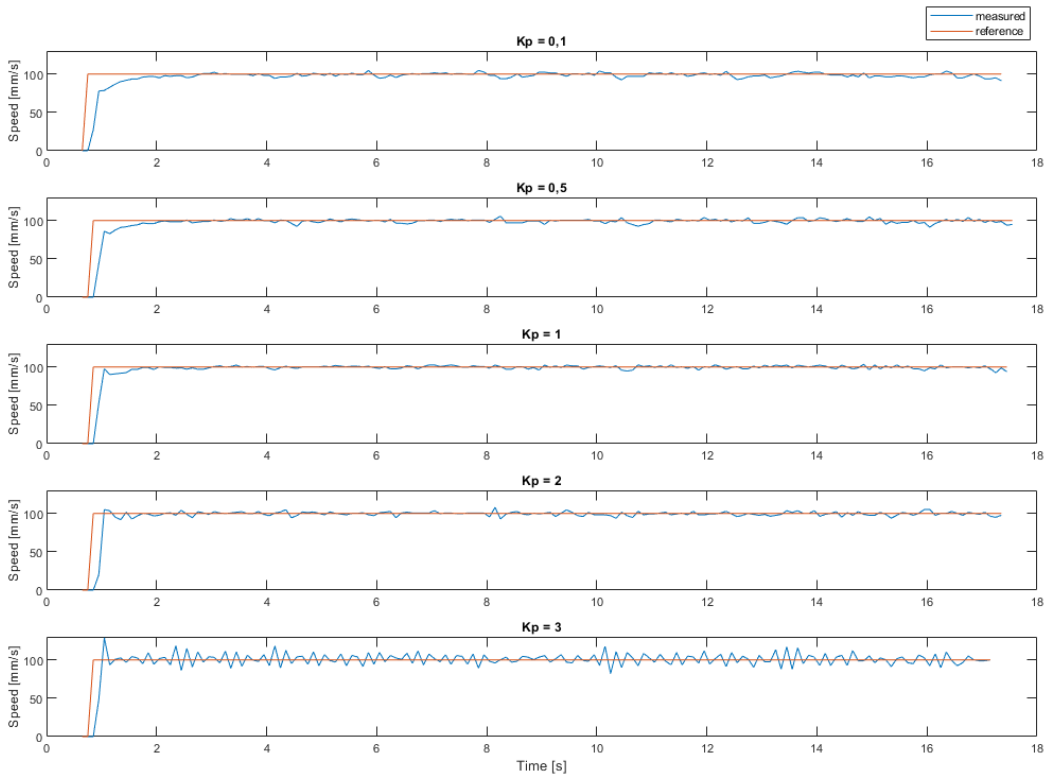


Figure 5.10: 'Feed forward' PI-regulator: 'Prøv og feil' Metode, Forskjellige K_p verdier for hastighetsregulering. Rød linje er referanse mm/s og blå linje er målt mm/s

Fra figur 5.10 observeres det at en høyere K_p verdi skaper raskere regulering, men det vil oppstå mer ustabilitet i form av svingninger i den målte hastigheten.

$K_p = 1$ når referansen innen 2 sek og har små svingninger.

$K_p = 3$ når referansen innen 0.2 sek, men skaper mer svingninger.

Fra plottene velges $K_p = 1$ som har rask respons på 0,3 sek, i tillegg til lite svingninger etter at referansen er nådd.

Valg av Ki:

For å finne den beste K_i verdien som fungerer sammen med K_p verdien ble det testet ulike verdier som vist i tabell 5.3.1.

Ki test					
Kp	1	1	1	1	1
Ki	0.1	0.25	0.5	1	2

Tabell 5.4: Justeringsfaktorer som ble testet i forsøket.

Prosedyren er lik som når vi fant K_p verdien. Resultatene fra forsøket er vist i figur 5.11.

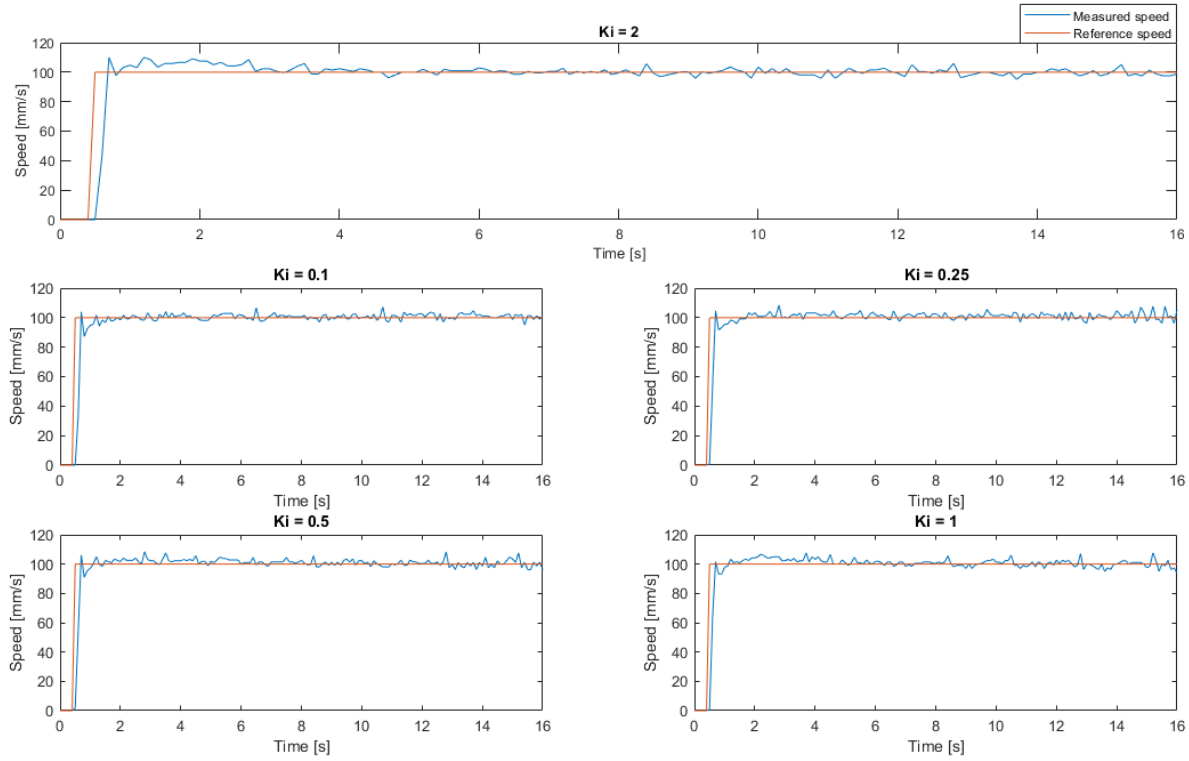


Figure 5.11: 'Feed forward' PI-regulator: 'Prøv og feil' Metode. Forskjellige K_i verdier for hastighetsregulering med $K_p = 1$. Rød linje er referanse mm/s og blå linje er målt mm/s

Fra figur 5.11 ser vi at tiden det tar før den målte hastigheten er ved referansen er 0.1-0.2 sek for alle verdiene for K_i. Det er ikke mye forskjell mellom de ulike verdiene. Vi valgte $K_i = 0.5$.

5.3.4 Tester 'prøv og feil'-metode 'feed forward' PI regulator (detaljert)

Alle testene under for 'feed forward' PI-regulatoren ble utført med verdiene $K_p = 1$ og $K_i = 0.5$ fra 'prøv og feil'-metoden.

Tre forskjellige last hastigheter:

Det ble utført en test av flere sprang for å se at responsen er like tilfredsstillende ved andre hastighetsreferanser. Resultatet vises i figuren 5.12.

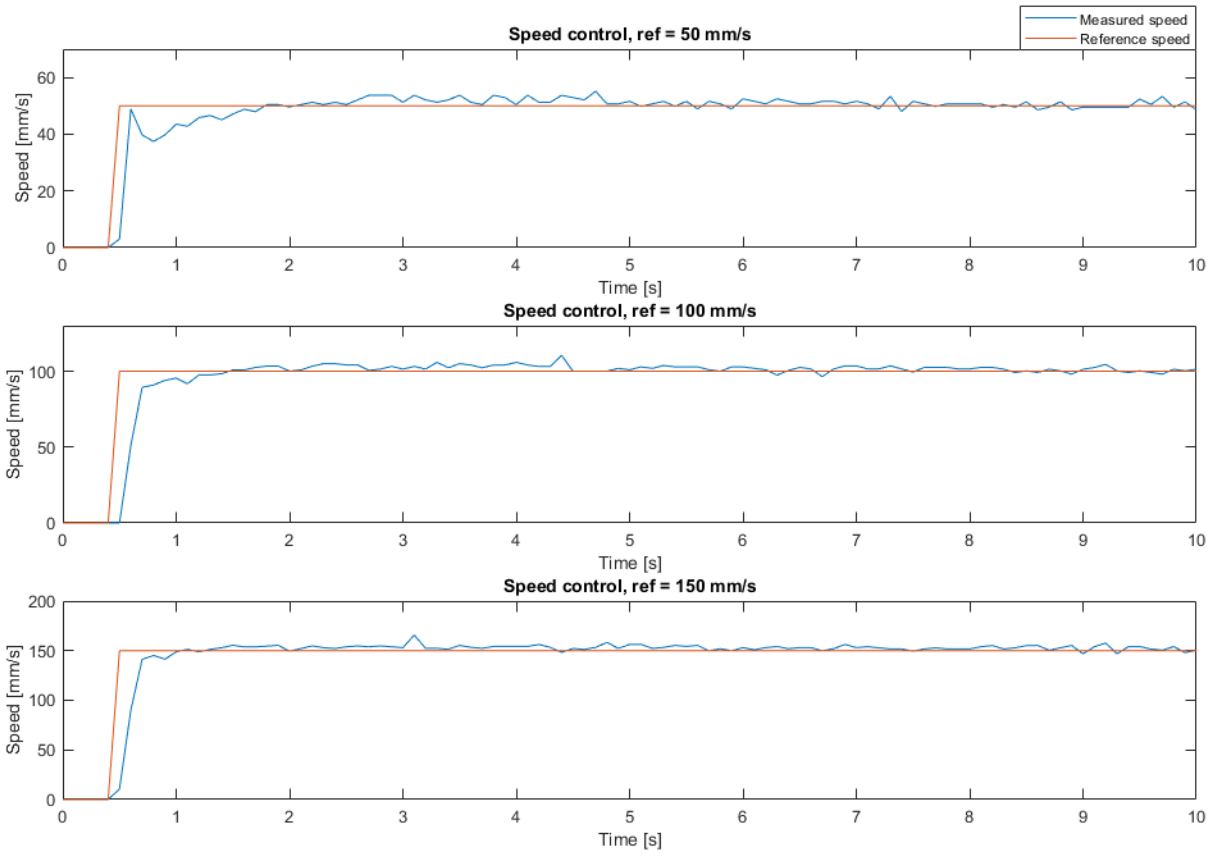


Figure 5.12: 'Feed forward' PI-regulator: 3 forskjellige hastighetsreferanser. $K_p = 1$ og $K_i = 0.5$. Rød linje er referanse i mm/s og blå linje er målt mm/s

Fra figur 5.12 ser vi at det brukes 1.8 sekund for å nå en lastreferanse på 50mm/s. 1.5 sekund for å nå 100mm/s og 1.1 sekund for å nå 150 mm/s.

Figur 5.13 viser flere variabler under testen.

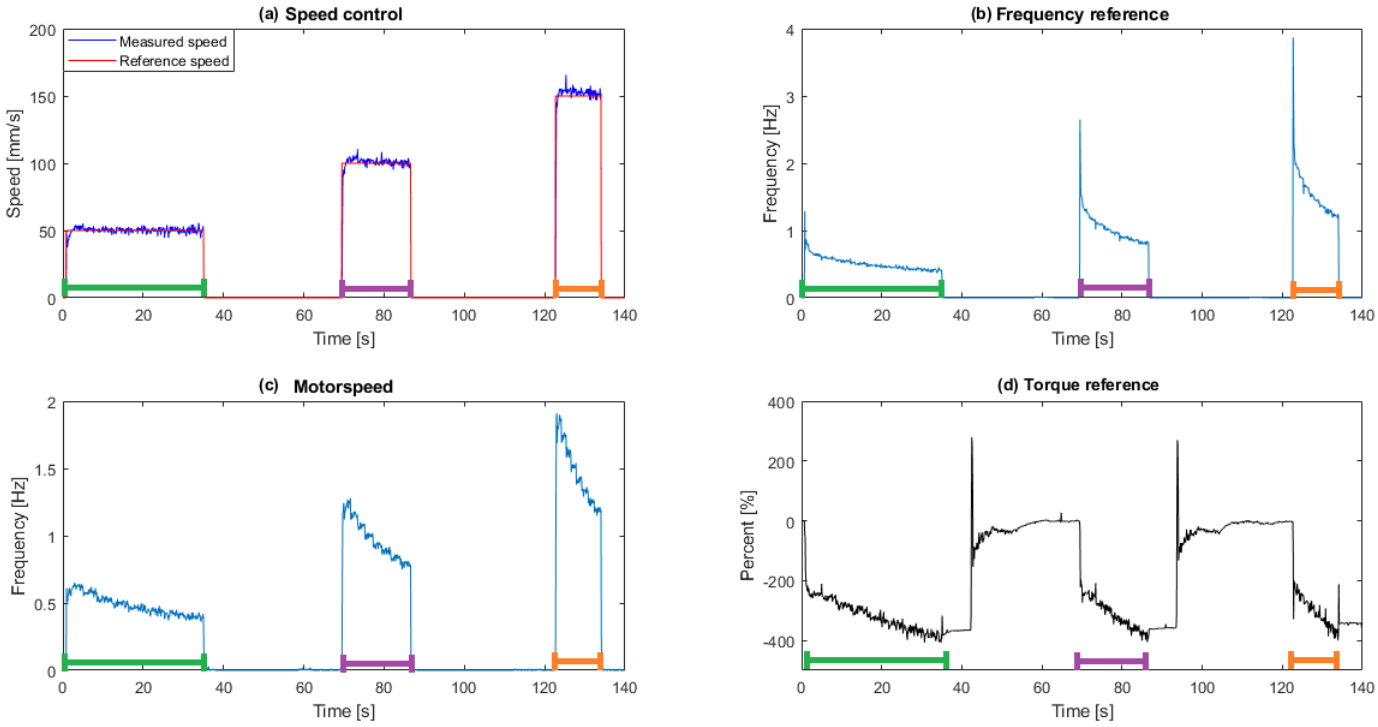


Figure 5.13: Mer detaljert informasjon fra testen i figur 5.12 Grønt tidsintervall er for 50mm/s, lilla for 100mm/s og oransje for 150mm/s.

I figur 5.13 plott B, vises det i starten av de fargede intervallene fordelen med feedforward PI-regulator. Vi får en døende eksponential i frekvens referansen. Det gjør at når motoren trenger høy så akselerasjon vil den få det. Samt at hastigheten reduseres så reduseres glatt inn mot referansen.

I plott C ser vi at motorhastigheten følger frekvensreferansen. Motorhastigheten endres som funksjon av antall runder som spoles inn på trommelen, som fører til at motorhastigheten blir redusert når lasten blir heistet opp.

I plott D i figur 5.13 ser vi momentet på motoren. Innspolingene er markert i grønt, lilla og oransje intervall. Mellom disse er det endringer i moment, men det kommer fra posisjonsendringen fra der hvor innspolingen stoppet til der hvor den nye innspolingen startet. Vi ser at momentet ved starten av de fargede intervallene er på samme verdi (≈ -200) og ender ved samme moment

når innspolingen er ferdig (≈ -380). Dette er forventet ettersom momentarmen blir den samme når innspolingen starter fra samme sted og ender ved samme sted. Det som er ulikt mellom de tre innspolingene er at momentet øker hurtigere ved raskere innspoling. Dette er også naturlig ettersom raskere innspoling fører til at det kjappere blir flere runder på trommelen og dermed utvides momentarmen raskere og momentets økning per tidsenhet går fortere.

Lastkompensering:

Det ble utført en test av kompenseringsegenskapene til regulatoren. Testen ble utført på samme måte som den for PI-regulatoren vist tidligere. Resultatet er vist i figur 5.14.

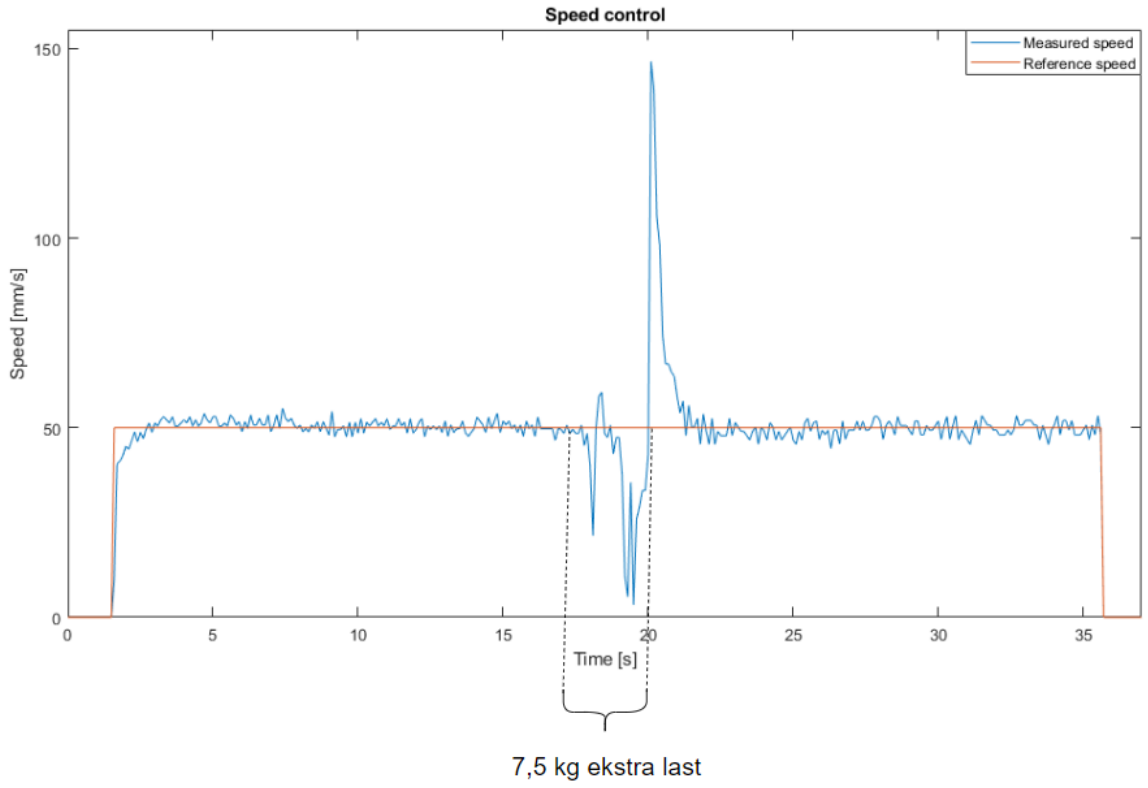


Figure 5.14: 'Feed forward' PI-regulator: 3 typer hastighetsreferanse test. $K_p = 1$ og $K_i = 0.5$. Rød linje er referanse i mm/s og blå linje er målt mm/s

Fra figuren ser vi at det er ekstra last i perioden 17-20 sekunder. Ved 20 sekunder blir lasten fjernet og hastigheten til motoren økes kraftig som følge av dette. Ved 21,5 sekunder har regulatoren justert tilbake til hastighetsreferansen.

Figur 5.15 under viser mer detaljert informasjon fra testen i figur 5.14:

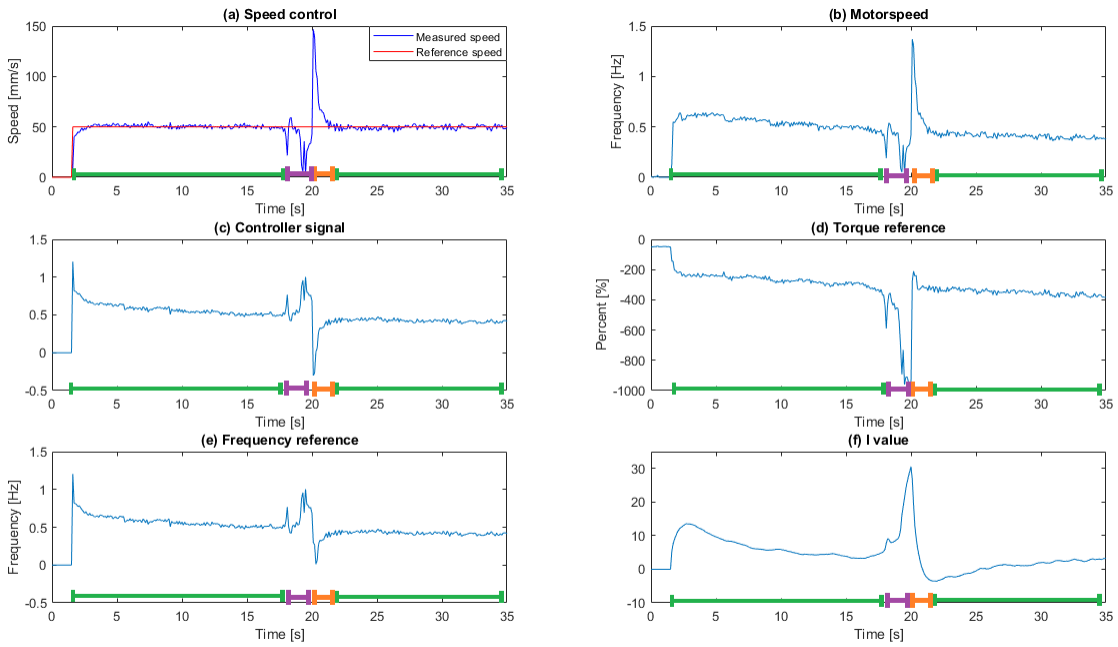


Figure 5.15: Lastkompensasjonsjon med mer informasjon . Grønt tidsintervall er uten ekstra last, lilla er med ekstra last og oransje er etter lasten er fjernet.

Grønt intervall[0-17sek]
Mellom 0-1 sekund starter regulatorsignalet (plott C) brått som funksjon og setter frekvensreferansen (plott E). Motorhastigheten (plott B) følger frekvensreferansen, men ligger litt under pga. sakking. Innspoling startet i bunn når momentet (plott D) er ca. 0. I-leddet (plott F) bygges opp kraftig i starten når det er stort avvik mellom motorhastigheten og referansen.

Mellom 1-17 sekund ser vi at lastenhastigheten (plott A) er 50 mm/s . Regulatorsignalet reduseres som funksjon av antall innspolinger. Frekvensreferansen følger dette signalet og blir redusert for å kontinuerlig kunne produseres 50 mm/s i lasthastighet under hele innspolingen. I-leddet blir gradvis redusert som følge av at avvikene blir mindre og mindre. Absolutt verdien til momentet blir høyere (mer negativt) fordi momentarmen blir større jo mer som spoles inn.

Lilla intervall [17-19sek] → ekstra last lagt til:
Lasthastigheten blir redusert som følge av den ekstra lasten. Regulatorsignalet økes for å justere hastigheten opp til referansen. Frekvensreferansen øker fra regulatorsignalet. Motorhastigheten prøver å øke, men den detter i verdi. I-leddet øker siden avviket øker, som over lengre tid hadde trukket motorhastigheten opp slik at avviket hadde forsvunnet. Ettersom forstyrrelsen er så rask så får vi ikke sett det på dette plottet. Momentet øker ved lastendring. Det når -1000 ved 19 sekund. Husk at både mer negativ eller positiv økning i moment indikerer mer moment. Fortegnet bestemmer bare om momentet virker mot eller med klokken på motorens momentarm.

Orange intervall [20-21.5sek] → ekstra last fjernet:
I første del av det oransje intervallet 20-20.5 sek så ser vi lasthastigheten at farten i mm/s øker som følge av at den ekstra lasten blir tatt av. Motorfarten i Hz øker naturligvis av dette. Regulatorsignalet reduseres for å få farten ned til referansen igjen. Frek. ref. synker fordi den følger regulatorsignalet. I-leddet reduseres siden avviket blir negativt. Momentet reduseres. Det når -300 ved 20.5 sekund.

Grønt intervall[21.5-35sek]
Det som skjer i den andre grønne intervallet er en repetisjon av det som ble nevnt for det første grønne intervallet. Regulatorsignalet, frek. ref, motorhastigheten har blitt redusert som følge av innspolingen og momentet har økt. I-leddet har blitt mer redusert som følge av det kraftige oversvinget i det oransje intervallet og ligger lavere enn ved første grønne intervall.

5.3.5 Oppsummering - valg av beste regulator

Det ble utført tester hvor responsene fra en PI-regulator og en 'feed forward' PI-regulator ble analysert. Responsen til regulatorene ble sammenlignet via tester opp mot fastsatte krav fra delkapittel 5.3. Reguleringsverdiene for regulatorene er vist i tabell 5.5.

Reguleringsparametre 'prøv og feil'-metoden			
Regulator	**Kp**	**Ki**	**Kd**
PI-regulator	2	3	-
'feedforward' PI-regulator	1	0.5	-

Tabell 5.5: Reguleringsparametrene til begge regulatorene ble funnet med 'prøv og feil'-metoden.

Fra PI-regulatorens tester på 3 forskjellige lasthastigheter brukte den 2,5 sekund for å nå 150mm/s referansen, mens 'feed forward' regulatoren brukte 1.1 sekund. Alle lasthastighetene som ble testet var raskest ved bruk av 'feed forward' PI-regulator.

Lastkompensasjonstesten viste at 'feed forward' PI-regulator taklet forstyrrelser bedre enn PI-regulatoren. 'Feed forward' PI-regulatoren klarte å tilfredsstille kravene bedre enn PI-regulatoren.

Videre ble 'Feed forward' PI-regulator med reguleringsparametre fra 'prøv og feil'-metoden målt opp mot responsen med reguleringsparametre funnet fra 'Skogestads'-metode.

5.3.6 'Skogestad'-tuning metode for 'feed forward' PI regulator

I dette delkapittelet finner vi parametrene K_p og T_i ved bruk av 'Skogestads'-metode. Disse reguleringsparametrene blir implementert i en feed forward PI-regulator og testet. Resultatet av disse testene blir målt opp mot resultatene til 'feed forward' PI-regulatoren med reguleringsparametre funnet fra 'prøv og feil'-metoden.

Finner reguleringsparametre

Vi bruker τ og K som ble funnet i delkapittel 4.3.1. Utregningen av K_p og T_i blir gjort etter "Index 2" i tabell 4.1:

$$Kp = \frac{\tau}{K \cdot Tc} = \frac{0,163}{1 \cdot 0,5} = 0,326 \tag{5.6}$$

Verdien for Ti blir den minste av de to verdiene nevnt i likningen under:

$$Ti = min[\tau, K1 \cdot Tc] \tag{5.7}$$

$$Ti = k1 \cdot Tc = 4 \cdot 0,5 = 2s \tag{5.8}$$

$$Ti = \tau = 0.163s \tag{5.9}$$

Ki finner vi fra:

$$Ti = \frac{Kp}{Ki} \rightarrow Ki = \frac{Kp}{Ti} = \frac{0,326}{0,163} = 2 \tag{5.10}$$

5.3.7 Tester 'Skogestads' metode for 'feed forward' PI-regulator

Alle testene under for 'feed forward' PI-regulatoren ble utført med verdiene $K_p = 0.326$ og $K_i = 2$ funnet fra 'Skogestads'-metoden.

Enkelt sprang:

Responsen til hastighetsreguleringen med parameterene fra 'Skogestads'-metode er vist i figur 5.16.

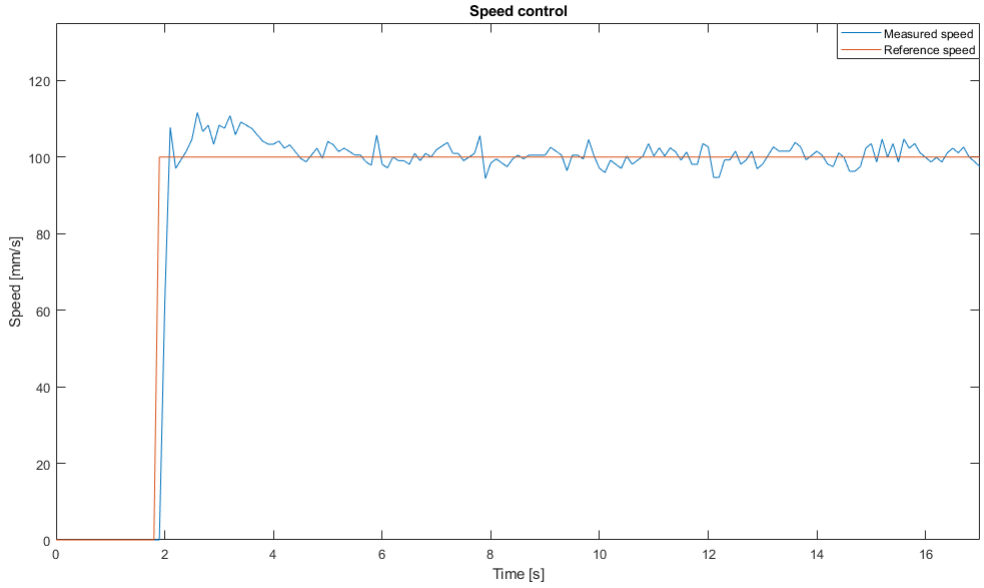


Figure 5.16: 'Feed forward' PI-regulator: 'Skogestads'-metode. Enkelt sprang. $K_p = 0.325$, $K_i = 2$. Rød linje er referanse i mm/s, blå linje er målt hastighet.

Fra figur5.16 ser vi at responsen er rask, men at det er mye over- og undersving rundt referansen.

Tre forskjellige last hastigheter:

Det ble utført en test av flere sprang. Testen er lik som den for PI-regulatorens vist tidligere. Resultatet av testen er vist i figur 5.17.

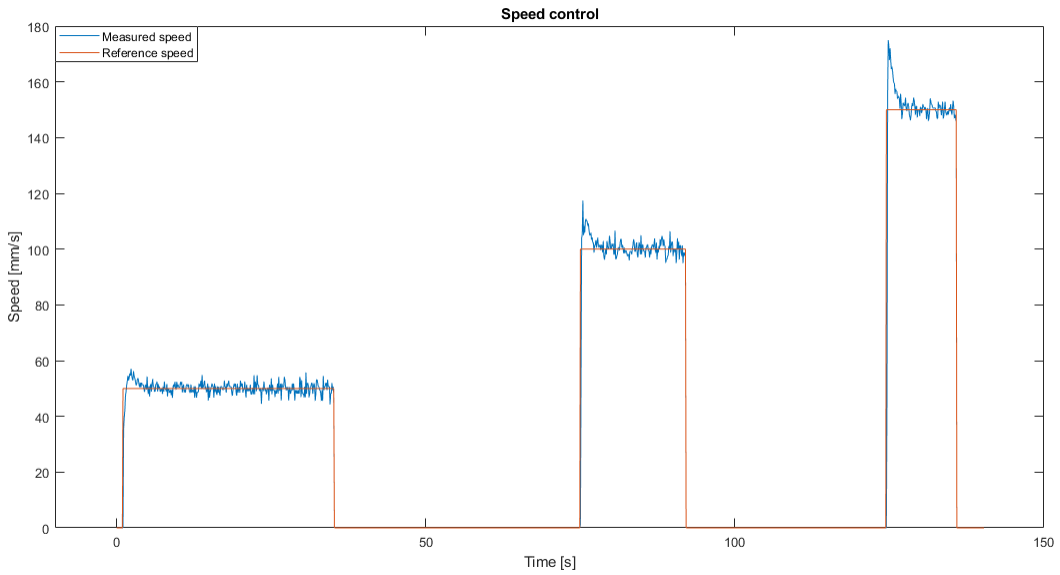


Figure 5.17: Feedforward PI-regulator: Skogestads metode. Tre sprang. $K_p = 0.325$, $K_i = 2$. Rød linje er hastighetsreferanse, blå linje er målt hastighet.

Fra figur 5.17 ser vi at responsen er nokså lik for alle hastighetene. Det er derimot et markant større oversving som følge av økende referanse lasthastigheter.

Lastkompensering:

Det ble utført en test av forstyrrelses kompenseringsegenskapene til regulatoren. Testen er lik som den for PI-regulatoren vist tidligere. Resultatet av testen er vist i figur 5.14.

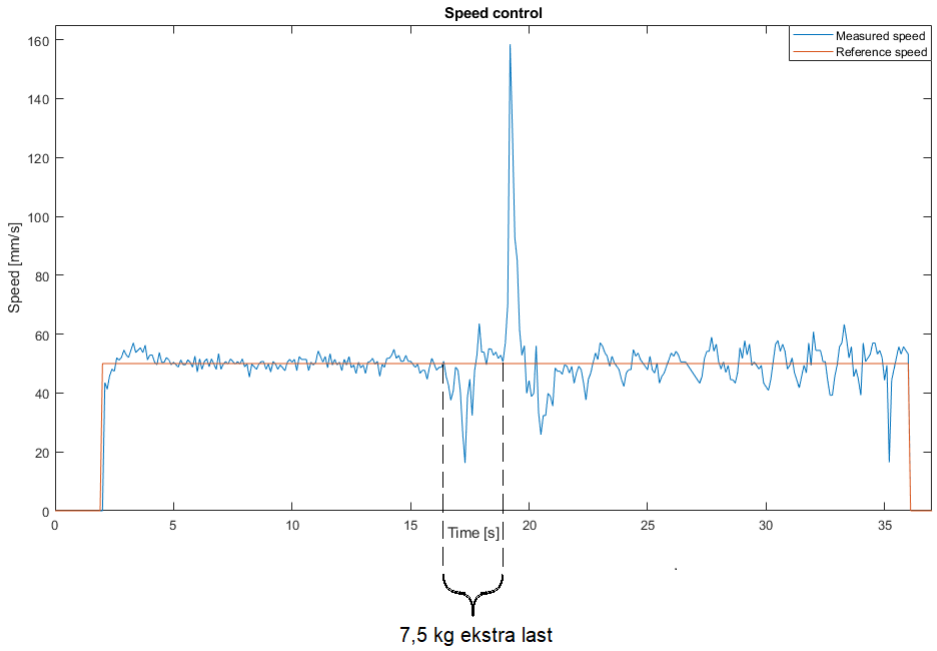


Figure 5.18: Feed forward PI-regulator: Skogestads metode. Lastkompensering. kp = 0.325, Ki = 2. Rød linje er referanse i mm/s, blå linje er målt mm/s.

Fra figur 5.18 så ser vi at regulatoren håndterer forstyrrelsen, men det medfører en del svingninger rundt referansen etter lasten er fjernet.

Oppsummering - valg av beste metode for den beste regulatoren

Det ble utført tester av feed forward PI-regulator med reguleringsparametre funnet fra 'prøv og feil'-metoden og fra 'Skogestads'-metoden. Responsen ble sammenlignet opp mot fastsatte krav fra delkapittel 5.3.

De følgende reguleringsverdiene fra sin respektive metoder er vist i tabellen 5.6:

Sammenligning			
Metoder og regulator	**Kp**	**Ki**	**Kd**
'Prøv og feil' 'feed forward' PI regulator	1	0.5	-
'Skogestad' 'feed forward' PI regulator	0,326	2	-

Tabell 5.6:

Responsen på et sprang (fra figur 5.16) med reguleringsparametre funnet fra skogestad er like raskt som ved 'prøv og feil'. Det første oversvinget er derimot større og varer lenger. De etterfølgende over- og undersvingene rundt referansen er litt større enn ved parametrene funnet fra 'prøv og feil'.

Responsen på større sprang (fra figur 5.17) med reguleringsparametre funnet fra skogestad viser at det første oversvinget bler mer markant større jo større lasthastighet som settes som referanse. Ellers er responsen lik som nevnt i avsnitt over. 'Prøv og feil' presterte totalt sett best ved denne testen.

Lastkompensasjonstesten viste at evnen til begge regulatoren å takle forstyrrelser er nokså lik, men etter som reguleringsparametre funnet fra 'prøv og feil'-metoden taklet perioden etter at den ekstra lasten var tatt av bedre i form av at den målte lasthastigheten var mye mer stabil enn ved parametrene funnet ved 'Skogestads'-metode. 'Prøv og feil'-metoden presterte best på denne testen.

Reguleringsparametre funnet med 'prøv og feil'-metoden tilfredsstiller kravene om raskets respons, lite oversving, best forstyrrelses kompensering og minst støyete oppførsel rundt referansen etter denne var nådd best.

Kapittel 6

Momentregulering

I dette kapittelet presenteres løsning og resultater for momentregulering av vinsjsystemet.

Det ble implementert momentregulering av vinsjsystemet. Momentregulatoren vil regulere pådraget til motoren for å oppnå ønsket momentreferanse. Når det oppdages slakk i vaier ved lavt moment vil motoren spole inn vaier inntil momentet ikke lenger er lavere enn satt referanseverdi. Skulle det derimot bli for mye spenn på vaieren vil motoren spole ut vaier inntil momentet ikke lenger er høyere enn satt momentreferanse. Reguleringen illustreres i figur 6.1.

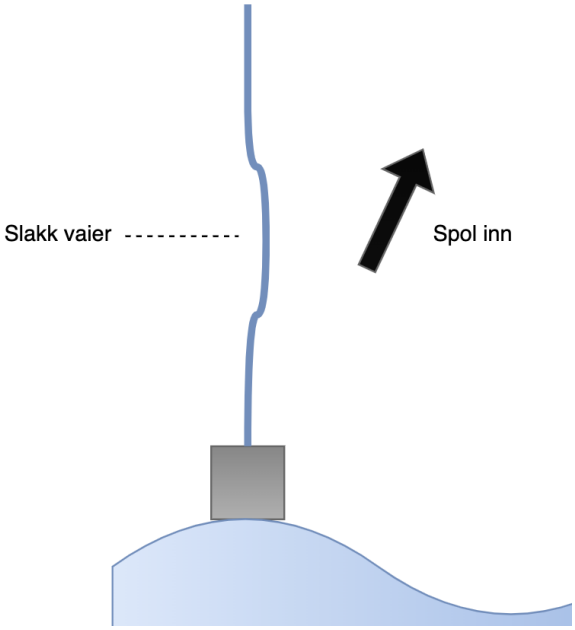
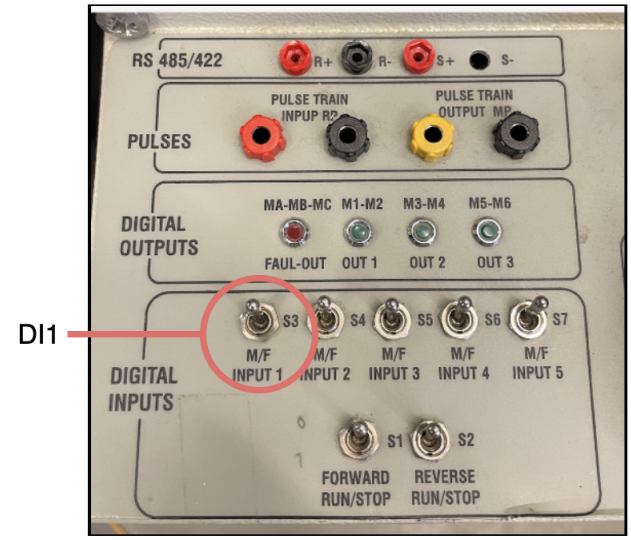


Figure 6.1: Figuren viser at ved slakk vaier vil momentregulatoren gi pådrag til motoren som spoler inn slakken. På denne måten kan bølgebevegelser følges.

For å kunne aktivere momentkontroll i frekvensomformeren må variabelen `d5-01; Torque Ctrl Selection` settes til 1. Denne står normalt på 0; `speed control`. Under prosjektet møtte vi på et problem som gjorde at det ikke var mulig å sette denne variabelen gjennom koding, feilen forklares ytterligere i kap. 7.1. Løsningen ble at bryterpanelet bakpå frekvensomformeren måtte kobles til. Gjennom parameterinnstillinger i frekvensomformeren ble det valgt at bryteren (DI1) skulle bytte mellom hastighetskontroll og momentkontroll.

I tillegg måtte variabelen `F6-06; Trq Ref Comms` settes lik 1. Denne variabelen gjør det mulig å skrive til momentreferansen som videre kan brukes i koden for å sette en referanseverdi for moment. I figur 6.2 vises bryterpanelet og setting av parameterne i programvaren Q2 edit.



H1-01	DI1 Function Selection	0013	Spd/Trq Switch
F6-06	Trq Ref/Lim Comms	1	Enabled

Figure 6.2: Bryterpanel og variabler som endrer mellom hastighetskontroll og momentkontroll. Bryteren DI1 som er utringet i figuren, ble brukt til å bytte mellom hastighetskontroll og momentkontroll i frekvensomformeren.

Når det byttes mellom hastighetskontroll og momentkontroll vil frekvensomformeren bytte fra å bruke en frekvensreferanse og en momentbegrensning (som i figur 2.6) til at den istedenfor benytter en momentreferanse og en frekvensbegrensning som vist i figur 6.3.

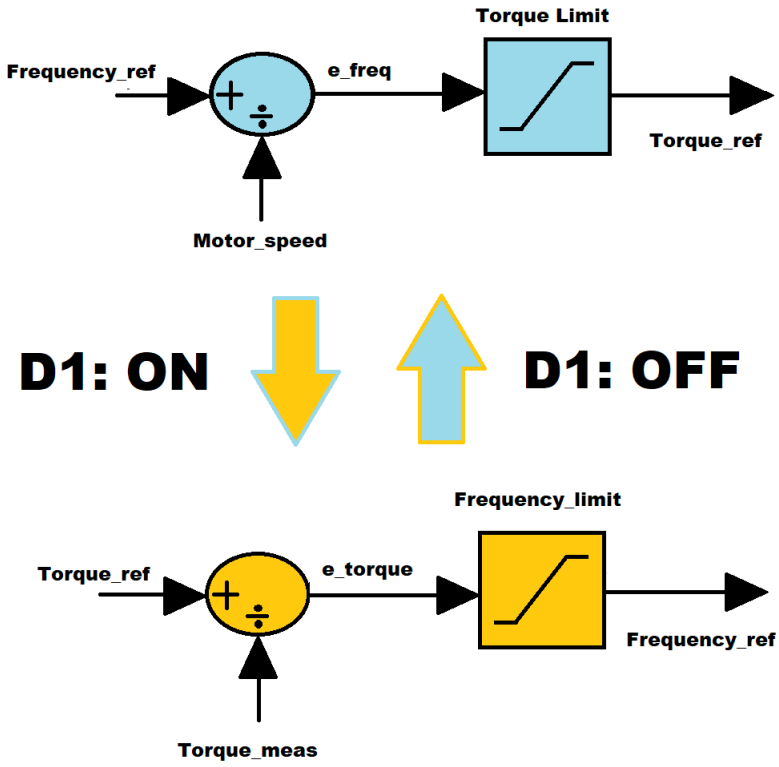


Figure 6.3: Forenklet bilde av bytting av frekvensomformerens operativ metode. Aktiveres bryteren DI1 går systemet fra å styres med en frekvensreferanse (blått) til å styres av en momentreferanse (oransje).

I den oransje delen i figur 6.3 blir det målt et moment (`torque_meas`). Denne momentverdien er ikke tilgjengelig fra frekvensomformeren ettersom det er en firma hemmelighet hos Omron. Dette medfører av vi ikke fikk lagd en klassisk avviksbasert regulator for moment . Det er kun mulig å sette en momentreferanse (`torque_ref`) så ordnes reguleringen fra den interne sløyfen i frekvens omformeren.

6.1 Manuell kjøring

Det ble implementert et program der det er mulig å kjøre vinsjsystemet manuelt. Området det kan kjøres mellom er begrenset av det kalibrerte området.

For å kunne kjøre vinsjsystemet manuelt og teste momentregulatoren ble det utviklet en HMI skjerm som vist i figur 6.4. På denne skjermen kan brukeren kjøre vinsjsystemet opp og ned i det kalibrerte høydeområdet, i tillegg til å aktivere momentkontroll funksjonen `Lower to sea`.

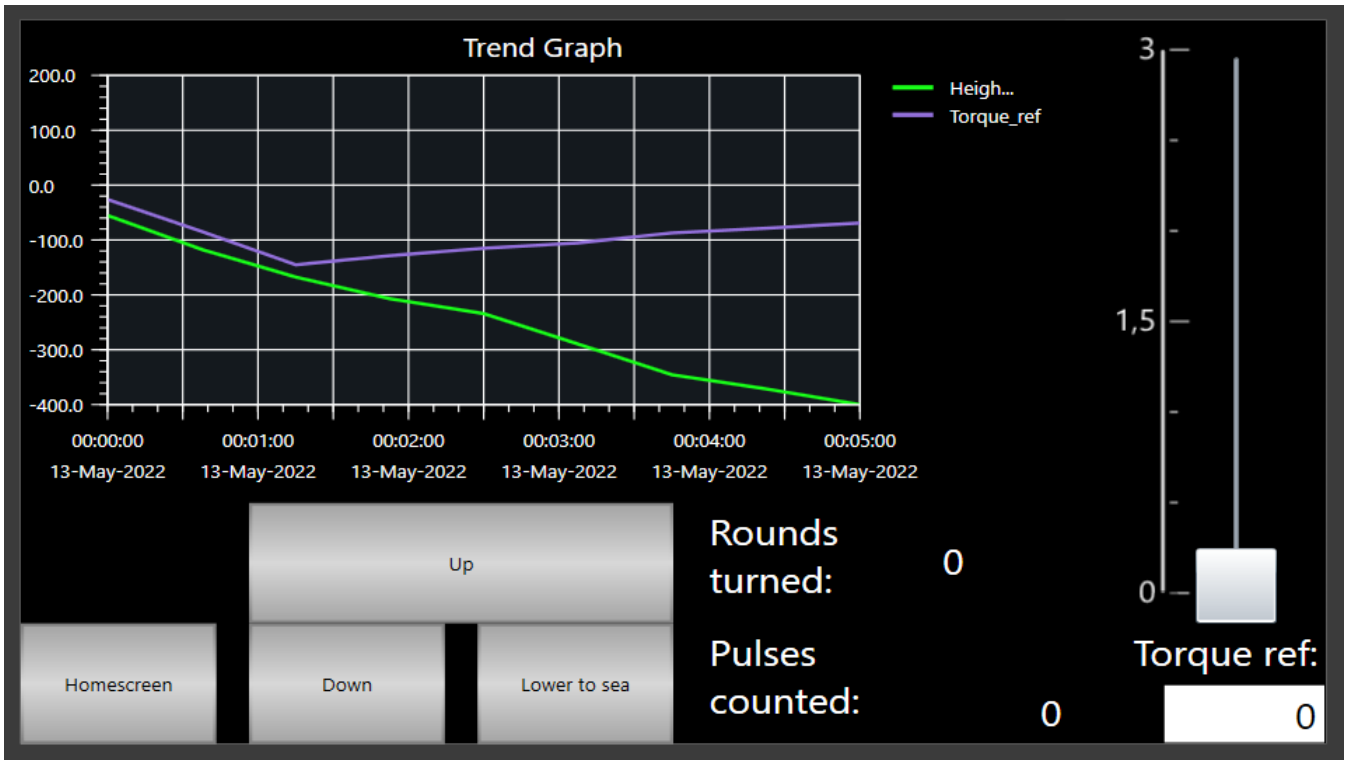


Figure 6.4: I figuren vises HMI skjerm for manuell kjøring. Brukeren kan på denne skjermen kjøre vinsjsystemet manuelt gjennom knappene `Up` og `Down`. `Homescreen` knappen vil føre brukeren tilbake til hjemskjermen. Momentreferansen kan settes gjennom numerisk display. Scrollbaren gir mulighet for styring av hastigheten (0-3 Hz) under manuell kjøring. I grafen vises høyden i mm sammen med momentreferansen. Verdiene i grafen er ikke reelle; bildet tatt ved simulering i Sysmac studio.

6.1.1 Momentkontroll funksjon

Momentkontroll funksjonen `Lower to sea` kjører vinsjsystemet ned til sjønivået for å deretter følge de bølgebevegelsene som måtte oppstå. Dette er mulig ved å bruke den interne momentregulatoren i frekvensomformeren, som ved avvik mellom momentreferansen og målt moment vil beregne pådrag til motoren for å foreta inn- og utspoling av vaier. Det vil merkes lavt moment som vil indikere at det er slakk i vaier når lasten treffer bunnivået. Skulle lasten løftes ved f.eks en bølge vil det settes pådrag på motoren for å spole inn slakken i vaieren.

Pseudokode under viser hvordan momentkontroll funksjonen er bygget opp.

Kodeutdrag 6.1: Pseudokode lower to sea

```
// 1. Kjører ned med bestemt hastighet
// 2. Stopper ved lavt moment (slakk vaier; truffet bunn)
// 3. Aktiver momentkontroll via bryter DI1
// 4. Spoler inn ved lavt moment (slakk vaier; bølge)
// 5. Spoler ut igjen når moment er for høyere en referansen.
```

I programmet utføres dette ved å sette et minne (`Sea_memory`) når lasten treffer bunnivået. Minnet blir satt til `TRUE` når det merkes lavt moment, noe som indikerer at lasten har nådd sjønivået.

I kodeutdrag 6.2 settes motor orientering og pådrag som fører til at lasten blir senket ned.

Kodeutdrag 6.2: Utdrag fra kode; Manuell kjøring

```
39 IF Sea_memory = FALSE THEN
40 // Motor orientation and frequency reference
41    frequency_ref := 1;
42    q2a_down := TRUE;
```

Når det merkes et moment som er større eller lik 0 i linje 66 vil minnet settes til `TRUE`. Lasten er nå i bunn (på sjø). Lavere moment som virker på motorens momentarm vil gi mer positiv momentreferanse (i %).

Kodeutdrag 6.3: Utdrag fra kode; Manuell kjøring

```
65 // Sets a memory when at sea level
66 IF torque_ref >= 0 THEN
67    Sea_memory := TRUE;
68 END_IF;
```

Videre aktiveres momentkontroll ved å aktivere bryteren DI1 på bryterpanelet til frekvensomformeren. Når momentkontroll er aktivert vil frekvensreferansen bli hastighetsgrensen. I kodeutdraget 6.4 settes frekvensreferansen (`frequency_ref`) til 15, som blir hastighetsgrensen til systemet.

På HMI skjermen kan brukeren sette ønsket momentreferanse, denne verdien skrives videre til `torq_ref_wright` på linje 79. Dette momentet er referanseverdien inn til den interne sløyfen i frekvensomformeren. Settes momentreferansen til en negativ verdi vil motoren produsere moment i motsatt retning av klokken når man ser inn på akslingen (altså oppover), og det motsatte om man setter en positiv verdi. Ved endring av det målte momentet, vil momentregulatoren i frekvensomformeren beregne et pådrag til motoren for hurtig inn- og utspoling av vaier.

Kodeutdrag 6.4: Utdrag fra kode; Manuell kjøring

```
75 IF Sea_memory = TRUE THEN
76 // Frequncy referance becomes the speed limit when DI1 is ON
77     frequency_ref := 15;
78 // Writes to the torque reference in Q2A
79    torq_ref_wright := HMI_userinput_torque_ref;
80 END_IF;
```

6.2 Resultater momentregulering

I dette delkapittelet vil resultatet ved momentregulering av systemet presenteres.

Det ble gjort et forsøk hvor vi aktiverte momentkontroll og laget bølgebevegelser for å observere oppførselen til momentregulatoren. Bølgebevegelsene ble laget manuelt ved å løfte loddet rolig opp og ned. Figur 6.5 viser et oversiktsbildet over hendelsesforløpet når momentkontroll aktiveres via momentkontroll funksjonen.

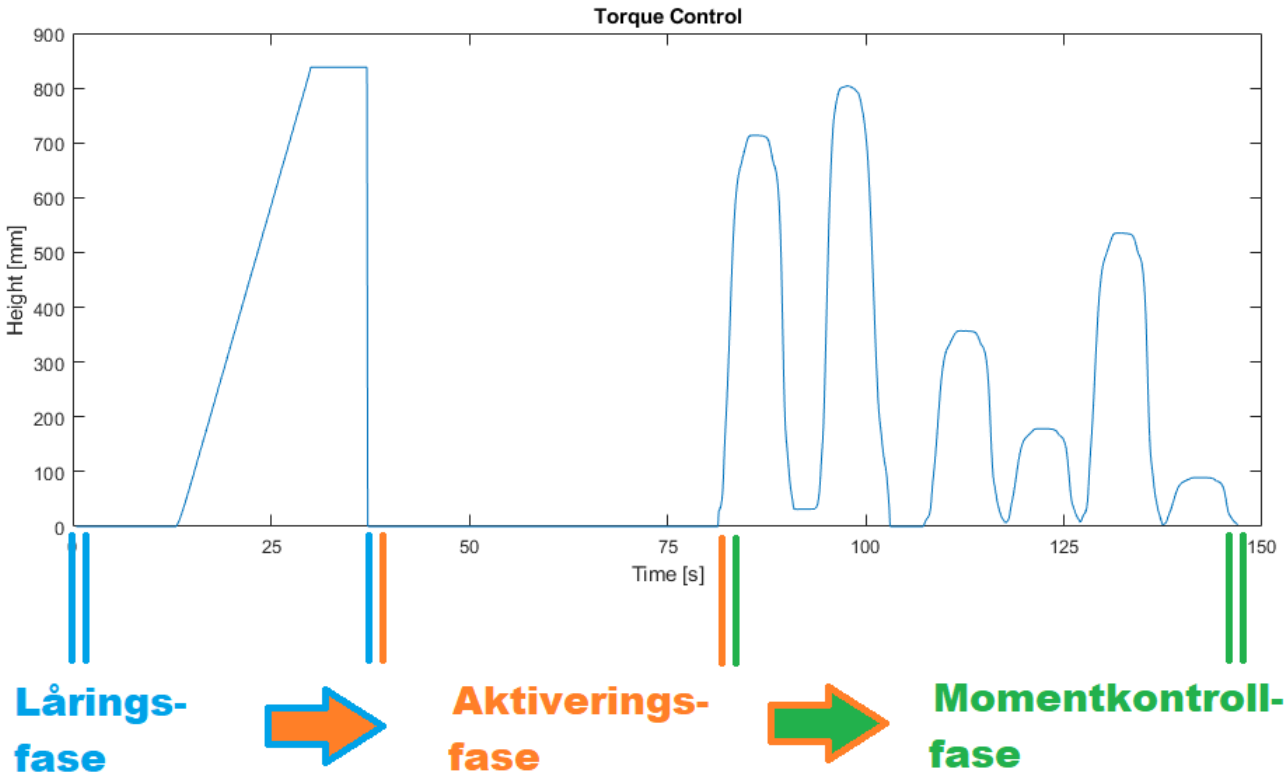


Figure 6.5: Figuren viser hendelsesforløpet etter aktivering av momentkontroll funksjonen. I blått vises låringfasen. I oransje er aktiveringsfasen. I grønt er momentkontroll fasen. Etter låring byttes målt høyde på sjøen fra å være 830mm til å være 0mm. Drift i posisjon som følge av bølger vil dermed være fra 0mm og ikke 830mm.

I blått intervall mellom 12 og 27 sekund låres lasten fra topposisjonen og ned til sjønivået (=830mm). Deretter ved 30s settes ny referanse for høyden til å være 0mm. I det oransje intervallet er aktiveringsfasen i gang, her vil bryteren DI1 aktiveres og frekvensomformeren bytter fra hastighetskontroll til momentkontroll. I grønt intervall ved 80 sek lages det bølger, momentregulatoren vil beregne pådrag til motoren for å foreta hurtig inn- og utspoling av vaier.

Figur 6.6 viser mer detaljert rundt motorens orientering som følge av bølgenes dynamikk:

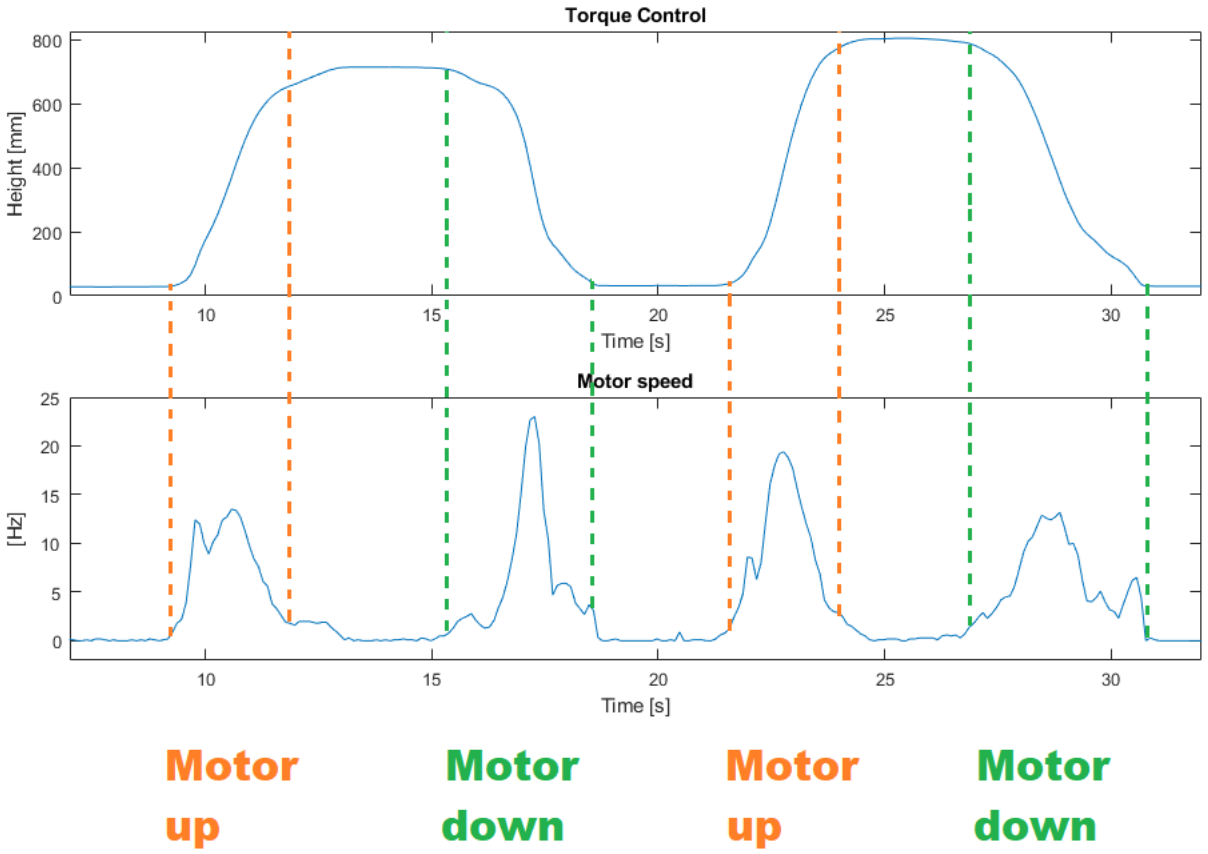


Figure 6.6: Momentregulatoren beregner pådrag til motoren som vil foreta hurtig inn- og ut spoling av vaier. De fargede intervallene indikerer når bølgen beveger seg raskest og momentregulatoren kompenserer for dette med å sette høye pådrag på motoren.

I det oransje intervallet i figur 6.6 merkes det slakk vaier som følge av en bølge. Motoren vil da få pådrag for å heise lasten. I grønt intervall merkes det at det blir spenn i vaier som følge av at bølgen går nedover og motoren får pådrag for å senke lasten.

Kapittel 7

Diskusjon

I dette kapittelet vil ulike utfordringer som oppsto under prosjektet diskuteres, og eventuelle løsninger til disse utfordringene. Det vil også beskrives forslag til forbedringer og videreutvikling av systemet.

7.1 Feil og mangler

Under programmering i Sysmac studio ble det oppdaget en feil / mangel som gjorde at programmering av reguleringene ble vanskeligere. Ved henting av variabler fra frekvensomformeren som vist i kap. 2.1, ble det gitt feilkode om mangel på kommunikasjon mellom frekvensomformeren og PLS. Dette førte til at programmet ikke kunne kjøres. Det er ukjent hva som skaper feilen, men mest sannsynlig er det er maskinvarefeil ettersom dette problemet ikke er kjent for Omron Norge. Feilen gjør at det ikke er mulig å skrive til variabler i frekvensomformeren, noe som hadde vært nyttig i kode for blant annet momentregulering. Det vil si at i stedet for å kunne endre mellom hastighetskontroll og momentkontroll via variabelen *D5-01; Spd/Trq selection* i koden måtte dette gjøres manuelt i frekvensomformerens display eller i programvaren Q2 edit.

Feilen er pr. 15.05.2022 ukjent og det er opprettet sak på feilen hos Omron tech support.

7.2 Forslag til forbedringer

Programmet for kalibreringsprosessen beskrevet i kapittel 3 vil være mulig å utvide til å ta hensyn til rampetiden ved opp- og ned ramping. Per idag vil det oppstå et lite avvik mellom den målte høyden fra programmet og den reelle målt av en tommestokk, men ved å ta hensyn til rampetiden i programmet vil man kunne oppnå en fin nedramping samtidig som man får en mer nøyaktig måling.

I programmet for posisjonsregulering beskrevet i kapittel 4 vil det være mulig å implementere en momentkontroll funksjon. Momentkontrollen vil kunne merke om det blir slakk i vaieren og overstyre posisjonsregulatoren og sette pådrag ut på motoren slik at det spoles inn helt til vaieren blir stram igjen.

Momentkontroll funksjonen for posisjonsregulering kan fungere på følgende måte:

Kodeutdrag 7.1: Pseudokode momentkontroll for posisjonsregulering

```
// 1. Hvis det merkes lavt moment vil momentregulator overstyre posisjonsregulator.
// 2. Det settes pådrag på motoren for åheise opp.
// 3. Når det er akseptabelt moment vil momentkontroll skrus av og
    posisjonsregulatoren styrer.
// 4. Det er da et avvik i posisjon.
// 5. Loddet vil da returneres til den ønskede posisjonen via
    posisjonsreguleringsprosessen.
```

I programmet for hastighetsregulering beskrevet i kapittel 5 vil det være mulig å implementere en momentkontroll funksjon. Om lasten blir låret ned til sjøen og det er bølger tilstedet vil det kunne oppstå slakk vaier før man har fått startet innspolingen via hastighetsregulatoren. Funksjonen kan merke om det er slakk i vaieren under denne fasen og ligge å holde kontroll på momentet på vaieren helt til man har fått startet innspolingen via hastighetsregulatoren.

Momentkontroll funksjonen for hastighetsregulering kan fungere på følgende måte:

Kodeutdrag 7.2: Pseudokode momentkontroll for hastighetsregulering

```
// 1. Hvis det merkes lavt moment vil momentregulator overstyre hastighetsregulatoren.
// 2. Det settes pådrag på motoren for åspole opp og spole ut for åfølge bølgene.
// 3. Ved aktivert hastighetsregulator vil momentkontroll skrus av og
    hastighetsregulatoren styrer.
// 4. Innspoling fortsetter med den ønskede hastigheten.
```

Programmet for momentregulering beskrevet i kapittel 6 vil være mulig å utvide til å bytte mellom hastighetskontroll og momentkontroll. Når feilen i kap. 7.1 løses, kan parameterene i frekvensomformeren som bytter mellom disse to metodene endres gjennom koden. Det vil gi muligheter til implementere momentkontroll i alle kodene, i stedet for å bruke en fysisk bryter.

7.3 Videreutvikling

For kalibreringsprosessen beskrevet i kap. 3 kan det videre være mulig å utvide programmet til å ta hensyn til endring av sjøhøyden. Prosessen fungerer pr.idag på den måten at det kalibreres et området, men om sjønivået synker etter kalibrering vil sjønivået være utenfor det kalibrerte området. Det kan muligens lages en logikk for måling av havnivå og justering av det kalibrert området etter denne dataen. Det er bare posisjonsreguleringen som er påvirke av dette. Låringsfunksjonen vil ikke bli negativt påvirket av dette ettersom den senker til sjø uavhengig av kalibrert området.

Kalibreringsprosessen kan også videreutvikles ved bruk av en sensor som merker når loddet er i topposisjonen, som videre kan brukes i programmet for å stoppe pådraget til motoren. Slik prosessen fungerer brukes det moment i topp ved kalibrering, men momentet er ugunstig å bruke ettersom det er mulig å få programmet til å tro det er i topposisjonen ved å legge trykk på loddet.

Vinsjmodellen er bygget med tanke på at ting kan videreutvikles eller endres. Det er mulig å oppnå en bedre simulasjon av bølgebevegelser. Dette kan skje ved å koble en motor til den andre enden av tauet. Videre kan det programmeres en funksjon til denne motoren som spoler inn og ut tauet på en slik måte at det ligner bølgebevegelser, eksempelvis en sinusbevegelse.

Kapittel 8

Konklusjon

I vårt arbeid med bacheloroppgaven har vi lært hvordan man programmerer i Sysmac Studio og hvordan man kan skape programmer for regulering av systemer. Resultatet av denne bacheloroppgaven er at vi har kalibrert og regulert et vinsjsystem via en frekvensomformer.

Posisjonsregulering av vinsjsystemet ble oppnådd med god nøyaktighet. Det ble funnet reguleringsparametere fra 'Skogestads'- og 'prøv og feil'-metoden. Responsen med disse parametrene ble prøvd i forskjellige tester, hvor konklusjonen er at $K_p = 2$ og $K_i = 0.01$ for en PI-regulator førte til mest tilfredsstillende oppførsel basert på kravene om null oversving, rask respons og god forstyrrelseskompensering.

Hastighetsregulering av vinsjsystemet ble oppnådd med god nøyaktighet. Reguleringsparametre for en vanlig PI-regulator og en feed forward PI-regulator ble funnet med 'prøv og feil'-metoden. Disse ble sammenlignet via tester og den beste regulatoren ble valgt. 'Feed forward' PI-regulatoren ble valgt og det ble funnet reguleringsparametre til regulatoren basert på 'skogestads'-metode. Responsen fra 'skogestads'-metode på testene ble analysert opp mot 'prøv og feil'-metoden. Konklusjonen er at 'feed forward' PI-regulatoren med $K_p = 1$ og $K_i = 0.5$ førte til mest tilfredsstillende oppførsel basert på kravene om lite oversving, rask respons og god forstyrrelseskompensering.

Momentregulering av vinsjsystemet ble oppnådd, men med forbedringspotensiale. Momentkontroll funksjonen fungerer som forventet og det ble funnet ut at en momentreferanseverdi på -150 ga god regulering av systemet. Det ble foretatt hurtig inn- og utspoling av vaier for å kompensere for bølgebevegelser. På grunn av feil var det ikke mulig å få implementert momentkontroll i posisjon- og hastighetsregulering.

For å konkludere har vi gjennom denne bacheloroppgaven skapt reguleringsprosesser som fungerer med gode resultater. Vi er fornøyd med arbeidet.

Bibliografi

[1] ABB. *Teknisk veiledning nr. 4 - Veiledning for frekvensomformere for hastighetsstyring*. Hentet 15. Februar 2022 fra https://library.e.abb.com/public/06961a5060b74233c125795b002b9029/NO_Technical_guide_No.4_REVC.pdf?fbcli

[2] BEVI. *4AK 712-4 electric motor*. Hentet 03. Mars 2022 fra https://www.bevi.com/electric-motors/three-phase-standard-motors/electric-motor-4ak-712-4-112130

[3] Collins, D. (2016, 30. Juli). *When are closed-loop and open-loop vector control useful?*. Hentet 21 Mars 2022 fra https://www.motioncontroltips.com/faq-when-are-closed-loop-and-open-loop-vector-control-useful/

[4] Gundersen, A.I. & Mahesan, N.(2021). *Posisjonsregulering og hastighetsregulering av en vinsj*. Hentet 05. Januar fra https://uis.brage.unit.no/uis-xmlui/handle/11250/2774435

[5] Omron Industrial Automation. *E6C2-C Incremental Rotary Encoder Datasheet*. Hentet 03. Mars 2022 fra https://assets.omron.eu/downloads/datasheet/en/v5/q109_e6c2-c_incremental_rotary_encoder_datasheet_en.pdf

[6] Omron Industrial Automation. *Q2A Datasheet*. Hentet 17. Februar fra https://assets.omron.eu/downloads/datasheet/en/v29/i175e_q2a_datasheet_en.pdf

[7] Omron Industrial Automation. *Q2A Technical Manual*. Hentet 25. Januar 2022 fra https://assets.omron.eu/downloads/manual/en/v7/q2a_technical_manual_en.pdf

[8] Rosvold, K.A. (2020, 21. november). *Asynkronmotor*. Hentet 09. Mars fra https://snl.no/asynkronmotor

[9] Haugen, F.(2003). *Praktisk reguleringsteknikk* (2.utg.).

Vedlegg A

Kommunikasjon

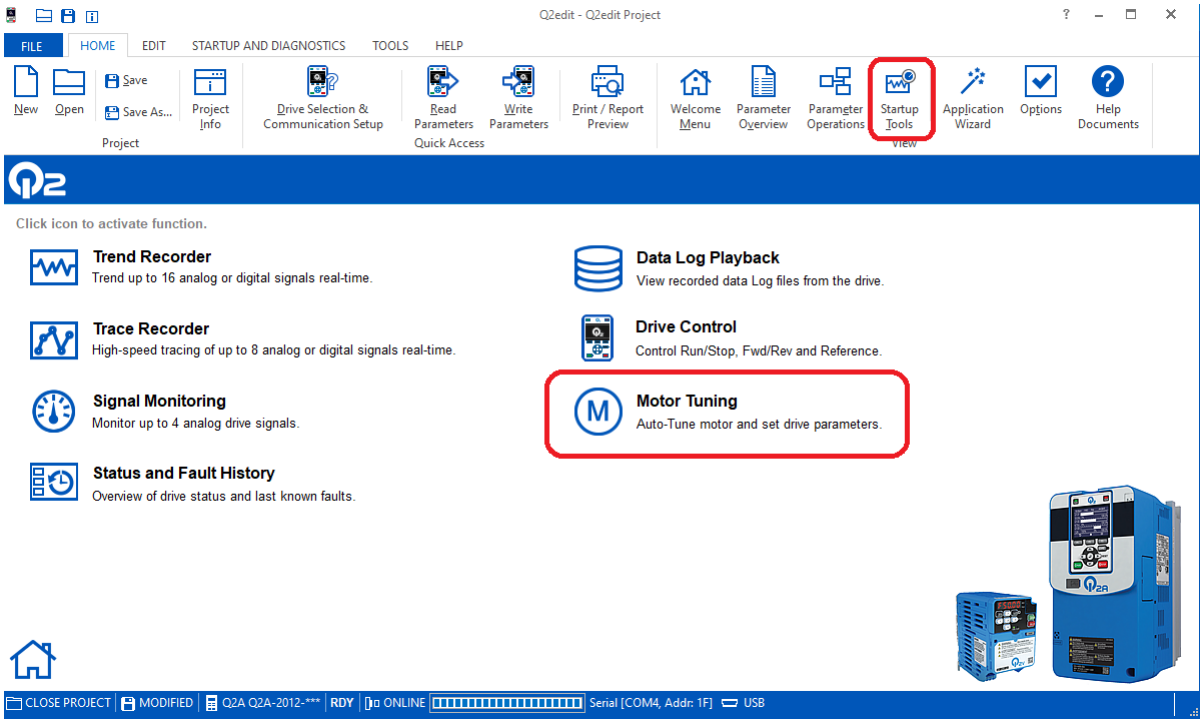
A.1 ESI-filer

Plasser nødvendige Esi filer i form av XML dokumenter på riktig sted: OS(C:) → Programmfiler → OMRON → Sysmac studio → IODevice Profiles → EsiFiles → UserEsiFiles

Vedlegg B

Autotuning

1) Startup Tools → Motor Tuning



2) Velg Mtr Param Tuning og Rotary Auto Tune fra menyen
3) Annerkjenner advarsel og trykk deretter Next
4) Fyll inn motordata → Next
5) Trykk RUN for å kjøre autotuning.