Universitetet
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag / Automatisering og elektronikkdesign	Vårsemesteret 2023 Åpen
Forfatter(e): Kasper Christensen, Søren Frigstad	
Fagansvarlig: Tormod Drengstig Veileder(e): Tormod Drengstig	
Tittel på bacheloroppgaven:Momentregulering ved bruk av Q2A frekvensomformer Engelsk tittel: Torque control using Q2A frequency converter	
Studiepoeng: 20	
Emneord: vinsj, kalibrering, frekvensomformer posisjons-, hastighets-, momentregulering	Sidetall: 108 +vedlegg/annet: 6 Stavanger 15. mai 2023

Innhold

Sammendrag

Denne bacheloroppgaven utforsker ulike metoder for å regulere et vinsjsystem. Først gir den en detaljert gjennomgang av systemets struktur, inkludert de sentrale komponentene og de anvendte programvarene. Videre ble det implementert koder for kalibrering og regulering av systemet ved hjelp av strukturert tekst og ladderkode, alt i Sysmac Studio-programvaren.

Vårt vinsjsystem er konstruert med en frekvensomformer, en programmerbar logisk styring (PLS), og en motor utstyrt med en enkoder. Motorens aksling har en vaiertrommel festet til seg, som holder en hengende last. For å sikre at brukeren får tilgang til relevant og verdifull informasjon, ble det utviklet brukergrensesnittet med HMI-skjermer.

For å bestemme høydeområdet som vinsjsystemet kan operere innenfor, ble det utviklet en kalibreringsprosess. Når aktivert, vil denne prosessen beregne et høydeområde, som deretter fungerer som en begrensning for hvor mye vaier/tau som kan spoles inn eller ut i programmene som regulerer systemet.

Bacheloroppgaven gjennomgår deretter de ulike reguleringene av vinsjsystemet i en trinnvis prosess. Den presenterer teori, løsninger og resultater for posisjons-, hastighets- og momentregulering. En 'prøv og feil'-metode og Skogestads metode ble brukt for å finne reguleringsparametrene til posisjons- og hastighetsregulatoren.

I denne oppgaven ble det utviklet regulatorer for både hastighet og posisjon. Posisjonsregulatoren tillater brukeren å spesifisere en ønsket høyde innenfor det kalibrerte området, og justerer deretter lastens posisjon i henhold til dette. På den andre siden lar hastighetsregulatoren brukeren sette en ønsket hastighet, og justerer deretter motorkraften for å heise eller senke lasten med den ønskede

hastigheten.

For å forhindre slakk og rykk i vaier under løft av bølgebevegelser, ble momentregulering implementert. Brukeren definerer et ønsket moment på motoren, og regulatoren justerer deretter motorkraften ved eventuelle avvik fra denne verdien. Resultatet er en rask inn- og utspoling av tauet ved løft/senking av last, noe som sikrer konstant spenning i vaieren.

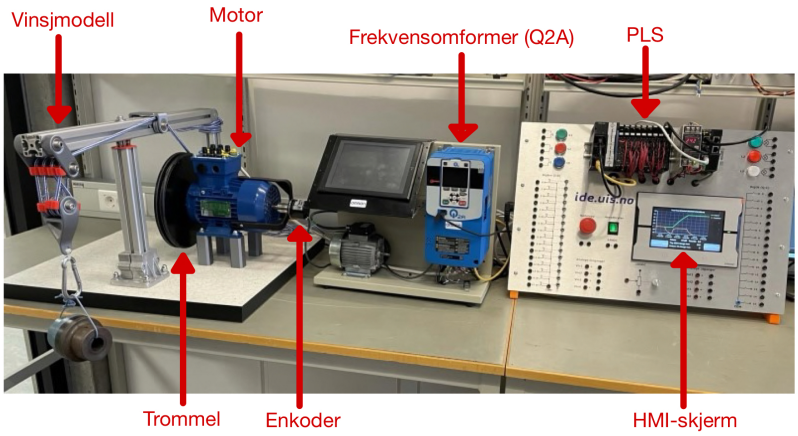
I oppgavens avsluttende del, foretas det en diskusjon rundt systemets "feil og mangler" og potensielle veier for videreutvikling av vinsjsystemet. Deretter rundes oppgaven av med en konklusjon.

Kapittel 1

Innledning

Bacheloroppgaven baserer seg på demomodellen fra Omron vist i figur 1.1. Modellen består av en PLS, en HMI-skjerm og en frekvensomformer som er koblet til en asynkron AC-motor med enkoder. På akslingen til motoren er det montert en vaiertrommel med tau som fungerer som et vinsjsystem med hengende last.

Oppgaven gikk ut på å utvikle koder som bidrar med å styre de innebygde reguleringsfunksjonene i frekvensomformeren (posisjons-, hastighets- og momentregulering) ved hjelp av PLSen. Under i figur 1.1 vises demomodellen.



Figur 1.1: Figuren viser demomodellen. Fra venstre i figuren: Vinsjmodell, motor med enkoder, frekvensomformer og PLS med HMI skjerm.

1.1 Problemstilling

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 vaier i store bølgehøyder. For å regulere kreftene i vaieren når livbåten treffer vannflaten samtidig med at det er bevegelser i skipet, er en mulighet å anvende momentregulering.

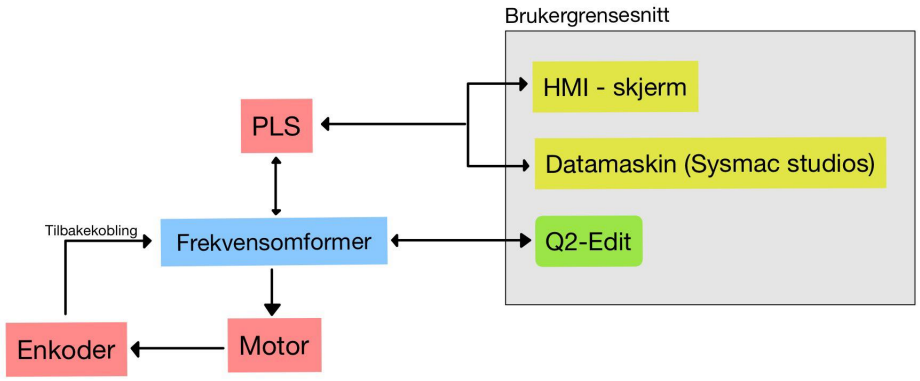
Basert på oppgaveteksten ble følgende punkter/mål utarbeidet:

- Sette seg inn i problemstillingen og forstå ønsket funksjonalitet
- Forbedre den fysiske konstruksjonen av demomodellen ved å lage et trinsesystem som kan simulere bølgebevegelser
- Lage til et fysisk underlag som kan simulere havoverflaten
- Sette seg inn i frekvensomformeren og medfølgende programvare
- Programmere og teste de ulike innebygde reguleringsfunksjonene i frekvensomformeren
- Koble bryterpanelet i modellen for ulike måter å operere frekvensomforeren
- Lage oppstartsmanual for demomodellen
- Lage HMI-skjerm på PLS-panel for presentasjon av data

1.2 Overordnet om systemet

Under i figur 1.2 vises et forenklet blokkdiagram som illustrerer oppbygningen av systemet. Systemet består av fire hovedkomponenter; brukergrensesnitt gjennom PLS, frekvensomformer, motor og tilbakekobling via enkoder.

1.2 Overordnet om systemet



Figur 1.2: Overordnet virkemåte for systemet

Programmene i oppgaven ble utviklet ved hjelp av Sysmac Studio-programvaren, som tillot bruk av både ladderkode og strukturert tekstkode.

Frekvensomformeren ble brukt til å kontrollere hastigheten og dreiemomentet til motoren ved å justere inngangsfrekvensen og spenningen. Når motoren roterer vil frekvensomformeren telle pulser fra enkoderen, denne tilbakekoblingen brukes videre i PLS programmene til å styre hastigheten og posisjonen til motoren.

Strømforsyningen leverer den nødvendige strømmen til frekvensomformeren, som endrer frekvensen til motorspenningen for å regulere motorens hastighet. Brukergrensesnittet ga muligheten til å overvåke systemets drift og endre variabler som integreres videre i systemets programmer. Sysmac Studio-programvaren ble brukt til å regulere vinsjsystemet og til kalibrering, samtidig som Q2 Edit-programvaren ga kommunikasjon mellom PC og Q2A-frekvensomformeren via USB-kabel. Q2-Edit ble også benyttet til å lese, overvåke og til å endre parametere til frekvensomformeren. Gjennom HMI-skjermen på PLS-panelet har vi tilgang til funksjoner som gir oss muligheten til å endre variabelverdier og observere resultatene av prosessen.

Kapittel 2

Beskrivelse av utstyr og programvare

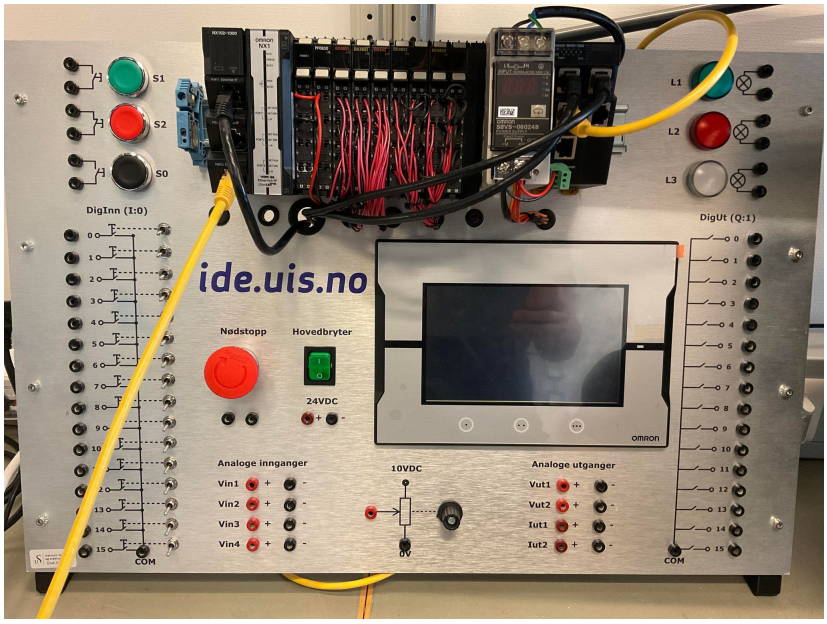
Dette avsnittet gir en kort innføring i de vesentlige komponentene og programvarene som danner grunnlaget for prosjektet. I tabell 2.1 gis en oversikt over hovedkomponentene i demomodellen.

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

Tabell 2.1: Liste over hovedkomponentene i demomodellen.

2.1 PLS

Figur 2.1 illustrerer PLS-stasjonen som ble benyttet i oppgaven. Stasjonen består av en NX102-1000 PLS fra Omron, som anvendes til å regulere vinsjsystemet. I tillegg til PLS-en, er det også en HMI-skjerm til stede, som brukes for å vise resultater og gi brukeren mulighet til å justere enkelte variabler.



Figur 2.1: Oversiktsbilde av PLS og HMI-skjerm

For å opprette kommunikasjon mellom PLSen og frekvensomformeren ble EtherCAT benyttes. EtherCAT (Ethernet for Control Automation Technology) blir brukt for å koble sammen ulike enheter i et automatiseringssystem, som for eksempel PLSer, frekvensomformere og sensorer. EtherCAT er basert på Ethernetteknologi og bruker en master-slave-arkitektur, der en EtherCAT-master styrer og kommuniserer med alle EtherCAT-slavene i systemet.

I PLSen er funksjonsblokkene for frekvensomformeren ikke inkludert i standard Sysmac Studio-pakken. For å implementere funksjonsblokkene til frekvensomformeren i programvaren, må en ekstra pakke legges til. Denne pakken inneholder de nødvendige filene for å opprette kommunikasjon og tilgang på funksjonsblokkene. Dette gjøres ved å følge trinnene som er beskrevet i vedlegg C.

Når pakken er lagt til, vil programvaren gjenkjenne frekvensomformeren som ”slave” og gjøre det mulig å legge den til i EtherCAT-seksjonen i Sysmac Studio. Dette gir oss tilgang til ønskede variabler, som brukes til å styre og overvåke frekvensomformeren.

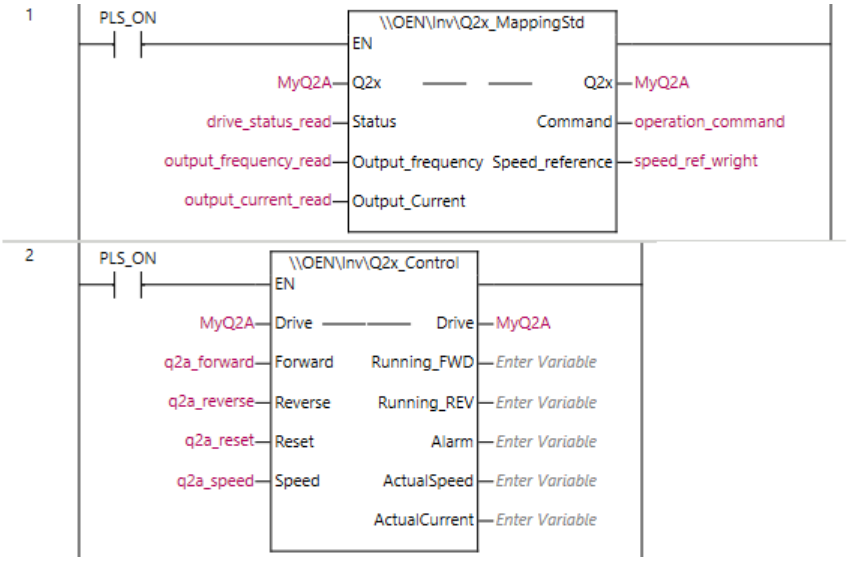
Under i figur 2.2 vises en PDO (Process Data Object) mapping som gir et eksempel på hvilke variabler som kan hentes ut fra frekvensomformeren.

Position	Port	Description	R/W	Data Type	Variable
	EtherCAT Network Configuration				
Node1	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
	20th receive PDO Mapping_Torque reference/limit_2020_01		W	INT	q2a_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_read
	38th transmit PDO Mapping_Output current_2120_01		R	UINT	output_current_read
	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
	40th transmit PDO Mapping_U6-31_2656_1F		R	UINT	torque_ref_filtered
	40th transmit PDO Mapping_U4-23_2654_17		R	UINT	DI_input_check

Figur 2.2: Oversikt over PDO mapping.

Ved å definere navn på variablene i kolonnen "Variable", kan disse variablene navngis for videre bruk i programvaren. PDO mapping gjør det mulig for frekvensomformeren å sende og motta data fra EtherCAT-bussen, og denne informasjonen kan brukes til å overvåke og styre frekvensomformeren.

Omron har utviklet to funksjonsblokker, *Q2x_ MappingStd* og *Q2x_ Control* , som er spesielt utviklet for styringen av Q2A frekvensomformeren. Disse funksjonsblokkene er vist under i figur 2.3.



Figur 2.3: Omrons funksjonsblokkene i Sysmac studios for Q2A-frekvensomformeren

Funksjonsblokken Q2x_MappingStd gir brukerne mulighet til å lese av variabler fra frekvensomformeren, noe som gjør det mulig å overvåke og kontrollere frekvensomformeren. Funksjonsblokken Q2x_Control blir benytte til å sette retning og pådrag til motoren.

Variablene "q2a_forward" og "q2a_reverse" styrer motorens kjøreretning. Endringer i frekvensreferansen til motoren utføres ved justering av variabelen "q2a_speed", som dermed påvirker pådraget. På grunn av frekvensreferansens begrensning til å kun behandle positive verdier, er det avgjørende å sikre at "q2a_speed"-verdien er positiv.

2.2 Q2A Frekvensomformer

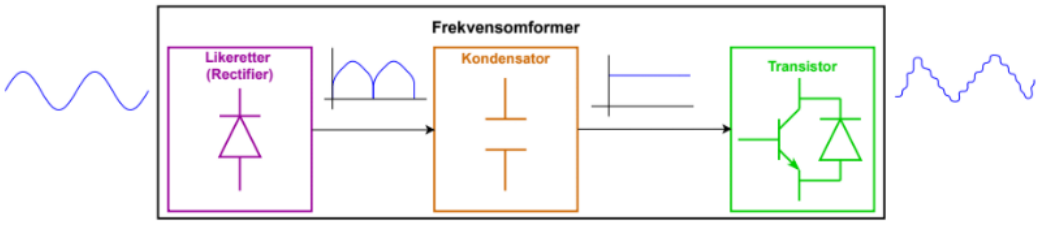
Til denne oppgaven ble det utlevert en Omron Q2A-A2012-AAA frekvensomformer (se figur 2.4). Denne frekvensomformeren er utviklet for å gi presis kontroll over hastigheten til en elektrisk motor ved å variere frekvensen på strømmen som leveres til motoren.

Figur 2.4: Bilde av Omron Q2A-A2012-AAA [1]

Q2A-A2012-AAA frekvensomformeren er en svært allsidig enhet som kan brukes til å kontrollere mange forskjellige typer motorer i både hastighet eller momentkontroll. Den kan styre elektriske motortyper opp til 590 Hz.

Med sin funksjon som motorregulator kan Q2Aen regulere spenningen og frekvensen som blir levert til motoren, og på denne måten sikre at den beveger seg med ønsket hastighet og moment. Dette gir oss muligheten til å justere motorhastigheten og momentet etter behov.

Under i figur 2.5 vises prosessen som skjer inne i frekvensomformeren for å styre en elektrisk motor.

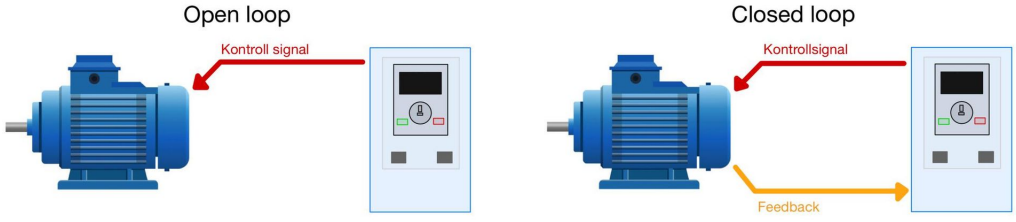


Figur 2.5: [7] Bildet gir en illustrering av prosessen som foregår i frekvensomformeren for å kontrollere en elektrisk motor.

En frekvensomformer inneholder elementer som likeretter, kondensator og vekselretter (transistor) for å styre hastigheten på en elektrisk motor. Likeretteren gjør om vekselstrøm (AC) til likestrøm (DC), som sendes til kondensatoren for filtrering. Til slutt endres likestrømmen tilbake til vekselstrøm av transistoren, som fungerer som en bryter og bruker teknikker som pulsbreddemodulering (PWM) for å justere frekvensen og dermed styre motorens hastighet. [3]

2.2.1 Vektorkontroll

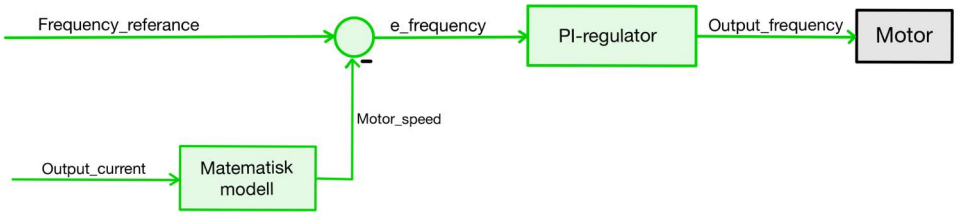
Vektorkontroll er en metode for å kontrollere en motor ved å modulere spenningen og strømmen som leveres til motoren. Målet med vektorkontroll er å oppnå optimal ytelse, høy effektivitet og rask dynamisk respons. Det skilles i to typer vektorkontroll: open-loop og closed-loop vektorkontroll. Under i Figur 2.6 blir det illustrert hvordan metodene sender og mottar(feedback) signaler mellom frekvensomformeren og motoren. [6]



Figur 2.6: Illustrasjon av signalutveksling i open-loop og closed-loop vektorkontroll

Open-loop vektorkontroll

Open-loop vektorkontroll(OLVC) er en metode for vektorkontroll. Denne metoden estimerer motorens tilstand, inkludert hastighet og moment. Basert på et estimatet, genererer frekvensomformeren et signal som styrer strømmen og spenningen til motoren. Endringer i lasten eller støy påvirker nøyaktigheten av estimatet og dermed redusere ytelsen til systemet. I figur 2.7 så illustreres et forenklet blokkskjema av open-loop vektorkontroll i vår Q2A-frekvensomformer.



Figur 2.7: Blokkdiagram over open-loop vektorkontroll i Q2A-frekvensomformeren

Parameteren "frequency_reference" indikerer den ønskede hastigheten til motoren, uttrykt i Hertz (Hz), som frekvensomformeren skal operere på. Frekvensomformeren bruker en innebygd matematisk modell av motoren for å beregne en verdi kalt "motor_speed", basert på motorparametrene som har blitt innhentet gjennom en prosess kalt autotuning.

Autotuning-prosessen er en metode der frekvensomformeren utfører målinger av strøm og spenning basert på de innlagte motorparametrene. Disse målingene brukes til å finjustere modellen og forbedre nøyaktigheten av dens prediksjoner.

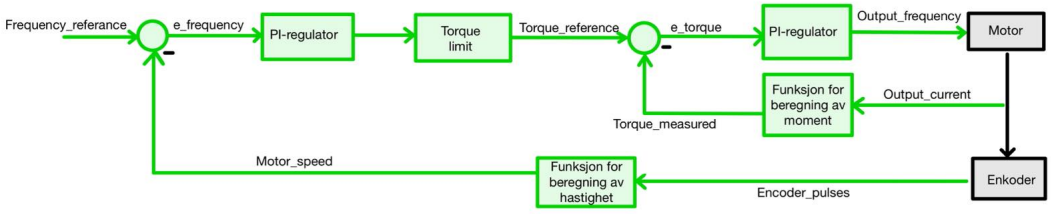
I denne modellen er "output_current", som er en måling av strømmen som blir levert til motoren, den eneste inngangsvariabelen. "motor_speed" er en utgangsverdi fra modellen og representerer et estimat av den faktiske hastigheten til motoren.

Innebygd i frekvensomformeren beregner den innebygde PI-regulatoren vist i 2.7 pådraget (output_frequency) basert på reguleringsavviket(e_frequency). Signalet (output_frequency) blir omgjort i frekvensomformeren til strøm og spenning, som deretter leveres til motoren.

Closed-loop vektorkontroll

Closed loop vektorkontroll(CLVC) er en metode som brukes for å kontrollere en vekselstrømsmotor. Closed loop vektorkontroll muliggjør kontrollering av motorens hastighet og moment på en nøyaktig og effektiv måte, samtidig som den gir oss informasjon om feil og avvik. Frekvensomformeren benytter informasjon fra evt. sensorer for å optimalisere systemet. I vårt system benyttes en enkoder som sensoren. Enkoderen sender pulser til en mikroprosessor koblet til frekvensomformeren. Pulsene gir tilbakemeldinger/informasjon om motorens posisjon og hastighet. Denne informasjonen blir brukt av mikroprosessoren til å generere PWM-signaler som sendes til frekvensomformeren. Frekvensomformeren konverterer disse signalene til den riktige frekvensen og spenningen som trengs for å kjøre motoren med ønsket hastighet og moment. I CLVC(Closed-loop vektorkontroll) vil en matematisk modell av motoren og belastningen brukes for å optimalisere styringen. Denne modellen blir justert gjennom tuningprosessen på samme måte som i open-loop.

Basert på generell teori om vektorkontroll kombinert med blokkdiagrammet fra brukermanualen vist i vedlegg D, ble et blokkdiagram for systemet utviklet. Blokkdiagrammet er vist under i figur 2.8.



Figur 2.8: Blokkdiagram av closed loop vektorkontroll i frekvensomformeren, den grønne delen er omformeren og den grå er maskinvare.

Figur 2.8 viser blokkskjemaet for closed loop vektorkontroll systemet. Enkoderpulsene fra enkoderen (encoder_pulses) brukes til å beregne motorhastigheten "motor_speed" gjennom en matematisk modell. Frekvensreferansen (frequency_reference) og den målte motorhastigheten ("motor_speed") sammenlignes. Reguleringsavviket (e_frequency) går deretter gjennom en innebygget PI-regulator og videre til en funksjon som begrenser momentet (Torque limit). Funksjonen er basert på en pro-

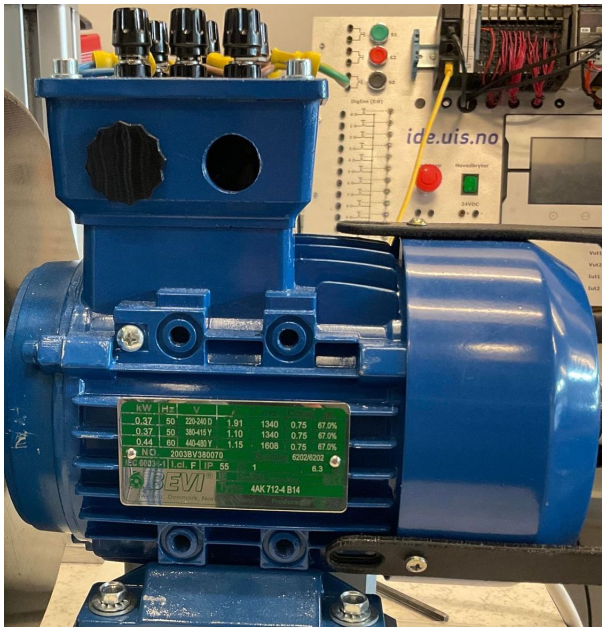
sentandel av hvor mye strøm motoren tåler [10]. Ut ifra strømtrekket blir det lagd en momentreferanse(torque_reference). "Torque_measured" er momentet motoren opplever. Basert på momentavviket(e_torque) mellom målt og referansen, beregner den innebygde PI-regulatoren pådraget(Output_frequency).

Closed loop vektorkontroll inkluderer muligheten til å få en vekselstrømsmotor til å utvikle kontinuerlig fullt dreiemoment selv ved null rotasjonshastighet, samt god regulering av hastighet, moment og muligheten for automatisk korreksjon av feil og avvik. Dette gjør teknikken spesielt egnet for kran- og heiseanvendelser, der motoren må produsere fullt dreiemoment før bremsen slippes.

Valget mellom open-loop og closed-loop vektorkontroll avhenger av oppgavens krav. Hovedsakelig ble det anvendt closed-loop vektorkontroll i deloppgavene. Imidlertid har ble det også benyttet open-loop vektorkontroll som et verktøy for systemtesting, samt for å identifisere nøkkelparametrene som gir innsikt i systemets dynamiske egenskaper.

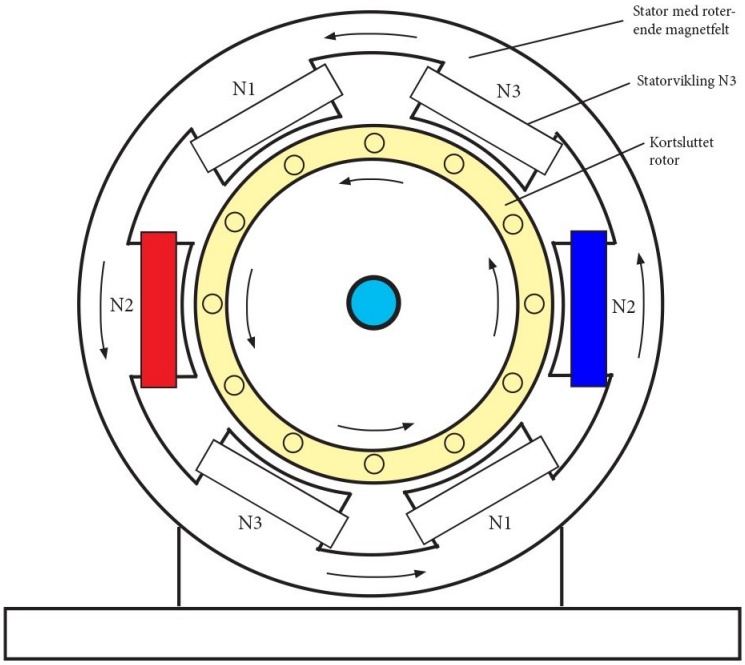
2.3 Motor

Motoren benyttet i oppgaven er en asynkron trefasemotor fra Bevi. Motoren har en effekt på 0.44kW.[5] Motoren er vist under i figur 2.9.



Figur 2.9: Motoren 4AK 712-4 4-pole B14 fra Bevi

I en asynkron trefase motor skapes et roterende magnetfelt i statorvindingene ved hjelp av trefase vekselstrøm. Dette magnetfeltet skaper en elektromagnetisk kraft som påvirker rotoren og får den til å rotere. Rotoren er laget av ledende materiale, og den er konfigurert slik at den kan samhandle med det roterende magnetfeltet i stator.



Figur 2.10: Illustrering av rotor og stator i en asynkron motor [8]

Figur 2.10 illustrerer hvordan en asynkron trefase motor fungerer. Den viser hvordan statorvindingene og rotoren samhandler for å skape det roterende magnetfeltet som driver motoren.

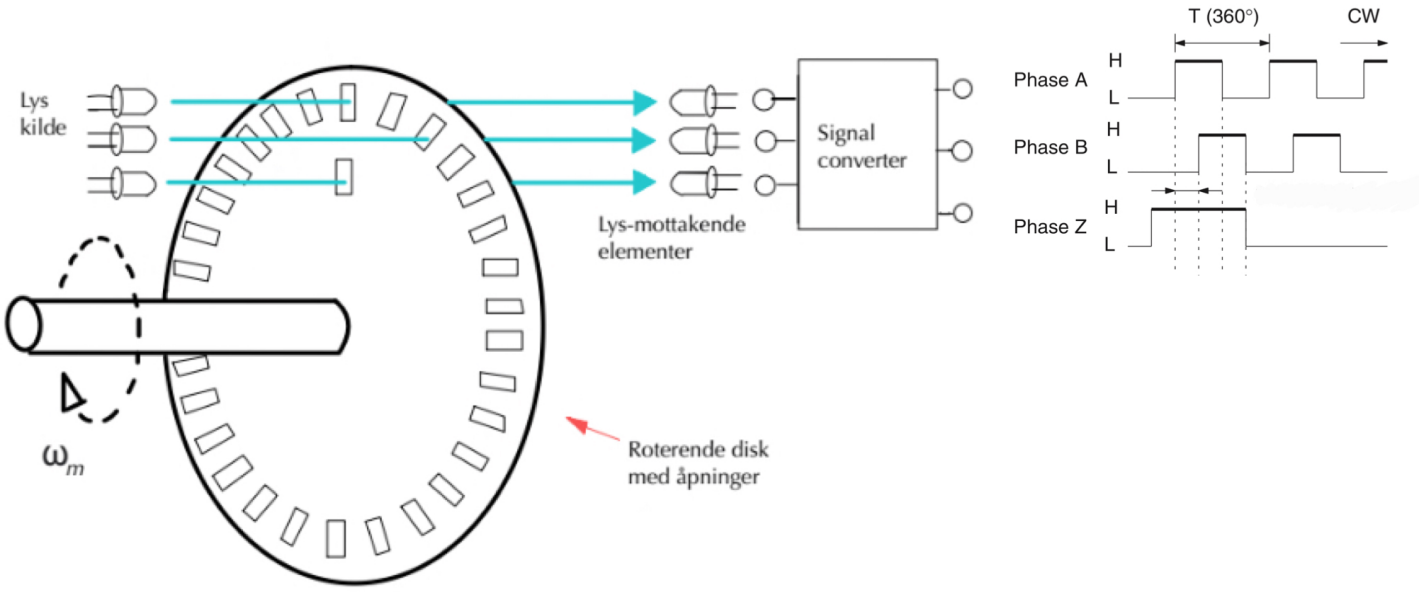
Styring av en asynkron trefasemotor kan oppnås ved å justere spenningen og frekvensen tilført statorvindingene. Dette blir i denne oppgaven gjort ved hjelp av en frekvensomformer. Ved å variere frekvensen kan motorens hastighet og dreiemoment kontrolleres.[11]

2.4 Enkoder

I oppgaven ble enkoderen E6C2-CWZ1X benyttet. En nøkkelspesifikasjon for denne enkoderen er dens "pulser per rotasjon" (PPR), som er oppgitt til å være 2000. Dette indikerer at for hver fulle omdreining av aksen, som tilsvarer 360 grader, vil enkoderen generere 2000 pulser. Enkoderen E6C2-CWZ1X er vist under i figur 2.11

Figur 2.11: Bilde av enkoderen E6C2-CWZ1X festet på motoren

E6C2-CWZ1X er en inkrementell enkoder med kvadraturutgang, noe som betyr at den produserer to utgangssignaler (vanligvis kalt A og B) som er 90 grader faseskiftet i forhold til hverandre. Dette tillater deteksjon av både rotasjonshastighet og retning. Under er logikken bak den inkrementelle enkoderen vist i figur 2.12



Figur 2.12: Pulsgenerering til en inkrementell enkoder med kvadratutgang

I en inkrementell enkoder kan antallet effektive pulser (eller "counts") som kan brukes til posisjonering være opptil 4 ganger PPR-verdien, siden det er 4 unike tilstander per syklus for de to kvadraturutgangene. Derfor vil en 2000 PPR inkrementellenkoder gi 8000 counts per omdreining. [4]

Det er viktig å merke seg at selv om PPR og ”counts” ofte brukes om hverandre, er de ikke det samme. PPR refererer til antall pulser som genereres på hver av de to kanalene (A og B), mens ”counts” refererer til totalt antall unike endringer i en syklus. Dette gjøres ved å registrere antall endringer fra lav til høy og høy til lav, i de to utgangssignalene.

Som vist i figuren 2.12 har enkoderen også et ekstra utgangssignal, Z, som indikerer en referanseposisjon til enkoderen. Dette signalet genereres én gang per omdreining og gir mulighet for å telle antall omdreininger som er utført. Z-signalet brukes også til å synkronisere posisjonsmålingen og til å nullstille posisjonstelleren for hver rotasjon.

Som forklart tidligere består vårt system av en asynkronmotor og en E6C2-CWZ1X-enkoder fra Omron. Dette oppsettet innebærer imidlertid noen utfordringer når det gjelder å lese av Z-signalet fra enkoderen.

En asynkronmotor har ikke en fast hastighet som er i perfekt synkronisering med frekvensen av strømforsyningen. En asynkronmotor kan slippe noe i forhold til strømforsyningen (det vil si, det kan være en viss slip mellom rotorens hastighet og frekvensen av strømforsyningen). Dette kan skape utfordringer med å nøyaktig korrelere Z-signalet med en bestemt motorposisjon, spesielt over flere rotasjoner.

Det skal bemerkes at utfordringene med å lese av Z-signalet ikke betyr at det er umulig å lese det med en asynkronmotor. Det indikerer bare at nøyaktigheten kan være begrenset. Med våre instillinger i Q2Aen ga ikke frekvensomformeren oss tilgang til å lese av Z-signalet.

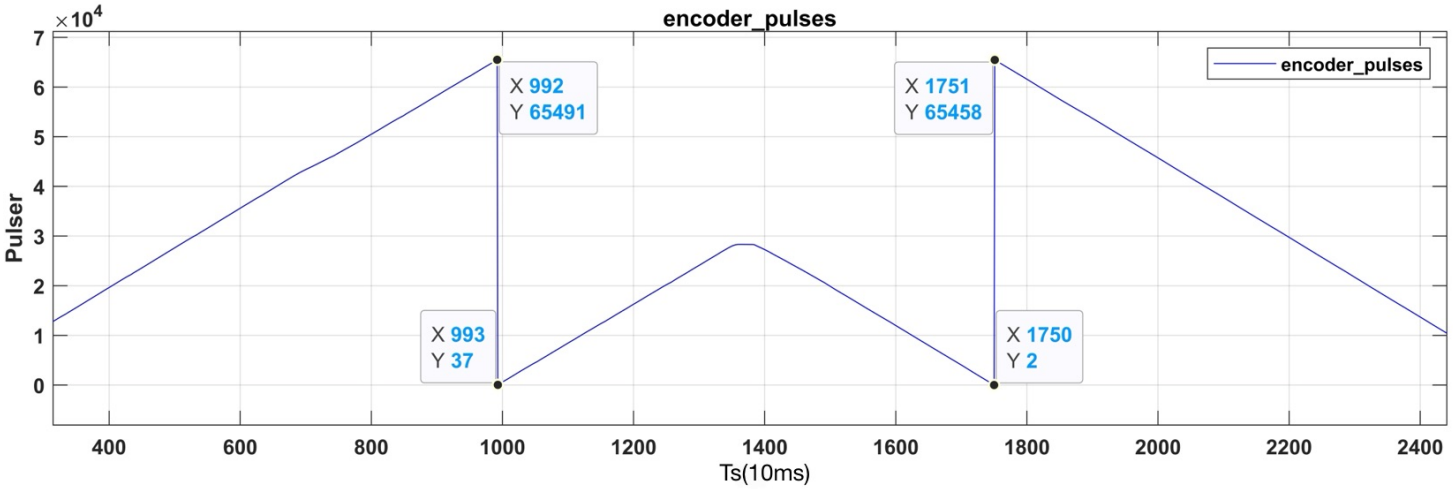
2.4.1 Q2A signalbehandling av enkoder pulser

Enkoderpulsene danner grunnlaget for beregningene både i kalibrering og regulering av vinsjsystemet, og for å håndtere disse pulssignalene ble frekvensomformeren sin innebygde pulsteller(”encoder_pulses”) benyttet 2.3. Variabeltelleren har ansvar for å telle antall pulser som blir generert av enkoderen, men med våre innstillinger og komponenter, registrers egentlig antall ”counts”. Dette betyr at senere i oppgaven når det refereres til variabelens pulser, er det viktig å ta med seg at det egentlig er registrerte ”counts”. En forklaring av den innebygde telleren er presentert i Figur 2.13, hentet direkte fra Q2A frekvensomformerens tekniske manual [10].

U6-18 (07CD)	SpdDetectPG1 Counter	Shows the number of pulses for speed detection (PG1). Unit: 1 pulse	10 V: 65536

Figur 2.13: Denne figuren illustrerer tellerens grunnleggende funksjonalitet, dens måleenhet for tellinger(pulses), samt den maksimale signalverdien den kan behandle (65536).

Siden Z-signalet som resatte rotasjonstelleren ikke var tilgjengelig, vil den innebygde pulstelleren i Q2Aen, registrere pulser helt opp til 16-bit (som er maks verdien til signalet). Når den når 16-bit som representerer en verdi på 2^{16} (eller 65536) vil pulstelleren resettes til 0. Motsatt vil den også gå fra 0 til en verdi på 65536 når motoren roterer i motsatt retning. Dette blir illustrert i figuren 2.14

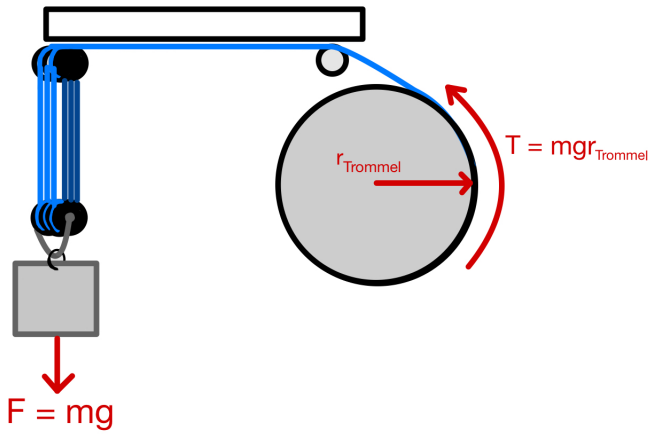


Figur 2.14: Figuren viser hvordan verdien til pulstelleren stiger eller synker etter som den registrer antall "counts" når motoren kjører fremover eller bakover. Motoren kjøre fremover til omtrent Ts = 1380, før den bytter til å kjøre bakover.

Som forklart tidligere vil verdien på telleren bli resatt når maksverdien på 65536 blir nådd. I figuren 2.14 vil man imidlertid se at den øverste verdien er på 65491 og ikke 65536. Dette skyldes at det ble benyttet DataTracing i Sysmac Studios med en samplingsrate på 10 ms, og på grunn av denne samplingsraten kan noen enkelte verdier gå tapt under plotting.

2.5 Vinsj

Vinsjen i vårt system består av en trommel tilknyttet en elektrisk motor som gir mulighet for løfting og senking av last. Moment og effekt er nøkkelkomponenter som bidrar til å drive vinsjen i løfteprosessen. Figur 2.15 gir en forenklet illustrasjon av de kreftene som virker på en vinsj under en løfteoperasjon.



Figur 2.15: Forenklet fremvisning av kreftene som virker på en vinsj under løft.

- $F \rightarrow$ Dette representerer summen av kreftene som virker på et objekt, målt i newton (N). Dette kan inkludere krefter som gravitasjon, friksjon, og andre påvirkende krefter.
- $r_{Trommel} \rightarrow$ Dette er radiusen til trommelen, som måles i millimeter (mm). Den er definert som avstanden fra midtpunktet til ytterkanten av trommelen (hvor tauet først legges). Mengden tau eller vaier som blir spolt inn eller ut per omdreining vil påvirke denne verdien.
- $d_{Tau} \rightarrow$ Dette representerer diameteren av tauet, målt i millimeter (mm). I praksis er det tykkelsen på tauet eller vaieren som brukes i vinsjen.
- $m \rightarrow$ Dette symboliserer massen til objektet som er involvert, målt i kilogram (kg).
- $T \rightarrow$ Dette representerer momentet, målt i newtonmeter (Nm). Moment er et mål på rotasjonskraften som virker på et objekt rundt et fast punkt. I

konteksten av en vinsj, vil dette være kraften som påvirker rotasjonen av trommelen der tauet eller vaieren er festet.

- $g \rightarrow$ Dette representerer tyngdeakselerasjonen, også kjent som gravitasjonskraften på jorden. Det er målt i meter per sekund i kvadrat (m/s^2).

2.5.1 Moment i vinsj

Under et vinsjløft er det to momenter som er sentrale: motormoment og lastmoment. Motormomentet, som må være større enn lastmomentet for å sikre løft av lasten, er avhengig av flere faktorer som friksjon, bevegelige delers treghet og lastens egen vekt.[2]

For å finne lastmoment benyttes ligning 2.1.

$$T_{last} = m \cdot g \cdot r_{total} \tag{2.1}$$

Basert på ligning 2.1, ble det konkludert med at lastmomentet er i direkte forhold til den totale radiusen. Dette innebærer at en økning i radiusen under innspoling resulterer i en tilsvarende økning i lastmomentet. Tilsvarende, under utspoling, når radiusen avtar, vil lastmomentet reduseres i samsvar.

For å beregne momentet til motoren benyttes ligning 2.2

$$T = 9550 \cdot \frac{P}{n} \tag{2.2}$$

- $T \rightarrow$ Dette symboliserer motormomentet, som er rotasjonskraften som genereres av motoren for å utføre det nødvendige arbeidet. Det er målt i newtonmeter (Nm).
- $P \rightarrow$ Dette representerer den mekaniske effekten som motoren genererer, målt i kilowatt (kW). Dette er grunnleggende sett mengden energi som motoren produserer per tidsenhet.
- $n \rightarrow$ Dette representerer motorens rotasjonshastighet, målt i antall omdreininger per minutt (rpm). 'n' symboliserer denne hastigheten og er et viktig mål på motorens ytelse.

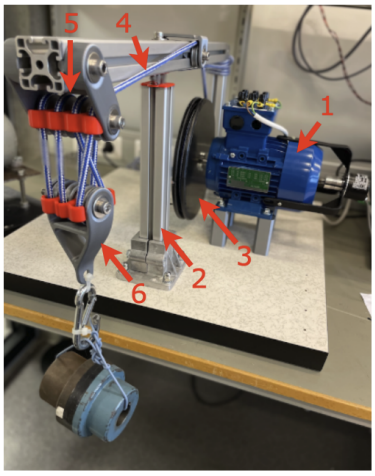
- 9550 → Dette er en konverteringsfaktor. Den brukes for å sikre korrekt sammenheng mellom enhetene kilowatt (kW), omdreininger per minutt (rpm), og newtonmeter (Nm).

Basert på formelen 2.2, er det klart at motormomentet er direkte proporsjonalt med motorens effekt og omvendt proporsjonalt med motorens hastighet. Dette innebærer at for en gitt effekt, vil en økning i motorens hastighet resultere i en tilsvarende reduksjon i motormomentet. Omvendt vil en reduksjon i hastigheten resultere i en tilsvarende økning i motormomentet. Dette illustrerer prinsippet om at momentet er et produkt av effekten og omvendt proporsjonalt med hastigheten, og hvordan endringer i disse variablene påvirker motormomentet.

2.5.2 Vinsjmodell

Vinsjmodellen vist i figur 2.16 ble gjort tilgjengelig fra instituttet. Den består av følgende:

- **1** → Motor
- **2** → Krankonstruksjon egnet for håndtering av stor kraft.
- **3** → 3d printet trommel med feste til tau.
- **4** → 6mm tykt tau.
- **5** → Seks 3d printede trinser
- **6** → 3d printet feste for lodd



Figur 2.16: [9] Bilde viser vinsjens deler. Tallene med piler referer til listen.

Vinsjen er utstyrt med seks trinser, som fungerer for å minske kraften som er nødvendig for å løfte en last. Dette oppnås ved å distribuere vekten av lasten på en mer effektiv måte. Derimot, medfører dette en økning med seks gangen så mye tau som kreves for å løfte lasten.

2.6 Programvare

I denne bacheloroppgaven er det primært benyttet to ulike programvareverktøy; Q2 Edit og Sysmac studios.

2.6.1 Q2-Edit

Omron's Q2-Edit programvare er spesielt designet for å jobbe med Omron's Q2A frekvensomformere, og inneholder en rekke funksjoner som gjør den effektiv for å redigere og konfigurere disse enhetene. Noen av de funksjonene som ble benyttet og som gjør Q2-Edit effektiv er:

- Parametere redigering: Q2-Edit gir brukerne muligheten til å konfigurere og redigere parametere for Q2A frekvensomformeren. Dette gjør det mulig for brukere å tilpasse enheten til deres spesifikke behov og krav.
- Overvåking: Q2-Edit gir også brukere muligheten til å overvåke enhetens status og ytelse i sanntid. Dette kan hjelpe brukere med å identifisere problemer.
- Diagnostikk: Q2-Edit har også diagnostiske verktøy som gjør det mulig for brukere å diagnostisere og feilsøke problemer med enheten.
- Sikkerhetskopiering og gjenoppretting: Q2-Edit lar brukere sikkerhetskopiere og gjenopprette enhetens konfigurasjonsdata.

For å etablere en kommunikasjonsforbindelse mellom programvaren og frekvensomformeren, brukes en USB-kabel. Det blir illustrert i vedlegg G, hvordan man oppretter forbindelse mellom frekvensomformeren og Q2-edit.

Parametere

Tabell 2.2 gir en oversikt over parametere i frekvensomformeren, anvendt denne oppgaven. En nærmere forklaring av disse parameterene er gitt under tabellen. Disse parameterene tillater konfigurering av frekvensomformeren for å imøtekomme ønskede behov. Justeringene kan utføres gjennom Q2 Edit-programvaren.

Standardinnstillinger

Nummer	Parameter	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
C2-01	Jerk@Start of Accel	0.2 sec
C5-01	ASR PGain 1	20
C5-02	ASR ITime 1	0.5
F1-01	Enc1 Pulse Count (PPR)	2000 PPR
D5-01	Torque ctrl selection	0: Speed control
F6-06	Trq Ref/Lim Comms	0

Tabell 2.2: Våre standard innstillinger i frekvensomformeren

- A1-02 : Control Methode
  Parameteren setter kontrollmetoden for systemet. Det finnes 8 forskjellige alternative kontrollmetoder, men i vår oppgave ble det benyttet closed-loop vektorkontroll, og dermed ble innstilling 3 (CLVector) valgt.

- B-01 : Freq. Ref. Sel 1
  Parameteren brukes til å velge metoden for innstilling av frekvensreferansen på frekvensomformeren. I vår oppgave ble innstilling 3 (Option PCB) benyttet.
  3: Option PCB - Frekvensreferansen kan settes via EtherCAT-grensesnittet på frekvensomformeren.

- B1-02 : Run. Comm. Sel 1
  Parameteren brukes til å velge metoden for setting av run-signalet. I vår oppgave ble innstilling 3 (Option PCB) benyttet.

- 3: Option PCB - Run-signalet kan settes via EtherCAT-grensesnittet på frekvensomformeren.
- C1-01 : Accel Time 1
  Parameteren setter lengden på tiden det tar å gå fra null til maks utgangsfrekvens. Det kan settes en verdi mellom 0-6000s og default verdien er på 10s. Vi valgte å benytte en verdi på 2 sekunder fordi vi arbeider på et lite arbeidsområde.
- C1-02 : Decel Time 1
  Parameteren setter lengden på tiden det tar å redusere fra maks utgangsfrekvens til null. Det kan settes en verdi mellom 0-6000s og default verdien er på 10s. Vi valgte å benytte en verdi på 2 sekunder fordi vi arbeider på et lite arbeidsområde.
- C2-01 : Jerk@Start of Accel
  Påvirker glattheten av akselerasjon og retardasjon.
- C5-01 : ASR PGain 1
  Påvirker den innebygde hastighetsregulatoren ved å justere forsterkningen.
- C5-02 : ASR ITime 1
  Påvirker den innebygde hastighetsregulatoren ved å justere integral tiden.
- D5-01 : Torque ctrl selection
  Lar bruker velge mellom hastighetsregulering (0) momentregulering (1).
- F1-01 : Enc1 Pulse Count (PPR)
  Setter enkoderens pulses per revolution [PPR].
- F6-06 : Trq Ref/Lim Comms
  Parameteren setter funksjonen som aktiverer og deaktiverer muligheten til å endre momentreferansen.

I tillegg til de nevnte parameterne, tok vi også i bruk motorparametrene. Tabell 2.3 viser de nødvendige parametrene for oppgaven, deres nummerkode, navnene deres i frekvensomformeren og variabelnavnene i programmet vårt. Vi leste av og justerte disse parameterne etter behov i kodene for reguleringsprosessene."

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
U1-52	Torque Ref from Comm	torq_ref_wright
U6-18	Encoder Pulse Counter	encoder_pulses

Tabell 2.3: Våre innstillinger i frekvensomformeren

- 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 "counts" fra enkoderen.

Autotuning

For at frekvensomformeren skal kunne identifisere hvilken motor den er koblet til, må det utføres en autotuning-prosess. En riktig autotunet frekvensomformer vil være mer effektiv, med lavere strømforbruk for samme dreiemoment, og gi bedre ytelse med en mer lineær og stabil drift. Autotuningen innebærer at frekvensomformeren kjører gjennom en rekke frekvens- og strømverdier, og kan utføres enten direkte via displayet på frekvensomformeren eller gjennom programvaren Q2 Edit. Resultatene fra autotuning-prosessen vises i vedlegg G.

2.6.2 Sysmac Studio

Sysmac studio er en programvareplattform som er utviklet av Omron for å designe, konfigurere og programmere automatiseringsløsninger basert på deres Sysmac-arkitektur. Vi benyttet Sysmac studio til å utvikle kodene som er nødvendig for å kalibrere og regulere systemet. Noen av funksjoner som gjør Sysmac Studio effektivt er:

- Integrasjon av flere teknologier: Sysmac Studio integrerer flere teknologier, inkludert PLC-programmering, motion control, HMI-design, ladder-programming og programmering i strukturert tekst.
- DataTracing: Sysmac Studio gir mulighet for samling og analyse av data fra ulike deler av automatiseringssystemet, slik at brukere kan overvåke ytelse og optimalisere driften.
- Fileksportering: Sysmac Studio er en brukervennlig programvare som fungerer godt sammen med andre programvarer som Excel, Matlab og Simulink. Dette gjorde det mulig for oss å analysere resultatene våre i andre programmer.

Kapittel 3

Teach funksjonen

For å muliggjøre regulering av vinsjsystemet var det nødvendig å fastsette et arbeidsområde for systemet. Dette kapittelet beskriver teorien, kodene og resultater for kalibrering av vinsjsystemet.

For å fastsette et arbeidsområdet ble det utviklet en kalibreringsfunksjon, kalt "teach". Denne funksjonen utfører en automatisert kjøring fra bunnen til toppen av vinsjsystemet for å beregne arbeidsområdet som vinsjen kan operere innenfor. Det kalibrerte området, fra bunnen til toppen, ble deretter satt som en begrensning for hvor mye vaier som kan spoles inn eller ut i de andre programmene.

3.1 Teori

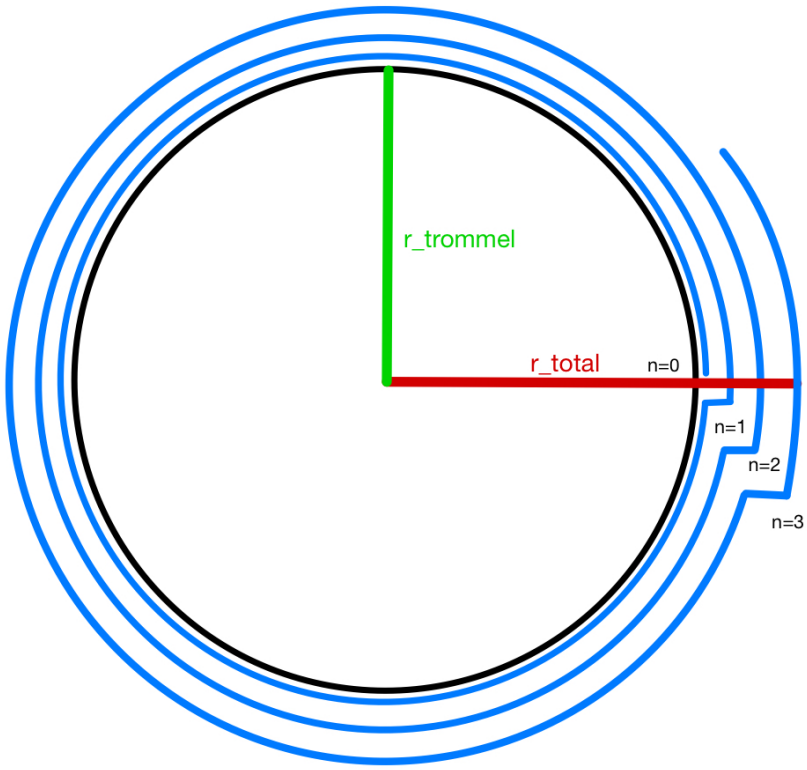
For å beregne hvor mye tau som var spolt inn på trommelen benyttet vi formelen for omkretsen av en sirkel, hvor n indikerer antall runder motoren har kjørt. Formelen for omkrets er vist i likning 3.1

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

Ettersom at tauet legger seg i trommelen vil den totale radiusen (r_{total}) variere. Utregningen av r_{total} blir vist i likning 3.2 der d_{vaier} er diameteren på vaieren.

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

Under i figur 3.1 er teorien bak totalradius illustrert.



Figur 3.1: Figur som viser forholdet mellom radius på trommel, runder og total radius

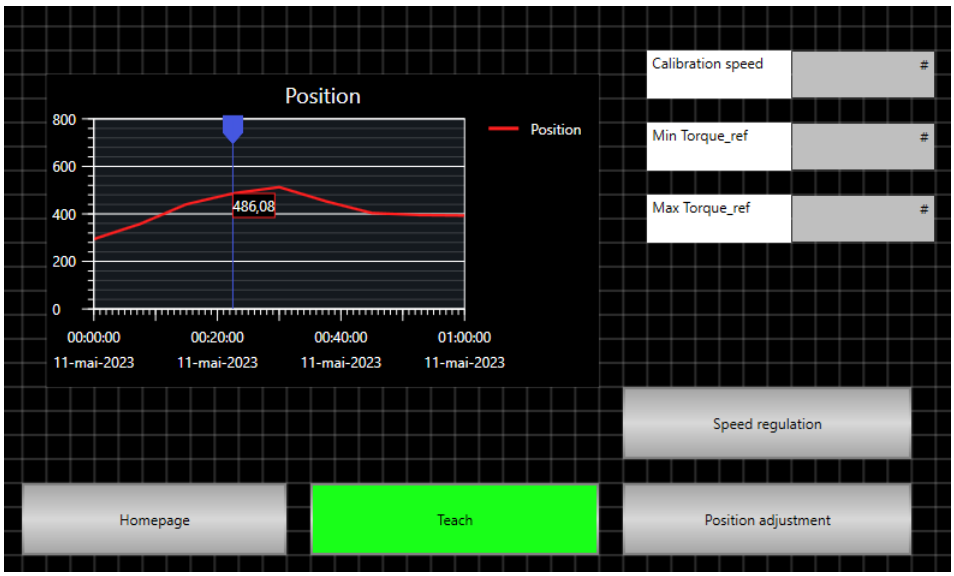
Som illustrert over i figur 3.1 vil tauet som legges hver runde øke radiusen. Oppdatert informasjon om radiusen spiller en sentral rolle i beregningen av høyde og kalibrering av arbeidsområdet. Likningen vi benytter for å holde den oppdaterte omkretsen til tauet blir vist i likning 3.3

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

3.2 Teach-funksjonen i praksis

HMI-skjerm til teach-funksjonen

Under i figur 3.2 illustreres HMI-skjermen som benyttes under 'teach'-funksjonen. HMI-skjermen muliggjør endringer av variablene benyttet under kalibreringen. I tillegg er HMI-skjermen utstyrt med en kontinuerlig oppdatert posisjonsgraf som viser nåværende høyde.



Figur 3.2: HMI-skjerm benyttet under kalibrering

Stegvis forklaring av funksjonen

Funksjonen fungerer på følgende måte:

- Loddet plasseres på bakken med vaieren helt utspolt. Ekstra tau som spoles inn før loddet løftes, håndteres av "teach"-funksjonen.
- Brukeren angir via HMI-skjermen hvilken hastighet systemet skal benytte under "teach"-funksjonen.

- Brukeren angir også en verdi som refererer til det motormomentet som kreves for å løfte lasten over bakken (altså når motormomentet > lastmomentet). Siden lastmomentet avhenger av mange faktorer som f.eks motorhastighet, mengde tau på trommelen før løft (2.1 og vekten til loddet, så benyttet vi manuell kjøring for å finne motormomentet som kreves for å løfte loddet.
- Brukeren setter også en momentgrense. Denne grensen indikerer at motoren må stoppe når momentet overstiger 700, og brukes for å registrere når loddet har nådd toppen.
- Når lasten når toppen (momentgrensen), vil hastigheten settes til 0, og loddet vil forbli på toppen inntil nye instrukser gis.
- PLSen mottar informasjon fra variabelen "encoder_pulses" for å beregne antall runder motoren har kjørt.
- Ved hjelp av matematikk og koder (som blir forklart senere i kapittel 3.1 og 3.3), kan vi nå bestemme hvor mye tau som er på trommelen, trommelens radius, og systemets arbeidsområde.

3.3 Koder

Under i kodene 3.1 vises kodene for starten av teach funskjonen.

Kode 3.1: Teach Koder for retning

```
14 IF HMI_Teach = TRUE THEN
15     q2a_forward:=TRUE;
16     q2a_reverse:=FALSE;
17     q2a_speed:=HMI_teach_speed;
```

Linje 14 kontrollerer at "teach" knappen er aktivert(HMI_Teach). Når teach knappen er aktivert kjører motoren i retningen som løfter loddet. Linje 17 bestemmer hastigheten motoren skal kjøre under "teach". Denne hastigheten bestemmes av variablen HMI_teach_speed som blir valgt på hmi-skjermen.

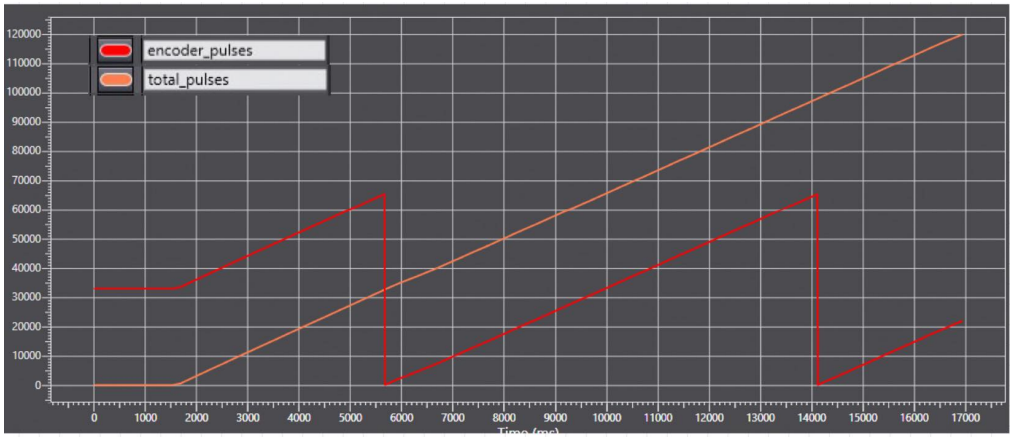
Som forklart i kap 2.4 så hadde vi ikke tilgang på Z-signalet til enkoderen. Vi implementerte derfor en "software counter" for å telle motorens rotasjoner. Kodene under i 3.2 viser logikken bak denne telleren.

Kode 3.2: Teach Koder for total_pulses

```
18 diff_pulses:=encoder_pulses - encoder_pulses_old;
19 IF q2a_forward = TRUE THEN
20     IF diff_pulses > 1 THEN
21         total_pulses:=total_pulses + diff_pulses;
22         encoder_pulses_old:=encoder_pulses;
23     END_IF;
24     IF diff_pulses < -1 THEN
25         encoder_pulses_old := encoder_pulses_old - 65536;
26     END_IF;
27 END_IF;
```

"total_pulses" er software-telleren som holder oversikt over enkoderens pulser. I kodelinje 18 beregnes endringer i enkoderverdiene, som representerer en ny puls. Fra linje 20 til 22, inkrementerer koden totalverdien av "total_pulses" med hver nye detekterte puls. I linje 25 blir det tatt høyde for enkoderens tilbakestilling, for å sikre nøyaktig beregning av forskjeller mellom veridene encoder_pulses og encoder_pulses_old.

Figur 3.3 illustrerer sammenhengen mellom software-telleren og enkoderpulser. Her ser vi at software-telleren(total_pulses) øker kontinuerlig, mens enkoderpulser nullstilles når de når en 16-bits grenseverdi på 65536.



Figur 3.3: Figur som viser forholdet mellom "encoder_pulses" og "total_pulses"

Figuren demonstrerer at enkoderen nullstilles når den når en verdi av 65536. Dette kunne ha ført til feil beregning av antall nye pulser registert("diff_puls"). For å unngå dette og sikre korrekt differanse mellom ny og gammel verdi, justerer vi

”encoder_pulses_old” ved å trekke fra 65536.

Siden radiusen på trommelen oppdateres med fulle rotasjoner ble kodene for ”roundcounter” og ”roundcounter_decimal” utledet. Hensikten med rundetellerene var å holde oversikt over hvordan radiusen ble oppdatert, samtidig som å opprettholde en nøyere oversikt over rundene som var kjørt. Kodeliste 3.3 viser kodene for tellerene.

Kode 3.3: Teach Koder for roundcounter

```
28 roundcounter:=TRUNC(LREAL_TO_REAL(roundcounter_decimals));
29 roundcounter_decimals:=total_pulses/8000;
```

Linje 29 viser hvordan vi regner ut ”roundcounter_decimals”. Som tidligere beskrevet i kap 2.4 var en runde på trommelen tilsvarende en enkoderverdi på 8000. Ved å dele ”total_pulses” på 8000, oppnår vi en oversikt over antall runder motoren har kjørt. Det var i tillegg ønskelig å inkludere hvor stor del av neste runde som var kjørt. Linje 28 viser hvordan variablene ”roundcounter” blir lagd ved å benytte ”roundcounter_decimals”. ”roundcounter” ble brukt til å bestemt hvor mange hele runder som var fullført. Vi ønsket kun at denne telleren tok i bruk heltall og kun tok med de hele rundene som hadde blitt kjørt. Vi benytet derfor ”TRUNC”-funksjonen. Denne funksjonen runder alltid ned til nærmeste heltall.

Koden under i kodeutdrag3.4 viser kodene som utfører beregningene lagt frem i teori delkapittelet 3.1.

Kode 3.4: Teach Koder for tau på trommel

```
30 IF roundcounter ≥ 1 THEN
31     IF roundcounter <> previous_roundcounter THEN
32         full_round_rope:= full_round_rope + circumference;
33         r_total:=22.9+5.8*roundcounter;
34         circumference:=2*3.14159*r_total;
35         previous_roundcounter := roundcounter;
36     END_IF;
37 END_IF;
```

Linje 30 sjekker om en runde er fullført. Videre vil linje 31 sjekke om den nye runden er anderledes fra forrige runde for å unngå kontinuerlig oppdatering av verdien uten at ny runde er fullført. Linje 32 regner så ut hvor mye tau som nå er

på trommelen med å benytte lengden på forrige omkrets. Den nye omkretsen vil være lengden til tauet som legges på trommelen med neste fullførte runde. Linje 33 og 34 regner ut radius(r_total) og omkrets(circumference) som forklart med ligningene 3.2 og 3.3 i kapittel 3.1. 22.9 er radiusen til trommelen i mm og 5.8 er diameter på tau.

Det vil også være gunstig for et system å kunne håndtere ekstra tau innspolt på trommelen før løft av en last. Kodene som tar hensyn til tilleggstauet på trommelen er vist under i koden 3.5.

Kode 3.5: Teach Koder for tau før løft

```
38 IF torque_ref ≥ HMI_teach_torque_ref_min THEN
39     IF first_run = TRUE THEN
40         roundcounter_BL:=roundcounter;
41         circumference_BL:=circumference;
42         pulses_BL:=total_pulses;
43         roundcounter_decimals_BL:=pulses_BL/8000;
44         additional_BL:= (circumference_BL) * ...
               (roundcounter_decimals_BL - roundcounter_BL);
45         full_round_rope_BL:=full_round_rope;
46         total_rope_before_lift:=full_round_rope_BL+additional_BL;
47         first_run:= FALSE;
48     END_IF;
49 END_IF;
```

I linje 38 blir det kontrollert om den pålagte momentverdien (”torque_ref”) har nådd den nedre momentgrensen som brukeren har angitt (”HMI_teach_torque_ref_min”). Dette momentet benyttes for å skille tauets lengde før løft (”total_rope_before_lift”) fra lengden etter løft. I linje 39 blir variabelen ”first_run” brukt for å sette visse verdier kun én gang. Dette skjer når ”first_run” er satt til TRUE. I linjene 40 til 46 blir verdier som er nådd før den nedre momentgrensen (dvs. når ”torque_ref” er større enn eller lik ”HMI_teach_torque_ref_min”) lagret. ”BL” står for Before Lift", og i linje 46 blir alle disse verdiene summert for å beregne den totale lengden på tauet før løft (”total_rope_before_lift”).

Kodeutdraget 3.6 under viser hvordan arbeidsområdet blir regnet ut.

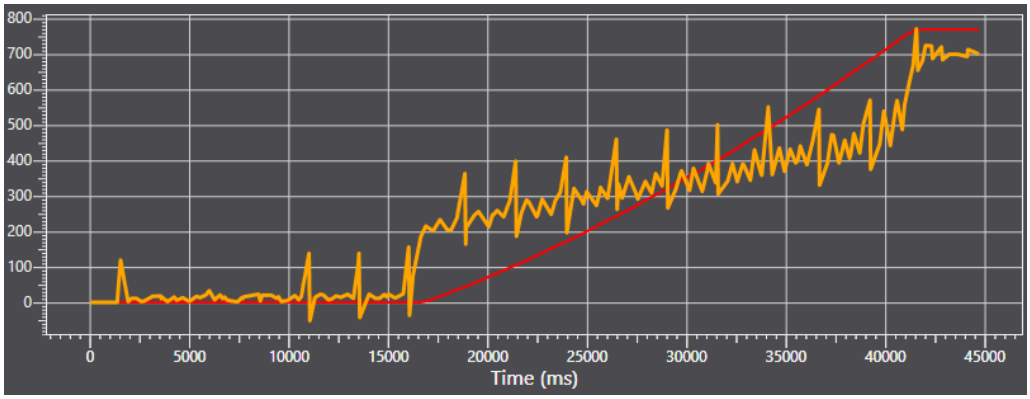
Kode 3.6: Teach Koder for totalt tau på trommel

```
50 additional_rope:= (circumference) * ...
       (roundcounter_decimals-roundcounter);
51 total_rope_drum:= full_round_rope + additional_rope;
52 rope_after_lift:=total_rope_drum - total_rope_before_lift;
53 IF torque_ref > HMI_teach_torque_ref_min AND HMI_TEACHER = TRUE ...
       THEN
54   max_pos:=(rope_after_lift/6);
55 END_IF;
56 IF torque_ref > HMI_teach_torque_ref_max THEN
57     q2a_speed:=0;
58 END_IF;
```

Koden beregner først i linje 50 mengden tau som blir spolt inn på trommelen etter at den siste fullførte runden er registrert. Dette gjøres ved å bruke den nye omkretsen og beregne prosentandelen av den nye runden(roundcounter_decimals - roundcounter). Deretter, i linje 51, beregner koden totalt mengde tau på trommelen ("total_rope_drum"). Koden skiller så i linje 52 tau som ble spolt inn på trommelen etter at løftet ble registrert av. I linjene 53 beregnes arbeidsområdet ("max_pos") etter at løftet er registrert. Arbeidsområdet blir deretter delt med 6, som er antallet trinser brukt i systemet. Til slutt, stopper koden lasten på toppen av vinsjen når en forhåndsbestemt momentverdi er nådd. Hastigheten ("q2a_speed") settes da til 0, og lasten blir hengende inntil det gis en ny instruksjon.

3.3.1 Resultat av teach-funksjonen

For å teste "teach"-funksjonen ble det utført flere forsøk. Forsøkene gikk ut på å heise opp en last fra bakken og stoppe den i toppen. Høyden ble deretter målt med tommestokk og sammenlignet med det kalibrerte arbeidsområdet fra "teach"-funksjonen. Grafen under viser hvordan moment og arbeisområdet(max_pos) beveger seg over tid(ms). En demonstrasjon av testen som er illustrert i figur 3.4, er tilgjengelig i form av en video. Du kan se videoen ved å følge YouTube-lenken https://youtu.be/5I6jVh3thk8



Figur 3.4: Arbeidsområdet("max_pos") og momentreferansen"torque ref" under teach-funksjonen

Ved å sammenligne figure 3.4 med det fysiske resultatet(vist i Youtube videoen), ser vi at avviket fra virkelighet og beregnet høyde var rundt 2 mm. Med å analysere resultatene, ser vi at det var lite forskjell på virkelig og utregnet høyde, dermed kunne vi konkludere med at funksjonen fungerte som forventet og etter hensikt.

Kapittel 4

Effekten av Q2A tuning parametere

Kodene utviklet i PLSen spiller en stor rolle i systemets dynamikk og oppførsel, men det er ikke den eneste påvirkningen på systemet. I dette kapitlet skal vi undersøke hvordan tuningparameterne påvirker systemets dynamikk og hvordan deres verdier kan bestemme systemets oppførsel. I vår oppgave vil vi imidlertid hovedsakelig benytte standardverdiene som blir bestemt gjennom autotuningsprosessen, men det er likevel viktig å analysere effekten av tuningparameterne, ettersom det vil gi verdifull innsikt i hvordan systemet fungerer og hvorfor dynamikken opptrer som den gjør.

For å utføre denne analysen vil vi benytte kodene for posisjonsregulering som senere blir presentert. Analysen av tuningparameterene vil hjelpe oss å identifisere eventuelle begrensninger og muligheter for ytterligere forbedringer.

Under i liste 4 gjentas og vises Q2-edit parameterene som vil bli endret, testet og forklart i kapittelet.

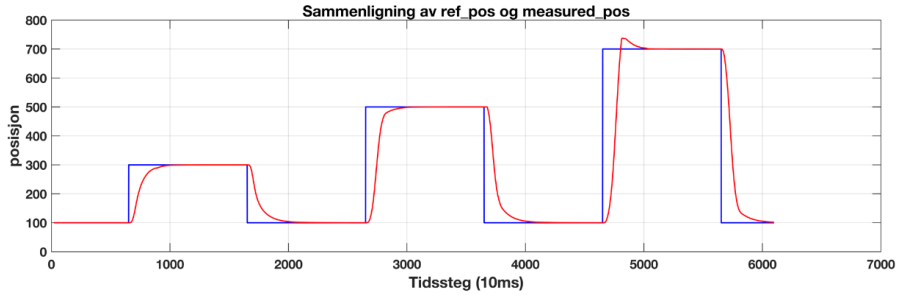
- Accel/Decel-tider (C1-01 til C1-08) styrer akselerasjon og retardasjon av motoren. Disse parameterne kan påvirke hvor raskt systemet reagerer på endringer i belastning eller hastighetskrav.
- ASR-parametere (C5-01 til C5-08) påvirker den innebygde hastighetsregulatoren.

- Jerk-parametere (C2-01 til C2-04) påvirker glattheten i akselerasjon og retardasjon.

4.1 Accel/Decel-tider (C1-01 og C1-02)

Første parametere testet var akserelerasjon og retardasjons parameterene "Accel/Decel time". Det ble utført en test med en serie av firkantpulser med økende spranghøyde for å analysere effekten av parameterne. Testen ble utført ved å starte lasten på en høyde 100mm over bakken og gi 3 økende sprang fra samme startposisjon. I testen ble kun en P-regulator benyttet.

Under i grafen 4.1 blir det vist en 3 sprangs test av en P-regulator med en Kp-verdi på 0.05. Her ble det det benyttet våre standardinnstillinger som vist i tabell 2.2.



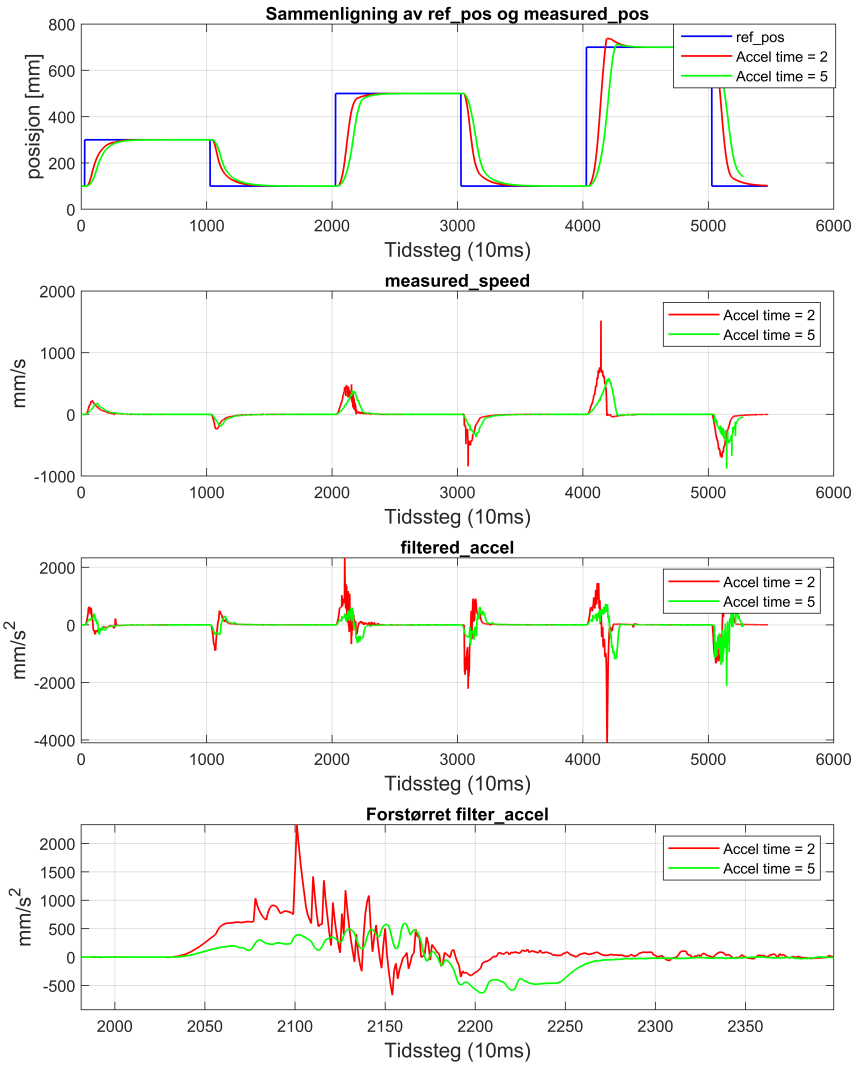
Figur 4.1: Sammenligning av referanse hastighet under en test av økende sprang.

Som vist i figur 4.1 så vil sprangresponsen endres som funskjonen av sprangets høyde. Dette skyldes tuningparameterene Accel og Decel time. Disse parameterene analyseres under.

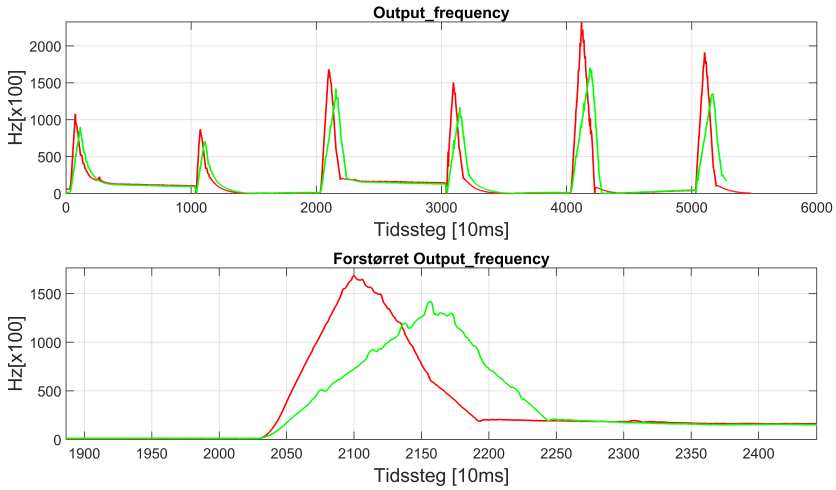
4.1.1 Accel time

Den tekniske manualen til vår Q2A frekvensomformer, viser at accel time-parameteren regulerer tiden det tar for motoren å akselerere fra null til maksimal output-frekvens [10, s. 585]. For å undersøke effekten denne parameteren har på systemet

vårt, sammenligner vi hva en endring fra accel time på 2s til accel time på 5s gjør med responsen. Vi gjennomfører samme test og resultatet er vist i Figur 4.2 og 4.3 der det rød signalet er testen med standardverdier og det grønne er med en accel time på 5.



Figur 4.2



Figur 4.3

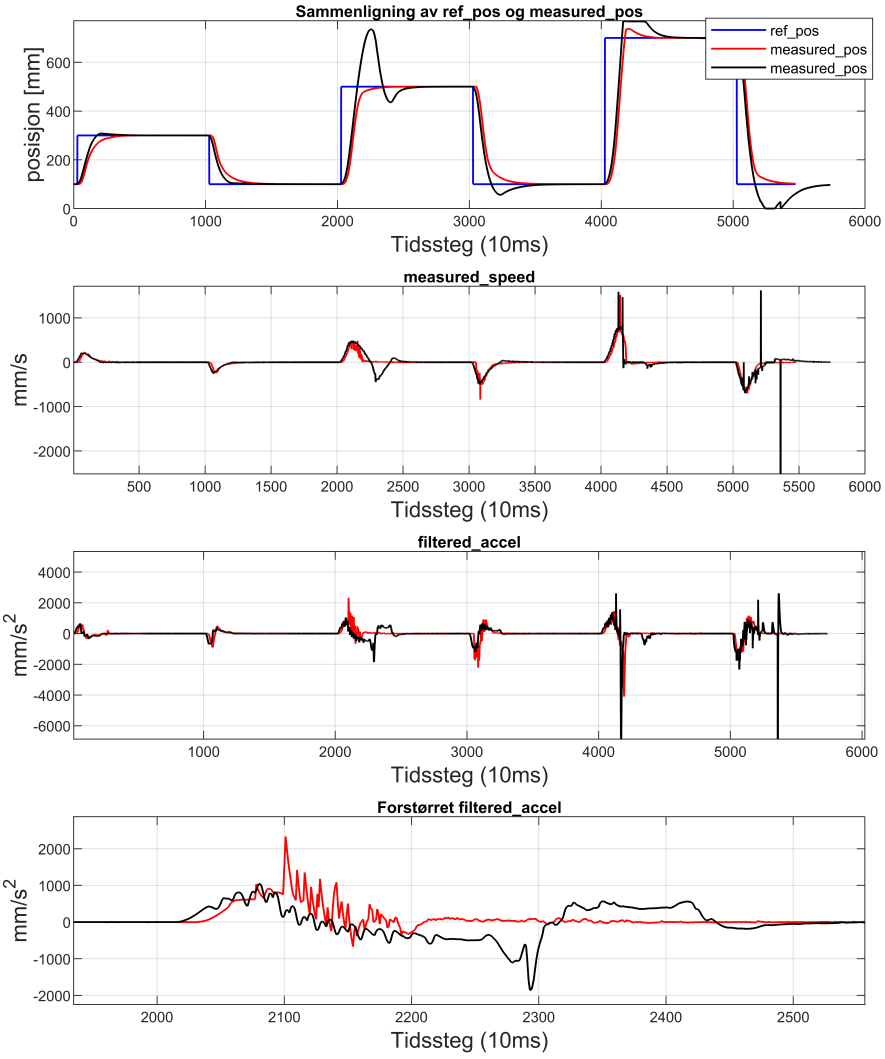
Ved å sammenligne grafene for 'filtered accel' i figur 4.2, ser vi at en accel time-verdi på 5s reduserer akselerasjonen. Dette resulterer i at motoren bruker lengre tid på å nå topphastigheten som vist i nederst i figur 4.3 til output frequency. Grunnen til at dette skjer er fordi vi har en maks utgangsfrekvens på 50 Hz og en P-regulator(Kp=0,05), som under et stort sprang, er kraftig nok til å gå fra 0 til 50 Hz før det har gått 5s. Selvom vår regulator ikke vil nå den maksimale utgangsfrekvensen innen for et sprang, så vil accel time fortsatt dempe signalet fordi akselerasjonen er for høy.

Vi ser i figur 4.3 at en høyere accel time-verdi gir en mer gradvis økning i hastigheten. Dette kan resultere i jevnere målinger, reduserte vibrasjoner og mindre mekanisk stress. På den annen side gir en lavere accel time-verdi raskere akselerasjon og responstid.

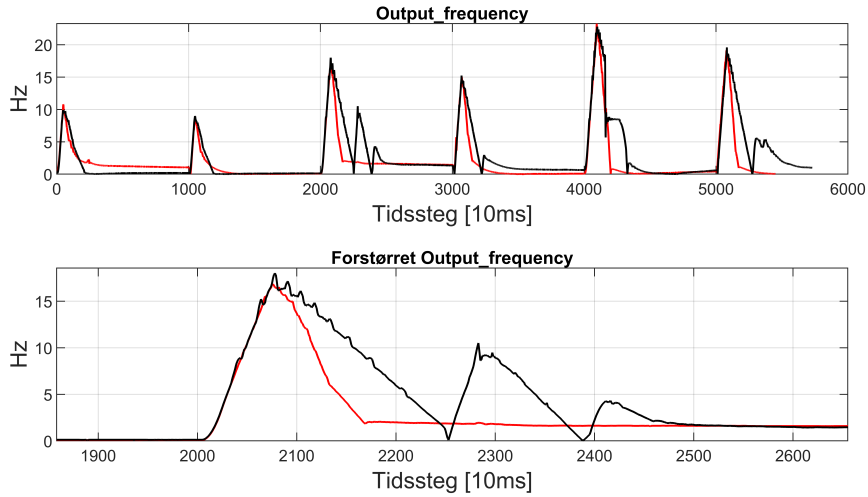
4.1.2 Decel time

I henhold til databladet styrer Decel Time-parameteren tiden det tar for motoren å redusere hastigheten fra maksimal utgangsfrekvens til null [10, s. 585]. For å se innvirkningen av denne parameteren på vårt system, sammenligner vi effektene av å endre fra decel time på 2 til en decel time på 5. Vi utfører den samme testen, og resultatene er presentert i figurene nedenfor (4.4 og 4.5). Det rød signalet er

testen med standardinnstillinger og det sorte signalet representerer testen med en Decel time på 5.



Figur 4.4



Figur 4.5

Ved å sammenligne grafene for "filtered accel" i figur 4.4, ser vi at en decel time-verdi på 5 reduserer retardasjonen. I vårt system resulterer dette i en betydelig oversving, ettersom systemet ikke er i stand til å redusere hastigheten før referanseposisjonen.

Oversvingen skyldes at når vår maksimale hastighet blir redusert av regulatoren, oppstår det en retardasjon, kraftig nok til å redusere frekvensen fra 50 Hz til null på under 5 sekunder. Decel Time-parameteren vil derfor forhindre denne raske retardasjonen, noe som resulterer i at systemet ikke klarer å bremse ned i tide før det når referanseposisjonen.

En høyere decel time-verdi gir en mer gradvis reduksjon i hastighet, noe som kan resultere i jevnere målinger, reduserte vibrasjoner og mindre mekanisk stress. På den andre siden gir en lavere decel time-verdi raskere retardasjon og responstid.

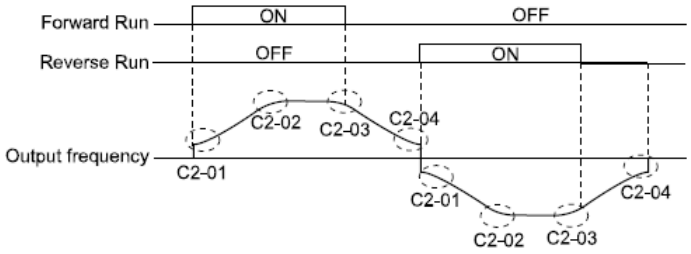
4.1.3 Oppsummering av accel og decel parameterene

Etter analyse av accel Time og decel time-parametrene, kom vi til den konklusjon at disse parametrene innfører visse begrensninger i systemet. Dette kan føre til ulineære oppførsler når regulatoren forsøker å overstige disse begrensningene.

4.2 Jerk-parametere (C2-01 til C2-04)

I Q2-edit refererer jerk-parametrene til endringsraten av akselerasjonen/retardasjonen i system. De fire Jerk-parametrene i Q2-edit lar oss justere og kontrollere hvor raskt akselerasjonen(accel) og retardasjonen(decel) skal endres i ulike faser av systemet.

- C2-01 Jerk@Start of Accel: Setter jerktiden ved starten av akselerasjonen. Standardverdien er 0,20 s
- C2-02 Jerk@End of Accel: Setter jerktiden ved slutten av akselerasjonen. Standardverdien er 0,20 s
- C2-03 Jerk@Start of Decel: Setter jerktiden ved starten av retardasjonen. Standardverdien er 0,20 s
- C2-04 Jerk@End of Decel: Setter jerktiden ved slutten av retardasjonen. Standardverdien er 0,00 s



Figur 4.6: S-kurve timing diagram for initialisering av jerk-parameterne

Ettersom jerk-parameterne også spiller en sentral rolle i systemets akselerasjonsprofil, vil den totale akselerasjonstiden bli bestemt ut fra følgende formel:

$$\textit{Acceleration time} = \textit{Selected acceleration time} + \textit{Jerk Accel} \tag{4.1}$$

$$\textit{Jerk Accel} = \frac{\textit{C2-01} + \textit{C2-02}}{2} = 0,2s \tag{4.2}$$

Ved å benytte en "jerk accel" verdi på 0,2 sekunder, sikres det at akselerasjonen ikke umiddelbart øker til sin maksimale verdi, men heller stiger gradvis over en tidsperiode på 0,2 sekunder. Denne innstillingen kan hjelpe med å minimere eventuelle plutselige belastninger eller støt i systemet ved å sikre en mer jevn overgang til toppakselerasjon.

$$\textit{Deceleration time} = \textit{Selected deceleration time} + \textit{Jerk Decel} \tag{4.3}$$

$$\textit{Jerk Decel} = \frac{\textit{C2-03} + \textit{C2-04}}{2} = 0,1s \tag{4.4}$$

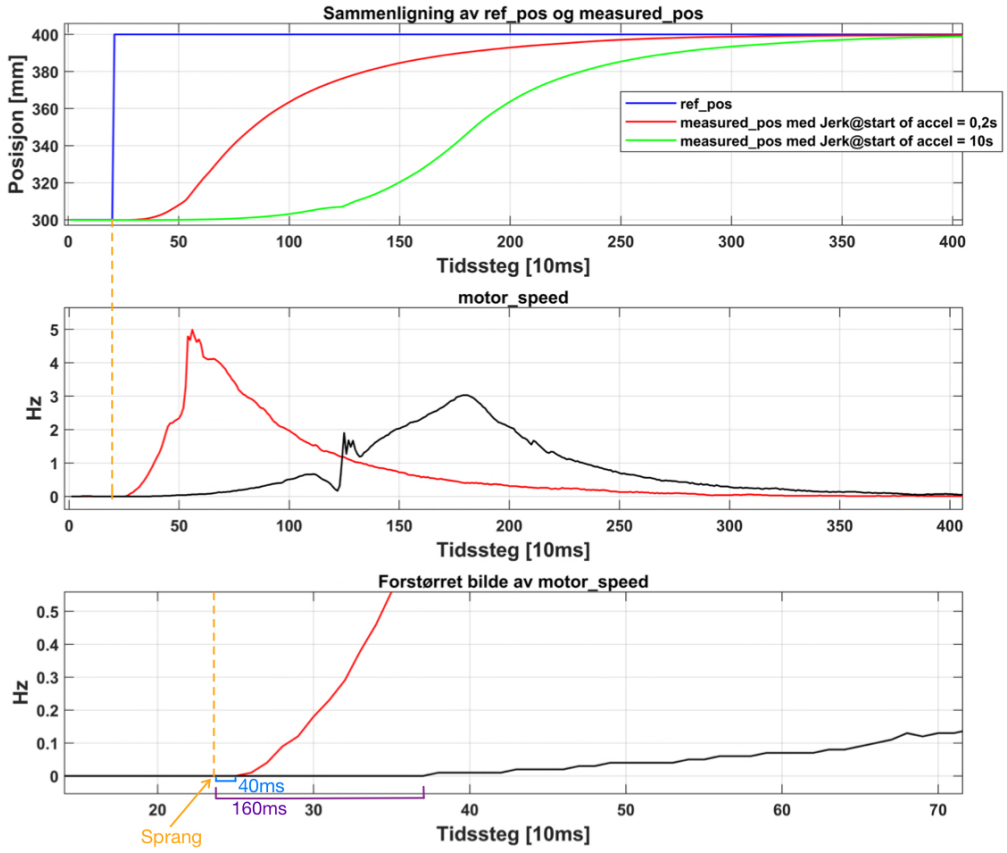
Ved å benytte en "jerk decel" verdi på 0,1

Ved å sette en "jerk decel" verdi på 0,1 sekunder, sikrer vi at systemets akselerasjon ikke umiddelbart faller til null når retardasjonen iverksettes, men i stedet reduseres gradvis over en periode på 0,1 sekunder. Denne innstillingen hjelper med å sørge for en mer jevn overgang fra maksimal akselerasjon til hviletilstand.

Siden vi allerede har analysert effekten av accel og decel time, så kjørte vi ikke tester for å se hvordan alle jerk-parameterne påvirket systemet, men tar med oss at disse også påvirker akselerasjons- og retardasjonstiden. Det ble derimot valgt å sjekke om endringen av Jerk@Start of Accel(C2-01) ville påvirke systemets tidsforsinkelse.

4.2.1 Jerk@Start of Accel(C2-01)

For å undersøke effekten av Jerk@Start of Accel-parameteren, utførte vi en enkel sprangtest der loddet ble flyttet fra 300 mm til 400 mm over bakken. Formålet med testen var å evaluere om parameteren påvirker systemets tidsforsinkelse. To tester ble utført, én med standardverdien på 0,20 s og én med en verdi på 10 s. Resultatene fra testene er presentert i figuren 4.7:



Figur 4.7

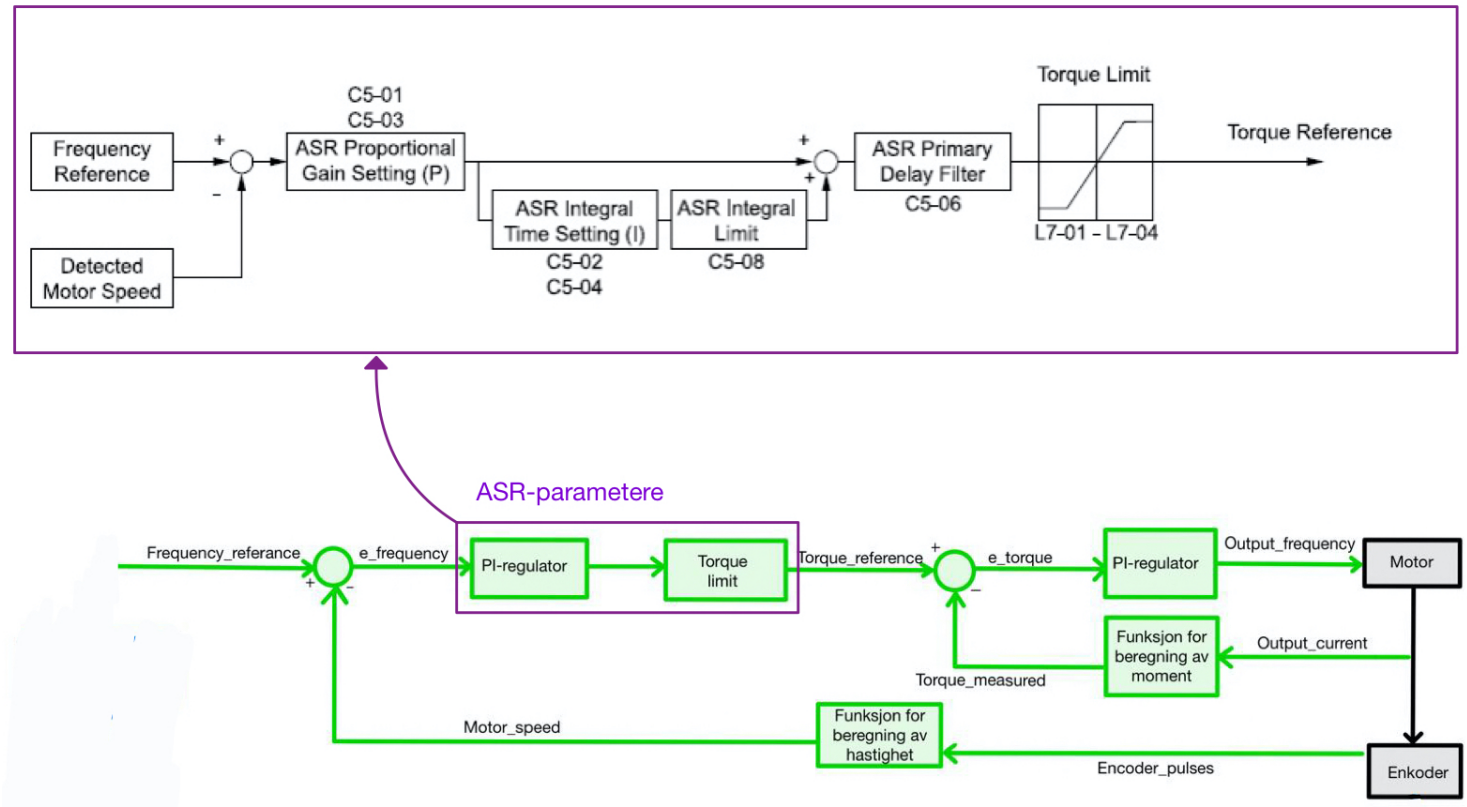
Som illustrert i den grønne linjen øverst figur 4.7 så ser vi at systemet vil som forventet reagere tregere med en høyere parameter verdi. Det viktigste fra disse testene var derimot å sjekk om motor_speed ville få en større tidsforsinkelse ved å øke parameterverdien. Som vist i figuren så ser vi at en høyere verdi vil gi mer tidsforsinkelse. Ved hjelp av Sysmac sin DataTracing og cursor verktøy så kom vi fram til:

- Jerk@start of Accel = 0,20s → motor_speed tidsforsinkelse = 40ms
- Jerk@start of Accel = 10s → motor_speed tidsforsinkelse = 160ms

4.3 ASR-parametere (C5-01 til C5-08)

Andre parametere testet var ASR-parameterne. ASR-parameterne (Automatic speed regulation) i en frekvensomformer er en samling innstillinger som brukes til å opprettholde en konstant motorhastighet under varierende belastninger. ASR fungerer ved å tilpasse "torque_reference" basert på forskjellen mellom "frequency_reference" og "motor_speed", slik at den kan reagere på og kompensere for endringer i belastning, samtidig som den reduserer svingninger og vibrasjoner.

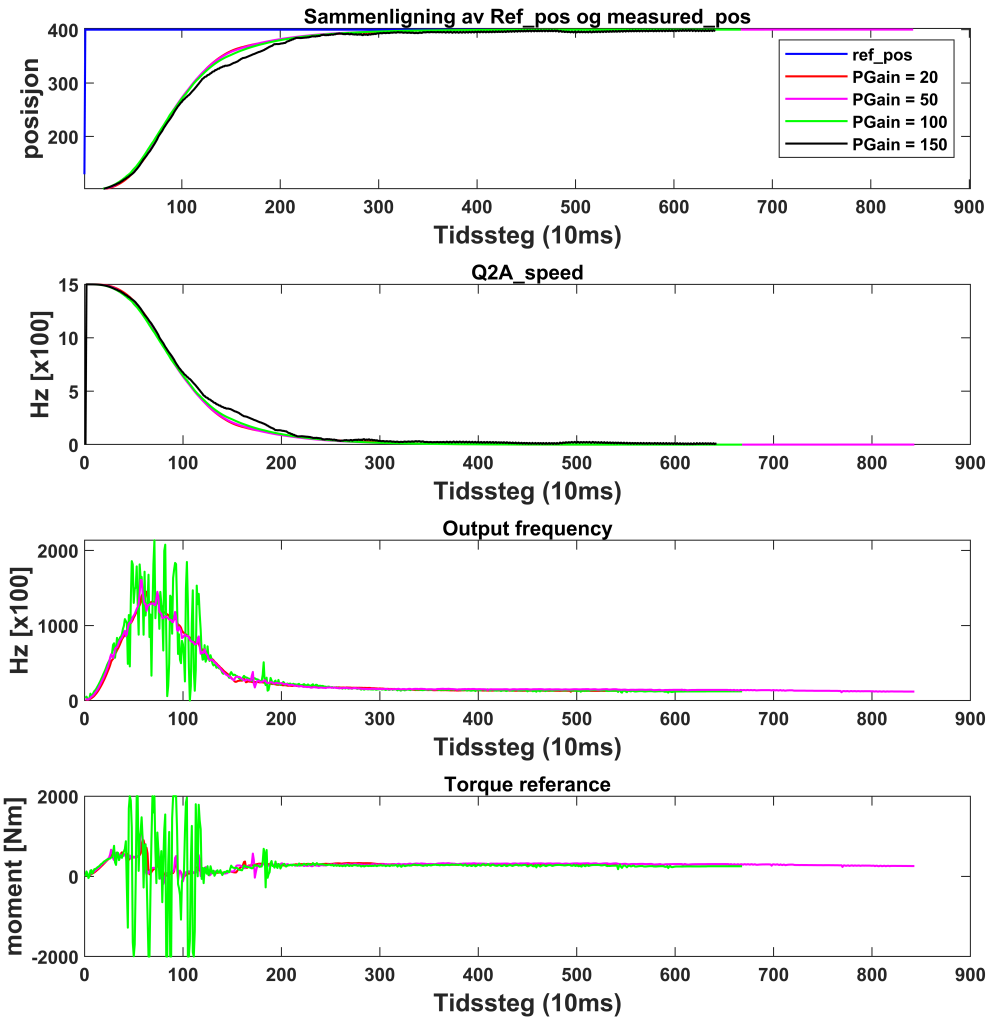
Figuren 4.8 nedenfor gir en visuell fremstilling av hvor ASR-parametrene blir brukt i CLVC-blokkdiagrammet. Den minste lilla boksen markerer det spesifikke området der disse parametrene blir benyttet.



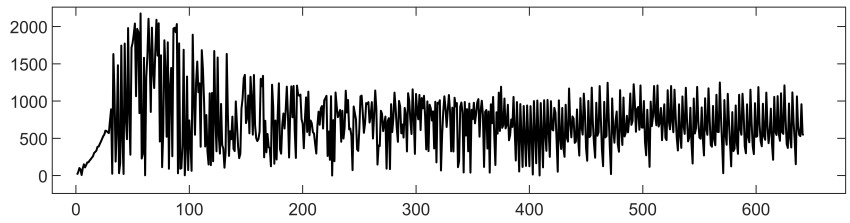
Figur 4.8: Blokkdiagram over CLVC som viser hvor ASR parameterene har sin funskjon

4.3.1 ASR PGain

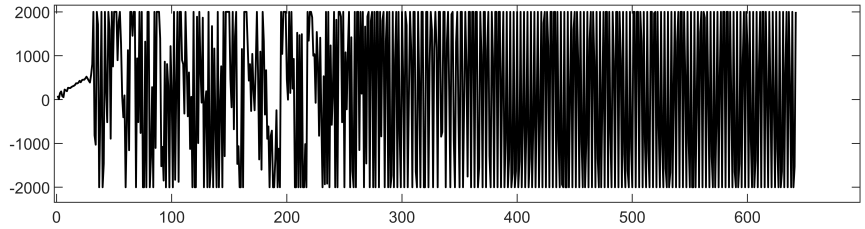
ASR PGain-parameteren definerer den proporsjonale forsterkningen i den integrerte PI-regulatoren. Denne forsterkningen er en koeffisient som styrer systemets respons på forskjeller mellom referanse- og faktiske verdier. I Q2-edit går PGain-verdien fra 0 til 300, med en standardinnstilling på 20. For å evaluere effekten av endringer i denne parameteren, gjennomførte vi en enkel sprangtest med en P-regulator (kp = 0,05). I testen ble det løftet en vekt fra 100 mm til 400 mm over bakken. Vi utførte totalt fire tester, hver med en forskjellig PGain-verdi - 20 (standard), 50, 100 og 150. Resultatet fra testene er vist under i figurene 4.9, 4.10og 4.11


Sammenligning av Ref_pos og measured_pos
posisjon
Tidssteg (10ms)
ref_pos
PGain = 20
PGain = 50
PGain = 100
PGain = 150
Q2A_speed
Hz [x100]
Tidssteg (10ms)
Output frequency
Hz [x100]
Tidssteg (10ms)
Torque referance
moment [Nm]
Tidssteg (10ms)

Figur 4.9



Figur 4.10: PGain = 150, output_frequency



Figur 4.11: PGain = 150, torque_ref

Figurene over illusterer responsen til systemet ved 4 forskjellige tester. Figurene 4.10 og 4.11 ble plottet i eget plot fordi signalet var for støyete til å plotte sammen med de andre. I plottene så viser:

- rød linje → PGain = 20,
- Lilla linje → PGain = 50,
- Grønn linje → PGain = 100,
- Sort linje → PGain = 150,

Ved å analysere resultatene fra de fire testene, er det tydelig at endringer i PGain-parameteren har en betydelig innvirkning på torque_referanse. I denne testen observerer vi kun grensen der PGain vil gjøre systemet vårt ustabilt og tilføre vibrasjoner uten ekstra vektbelastning. En for lav PGain-verdi vil resultere i en mykere og langsommere respons, noe som kan redusere systemets følsomhet for støy og forstyrrelser. På den andre siden kan en for lav PGain-verdi føre til redusert responstid og nøyaktighet i reguleringen.

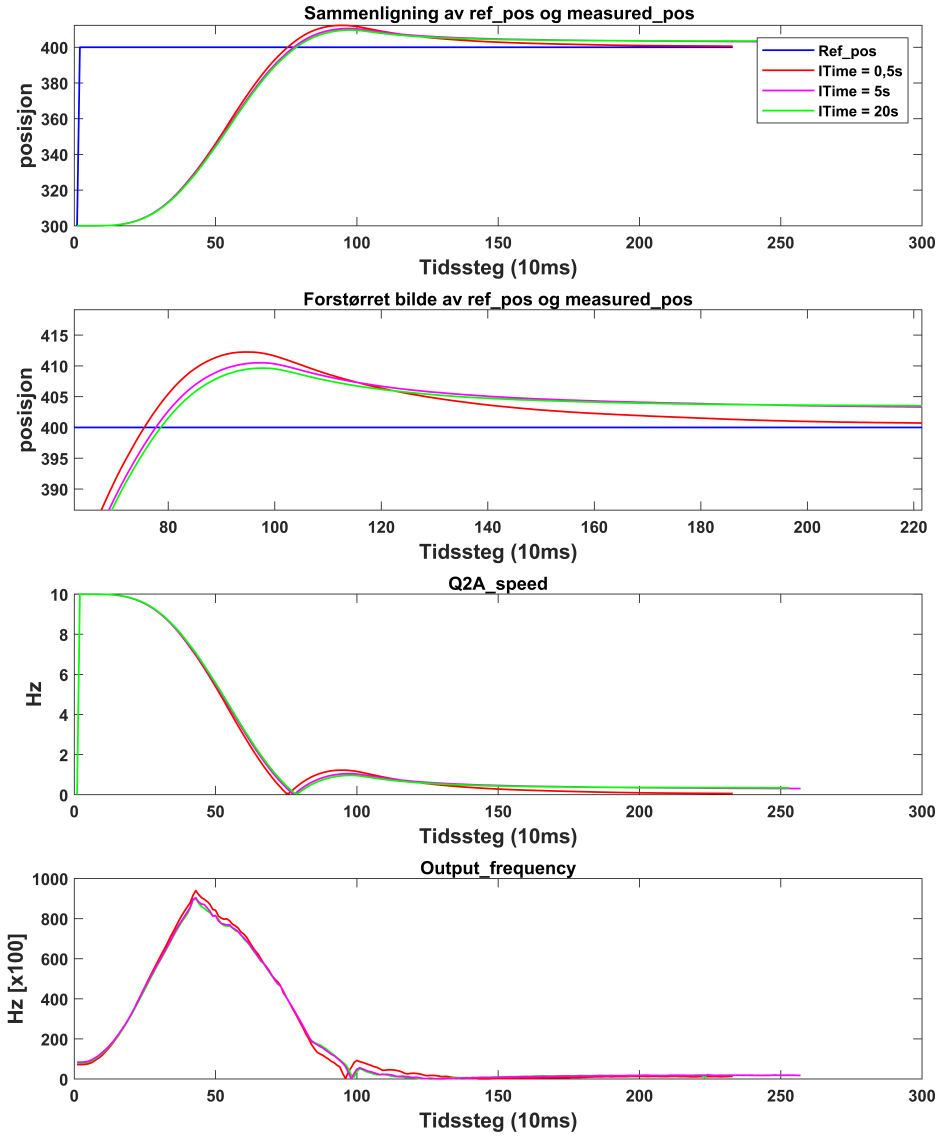
En høyere PGain-verdi vil gjøre at systemet kan reagere raskere og mer aggressivt på større belastninger. Imidlertid vil en for høy PGain-verdi føre til ustabilitet, overdreven oscillasjon og vibrasjoner i systemet. Da torque_reference signalet blir kraftig forsterket under høyere PGain-verdi, vil også Output_frequency bli kraftig forsterket. Dette fører til at motoren vil hakke ved for høy verdi.

4.3.2 ASR ITime

ITime-parameteren i Q2-edit står for "Integral time". Den brukes til å justere den integrerte tidskonstanten for ASR-kontrolleren. Den integrerte tidskonstanten påvirker responsen til systemet ved å kontrollere hvor raskt integratoren vil reagere på svingninger og eventuelle feil i stabil tilstand. For å analysere effekten av endre på denne parameteren så ble det gjennomført en enkelt sprang test med en P-regulator(kp = 0,1). Sprangtesten ble utført ved å heve loddet fra en høyde på 300 mm til 400 mm over bakken. Det ble utført 3 tester med 3 forskjellige ITime verdier, 0.5(standard), 5 og 20.

Under i delfigurene 4.12 illusteres responsen til systemet i de 3 forskjellige testene. I figurene så viser:

- rød linje → ITime = 0,5,
- Lilla linje → ITime = 5,
- Grønn linje → ITime = 20,


Sammenligning av ref_pos og measured_pos
posisjon
Tidssteg (10ms)
Ref_pos
ITime = 0,5s
ITime = 5s
ITime = 20s
Forstørret bilde av ref_pos og measured_pos
posisjon
Tidssteg (10ms)
Q2A_speed
Hz
Tidssteg (10ms)
Output_frequency
Hz [x100]
Tidssteg (10ms)

Figur 4.12

ITime-reguleringen avgjør hastigheten på oppbygningen av integreringsprosessen i systemet. Ved å analysere resultatene fra de tre forsøkene, kan vi observere i de to første delfigurene at en redusert ITime-verdi akselererer integreringshandlingen. Dette fører til en raskere reduksjon av feilen i steady-state. Imidlertid kan en lavere verdi også resultere i større oversving, noe som potensielt kan øke risikoen for oscillasjoner og ustabilitet i systemet.

Å øke ITime-verdien vil gjøre den integrerende aksjonen tregere, noe som resultere i mindre risiko for oscillasjoner og bedre stabilitet, men som vi ser i det forstørrede bilde av measured_pos, så vil det ta lengre tid for systemet å nå referanse posisjonen.

4.3.3 ASR Delay Time

ASR Delay Time (C5-06) angir tidsforsinkelsen mellom to påfølgende ASR operasjoner. Denne parameteren lar systemet vente en viss tid før det prøver å justere torque_reference. Denne er nyttig for å forhindre overdreven justering eller for å gi systemet tid til å stabilisere seg etter en endring i driftsforholdene.

Standardverdien for parameteren er 4 ms. Ifølge databladet er det normalt sett ikke nødvendig å justere denne verdien. Det ble imidlertid gjennomført en test der vi endret denne parameteren, men vi observerte ingen effektendring og resultatet forble det samme.

4.3.4 ASR Intergral limit

ASR integral limit setter den øvre grensa for ASR som en prosent(%) av rate loaden. Ifølge databladet kan denne parameteren ha en verdi fra 0-400%, med en standardverdi på 400%. Denne verdien blir satt etter autotuning og blir bestemt ut ifra belastningen på motoren. Under testing ble det oppdaget at med en verdi mindre enn 20% så vil systemet risikere å kortslutte og miste kommunikasjonen mellom PLSen og frekvensomformeren. Grunnen til at dette skjer er ukjent.

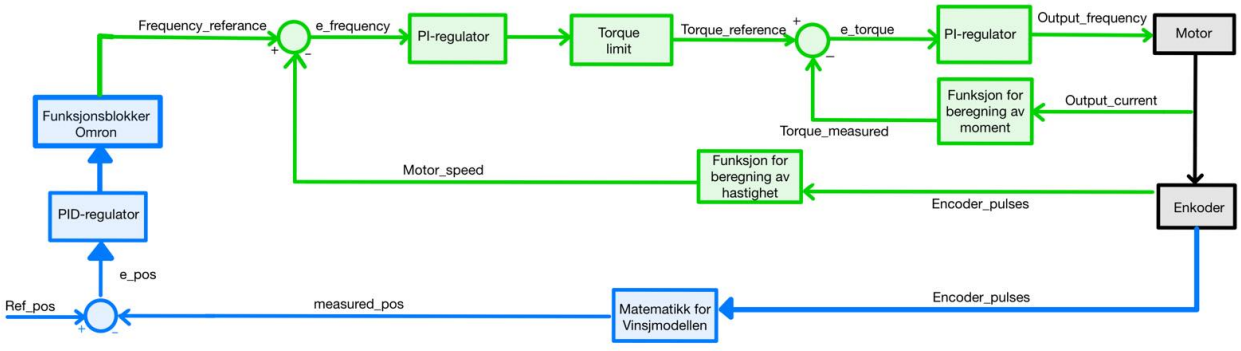
4.4 Konklusjon av tuningparametere

Etter å ha undersøkt flere av tuningparameterne i Q2A-enheten, har vi observert hvordan noen av dem kan skape begrensninger for regulatoren vår og påvirke dens ytelse. Vi har også sett hvordan disse parameterne endrer systemets dynamikk og dette har bidratt til å gi oss økt forståelse til hvorfor systemet oppfører seg slik som det gjør. I vedlegg A presenteres en tabell som viser hvordan man kan benytte tuningparametere for å tune responsen etter det ønskede formålet. Videre i oppgaven vil vi derimot forklare hvordan vi har anvendt PLS-en for å regulere posisjon, hastighet og moment i systemet rundt Q2A-ens standardinnstillinger.

Kapittel 5

Posisjonsregulering av last

I dette kapittelet vil vi introdusere problemstillingen rundt posisjonsregulering, samt presentere teori, løsninger og resultater. For at en bestemt høyde over bakkenivået blir nådd og opprettholdt, selv når det forekommer variasjoner i lasten, har vi implementert en metode for posisjonsregulering. Denne metoden tar hensyn til endringer i lasten og justerer deretter systemets respons for å sikre at den ønskede høyden opprettholdes kontinuerlig. Dette innebærer å bruke matematiske modeller av systemet for å justere posisjonen til lasten. Blokkskjemaet i figur 5.1 illustrerer hvordan denne posisjonsreguleringen fungerer.



Figur 5.1: Blokkskjema over posisjonsregulering. Den blå delen viser styringen i PLSen, den grønnedelen viser styringen i Q2Aen og den grå delen viser maskinvaren.

Pulsene som genereres av enkoderen blir registrert av frekvensomformeren og lagret i variabelen "encoder_pulses". Disse pulsene brukes i en matematisk utregning for å bestemme høyden over bakken (measured_pos). Referanseposisjonen (ref_pos) er den ønskede posisjonen i millimeter. Ref_pos blir satt av brukeren via HMI skjermen. Reguleringsavviket (e_pos) beregnes som differansen mellom referanseposisjonen(ref_pos) og den faktiske målte posisjonen (measured_pos). Dette avviket brukes som input i en PID-regulator som via funksjonsblokkene fra Omron sender singalet "frequency_referance" til frekvensomformeren. Signalverdien blir deretter regulert i en lukket reguleringssløyfe med vektorkontroll, som er forklart nærmere i kapittel 2.2.1. Basert på denne reguleringen, beregnes pådraget (output_frequency) i Hz.

Det ble også utforsket ytelsen til en P-regulator og vurdert om det var nødvendig å benytte en PID-regulator for posisjonsregulering. Vi vil også teste alternative metoder for tuning, som Skogestad-metoden, for å undersøke om vi kan oppnå bedre respons.

5.1 Teori

Oppgaven til PID-regulatoren er å regulere avviket mellom referanseposisjon og den målte posisjonen. Regulatoren beregner et pådrag fra avviket som settes ut til motoren. Motoren vil kjøre og justere avviket. Pådraggssignalet som beregnes av PID-regulatoren vises under i likning 5.1

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \frac{de(t)}{dt} \tag{5.1}$$

Under i listen 5.1 er de ulike leddene representerte i separate likninger.

- P-ledd: $K_p \cdot e(t)$
- I-ledd: $K_i \cdot \int_0^t e(t)dt$
- D-ledd: $K_d \cdot \frac{de(t)}{dt}$

P-leddet i regulatoren bidrar med en komponent som er proporsjonal med avviket,

mens I-leddet fungerer som et minne for alle målte avvik som akkumuleres over tid. D-leddet gir et bidrag basert på raske endringer i avviket.

I likning 5.1 benyttes justeringsfaktorer Kp for P-leddet, Ki for I-leddet og Kd for D-leddet. Disse faktorene multipliseres med de respektive bidragene. Størrelsen på disse faktorene bestemmes senere ved hjelp av "prøv og feil"-tuningsmetoden og "Skogestads"-tuning metode.

For å konstruere I og D ledd ble numerisk integrasjon og numerisk derivasjon benyttet. Likningen for numerisk integrasjon for I-ledd er vist i likningen 5.2.

$$u_I(k) = u_I(k-1) + e(t) \cdot T_s \tag{5.2}$$

Hvor variablene i likning 5.2 er:

- $u_I(k)$ - Det nye I-leddet
- $u_I(k-1)$ - Det gamle I-leddet
- $e(t)$ - Avviket i posisjon
- T_s - Tidssteget i sekunder

Den numeriske derivasjonen som ble brukt for å lage D-leddet er vist i likning 5.3

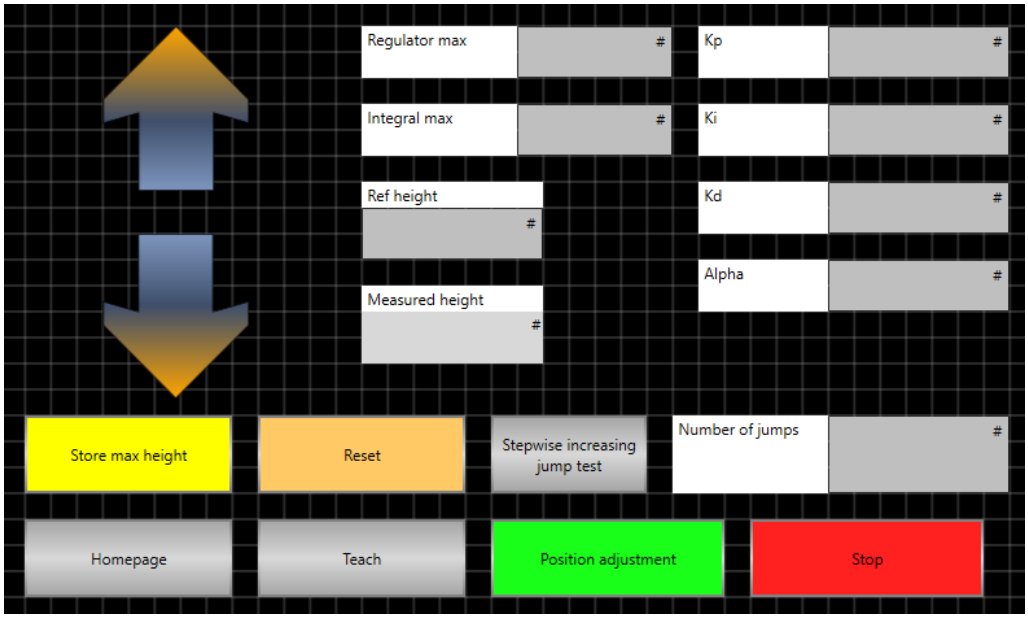
$$u_D(k) = \frac{e(k) - e(k-1)}{T_s} \tag{5.3}$$

Hvor variablene i likning 4.3 er:

- $u_D(t)$ - D-leddet
- $e(k)$ - Avviket i posisjon
- $e(k)$ - Det gamle avviket i posisjon
- T_s - Tidssteget i sekunder

5.2 Kodene for posisjonsregulering

I denne delen av kapittelet skal vi forklare hvordan kodene for posisjonsregulering er bygd opp og hvordan HMI-skjermen kan benyttes. For å starte posisjonsreguleringen må det først gjennomføres en vellykket "teach" av systemet. De globale variablene fra "teach"-programmet vil bli lagret og gjort tilgjengelig i posisjonsreguleringsprogrammet.



Figur 5.2: HMI-skjermen for posisjonsregulering

På HMI-skjermen er det lagt inn 8 DataEdit-vinduer. Disse gjør det enkelt for brukeren å endre PID-koeffisientene, filterkonstanten (alpha), samt også sette begrensninger for I-leddet og regulatorens output. Endring av disse vil bli forklart senere i kapittelet. Det ble også lagt inn en test (Step wise increasing jump test) som ble brukt til å gjennomføre tester.

HMI-skjermen har 4 fargede knapper: Position Adjustment, Stop, reset og store max height. Funksjonene til disse er beskrevet nedenfor:

- Store max height → Lagrer alle verdiene til de globale variablene fra Teach-funksjonen

- Reset → Tilbakestiller alle verdiene til de som ble lagre fra "lagre maks høyde"-knappen. Nullstiller også Integral-ledd og regulatoren.
- Position adjustment → Starter posisjonsreguleringen og loddet blir senket/-løftet til ønsket høyde
- Stop → Nødstopp. Setter q2a_speed = 0

Kode som ble utviklet tillot brukeren å angi ønskede parameterverdier. Videre ble det implementert funksjonalitet for PID-regulering. Lastens høyde ble konsekvent oppdatert basert på målte enkoderpulser. Metoden for å beregne høyden er den samme som brukes for beregning av arbeidsområdet, noe som er forklart i kapittelet om ”teach”.

Den første regulatoren implementert var P-regulatoren. Kodene for dette er vist under i kodeudraget 5.1.

Kode 5.1: P-ledd i PID

```
36  ref_pos := HMI_ref_pos;
37  kp:=HMI_kp_pos;
38  ki:=HMI_ki_pos;
39  kd:=HMI_kd_pos;
40  e_pos := ref_pos - measured_pos;
41
42  P:=LREAL_TO_REAL(Kp * e_pos) ;
```

Kode 5.1: P-ledd i PID:
Linjene 36 til 39 gir bruker mulighet til å velge ønsket posisjon(ref_pos), ”kp”, ”ki” og ”kd” verdier på HMI-skjermen vist i fig 6.2. I linje 40 beregnes ”e_pos” som er differansen mellom ønsket posisjon (ref_pos) og målt posisjon (measured_pos). Denne differansen vil bli brukt til å justere styringen for å nå ønsket posisjon. I linje 42 beregnes P-leddet i PID regulatoren. Her blir avviket (e_pos) mellom ønsket posisjon(ref_pos) og målt posisjon(meas_pos) ganget med kp-verdien valgt av bruker.

Videre steg var implementering av I og D ledd for pid kontrolleren. For å få til dette måtte et tidsskritt bli bestemt. Koden for tidsskrittet er vist under i kodeutdraget 5.2

Kode 5.2: Tidsskritt

```
43      time_now :=GetTime();
44      ts:=SUB_DT_DT(time_now, time_old);
45      time_old := time_now;
46      ts_nano :=TimeToNanoSec(ts);
47      ts_nano_real :=LINT_TO_REAL(ts_nano) / 1000000000;
```

Kode 5.2: Tidsskritt
I linje 43 blir GetTime() funksjonen kalt. Datoen og klokkeslettet på nåværende tidspunkt lagres i variabelen "time_now". I linje 44 trekker en funksjon SUB_DT_DT() verdien til variabelen "time_old" fra nåværende tidspunkt time_now og beregner differansen mellom de to tidspunktene, resultatet blir lagret i variabelen tidsskritt("ts"). I linje 46 konverterer TimeToNanoSec() funksjonen tidsintervallet, ts, til nanosekunder og lagrer resultatet i variabelen "ts_nano". I linje 47 konverterer LINT_TO_REAL() funksjonen "ts_nano" fra heltallstype til flyttallstype og dividerer resultatet med 1 milliard for å konvertere nanosekunder til sekunder. Resultatet lagres i variabelen "ts_nano_real", som representerer tidsintervallet i sekunder mellom de to siste registrerte tidsstemplene. Fordelen med å gjøre om tidsskrittet til nanosekunder som vist i koden er for å øke nøyaktigheten og presisjonen i beregningene.

Med koder for tidsskrittet var det mulig å implementere koder for Integralleddet. Kodene under i kodeudraget 5.3 ble benyttet for I-leddet i PID-regulatoren.

Kode 5.3: I-ledd i PID

```
48 integral:=previous_integral+ki*(e_pos*ts_nano_real);
49 IF integral ≥ HMI_integral_maks THEN;
50     integral:=previous_integral;
51     ELSIF integral ≤ -HMI_integral_maks THEN;
52     integral:=previous_integral;
53 END_IF;
54
55 I:=LREAL_TO_REAL(integral);
```

Kode 5.3: I-ledd i PID:
Kodene over i 5.3 utgjør integratorleddet i en PID-kontrolleren, der "e_pos" representerer feilen mellom den ønskede verdien(ref_pos) og den faktiske verdien målt i kontrolleren (meas_pos). I koden er "ts_nano_real" tidsstegene som kontrolleren opererer med oppgitt i sekunder.

I linje 48 beregnes integralverdien ved å legge til den nåværende feilen multiplisert

med tidsstegene og koefiisienten "ki" valgt av bruker. Dette oppdaterer integratorens verdien med hensyn til tiden som har gått siden forrige oppdatering. I linjene 49-52 blir integratorens verdi begrenset av en øvre og nedre grense, gitt av en verdi brukeren velger på HMI-skjermen("HMI_integral_maks"). Hvis verdien overskrider disse grensene, blir integratorens verdi satt tilbake til forrige verdi, som er lagret i variabelen "previous_integral". Dette er en viktig forsiktighetsregel for å unngå overdreven akkumulering av integrasjonsfeil, noe som kan føre til overstyrt kontroll eller ustabilitet i systemet. I linje 55 beregnes koden for utgangen av I-leddet.

Videre ble det lagd koder for D-regulatoren i PID kontrolleren. Kodene for D-leddet er vist under i kodeudraget 5.3

Kode 5.4: D-ledd i PID

```
56      e_pos_f:=(alpha*e_pos) + (1-alpha)*e_pos_f_old;
57      alpha := HMI_alpha;
58      e_pos_derived:=(e_pos_f-e_pos_f_old)/ts_nano_real;
59      e_pos_f_old:=e_pos_f;
60      D:=LREAL_TO_REAL(kd * e_pos_derived);
```

Kode 5.4: D-ledd i PID:
I linje 56 blir et IIR-filter benyttet for å beregne en filtrert verdi("e_pos_f") av feilen mellom ønsket verdi ("ref_pos") og den faktiske verdien som kontrolleren har ("meas_pos"). I linje 57 oppdateres verdien av parameteren "alphafra en HMI-skjerm, slik at brukeren kan justere filtreringsparametret. I linje 58 beregner koden den deriverte av "e_pos_f" ved å trekke forrige filtrerte feilverdi ("e_pos_f_old") fra den nåværende filtrerte verdien ("e_pos_f"), og deretter dividere dette med tidsstegene mellom oppdateringer ("ts_nano_real"). Dette gir en indikasjon på hvor raskt feilen endres i systemet. I linje 60 beregner koden utgangen av D-leddet, "D", ved å multiplisere verdien av den deriverte feilen, "e_pos_derived", med en konstant "kd", som er en parameter som brukeren kan justere fra HMI-grensesnittet. Når kodene for P,I og D leddet i PID-regulatoren var implementert summerte vi verdiene i en "regulator" variabel. Koden for regulatoren er vist under i kodeutdrag 5.5.

Kode 5.5: Regulator

```
61  regulator := P + I + D;
```

Kode 5.5: Regulator:
Med en regulator kunne kodene for kjøring under posisjonsregulering bli implementert, styrt av hastigheten bestemt av regulator variabelen. Koden for kjøringen er vist under i kodeudtdraget 5.6.

Kode 5.6: Kjøring under posisjonsregulering

```
62 IF regulator > HMI_regulator_maks THEN
63     regulator := HMI_regulator_maks;
64 ELSIF regulator <-HMI_regulator_maks  THEN
65     regulator := -HMI_regulator_maks;
66 END_IF;
67 IF regulator > 0 THEN
68     q2a_forward := TRUE;
69     q2a_reverse := FALSE;
70     q2a_speed := ABS(regulator);
71 ELSIF regulator < 0 THEN
72     q2a_forward := FALSE;
73     q2a_reverse := TRUE;
74     q2a_speed := ABS(regulator);
75 ELSE
76     q2a_forward := FALSE;
77     q2a_reverse := FALSE;
78     q2a_speed := 0;
79 END_IF;
80 IF stop = TRUE THEN
81     q2a_forward:=FALSE;
82     q2a_reverse:=FALSE;
83     q2a_speed:=0;
84 END_IF;
```

Kode 5.6: Kjøring under posisjonsregulering:
I linje 62-66 Begrenser 'regulator'-variabelen til et maksimalt tillatt område, definert av "HMI_regulator_maks". Hvis "regulator" er større eller mindre enn "HMI_regulator_maks", settes "regulator" til "HMI_regulator_maks". I linje 67-79 kontrollereres motorens retning og hastighet basert på "regulator"-variabelen. Hvis "regulator" er større enn 0, går motoren fremover ("q2a_forward" satt til TRUE) med hastigheten satt til absoluttverdien av "regulator" ("q2a_speed" = ABS(regulator)). Hvis "regulator" er mindre enn 0, går motoren bakover ("q2a_reverse" satt til TRUE) med hastigheten satt til absoluttverdien av "regulator". Hvis "regulator" er lik 0, stopper motoren. I linje 81-85 håndteres nødstopp-knappen. Hvis nødstopp-knappen er aktivert, stopper motoren umiddelbart ved å sette både "q2a_forward" og "q2a_reverse" til FALSE og "q2a_speed" til 0.

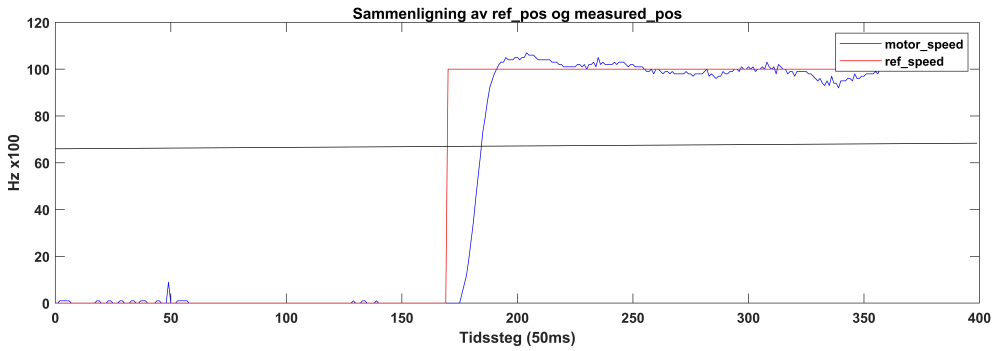
5.3 Systemets dynamikk

I dette delkapittelet analyseres tidkonstanten, systemforsterkningen og tidsforsinkelsen i forbindelse med våre standardinnstillinger for tuningparameterne, vist i tabell 2.2.

Tidskonstanten (Tau (τ))

Tidskonstanten tau, ofte betegnet med symbolet τ, er en viktig parameter som beskriver systemets dynamikk. Tau er definert som den tiden det tar for systemet å nå 63,2% av sin endelige verdi når det blir utsatt for et sprang/pådrag.

Måten vi fant vår tau-verdi var med å gi motoren et sprang på 1Hz og observerte hvor lang tid det tok for motor_speed å nå 63,2% av denne hastigheten. Dette blir illustrert i figur 5.3. Det ble funnet ut at systemet brukte 0,151s på å nå denne hastigheten. Dvs Tau (τ) = 0,151s.



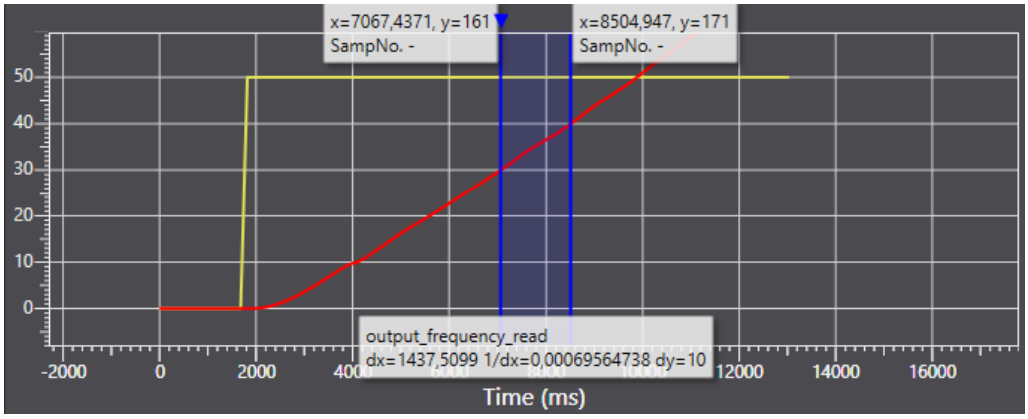
Figur 5.3: Figuren viser et enkelt sprang som ble benyttet for å finne systemets tidskonstant.

Systemets forsterkning (K)

For å fastslå systemets forsterkning, benyttet vi en metode der frekvensreferansen ble justert og etterfølgende endringer i utgangssignalet(loddets høyde), ble observert. Hovedmålet var å undersøke hvor raskt loddet hevet seg ved en spesifikk frekvens. Ved å justere frekvensreferansen opp med 0,5 Hz, noterte vi endringen

i loddets høyde.

Videre ble forsterkningen beregnet ved å evaluere stigningen eller endringen mellom to høyder, noe som ga oss to datapunkter på grafen vår: (X1,Y1) og (X2,Y2). Dette er illustrert i Figur 5.4, der vi viser hvordan målingen ble utført og hvordan vi kom frem til verdiene for X1=7, X2=8.5, Y1=30 og Y2=40.



Figur 5.4: Figuren illustrerer stigninen mellom to høyder for å finne systemets forsterkning.

Videre benyttes disse formlene for å regne ut forsterkningen(K):

$$K = \frac{S}{\Delta U} \tag{5.4}$$

$$\Delta U = 0,5 \tag{5.5}$$

$$S = \frac{Y2 - Y1}{X2 - X1} = \frac{40 - 30}{8,5 - 7} = 6,666[mm/s] \tag{5.6}$$

Resultat:

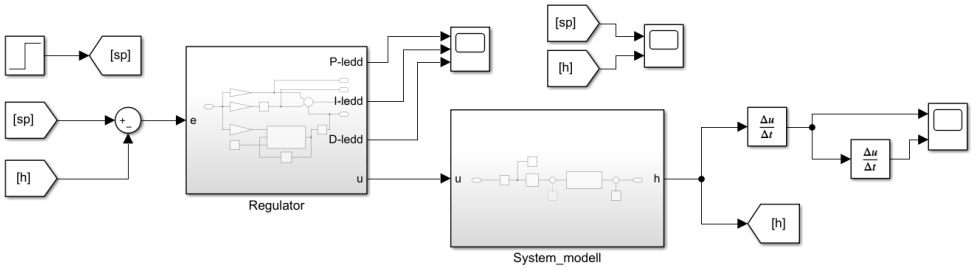
$$K = \frac{6,666}{0,5} = 13,333[mm/Hz] \tag{5.7}$$

Systemets tidsforsinkelse

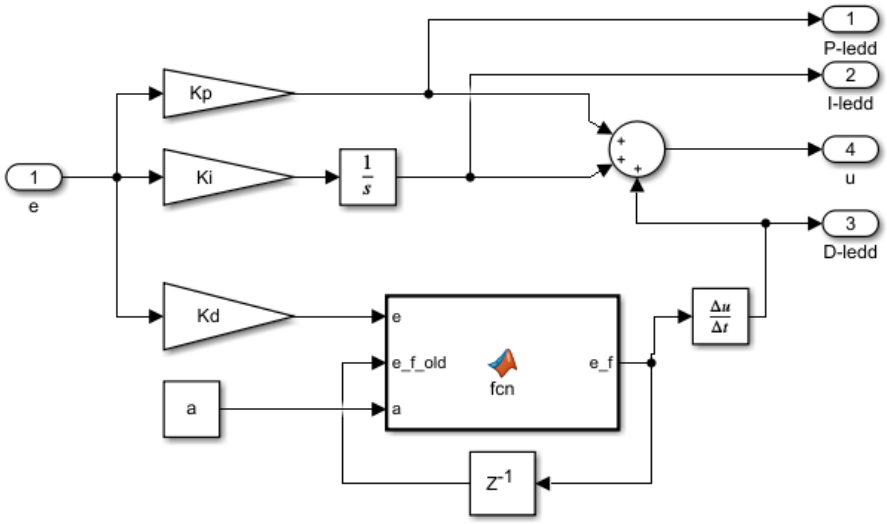
I kapittel 4.2 for jerk-parameterne så ble det forklart at systemet vil ha en tidsforsinkelse på 40ms. Hvor denne oppstår er delvis ukjent. Det ble vist at "jerk@start of accel"-parameteren har en innvirkning på systemets respons. Imidlertid, selv etter fullstendig eliminering av denne parameteren, vil systemet fremdeles oppleve en tidsforsinkelse.

5.4 Simulink modell

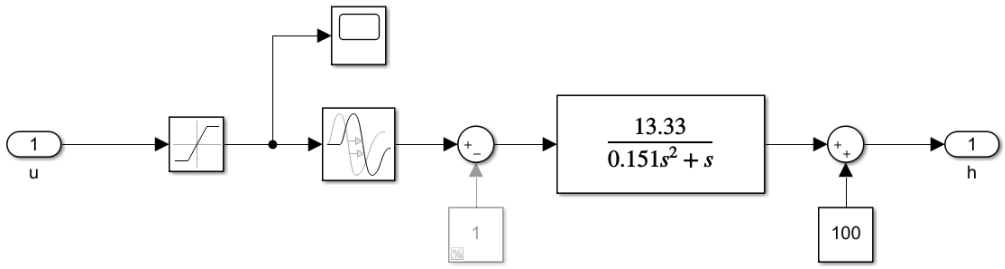
Basert på verdiene vi fikk under analysen av systemets dynamikk, teorien bak en PID-kontroller og matematikken bak overføringsfunksjonen, ble det lagd en modell i Simulink. Modellen ville gi oss tilnærmet like simuleringer som i det fysiske systemet, og dette gjorde det lettere for oss å teste ulike metoder for å finne riktige verdier til PID-regulatoren benyttet i oppgaven. I modellen brukes samme tidsskritt(0.002) og alpha verdi(0,2) som vi benyttet i programmet vårt.



Figur 5.5: Simulink modellen



Figur 5.6: Subsystem av regulator

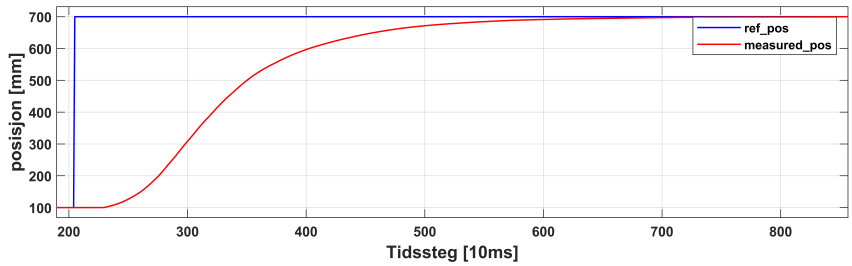


Figur 5.7: Subsystem av system_modell

En betydelig fordel ved å anvende Simulink-modellen var muligheten til å utforske flere kombinasjoner av PID-verdier uten å risikere skade på systemet vårt. Siden systemet vårt opererer innenfor et begrenset område, var et av de viktigste kriteriene for PID-regulatoren å unngå å oversving. Modellen gjorde det derfor lettere for oss å kjøre flere tester, raskere og på en litt tryggere måte. Derimot var ikke modellen helt feilfri. Siden den ikke tok hensyn til Q2Aen sine innebygde begrensinger og funksjoner(som forklart i kap 4), så vil responsen fra det fysiske systemet ikke bli helt identisk. Målingene ble tilnærmet like og dette hjalp oss nok til å unngå tester som ville skadet systemet.

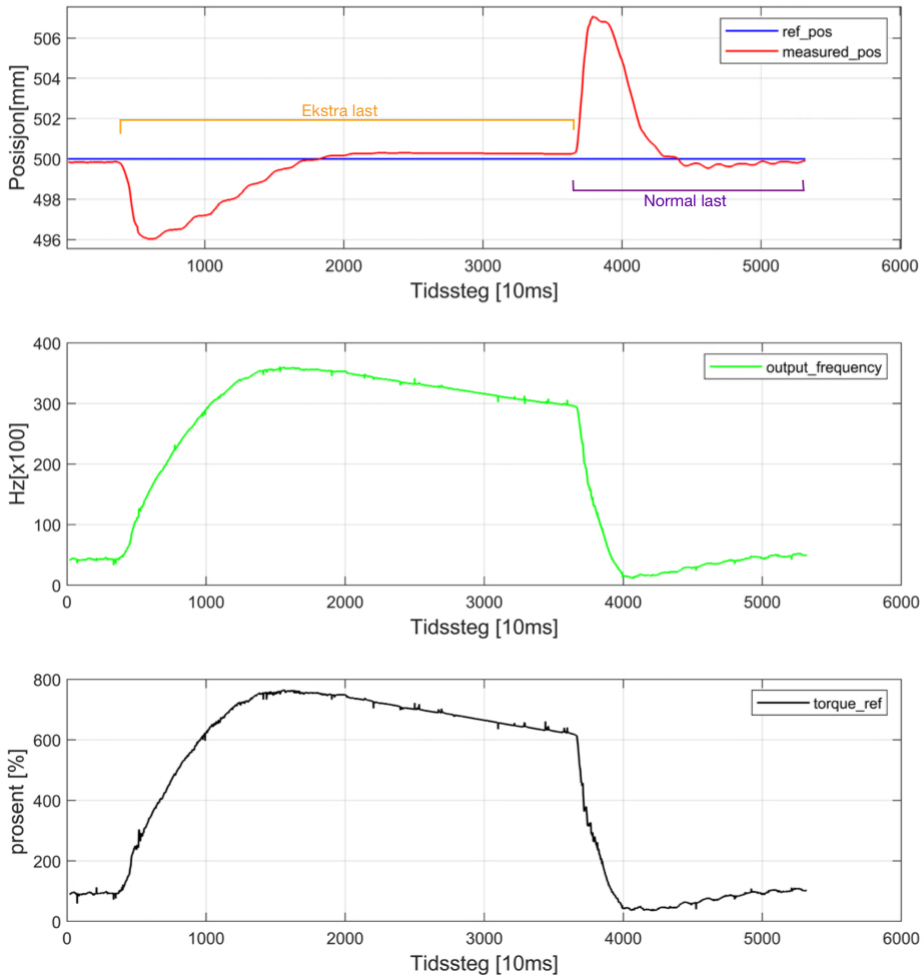
5.5 Manuell tuning av P-regulator

Ved hjelp av Simulink-modellen og manuell-tuning, som innebærer en ”prøv og feil”-metode, ble det funnet en Kp-verdi på 0,03. Dette er den verdien som ga oss den raskeste responsen, oppfylte våre krav til systemet og som heller ikke brøyt noen av Q2A-ens begrensninger. For å avgjøre om en PID-regulator vil gi bedre respons, var det viktig at vi først teste en P-regulator. Dette ble gjort for å kunne sammenligne resultatene og deretter danne et grunnlag for å forklare hvorfor vi valgte den ene regulatoren fremfor den andre for å utføre posisjonsregulering.



Figur 5.8: Fysisk kjøring av en P-regulator. Vi gjennomførte målingen med et sprang på 600mm fra 100 til 700mm.

Figur 5.8 viser responsen til P-regulatoren med en enkel sprangtest fra 300 til 400 mm. Ved å benytte en Kp-verdi på 0,03, ble det oppnådd en stabil og rask respons. Denne Kp-verdien kan benyttes til justering av systemet, slik at ønsket høyde oppnås. Siden Q2A-enheten inneholder et innebygget integralledd i ASR-løkken, er det en teori om at en P-regulator kan håndtere lastkompensasjon. For å teste om P-regulatoren har lastkompenseringsegenskaper, utførte vi et eksperiment der loddet ble holdt på en konstant høyde på 500 mm over bakken, og en ekstra last ble lagt til. Resultatet fra dette eksperimentet er vist i figuren under:



Figur 5.9

Som illustrert ovenfor i den midterste figuren 5.9 med det grønne signalet, øker systemets pådrag gradvis etter at den ekstra lasten er lagt til. Dette skjer fordi det innebygde integralet i ASR-sløyfen vokser. Dette viser at systemet er i stand til å kompensere for endringer i last og opprettholde en stabil posisjon med kun en P-regulator. Det vil derfor ikke være bruk for en PID-regulator, men videre i oppgaven blir det undersøkt om en PD-regulator vil gi bedre respons enn P-regulatoren.

5.6 Manuell tuning av PD-regulator

Under manuell tuning av PD-regulatoren, benyttet vi Simulink-modellen og "Prøv og feil"-metoden. Vi gikk ut ifra Kp-verdien til P-regulatoren på 0.03 som en start verdi, og benyttet tabellen 5.1 til å finne de beste verdiene for Kp og Kd.

↑ Parameter	Rise Time	Overshoot	Settling time	Steady State error	Stability
Kp	Decrease	Increase	Smal change	Decrease	Degrade
Kd	Minor change	Decrease	Decrease	No effect	Improve

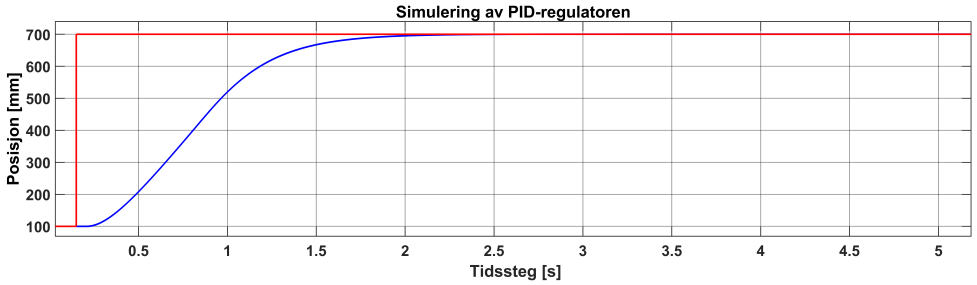
Tabell 5.1

5.6.1 Simulering i simulink av PD-regulator, enkelt sprang [100 → 700mm]

Etter flere simuleringer og tester så fant vi til slutt disse PD-verdiene for systemet vårt:

- Kp = 0,215
- Kd = 0,135

Figur 5.10 viser resultatet fra simuleringen av systemet.



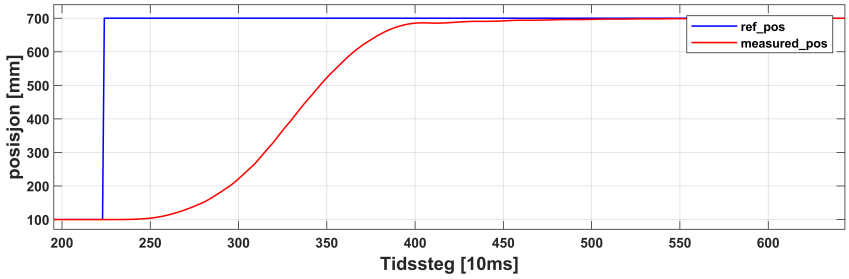
Figur 5.10: Simulering i simulink av enkelt sprang

Ved å bruke verdiene Kp = 0,215 og Kd = 0,135 i PD-regulatoren, har vi oppnådd et system som både er stabilt, uten oversving og med rask respons. Simuleringen

ble utført ved å bruke gi et sprang fra 100 til 700 mm i referanseposisjonen. Dette ble gjort for å sikre at simuleringen i Simulink lignet så mye som mulig på den virkelige responsen.

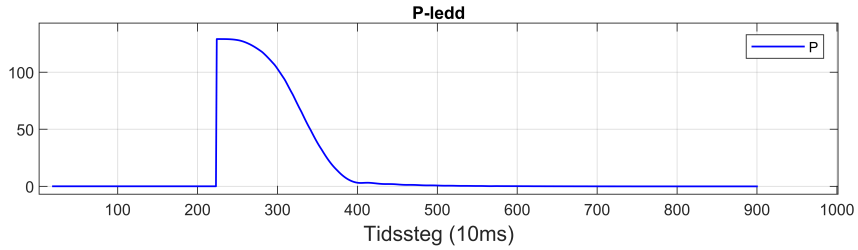
5.6.2 Fysisk kjøring av PD-regulatoren, enkelt sprang [100 → 700mm]

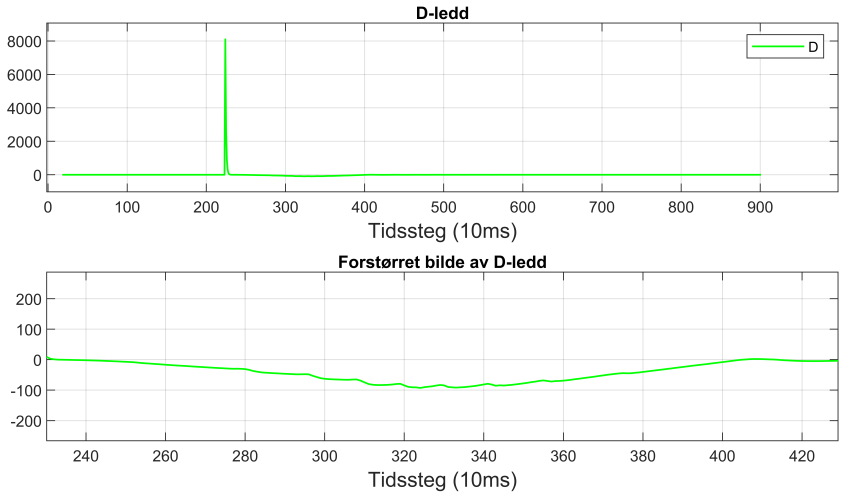
Etter å ha funnet de optimale PD-verdiene fra simulink-modellen, ble disse testet i praksis for å vise at de gir ønsket respons til systemet. Figur 5.11 viser responsen til systemet og vi kan se at det har tilnærmet lik respons som i simuleringen. Som antatt, bemerker vi at det fysiske systemet viser en viss avvikelse fra det simulerte, dette er som forklart tidligere på grunn av Q2Aen sine innebygde funksjoner og begrensinger.



Figur 5.11: Resultat fra fysisk kjøring av enkelt sprang

Som vist i kodeudratet 5.5 er variabelen "regulator" referansehastigheten under posisjonsregulering. Variabelen er en sum av resultat fra P, I og D leddene vist i kodene for P 5.1, I 5.3 og D 5.4. Under i figuren 5.12 er det vist hvordan P- og D-leddet bidrar med å styre hastigheten på motoren.

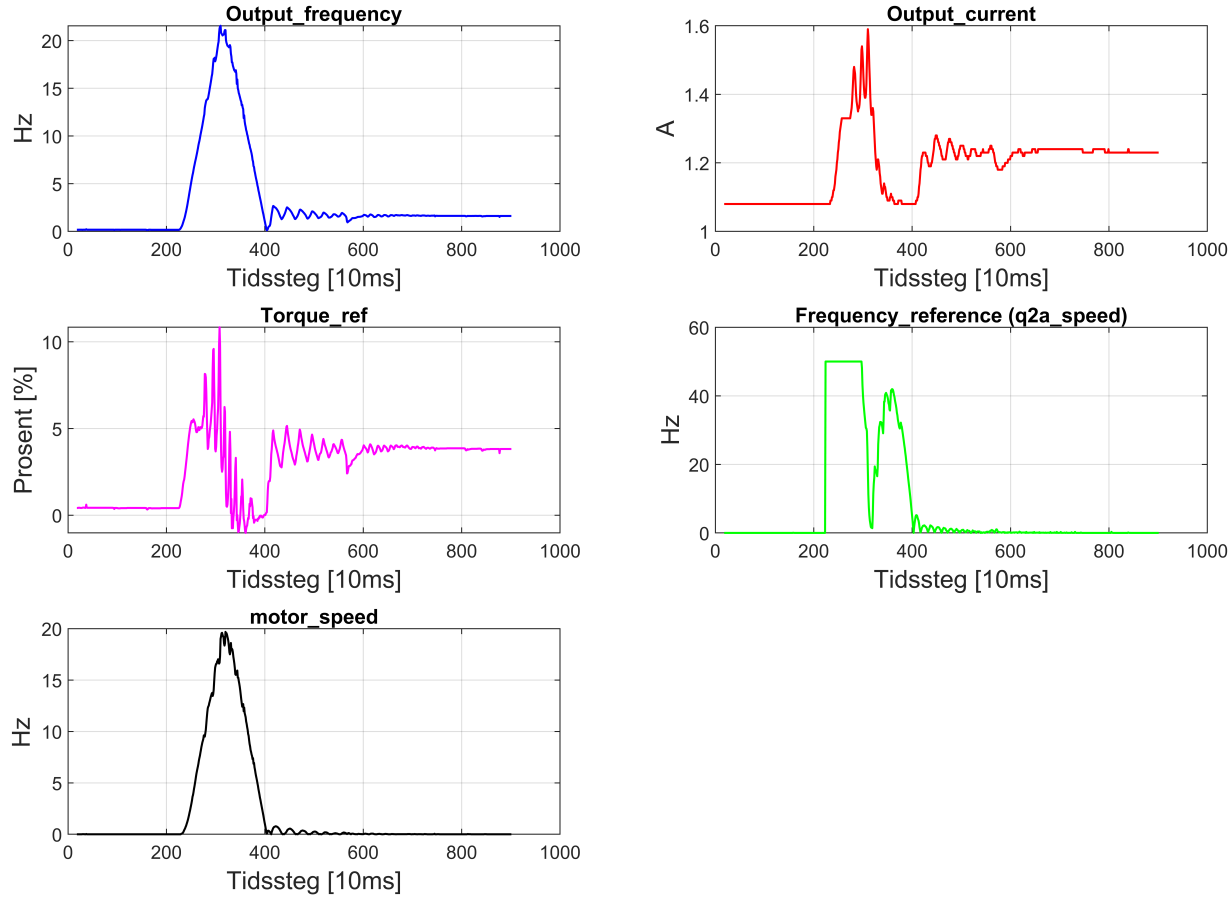




Figur 5.12: P og D ledd i PD-regulatoren under en enkel sprangtest

Ved å analysere tidspunktet for spranget til referanseposisjonen, som vist i Figur 5.11, og responsen fra P-leddet til dette spranget, som illustrert i Figur 5.12, kan vi observere at P-leddet i PD-regulatoren justerer hastigheten basert på avviket mellom measured_pos og ref_pos. P-leddet vil gi et betydelig, men avtagende sprang. D-leddet i regulatoren representerer det deriverte avviket mellom referansen og målingen. Dette resulterer i et stort sprang i begynnelsen, etterfulgt av en negativ verdi som skyldes den avtagende deriverte av avviket. Når disse verdiene kombineres, vil de gi et betydelig sprang i starten og i tillegg håndtere stabiliseringen rundt referanseposisjonen.

Figur 5.13 viser oppførselen til andre variabler under testing av PD-regulatoren.



Figur 5.13

Fra delfiguren "frequency_reference" kan vi observere at når frekvensen når en verdi på 50Hz, blir den holdt konstant ved denne verdien. Dette er en konsekvens av grensen til utgangsfrekvensen implementert på HMI-skjermen. Verdien til maksgrenesen ble satt til 50Hz for å unngå at regulatoren gir et større pådrag enn motorens maksimale utgangsfrekvens (E1-04 i Q2-edit). Hvis "frequency_reference" overstiger 50 Hz, kan tilkoblingen mellom PLSen og Q2Aen potensielt bli avbrutt.

Fra delfiguren "motor_speed" 5.13 observerer vi at motorhastigheten når 20 Hz og reduseres etter hvert som lasten nærmer seg ønsket posisjon. En rask oppstart

skyldes det store avviket ved begynnelsen av spranget.

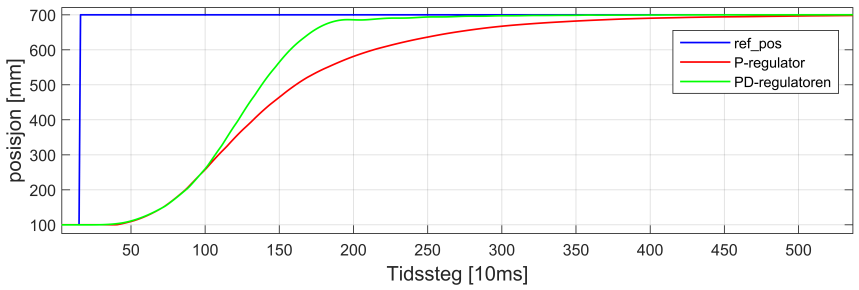
Delfiguren "output_current" 5.13 viser at strømtrekket varierer med endringer i avviket. Videre, ved å sammenligne med figuren for "torque_ref", ser vi at "output_current" speiler "torque_ref" gjennom hele prosessen. Dette er fordi i elektriske motorer er moment og strøm direkte relatert. Når momentet øker, krever motoren mer kraft og dermed mer strøm for å overvinne motstanden.

Om lag tidssteg 600 så legger "output_current" seg stabilt på 1,23 A og ikke på 1,1 A som den lå på før prosessen. Dette skyldes at når det er mye innspolt vaier, blir totalradiusen større og det kreves mer strøm for å løfte lasten. Strømforbruket vil derfor forbli stabilt på en høyer verdi enn utgangsverdier før prosessen startet.

Fra delfiguren "output_frequency" observerer vi at utgangsfrekvensen overstiger 20 Hz. Årsaken til at denne må være høyere enn "motor_speed" er at den må kompensere for lasten, som veier 1,5 kg. Som tidligere nevnt, representerer utgangsfrekvensen den frekvensen frekvensomformeren må generere for å oppnå ønsket motorhastighet. Videre kan vi se at når lasten når referanse posisjonen og den legger seg stabilt, omkring tidssteg 600, reduseres utgangsfrekvensen til omtrent 2,3 Hz. Igjen som forklart tidligere vil systemet oppleve mer moment når lasten er høyere oppe, og derfor vil "output_frequency" ha en høyere verdi når lasten ligger stabilt på 700m sammenlignet med 100mm.

5.6.3 Sammenligning av P-regulatoren og PD-regulatoren

I dette delkapittelet blir det sammenlignet forskjellen mellom en P og PD-regulator. Under i figur 5.14 blir det vist resultatet fra en maks sprang test(100mm → 700mm).



Figur 5.14: Sammenligning av P-regulatoren og PD-regulatoren under maks sprang test

Tabellen viser parameterne som er benyttet i testen

Regulator	Kp	Ki	Kd
P-regulator	0,03		
PD-regulator	0,215		0,135

Tabell 5.2

Figur 5.14 presenterer en sammenligning av ytelsen til P-regulatoren og PD-regulatoren ved implementering av et sprang fra 100mm til 700mm over bakken. Ut fra figuren kan vi observere at PD-regulatoren har en raskere respons sammenlignet med P-regulatoren. Figuren viser at mens P-regulatoren tar omtrent 4,5 sekunder for å nå referanseposisjonen, oppnår PD-regulatoren den samme posisjonen på kun 2.5 sekunder. Basert på disse observasjonene, har vi konkludert med at en enkel P-regulator vil være i stand til posisjonsregulering etter våre krav. Derimot vil en PD-regulator være det mest effektive alternativet for posisjonsregulering av vårt system.

5.7 Skogestads metode

Skogestad's metode er en metode for å optimalisere parametrene til en PID-regulator for å oppnå ønsket dynamisk respons i et reguleringssystem.

Formålet med å bruke Skogestads tuningmetode i vår oppgave var å undersøke muligheten for å finne hensiktsmessige PID-verdier. Ved å anvende Skogestads metode kan vi potensielt etablere solide utgangsverdier, eller til og med oppnå direkte optimale parametere. Selv om Skogestads tuningmetode kalkulerer verdier for en PID-regulator, antok vi at denne metoden ikke vil være optimal for vårt system. Vi hadde fortsatt en hypotese om at metoden fortsatt kunne generere nyttige parametere hvis vi utelukket I-leddet, og utelukkende benyttet oss av verdiene til en PD-regulator. Prosessen for beregning og resultater fra våre tester vil bli presentert i dette underkapittelet.

I systemets prossess er det en tidsdelay vis i figur ref(timedelay)

Under i tabell 5.3 er formlerene for skogestad tuning av systemer med tidsdelay.

$Hp(s)$ (process)	K_p	T_i	T_d
$\frac{K}{s}e^{-\tau s}$	$\frac{1}{K(TC+\tau)}$	$k_1(TC+\tau)$	0
$\frac{K}{Ts+1}e^{-\tau s}$	$\frac{T}{K(TC+\tau)}$	$\min(T, K_1(TC+\tau))$	0
$\frac{K}{s(Ts+1)}e^{-\tau s}$	$\frac{1}{K(Tc+\tau)}$	$K_1(T_c+\tau)$	T
$\frac{K}{(T_1s+1)(T_2s+1)}e^{-\tau s}$	$\frac{T_1}{K(T_c+\tau)}$	$\min(T_1, K_1(T_c+\tau))$	T_2
$\frac{K}{s^2}e^{-\tau s}$	$\frac{1}{4K(T_c+\tau)^2}$	$4(T_c+\tau)$	$4(T_c+\tau)$

Tabell 5.3

- $K_1 \rightarrow$ Kompenseringsfaktor. De vanligste verdiene for K1 er enten 1,44 eller 4. Vi har valgt å benytte 1,44 som vår kritiske forsterkning for raskere forstyrrelseskompensering.
- $T_c \rightarrow$ Tidskonstant for prosessdynamikken. Dette er tidskonstanten som beskriver hvor raskt systemet reagerer på endringer i inngangssignalet. I tuningen ble T_c lik τ benyttet.
- $\tau \rightarrow$ Tiden det tar før den målte verdien er nådd 63.2% av sprangets magnitude.
- $K \rightarrow$ Systemets forsterkning.

Under tuning ble likningene i rad 4, tabell 5.3 benyttet. Disse ble brukt for å regne ut PID-verdier med god resons basert på skogestads matematiske formler. Formelene som samsvarer med vårt 2.ordens system er vist under i tabellen 5.4.

Tabell 5.4: Skogestad 2.ordens formler

$H_p(s)$ (process)	K_p	T_i	T_d
$\frac{K}{s(Ts+1)}e^{-\tau s}$	$\frac{1}{K(Tc+\tau)}$	$K_1(T_c+\tau)$	T

Verdiene oppnådd med skogestad fra tabell 5.4 er hovedsaklig egnet for en seriell PID-regulator, mens det i denne oppgaven blir benyttet en parallell PID-regulator. Derfor ble det nødvendig å endre fra serielle til paralelle verdier. Endringene er vist under i liste 5.7

- $Kp_{Parallell} = Kp_{seriell} \cdot (1 + \frac{Td_{seriell}}{Ti_{seriell}})$

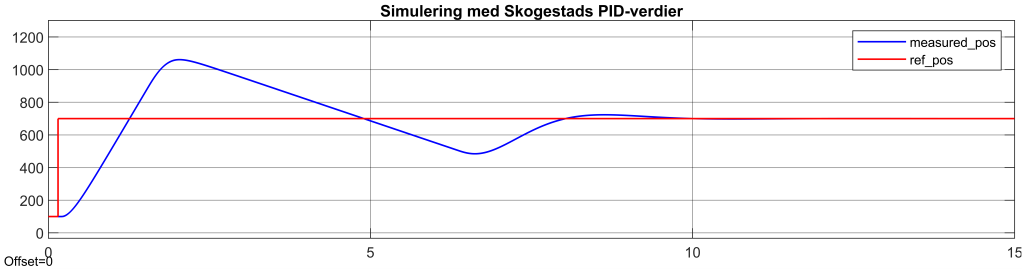
- $Ti_{Parallell} = Ti_{seriell} \cdot (1 + \frac{Td_{seriell}}{Ti_{seriell}})$
- $Td_{Parallell} = Td_{seriell} \cdot (\frac{1}{1+\frac{Td_{seriell}}{Ti_{seriell}}})$

I kapittel 4.2 ble det fremstilt hvordan $\tau \rightarrow 0.15$ og gain (k) $\rightarrow 13.33$ ble funnet. Under i listen 5.7 er utregningene og verdiene funnet for skogestad tuning parametere til parallell PID-regulator.

- $Kp_{seriell} \rightarrow \frac{1}{Ts+1}e^{-\tau s} \rightarrow 0.25$
- $Ti_{seriell} \rightarrow K_1(T_c + \tau) \rightarrow 0.432$
- $Td_{seriell} \rightarrow \tau \rightarrow 0.15$
- $Ti_{parallell} \rightarrow Ti_{seriell} \cdot (1 + \frac{Td_{seriell}}{Ti_{seriell}}) \rightarrow 0.582$
- $Td_{parallell} \rightarrow Td_{seriell} \cdot \frac{1}{1+\frac{\tau}{Ti_{seriell}}} \rightarrow 0.111$
- $Kp_{parallell} \rightarrow Kp_{parallell} \cdot (1 + \frac{Td_{seriell}}{Ti_{seriell}}) \rightarrow 0.3369$
- $Ki_{parallell} \rightarrow \frac{Kp_{parallell}}{Ti_{parallell}} \rightarrow 0.579$
- $Kd_{parallell} \rightarrow Kp_{parallell} \cdot Td_{parallell} \rightarrow 0.0375$

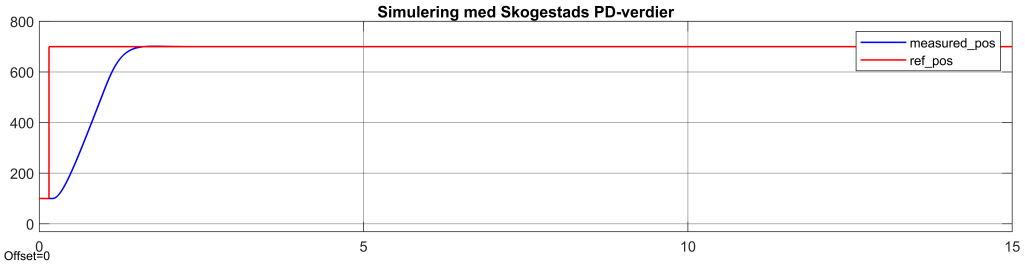
5.7.1 Simulering med bruk av Skogestads tuning metode

Før verdiene fra Skogestads tuning metode ble benyttet i den fysiske PID-regulatoren, ble verdiene i Simulink modellen. Resultatet fra simuleringen er vist i figur 5.15.



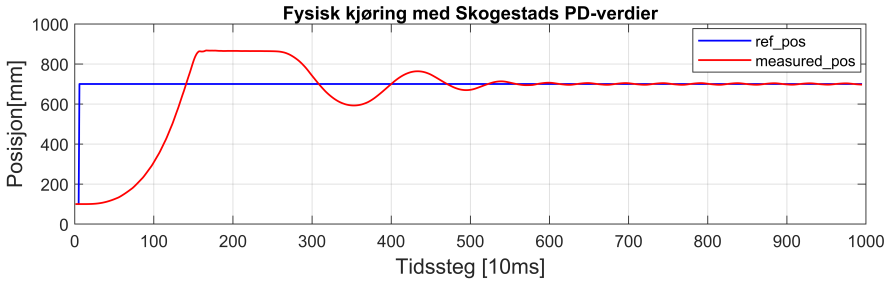
Figur 5.15: Enkelt sprangrespons til PID-regulator med Kp = 0.3369, Ki = 0.579 og Kd = 0.0375

Som illustrert i figuren over så vil verdiene fra Skogestads metode gi for kraftig respons og disse verdiene vil ikke være optimale å teste i systemet. Vi hadde derimot en teori om at skogestads metode kunne fortsatt benyttes dersom vi fjernet I-leddet. Vi gjennomførte en ny simulering i Simulink-modellen, men denne gangen kun med en PD-regulator. Resultatet fra simuleringen er vist i figur 5.16.



Figur 5.16: Enkelt sprangrespons med PD-regulator (Kp = 0.3369 og Kd = 0,0375)

I figuren ovenfor kan vi se at Skogestads tuningmetode gir en troverdig simulering når vi eliminerer I-leddet fra PID-regulatoren, og dermed kun benytter en PD-regulator. På grunn av forskjellene mellom Simulink-modellen og det fysiske systemet er det nødvendig å bekrefte PD-regulatorens ytelse med Skogestads verdier under faktiske forhold. Resultatet fra testingen i det virkelige systemet er fremstilt i figuren (5.17) nedenfor.



Figur 5.17: Enkelt sprangrespons med PD-regulator (Kp = 0.3369 og Kd = 0,0375)

Ved å analysere systemets respons ved bruk av Skogestads tuningparametere i en PD-regulator, kan vi konkludere med at Skogestads metode ikke er optimal for vårt system. Denne metoden tar ikke i betraktning parametrene accel og decel Time, noe som resulterer i at responsen (akselerasjonen) blir betydelig påvirket av Accel Time-parameteren, som fører til at motoren oppnår ønsket hastighet

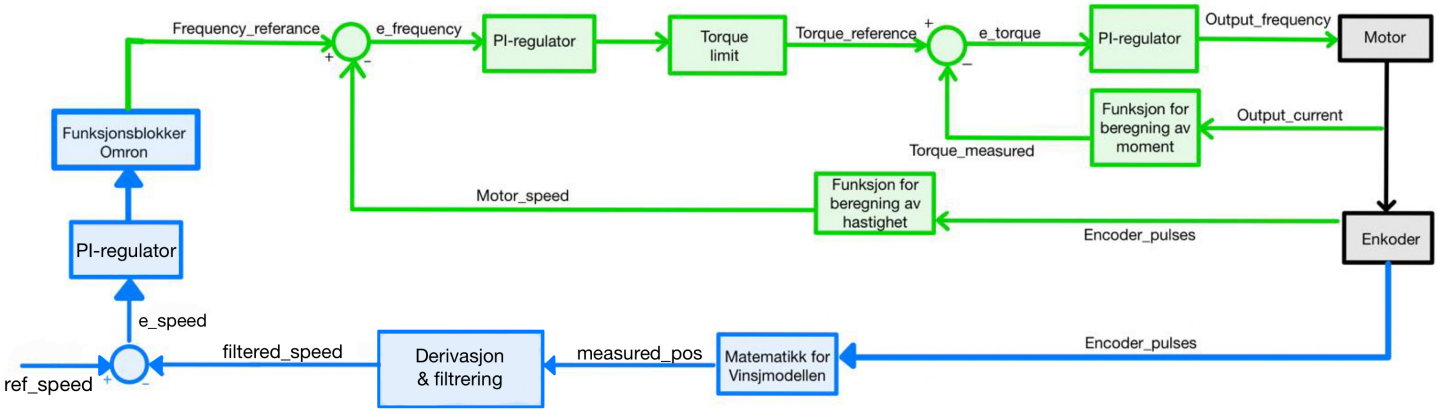
ved en høyde for tett på referanseposisjonen(ref_pos). Decel Time-parameteren hindrer så motoren i å bremse tilstrekkelig raskt, noe som resulterer i betydelig oversving.

Etter testen med bruk av skogestads tuningverdier, kunne vi også konkludere med at manuelle tuningmetoder er den mest passende tilnærmingen for dette systemet, ettersom at andre metoder ikke tar hensyn til Q2Aens innebygde funksjoner og begrensninger.

Kapittel 6

Hastighetsregulering

I dette kapittelet presenteres teori, løsning og resultater for hastighetsregulering av et vinsjsystem. For å oppnå en konstant innspolingshastighet i systemet, ble hastighetsregulering implementert. Ved å derivere posisjonen til lasten, blir innspolingshastigheten beregnet. Lastens høyde over bakken (measured_pos) beregnes basert på de matematiske utregningene, som forklart i delkapittelet om teach-funksjonen 3.1.



Figur 6.1: Blokkdiagram over hastighetskontroll

Referansehastigheten(ref_speed) er ønsket hastighet, og reguleringsavviket(e_speed) er differansen mellom referansen og målt hastighet. Differansen bli så sendt inn i en PI-regulator, som via en funksjonsblokk fra Omron sender referansesignalet "frequency_reference" til frekvensomformeren. Videre blir signalet beregnet i reguleringsløkken for closed loop vektorkontroll(kap 2.2.1) som beregner pådraget "output_frequency".

Hastighetsregulatoren vil regulere pådraget til motoren for å oppnå ønsket innspolingshastighet i vinsjsystemet. Ved lastøkning eller andre forstyrrelser under innspolingen vil regulatoren øke pådraget til motoren for å vedlikeholde hastigheten. Brukeren setter en hastighetsreferanse for lasten i mm/s på HMI-skjermen. Pulsene fra enkoderen som telles av variabelen (encoder_pulses) i frekvensomformeren, blir brukt til å oppdatere trommelradiusen og nåværende høyde.

6.1 Teori

En PI-regulator ble benyttet for hastighetsregulering av en vinsj på grunn av dens evne til å kombinere fordelene ved både proporsjonal- og integralregulering, noe som resulterer i bedre ytelse og større stabilitet.

En ren P-regulator vil i teorien ikke være tilstrekkelig for hastighetsregulering, ettersom den kun ville tilpasse seg proporsjonalt til avviket mellom ønsket og faktisk hastighet. Dette kan føre til ustabilitet og konstante svingninger i hastigheten.

Likningen for PI-regulatoren benyttet i oppgaven er vist under i likning 6.1.

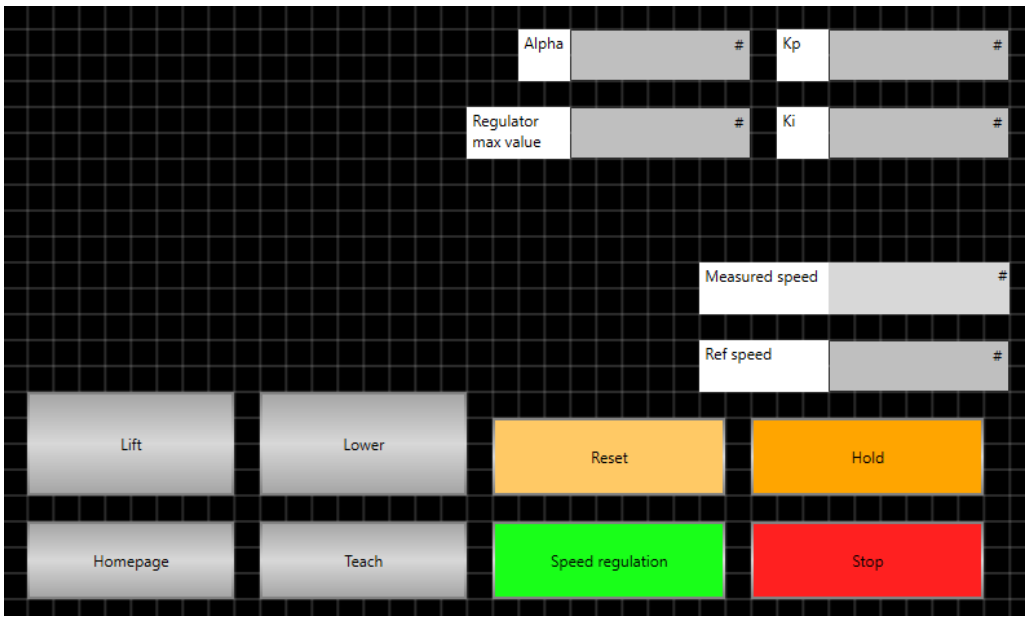
$$u(t) = K_p e + K_i \int_0^t e(\tau) d\tau \tag{6.1}$$

Den numeriske integrasjonen basert på *Eulers forovermetode* som ble brukt for å lage I-leddet er vist i likning (6.2):

$$u_I(k) = u_I(k-1) + e(t) \cdot T_s \tag{6.2}$$

6.2 Kodene til hastighetsreguleringen

I denne delen av kapittelet skal vi forklare hvordan koden vår er bygd opp og hvordan brukeren kan bruke HMI-skjermen for å gjennomføre hastighetsregulering. For å starte hastighetsreguleringen må brukeren først gjennomføre en vellykket "teach" av systemet. De globale variablene fra "teach"-programmet vil bli lagret og gjort tilgjengelig i hastighets reguleringsprogrammet.



Figur 6.2: HMI-skjermen for posisjonsregulering

På HMI-skjermen er det lagt inn 5 DataEdit-vinduer. Disse gjør det enkelt for brukeren å endre verdiene til PI-regulatoren(Kp og ki), filterkonstanten (alpha) og limit-verdien som er satt inn i regulatorens output. HMI-skjermen består også av 4 fargede kontroller: Speed regulation, Stop, Reset og Hold. Funksjonene til disse er beskrevet nedenfor:

- Hold → Holder lasten i nåværende høyde.
- Reset → Tilbakestiller alle verdiene til de som ble lagre fra teach-funksjonen. Nullstiller også integral-leddet

- Speed regulation → Starter hastighetsreguleringen og loddet blir senket/-løftet til ønsket høyde
- Stop → Nødstopp. Setter q2a_speed = 0

Kode 6.1: Hastighet

De første kodene implementert var kodene for beregning av hastighet. Kodene er vist under i kodeliste 6.1

Kode 6.1: Hastighet

```
 9  measured_pos:=(rope_after_lift/6);
10  alpha := HMI_alpha;
11  position := measured_pos;
12  measured_speed:=(position-position_old)/ts_nano_real;
13  position_old := position;
14  filtered_speed:=alpha * ABS(measured_speed) + ...
        (1-alpha)*filtered_speed_old;
15  filtered_speed_old:=filtered_speed;
16  e_speed := ref_speed - filtered_speed;
17  P:=LREAL_TO_REAL(kp*e_speed);
```

Den målte posisjonen beregnes i linje 9 basert på hvor mye tau som er løftet. I linje 12 blir den målte hastigheten beregnet ut ifra differansen mellom nåværende og tidligere posisjon, delt på tidskrittet ts_nano_real I linje 14 blir den filtrerte hastigheten beregnes ved hjelp av et IIR-filter med alfa-verdien og oppdateres kontinuerlig. I linje 16 beregnes hastighetsavviket mellom referansehastigheten og den målte filtrerte hastigheten. linje 17 multiplisere hastighetsfeilen med verdien "kp" valgt av bruker på HMI-skjerm.

Videre ble integralleddet i hastighetsregulatoren implementert. Kodene er illustrert under i kodelisten 6.2

Kode 6.2: Integral

```
18 // Beregning av tidskritt
19 time_now :=GetTime();
20 ts:=SUB_DT_DT(time_now, time_old);
21 time_old := time_now;
22 ts_nano :=TimeToNanoSec(ts);
23 ts_nano_real :=LINT_TO_REAL(ts_nano) / 1000000000;
24
25 integral:=previous_integral+ki*(e_speed*ts_nano_real);
26 IF integral ≥ 1 THEN
27 integral:=1;
28 ELSIF integral ≤ -1 THEN
29 integral:=-1;
30 END_IF;
31
32 previous_integral :=LREAL_TO_REAL(integral);
33 I:=LREAL_TO_REAL(integral);
```

Kode 6.2: Integral

I linje 19-23 beregnes tidskrittet. linje 25-29 beregner integralverdien ved å legge til produktet av hastighetsfeilen, tidskrittet og koeffisienten ki valgt av bruker. Integralverdien begrenses mellom -1 og 1 i linje 26-29. Denne begrensningen ble brukt for å forhindre et fenomen kjent som "integrator windup". Integrator windup oppstår når det akkumuleres en for stor feilsum over tid i integrasjonsleddet. Videre blir I-leddet beregnet i linje 33.

Videre ble det implementert koder for motorstyringen under hastighetsreguleringen. Kodene er vist under i kodelisten 6.3

Kode 6.3: Motorstyring

```
40 regulator:=LREAL_TO_REAL(P+I);
41 IF regulator > HMI_regulator_maks THEN
42 regulator := HMI_regulator_maks;
43 ELSIF regulator <- HMI_regulator_maks THEN
44 regulator := - HMI_regulator_maks;
45 END_IF;
46
47 lift:=HMI_lift;
48 lower:=HMI_lower;
49
50 IF lift = TRUE THEN
51 q2a_forward := TRUE;
52 q2a_reverse := FALSE;
53 q2a_speed := ABS(regulator);
54 END_IF;
55
56 IF lower = TRUE THEN
57 q2a_forward := FALSE;
58 q2a_reverse := TRUE;
59 q2a_speed := ABS(regulator);
60 END_IF;
61
62 IF HMI_hold = TRUE THEN
63 q2a_forward := TRUE;
64 q2a_reverse := FALSE;
65 q2a_speed :=
66 0;
67 END_IF;
```

Kode 6.3: Motorstyring

I linje 40 blir regulatorverdien beregnet som summen av P- og I-leddene. Regulatorverdien begrenses i linje 41-44 til maksimumsverdien satt i HMI-skjermen. Linje 47-48 bestemmer om lasten skal løftes eller senkes ved bruk av ”lift” og ”lower” bestemt fra HMI-skjermen. Er hold akrivert i linje 69, settes hastighet til 0 og lasten henger i ro til neste instruks blir gitt.

6.3 Resultat hastighetsregulering

I dette delkapittelet vil vi presentere resultatene fra ulike tester knyttet til hastighetsregulering av systemet. Reguleringsparametrene blir funnet ved hjelp av "prøv og feil"-metoden. Deretter sammenlignes parameterene fra "prøv og feil"-metoden med verdier funnet gjennom skogestads metode.

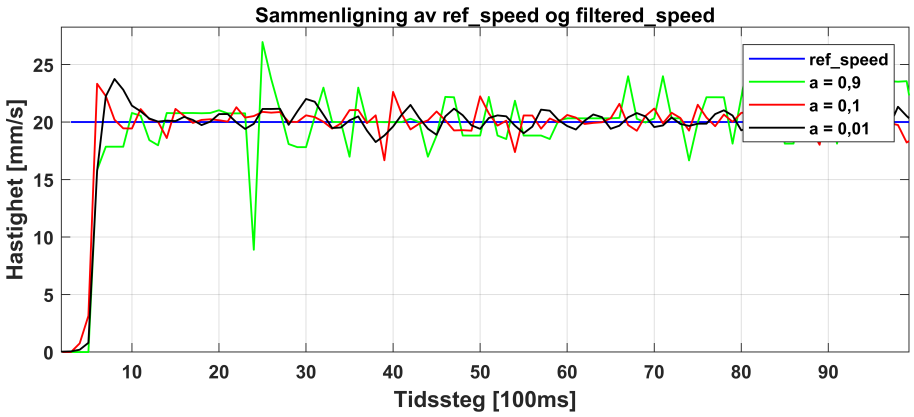
Følgende krav er satt for prosessen:

- Lite oversving
- Rask respons
- Håndtering av forstyrrelser

6.3.1 Betydning av filterverdi

For å se betydningen av filterets parameterverdi ble tre ulike parametre testet. Filterverdien er viktig for å kunne filtrere støyete verdier som følger av den deriverte posisjonen.

I testene ble en I-regulator brukt med en Ki-verdi på 0.1. Alle testene ble utført ved at lasten ble hevet med en innspolingshastighet på 20 mm/s. De testede parameterverdiene var 0.01, 0.1 og 0.9. Under i figurene 6.3 illustreres resultatene etter gjennomføringen av testene.



Figur 6.3: Viser sammenligning mellom referansehastigheten og den målte hastigheten ved bruk av ulike filterverdier

Ved å justere alpha-verdien i filteret vårt, kan vi kontrollere hvor raskt systemet vårt reagerer på endringer, og hvor følsomt det er for støy.

En alpha-verdi på 0.01 vektlegger tidligere målinger mer, noe som resulterer i en betydelig glatting av hastigheten. Dette kan gjøre systemet mindre sensitivt for støy, men det kan også medføre en tregere respons fra regulatoren på grunn av den økte vektleggingen av tidligere verdier.

Når alpha-verdien økes til 0.1, begynner filteret å vektlegge den nåværende målingen mer og tidligere målinger mindre. Dette reduserer graden av glatting, noe som kan gjøre regulatoren raskere til å reagere på endringer, men også mer følsom for støy.

Med en enda høyere alpha-verdi på 0.9, legger filteret nesten all vekt på den nåværende målingen, med minimal vekt på tidligere målinger. Mens dette gjør systemet svært følsomt for endringer i hastighet, kan det også gjøre det mer utsatt for støy, noe som kan føre til betydelige og potensielt uønskede svingninger i regulatorens respons.

Valget av alpha-verdi er direkte avhengig av systemets dynamikk og forventet støynivå i hastighetsmålingene. I vårt system beregnes hastigheten som den deriverte av posisjonen, og posisjonen igjen beregnes basert på enkoder counts. Dermed oppdateres hastigheten hyppig, noe som fører til en betydelig grad av støy i målingene. På grunn av den store graden av støy, krever systemet vårt en lav

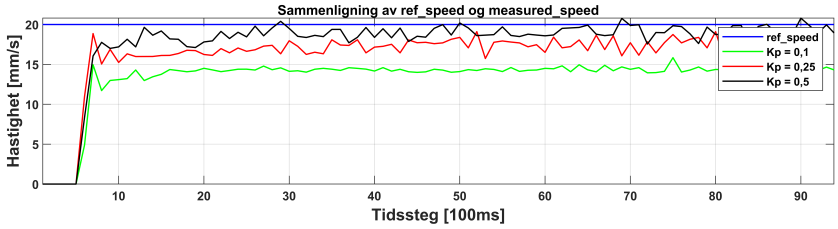
alpha-verdi for å sikre glatting og dermed forbedre nøyaktigheten og stabiliteten i systemets respons.

6.3.2 ”Prøv og feil”- Tuning for PI-regulator

For å finne parametere til PI-regulatoren ble ”prøv og feil”-tuning metoden benyttet. Som forklart i kap 4 har frekvensomformeren en integrert ASR-regulator(Automatic speed regulator). Vi gjennomførte derfor tester for å se om en P-regulator i kombinasjon med våre standardinnstillinger, vil være tilstrekkelig for å oppfylle kravene til hastighetsreguleringen.

Valg av Kp

For å finne en passende Kp verdi, ble Ki verdien satt lik 0 og det ble kun benyttet en P-regulator. Under er det vist 3 ulike verdier testet på det simulerte systemet. Testene ble utført med et sprang fra 0mm/s til 20mm/s.



Figur 6.4: Kp=0.1 og Ki = 0.1

Basert på testresultatene, observerer vi at den målte innspolingshastigheten i alle tre testene er lavere enn referansehastigheten. Dersom vi hadde ytteligere økt kp-verdien til P-regulatoren, ville dette ha resultert i målinger som er preget av betydelig støy.

Ved en Kp-verdi på 0.1, kan man se fra det grønne signalet at den faktiske innspolingshastigheten ligger godt under den ønskede hastigheten. Gjennomsnittshastigheten beregnet fra testen var 14.52 mm/s, men det er fortsatt et betydelig avvik fra ønsket hastighet.

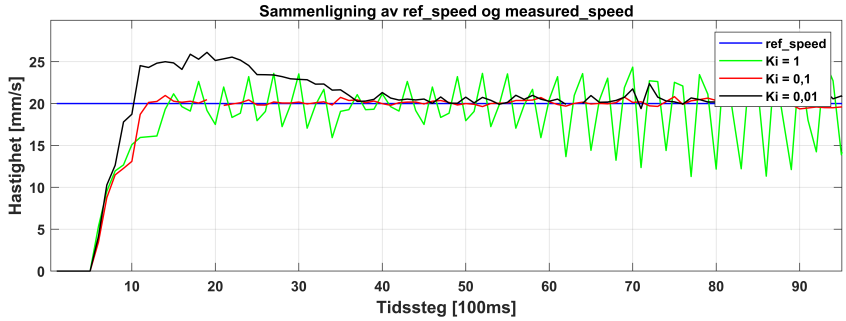
Når Kp-verdien øker ytterligere til 0.25, viser det rød signalet at det tar inn mer støy. Gjennomsnittshastigheten beregnet fra testen er 17.21 mm/s, som er nærmere den ønskede innspolingshastigheten, men signalet er mer preget av støy.

Når Kp-verdien ytterligere øker til 0.5, viser det sorte signalet at det igjen blir mer støyete. Gjennomsnittshastigheten beregnet fra testen er 18.71 mm/s, som er nærmere den ønskede innspolingshastigheten, men signalet er enda mer preget av støy.

Testene viser at lave Kp-verdier fører til en lavere innspolingshastighet og et større avvik fra den ønskede hastigheten. Derimot vil en lavere Kp verdi gi betraktelig mindre støy noe som er et viktig krav for vårt system. En ren P-regulator er ikke tilstrekkelig for dette systemet, siden den ikke klarer å oppnå den ønskede innspolingshastigheten uten støy.

Valg av Ki

For å finne en passende Ki-verdi, utførte vi en rekke tester. Resultatene fra testene, med forskjellige Ki-verdier, er presentert i figuren nedenfor **??**.



Figur 6.5: Kp = 0.1 og Ki = 0.1

Den sorte linjen viser at med en Ki på 0.01 resulterer i et betydelig oversving i starten, opptil omtrent 25 mm/s. Etter 4 sekunder nærmer innspolingshastigheten seg den ønskede hastigheten. Imidlertid er målet å minimere oversving, og derfor ble denne Ki-verdien ikke brukt.

Den rød grafen viser at med en Ki på 0.1, er oversvinget ved oppstart minimal og hastigheten justerer seg raskt til ønsket nivå. Innspolingshastigheten gjennom testen ligger generelt nær den ønskede hastigheten.

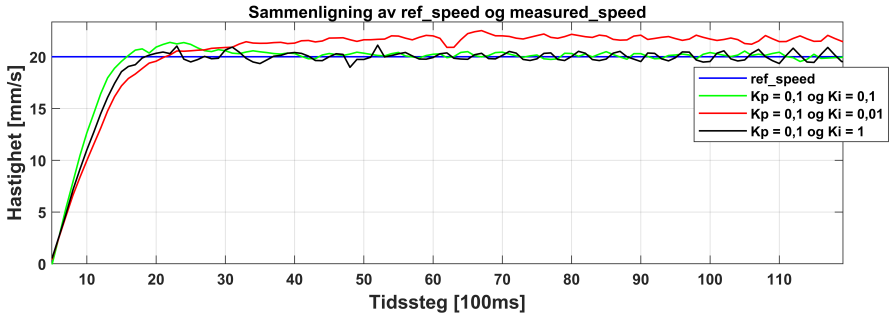
Grafen i grønt demonstrerer at ved å sette Ki-verdien til 1, blir hastigheten ujevn. Dette fører til at lasten heises i en rykkende bevegelse. Disse svingningene oppstår som en konsekvens av en overdreven akkumulering i det integrerende leddet.

Basert på grafene ovenfor, kan vi konkludere med at en Ki-verdi på 0.1 vil være det beste valget for en I-regulator i vårt system.

6.3.3 Valg av Kp og Ki kombinasjon

For å identifisere en optimal kombinasjon av Kp- og Ki-verdiene, utførte vi en rekke tester med varierte verdikombinasjoner. Det ble tatt utgangspunkt i verdiene som de tidligere testene hadde indikert som de beste.

De første testene ble utført med fast Kp-verdi på 0.1. Resultatet fra testene med 3 forskjellige Ki-verdier er vist under i figur 6.6.



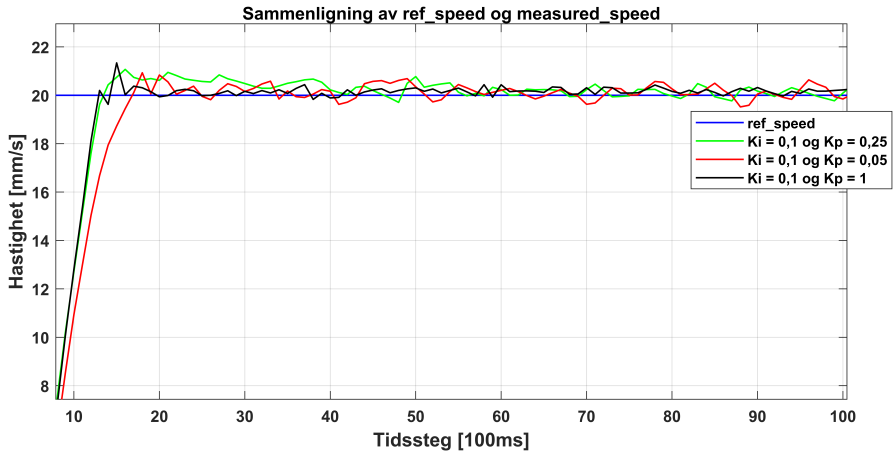
Figur 6.6: Kp = 0.1 og Ki = 0.01

I figur 6.6 ser vi fra den røde grafen at med Ki-verdi på 0.01 legger den målte hastigheten seg over den referansehastigheten. Den grønne grafen viser så at en Ki på 0.1 fører til at den målte hastigheten vil ha en oversving fra start, men over tid legger seg rundt referansehastigheten.

Den sorte grafen viser test med en Ki på 1. Det blir observert at denne verdien

fører til at den målte hastigheten også holder seg rundt referansehastigheten, men er mer preget av støy.

Basert på observasjonene fra testene, ble det valgt å videre teste forskjellige Kp-verdier med en Ki på 0.1. Resultatene fra testene er vist under i figur 6.7.



Figur 6.7: Ki = 0.1 og Kp = 1

Den røde linjen i delfigur i Figur 6.7 viser en kombinasjon av en Kp 0.05 og Ki 0.1. Ved starten er det en oversving til 21 mm/s, men denne reguleres til ønsket hastighet i løpet av omtrent 2 sekunder. Innspolingshastigheten er stabil, og det er lite avvik mellom ønsket og målt hastighet. Den grønne linjen representerer en Kp verdi 0.25 og Ki på 0.1. Her er det tydelig at andre Kp-verdier gir bedre resultater grunnet dårlig følging av referanselinjen.

Den sorte linjen viser kombinasjonen av en Kp på 1 og Ki på 0.1.Den sorte linjen viser en oversving i begynnelsen til 21 mm/s, men justerer seg raskt tilbake til den øsnkede hastigheten. Avviket mellom den ønskede og den faktiske hastigheten er minimal, og hastigheten viser lavt støynivå. Basert på testene i dette kapittelet, kom vi frem til at Kp-verdien på 1 og Ki-verdien på 0.1 best oppfylte våre kriterier for hastighetsregulering. Derfor ble verdiene Kp på 1 og Ki på 0.1 valgt for videre bruk i PI-regulatoren.

6.4 Skogestads metode for tuning av PI-regulator

Gjenntatt fra kapittelet om posisjonsregulering er skogestad's metode en metode for å optimalisere parametrene til en PID-regulator for å oppnå ønsket dynamisk respons i et reguleringssystem.

Formålet med å bruke Skogestads tuningmetode i hastighetsreguleringen var igjen å undersøke muligheten for å finne hensiktsmessige PID-verdier.

I dette kapittelet blir parameterene Kp og Ki funnet ved hjelp av Skogestads metode. Parameterene blir testet i PI-regulatoren og sammenliknet med verdiene vi fant ved hjelp av "prøv og feil"- metoden.

Utregning av parametere

Tabellen for skogestads-tuningmetode er vist i kapittelet om posisjonsregulering i tabell 5.3. Vi benytter likningene for første orden vist under i tabell 6.1.

Tabell 6.1: Tabell for skogestad tuning av hastighetsregulator

$H_p(s)$ (process)	K_p	T_i	T_d
$\frac{K}{Ts+1}e^{-\tau s}$	$\frac{T}{K(T_C+\tau)}$	$\min[T, k_1(T_C+\tau)]$	0

Videre blir verdiene for skogestads tuning regnet ut. Utrengingene er vist under:

$$Kp_{seriell} = \frac{T}{K(T_C+\tau)} = \frac{0.15}{13.33(0.15+0.15)} = 0.0375 \tag{6.3}$$

$$Ti_{seriell} = \min[T, k_1(T_C+\tau)] = 1.44 \cdot (0.15+0.15) = 0.432 \tag{6.4}$$

$$Ki_{seriell} = \frac{Kp_{seriell}}{Ti_{seriell}} = \frac{0.0375}{0.432} = 0.0868 \tag{6.5}$$

Vi benytter også en parallell PI regulator for hastighetsregulering, og som gjen-

ntatt fra kapittelet om posisjonsreguleringen er det nødvendig å omgjøre de parallelle verdiene til serielle. Utregningen for omgjøringen er vist under:

Skogestads verdier gjøres om fra seriell til parallell

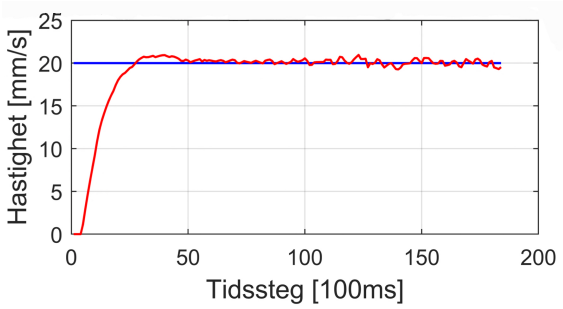
$$Ti_{parallell} = Ti_{seriell} \cdot (1 + \frac{\tau}{Ti_{seriell}}) = 0.432 \cdot (1 + \frac{0.155}{0.432}) = 0.582 \tag{6.6}$$

$$Kp_{parallell} = Kp_{seriell} \cdot (1 + \frac{\tau}{Ti_{seriell}}) = 0.0375 \cdot (1 + \frac{0.155}{0.432}) = 0.0505 \tag{6.7}$$

$$Ki_{parallell} = \frac{Kp_{parallell}}{Ti_{parallell}} = \frac{0.0505}{0.582} = 0.0868 \tag{6.8}$$

6.4.1 Testing av PI-regulator ved bruk av Skogestads tuning

Under i figur 6.8 vises resultat av et sprang fra 0 til 20mm/s ved bruk av verdiene funnet ved hjelp av skogestad.

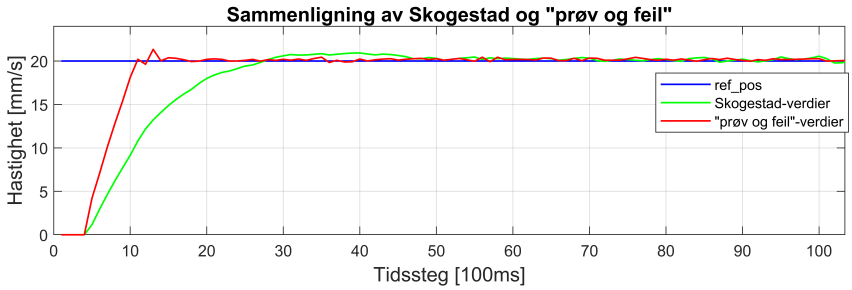


Figur 6.8: Hastighetsregulering ved bruk av skogestads tuningverdier. Kp = 0.0505, Ki = 0.0868

Den røde linjen i delfigur "Sammenligningn av ref_speed og measured_speed" viser en kombinasjon av Kp = 0.0505 og Ki = 0.0868. Etter 3.5s er det en oversving til 21 mm/s, men dette reguleres til ønsket hastighet etter 2 sekunder kjøring. Innspolingshastigheten er stabil, og det er lite avvik mellom ønsket og målt hastighet. Som bevist fra denne testen vil skogestad-tuning fungere for hastighetsregulering av systemet.

6.5 Testing av PI-regulator

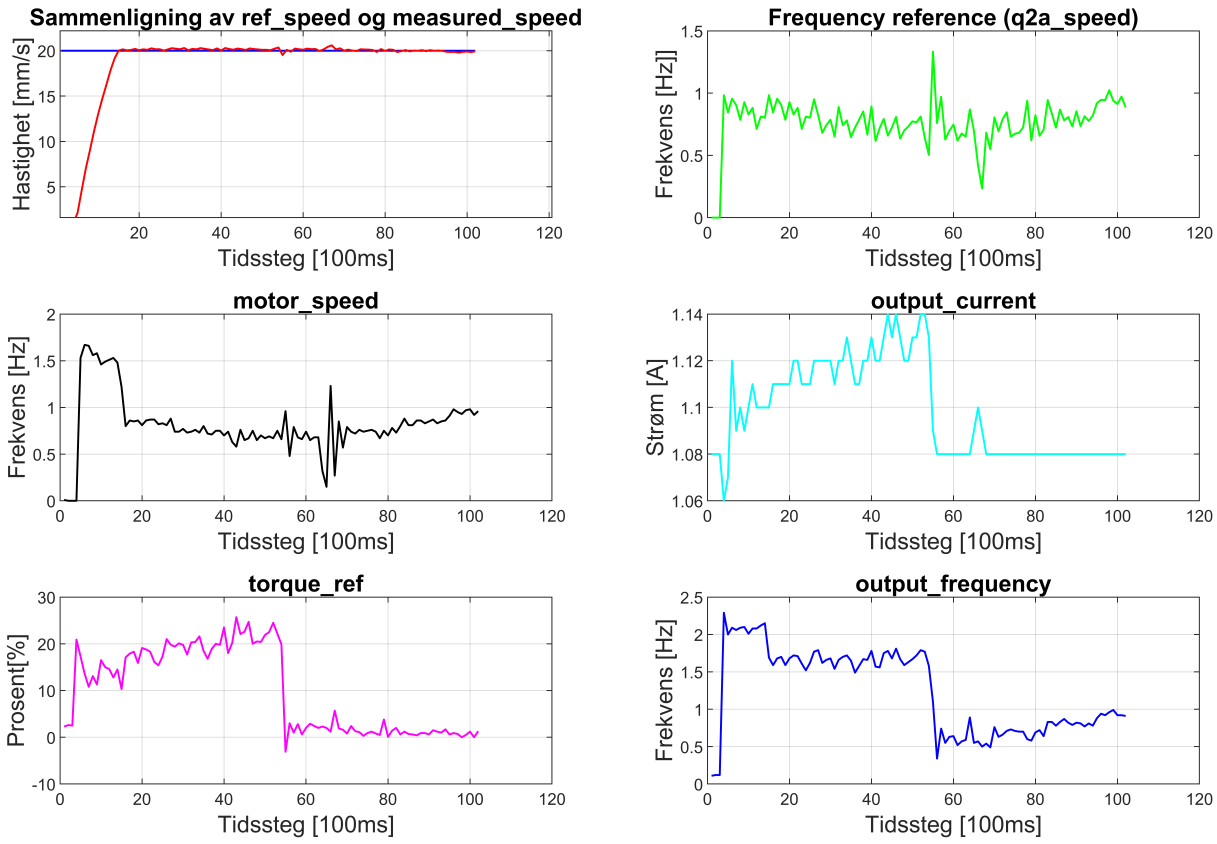
Etter godkjente parameterverdier fra ”prøv og feil” og ”skogestads-tuning”, ble verdiene sammenlignet i en test hvor hastigheten fikk et sprang fra 0 til 20 mm/s. Resultatene i testen er vist under i figur 6.9



Figur 6.9

Sammenlignet med verdiene fra ”prøv og feil” vil skogestads-tuningverdier ha en tregere respons, og vi valgte derfor videre testing med verdiene fra ”prøv og feil”.

Etter sammenligning av et sprang mellom ”prøv og feil” og ”skogestad” ble det utført ny test på systemet. Her ble PI-regulatoren kjørt opp med en konstant hastighet på 20mm/s i 5 sekunder hvor den så snudde og kjørte ned i 5 sekunder. Resultatet av testen er vist under i figur



Figur 6.10: Ki = 0.1 og Kp = 1

I delfiguren "frequency reference(q2a_speed)" blir variasjonene i referansen som sendes til den interne hastighetsregulatoren illustrert. Referansen satt ved hjelp av våre koder, forblir stort sett stabil mellom 0.5 og 1 Hz under kjøring. Likevel, ved retningsendringer, er justeringer i utgangsfrekvensen nødvendige, noe som fører til endringer i referansen.

Fra delfiguren "Output Frequency" ser vi at utgangsfrekvensen må øke for å oppnå den ønskede hastigheten. Ettersom at motoren har nådd ønsket hastighet(etter 1,7s) dempes utgangsfrekvensen til rundt 1.6-1.7 Hz. Etter retningsendringen faller utgangsfrekvens kraftig. Dette skyldes at gravitasjonen hjelper under senking

og momentet blir lavere. Derimot når lasten senkes, må utgangsfrekvensen fortsatt øke. Dette skyldes at motorhastigheten (motor_speed) må økes for å oppnå den ønskede hastigheten på grunn av en stegvis minkende total radius.

Disse grafene, samlet sett, bekrefter at systemet oppfører seg som forventet under drift, selv om det er noen mindre forstyrrelser ved retningsendringer. Generelt sett er ytelsen solid, men det kan være potensial for ytterligere forbedringer, spesielt med hensyn til å redusere støy og forstyrrelser ved retningsendringer.

Kapittel 7

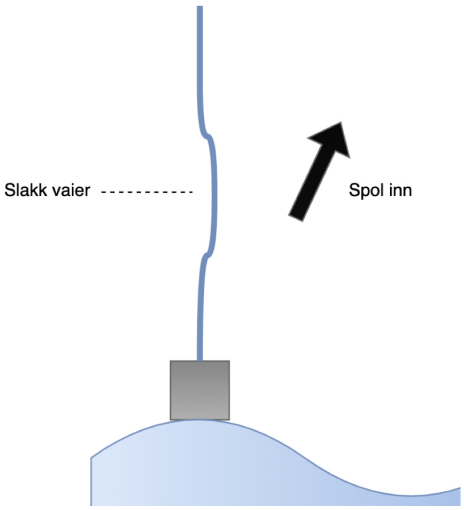
Momentregulering

Som repetert fra kapittel 1 under problemstillingen 1.1, tar oppgaven vår for seg det kritiske øyeblikket når livbåter senkes fra plattform eller skip i høye bølger. Vi ønsker å opprettholde en konstant trekkraft i livbåtvaieren etter at båten først berører bølgene, slik at vi unngår rykk eller slakk vaier i store bølgehøyder. Dette oppnås gjennom rask inn- og utspoling av vaieren. Hovedutfordringen ligger i å regulere kreftene i vaieren når livbåten treffer vannoverflaten samtidig som skipet er i bevegelse. En mulig løsning på dette er å benytte momentregulering.

I dette kapittelet vil vi undersøke interaksjonene mellom båten, vaieren og det ustabile havmiljøet. Vi vil presentere teorien bak vår tilnærming, utforske de kodene vi har utviklet for å regulere systemet, og til slutt vil vi diskutere resultater fra tester og analysere effekten av vår løsning.

7.1 Teori

Når en båt treffer vannoverflaten og løftes/senkes av bølgene, kan det føre til rykk eller slakk i vaieren. Dette kan medføre i uønskede situasjoner. For å løse dette problemet ble momentregulering implementert. Med å finne gode verdier til en intern regulator, vil motoren øke pådraget hvis slakk i tauet er oppdaget, og motsatt hvis spenningen i tauet blir for stor. Prinsippene bak denne reguleringen er illustrert i figur 7.1

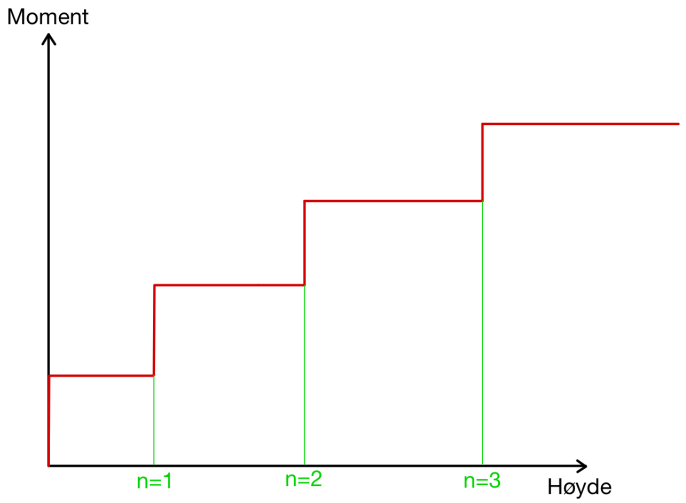


Figur 7.1: [9]Figuren demonstrerer at i tilfelle av en slakk vaier, vil momentregulatoren aktiveres og tilføre kraft til motoren for å håndtere overskuddet. Denne mekanismen muliggjør effektiv justering i takt med bølgebevegelsene.

I kapittel 2.1 blir det forklart at lastmomentet vil øke stegvis ut ifra antall omdreininger med tau på trommelen. Med andre ord, ettersom tauet vikles rundt trommelen og flere lag legges til, kreves det stegvis høyere momentreferanse for hver påfølgende omdreining, til å holde loddet på en bestemt høyde. Formelen for å utlede lastmomentet blir repetert i ligning 7.1 og figuren 7.2 illustrer forholdet mellom moment og høyden til lasten.

$$T_{last} = m \cdot g \cdot r_{total} \tag{7.1}$$

$$r_{total} = r_{trommel} + d_{vaier} \cdot n \tag{7.2}$$



Figur 7.2

Figuren ovenfor illustrer hvordan antall runder(n) stegvis endrer kravet til momentet for å holde en last på ulike høyder. I tillegg viser figuren at høydeintervallene mellom antall runder blir stegvis lengre. Dette er et resultat av økt radius som forklart i kapittel 2.5.1.

7.2 Q2A-innstillinger til momentregulering

Standardinnstillingene vist i tabell 2.2 ble benyttet i Q2-edit under kalibrering, posisjonsregulering og hastighetsregulering har benyttet hastighet som referanse(hastighetsregulering/speed control). I dette kapitellet var det nødvendig å endre hastighetsregulering til den interne momentreguleringen i frekvensomformeren. For å kunne aktivere momentkontroll i frekvensomformere er det nødvendig å endre parametere i Q2-edit. Listen over de nødvendige endringene av innstillingene er vist under i liste 7.2.

- **d5-01 : Torque ctrl selection**
  Velger mellom momentregulering (1) og hastighetsregulering (0). Denne ble satt til 1.

- **F6-06 : Trq Ref/Lim Comms**
  Parameteren setter funksjonen som aktiverer og deaktiverer muligheten til å endre momentreferansen. Innstillingen ble satt til 1 (enable). Når F6-06 er det mulig å bestemme "external torque reference" vist i vedlegg B.

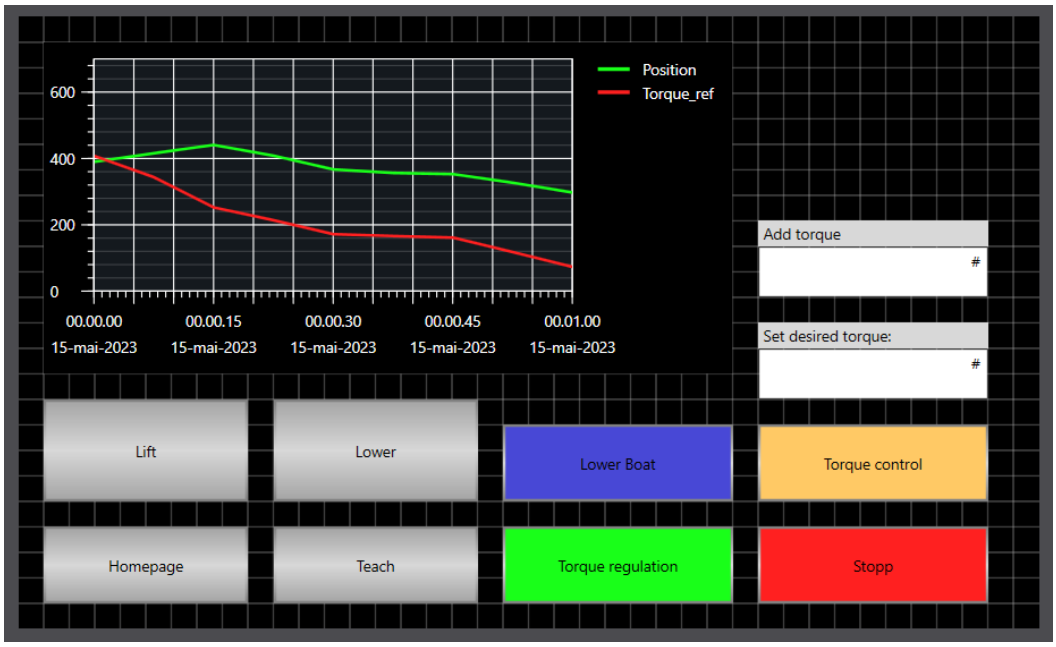
For å senke båten til vannet ved hjelp av konstant hastighet og derretter benytte momentregulering, var ideen å bytte reguleringsmetode når båten treffer vannet. Ved å benytte hastighetsreguleringen forklart i kapittel 6, kunne vi kjøre ned med en bestemt hastighet, hvor så brukeren kunne bytte til momentregulering når båten treffer vannet.

Underveis i prosjektet ble det oppdaget et problem som ikke gjorde det mulig å endre aktivt mellom hastighetsregulering og momentregulering. Bryterpanelet som bestod av bryteren styrt av parameter "**H1-01 DI1 Function selection -> Spd/Trq switch**", som gjorde det mulig å endre reguleringsmetode under kjøring, sluttet å fungere. Bytte mellom reguleringene var derfor kun mulig når motoren var av. Derfor var momentkontroll valgt av parameteren "**d5-01 -> Torque control**" aktiv under hele kjøringen.

7.3 Manuell kjøring

7.3.1 Koder

I dette delkapittelet presenteres kodene for manuell kjøring under momentkontroll. For å forbedre muligheten til å overvåke og justere kodens verdier, ble HMI-skjermen, presentert i Figur 7.3, implementert.



Figur 7.3: Bilde over HMI-skjermen benyttet under momentregulering.

HMI-skjermen inneholder 2 DataEdit-viduer og en Trend Graph. Trend Graphen brukes til å observere systemets momentreferanse og posisjonen til loddet. DataEdit-viduene benyttes til å bestemme verdier benyttet i kodene.

HMI-skjermen har også 4 fargede knapper: Lower Boat, Stop, Torque control og Torque regulation. Funksjonene til disse er beskrevet nedenfor:

- Torque control → Fastsetter den siste målte momentreferansen.
- Lower Boat → Når denne er aktivert reduseres momentrefansen kontinuerlig,

noe som åpner for manuell kjøring. Om momentreferansen øker eller synker blir bestemt av knappene "lift" og "lower".

- Torque regulation → Dette er knappen som aktiverer momentkontroll.
- Stop → Nødstopp.

For å verifisere at teorien stemmer med det fysiske systemet så ble det implementert koder i PLSen som skulle stegvis øke momentet frem til maks høyde (750mm) var nådd, og motsatt til havoverflaten(160mm) var nådd. Kodeutdraget er vist i listen 7.1

Kode 7.1: Låring av båt

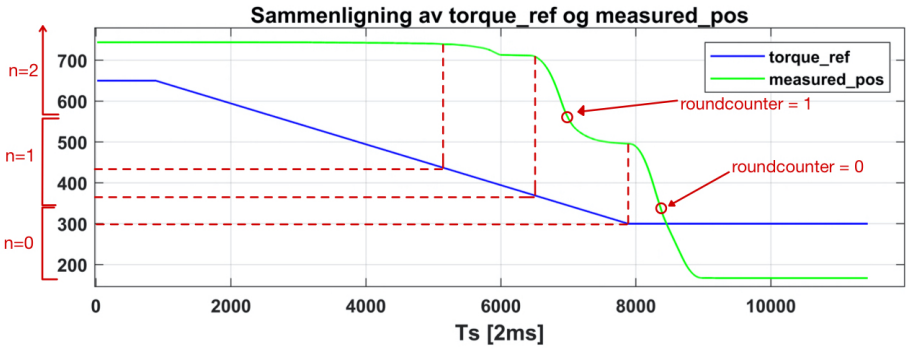
```
62 IF manual_boat = TRUE THEN
63 add_torque:=HMI_add_torque;
64 IF lift THEN
65     control:=previous_control+add_torque;
66 END_IF;
67
68 IF lower THEN
69     control:=previous_control-add_torque;
70 END_IF;
71 q2a_torq_ref_wright:=LREAL_TO_INT(control);
72 previous_control:=control;
73 last_torque_value:=q2a_torq_ref_wright;
74 END_IF:
```

Når variabelen "manual_boat" i linje 62 var aktiv kunne vi løfte og senke båten ved å stegvis legge til eller trekke fra moment. I linje 63 velges en verdi som kontinuerlig multipliseres med tidsskrittet. Linjene 64-70 bestemmer om momentet skulle øke eller reduseres. Når variabelen "lift" er aktivert, adderes produktet med "control" -variabelen, og trekkes fra når variabelen "lower" er aktivert. i linje 71 brukes "control"-variabelen til å bestemme momentreferansen("q2a_torq_ref_wright"). I linje 73 blir "last_torque_value" kontinuerlig oppdatert til sist satte momentreferanse. Bruken av "last_torque_value" vil bli forklart senere i oppgaven.

Ved hjelp av denne koden kunne vi gjennomføre en test hvor vi kjørte loddet fra havoverflaten (160mm), som ble simulert ved hjelp av skumgummi, til maks høyde (750mm) og derretter ned igjen til havoverflaten. Det ble lagt inn en øvre momentgrense på 650, for å sikre at kjøringen ikke ville skade systemet. For å hindre uønsket friksjon mellom trinsene, så gjennomførte vi testen med kun 1 trinse.

7.3.2 Resultat fra test

Figur 7.4 viser resultatet fra å låre loddet fra maks høyde til havoverflaten og hvordan roundcounter(n) påvirker lastmomentet.



Figur 7.4

Som illustrert i figuren, blir det vist hvordan posisjonen til loddet beveger seg ut ifra endringen i momentreferansen(motormomentet) og lastmomentet. Momentreferansen vil stegvis synke med 2,5 per tidssteg, og lastmomentet vil endres ut ifra formelen 7.1. Som det ble forklart i kapittel 3, i seksjonen med kodeutdrag 3.3, vil "roundcounter" bli oppdatert hver gang motoren fullfører en hel omdreining, det vil si når enkoderen har registrert 8000 pulser. I delkapittel 3.1 ble det forklart hvordan omkretsen øker for hver runde. Omkretsen ble benyttet til å regne ut høydeområdet for hver runde. Formelen for omkretsen er gjengitt i ligningen 7.3.

$$\text{Omkrets}(n) = 2\pi \cdot (r_{trommel} + (d_{vaier} \cdot n)) \tag{7.3}$$

Under er det gitt en oversikt over de ulike høydeområdene for hver runde som ble registrert i testen. Disse høydeområdene er illustrert på venstre side i figuren 7.4, på venstre side.

- roundcounter(n) = 2 → h = [556mm, 809mm]
- roundcounter(n) = 1 → h = [340mm, 556mm]
- roundcounter(n) = 0 → h = [160mm, 340mm]

Den midtre stiplete linjen indikerer når motormomentet blir mindre enn lastmomentet, noe som resulterer i at loddet begynner å senke seg. Sirkelen som pilen peker på ved runde 1 symboliserer skiftet av roundcounter fra 2 til 1, og en endring i lastmomentet oppstår. Som et resultat av denne endringen, blir motormomentet nå større enn lastmomentet, og loddets nedstigning bremses. Dette mønsteret gjentar seg når motormomentet nok en gang blir mindre enn lastmomentet.

Gitt den øvre momentgrensen på 650 som ble brukt i testen, vil loddet under oppstigning tendere mot et høydeområde som overstiger maksimal høyde (750mm), når momentreferansen når momentgrensen. Dermed vil momentreferansen fortsette å stige selv etter at maksimal høyde er nådd. Dette forklarer hvorfor momentreferansen i figur 7.4, må reduseres til omtrent 440 før loddet begynner å senke seg.

Vi valgte en momentgrense på 650 fordi det krever mer moment å sette et objekt i bevegelse fra en stillestående posisjon, sammenlignet med å fortsette bevegelsen av et allerede bevegelig objekt. Hvis vi hadde valgt å bruke en fast momentreferanse på 500 i stedet for en gradvis økning i momentreferansen for å løfte loddet, kunne vi ha unngått at loddet ble stående i ro når roundcounteren oppdateres. Imidlertid kunne dette ha medført en risiko for at loddet ble hevet med for høy hastighet opp til maksimal høyde, noe som potensielt kunne ha skadet systemet.

7.4 Resultat momentregulering

7.4.1 Koder

For å analysere oppførselen til den interne regulatoren i varierende bølgebevegelser, ble det implementert koder for regulering rundt en fast momentreferanse. Kodene er vist under i kodelisten 7.2.

I linje 73 fra kodeutdraget 7.1, lagres "last_torque_value" for å bevare informasjon om motorens moment ved berøring med vannoverflaten. Når båten berører vannet, vil motorens moment bli holdt konstant på den sist registrerte momentreferansen. Dette fører til at motoren automatisk justerer seg i respons til bølgene: motoren spoler inn tau når en bølge løfter båten, og slipper ut mer når bølgen faller. Ved å spole inn og ut vaier vil avviket fra motormoment til satt referanse regules. Kodene momentregulering rundt en fast referanse er vist under i kodelisten 7.2

Kode 7.2: Momentregulering

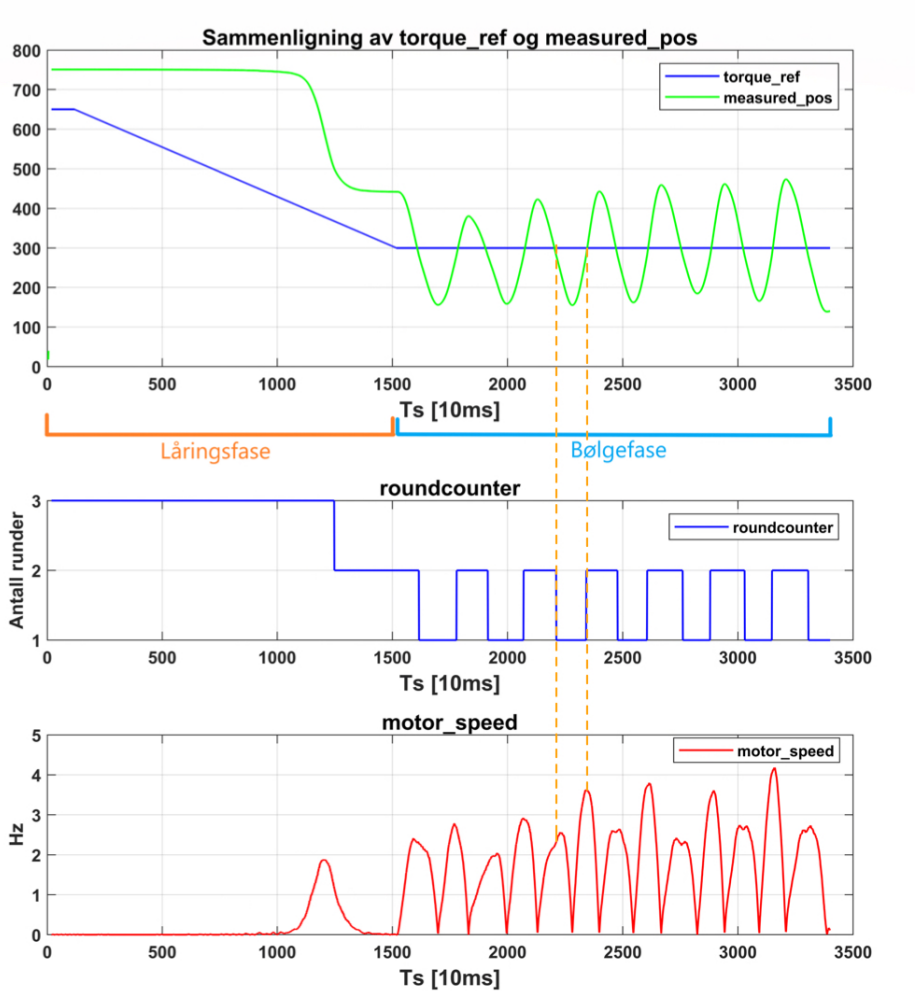
```
81 IF control < 300 AND lower = TRUE THEN
82 torque_control:= TRUE;
83 END_IF;
84
85 IF HMI_torque_reg = TRUE AND torque_control = TRUE THEN
86     manual_boat:=FALSE;
87     q2a_torq_ref_wright := LREAL_TO_INT(last_torque_value);
88 END_IF;
```

Gjennom testing fant vi ut at en momentreferanse på 300 ville være tilstrekkelig for å regulere vinsjen på en måte som opprettholder en stram vaier, uten å løfte loddet eller holde det over vannet. Derfor implementerte vi en endring i nedkjøringsfasen, hvor vi gikk fra en stegvis reduserende momentreferanse til en fast referanse("last_torque_value"), så snart momentreferansen falt under 300. Et automatisk bytte bidro til konsistent momentreferanseendring i testene, noe som forenklet analysene og sammenligningene av resultatene.

I linje 82 blir torque_control aktivert når momentreferansen når er verdi under 300 og variabelen "lower"(kontinuerlig redusering av moment) er aktiv. Linjene 86-87 setter så manual_boat lik FALSE for å avslutte den kontinuerlige reduseringen av moment. Momentreferansen blir videre satt til en fast verdi lik den sist registrerte momentreferansen under låringen.

7.4.2 Resultat fra test

Figuren 7.5 viser resultatet fra en test hvor vi kjørte loddet fra makshøyde og ned til vannet. Gjennomføring av testen kan også ses i en video med Youtube-linken https://youtu.be/5I6jVh3thk8.



Figur 7.5

Fra den øverste delfiguren blir det illustrert hvordan loddet beveger seg i de to ulike fasene av testen (låringsfasen og bølgefasen). Under låringsfasen benytter vi

samme funksjon som vi benyttet i forrige test. Ved omtrent Ts=1500, blir momentreferansen fast på 300 og testen går over i en bølgefase. Under bølgefasen blir loddet løftet opp og ned i en forskjellige bølgebevegelser. Den nederste delfiguren viser hvordan motor_speed øker og synker i samme takt som bølgebevegelsene. Motoren spoler inn tau når en bølge løfter loddet og spoler ut når bølgen faller tilbake. På denne måten vil motoren følge momentreferansen. Den midterste figuren viser hvordan roundcounteren teller opp og ned når bølgene beveger loddet opp og ned. Siden momentreferansen på 300 ikke vil være høyere enn lastmomentet, selv ved en roundcounter på 0, så vil ikke en endring i roundcounteren påvirke momentreguleringen.

Den øverste figuren illustrerer loddets bevegelsesmønster gjennom testens to faser: låringsfasen og bølgefasen. I løpet av låringssfasen brukes samme funksjon som i den tidligere testen. Ved tidspunktet Ts=1500, blir momentreferansen fast satt til 300, og testen overgår deretter til bølgefasen. I denne fasen løftes og senkes loddet gjennom ulike bølgebevegelser.

I den nederste figuren blir det vist hvordan motorhastigheten justeres i takt med disse bølgebevegelsene. Motoren trekker inn tauet når bølgen løfter loddet, og gir slipp på det når bølgen reduseres. Dette gjør at motormomentet reguleres etter momentreferansen.

Midtfiguren gir en visuell fremstilling av hvordan ”roundcounter” stiger og faller i takt med loddets bølgebevegelser. En endring i roundcounter", og dermed lastmomentet, fører til redusert nødvendig pådrag for å oppnå det ønskede momentet. Dette skyldes en endring i differansen mellom lastmomentet og motormomentet. Itillegg vil motormomentet som kreves, når en bølge løfter loddet, være større enn momentet som kreves, når en bølge senker loddet. Dette blir illustrert ved å sammenligne ”motor_speed” når ”measured_pos” øker og faller.

Kapittel 8

Diskusjon

I det følgende kapittelet vil vi dykke dypt ned i diskusjonen og refleksjonen rundt de mest sentrale funnene og utfordringene vi har møtt på underveis i denne bacheloroppgaven. Vi reflekterer uforutsette problemer og svakheter som dukket opp, og undersøker mulige løsninger som kunne blitt anvendt. Til slutt vil vi se fremover, og vurdere potensialet for videre utvikling og forbedringer av systemet.

8.1 Sentrale funn

Gjennom grundige tester av systemets komponenter, oppnådde vi en dypere forståelse av hvordan teorien korresponderte med praksis. En enkel variabel som hadde betydelig innvirkning på vårt system, var diameteren på tauet. Tauet hadde en diameter på 6 mm, men ble påvirket av tvinn og press, samtidig som det ikke la seg perfekt oppå tauet i trommelen. Etter en rekke tester, kom vi frem til at en justert verdi på 5.8 mm for tauet kompenserte for avvikene og samsvarte mer nøyaktig med de fysiske målingene.

Vi observerte også systemresponser i systemet som ikke stemte overens med våre eksisterende koder og tidligere observasjoner. Ved å undersøke disse atferdene grundig, avdekket vi viktigheten av de justerbare innstillingene i frekvensomformeren. Gjennom detaljert analyse av parametere, kunne vi forstå systemets potensielle oppførsel og dermed utforme koder og finne parametere som var i samsvar med den faktiske atferden.

8.2 Feil og mangler

I løpet av oppgaven ble det oppdaget at panelet frekvensomformeren var koblet til, hadde sluttet å fungere. Dette resulterte i at aktiv bytte mellom moment- og hastighetsregulering ikke lenger var et tilgjengelig alternativ. Vi var derfor nødt til å gjennomføre alle tester av momentregulering uten tilgang hastighetsreguleringen.

Frekvensomformeren sin keypad og skjerm var dessverre også defekt. Dette betydde at eventuelle innstillinger som tidligere kunne justeres direkte via skjermen, nå måtte endres gjennom programvaren Q2-Edit.

8.3 Videreutvikling

Hvis vi hadde hatt mer tid tilgjengelig, ville vi ha gjennomført tester på flere av Q2A-parameterene for å forstå deres påvirkning på vinsjsystemet. Spesielt ville vi ha undersøkt flere av kontrollparametrene (vedlegg B) for den innebygde momentregulatoren, med mål om å undersøke hvordan disse påvirker systemets respons.

For videreutvikling av systemet ville vi ha undersøkt om en mulig forbedring kunne vært å kombinere posisjonsregulering og hastighetsregulering, i håp om at dette ville resultere i et mer sofistikert system som ikke bare overvåker og kontrollerer den nåværende posisjonen, men også sikrer en konstant innspolingshastighet.

Vi ville også ha undersøkt mulighetene for å implementere både posisjons- og hastighetsregulering gjennom bruk av momentkontroll. Dette ville involvert dynamisk justering av momentreferansen basert på potensielle avvik, samt å ta i betraktning endringer i momentet, slik at både hastighet og posisjon kunne vært styrt ved å benytte momentet som referanse for frekvensomformeren.

Kapittel 9

Konklusjon

I vår bacheloroppgave har vi vellykket kalibrert og kontrollert et vinsjsystem ved hjelp av en frekvensomformer. Dette ble realisert gjennom bruk av en Omron-demomodell, hvor vi utledet en matematisk modell for vinsjen og implementerte denne i PLS-programvaren Sysmac Studio.

I bacheloroppgaven har vi gjennomført detaljerte analyser av de innebygde parameterne i frekvensomformeren. Gjennom denne prosessen har vi opparbeidet en dypere forståelse av systemets dynamikk, noe som har gitt oss verdifull innsikt i årsakene til systemets oppførsel. Denne kunnskapen har vært sentral for vårt arbeid og bidratt til kvaliteten på de endelige resultatene.

I løpet av vår bacheloroppgave har vi oppnådd presis posisjonskontroll og hastighetskontroll av vinsjsystemet. Ved å bruke Skogestads metode og en 'prøv og feil'-tilnærming, identifiserte vi de nødvendige kontrollparametrene for både PD- og PI-regulatorene. Disse regulatorene ble identifisert og finjustert gjennom omfattende tester og analyser. Ved å evaluere systemresponsen fra disse metodene, kunne vi bekrefte at begge regulatorene oppfylte de ønskede funksjonalitetskriteriene, som inkluderte ingen eller minimal oversving, rask respons, og effektiv lastkompensering.

Vi lyktes også i å implementere vellykket momentkontroll av vinsjsystemet. Momentreguleringen fungerte som ønsket ved å vedlikeholde en konstant spenning i vaieren. For å kompensere for bølgebevegelser, ble det utført rask inn- og utspoling av vaieren. Dette arbeidet illustrerer et vellykket eksempel på hvordan en

teoretisk modell har blitt anvendt i en praktisk kontekst. Vår suksess med momentregulering av demomodellen demonstrerer potensialet for å overføre teoretisk kunnskap til praktiske, tekniske løsninger.

Bibliografi

[1] Q2a frekvensomformer. https://industrial.omron.no/no/products/Q2A-A2012-AAA.pdf, 2023. "Hentet: 01-15-2023".

[2] ABB. Teknisk veiledning nr. 4 veiledning for frekvensomformere for hastighetsstyring. https://library.e.abb.com/public/06961a5060b74233c125795b002b9029/NO_Technical_guide_No.4_REVC.pdf, 11 2011. Hentet: 02-01-2023".

[3] ABB. Teknisk veiledning nr. 7 dimensjonering av frekvensomformer og motor. https://library.e.abb.com/public/ab94567b522bad0ac125795b002cb987/NO_Technical_guide_No.7_REVC.pdf, 2011. "Hentet: 02-01-2023".

[4] Omron Industrial Automation. E6c2-c incremental rotary encoder datasheet. https://assets.omron.eu/downloads/datasheet/en/v5/q109_e6c2-c_incremental_rotary_encoder_datasheet_en.pdf, 2022. "Hentet: 01-15-2023".

[5] BEVI. 4ak 712-4 electric motor. https://www.bevi.com/electric-motors/three-phase-standard-motors/electric-motor-4ak-712-4-112130, 2022. "Hentet: 02-05-2023".

[6] D. Collins. When are closed-loop and open-loop vector control useful? https://www.motioncontroltips.com/faq-when-are-closed-loop-and-open-loop-vector-control-useful/, 2016. "Hentet: 02-20-2023".

[7] A.I. Gundersen and N. Mahesan. Posisjonsregulering og hastighetsregulering av en vinsj. https://uis.brage.unit.no/uis-xmlui/handle/11250/2774435, 2021. "Hentet: 01-10-2023".

[8] Arne Gylseth. Asynkronmotoren. https://docplayer.me/23258484-Asynkronmotoren-arne-gylseth-stator-med-roterende-statorvikling-n3-kortsl html. "Hentet: 03-10-2023".

[9] Stig Haaland. Momentregulering ved bruk q2a frekvensomformer. https://uis.brage.unit.no/uis-xmlui/handle/11250/181614/browse?type=author&value=Haaland%2C+Stig, 2022. "Hentet: 01-10-2023".

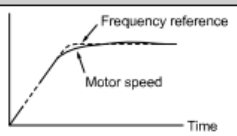
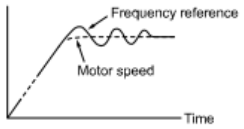
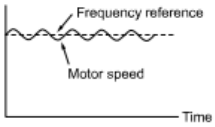
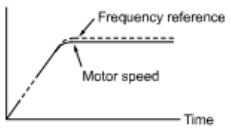
[10] [Omron]. *Q2A Brukermanual*, [2018].

[11] K. A. Rosvold. Asynkronmotor. https://snl.no/asynkronmotor, 2020. "Hentet: 03-010-2023".

Vedlegg A

ASR tuning instruksjon

Table 12.18 ASR Response and Possible Solutions

Problem		Possible Solutions
Speed response is slow.		• Increase *C5-01/C5-03 [ASR PGain 1/ASR PGain 2]*. • Decrease *C5-02/C5-04 [ASR ITime 1/ASR ITime 2]*.
Overshoot or undershoot occurs at the end of acceleration or deceleration.		• Decrease *C5-01/C5-03*. • Increase *C5-02/C5-04*.
Vibration and oscillation occur at constant speed.		• Decrease *C5-01/C5-03*. • Increase *C5-02/C5-04*. • Increase *C5-06 [ASR Delay Time]*.
Speed accuracy is unsatisfactory when you operate a motor that has a large quantity of rated slip in Closed Loop V/f Control.		• Check the pulse number set to *F1-01 [Enc1 Pulse Count (PPR)]* and the gear ratio to *F1-12 [Enc1 Gear Teeth1]* and *F1-13 [Enc1 Gear Teeth2]*. • Make sure that you correctly set the pulse signal from the encoder. • Check *U6-04 [ASR Output]* to make sure that the ASR operates at its output limit set to *C5-05 [ASR Limit]*. If the ASR is at the output limit, increase *C5-05*.
If *C5-12 = 1* or *C5-32 = 1 [Enabled]* in Closed Loop V/f Control and over/undershoot occurs when you change speeds.	-	• Decrease *C5-01/C5-03*. • Increase *C5-02/C5-04*. • Decrease the value set to *C5-05*.
Oscillation at low speed and response is too slow at high speed. Oscillation at high speed and response is too slow at low speed.	-	• Closed Loop V/f Control Mode: Use *C5-03* and *C5-04* at maximum speed and *C5-01* and *C5-02* at minimum speed to set different ASR settings. • Closed Loop Vector Control, PM Advanced Open Loop Vector Control, and PM Closed Loop Vector Control: Use *C5-01* to *C5-04* to set the best ASR settings for high and low speed. Use *C5-07 [ASR Gain Switch Frequency]* to switch the ASR proportional gain and ASR integral time as specified by the output frequency.

Figur A.1

Vedlegg B

Blokkdiagram av momentkontroll

■ Torque Control Operation

Figure 12.64 shows the operation principle of Torque Control:

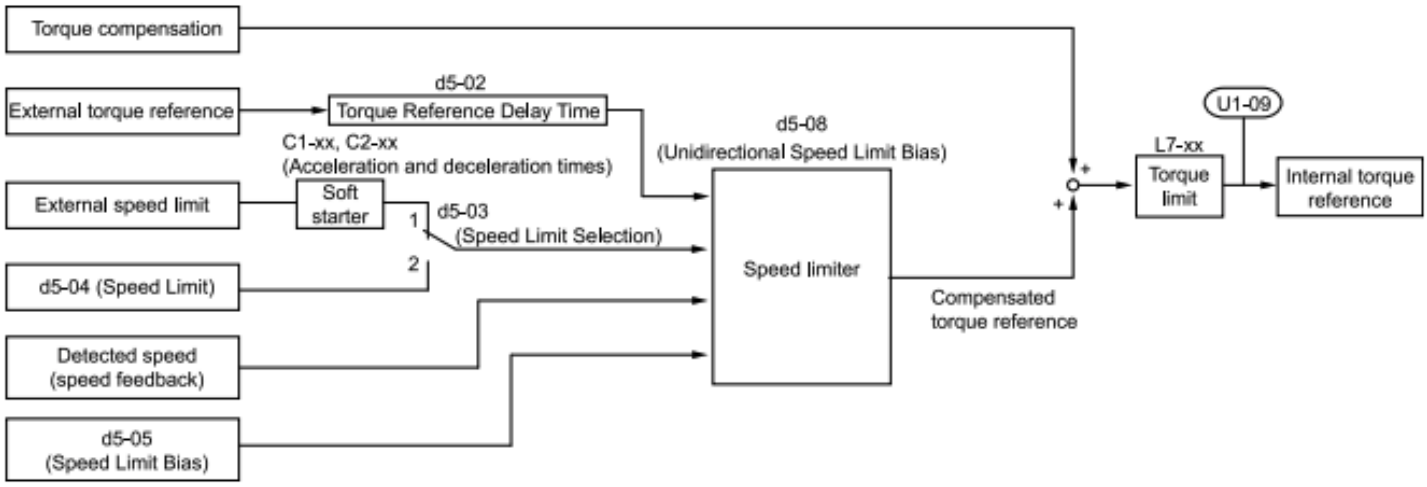


Figure 12.64 Torque Control Block Diagram

Figur B.1

Vedlegg C

Funksjonsblokker til frekvensomformeren

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 D

Blokkdiagram av hastighetskontroll

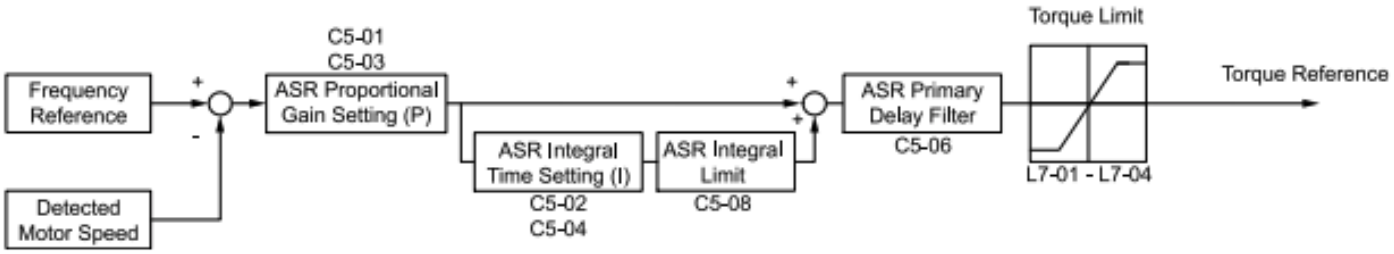


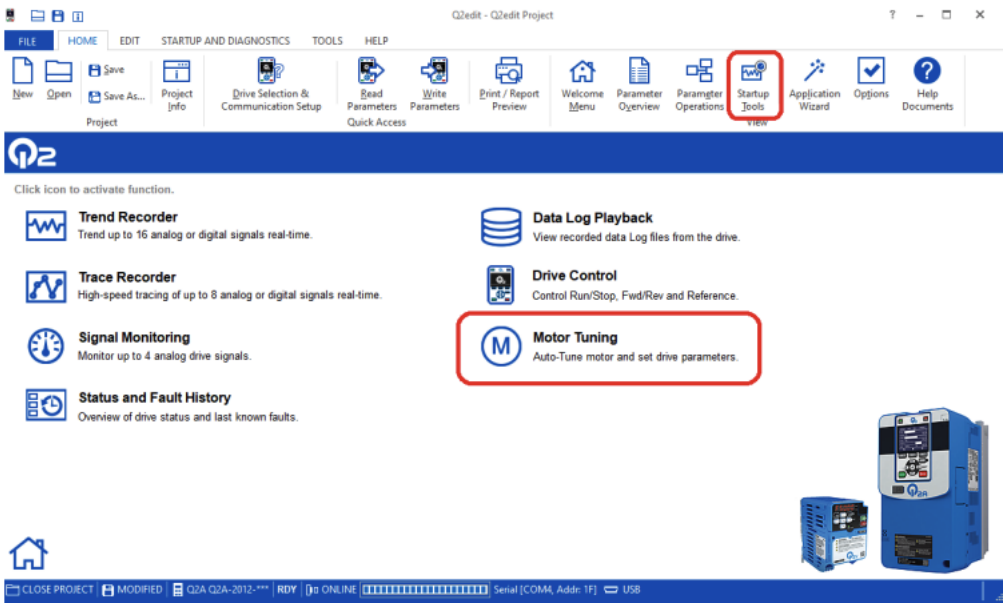
Figure 12.48 Speed Control Block Diagram for CLV, AOLV, CLV/PM, AOLV/PM, and EZOLV

Figur D.1

Vedlegg E

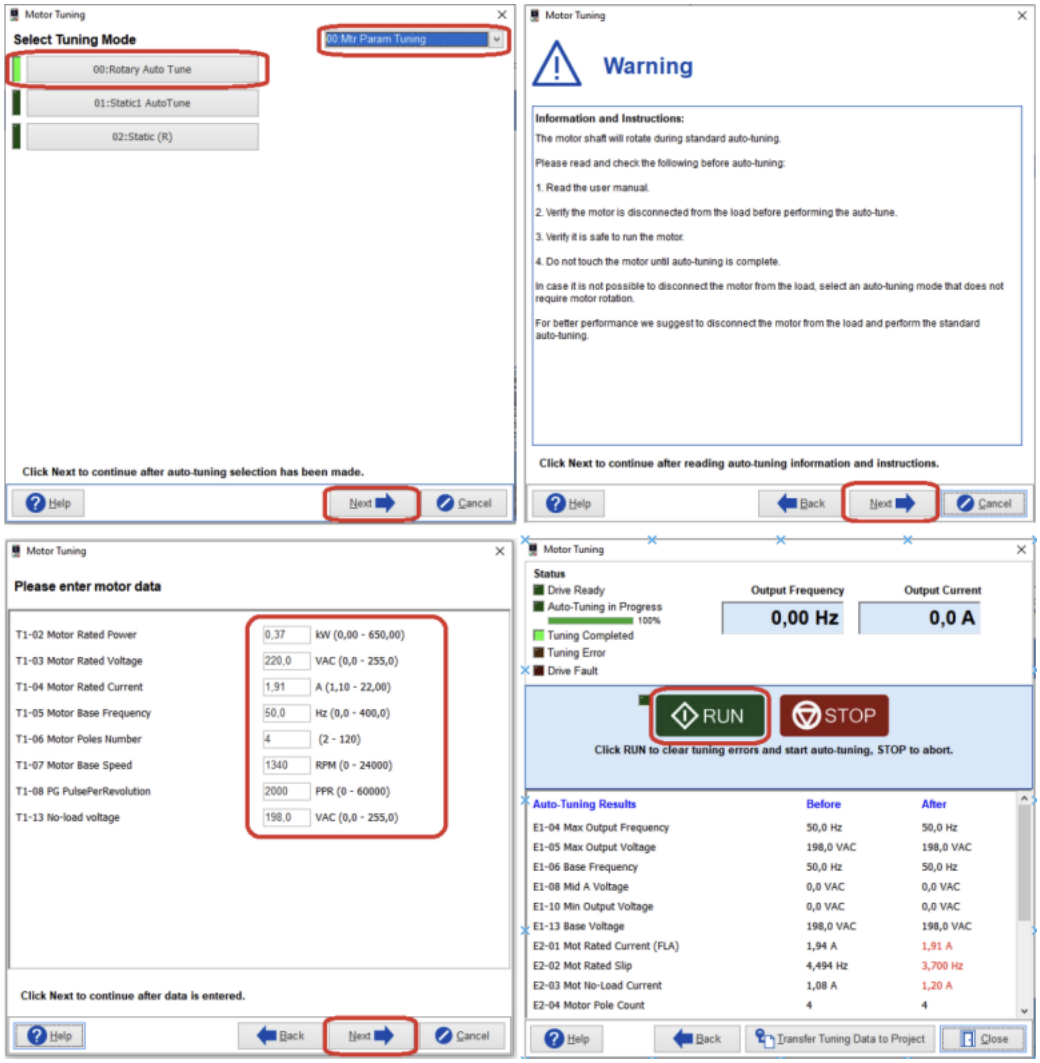
Autotuning

1. Startup Tools → Motor Tuning



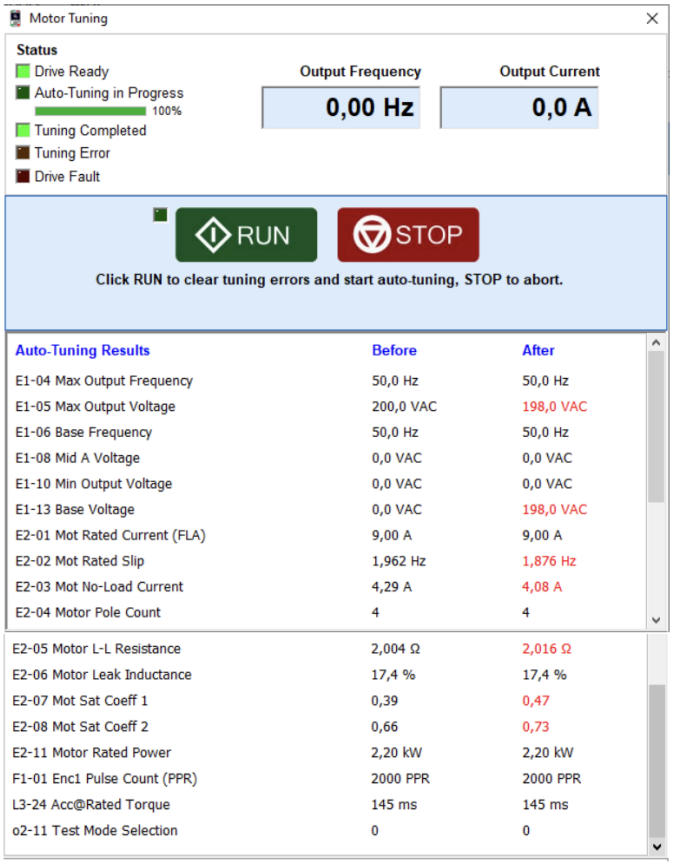
Figur E.1: [9]

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



Figur E.2: [9]

Figuren under viser resultatet vårt fra Autotuning av motoren:



Figur E.3

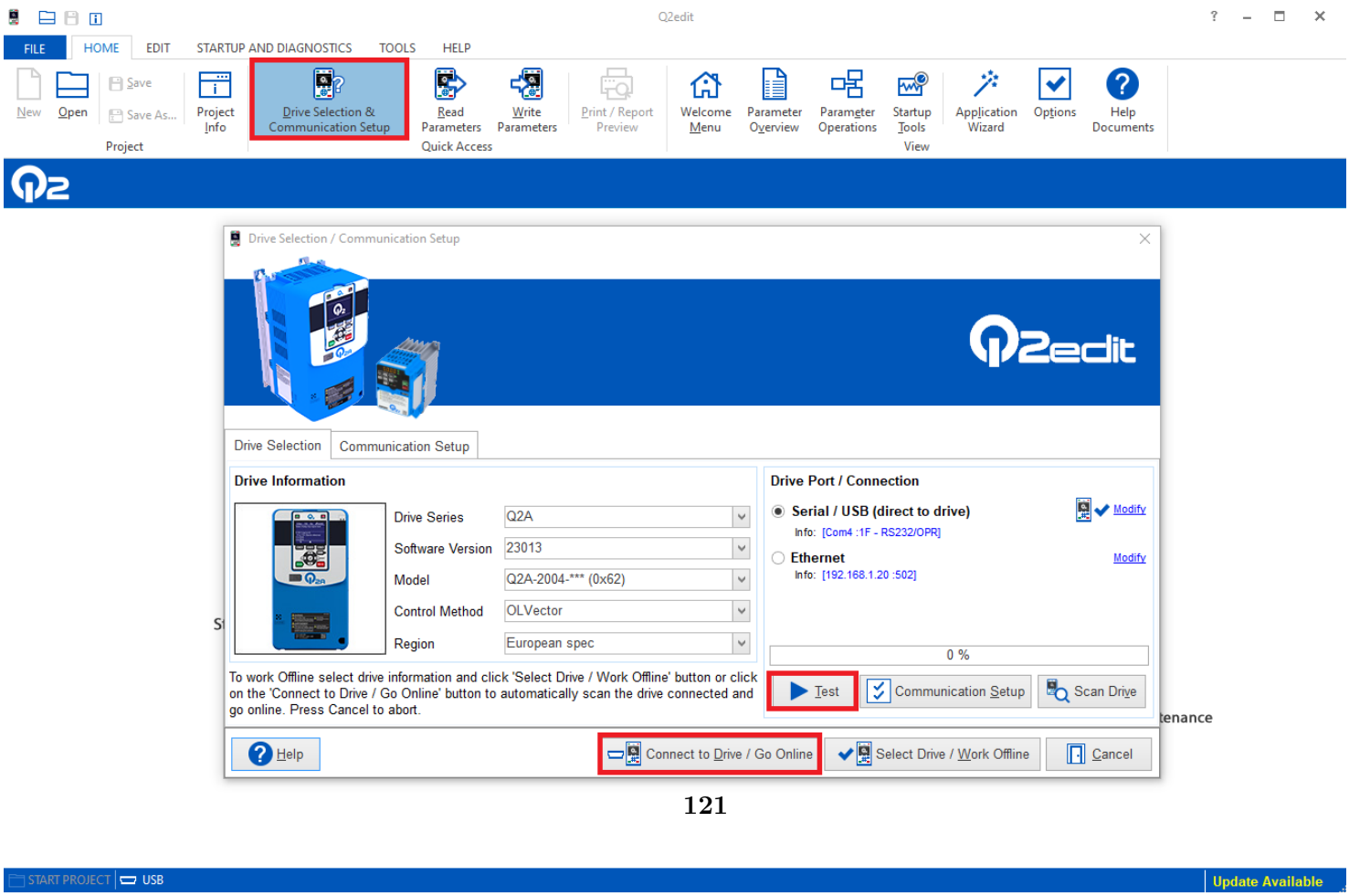
Vedlegg F

Youtube videoer

- Teach-funksjonen → https://youtu.be/5I6jVh3thk8
- Momentregulering → https://youtu.be/qgdWdFf25gM

Vedlegg G

Kommunikasjon mellom frekvensomformeren og Q2-edit



Figur G.1: Etablering av forbindelse mellom Q2A og Q2-edit: I figuren markeres det med rødt hvor man skal klikke for å initiere denne kommunikasjonen.