Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for elkraftteknikk

Per Kristian Engen
Wilhelm Bergesen
Sivert Heidsve

Modellering og simulering av drivsystem for permanentmagnet synkronmaskin

Bacheloroppgave i Elkraftteknikk
Veileder: Jonas Kristiansen Nøland
Mai 2021

NTNU
Kunnskap for en bedre verden

Per Kristian Engen
Wilhelm Bergesen
Sivert Heidsve

Modellering og simulering av drivsystem for permanentmagnet synkronmaskin

Bacheloroppgave i Elkraftteknikk
Veileder: Jonas Kristiansen Nøland
Mai 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for elkraftteknikk

NTNU
Kunnskap for en bedre verden

NTNU

Institutt for elkraftteknikk

Oppgavens tittel: Modellering og simulering av drivsystem for permanentmagnet synkronmaskin **Project title:** Modeling and simulation of drive system for permanent magnet synchronous machine	**Gitt dato**: 21.12.2020 **Innleveringsdato**: 20.05.2021 **Gradering**: Åpen **Antall sider/bilag**: 108/38
Gruppedeltakere: Per Kristian Engen Wilhelm Bergesen Sivert Heidsve	**Veileder internt:** Jonas Kristiansen Nøland Email: jonas.k.noland@ntnu.no Tlf: 91358759
Studiereting: Elektroingeniør - Elkraftteknikk	**Prosjektnummer:** E2127
Oppdragsgiver: Haf Power Solutions AS	**Kontaktperson hos oppdragsgiver:** Sigbjørn Svendsen Email: Sis@hpsolutions.no
Stikkord: Permanentmagnet synkronmaskin, MATLAB, Simulink, Python, Feltorientert Kontroll, Maskimalt dreiemoment per ampere, Simulering	
Keywords: Permanent magnet synchronous machine, MATLAB, Simulink, Python, Field oriented control, Max torque per ampere, Simulation	

Forord

Bacheloroppgaven er utarbeidet våren 2021 ved NTNU - Norges tekniske-naturvitenskapelige universitet - Trondheim, ved institutt for elkraftteknikk. Oppgaven er en avslutning på et 3-årig bachelorprogram for elektroingeniør, studieretning elkraftteknikk. Formålet med oppgaven er å modellere og simulere drivsystemet til en permanentmagnet synkronmaskin.

Oppgaven er gitt av Haf Power Solutions. Omfanget av oppgaven er utarbeidet av prosjektgruppen og veileder.

I løpet av prosjektperioden har vi tilegnet oss ny kunnskap om permanentmagnet synkronmaskiner (PMSM) og drivsystemet som kontrollerer maskinen. PMSM er en relativ ny type maskin som har vært lite berørt i løpet av studietiden. Det var vanskelig å finne gode informative kilder i starten av prosjektperioden. Gruppen har lært hva som inngår i drivsystemet for elektriske maskiner. Det har vært en utfordring å designe regulatorer som kontrollerer drivsystemet tilfredstillende. Arbeidet har resultert i et sluttprodukt som prosjektgruppen er fornøyd med, og har gitt oss verdifull erfaring innenfor elektriske maskiner og drivsystem.

Vi ønsker å takke veileder, Jonas Kristiansen Nøland for god veileding og gode innspill. Vi ønsker også å takke våre oppdragsgivere fra Haf Power Solutions, Sigbjørn Svendsen og Bengt Olav Berntsen for gode råd og en spennende oppgave.

Per Kristian Engen	Sted/dato
Wilhelm Bergesen	Sted/dato
Sivert Heidsve	Sted/dato

Sammendrag

For drifting av elektriske fartøy har permanentmagnet synkronmaskinen blitt et populært valg av motor. Formålet med oppgaven er å utvikle en digital modell av permanentmagnet synkronmaskinen og drivsystemet som kontrollerer maskinen. Modellen blir brukt til å simulere turtall, dreiemoment og strømmer i maskinen.

Den digitale modellen av drivsystemet er utviklet i programmeringspråket Python. Det er også utviklet en modell i Matlab/Simulink for verifisering og undersøkelse. Permanentmagnet synkronmaskinen er modellert basert på den matematiske modell presentert i rapporten. Drivsystemet bruker feltorientert kontroll med dq-roterende referanseramme. For kontroll bruker modellene en hastighetregulator for å styre turtall, og to strømregulatorer for å styre statorstrømmer i maskinen. Modellen benytter maksimum dreiemoment per ampere kalkulasjon for å beregne referansestrømmene.

De digitale modellene er utviklet med en rekke forenklinger. Det er potensial for videreutvikling, og rapporten tilrettelegger for framtidig arbeid.

Abstract

For the operation of electric vehicles, the permanent magnet synchronous machine has become a popular choice of motor. The purpose of this thesis is to develop a digital model of a permanent magnet synchronous machine and the drive system that controls the machine. The model is then used to simulate the machine speed, torque and currents.

The digital model of the drive system is developed in the programming language Python. A model has also been developed in Matlab/Simulink for verification and investigation. The permanent magnet synchronous machine is modeled based on the mathematical model presented in this thesis. The drivesystem uses field-oriented control with dq-rotating reference frame. For control, the models use a speed controller to control speed, and two current controllers for control of stator currents in the machine. The model utilises maximum torque per ampere calculation to find the reference currents.

The digital models have been developed with a number of simplifications. There is potential for further development, and the report facilitates future work.

Forkortelser

AC	Alternating Current
Back-EMF	Back-Electromotive Force
DC	Direct Current
FEMM	Finite Element Method Magnetic
FOC	Field Oriented Control
HPS	Haf Power Solutions
IM	Internal Mounted
IMC	Internal Model Control
IPM	Interior Permanent Magnet
MMF	Magneto-motive Force
MTPA	Max Torque Per Ampere
PI	Proporsjonal-integral
PM	Permanentmagnet
PMSM	Permanentmagnet synkronmaskin
RM	Reluktansmotor
SM	Surface Mounted
SMPM	Surface Mounted Permanent Magnet
SVPWM	Space Vector Pulse Width Modulation

Nomenklatur

B	Viskositet koeffisient
I_{nom}	Nominell statorstrøm
J	Motorens treghetsmoment
K_i	Integralkonstant
K_p	Proposjonalkonstant
L_d	d-akse induktans
L_q	q-akse induktans
R_s	Stator resistans
T_e	Elektrisk dreiemoment
T_{Last}	Lastmoment
$T_{e,ref}$	Referanse dreiemoment
T_{nom}	Nominelt elektrisk dreiemonent
Θ_e	Elektriske vinkelposisjon
Θ_m	Mekanisk vinkelposisjon
β	Dreiemomentvinkel
λ_{pm}	Permanentmagnet flukskobling
ω_b	Mekanisk basehastighet
ω_e	Elektrisk rotasjonshastighet
ω_m	Mekanisk rotasjonshastighet
i_d	d-akse strøm
i_q	q-akse strøm
$i_{\alpha\beta}$	Stømmer i $\alpha\beta$-referanserammen
i_{abc}	Trefase statorstrømmer i tidsdomene
k	Utpregnings-forhold
p	Polpar
v_d	d-akse spenning
v_q	q-akse spenning

Innhold

Figurliste

Tabelliste

1 Innledning

1.1 Bakgrunn

Transport står for omtrent en tredjedel av totalt klimagassutslipp i Norge. Sjøtransporten utgjør 19% av utslippene [17]. Miljøvennlig skipsfart er ett av fem prioriterte innsatsområder innenfor dagens klimapolitikk [7]. Et av tiltakene for å redusere utslipp er mindre bruk av fossile brennstoff. Med disse målene, og et ønske om et grønt skifte innenfor sjøtransport, utvikler Haf Power Solutions AS en fullelektrisk turistbåt.

Båten benytter to elektriske motorer for fremdrift. Permanentmagnet synkronmaskin er et forslag til motortype. PMSM har i nyere tid vist seg å ha flere fordeler sammenlignet med den konvensjonelle asynkronmotoren. Maskinen har høy effektivitet og høy effekttetthet [18].

For å kontrollere hastighet og dreiemoment til motorene bennytes et elektrisk drivsystem. Et viktig instrument for utvikling av drivsystemet er en digital modell. Med en digital modell kan drivsystemet simuleres for å danne et oversiktsbilde av motoroppførsel. En digital modell kan også forenkle designprosessen og minske kostnader.

1.2 Problemstilling

Lag en modell av drivsystemet til en permanentmagnet synkronmaskin i åpen kildekode. Modellen skal fungere som et utgangspunkt for en digital tviling til et elektrisk fartøy.

1.3 Omfang og begrensninger

Prosjektgruppen skal lage en digital modell av drivsystemet for en permanentmagnet synkronmaskin i åpen kildekode. Modellen skal være et utgangspunkt for å simulere dreiemomentkurver, turtallkarakteristikk og statorstrømmer i motoren. Drivsystemet skal inneholde kontroll for hastighet og statorstrømmer, og modellen av permanentmagnet synkronmaskinen. Gruppen skal finne en metode for å beregne referansestrømmer. Ved å simulere en digital modell kan tiden det tar å utvikle et reelt drivsystem kortes ned. Kostnadene blir også mindre.

Modellen skal utvikles i åpen kildekode som støtter simulering i et sanntidsoperativsystem. Åpen kildekode har blitt valgt fordi arbeidsgiver ønsker en modell som medfører så få kostnader som mulig. Løsningen for åpen kildekode som skal brukes er programmeringsspråket Python. Python støtter sanntidsoperetivsystemer.

Et mål med oppgaven er å lage en mest mulig universell modell av et PMSM drivsystem. Det skal være enkelt å tilpasse modellen etter hvilken motor som skal simuleres.

For verifisering av modellen i Python skal det lages en modell i Matlab/Simulink. Simulink-modellen kan inneholde et mer kompleks drivsystem, for å undersøke løsninger som kan bli implementert i Python-modellen og for å danne et grunnlag for verifisering av Python-modellen.

Følgende avgrensninger er gjort i modellen: Drivsystemet for PMSM vil være en forenklet modell som ikke inneholder essensielle komponenter for et reelt system. Modellen skal betraktes som et utgangspunkt for videreutvikling og optimalisering. Sanntidssimulering gjennomføres ikke som en del av resultatet i denne rapporten. Målet er at modellen kan være et utgangspunkt for det i framtidig arbeid. Modellen er kun gjeldene for permanentmagnet synkronmaskiner. Det er ønskelig at modellen skal kunne fungere for PMSM både med og uten utpregete poler.

For å gjøre mulighetene for videreutvikling bedre, skal modellen bli godt dokumentert.

Modellen skal brukes som utgangspunkt for utvikling av en digital tvilling til en elektrisk turistbåt.

1.4 Rapportens oppbygning

Rapporten er strukturert etter standard oppbygging for akademiske oppgaver ved NTNU. Målet er at rapporten skal være oversiktlig for leseren, og at formålet og fremgangsmåten til gruppen kommer godt frem.

Kapittel 1 *Innledning* presenterer bakgrunnen for bacheloreoppgaven og legger frem problemstilling og omfanget til oppgave. Kapittelet gir en oversikt over innholdet i rapporten.
Kapittel 2 *Teori* inneholder generell og relevant teori som oppgaven baserer seg på. Kapittelet presenterer den matematiske modellen for PMSM og teoretisk bakgrunn som brukes for å designe drivsystemet.
Kapittel 3 *Metode* beskriver fremgangsmåten prosjektgruppen har valgt for å designe og simulere modellen. Kapittelet presenterer også prosjektorganisering og ressurser.
Kapittel 4 *Modell og simulering* skal gi leseren innsikt i hvordan Simulink-modellen og Pythonmodellen er designet. Leseren skal få innblikk i hvordan relevante formler er implementert. Kapittelet skal også redgjøre for hvilke forenklinger som er gjort.
Kapittel 5 *Resultater* presenterer og analyserer resultatene fra alle simuleringene, i form av grafer med tilhørende forklaringer.
Kapittel 6 *Døfting* diskuterer drivsystemet som har blitt utviklet og viser forskjeller mellom modellene. Svakheter, forbedringer og framtidig arbeid blir også diskutert.
Kapittel 7 *Konklusjon* avslutter rapporten med en konklusjon av problemstillingen.

2 Teoretisk grunnlag

2.1 Clarke og Park transformasjon

Oppførselen til trefase AC-maskiner blir vanligvis beskrevet med strøm- og spenningsligninger. Koeffisientene til de relevante differensialligningene varierer med tiden så lenge motoren roterer. Den matematiske modellen er kompleks ettersom strøm, spenning og flukskobling vil endre seg kontinuerlig med den relative bevegelsen til kretsen [13].

Trefasesystemer kan representeres i flere referanserammer. Vanligvis er størrelsene representert i tidsdomenet, uttrykt med romvektorer og trigonometrisk funksjoner, abc-referanserammen. For kontroll av AC-maskiner kan det være nyttig å transformere verdiene til en rettvinklet roterende dq-referanseramme [2]. Endringen av referanseramme forenkler arbeidet med å løse de tidsvarierende differensialligningene. For å endre referanseramme brukes Clarke og Park transformasjon.

2.1.1 Clarke transformasjon

Clarke transformasjon omgjør balanserte trefasestørrelser til balanserte tofase størrelser i en stasjonær αβ-referanseramme. Referanseakser er rettvinklet i forhold til hverandre.

Følgende ligninger beskriver hvordan Clarke transformasjon utføres for trefase strømmer:

Trefase strømmer representert i abc-referanseramme:

$$\begin{aligned} i_a &= i_{peak} * cos(\phi) \\ i_b &= i_{peak} * cos(\phi + \tfrac{2\pi}{3}) \\ i_c &= i_{peak} * cos(\phi - \tfrac{2\pi}{3}) \end{aligned} \tag{2.1}$$

For balanserte system gjelder:

$$\begin{aligned} i_a + i_b + i_c &= 0 \\ i_c &= -(i_a + i_b) \end{aligned} \tag{2.2}$$

Ligning som beskriver utregning av Clarke transformasjon:

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} = \frac{2}{3} \times \begin{bmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.3}$$

Ved å bruke ligning (2.3) og (2.4) kan strømmer i_α og i_β uttrykkes som:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} \tag{2.4}$$

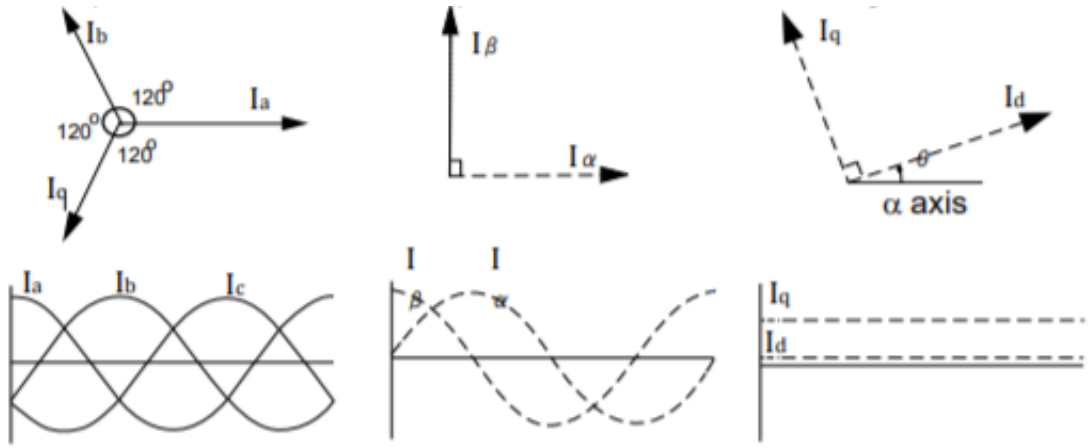
2.1.2 Park transformasjon

Park transformasjon omgjør størrelser som er representert i den stasjonære αβ-referanseramme til størrelser representert i en roterende dq-referanseramme. Referanse aksene d og q er rettvinklet i forhold til hverandre. Vinkelen Θ er rotasjonsvinkelen mellom den roterende d-aksen og den stasjonære α-aksen.

Ligning som beskriver utregning av Park transformasjon:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} cos(\Theta) & sin(\Theta) \\ -sin(\Theta) & cos(\Theta) \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{2.5}$$

Figur 2.1 viser hvordan trefasestrøm I_{abc} representeres i de ulike referanserammene.



Figur 2.1: Representasjon av strømmer I_{abc}, $I_{\alpha\beta}$ og I_{dq} i forskjellige referanserammer (bilde [13])

2.2 Permanentmagnet synkronmotor

Permanentmagnet synkronmaskin er en vekselstrøm maskin. I motormodus omformer den elektrisk energi til mekanisk energi. Fysikken bak elektriske maskiner er beskrevet av Faradays induksjonslov og Lenz' lov.

Elektriske AC-maskiner består av en roterende rotor og en stasjonær stator. Permanentmagnet synkronmotor er en del av en større motorfamilie, børsteløse AC-motorer. Andre motorer som tilhører familien er induksjonsmotor, børsteløs DC-motor og reluktansmotor. Hovedforskjellen mellom motorer i denne familien er rotorkonstruksjon. Statorkonstruksjon er tilmærmet lik.

Stator i PMSM består av en ytre ramme og en kjerne med trefase viklinger. Rotor er typisk plassert inne i stator. PMSM har montert permanentmagneter enten på overflaten, eller på innsiden av rotor. Maskinen blir delt inn i to grupper [14]:

- Overflate montert permanentmagnet synkronmaskin
- Innvendig montert permanentmagnet synkronmaskin

Synkronmotorer blir også delt inn etter hvordan selve rotor er designet:

- Rotor med utpregete poler
- Rotor uten utpregete poler

Ved å føre trefasestrøm gjennom viklingene i stator, skapes det ett roterende magnetfelt. Polene i permanentmagnetene på rotor vil forsøke å tilpasse seg det skapte magnetfeltet. Det oppstår dermed et elektromagnetisk dreiemoment som driver rotasjonen i rotor. Et eksempel er illustrert i figur 2.2a. Rotasjonen vil ha samme hastighet som magnetfeltet i stator. Hastigheten refereres til som synkron hastighet, derav navnet permanentmagnet synkronmotor

På grunn av treghetsmomentet i rotor, er ikke PMSM selvstartende. Hvis en stillestående PMSM påføres vekselstrøm, vil ikke rotor akselerere raskt nok til å holde følge. Da vil tiltrekningen fra neste induserte magnetiske pol i det roterende magnetfeltet i rotor blir større en den passerende magnetiske polen. Rotor trekkes først i samme retning som det roterende magnetfeltet, før den trekkes i motsatt retning, av neste pol. Netto turtall blir da null. Oppstartsproblemet løses på forskjellige måter. Men en større motor bruker ofte dempeviklinger. Dempeviklinger er staver som tres inn i spor på polene til rotor. Dempeviklingene vil fungere som burviklinger i en selvstartende asynkronmotor. Dempeviklingene hjelper derfor rotor med å komme opp i synkron hastighet.

Det er også et problem at poler glipper når PMSM kjøres i høye hastigheter. Lastmomentet blir da for stort, som resulterer i at polene glipper. Resultatet blir virvelstrømmer i permanentmagnetene, som fører til varmeutvikling, som kan føre til demagnetisering.

Det er flere fordeler ved å bruke en PMSM. Bruken av permanentmagneter i rotor, istedenfor viklinger, fører høyere effektivitet og tillitsgrad. Ved å fjerne rotorviklinger blir det ikke noe rotortap. Mangel på børster og sleperinger fører til mindre vedlikehold. En synkronmotor er enklere å regulere, ettersom sammenhengen mellom mekanisk turtall og elektrisk frekvens er svært pålitelig. Maskinen har et lavt treghetsmoment som sørger for ytterligere pålitelighet til styresystem og god dynamisk oppførsel [10].

En motor med utpregete poler på rotor, vil ha et reluktansbidrag i dreiemomentet. Reluktans er magnetisk motstand i magnetisk ledende metaller. Reluktans kan sammenlignes med resistans i elektriske ledere, bortsett fra at energien blir lagret som et magnetisk felt, istedenfor å bli forbrukt som varme. Noen metaller er ferromagnetiske, som betyr at de har mange små domener med tilfeldig orientert magnetfelt. Summen av magnetfeltene til alle domenene blir null. Ved å utsette et ferromagnetisk legeme for et ytre magnetfelt, vil domenene endre orientering, og legemet får karakteristikk som ligner permanentmagneter.

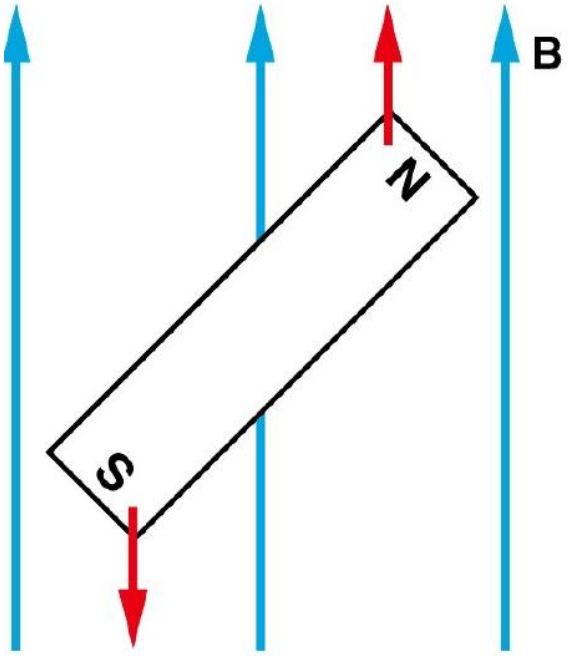
Magnetfelt vil gå minste reluktans vei. Luft har høyere reluktans enn metall, som gjør at magnetfelt kan påvirkes av- og påvirke magnetiske ledere. Som vist i figur 2.2b vil en stav mellom to magneter endre formen på magnetfelte mellom dem. Siden staven står skrått imellom magnetene, blir den utsatt for et dreiemoment, ettersom magnetfeltet tilstrever minste mulige reluktansvei. Det resulterende dreiemomentet er prinsippet bak en reluktansmotor (RM). Et eksempel av magnetfeltsendringen er illustrert i figur 2.2b. RM-armatur er illustrert i figur 2.3d.

En RM sin avhengighet av de roterende polene i stator, er den samme som i en PMSM. Det gjør dem kompatible med hverandre, som gjør kombinering av konseptene mulig. Rotor vil da enten ha IPM montert i spalter, eller SMPM. Eksempler er illustrert i figurene 2.3b og 2.3c.

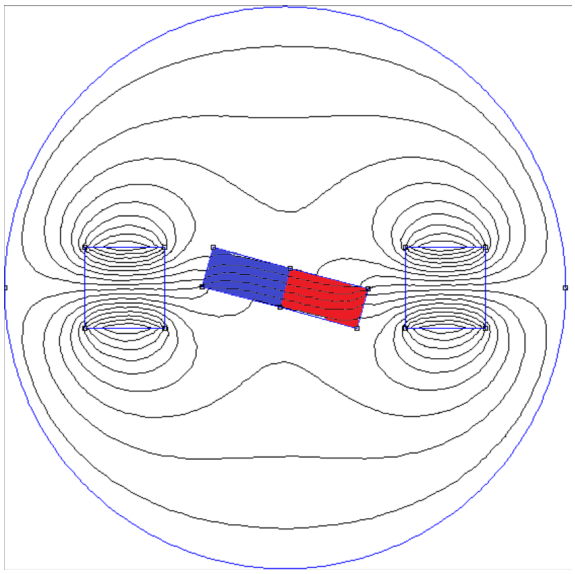
Reluktansbidraget i en PMSM kan beskrives ved hjelp av utpregningsforholdet. Utpregningsforholdet er størrelsesforholdet til induktansene L_d og L_q. Det kan også forklares som avhengigheten av rotorvinkelen til rotorreluktansen. Formelen for reluktansdreiemomentet beskrives som:

$$T_{\text{rel}} = 1.5p(L_{\text{d}} - L_{\text{q}})i_{\text{d}}i_{\text{q}} \tag{2.6}$$

Formelen viser at hvis $L_d < L_q$ blir T_{rel} negativ. Det er derfor ønskelig at id alltid er negativ, slik at T_{rel} blir positivt. Formelen viser at et større forhold mellom L_d og L_q resulterer i et større dreiemoment. Delkapittel 2.3.4 forklarer dreiemomentet i PMSM mer utdypende.

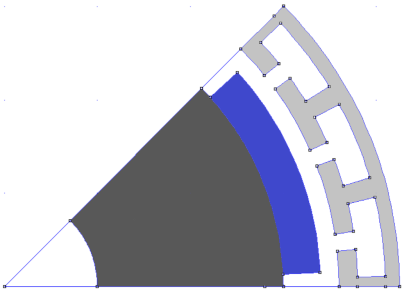


(a) Permanentmagnet i magnetfelt. Polene på magneten tilpasser seg magnetfeltet og trekkkreftene skaper dreiemoment mot klokkeretningen.

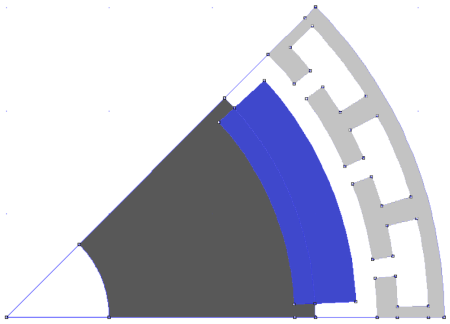


(b) Stav i magnetfelt. Magnetfeltet går fra venstre til høyre i midten. Staven tilpasser seg magnetfeltet, der rød del får karakteristikk til en magnetisk nordpol, og blå til en magnetisk sørpol. Det oppstår da et elektromagnetisk dreiemoment mot klokkeretningen som i eksempel (a).

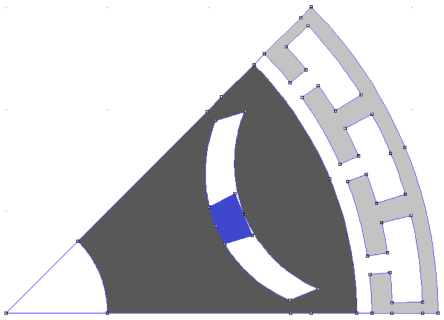
Figur 2.2: Elektromagnetisk dreiemoment i magnetfelt.



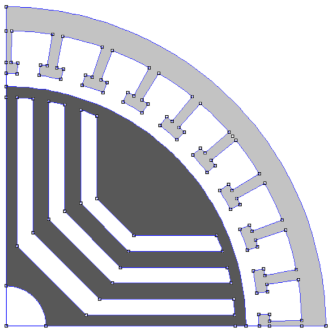
(a) SMPM, uten utpregning. $L_d \approx L_q$



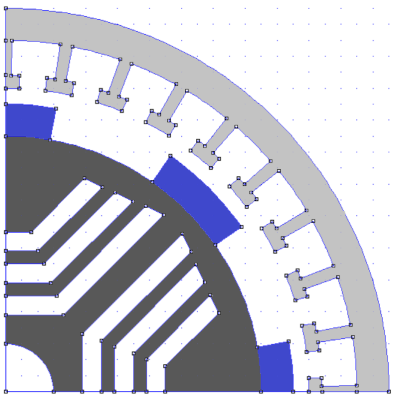
(b) PMSM med mild utpregning. $L_q < L_d$



(c) IPM montert i spalte, høy utpregning. $L_d < L_q$



(d) Reluktans motor armatur. T_e rent basert på utpregning. Spaltene former det innvendige i rotor slik at magnetfeltet lettere skaper poler. $L_d < L_q$



(e) PMSM med permanent magneter montert ved polene til en reluktans maskin. $L_d > L_q$. Resulterer i et motsatt rettet reluktans dreiemoment i forhold til permanentmagnet dreiemomentet. Dårlig rotor design.

Figur 2.3: Illustrasjon av ulike rotorkonstruksjoner. Fargekoder: Blå = permanentmagnet, lysegrå = statorramme, mørkegrå = stål i rotor, hvit = luft. Statorviklinger er ikke tegnet inn.

Tabell 1 oppsummerer fordelene og ulempene ved å bruke en PMSM:

Tabell 1: Fordeler og ulemper ved permanentmagnet synkronmotor

Fordeler	Ulemper
Høyt dreiemoment over et stort område	Permanentmagnetene kan demagnetisere
Høy dreiemomenttetthet	Ikke selvstartende
Lave vedlikeholdskostnader	Høye initial kostnader
Lavt treghetsmoment	Bruker sjeldne jordmetaller i PM
Enkel kontroll	
Høy energieffektivitet	
Stillegående	

2.2.1 Matematisk modell av PMSM

For å gjøre PMSM lettere å analysere brukes det Clarke- og Park-transformasjoner til å gjøre et trefase stasjonært referansesystem(abc), om til et tofase roterende referansesystem(dq). Fordelen er at problemene med å ha tidsvarierende parametere blir eliminert, og det blir lettere å lage en ekvivalentkrets for maskinen. Metoden bak Clarke- og Parktransformasjoner er forklart tidligere i kapittelet. Den matematiske modellen av PMSM er basert på størrelser representert i dq-referanserammen.

Det har blitt gjort mye forskning på permanentmagnet synkronmaskiner, og den matematiske modellen til en PMSM er godt kjent innen fagområdet for elektriske maskiner [15]. Derfor er det ikke tatt med en detaljert utledning av den matematiske modellen til en PMSM i rapporten. Den matematiske modellen av en PMSM i dq-referanserammen er gitt av formelene 2.9-13. Ekvivalentkretsen for en PMSM i dq-referanserammen er vist i figur 2.4.

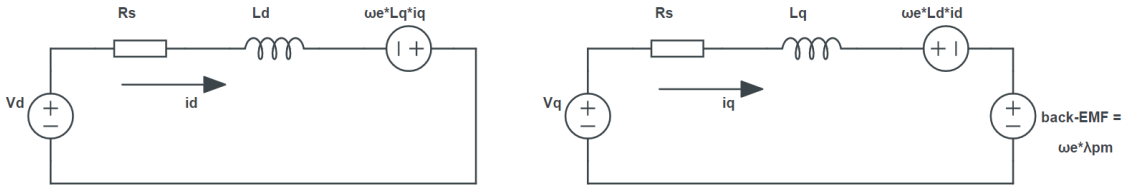
$$\frac{di_d}{dt} = \frac{1}{L_d}v_d - \frac{R_s}{L_d}i_d + \frac{L_q}{L_d}p\omega_m i_q \tag{2.7}$$

$$\frac{di_q}{dt} = \frac{1}{L_q}v_q - \frac{R_s}{L_q}i_q - \frac{L_d}{L_q}p\omega_m i_d - \frac{1}{L_q}p\omega_m \lambda_{pm} \tag{2.8}$$

$$T_e = \frac{3}{2}p(\lambda_{pm}i_q + (L_d - L_q)i_d i_q) \tag{2.9}$$

$$\frac{d\omega_m}{dt} = \frac{1}{J}(T_e - T_{Last} - B\omega_w) \tag{2.10}$$

$$\frac{d\Theta_m}{dt} = \omega_m \tag{2.11}$$



Figur 2.4: Ekvivalentkrets for PMSM i dq-referanseramme.

Den matematiske modellen av en PMSM er basert på følgende antagelser: [6]

- Stator viklinger produserer sinusformet magnetomotorisk spennings fordeling. Harmoniske i luftgapet er neglisjert.
- Spenningskilde produserer balansert trefase spenning.
- Virvelstrøm og hysterese-effekt er neglisjert.
- Jerntap er neglisjert
- Resistanser er uavhengig av temperatur og frekvens.
- Motindusert spenning er sinusformet.

For å finne verdien av den elektriske rotasjonshastighetet til en PMSM så må det mekaniske rotasjonshastighetet multipliseres med antall poler (2.12). Elektrisk vinkelposisjon blir funnet ved å bruke formel 2.13

$$\omega_e = p\omega_m \tag{2.12}$$

$$\frac{d\Theta_e}{dt} = \omega_e \tag{2.13}$$

2.3 Kontroll av permanentmagnet synkronmaskin

Målet med kontrollsystemet til elektriske maskiner er å oppnå ønsket posisjon, hastighet, eller dreiemoment. For maskiner som blir brukt i elektriske fartøy er kontroll av hastighet og dreiemoment mest aktuelt.

I nyere tid er det to hovedstrategier for kontroll av PMSM. Hovedstrategiene for kontroll av PMSM er dirkete momentstyring og feltorientert kontroll (FOC). Utførelsen av kontroll er forskjellig for strategiene, men målet er det samme. Begge prøver å oppnå effektiv kontroll av dreiemomentet og fluks slik at motoren følger referansen uavhengig av variasjon i last og maskin [10]. Begge metodene er velprøvd med suksess, og det er ikke et klart svar på hvilken metode som er overlegen [10].

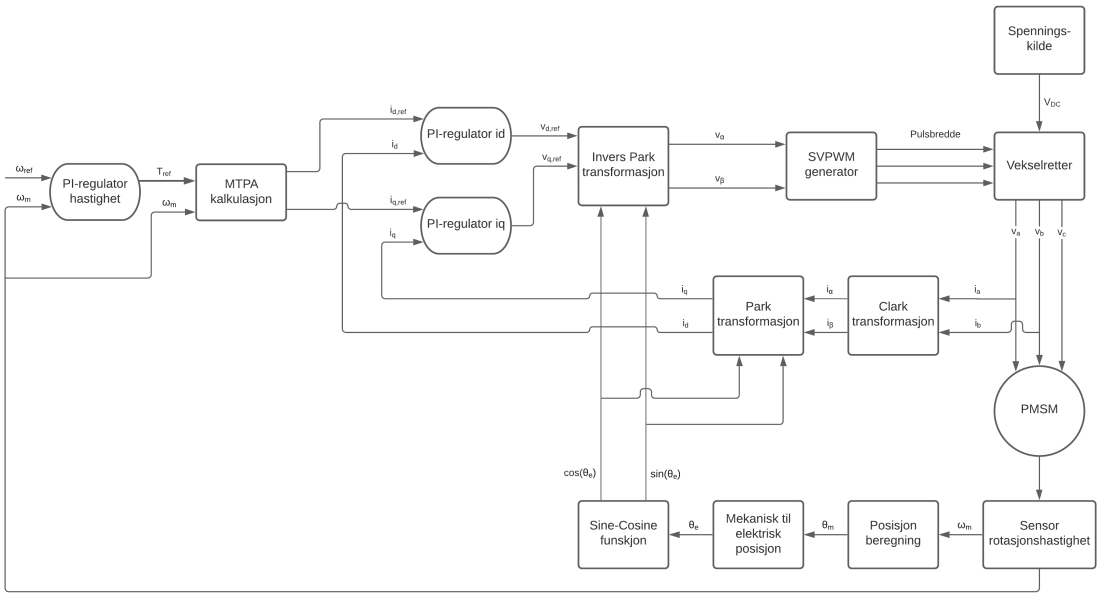
Rapporten forklarer feltorientert kontroll, ettersom det er kontrollstrategien som blir brukt i modellen.

2.3.1 Feltorientert kontroll

Innenfor feltorientert kontroll er det mest vanlig å enten kontrollere hastighet eller dreiemoment. Det er også mulig å kontrollere posisjon, men det er mindre brukt. For dreiemomentkontroll følger drivsystemet en gitt referanse for dreiemoment. For hastighetskontroll følger drivsystemet en hastighetsreferanse som lager en referanse for dreiemomentet, $T_{e,ref}$ [4].

Målet med FOC er å kontrollere dreiemomentet og det magnetiske feltet i maskinen. Det blir gjort ved å kontrollere id- og iq-komponentene til statorstrømmen [11]. Den matematiske modellen som beskriver PMSM transformeres til et referansesystem som roterer synkront med rotorfluks-vektoren, se kapittel 2.1. Endring av referanseramme gjør styresystemet mindre komplisert og det blir enklere å kontrollere komplekse trefase AC-motorer. For å gjennomføre FOC trenger systemet tilbakekobling av rotasjonshastighet ω_m, rotorposisjon Θ_e og statorstrømmer i_{abc}. Kontrollsystemet inneholder to strømregulatorer for i_d og i_q og en regulator for hastighet ω_m. Figur 2.5 viser overordnet struktur for feltorienter kontroll.

Fordelene med å bruke FOC for kontroll er hurtig respons og små ringvirkninger i dreiemomentet [11].



Figur 2.5: Feltorientert kontroll av permanentmagnet synkronmaskin.

2.3.2 Strømregulator

Funksjonen til strømregulatoren er å sette referansespenninger til motor basert på referansestrømmer og tilbakekoblingen av statorstrømmer. Inngangsparametrene til strømregulatoren er referansestrømmer $i_{d,ref}$, $i_{q,ref}$, og statorstrømmer i_d, i_q. Utgangsparametrene er $v_{d,ref}$ og $v_{q,ref}$. Strømregulatoren anses å være et eget lukket sløyfesystem i drivsystemet.

Strømregulatoren som presenteres er en proporsjonal-integral regulator som bruker intern modell kontroll, krysskoblingskompensjasjon og aktiv demping for styring.

Metoden for å modellere strømkontroll og hastighetskontroll for permanentmagnet synkronmaskinen er intern modell kontroll (IMC) [18]. Resultatet er PI-regulatorer med kontrollparametere, K_p og K_i, uttrykt direkte med båndbredden til sløyfesystemet og motorparameterne, L_d og L_q. Båndbredde er definert som frekvensområdet regulatoren arbeider innenfor. Ved å bruke IMC metode for å modellere kontrollsystemet blir "prøve og feile" metoden for å finne regulatorparameterne unødvendig. I tillegg blir det enklere å kompansere for krysskoblingene i spenningsligningene 2.14 og 2.15 [18].

For å kontrollere permanentmagnet synkronmaskinen må man kontrollere statorspenningene v_d og v_q:

$$v_d = R_s i_d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \tag{2.14}$$

$$v_q = R_s i_q + L_q \frac{di_q}{dt} + \omega_e L_d i_d + \omega_e \lambda_{pm} \tag{2.15}$$

Utfører Laplace transformasjon på spenningsligningene og uttrykker dq-aksestrømmene som:

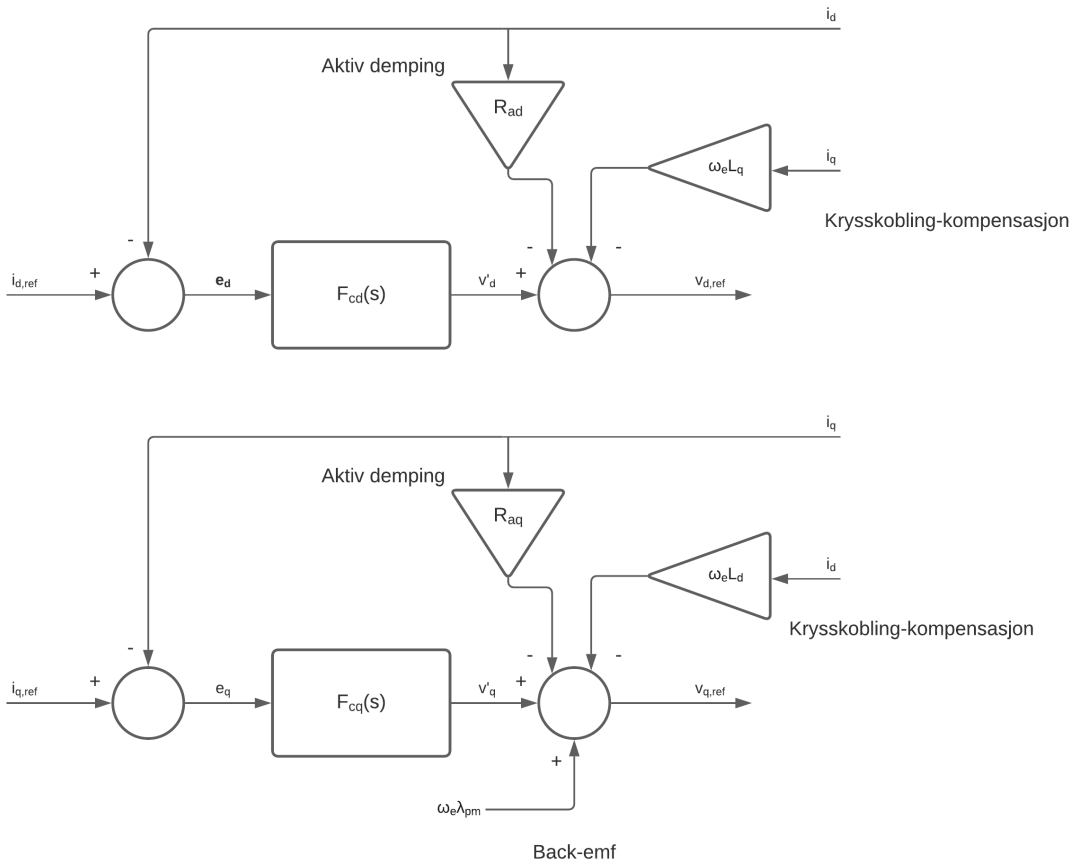
$$i_d = \frac{1}{L_d s + R_s}(v_d + \omega_e L_q i_q) \tag{2.16}$$

$$i_q = \frac{1}{L_q s + R_s}(v_q + \omega_e L_d i_d - \omega_e \lambda_{pm}) \tag{2.17}$$

Ligningene for i_d og i_q inneholder krysskoblings-komponenter og et back-emf ledd:

$$\begin{aligned} i_d &: \omega_e L_q i_q \\ i_q &: \omega_e L_d i_d \\ Back-EMF &= \omega_e \lambda_{pm} \end{aligned} \tag{2.18}$$

I strømregulatoren utføres det krysskoblings-kompenasjon for å ta hensyn til komponentene, og back-EMF leddet blir matet fram i systemet. Spenningene v'_d og v'_q defineres som regulert spenning før teknikken utføres. Figur 2.6 viser hvordan regulatorene utfører krysskoblings-kompensasjon, framflytting av back-EMF og aktiv demping.



Figur 2.6: Oversiktsbilde av strømregulator.

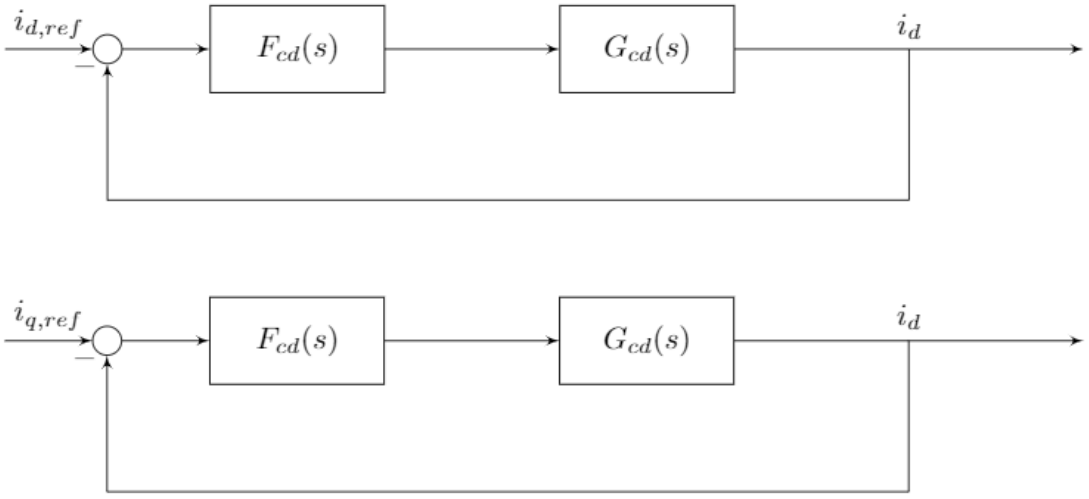
Fra ligning 2.16 og 2.17 blir overføringsfunksjonene fra v'$_d$ til i$_d$ og v'$_q$ til i$_q$ definert som:

$$\begin{aligned} G_{cd}(s) &= \frac{i_d}{v'_d} = \frac{1}{L_d s + R_s} \\ G_{cq}(s) &= \frac{i_q}{v'q} = \frac{1}{L_q s + R_s} \end{aligned} \tag{2.19}$$

Etter implementering av aktiv demping blir overføringsfunksjonene:

$$\begin{aligned} G_{cd}(s) &= \frac{i_d}{v'_d} = \frac{1}{L_d s + R_s + R_{ad}} \\ G_{cq}(s) &= \frac{i_q}{v'_q} = \frac{1}{L_q s + R_s + R_{aq}} \end{aligned} \tag{2.20}$$

For å bestemme kontrollparametre K$_p$ og K$_i$ til PI-regulatorene benyttes det lukkede sløyfesystemet for i$_d$ og i$_q$. Figur 2.7 viser lukket sløyfesystem fra i$_{dref}$ til i$_d$ og i$_{qref}$ til i$_q$.



Figur 2.7: Lukket sløyfesystem for i_d og i_q.

Funksjonene $H_d(s)$ og $H_q(s)$ er definert som henholdsvis overføringsfunksjonene fra $i_{d,ref}$ til i_d og $i_{q,ref}$ til i_q. Ved å se på funksjonene som første ordens lavpassfilter kan de uttrykkes med båndbredden til strømregulatoren α_c [6]:

$$H_d(s) = \frac{i_d}{i_d, ref} = \frac{\alpha_c}{s + \alpha_c} = \frac{\frac{\alpha_c}{s}}{1 + \frac{\alpha_c}{s}} = \frac{F_{cd}(s) * G_{cd}(s)}{1 + F_{cd}(s) * G_{cd}(s)} \tag{2.21}$$

$$H_q(s) = \frac{i_q}{i_q, ref} = \frac{\alpha_c}{s + \alpha_c} = \frac{\frac{\alpha_c}{s}}{1 + \frac{\alpha_c}{s}} = \frac{F_{cq}(s) * G_{cq}(s)}{1 + F_{cq}(s) * G_{cq}(s)} \tag{2.22}$$

Dermed kan overføringsfunksjonene $F_{cd}(s)$ og $F_{cq}(s)$ bli funnet med å sette inn ligning 2.20 og uttrykke ligningene på PI-regulator form:

$$F_{cd} = \frac{\alpha_c}{s}\frac{1}{G_{cd}} = \frac{\alpha_c}{s}(L_d s + R_s + R_{ad}) = \alpha_c L_d + \frac{\alpha_c(R_s + R_{ad})}{s} = K_{pd} + \frac{K_{id}}{s} \tag{2.23}$$

$$F_{cq} = \frac{\alpha_c}{s}\frac{1}{G_{cq}} = \frac{\alpha_c}{s}(L_q s + R_s + R_{aq}) = \frac{\alpha_c}{s}(L_q s + R_s + R_{aq}) = K_{pq} + \frac{K_{iq}}{s} \tag{2.24}$$

For å minske avvik mellom referansestrøm og tilbakekoblet strømmer blir det brukt aktiv demping [18]. Metoden går ut på å implementere en aktiv resistans inne i kontrollsløyfen.

Aktiv demping R_{ad} og R_{aq} kalkuleres som [6]:

$$\begin{aligned} R_{ad} &= \alpha_c L_d - R_s \\ R_{aq} &= \alpha_c L_q - R_s \end{aligned} \tag{2.25}$$

Løser ligning 2.23 og 2.24 for proporsjonalkonstant K_p og integrasjonskonstant K_i. Setter inn ligning for aktiv demping 2.25 og finner kontrollparametrene uttrykt med båndbredde og motorparametre:

$$\begin{aligned} K_{pd} &= \alpha_c L_d \\ K_{pq} &= \alpha_c L_q \\ K_{id} &= \alpha_c^2 L_d \\ K_{iq} &= \alpha_c^2 L_q \end{aligned} \tag{2.26}$$

Strømregulatorene må ta hensyn til spenningen som vekselretteren klarer å levere til motoren. Dette gjøres med å implementere metningsgrenser i regulatorene. De passer på at utgangssignalet ikke overstiger spenningsgrensen. Når grensene blir nådd oppstår integral windup. Integraldelen i regulatoren fortsetter å akkumulere avviket samtidig som utgangsspenning ligger på maksgrensen [11]. Når tilbakekoblet strøm i nærmer seg i_{ref} har integratoren bygget seg opp slik at v_{ref} vil fortsette å være stor [6].

For å forhindre windup, bruker strømregulatoren anti-windup teknikk. Avviksignalet til integratoren blir modifisert. Så lenge spenningen ikke overstiger grenseverdien til metiningen er avviksignalet uendret. Hvis spenningsverdien når metningsgrensen blir avviksignalet modifisert.

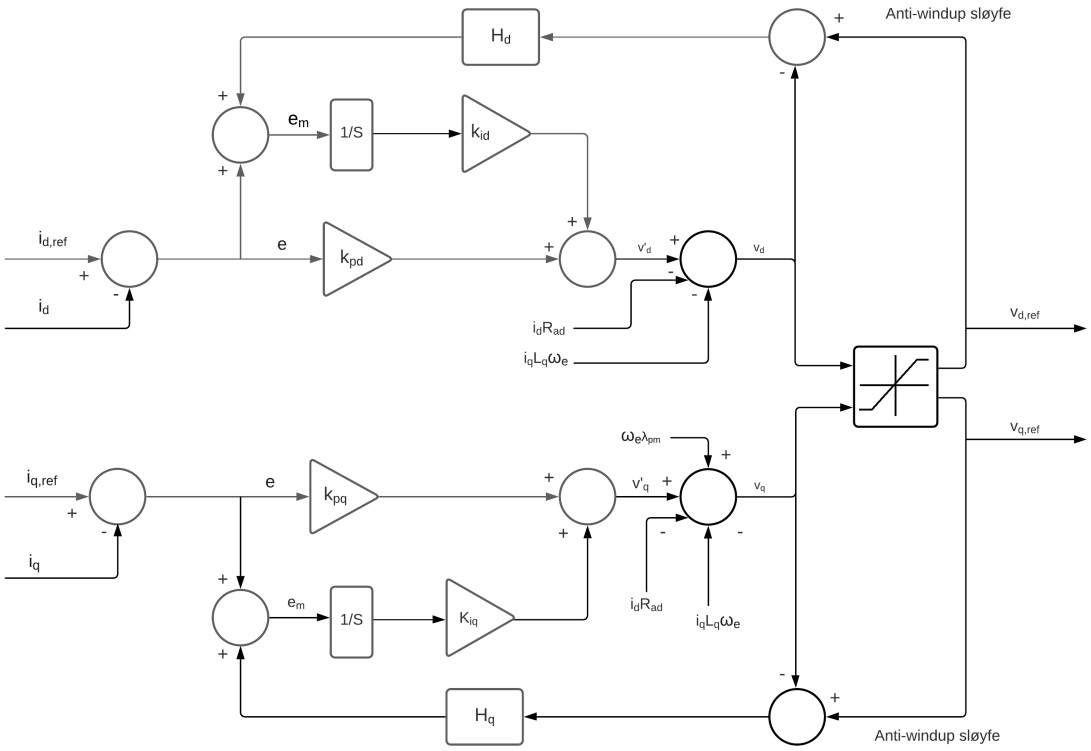
Modifisert avvik, e_m til integratordelen beregnes slik:

$$\begin{aligned} -V_{max} < v_{ref} < V_{max} &\Rightarrow e_m = e \\ -V_{max} > v_{ref} > V_{max} &\Rightarrow e_m = e + H(v_{ref} - v) \end{aligned} \tag{2.27}$$

Hvor H er forsterkning for anti-windup sløyfe, og v er spenning før metning. Anti-windup forsterkning for i_d og i_q settes til [6]:

$$\begin{aligned} H_d &= \tfrac{1}{K_{pd}} \\ H_q &= \tfrac{1}{K_{pq}} \end{aligned} \tag{2.28}$$

Figur 2.8 viser det endelige regulatorsystemet for i_d og i_q med metning og anti-windup implementert.



Figur 2.8: Strømregulator med anti-windup og aktiv demping.

2.3.3 Hastighetsregulator

For å bestemme parametrene til hastighetsregulatoren blir det brukt samme metode som for strømregulatorer, IMC. Hatighetsregulatoren består av en PI-regulator der kontrollparametrene er bestemt fra båndbredden til hastighetsregulatorsløyfen og maskinparametrene B og J. Ligning som beskriver mekaniske forhold til motoren:

$$\frac{d\omega_m}{dt} = \frac{1}{J}(T_e - T_{Last} - B\omega_m) \tag{2.29}$$

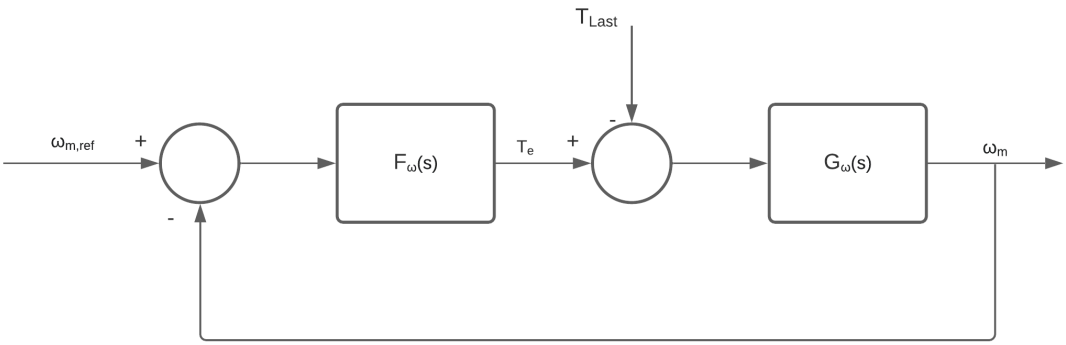
Gjennomfører Laplace transformasjon for hastighetsuttrykket:

$$\omega_m(s) = \frac{1}{Js + B}(Te - T_{Last}) \tag{2.30}$$

Overføringsfunksjonen $G_\omega(s)$ blir overføringsfunskjonen fra T_e-T_{Last} til ω_m:

$$G_w(s) = \frac{1}{Js + B} \tag{2.31}$$

Figur for lukket sløyfe hastighetskontroll:



Figur 2.9: Lukket sløyfesystem for hastighet ω_m

Overføringsfunksjonen for den lukka sløyfa, $H_\omega(s)$:

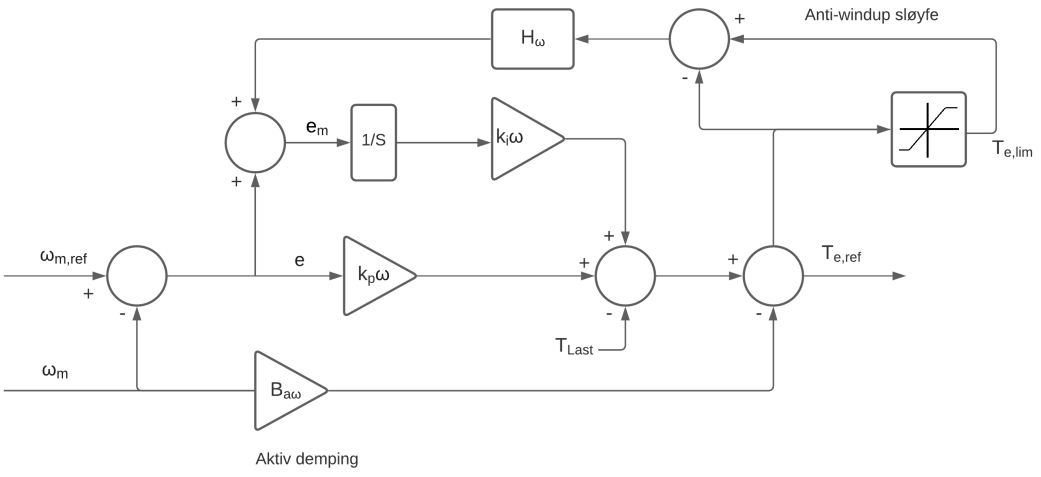
$$H_w(s) = \frac{\omega_m}{\omega_{m,ref}} = \frac{F_\omega G_{\omega,}}{1 + F_\omega G_\omega} = \frac{\frac{\alpha_\omega}{s}}{1 + \frac{\alpha_\omega}{s}} \tag{2.32}$$

Finner uttrykket for $F_\omega(s)$ ved å sette inn uttrykket for $G_\omega(s)$, ligning 2.31, inn i ligning 2.32. Deretter kan den bli løst for å finne kontrollparametrene $K_{p,\omega}$ og $K_{i,\omega}$:

$$F_\omega = \frac{\alpha_\omega}{s}(sJ + B) = \alpha_\omega J + \frac{\alpha_\omega}{s} = K_{p,\omega} + \frac{K_{i,\omega}}{s} \tag{2.33}$$

$$\begin{aligned} K_{p,\omega} &= \alpha_\omega J \\ K_{i,\omega} &= \alpha_\omega B \end{aligned} \tag{2.34}$$

Implementere aktiv demping og anti-windup i hastighetsregulatoren som i strømregulatoren. Figur 2.10 viser oppbygging for hastighetregulatoren.



Figur 2.10: Hastighetregulator med anti-windup og aktiv demping.

Aktiv demping for hastighetsregulator:

$$B_{a\omega} = \alpha_\omega J - B \tag{2.35}$$

H_ω er forsterkning for anti-windup sløyfen i hastighetsregulator. Forsterkningen må ikke forveksles med Hω(s) som er overføringsfunksjon for lukket sløyfesystem for hastighet. H_ω blir bestemt av:

$$H_w = \frac{1}{K_{p,\omega}} \tag{2.36}$$

Nye kontrollparametre for hastighetsregulator blir:

$$K_{p,\omega} = \alpha_\omega J K_{i,\omega} = \alpha_\omega (B + B_{a\omega}) \tag{2.37}$$

2.3.4 Maskimalt dreiemoment per ampere

Hastighetsregulatoren setter referansen til drivmomentet $T_{e,ref}$. Det finnes uendelig mange kombinasjoner av i_d og i_q som kan gi samme $T_{e,ref}$. Maksimalt dreiemoment per ampere er en kontrollstrategi for å finne den kombinasjonen av i_d og i_q som produserer maksimum dreimoment for spesifiserte verdiene av statorstrømmene i_d og i_q.

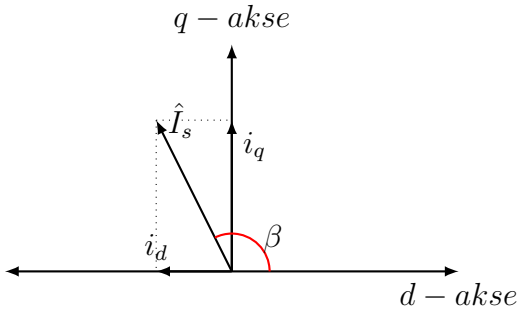
For å finne referansestrømmene som gir MTPA må ligningene for elektromagnetisk dreiemoment undersøkes:

$$T_e = 1.5p[\lambda_{pm} i_q + (L_d - L_q) i_d i_q] \tag{2.38}$$

Ligningen består av to komponenter som produserert dreiemoment, det er magnetisk dreiemoment og reluktans dreiemoment:

$$\begin{aligned} T_e &= T_{mag} + T_{rel} \\ T_{mag} &= 1.5p\lambda_{pm} i_q \\ T_{rel} &= 1.5p(L_d - L_q) i_d i_q \end{aligned} \tag{2.39}$$

Uttrykkene for elektromagnetisk dreiemoment må uttrykes med dreiemomentvinkelen β. β er vinkelen mellom d-aksen og vektor for amplituden på statorstrømmen, $\hat{I}_s$. Vinkelen går fra 0 til 180 grader.



Figur 2.11: Dreiemomentvinkel β

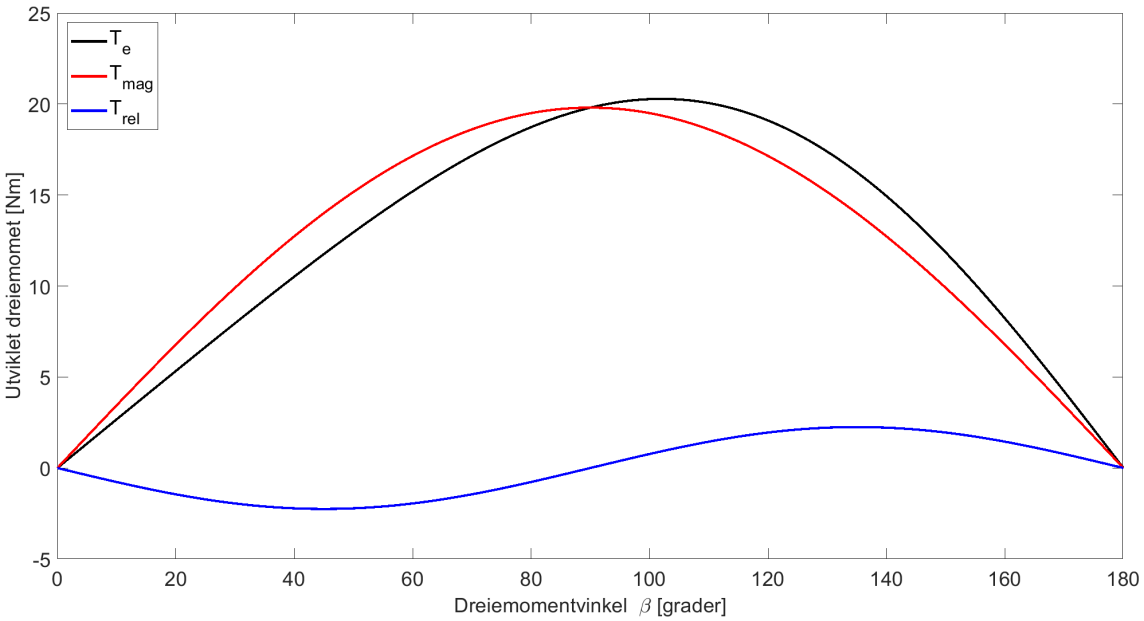
Statorstrømmer i_d og i_q uttrykt med dreiemomentvinkel β:

$$\begin{aligned} \hat{I}_s &= \sqrt{i_d^2 + i_q^2} \\ i_d &= \hat{I}_s cos(\beta) \\ i_q &= \hat{I}_s sin(\beta) \end{aligned} \tag{2.40}$$

For SM-PMSM uten utpregning er d-akse induktans lik som q-akse induktans og dermed vil reluktans bidraget falle bort i uttrykket for T_e. For SM-PMSM er det mest hensiktsmessig å sette $i_d = 0$ og $i_q = \hat{I}_s$ som referansestrømmer til strømregulator.

Siden q-akse induktansen er større enn d-akse induktansen i IM-PMSM finnes det et reluktans bidrag, og uttrykket for MTPA blir annerledes.
Setter inn ligninger for i_d og i_q fra 2.40 inn i ligning 2.38.

$$T_e = 1.5[\lambda_{pm}\hat{I}_s sin(\beta) + (L_d - L_q)\hat{I}_s^2 \frac{sin2\beta}{2}] \tag{2.41}$$



Figur 2.12: Utviklet dreiemoment som funksjon av dreiemomentvinkel β. Figuren viser dreiemoment for en eksempel-PMSM med utpregning. p = 3, L_d = 0.005 [H], L_q = 0.010 [H], λ_{pm} = 0.44 [Wb], $\hat{I}_s$ = 20 [A].

Figur 2.12 viser ett eksempel over hvordan komponentene i ligning 2.41 påvirker produsert dreiemoment når T_e er en funksjon av β. For en PMSM med utpregning vil maksimum T_e ligge mellom toppunktene til T_{mag} og T_{rel}. Figuren viser også at en IM-PMSM produserer mer dreiemoment enn en SM-PMSM der $T_e = T_{mag}$.

For å finne dreiemomentvinkelen som gir maksimalt dreiemoment blir den deriverte av dreiemomentet satt til null, med hensyn på dreiemomentvinkel β:

$$\frac{dT_e}{d\beta} = 1.5(\lambda_{pm}\hat{I}_s cos\beta + (L_d - L_q)\hat{I}_s^2 cos2\beta) = 0 \tag{2.42}$$

Setter inn uttrykk for i_d og i_q fra ligning 2.40 inn i ligning 2.42 og får uttrykket som gir maksimalt dreiemoment uttrykt med i_d og i_q:

$$\lambda_{pm} i_d + (L_d - L_q)(i_d^2 - i_q^2) = 0 \tag{2.43}$$

Setter inn i_q uttrykt fra med i_d og $\hat{I}_s$, ligning 2.44 inn i ligning 2.43 for å finne løsningen for i_d.

$$i_q = \sqrt{\hat{I}_s^2 - i_d^2} \tag{2.44}$$

$$2(L_d - L_q)i_d^2 + \lambda_{pm} i_d - (L_d - L_q)\hat{I}_s^2 = 0 \tag{2.45}$$

Det finnes to løsninger for i_d i ligning 2.45. For IM-PMSM er induktansen gjennom d-aksen normalt mindre enn induktansen gjennom q-aksen.

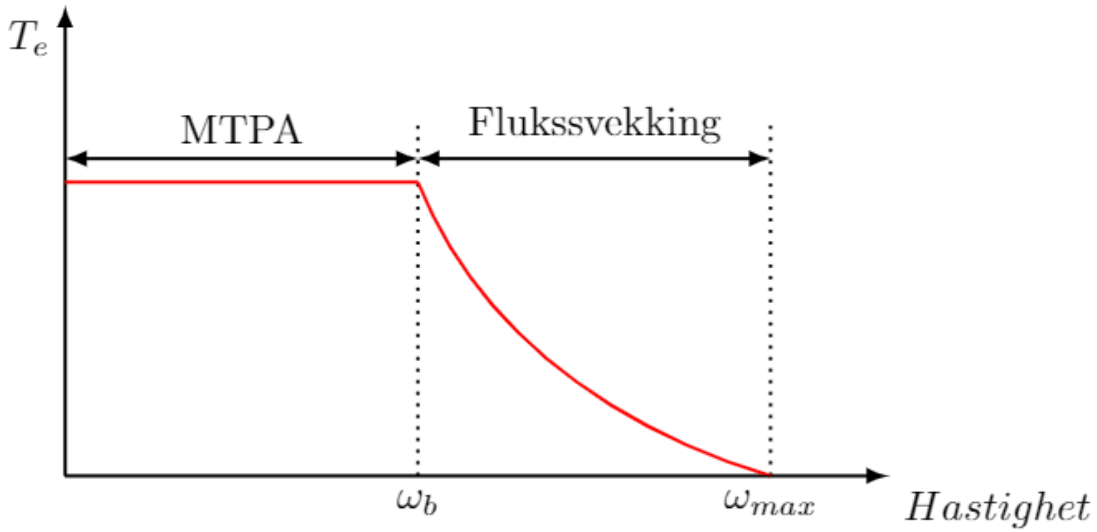
Dermed vil leddet ($L_d - L_q$) være negativt. En positiv verdi av i_d gir negativt T_{rel}. Derfor er det aldri ønskelig med en positiv verdi for i_d. Velger den løsningen av ligning 2.45 som gir en negativ verdi for i_d, og finner uttrykkene for d- og q-akse strømmene som gir MTPA:

$$i_{d,MTPA} = \frac{\lambda_{pm} - \sqrt{\lambda_{pm}^2 + 8(L_q - L_d)^2 \hat{I}_s^2}}{4(L_q - L_d)} \tag{2.46}$$

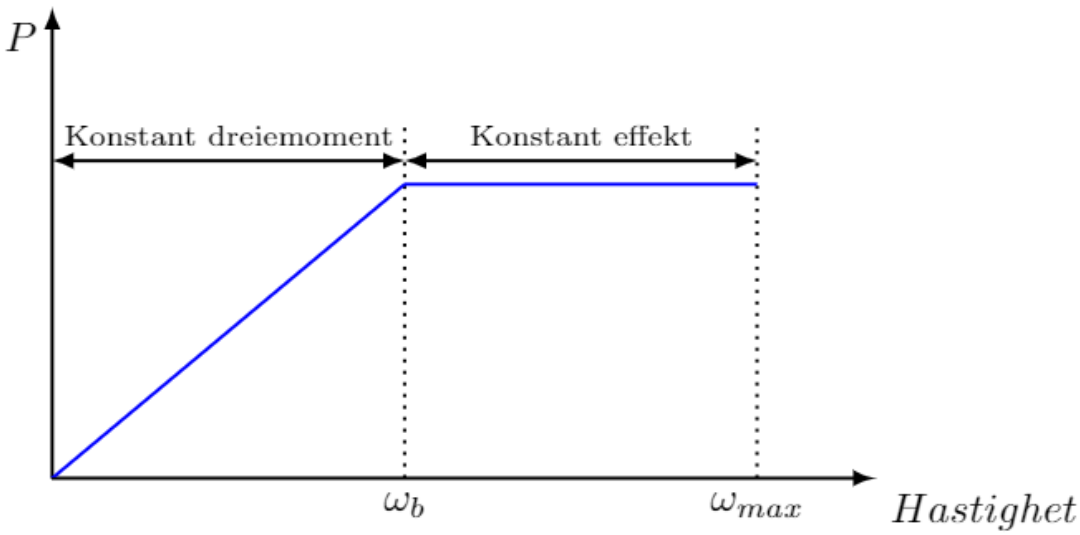
$$i_{q,MTPA} = \sqrt{\hat{I}_s^2 - i_d^2} \tag{2.47}$$

2.3.5 Flukssvekking

For PMSM drivsystem brukt til trekkraft som elektriske fartøy, er det typisk at drivsystemet krever et bredt hastighetsområde med konstant effekt. Ved høye hastigheter vil vekselretteren få problemer med å drifte systemet. Det kommer av at motindusert spenning er proporsjonal med hastighet ω_m og luftgap fluks. Når motindusert spenning blir større enn maksimum utgangspenning for drivsystemet vil ikke permanentmagnet maskinen klare å trekke strøm. Dermed vil ikke motoren klare å utvikle dreiemoment så lenge fluks i luftgapet ikke blir svekket [8]. Basehastighet ω_b er definert som maksimum motorhastighet ved nominell spenning og nominell last. For å kjøre motoren med hastighet over ω_b må fluks i luftgap svekkes. Fluks som kommer av permanentmagneter kan svekkes indirekte med avmagnetisering. Under fluksvekking må statorstrømmer sette opp en magnetomotorisk spenning, MMF, som motvirker den tilsynelatende MMF satt opp av permanentmagnetene. Resultatet er svekket fluks i luftgapet [8]. Motoren leverer konstant dreiemoment med hastigheter mindre enn ω_b og konstant effekt med hastigheter over ω_b.



Figur 2.13: Dreiemoment/hastighet -graf. Viser MTPA-området og flukssvekking-området for hastighet.



Figur 2.14: Effekt/hastighet -graf. Viser konstant dreiemoment-området og konstant effekt-området for hastighet

3 Metode

3.1 Prosjektgruppen

Administrative oppgaver har vært fordelt mellom gruppen. Det ble ikke designert en leder, ettersom kommunikasjon innad i gruppen ble ansett som god. Gruppen var enige i oppgavens retning og omfang. Gruppen foreslo å ha møter annenhver uke med veileder og arbeidsgiver. Arbeidsgiver mente at det ikke ville være nødvendig. Møter ble satt opp etter behov. På grunn av koronasituasjonen ble møtene gjennomført via Microsoft Teams.

3.2 Utgangspunktet til modellen

Gruppen måtte velge en motor som den digitale modellen skulle baseres på. HPS kom med forslag på motorer, som hadde blitt vurdert til turistbåten. Det ene forslaget krevde mye tverrfaglig kompetanse og tid, så gruppen valgte å jobbe med det andre forslaget. Databladet til den andre motoren viste seg å mangle essensiell informasjon som var nødvendig for å gjennomføre oppgaven. Etter et mislykket forsøk på å kontakte produsenten kom HPS med data fra en tilsvarende motor fra et tidligere prosjekt. Oswald MFS 13.3-6 ble derfor utgangspunktet.

Figur 3.1: Oswald MFS 13.3-6. Bilde av motoren som HPS har brukt i tidligere prosjekt. Motorparametre er hentet fra denne motoren Databladet for motoren er vist i vedleg A og B

3.3 Programvare

Det har blitt brukt en rekke programmer og programvare for å gjennomføre oppgaven. MATLAB, Simulink, Python og PyCharm har blitt brukt til å utvikle digitale modeller av permanentmagnet synkronmaskiner. Overleaf og Zotero har blitt brukt

til å lage prosjektrapporten. FEMM har blitt brukt til å illustrere motorarmatur. For å tegne figurer er det brukt Latex og Lucidchart.

MATLAB

MATLAB står for Matrix Laboratory, og er et program som er utviklet av selskapet MathWorks. Programmet er spesielt egnet til å løse matematiske problemer. MATLAB støtter også en rekke tilleggspakker som gjør det lettere å analysere, teste, og simulere forskjellige områder som elektriske maskiner, robotikk, kontrollsystemer og signalbehandling [9].

Simulink

Simulink er en av tilleggspakkene som kan skaffes til MATLAB. Programmet er et MATLAB basert grafisk utviklingsmiljø for modellering, simulering og analyse av dynamiske systemer. Simulink bruker i hovedsak blokkprogrammering til å utvikle systemer. Simulink støtter forskjellige tilleggspakker som legger til nye blokker og funksjoner i programmet [16].

Python

Python er et objektorientert programmeringsspråk, som ble laget av Guido van Rossum og utgitt i 1991. Python er spesielt egnet for utvikling av enkle verktøy og applikasjoner. Programmeringsspråket har som filosofi at det skal være lett å lese. Python har et stort bibliotek av kode og funksjoner som språket kan utvides med [19].

Pycharm

PyCharm er et integrert utviklingsmiljø som blir brukt til å utvikle programvare. PyCharm er laget for å utvikle program med programmeringsspråket Python. PyCharm støtter mange funksjoner som gjør det lettere å utvikle programmer i Python [5].

Overleaf

Overleaf er et nettbasert LaTeX redigeringsprogram. Programmet tillater at flere personer redigerer et prosjekt samtidig, og lar brukerne se hverandres forandringer i sanntid. Programmet støtter forskjellige pakker som gjør det lettere å sette opp bilder, grafer, tabeller og formler på en ryddig måte [12].

Zotero

Zotero er et program for å håndtere, samle inn og sitere forskjellig litteratur og bibliografier. Programmet lagrer kilder på standardform [1].

Finite Element Method Magnetics (FEMM)

FEMM er et program for å tegne og løse lavfrekvente elektromagnetiske problemer på et todimensjonalt plan. Programmet kan brukes til å tegne motorens armatur og vise fordelingen av magnetfelt [3].

3.4 Simulasjon

Den første modellen ble laget i Simulink. Simulink er et program gruppen har et forhold til. Gruppen hadde også allerede Matlab lisens, som studenter ved NTNU. Det ble funnet ferdiglagde Simulink-modeller av drivsystem og PMSM på nettet. Modellene ble brukt til å utforske hvordan PMSM fungerer og som inspirasjon for egen modell i Simulink.

3.5 Validering og verifisering

Validering og verifisering av digitale simuleringsmodeller er en prosess som skal handler om å forsikre at den digitale modellen er sikker og nøyaktig. For å verifisere modellene blir resultatene fra simuleringene sammenlignet med teoretisk kunnskap om systemet, fysiske tester av lignende system, og andre digitale modeller som er validert. Siden en digital modell bare er en tilnærming av et fysisk system så vil de aldri bli 100% nøyaktige. Derfor burde modellen bli validert til den grad anvendelsen av modellen krever.

3.6 Testoppsett for Python- og Simulinkmodell

3.6.1 Fremgangsmåte for simulering i Python

For å gjennomføre simuleringer med Python-modellen så er det nødvendig med en PC som har PyCharm eller lignende program installert. For å kjøre programmet blir følgende steg gjort:

1. Åpne filen "PMSM_Python-modell" i PyCharm.
2. Hvis det er første gangen filen blir kjørt så må utvidelsene programmet bruker bli installert. Utvidelsene er numpy og matplotlib.pyplot.
3. Parameterene brukt i modellen er fra PM-maskinen Oswald MFS13.3-6W. Hvis en annen maskin skal simuleres, så må parameterene i filen "PMSM_Parameters" oppdateres.
4. Velg verdi for referansehastighet og lastmoment. Det blir gjort ved å forandre på verdien til variablene "rpm_ref" og "T_l" i filen "Intialverdier.py".
5. Start programmet ved å kjøre filen "main.py".

6. Når programmet kjører spør det om hvor lang simuleringtiden skal være. Skriv inn ønsket tid i sekunder og trykk enter.

7. For å se resultatene fra simuleringen se dokumentet "Results.txt", alternativt så kan koden i vedlegg F.5 bli brukt til lage grafer av resultatene.

3.6.2 Fremgangsmåte for simulering i Simulink

For å gjennomføre simuleringer med Simulink-modellen er det nødvendig med en PC som har Matlab instalert og Matlab lisens. Modellen er laget i versjon R2021a, men det er mulig å eksportere filen til en tideligere versjon. For å kjøre programmet blir følgende steg gjort:

1. Pakk ut Zip filen til ønsket destinasjon.
2. Åpne og kjør filen PMSM_Drive_script.m.
3. Åpne Simulink filen PMSM_Drive_model.slx.
4. Sørg for at verdier er lastet inn i modellen. Blokker er markert i rødt, dersom de ikke er lastet inn.
5. Velg ønsket simulasjonstid i Simulink.
6. Velg referansehastighet i hastighet_ref blokken.
7. Velg størrelse og oppførsel for lastmoment i lastmomentblokken.
8. Kjør Simulink-modellen.
9. Åpne "Simulation Data Inspector" for å se ønsket grafer.

Se vedlegg D for ytterligere informasjon om fremgangsmåte og tilording av parametre.

3.6.3 Testscenarioer for Python- og Simulink-modell

PM-maskinen brukt til gjøre testene er Oswald MFS13.3-6W. Parametere til maskinen er tatt fra databladet og funnet ved målinger. For flere detaljer om maskinen se datablad i vedlegg A og B. Det er brukt forskjellige verdier for B i Python og Simulink. B_{Py} er verdien brukt i Python, og B_{Sim} er verdien brukt i Simulink.

Tabell 2: Oswald MFS13.3-6W testparametere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm

Testscenario 1: Tomgangstest
Det første testscenarioet av modellen skjer på tomgang. Det blir oppnådd med å sette T_{Last} til null under hele simuleringen. Se vedlegg H.1 for flere detaljer.

Tabell 3: Simuleringsparametere: Tomgangstest

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	0.2	s
T_{Last}	0	Nm
Referansehastighet	2150	rpm

Testscenario 2: Sprang i last - nominell hastighet
Det andre testscenraioet av modellen skjer med et sprang i lastmomentet (T_{Last}). Spranget skjer etter 0.2 sekunder og verdien for T_{Last} går fra 0 til 0.5*T_{nom}. Se vedlegg H.2 for flere detaljer.

Tabell 4: Simuleringsparametere: Sprang i last

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	0.4	s
Sprangtid	0.2	s
$T_{Last,1}$	0	Nm
$T_{Last,2}$	$0.5 * T_{nom}$	Nm
Referansehastighet	2150	rpm

Testscenario 3: Drivsystem uten hastighetskontroll
Det tredje testscenarioet av modellen skjer med konstante verdier for $i_{d,ref}$ og $i_{q,ref}$.

Det blir oppnådd ved å kommentere ut de delene av programmene som styrer $T_{e.ref}$, $i_{d,ref}$ og $i_{q,ref}$. Når den delen av koden er frakoblet settes $i_{d,ref}$ og $i_{q,ref}$ til konstante verdier. Se vedlegg H.3 for flere detaljer.

Tabell 5: Simuleringsparametere: Drivsystem uten hastighetskontroll

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	3.0	s
T_{Last}	$0.5 * T_{nom}$	Nm
$i_{d,ref}$	-5	A
$i_{q,ref}$	140	A

Testscenario 4: Stor sprang i last - høy hastighet
I det fjerde testscenarioet blir modellene testet med en referansehastighet som er større enn den nominelle hastigheten til maskinen. Etter 0.2 sekunder blir det lagt inn en last på 90% av T_{nom}, som er mer dreiemoment enn det maskinen burde klare å levere ved denne hastigheten. Se vedlegg H.4 for flere detaljer.

Tabell 6: Simuleringsparametere: Stort sprang i last - høy hastighet

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	0.4	s
Sprangtid	0.2	s
$T_{Last,1}$	0	Nm
$T_{Last,2}$	$0.9 * T_{nom}$	Nm
Referansehastighet	3000	rpm

Testscenario 5: Varierende turtallsreferanse
I det femte testscenarioet blir modellen testet med en varierende hastighetsreferanse. Hastighetsreferansen er et sinussignal med amplitude på 1000 rpm og en frekvens på 0.5 Hz. Se vedlegg H.5 for flere detaljer.

Tabell 7: Simuleringsparametere: Varierende turtallsreferanse

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	4	s
T_{Last}	0	Nm

Tabell 8: Simuleringsparametere: Sinusreferanse

Turtallsreferanse		
Bølgeform	Sinus	-
Faseforskyvning	0	-
Amplitude	1000	rpm
Frekvenes	0.5	Hz

4 Modell og simulering

For å løse oppgaven har det blitt utviklet to modeller av permanentmagnet synkronmaskinen, en i MATLAB/Simulink og en i Python. Kapittelet tar for seg hvordan de to modellene er bygd opp og hvilke løsninger som har blitt brukt i utviklingen av modellene.

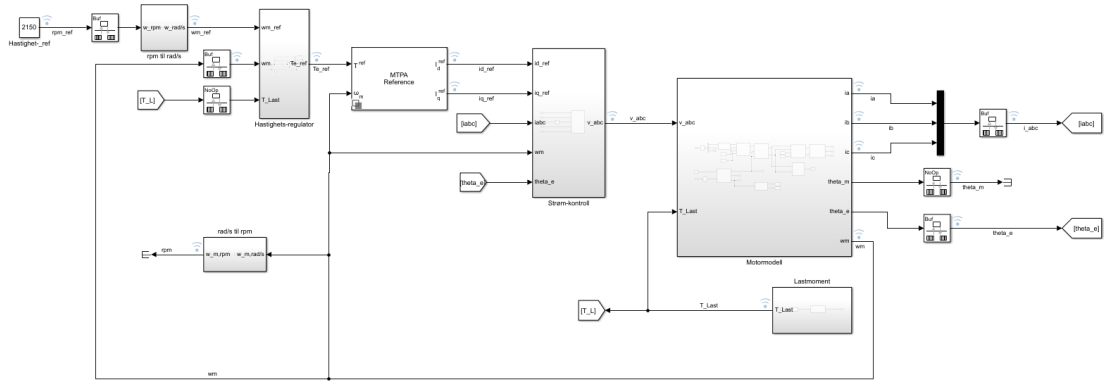
4.1 Simulink-modell

Modellen i Simulink består av et Matlab script og en Simulink modell. Konstante motorparametere og simuleringsparametere er deklarert og tilordnet verdier i Matlab scriptet, som må kjøre sammen med Simulink-modellen. Større bilder av Simulink-modellen kan bli funnet i vedlegg E

4.1.1 Kontrollsystem

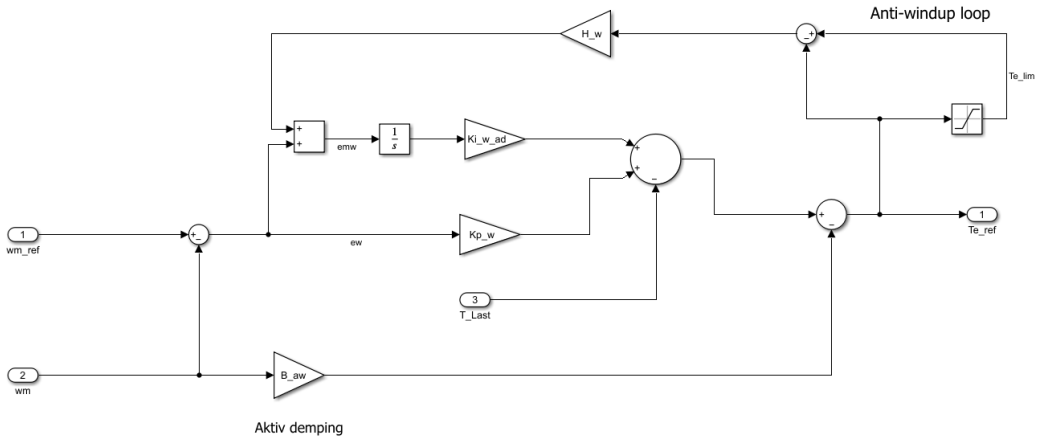
Kontrollsystemet i Simulink er basert på feltorientert kontroll (FOC) strategi for å styre motoren. Kontrollsystemet har en ytre kontrollsløyfe med tilbakekobling av mekanisk hastighet ω_m og en indre kontrollsløyfe med tilbakekobling av statorstrømmer i_{abc}. Referansehastighet tilordnes som turtall i [rpm], men modellen opererer med mekanisk rotasjonshastighet ω_m i [rad/s]. Figur 4.1 viser FOC strukturen for drivsystemet.

Modellen inneholder en hastighetsregulator for kontroll av hastighet, og to strømregulatorer for kontroll av i_d og i_q. Begge regulatorene er basert på teori presentert i kapittel 2.3. Modellen bruker blokken "Rate Transition" for å sette rett sample tid i de ulike delene av modellen.



Figur 4.1: FOC av PMSM i Simulink-modellen.

Figur 4.2 viser hastighetsregulatoren. Det er en PI-regulator med aktiv demping og anti-windup.

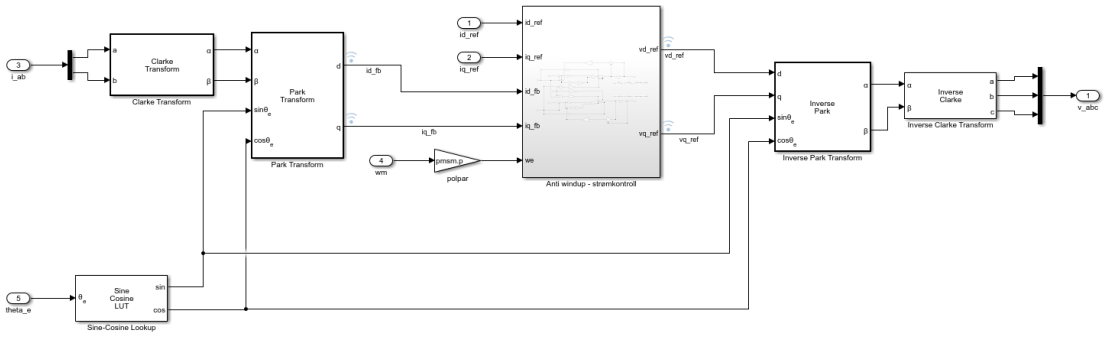


Figur 4.2: Hastighetsregulator i Simulink-modellen.

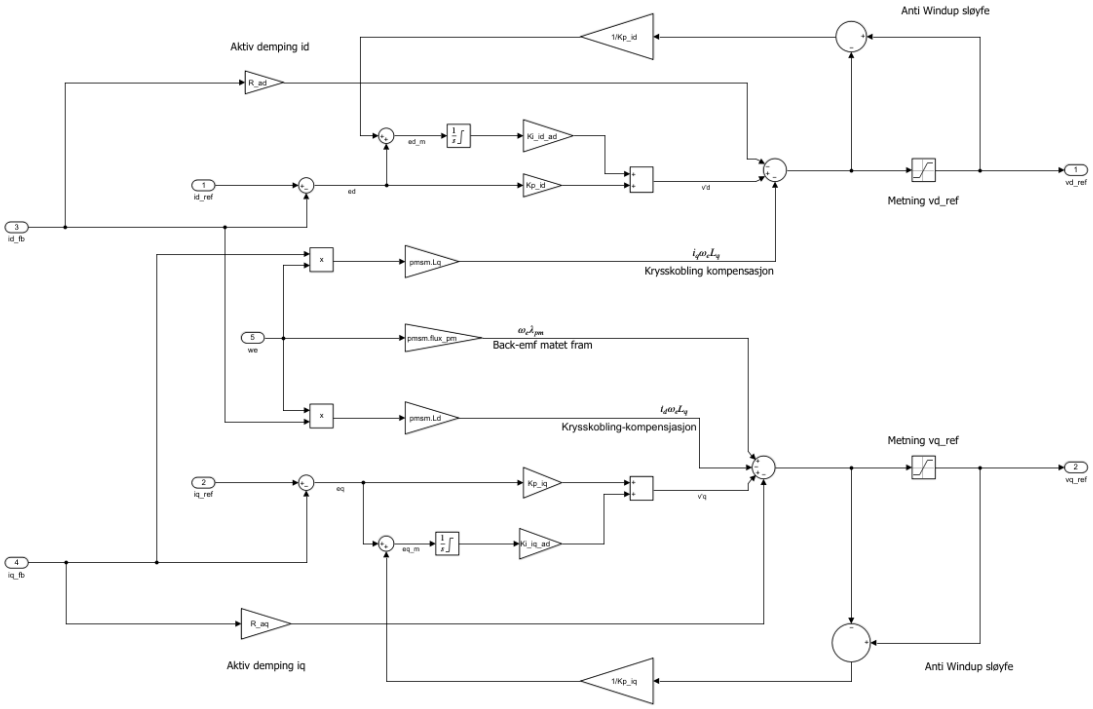
For å sende referansestrømmer til strømregulator bruker modellen den innebygde blokken "MTPA reference". Blokken regner ut $i_{d,ref}$ og $i_{q,ref}$ basert på utgangssignalet til hastighetsregulatoren $T_{e,ref}$ og tilbakekoblet hastighet ω_m sammen med motorparametrene R_s, L_d, L_q, λ_{pm},i_{max}, V_{DC} og ω_b.

Strømkontroll har referansestrømmer, statorstrømmer, hastighet og rotorposisjon som inngangsvariabler. Statorstrømmene i_a og i_b blir transformert til i_d og i_q ved bruk av Clarke og Park transformasjon samt rotor posisjon, Θ_e. Utgangsparametre fra strømregulatoren er $v_{d,ref}$ og $v_{q,ref}$ som blir invers transformert til v_{abc}. Se figur 4.3.

Inne i strømregulatoren reguleres i_d og i_q etter teori presenter i kapittel 2.3.2. For å kontrollere i_d og i_q brukes to PI regulatorer, krysskobling kompensasjon, motindusert spenning er matet fram, aktiv demping og anti windup. Se figur 4.4.



Figur 4.3: Strømkontrollsystem i Simulink-modellen.

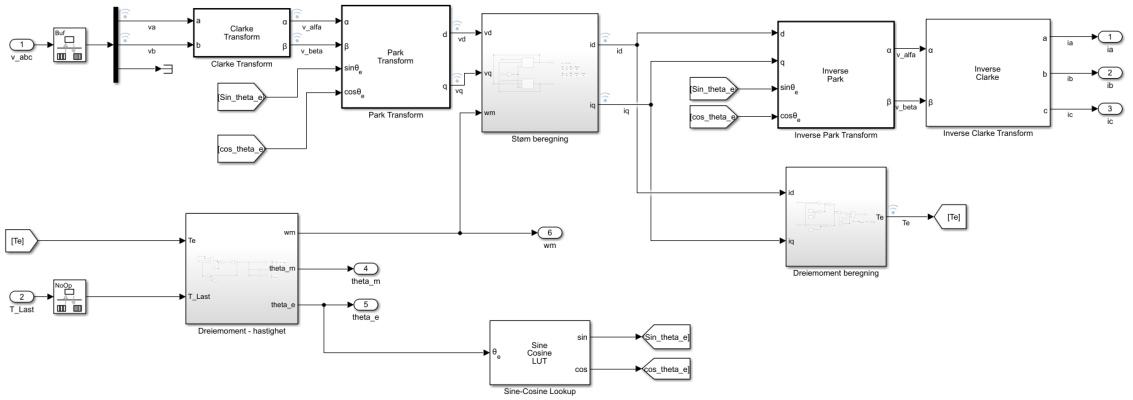


Figur 4.4: Strømregulator i Simulink-modellen.

4.1.2 PMSM-modell

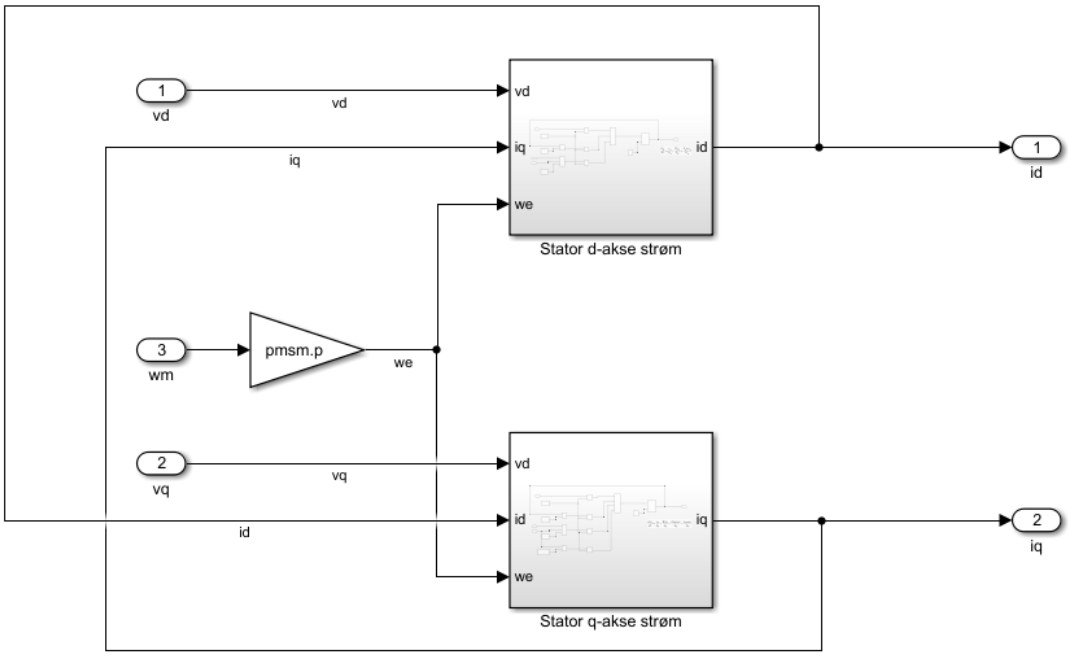
Motormodellen av PMSM har fasespenning v_{abc} og lastmoment T_{Last} som inngangsvariabler. Fasespenningene transformereres til dq-referanse rammen før statorspenningene i_d og i_q blir beregnet. Statorstrømmene i_{abc} er tilgjengelig etter invers Park og Clarke transformasjon. Strømmene blir tilbakekoblet til den indre kontrollsløyfen i drivsystemet.

Dreiemoment T_e regnes ut ved bruk av i_d og i_q i blokken "Dreiemoment beregning". Beregnet dreiemoment T_e blir sammen med inngangsparameter T_{Last} brukt til å regne ut hastighet ω_m og rotorposisjon Θ_e. Hastighet ω_m blir tilbakekoblet til den ytre kontroll sløyfen i drivsystemet. Rotorposisjon Θ_e blir brukt i Clarke og Park transformasjon. Figur 4.5 viser overordnet struktur inne i motormodell-blokken.

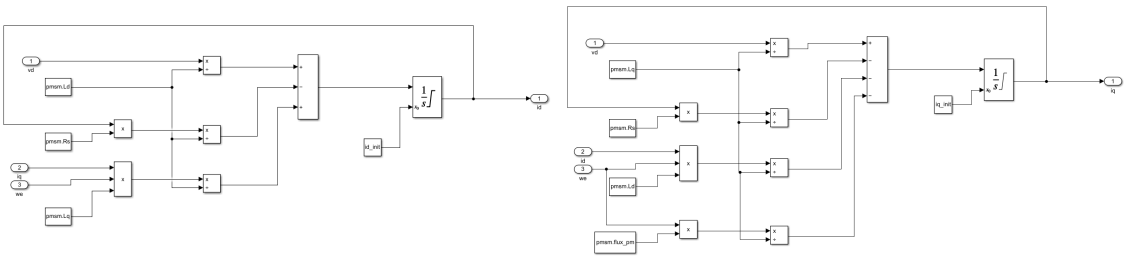


Figur 4.5: PMSM-motormodell i Simulink-modellen.

Figur 4.6 og 4.7 viser blokkene for utregning av stator d- og q-aksestrømmene ved å bruke formel 2.7 og 2.8.



Figur 4.6: Beregning av statorstrømmer i Simulink-modellen, utside.

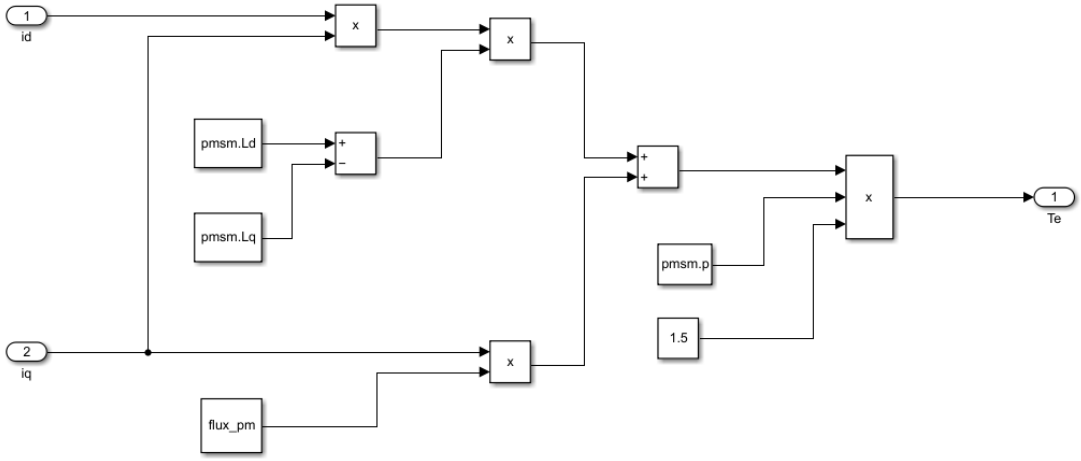


(a) d-akse strøm

(b) q-akse strøm

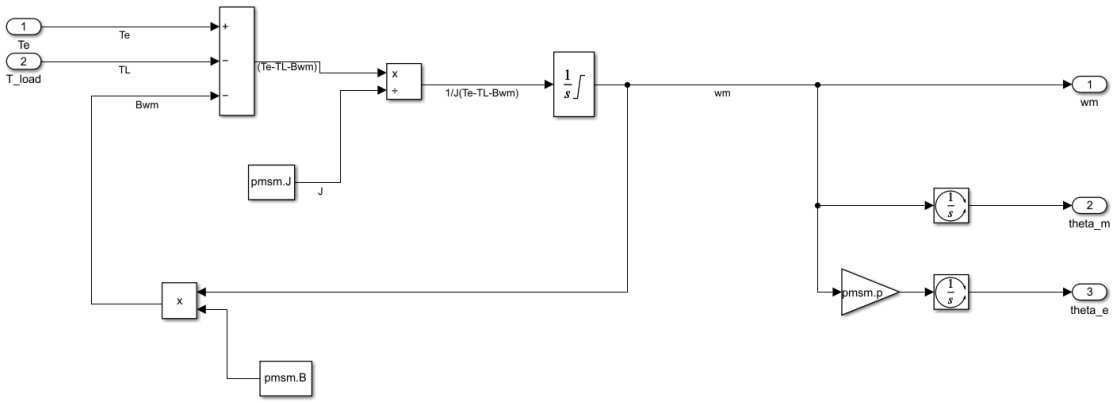
Figur 4.7: Beregning av statorstrømmer i Simulink-modellen, inside.

Figur 4.8 viser blokken som beregner elektromagnetisk dreiemoment ved å bruke formel 2.9.



Figur 4.8: Beregning av dreiemoment T_e i Simulink-modellen.

Figur 4.9 viser blokken som beregner mekanisk rotasjonshastighet ω_m [rad/s], og elektrisk rotorposisjon Θ_e [rad] ved å bruke formlene 2.10, 2.11, 2.12 og 2.13.



Figur 4.9: Beregning av ω_m og Θ_e i Simulink-modellen

4.2 Python-modell

Python-modellen av PM-maskinen og kontrollsystemet tar utgangspunkt i Simulink-modellen av PM-maskinen. Python-modellen består av flere Pythonscripts. Et hovedscript, et script for funksjonene som brukes, et script for motorparametere og et script for initialverdier.

4.2.1 Kontrollsystem

Kontrollsystemet i Python benytter seg av FOC metoden for å kontrollere PM-maskinen. Kontrollsystemet består av tre PI-regulatorer som styrer turtallet ved å kontrollere spenningene som sendes til PM-maskinen. Regulatorene tar ω_m, i_d og i_q som tilbakekoblinger. Referansehastighetet blir satt av brukeren, mens i_d og i_q referansene blir funnet med MTPA metoden. Koden for kontrollsystemet blir kan bli sett i vedlegg F.1 linje 25-33. Koden for funksjonene brukt i main.py kan bli funnet i vedlegg F.2.

Regulatoren som kommer først i systemet er hastighetsregulatoren. For å styre turtallet setter brukeren en turtallsreferanse. Referansen blir omgjort til en vinkelfrekvensreferanse som herfra kalles $\omega_{m,ref}$. $\omega_{m,ref}$ sendes inn i hastighetsregulatoren der den sammenlignes med tilbakekoblingen, ω_m. Regulatoren lager en referanseverdi for det elektriske dreiemomentet som PM-maskinen skal produsere. Denne referansen kalles $T_{e,ref}$ og beregnes som vist formel 4.1.

$$T_{e,ref} = K_{p,w}(w_{ref} - w_m) + K_{i,w}(w_{ref} - w_m)dt + sum_w \tag{4.1}$$

For å finne i_d og i_q referansene blir MTPA metoden brukt. MTPA blir forklart i kapittel 2.3.4. MTPA kalkulasjonene i Python støtter kun PM-maskiner med reluktansbidrag i dreiemomentet, det vil si maskiner der L_q er større enn L_d. I Python-modellen blir MTPA brukt ved å finne dreiemomentvinekelen β og statorstrøm referansen $I_{s,ref}$. β blir funnet med formel 4.2, der I_s blir satt til maskinens nominelle strøm og resten av verdiene er motorparametere.

$$\beta = arccos(-\frac{\lambda_{pm}}{4(L_d - L_q)I_s} - \sqrt{\frac{1}{2} + ((\frac{\lambda_{pm}}{4(L_d - L_q)I_s})^2)}) \tag{4.2}$$

For å finne $I_{s,ref}$ blir $T_{e,ref}$ og β brukt i formelen 4.3

$$I_{s,ref} = -\frac{\lambda_{pm}}{2(L_d - L_q)cos\beta} + \sqrt{\frac{2}{3n_p(L_d - L_q)sin\beta cos\beta}T_{e,ref} + (\frac{\lambda_{pm}}{2(L_d - L_q)cos\beta})^2} \tag{4.3}$$

Likning 4.2 og 4.3 viser hvorfor MTPA kalkulasjonen kun støtter PM-maskiner med reluktansbidrag. Hvis L_d er lik L_q så vil (L_d - L_q) leddet under brøkstrekene føre til at nevneren i brøken blir null. Det vil da bli umulig å gjennomføre beregningene.

Når verdien for $I_{s,ref}$ er kjent så kan $i_{d,ref}$ og $i_{q,ref}$ bli funnet med trigonometri, som vist i figur 2.11 fra kapittel 2. Metoden brukt til å finne $i_{d,ref}$ og $i_{q,ref}$ er vist i formlene 4.4.

$$\begin{aligned} i_{d,ref} &= I_{s,ref} cos\beta \\ i_{q,ref} &= I_{s,ref} sin\beta \end{aligned} \tag{4.4}$$

PI-regulatorene som lager spenningsreferansene $v_{d,ref}$ og $v_{q,ref}$ tar inn referansene $i_{d,ref}$ og $i_{q,ref}$. Siden modellen ikke tar med vekselretter som skal levere spenning til PM-maskinen, så blir $v_{d,ref}$ og $v_{q,ref}$ sendt rett til PMSM modellen som om det var spenningen fra vekselretteren. $v_{d,ref}$ og $v_{q,ref}$ beregnes ved å bruke formel 4.5 og 4.6.

$$v_{d,ref} = K_{p,d}(i_{d,ref} - i_d) + K_{i,d}(i_{d,ref} - i_d)dt + sum_d + \omega_e L_q i_q \tag{4.5}$$

$$v_{q,ref} = K_{p,q}(i_{q,ref} - i_q) + K_{i,q}(i_{q,ref} - i_q)dt + sum_q + \omega_e L_s i_d + \omega_e \lambda_{pm} \tag{4.6}$$

Sum-leddet i formel 4.1, 4.5 og 4.6 blir funnet ved å summere alle tidligere integrasjonsledd fra simuleringen. Dette blir løst som vist i formel 4.7.

$$sum = sum_{t-1} + K_i(referanseverdi - tilbakekobling)dt \tag{4.7}$$

Leddene som kommer etter sum-leddene i formel 4.5 og 4.6 legger til krysskoblings kompensasjon i regulatorene.

For å etterligne metning i systemet blir det lagt inn øvre og nedre grenser i regulatorene. Det blir lagt inn metningsverdier for $I_{s,ref}$, $v_{d,ref}$ og $v_{q,ref}$. Metningsverdiene er hentet fra spesifikasjonene til vekselretteren som HPS mente er aktuell for drivsystemet. Kodesnutten under viser hvordan PI-regulatoren for $v_{d,ref}$ er blitt laget i Python. Koden er hentet fra vedlegg F.2 linje 47-55.

```
1  def PI_Vd(Kp, Ki, i_dref, i_d, sum_d, dt, V_max, w_e, L_q, i_q):
2      Vd = Kp*(i_dref - i_d) + Ki*(i_dref - i_d)*dt + sum_d + w_e*L_q*i_q
3      # Metningsgrense. Spenningen blir ikke større enn det
4      # vekselretteren kan levere
5      if Vd > V_max:
6          Vd = V_max
7      elif Vd < -V_max:
8          Vd = -V_max
9      sum_d = sum_d + Ki*(i_dref - i_d)*dt
10     return Vd, sum_d
```

Figur 4.10: Pythonkode for i_d-strømregulator

4.2.2 PMSM-modell

Motormodellen i Python har v_d, v_q og T_{Last} som inngangsverdi. Verdien for spenningene blir laget av spenningsregulatorene og lastmomentet blir satt av brukeren. Modellen beregner verdiene for i_d, i_q, ω_m og T_e. Koden for PMSM modellen kan bli funnet i vedlegg F.1 linje 37-55. Koden for funksjonene brukt kan bli funnet i vedlegg F.2.

Alle beregningene gjort i modellen skjer i dq-referanserammen. Det blir gjort ettersom all informasjon om systemet finnes i dq0-referanserammen. Ved å holde seg i dq-referansesystemet blir det gjort færre utregninger, som gjør at simuleringene tar kortere tid.

Den matematiske modellen fra kapittel 2.2.1 blir brukt til å simulere permanentmagnet maskinen. Likningene brukt i Python-modellen er 4.8-4.11.

$$\frac{di_d}{dt} = \frac{1}{L_d}v_d - \frac{R_s}{L_d}i_d + \frac{L_q}{L_d}p\omega_m i_q \tag{4.8}$$

$$\frac{di_q}{dt} = \frac{1}{L_q}v_q - \frac{R}{L_q}i_q - \frac{L_d}{L_q}p\omega_m i_d - \frac{1}{L_q}p\omega_m\lambda_{pm} \tag{4.9}$$

$$T_e = \frac{3}{2}p(\lambda_{pm}i_q + (L_d - L_q)i_d i_q) \tag{4.10}$$

$$\frac{d\omega_m}{dt} = \frac{1}{J}(T_e - T_{Last} - B\omega_w) \tag{4.11}$$

Simuleringen av permanentmagnet maskinen blir gjort i en for-løkke. For-løkken kjøres 40000 ganger for hvert sekund som skal simuleres. Antallet kan forandres ved å forandre på sampletiden i scriptet. I for-løkken blir simuleringen av systemet gjort sekvensielt. Det første steget i hver runde av simuleringen er regulatorene som finner nye inngangsverdier for PMSM-en. Steg to er i finne verdiene for i_d- og i_q-strømmene. Dette blir gjort ved å løse likning 4.8 og 4.9. Det tredje steget er i finne det elektriske dreiemomentet som maskinen produserer. Det blir gjort ved å løse likning 4.10. Det siste steget i simuleringen er å finne den mekaniske vinkelhastigheten, som blir gjort med å løse likning 4.10. I Python-modellen har Eulers metode blitt brukt til å løse differanselikninger. Kodesnutten under viser hvordan Eulers metode blir brukt til å løse likning 4.8 i Python. Koden er hentet fra linje 82-85 i vedlegg F.2.

```
1       def id_calc(v_d, L_d, L_q, R, i_d, i_q, Dt, w_m, P):
2           diddt = (1 / L_d) * (v_d - R * i_d - L_q * P * w_m * i_q)
3           new_id = i_d + Dt * diddt
4           return new_id
```

Figur 4.11: Python kode for beregning av i_d med Eulers metode

Matematisk så kan det formuleres som i 4.12.

$$i_q^t = i_q^{t-1} + \frac{\Delta i_q}{\Delta t^{t-1}} \tag{4.12}$$

4.3 Forenklinger

Det er gjort en rekke forenklinger i begge modellene. Valget av forenklinger er gjort ved å se hvilke forenklinger det er vanlig å gjøre for digitale PMSM-modeller. Det har også blitt sett på hvilke forenklinger som vil ha størst påvirkning på modellens nøyaktighet, og hvor mye tid gruppen har hatt til å implementere funksjoner i modellen.

Forenklingene gjort i Python-modellen er:

- Ingen flukssvekking.
- Ingen vekkselretter/SVPWM.
- Ved ukjent verdi for B, så vil $B\omega_m$ leddet i likning 4.11 bli erstattet med 0.1% av T_{Last}.

Forenklinger gjort i Simulink-modellen er:

- Innebygd Simulink blokk som beregner MTPA og fluksvekking.
- Innebygd Simulink blokk som beregner $\mathrm{Sin}\Theta_\mathrm{e}$ og $\mathrm{Cos}\Theta_\mathrm{e}$.
- Innebygde blokker som gjennomfører Clarke og Park transformasjon.
- Ingen vekkselretter/SVPWM.

5 Resultater

Kapittel 5 presenterer tester som er utført på PMSM modellen i Matlab/Simulink og Python. Utvalgte resultater som fremhever funksjonaliteten til modellene blir trukket frem. Det er brukt samme testscenarioer i Matlab/Simulink. For fullstendige testscenarioer se vedlegg H - Testresultater.
Målet med simuleringene er å verifisere og validere den matematiske modellen og drivsystemet som kontrollerer motoren. Testene skal vise problemer og forbedringer ved modellen, samt vise forskjeller mellom de ulike modellene. I undersøkelsen studeres transient tilstand og stabil tilstand til drivsystemet.

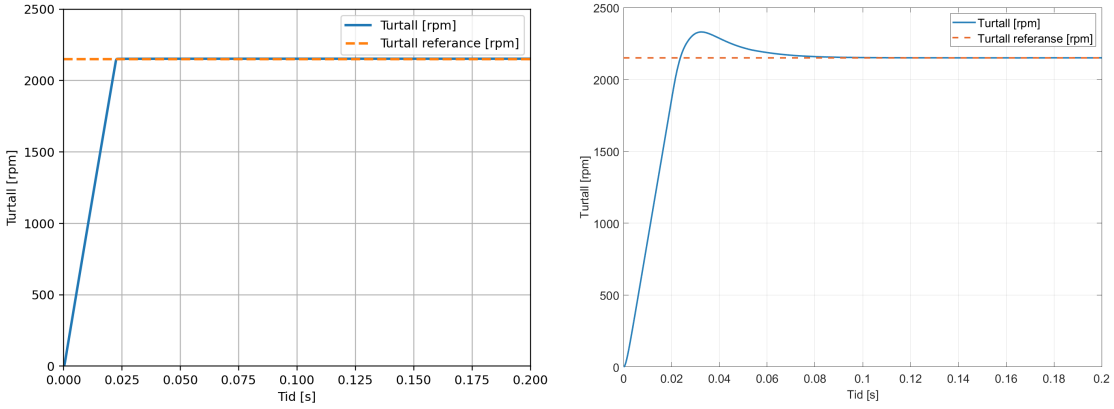
Motorparametre brukt i testene er gitt i tabell 9. Motorparametrene er identiske for begge modellene bortsett fra viskositetskoeffisient B.

Tabell 9: Oswald MFS13.3-6W testparametere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm
V_{DC}	800	V
V_{max}	$\frac{0.95V_{DC}}{\sqrt{3}}$	V

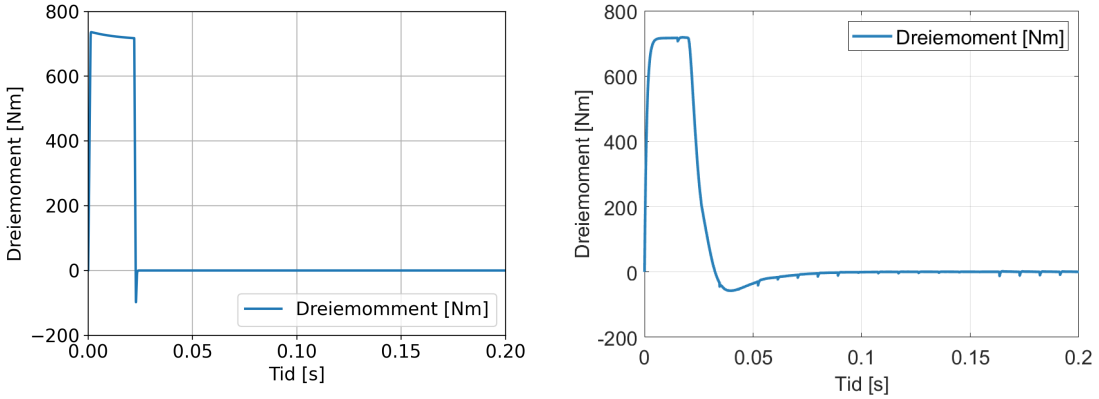
5.1 Tomgangstest og sprang i last - nominell hastighet

I den første testen blir motoren simulert uten ekstern mekanisk last. Motoren skal oppnå og driftes på nominelt turtall. Når motoren kjører på tomgang er motorens friksjon den eneste mekaniske motstanden. Ettersom Python-modellen opererer med viskositetskoeffisient B_{Py} satt til null, blir dreiemomentet $\mathrm{T_e}$ lik 0 ved stabil tilstand. I Simulink er bidraget aktivt, og motoren produserer et lite dreiemoment. Figur 5.1 viser turttall-karakteristikk for tomgangstesten. Figur 5.2 viser utviklet dreiemoment for tomgangstesten.



(a) Python **(b)** Simulink

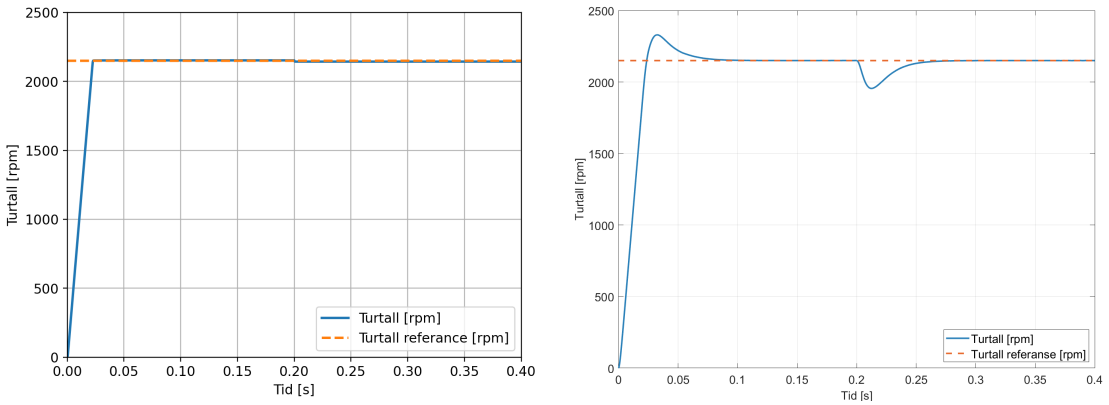
Figur 5.1: Test 1 - Tomgangstest. Figuren viser turtall og referanse når drivsystemet kjører uten ekstern mekanisk last, $\mathrm{T_{Last}} = 0$.
Referanseturtall = 2150 [rpm], $\mathrm{B_{Py}}$= 0 $[\frac{kgm^2}{s}]$, $\mathrm{B_{Sim}}$ = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Figur (a) viser respons i Python-modellen. Stigningstid = 0.0202 [s].
Figur (b) viser respons i Simulink-modellen. Stigningstid = 0.0208 [s].



(a) Python **(b)** Simulink

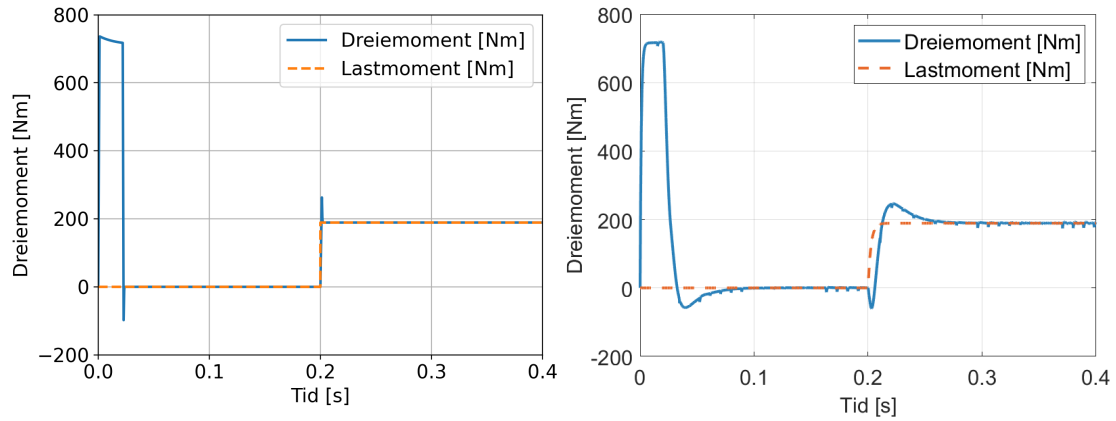
Figur 5.2: Test 1 - Tomgangstest. Figuren viser dreiemoment når drivsystemet kjører uten ekstern mekanisk last , $\mathrm{T_{Last}} = 0$. Referanseturtall = 2150 [rpm],
$\mathrm{B_{Py}}$= 0 $[\frac{kgm^2}{s}]$, $\mathrm{B_{Sim}}$ = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Figur (a) viser respons i Python-modellen. Toppunkt = 736 [Nm].
Figur (b) viser respons i Simulink-modellen. Toppunkt = 719 [Nm].

I testscenario 2 skjer det et steg i lastmomentet T_{Last} etter 0.2 sekunder. Ved oppstart er T_{Last} lik null og etter spranget er lastmomentet halvparten av det nominelle dreiemomentet til motoren, $T_{Last} = 189$. Se figur 5.4. Simulasjonene er utført for å vise hvordan drivsystemet reagerer ved transient og stabil tilstand etter lastendring. Det er av interesse å undersøke turtall og dreiemoment, samt strømmer som produserer dreiemomentet for å validere modellene.



(a) Python **(b)** Simulink

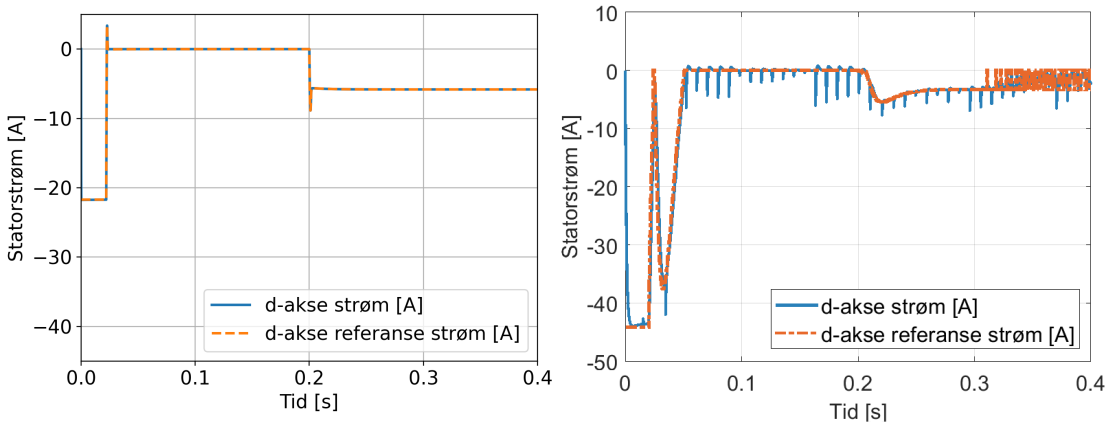
Figur 5.3: Test 2 - Sprang i last. Figuren viser turtall og referanse når drivsystemet blir utsatt for et sprang i lasten. Referanseturtall = 2150 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Sprang i T_{Last} går fra 0 [Nm] til 189 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) viser respons i Python-modellen. Stigningstid = 0.0202 [s].
Figur (b) viser respons i Simulink-modellen. Stigningstid = 0.0208 [s].



(a) Python **(b)** Simulink

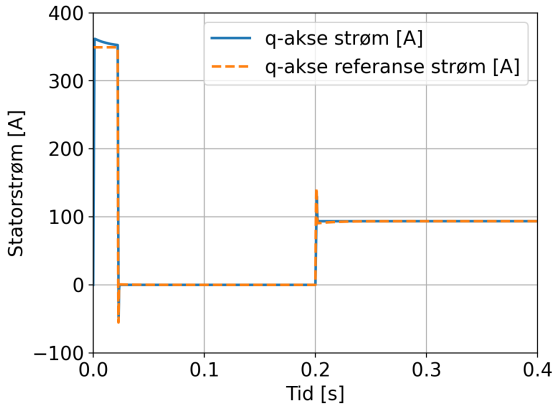
Figur 5.4: Test 2 - Sprang i last. Figuren viser dreiemoment og lastmoment når drivsystemet blir utsatt for et sprang i lasten. Referanseturtall = 2150 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Sprang i T_{Last} går fra 0 [Nm] til 189 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) viser respons i Python-modellen. Toppunkt = 736 [Nm].
Figur (b) viser respons i Simulink-modellen. Toppunkt = 719 [Nm]

Produsert dreiemoment i motoren er basert på statorstrømmer og magnetisk fluks. Figur 5.5 viser hvordan statorstrømmer i_d og i_q oppfører seg under oppstart, stabil tilstand og ved lastendring. Figuren viser aksestrømmer i_d og i_q i forhold til referanser bestemt av MTPA. Ved oppstart blir i_d negativ og i_q positiv for å produsere dreiemomentet som akselererer motoren opp til referanseturtall. Under stabil tilstand før lastendringen er i_d og i_q tilnærmet lik null ettersom motoren ikke krever dreiemoment. Ved lastendring må motoren utvikle dreiemoment for å motvirke lastmomentet. Dreiemomentet blir produsert av statorstrømmer i_d og i_q. Statorstrøm i_d er negativ for å produsere reluktans dreiemoment. Størrelsen på strømmene øker ved positivt sprang i lastmoment, og er konstante ved stabil tilstand.

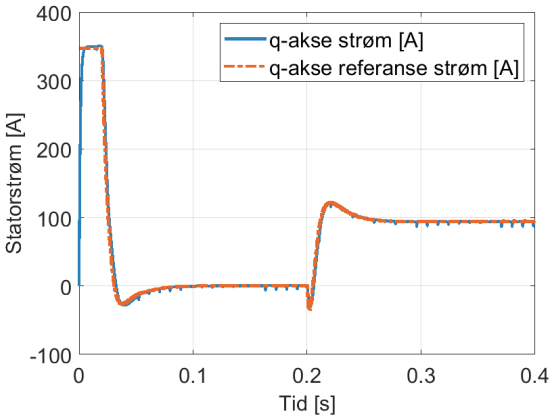


(a) i_d og $i_{d,ref}$ - Python

(b) i_d og $i_{d,ref}$ - Simulink



(c) i_q og $i_{q,ref}$ - Python

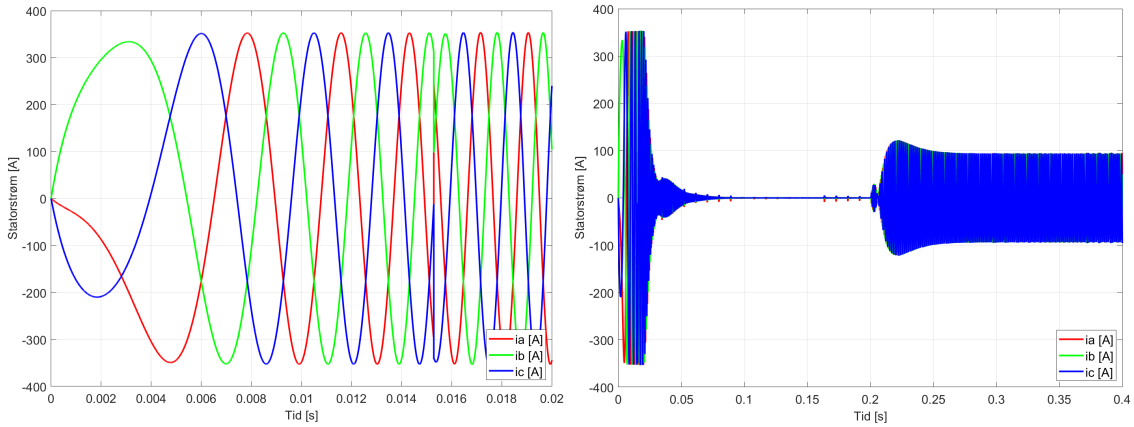


(d) i_q og $i_{q,ref}$ - Simulink

Figur 5.5: Test 2 - Sprang i last. Figuren viser statorstrømmer i_d og i_q med respektive referansestrømmer. Drivsystemet blir utsatt for et sprang i lasten. Referanseturtall = 2150 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2]. Sprang i T_{Last} går fra 0 [Nm] til 189 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) og (c) viser respons i Python-modellen.
Figur (b) og (d) viser respons i Simulink-modellen.

Figur 5.6 viser hvordan statorstrømmene i_{abc} oppfører seg ved oppstart, lastendring og stabil tilstand. Under oppstart er det tydelig at strømmene ikke er rent sinusformet. Det skjer mens motoren oppnår referansehastigheten. Før lastendring er amplituden til strømmene konstant tilnærmet lik null. Ved lastendring øker strømmene.

Når motoren opererer under stabil tilstand etter lastendringen blir strømmene sinusformet. Figuren samsvarer med figur 5.5.



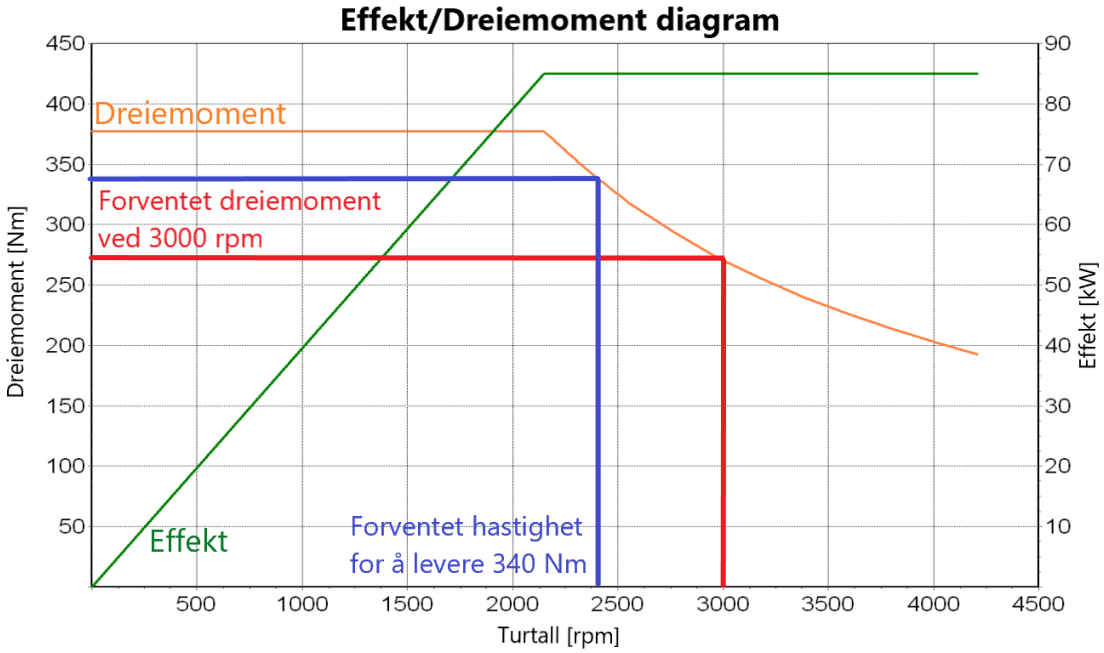
(a) Oppstart **(b)** Oppstart til stabil tilstand

Figur 5.6: Test 2 - Sprang i last. Figuren viser statorstrømmer i_a, i_b og i_c når drivsystemet blir utsatt for et sprang i lasten.
Figuren viser respons i Simulink-modellen.
Referanseturtall = 2150 [rpm], B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Sprang i T_{Last} går fra 0 [Nm] til 189 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) viser nærbilde av oppstart til i_{abc}. Amplitude = 353 [A].
Figur (b) viser i_{abc} fra oppstart til stabil tilstand. Amplitude stabil tilstand = 94[A].

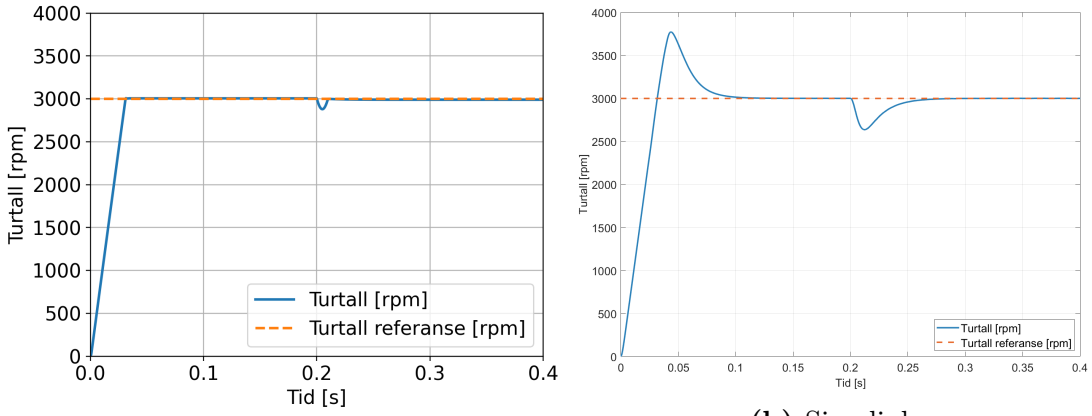
5.2 Stort sprang i last - høy hastighet

I testscenario 4 simulereres drivsystemet med en turtallrefereanse som ligger over nominell hastighet, Referanseturtall = 3000 [rpm]. Etter 0.2 sekund skjer det et sprang i lastmomentet med størrelse 0.9*T_{nom}, T_{Last} = 340.2 [Nm].
I følge databladet er det mer dreiemoment enn det maskinen klarer å levere med satt turtallreferanse. Figur 5.7 viser effekt/dreiemoment diagrammet fra vedlegg A med markerte grenser.

Test 4 er gjennomført for å vise hvordan drivsystemet reagere under forhold den ikke er designet for å takle. Det er av interesse å undersøke hvordan motindusert spenning påvirker drivsystemet ettersom Back-EMF er direkte relatert til hastighet. Figur 5.8, 5.9 og 5.10 viser simulasjoner fra test 4.

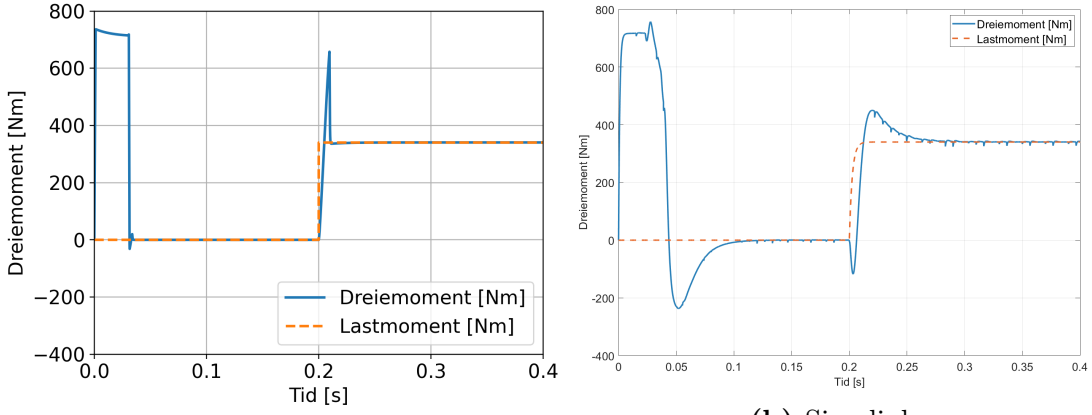


Figur 5.7: Effekt/dreiemoment diagram med grenser. Hentet fra datablad til Oswald MFS13.3-6W. Rød markering viser forventet levert dreiemoment ved 3000 [rpm]. Blå markering viser maks turtall maskinen kan oppnå ved T_L = 340 [Nm].



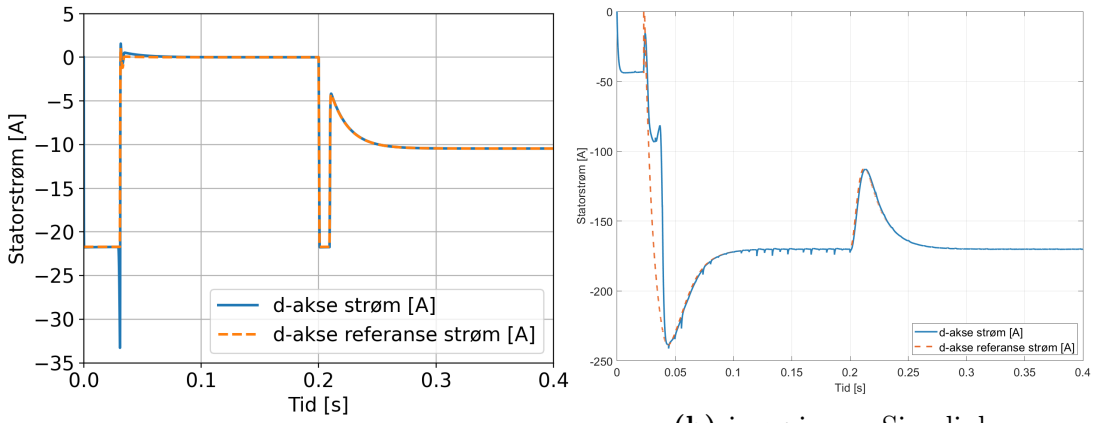
(a) Python **(b)** Simulink

Figur 5.8: Test 4 - Høy turtallreferanse med stort sprang i last.
Referanseturtall = 3000 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Sprang i T_{Last} går fra 0 [Nm] til 340.2 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) viser respons i Python-modellen. Stigningstid = 0.0280 [s].
Figur (b) viser respons i Simulink-modellen. Stigningstid = 0.0286 [s].



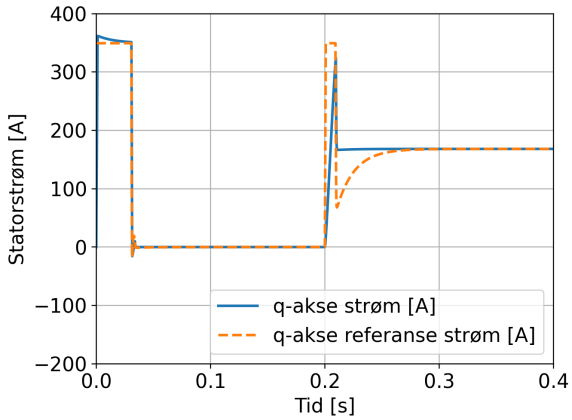
(a) Python **(b)** Simulink

Figur 5.9: Test 4 - Høy turtallreferanse med stort sprang i last.
Referanseturtall = 3000 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Sprang i T_{Last} går fra 0 [Nm] til 340.2 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) viser respons i Python-modellen. Toppunkt = 736 [Nm].
Figur (b) viser respons i Simulink-modellen. Toppunkt = 757 [Nm].

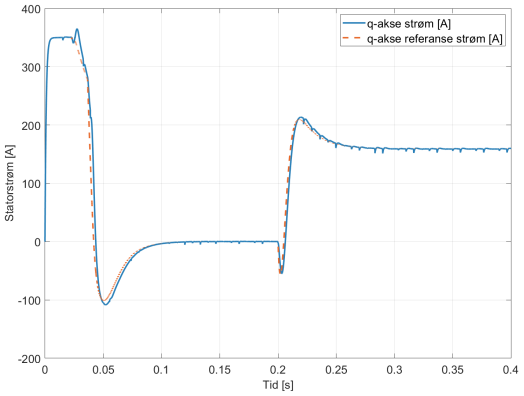


(a) i_d og $i_{d,ref}$ - Python

(b) i_d og $i_{d,ref}$ - Simulink



(c) i_q og $i_{q,ref}$ - Python



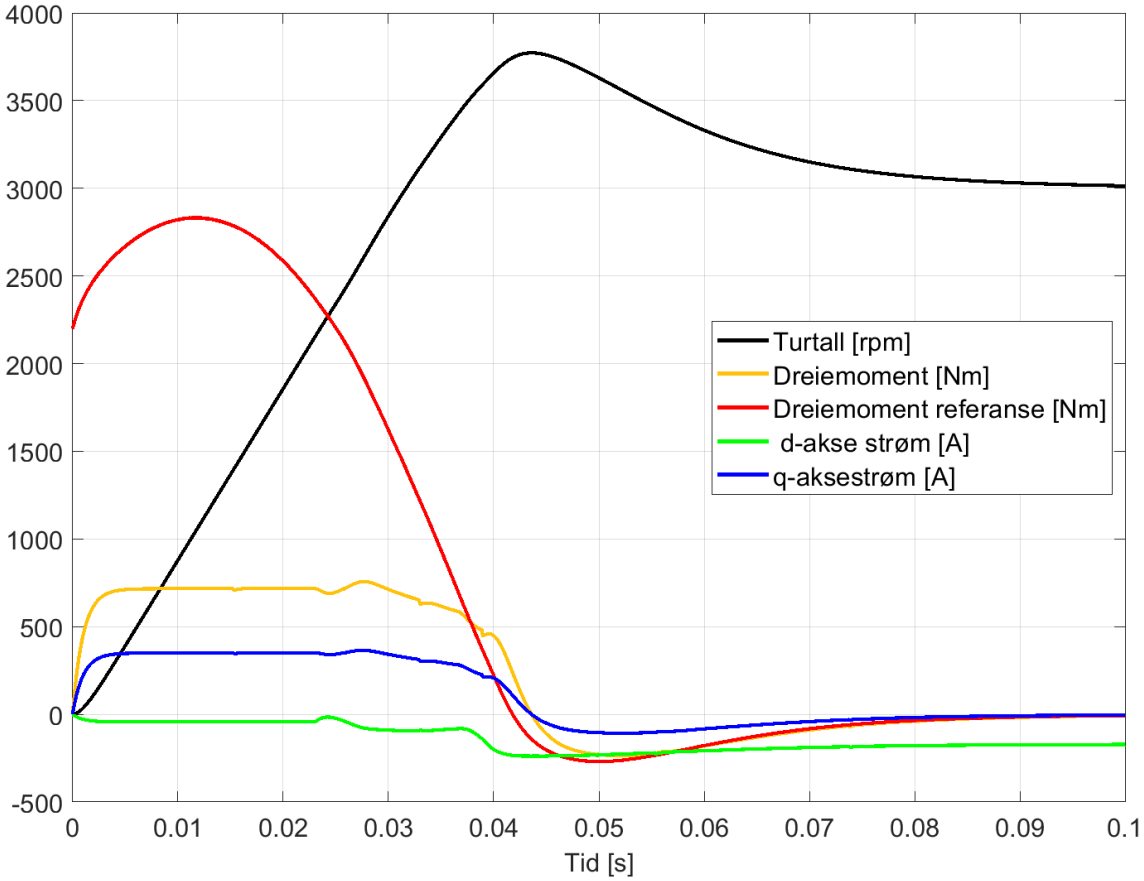
(d) i_q og $i_{q,ref}$ - Simulink

Figur 5.10: Test 4 - Høy turtallreferanse med stort sprang i last. Figuren viser statorstrømmer i_d og i_q med respektive referansestrømmer. Drivsystemet blir utsatt for et sprang i lasten.
Referanseturtall = 3000 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2].
Sprang i T_{Last} går fra 0 [Nm] til 340 [Nm]. Tidspunkt lastendring: 0.2 [s].
Figur (a) og (c) viser respons i Python-modellen.
Figur (b) og (d) viser respons i Simulink-modellen. Det er ulik størrelse på y-akser.

Figur 5.8 viser at drivsystemet klarer å driftes på 3000 [rpm] i begge modellene. Fra figur 5.9 leverer motoren dreiemoment for å akselerere under oppstart. Drivsystemet klarer å opprettholde referanseturtall under stabil tilstand når T_{Last} blir påført motoren. Ifølge effekt/dreiemoment diagrammet (5.7) fra databladet til motoren så vil ikke motoren kunne levere mer enn 220 Nm ved et turtall på 3000 rpm. Det ønskede resultatet er at motoren går ned til 2450 rpm, som er det største turtallet der den klarer å levere 340 Nm. Det kan tyde på at modellene ikke fungerer optimalt for turtallreferanse over nominelt turtall.

I transient tilstand opplever Simulink-modellen et stort oversving for turtall. Motoren går fra å levere positivt dreiemoment til negativt dreiemoment under oppstart. Motoren leverer også negativt dreiemoment i starten av lastendring. Det samme gjelder i test 2 for Simulink-modellen. I test 4 er utslaget større.

Figur 5.11 viser turtall og dreiemoment i samme graf som statorstrømmer i_d og i_q ved oppstart av drivsystemet i Simulink-modellen. Den viser også dreiemoment referansen, $T_{e,ref}$ satt av hastighetskontrollen. MTPA-utregning bruker $T_{e,ref}$ for å beregne referansestrømmer. Dreiemoment-referansen er større enn produsert dreiemoment, ettersom hastighetskontroll har uendelig metning for $T_{e,ref}$.
Når turtallet overstiger 2570 [rpm] starter i_q å synke. Det kan komme av at motindusert spenning blir større enn maksimum utgangsspenning drivsystemet kan levere. Simulink-modellen bruker fluksvekking til å beregne referansestrømmer når turtallet overstiger basehastigheten ω_b. Referansen for i_q synker for å svekke luftgap-fluks slik at motorhastighet kan øke. Konsekvensen av fluksvekking er at dreiemomentet synker. Se kapitel 2.3.5 - Flukssvekking. Under transient tilstand etter sprang i last øker dreiemomentet



Figur 5.11: Test 4 - Høy turtallreferanse med stort sprang i last. Figuren viser turtall, T_e, $T_{e,ref}$, i_d og i_q samme graf ved oppstart for Simulink-modellen. Y-akse representere størrelsen for verdiene. Referanseturtall = 3000 [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, $B_{Sim} = 0.0012$ $[\frac{kgm^2}{s}]$,
J = 0.07 [kgm^2]. Sprang i T_{Last} går fra 0 [Nm] til 340 [Nm]. Tidspunkt lastendring: 0.2 [s].

5.3 Kontrollrespons

For å analysere respons til kontrollsystemet presenteres stigningstid, innsvingningstid, oversving/undersving og stasjonært avvik for Python-modellen og Simulink-modellen. Stiginingstid er definert som tiden det tar før kontrollert verdi oppnår 90% av utgangsverdien. Innsvingningstid er definert som tiden det tar før kontrollert verdi ligger innenfor et spesifisert feilbånd. Oversving og undersving er definert som differansen mellom transient tilstand toppunkt/bunnpunkt og stabil tilstand verdi. Stasjonært avvik er definert som differansen mellom stabil tilstandverdi og referanseverdi. Spesifisert feilbånd for innsvingningstid er satt til $\pm 2\%$ av stasjonær tilstandverdi.

Tabell 10: Test 1 - Kontrollrespons

	Simulink	Python	Enhet
Stigningstid	0.0208	0.0203	s
Innsvingningstid (2%)	0.06	0	s
Oversving	8.37	0.23	%
Stasjonært avvik	0.05	0.14	%

Tabell 11: Test 2 - Kontrollrespons

Før sprang i last

	Simulink	Python	Enhet
Stigningstid	0.0208	0.0202	s
Innsvingningstid (2%)	0.06	0	s
Oversving	8.37	0.23	%
Stasjonært avvik	0.05	0.14	%

Etter sprang i last

	Simulink	Python	Enhet
Innsvingningstid (2%)	0.04	0	s
Undersving	7.21	0.51	%
Stasjonært avvik	0.03	0.28	%

Tabell 12: Test 4 - Kontrollrespons

Før sprang i last

	Simulink	Python	Enhet
Stigningstid	0.0286	0.0280	s
Innsvingningstid (2%)	0.08	0	s
Oversving	25.67	0.27	%
Stasjonært avvik	0.01	0.17	%

Etter sprang i last

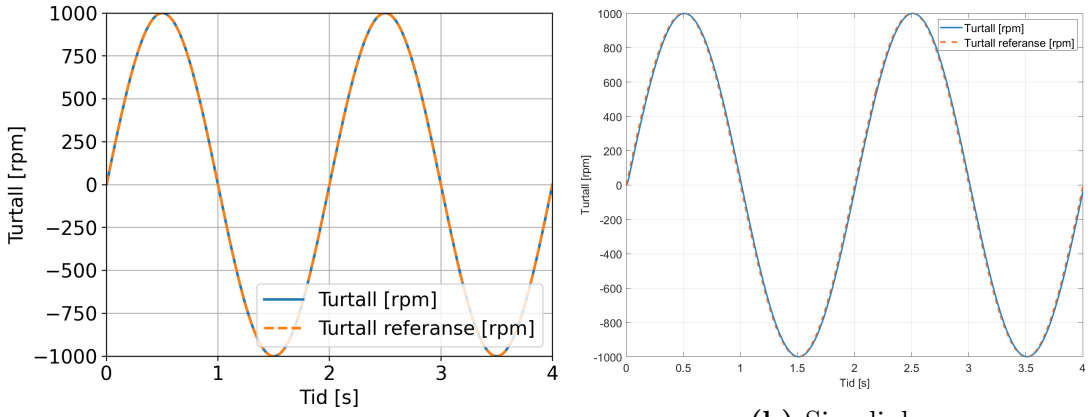
	Simulink	Python	Enhet
Innsvingningstid (2%)	0.045	0	s
Undersving	12.13	4.03	%
Stasjonært avvik	0.03	0.37	%

Stigningstid er tilnærmet lik i Simulink-modellen og Python-modellen. Simulink har merkbart større oversving/undersving og innsvingingstid. Stasjonært avvik er mindre i Simulink-modellen. Kontrollresponsen er gjentakende for test en, to og fire. Tabellene viser at Python-modellen har bedre kontrollrespons enn Simulink-modellen, ettersom oversving/undersving ikke er tilfredstillende. Simulink-modellen har større oversving i test 4 enn i test 2. I Python-modellen er oversvinget tilnærmet likt.

Forskjell i kontrollrespons kommer av ulikheter mellom regulatorer i Simulink-modellen, se kap. 4.1.1, og Python-modellen, se kap.4.2.1.

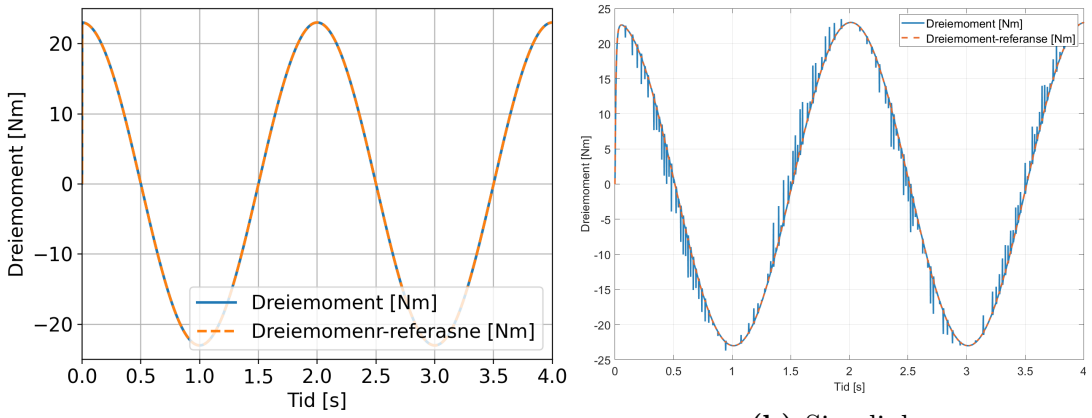
5.4 Varierende turtallreferanse

I testscenario blir drivsystemet simulert med varierende turtallreferanse. Referanseturtall = $1000sin(\pi t)$. Referansen er sinusformet. I test 5 blir drivsystemet simulert med $T_{Last} = 0$ [Nm]. Test 5 er gjennomført for å vise at drivsystemet takler å driftes med varierende turtallreferanse. Figur 5.12 viser hvordan turtall responderer med sinusreferanse. Figur 5.13 viser utviklet dreiemoment



(a) Python **(b)** Simulink

Figur 5.12: Test 5 - Varierende turtallreferanse. Figuren viser turtall og referanse når drivsystemet følger sinusformet turtallreferanse.
Referanseturtall = 1000sin(πt) [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2]. $T_{Last} = 0$ [Nm]
Figur (a) viser respons i Python-modellen. Figur (b) viser respons i Simulink-modellen.

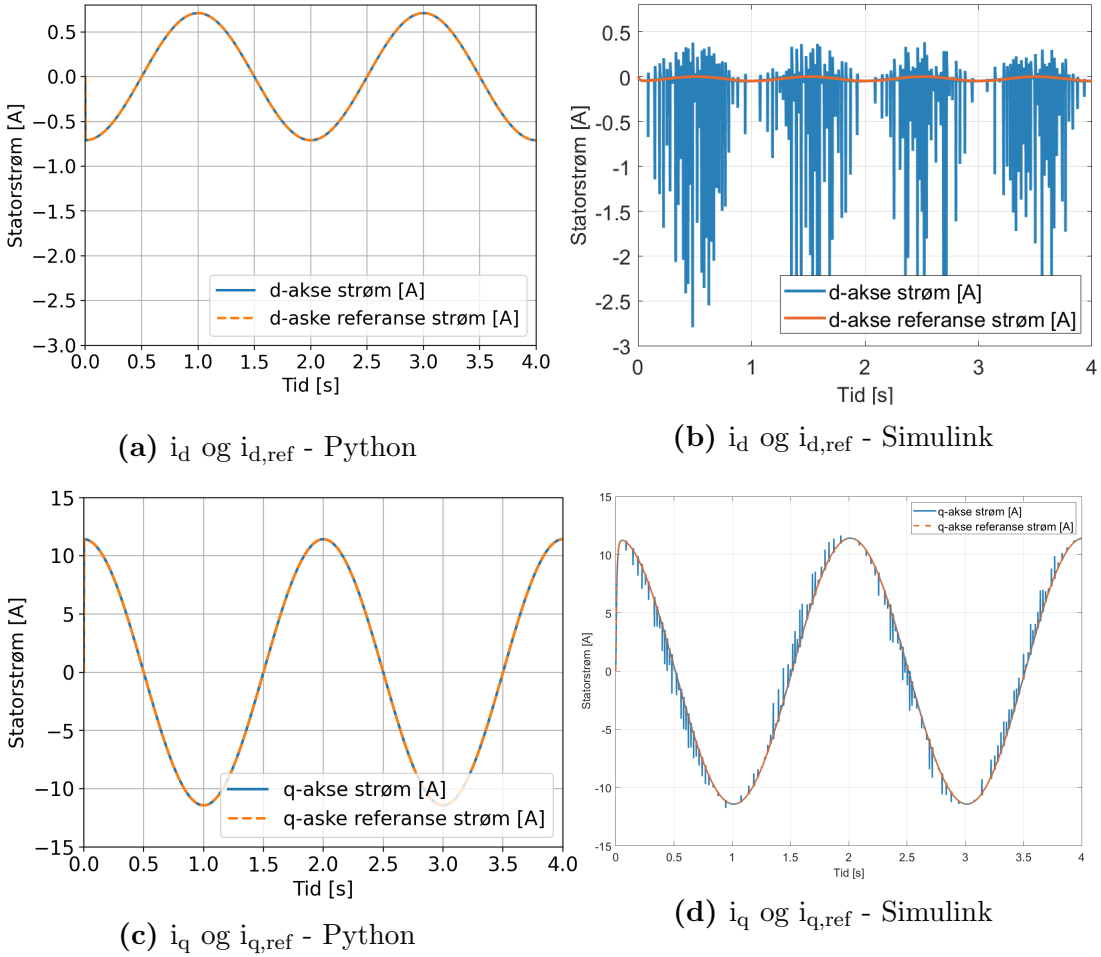


(a) Python **(b)** Simulink

Figur 5.13: Test 5 - Varierende turtallreferanse. Figuren viser turtall og referanse når drivsystemet følger sinusformet turtallreferanse.
Referanseturtall = 1000sin(πt) [rpm], B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2]. $T_{Last} = 0$ [Nm]
Figur (a) viser respons i Python-modellen. Figur (b) viser respons i Simulink-modellen.

Figur 5.12 viser hvordan turtall responderer med sinusreferanse. Begge modellene følger referansen tilfredstillende. Figur 5.13 viser utviklet dreiemoment og dreiemomentreferansen med sinusformet turtall. Drivmoment blir cosinusformet når turtallet er sinusformet. Det er ønsket oppførsel ut ifra ligning (2.10) for mekanisk dreiemoment.

Figur 5.14 viser hvordan i_d og i_q oppfører seg i forhold til referansestrømmer med sinusformet turtallreferanse. Figur 5.14 (b), viser at statorstrøm i_d i Simulink-modellen skiller seg fra Python-modellen. Det kommer av at MTPA-blokken i Simulink beregner en mindre referanse for i_d. Det kan tyde på at taggene for i_d kommer av at Simulink-modellen bruker Clarke/Park transformasjon. Figur 5.15 viser at hver gang elektrisk rotorposisjon Θ_e går fra π [rad] til -π [rad] får sker det et fall i i_d. Det gjelder også for i_q, men er mer tydelig i figur 5.15. Ettersom statorstrøm i_d og i_q brukes i beregning for T_e vil dreiemoment også bli hakkete. Oppførselen for i_d og i_q er ikke et spesialtilfelle for testscenario 5. Det gjelder generelt i drivsystemet.



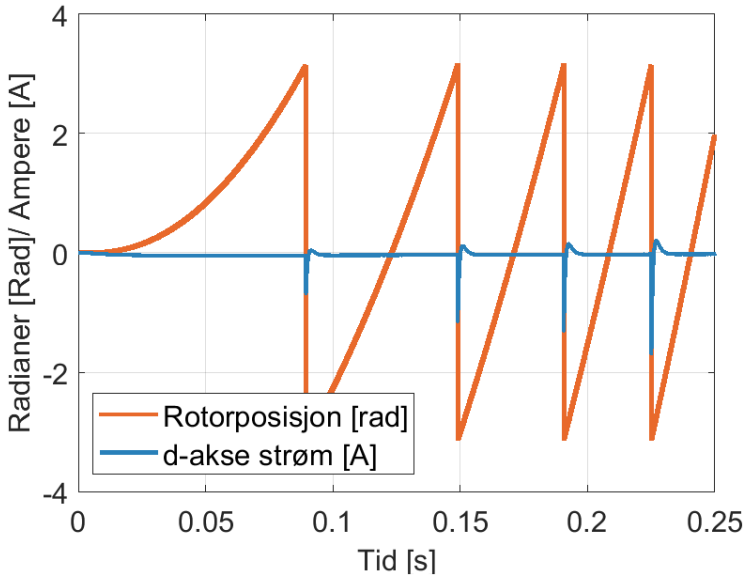
(a) i_d og $i_{d,ref}$ - Python

(b) i_d og $i_{d,ref}$ - Simulink

(c) i_q og $i_{q,ref}$ - Python

(d) i_q og $i_{q,ref}$ - Simulink

Figur 5.14: Test 5 - Varierende turtallreferanse. Figuren viser statorstrømmer i_d og i_q med respektive referanser. Drivsystemet følger sinusformet turtallreferanse. Referanseturtall = 1000sin(πt) [rpm], B_{Py}= 0 [$\frac{kgm^2}{s}$], B_{Sim} = 0.0012 [$\frac{kgm^2}{s}$], J = 0.07 [kgm^2], T_{Last} = 0 [Nm]
Figur (a) og (c) viser respons i Python-modellen. Figur (b) og (d) viser respons i Simulink-modellen.

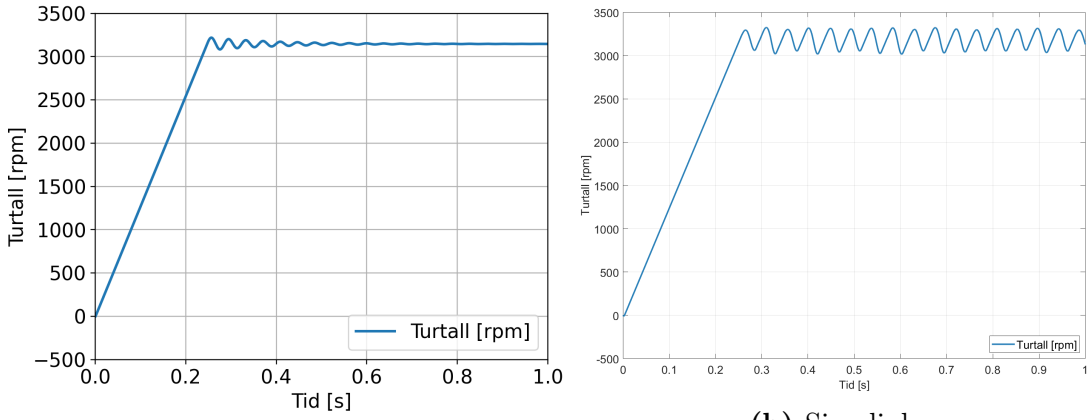


Figur 5.15: Test 5 - Varierende turtallreferanse. Figuren viser rotorposisjon Θ_e og statorstrøm i_d når drivsystemet følger sinusformet turtallreferanse.
Referanseturtall = 1000sin(πt) [rpm], B_{Sim} = 0.0012 [$\frac{kgm^2}{s}$], J = 0.07 [kgm^2], T_{Last} = 0 [Nm]. Figuren viser respons i Simulink-modellen

5.5 Drivsystem uten hastighetskontroll

I testscenario 3 blir drivsystemet simulert uten hastighetskontroll. Modellen frakobler hastighetsreferanse, hastighetsregulator og MTPA beregning. I stedet for å bruke hastighetreferans som settpunkt bruker drivsystemet konstante verdier for i_d og i_q. Referansestrømmer er satt til: i_d = -5 [A] og i_q = 140 [A]. I test 3 blir drivsystemet simulert med konstant lastmoment, T_{Last} = 189 [Nm].

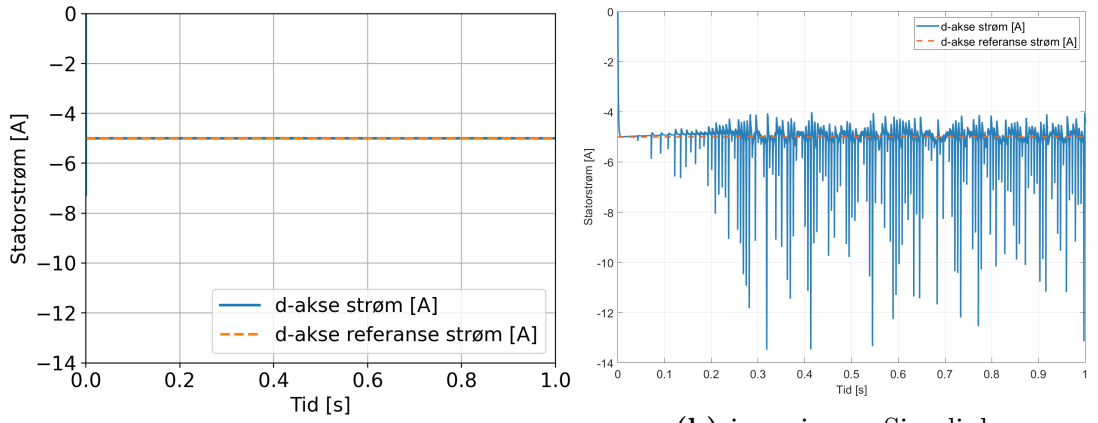
Test 3 er gjennomført for å undersøke motorrespons når det kun er strømmer som setter referanse for drivsystemet. Det kan være nyttig å se hvor godt strømregulatorene klarer å følge referanse, og hvordan turtall vil oppføre seg.



(a) Python **(b)** Simulink

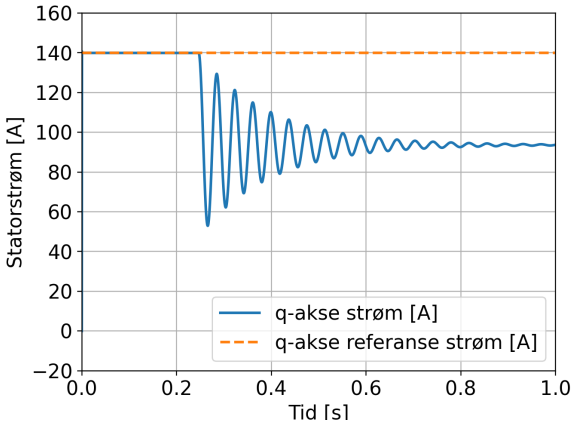
Figur 5.16: Test 3 - drivsystem uten hastighetskontroll. Figuren viser turtall når drivsystemet ikke har hastighetskontroll. $i_{d,ref}$ = -5 [A], $i_{q,ref}$ = 140 [A]
B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2], T_{Last} = 189 [Nm].
Figur (a) viser respons i Python-modellen. Figur (b) viser respons i Simulink-modellen.

Figur 5.17 viser hvordan aksestrømmer i_d og i_q følger satte referanseverdier

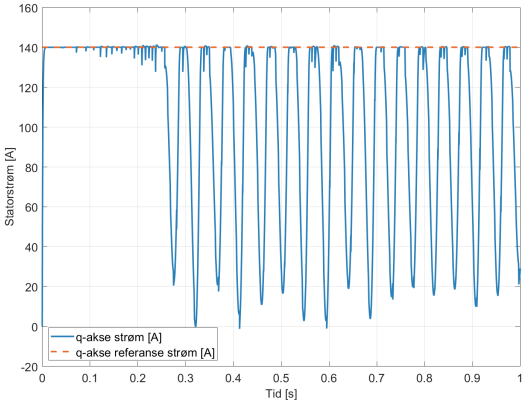


(a) i_d og $i_{d,ref}$ - Python

(b) i_d og $i_{d,ref}$ - Simulink

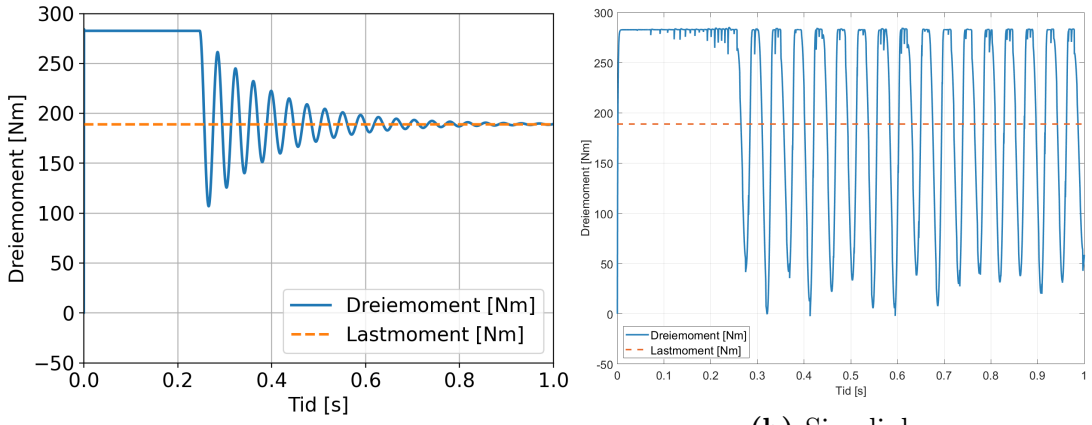


(c) i_q og $i_{q,ref}$ - Python



(d) i_q og $i_{q,ref}$ - Simulink

Figur 5.17: Test 3 - drivsystem uten hastighetskontroll. Figuren viser statorstrømmer i_d og i_q når drivsystemet ikke har hastighetskontroll. $i_{d,ref}$ = -5 [A], $i_{q,ref}$ = 140 [A] B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2], T_{Last} = 189 [Nm]. Figur (a) og (c) viser respons i Python-modellen. Figur (b) og (d) viser respons i Simulink-modellen.



(a) Python

(b) Simulink

Figur 5.18: Test 3 - Drivsystem uten hastighetskontroll. Figuren viser dreiemoment og lastmoment når drivsystemet opererer uten hastighetskontroll. Referanser for systemet er: $i_{d,ref}$ = -5 [A] og $i_{q,ref}$ = 140 [A].
B_{Py}= 0 $[\frac{kgm^2}{s}]$, B_{Sim} = 0.0012 $[\frac{kgm^2}{s}]$, J = 0.07 [kgm^2], T_{Last} = 189 [Nm].
Figur (a) viser respons i Python-modellen. Figur (b) viser respons i Simulink-modellen.

Figur 5.16 viser turtall uten hastighetskontroll. Turtall øker lineært opp til 3300 [rpm] i begge modellene. Etter stigningen starter turtall å oscillerer fra 3060 [rpm] til 3300 [rpm]. I Python-modellen avtar oscillasjonene. I Simulink-modellen avtar ikke oscillasjonene. Hastigheten er bestemt utifra dreiemoment som er beregnet med statorstrømmer i_d og i_q. Figur 5.17 viser at oscillasjonene kommer fra oscillasjoner i statorstrømmer. Strømregulatoren regulere ikke strømmene tilfredstillende etter stigningen avtar. Responsen i Simulink er ustabil. Test 3 forsterker inntrykket av at kontrollsystemet i Python-modellen er bedre innstilt.

Figur 5.18 viser at dreiemoment varier med oscillasjonene til statorstrømmene. Det er et stasjonært avvik for i_q i Python-modellen. I Python-modellen vil motoren levere nok dreiemoment for å følge lastmomentet.

6 Drøfting

Dette kapittelet diskuterer modellene som er utviklet. Forskjeller mellom modellene i Matlab/Simulink og Python blir presentert. I tilegg blir det trukket frem svakheter og forbedringspotensial ved modellene. Forslag for framtidig arbeid blir presentert i slutten av kapittelet.

6.1 Forskjeller mellom Matlab/Simulink og Python-modellen

Simulink er et grafisk programmeringsmiljø, som baserer seg på visuell modellering med et bibliotek av ulike blokker. Python er et objektorientert programmeringsspråk med et stort bibliotek av utvidelser og funksjoner med åpen kildekode.

Hovedforskjellene mellom modellene ligger i utførelse av kontrollsystemet. Se kapittel 4. Hastighet- og strømregulatorer er designet forskjellig, der regulatorene i Python er konstruert enklere og bruker en annen metode for å bestemme regulatorparametere. Dette er tydelig i alle testene som er gjennomført. Simulink-modellen har dårligere kontrollrespons.

Simulink bruker FOC som overordnet kontrollstruktur. Det utføres Clarke og Park transformasjon på tilbakekoblet statorstrømmer, og inverstransformasjon før inngangspenning blir påført motoren. Inne i motormodellen skjer Clarke og Park transformasjon en gang til, for å beregne utgangsparameterene for motoren. Statorstrømmene i_d og i_q blir så inverstransformert til i_{abc} før tilbakekoblingen til strømkontrollsystemet. I en modell uten vekselretter er Clarke og Park transformasjon en unødvendig operasjon. Det er implementert for å demonstrere hvordan strømmer oppfører seg i abc-referanserammen, samt gi et bedre bilde over hvordan feltorientert kontroll fungerer. Referansestrømmer blir beregnet ved å bruke den innebygde blokken " MTPA reference", i Simulink biblioteket. MTPA-blokken bruker også fluksvekking til å bestemme referansestrømmene.

Python-modellen bruker også FOC som overordnet kontrollstruktur, men det blir hverken gjort Park eller Clarke transformasjoner i modellen. Siden modellen alltid befinner seg i dq-referanserammen. Det er ingen innebygd MTPA funksjon i Python, så MTPA kalkulasjoner har blitt laget i Python-modellen. Løsningen for MTPA er forklart i kapittel 4.2.1.

Regulatorer i Simulink er basert på teori presentert i kapittel 2.3. Regulatorparametre bestemmes ved hjelp av intern modell kontroll, og beregnes ved å bruke motorparametre og båndbredde. Både hastighetsregulator og strømregulator bruker aktiv demping og anti-windup teknikk for å forbedre kontrollrespons. Intern modell kontroll er valgt for å oppnå et universelt kontrollsystem.

Regulatorene i Python baserer seg også på teorien presentert i kapittel 2.3, men det har blitt valgt en annen metode for å finne regultorparameterene. Det ble først prøvd å bruke parameterene funnet for Simulink-modellen, men verdiene førte til at reguleringssystemet ble ustabilt. For å finne passende paremetere for regulatorsystmet ble "prøve og feile" metoden brukt. Forskjellen i måten for å finne regulator-

parametere har ført til at Python-modellen har bedre kontrollrespons, men kun for motoren som er valgt for oppgaven. Hvis det skal simuleres en annen motor så vil det være nødvendig å manuelt finne de nye kontroll parametrene. Python-modellen er derfor mindre universell enn Simulink-modellen.

For modellen av permanentmagnet maskinen er det ingen store forskjeller. Begge modellene er basert på den samme matematiske modellen for PMSM som er presentert i kapittel 2.2.1. Foruten programmeringsstruktur, ligger den største forskjellen i metoden som løser differensialligningene for aksestrømmene i_d og i_q. Simulink bruker en integratorblokk som kontinuerlig integrerer inngangsvariabelen, mens Python bruker Eulers metode for å løse differanselikninger. Løsningene fra Eulers metode kan ha et lite avvik fra den faktiske løsningen for differanselikningen, men testing har vist at det ikke er et merkbart problem for modellen.

6.2 Svakheter ved modellene og forbedringer

Modellene er basert på en rekke forenklinger, se kapittel 4.3. De er gjort for å forenkle designprosessen, og for å holde tidsskjema til oppgaven.

En svakhet med forenklingene er at modellen ikke har implementert vekselretter, samt en metode for å bestemme pulsbreddemodulasjon til vekselretteren. Vekselretter-systemet er en nødvendighet for å drifte PMSM drivsystemet med DC-spenningskilde, siden det realiserer DC-AC omforming. En konsekvens av forenklingen er at drivsystemet vil oppføre seg mer ideelt, ettersom inngangspenning på motoren er rent sinusformet.

MTPA kalkulasjonene i Python-modellen støtter ikke permanentmagnet maskiner uten reluktansbidrag i dreiemomentet, som gjør modellen mindre universell. Python-modellen bruker ikke fluksvekking, se kapittel 2.3.5, for å beregne referansestrømmer. De er kun basert på MTPA kalkulasjon. Modellen vil derfor kun oppføre seg normalt ved hastigheter opp til punktet hvor motindusert spenning blir større enn maksimal utgangspenning for drivsystemet.

Regulatorene i Python-modellen er ikke like avanserte som i Simulink-modellen. Anti-windup og aktiv demping er ikke implementert, noe som fører til at drivsystemet har større tendenser til å bli ustabilt. Når hastighetsreferansen blir satt til en verdi større enn ca. 3000 rpm, vil systemet bli ustabilt. Det fører til at prosessverdiene øker opp mot uendelig. Problemet kunne blitt løst med anti-windup og aktiv demping. Anti-windup og aktiv demping ble ikke implementert i Python-modellen på grunn av dårlig tid, som gjorde at andre mer avgjørende oppgaver ble prioritert.

Arbeidsgiveren HPS ønsker en modell som kan simuleres i sanntid. Det har ikke blitt gjort et forsøk på å utvikle en modell som støtter sanntidssimulering, siden gruppen mangler kunnskap innen fagfeltet. Det ble gjort et studie i hvordan sanntidssimulering kunne oppnås, og gruppen kom til en konklusjon. Det ville bli for mye arbeid, og oppgaven ville blitt mindre gjennomførbar.

Det har ikke blitt gjort testing på simuleringshastigheten eller effektiviteten til Python-modellen. Gruppen mangler kunnskap til å teste og forbedere effektiviteten til modellen på en god måte. Siden gruppen mangler erfaring innen fagområdet, så

er det ukjent om programvaren holder opp til bransjens krav om effektivitet. Det er også ukjent om programmet er effektivt nok til å kunne kjøres i sanntid.

6.3 Framtidig arbeid

Det er fortsatt videreutviklingspotensiale for både Python- og Simulink-modellen. Flere av ønskene fra arbeidsgiver har ikke blitt implementert, og det har blitt gjort en rekke forenklinger som kan erstattes med mer fullførte løsninger. Det er mer fremtidig arbeid med Python-modellen siden det er den enkleste modellen av systemet. Det er også den som er mest relevant for oppdragsgiver.

For videreutvikling av Simulink-modellen bør vekselretter og SVPWM implementeres i modellen. Dermed kan tap og effektivitet beregnes. Kontrollresponsen i Simulink er ikke optimal. Regulatorer burde forbedres.

Det blir brukt en simpel modell for lastmomentet som virker på rotor. Videre arbeid vil være å utvikle en modell som imiterer mer realistiske forhold for lastmoment.

Den nåværende MTPA-kalkulasjonen i Python-modellen fungerer kun for permanentmagnet maskiner med utpregete poler. Videre arbeid vil være å finne en løsning som kan finne i_d og i_q referanser for maskiner med og uten utpregete poler.

6.3.1 Framtidig arbeid for Python-modellen

Flere områder for videreutvikling er kun gjeldene for Python-modellen. Grunnet den er enklere og at funksjoner kun er relevant for Python-modellen.

Regulatorene i Python benytter ikke anti-windup eller aktiv demping. Det kan implementeres i framtidig arbeid. Flukssvekking er ikke implementert i Python-modellen. Framtidig arbeid er å finne og implementere en algoritme som beregner referansestrømmer i flukssvekking-området.

For å få mest nytte av modellen ønsket HPS at Python-modellen skulle kunne kjøre i et sanntidsoperativsystem. Det er ikke blitt utforsket om modellen støtter sanntidssimulering. Videre arbeid vil være å finne ut om modellen støtter sanntidssimulering, og finne ut hvilke endringer som må til. Viderer testing og forbedring av simuleringshastigheten og effektiviteten til programmet er nødvendig.

7 Konklusjon

Prosjektgruppen har laget en forenklet modell av drivsystemet til en permanentmagnet synkronmaskin. Modellen kan brukes til å simulere hastighet, dreiemoment og statorstrømmer. Drivsystemet inneholder en hastighetsregulator og to strømregulatorer for kontroll. Til beregning av referansestrømmer bruker modellen maksimum dreiemoment per ampere. Det er mulig å studere kontrollrespons for drivsystemet. Modellen av drivsystemet er utviklet i programmeringsspråket Python. Simulink-modellen er utviklet for verifisering.

Den digitale modellen inneholder ikke vekselretter eller metode for å drifte motoren med hastigheter som gir høy motindusert spenning. Det er foreslått som framtidig arbeid. Motormodellen og regulatorer er gjeldene for begge typer permanentmagnet synkronmaskin. Utregning av referansestrømmer er kun gjeldene for maskiner som har utpreget poler. Det kan videreutvikles.

Det gjenstår fortsatt arbeid med den digitale modellen av drivsystemet, før den kan implementeres i en digital tvilling. Den nåværende modellen er godt dokumentert, og forlag til videre arbeid er presentert.

Kilder

[1] *Bruke Zotero - Wiki - innsida.ntnu.no*. nb-NO. URL: https://innsida.ntnu.no/wiki/-/wiki/Norsk/Bruke+Zotero (visited on 26th Apr. 2021).

[2] *Clarke and Park Transforms*. en. URL: https://se.mathworks.com/solutions/power-electronics-control/clarke-and-park-transforms.html (visited on 9th May 2021).

[3] *Documentation:Finite Element Method Magnetics*. URL: https://www.femm.info/wiki/Documentation/ (visited on 27th Apr. 2021).

[4] *Field-Oriented Control (FOC) - MATLAB & Simulink - MathWorks Nordic*. URL: https://se.mathworks.com/help/mcb/gs/implement-motor-speed-control-by-using-field-oriented-control-foc.html (visited on 10th May 2021).

[5] *Get started — PyCharm*. en-US. URL: https://www.jetbrains.com/help/pycharm/2021.1/quick-start-guide.html (visited on 26th Apr. 2021).

[6] Wang Han. 'Simulation model development of electric motor and controller'. en. In: *undefined* (2017). URL: /paper/Simulation-model-development-of-electric-motor-and-Han/3ac035f9868961eac064b2e2f1a3fd0a940ee4d4 (visited on 5th May 2021).

[7] 'Kunnskapsgrunnlag for lavutslippsutvikling'. no. In: (), p. 346.

[8] Muyang Li. 'Flux-Weakening Control for Permanent-Magnet Synchronous Motors Based on Z-Source Inverters'. en. In: (), p. 111. (Visited on 7th May 2021).

[9] *MATLAB - MathWorks*. en. URL: https://se.mathworks.com/products/matlab.html (visited on 26th Apr. 2021).

[10] M. S. Merzoug and F. Naceri. *Comparison of Field-Oriented Control and Direct Torque Control for Permanent Magnet Synchronous Motor (PMSM)*.

[11] David Vindel Muñoz. 'Design, Simulation and Implementation of a PMSM Drive System'. en. In: (), p. 87.

[12] *Overleaf - Wiki - innsida.ntnu.no*. nb-NO. URL: https://innsida.ntnu.no/wiki/-/wiki/English/Overleaf (visited on 26th Apr. 2021).

[13] 'Park, Inverse Park and Clarke, Inverse Clarke Transformations MSS Software Implementations User Guide'. en. In: (), p. 20.

[14] *Permanent Magnet Synchronous Motor*. URL: https://en.engineering-solutions.ru/motorcontrol/pmsm/ (visited on 10th May 2021).

[15] P. Pillay and R. Krishnan. 'Modeling of permanent magnet motor drives'. In: *IEEE Transactions on Industrial Electronics* 35.4 (1988), pp. 537–541. DOI: 10.1109/41.9176.

[16] *Simulink - Simulation and Model-Based Design*. en. URL: https://se.mathworks.com/products/simulink.html (visited on 26th Apr. 2021).

[17] *Transport står for 30 prosent av klimautslippene i Norge*. no. URL: https://www.ssb.no/natur-og-miljo/artikler-og-publikasjoner/transport-star-for-30-prosent-av-klimautslippene-i-norge (visited on 21st Jan. 2021).

[18] Oskar Wallmark. 'Control of permanent-magnet synchronous machines in automotive applications'. en. In: (), p. 96.

[19] *Welcome to Python.org*. en. URL: https://www.python.org/about/ (visited on 26th Apr. 2021).

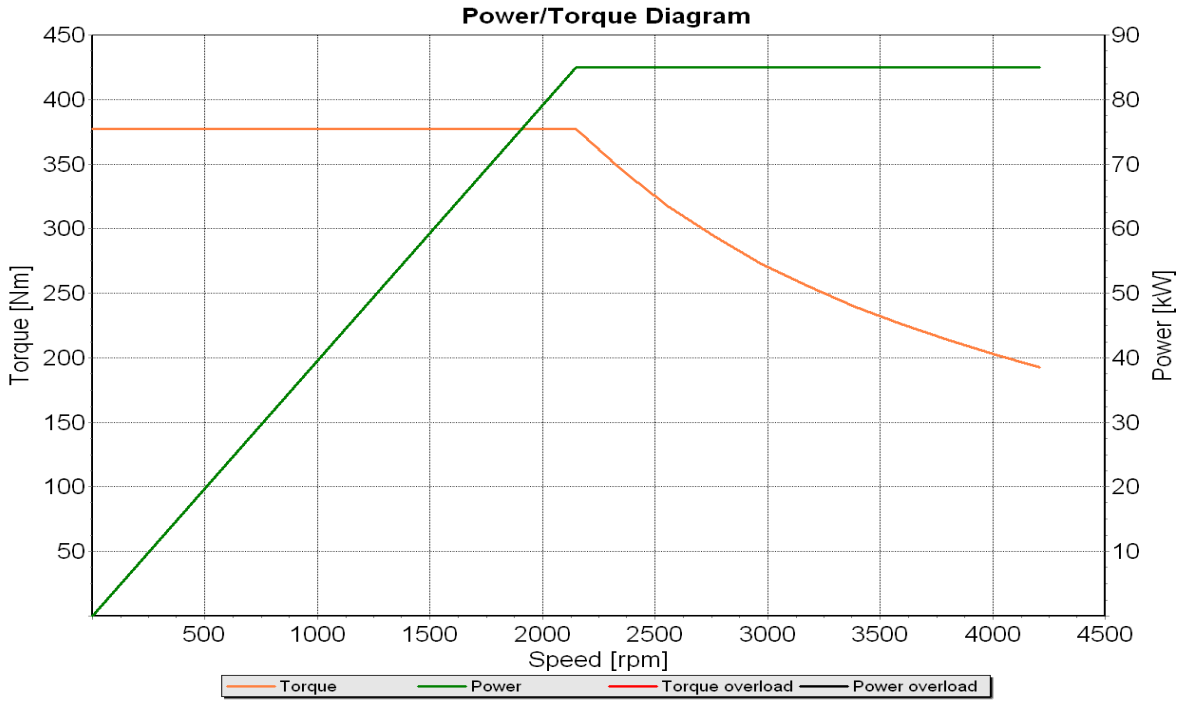
Vedlegg

A Datablad Oswald MFS13.3-6W

TECHNICAL DATA SHEET
AC SYNCHRONOUS GENERATOR

OSWALD
REGELBARE ELEKTROMOTOREN

Part No.	0481-000001a
Type	MFS13.3-6W



Absorbing Data			Rated				Overload			
			$P_{red.}$	P	Fw $P_{con.}$	Fw $P_{red.}$	$P_{red.}$	P	Fw $P_{con.}$	Fw $P_{red.}$
Voltage	U	[V]		390	400					
Power mech.	P	[kW]		85	85					
Current	I	[A]		140						
Frequency	f	[Hz]		107,5	200,0					
Speed	n	[min^{-1}]		2150,0	4209,0					
Torque	T	[Nm]		378	193					
Power factor	cos φ	[-]		0,88						
Efficiency	η	[%]		96,5						

Poles	6	
Connection	Star	
Duty	S1	
Allowed inverter frequency ≥	4	kHz
Allowed pulse rise time ≥	0,3	µs

Equivalent circuit (each phase to phase values)		
R1 u-v	66,0	mΩ
Lges u-v	1,90	mH
U_0 u-v	169	V/1000rpm

Resistance R at room temperature (20°C/68°F).
Conversion to operation temp.: $R_{hot} = R_{20°C/68°F}$ x 1,3

Protection:

Temperature sensors

3 x PT100 (Winding)	Protection IP	54	
2 x PT100 (Bearing)	Insulation class	F	
	Winding insul. max.	1,56	kV peak

TECHNICAL DATA SHEET
AC SYNCHRONOUS GENERATOR

OSWALD
REGELBARE ELEKTROMOTOREN

Part No.	0481-000001a
Type	MFS13.3-6W

Cooling:	IC	71W water cooled
	1	Material: copper, Medium: water, Inlet temperature: 35 °C, Flow rate: 8 l/min, dp: ca. 1,0 bar, max. pressure: 8 bar, Mounting position: axial NDE,
	2	
	3	

Mechanical Design:				gr	hrs
Bearing DE	Spindle bearing, current insulated	**Regreasing DE**	ohne / without		
Bearing NDE	Spindle bearing, current insulated	**Regreasing NDE**	ohne / without		
Shaft DE	Ø 48k6 x 86 mm, mit Paßfeder / with key	**Torque flange DE**			
Shaft NDE	ohne Geber / without encoder	**Torque flange NDE**			
Mounting	B5	**Terminal box**	oben / on top	**regarding B3**	
Balancing	H	**Cable outlet**	AS / DE	**regarding B3**	
		Cable gland	2 x M50, 1 x M16, metr. Gewinde / metric threads		
Vibration level	R acc. QMH-BN-100-01; keine Abnahme / No Approval				

Options	tropical insulated, increased winding insulation,

Encoder	1	
	2	

Brake	

Misc:					
Dim. sheet	MDW132052 [tbd]		**Inertia**	0,07	**kg m²**
Paint	order related		**Max. altitude. ≤**	1000	**m a.s.l.**
Weight approx.	220	**kg**	**Max. ambient temperature ≤**	40	**°C**

Comment: Motorwelle mit 3.2 Zertifikat (DNV-GL), aber keine Abnahme des Motors durch DNV-GL /
Motor shaft with 3.2 certificate (DNV-GL), but no need for witnessed Approval with Surveyor
** vorläufiges Datenblatt / preliminary datasheet - created 14-Aug-2019 SK / TB **

preliminary - approximative values based on computation

Rev| Text	| Name	| Date	| valid SerNo.

B Datablad Oswald MFS PMSM Serie

OSWALD

hochdynamische PM-Synchron-Motoren

Baureihe MFS wassergekühlt

high dynamic PM synchronous motors

Series MFS water-cooled

INFO

Made in Germany
Hergestellt in Deutschland

OSWALD Elektromotoren GmbH
63897 Miltenberg - Benzstraße 12 - Telefon: ++49 9371 9719-0
www.oswald.de - eMail: oswald@oswald.de - Telefax: ++49 9371 9719-66

hochdynamischer Servoantrieb

- Überlast-Drehmomente bis ca. 30.000 Nm
- kompakt, robust
- hoher Wirkungsgrad
- wartungsfrei, verschleißfrei
- geräuscharm
- sehr trägheitsarm
- kundenspezifische Wicklungsauslegung

High Dynamic Servomotor

- overload torque up to 30.000 Nm
- compact, robust
- high efficiency
- maintenance free
- low-noise
- very low inertia
- customised electrical design

Typ	M_{max} (Nm)	n_N (min^{-1})	M_N 2) (Nm)	P_N 2) (kW)	I_N (A)	eta	n_{max} 1) (min^{-1})	J (kgm^2)	m (kg)	DF (L/min)
MFS 11.1 - 6	180	2500	68	18	36	90%	7000	0,005	40	3
MFS 11.2 - 6	360		135	35	71	91%		0,009	65	5
MFS 11.3 - 6	540		203	53	106	92%		0,014	100	8
MFS 11.4 - 6	720		270	71	139	93%		0,018	130	11
MFS 11.5 - 6	900		338	88	172	94%		0,023	165	14
MFS 13.1 - 6	260	2000	120	25	50	93%	5500	0,020	55	2
MFS 13.2 - 6	520		240	50	98	94%		0,040	110	4
MFS 13.3 - 6	780		360	75	145	95%		0,060	165	7
MFS 13.4 - 6	1040		480	101	194	95%		0,080	220	9
MFS 13.5 - 6	1300		600	126	242	95%		0,100	275	11
MFS 16.2 - 6	780	2000	333	70	133	96%	4500	0,090	200	6
MFS 16.3 - 6	1160		500	105	200	96%		0,140	260	8
MFS 16.4 - 6	1550		667	140	267	96%		0,180	320	11
MFS 16.5 - 6	1940		834	175	333	96%		0,230	380	14
MFS 16.6 - 6	2330		1000	209	396	97%		0,270	450	17
MFS 20.2 - 8	1300	1500	640	101	190	97%	3500	0,20	300	6
MFS 20.3 - 8	1950		960	151	285	97%		0,30	400	10
MFS 20.4 - 8	2600		1280	201	380	97%		0,40	500	13
MFS 20.5 - 8	3250		1600	251	475	97%		0,50	600	16
MFS 20.6 - 8	3900		1920	302	570	97%		0,60	700	19
MFS 25.3 - 8	2580	1500	1380	217	405	98%	3000	0,9	650	11
MFS 25.4 - 8	3440		1840	289	541	98%		1,2	800	14
MFS 25.5 - 8	4300		2300	361	676	98%		1,5	950	18
MFS 25.6 - 8	5160		2760	434	811	98%		1,8	1140	21
MFS 25.7 - 8	6020		3220	506	946	98%		2,0	1330	25
MFS 31.3 - 8	6500	1250	3210	420	691	98%	2000	5,0	1700	20
MFS 31.4 - 8	8700		4280	560	921	98%		7,0	1900	30
MFS 31.5 - 8	10900		5350	700	1152	98%		8,0	2200	40
MFS 31.6 - 8	13100		6420	840	1382	98%		10,0	2500	50
MFS 31.7 - 8	15200		7490	980	1613	98%		12,0	2800	50
MFS 31.8 - 8	17400		8560	1120	1843	98%		13,0	3100	60
MFS 45.3 - 8	10800	1000	7200	754	1234	98%	1500	22,0	2200	30
MFS 45.4 - 8	14400		9600	1005	1645	98%		29,0	2500	40
MFS 45.5 - 8	18000		12000	1257	2056	98%		36,0	2800	50
MFS 45.6 - 8	21600		14400	1508	2468	98%		44,0	3100	70
MFS 45.7 - 8	25200		16800	1759	2879	98%		50,0	3400	80
MFS 45.8 - 8	28800		19200	2010	3290	98%		58,0	3700	90

vorläufige Werte aus Berechnung / preliminary data; subject to change

1) mechanisch zulässige Höchstdrehzahl
2) Werte für Taktfrequenz 4kHz
U_ZK max. 750V

Auslegungsbasis 400 .. 500V - Vorlauftemperatur 25°C
- bei höheren Drehzahlen ist eine Innenumluft, Fremdkühlung oder andere Rotorkühlung nötig!
bei langen Maschinen ergibt sich tendenziell eine geringe Induktivität, darum kann ein Sinus-Filter nötig werden!

Typical applications:
ship propulsion, servo electric hydraulic pumps, forming presses, injection moulding machines, machine tools, testing systems, winches, hydro power

Version 1118

Qualitätsmanagement ISO 9001:2015
Energiemanagement ISO 50001
Umweltmanagement ISO 14001:2015

C Motor parametere fra HPS

Motorparametere fra HPS

Motortype	PMSM	-
Nominell spenning	390	V
Nominell frekvens	107.5	Hz
Nominelt turtall	2150	rpm
Nominell strøm	140	A
Nominell effekt	85	kW
Nominell cosφ	0.88	-
PMSM R_s	0.02	p.u.
PMSM L_d	0.5	p.u.
PMSM L_q	0.57	p.u.

D Matlab/Simulink kode

```
% Author Sivert Heidsve

%% Setup and guide for PMSM drivesystem model
% This section is an instruction list for running and simulate the PMSM drive
% model in Matlab/Simulink.

% 1. Run the script PMSM_Drive_script.m
%    This saves the parameters in the matlab workspace.
% 2. Open the simulink file PMSM_Drive_model.slx
%    Make sure the parameters for the model are loaded according to
%    the script.
% 3. Select desired simulation time.
% 4. Select reference speed [rpm] by changing value in the "Hastighet_ref"
%    block. Default is set to 2150 [rpm].
% 5. Select applied load torque and behavior. Default is set to a step from
%    0 to 0.5*T_rated after 1 sek.
% 5. Run the simulink model.
% 6. Open Simulation Data Inspector to watch desired plots.

%% Section 1 - Per unit to SI calculations
% This section calculates SI-unit values for Rs, Ld, Lq and flux_pm based
% on the values in p.u. and other motorparameters in SI-units.

% Insert values based on your system and motor.
Vnom = 390;                 % Nominal motor voltage      [V]
Pnom = 85e3;                % Nominal motor power        [W]
Inom = 140;                 % Nominal motor current      [A]
cosphi_nom = 0.88;          % Power factor
m_freq_nom = 107.5;         % Nominal motor frequency    [hZ]
```

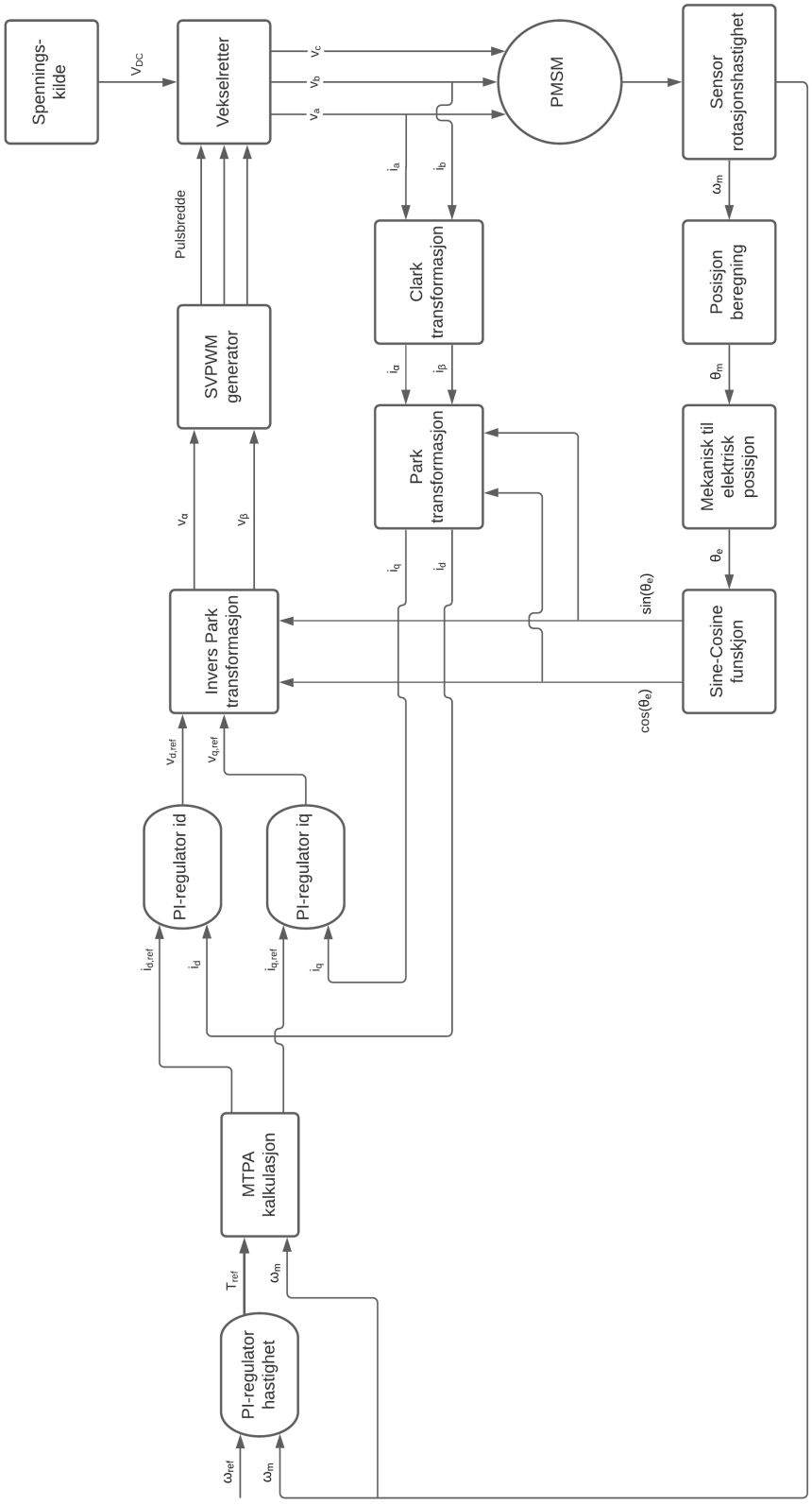
```
m_speed_nom = 2150;       % Nominal motor speed        [rpm]

% Calculated apparant power
Snom = sqrt(3)*Vnom*Inom; % Apparant power             [VA]

% Insert per unit values based on your motor:
% (these values are per phase)
Rs_pu = 0.013;            % Stator resitance per phase    [p.u.]
Ld_pu = 0.5;              % d-axis inductance             [p.u.]
Lq_pu = 0.57;             % q-axis inductance             [p.u.]
flux_pm_pu = 0.95;        % Flux linkage permanet magnets [p.u.]

% These equations calculates the base values.
Base_power = Snom/3;
Base_voltage = Vnom/sqrt(3);
Base_current = (Base_power)/(Base_voltage);
Base_resistance = (Base_voltage)/(Base_current);
Base_impedance = (Base_voltage)/(Base_current);
Base_inductance = (Base_impedance)/(2*pi*m_freq_nom);
Base_flux_linkage = (Base_voltage*sqrt(2))/(2*pi*m_freq_nom);

% These equations calculates the values in SI-unit.
Rs_SI = Rs_pu*Base_resistance;                % Rs in SI-units      [ohm]
Ld_SI = Ld_pu*Base_inductance;                % Ld in SI-units      [H]
Lq_SI = Lq_pu*Base_inductance;                % Lq in SI-units      [H]
flux_pm_SI = flux_pm_pu*Base_flux_linkage;    % flux linkage of permanent
                                              % magnets in SI-units [wb]

%% Section 2 - Motor parameters for PMSM
% This section declares and sets the motor parameters.
% It is possible to set values of Rs, Ld, Lq and flux_pm directly in
% with SI units and skip Section 1.

% Insert values based on your motor:
pmsm.p       =  3;            % Motor polepairs
pmsm.Rs      =  Rs_SI;        % Per phase Stator resistance       [ohm]
pmsm.Ld      =  Ld_SI;        % d-axis inductance                 [H]
pmsm.Lq      =  Lq_SI;        % q-axis inductance                 [H]
pmsm.flux_pm =  flux_pm_SI;   % Flux linkage permanent magnets    [wb]
pmsm.J       =  0.07;         % Inertia                           [kgm^2]
pmsm.B       =  0.0012;       % Viscous friction coefficient      [Ns/m]
T_rated      =  378;          % Rated torque                      [Nm]
V_dc_max     =  800;          % DC input voltage                  [V]
I_max        =  350;          % Maximum phase current of the motor [A]

%% Section 3 - Initial values and saturation limitations
% This section declares and sets the initial values used in the drivesystem
% and PMSM model. It also declares and sets upper and lower saturation
```

```
78  % limitations for saturation blocks and integrators.
79
80  % Insert initial values based on your testscenario:
81  id_init    = 0;    % Initial d-axis current  [A]
82  iq_init    = 0;    % Initial q-axis current  [A]
83  theta_init = 0;    % Initial theta           [rad]
84  omega_init = 0;    % Initial omega           [rad/s]
85
86  % Insert saturation limits based on your system:
87  Upper_sat_lim = 10e6;       % Upper saturation limit current calculation
88  Lower_sat_lim = -10e6;      % Lower saturation limit current calculation
89
90  Upper_sat_lim_Te_wc = inf;  % Upper saturation limit for Te in speed
91                              % control inside anti windup loop.
92  Lower_sat_lim_Te_wc = -inf; % Lower saturation limit for Te in speed
93                              % control inside anti windup loop.
94
95  % Calculation of saturation limits for voltage reference out from
96  % controller
97  Upper_sat_lim_vref_cc = 0.95*V_dc_max/sqrt(3); % Upper saturation limit for
98                                                 % v_ref in current control.
99  Lower_sat_lim_vref_cc = -0.95*V_dc_max/sqrt(3);% Lower saturation limit for
100                                                % v_ref in current control.
101
102
103 %% Section 4 - Sample frequency and sample time
104 % This section declares and sets the sample frequency and sample time
105 % for the simulations of the modell.
106 % Sample rate is calculated based on an imaginary inverter
107 % This modell does not implement an inverter.
108 % It is posible to set sample rate for different parts of the model,
109 % however they are set the same here.
110
111 PWM_frequency         = 10e3;            % Switching frequency inverter  [Hz]
112 Ts                = 1/PWM_frequency; % Sampling time inverter        [s]
113 Ts_simulink       = Ts/4;            % Sampling time drivesystem     [s]
114 Ts_motor          = Ts/4;            % Sampling time motor modell    [s]
115 Ts_inverter       = Ts/4;            % Sampling time inverter        [s]
116
117 %% Section 5 - Parameters for PI-controllers
118 % This sections declares, calculate and sets the controll parameters for
119 % the speed controller and current controller. The calculations are based
120 % on motor parameters and bandwith
121
122 % Insert bandwith for current controller and speed controller:
123 alfa_c = 1000; % Bandwith current controller  [rad/s]
124 alfa_w = 100;  % Bandwith speed controller    [rad/s]
125
```

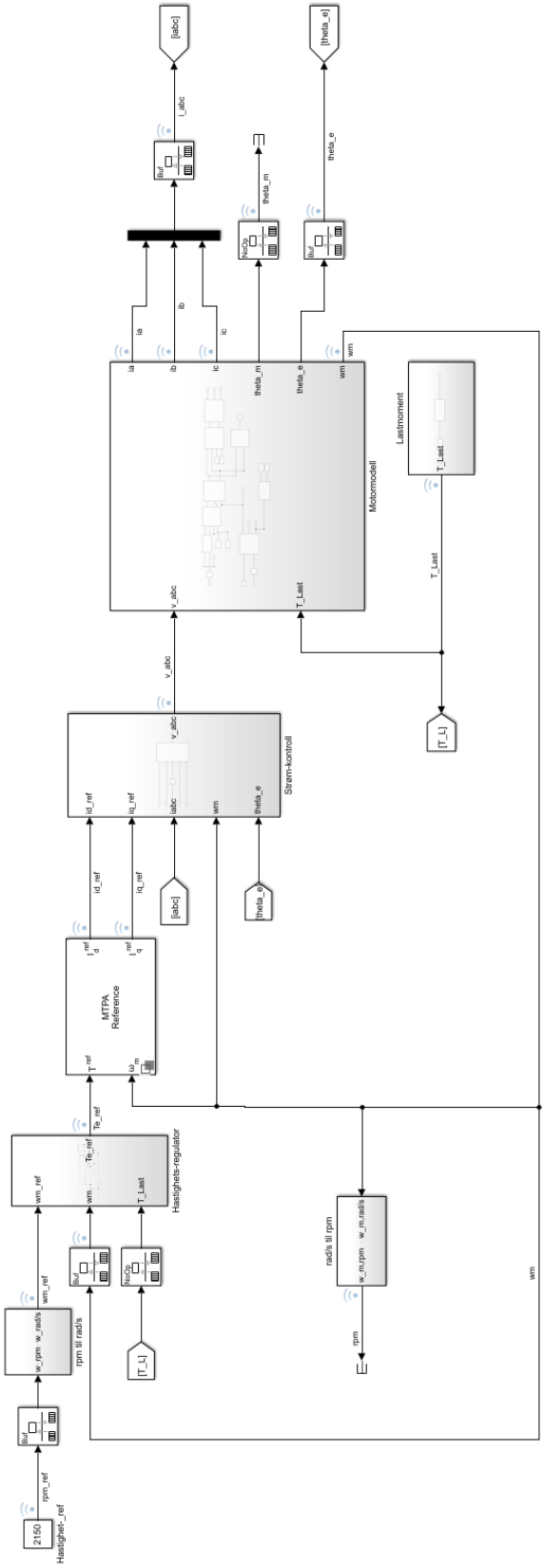
```
126
127 % Current controll parameter calculations:
128 % Integral constant are calculated with and without activ damping
129
130 R_ad = (alfa_c*pmsm.Ld) - pmsm.Rs; % Active damping for id current.
131 R_aq = (alfa_c*pmsm.Lq) - pmsm.Rs; % Active damping for iq current.
132
133 Kp_id = alfa_c*pmsm.Ld;            % Proportional gain for id.
134 Ki_id = alfa_c*pmsm.Rs;            % Integral constant for id without
135                                    % activ damping (not in use).
136 Ki_id_ad = ((alfa_c)^2)*pmsm.Ld;   % Integral constant for id with
137                                    % active damping (used in model).
138 H_cd = 1/Kp_id;                    % Anti windup loop gain for id.
139
140 Kp_iq = alfa_c*pmsm.Lq;            % Proportional gain for iq.
141 Ki_iq = alfa_c*pmsm.Rs;            % Integral constant for iq  without
142                                    % activ damping (not used).
143 Ki_iq_ad = ((alfa_c)^2)*pmsm.Lq;   % Integral constant for iq with
144                                    % active damping (used in model).
145 H_cq = 1/Kp_iq;                    % Anti windup loop gain for iq.
146
147 % Speed control parameter calculations
148 B_aw = (alfa_w*pmsm.J)-pmsm.B;     % Active damping speed.
149 Kp_w = alfa_w*pmsm.J;              % Proportional gain for speed control.
150 Ki_w = alfa_w*pmsm.B;              % Integral constant for speed control.
151                                    % without active damping (not used).
152 Ki_w_ad = ((alfa_w)^2)*pmsm.J;     % Integral contant for speed control with
153                                    % active damping (used in modell).
154 H_w = 1/Kp_w;                      % Anti windup loop gain.
```

E Simulink-modell figurer

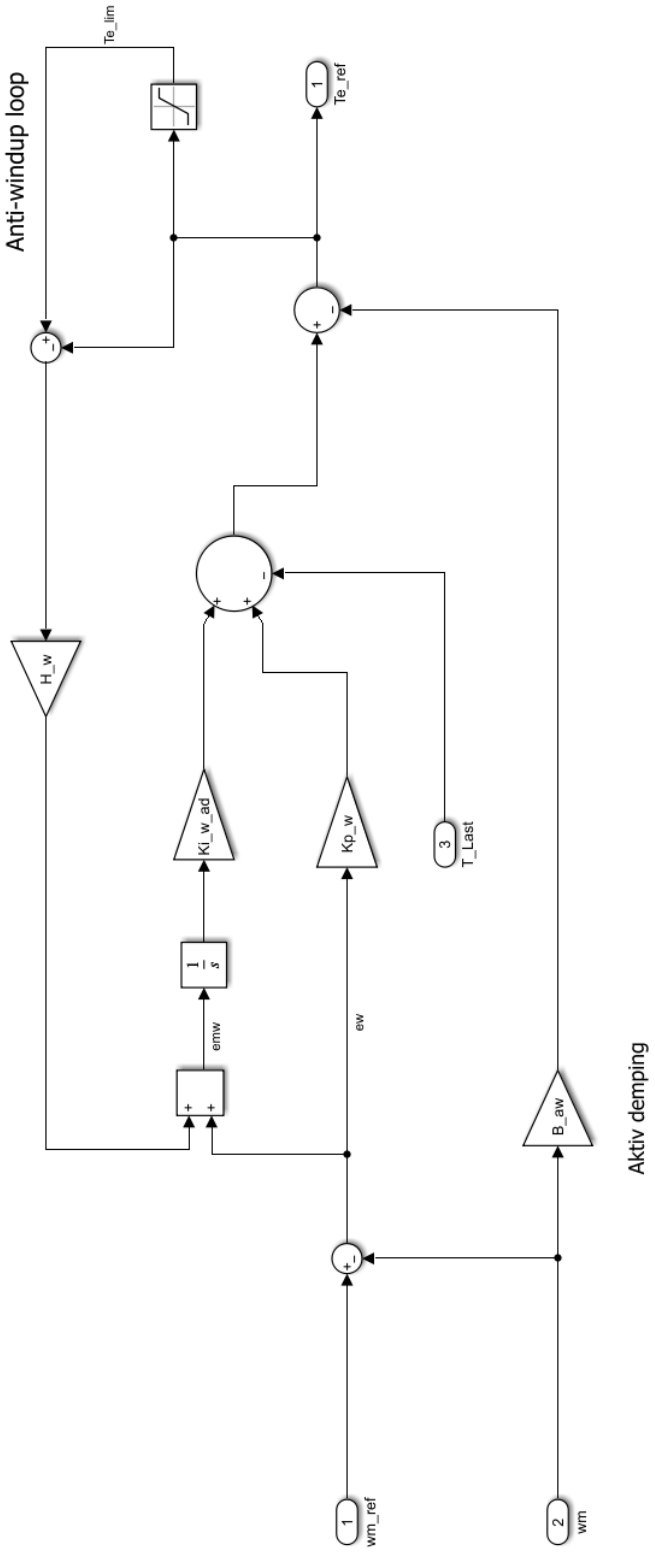
E.1 Feltorientert kontroll av permanentmagnet synkronmaskin



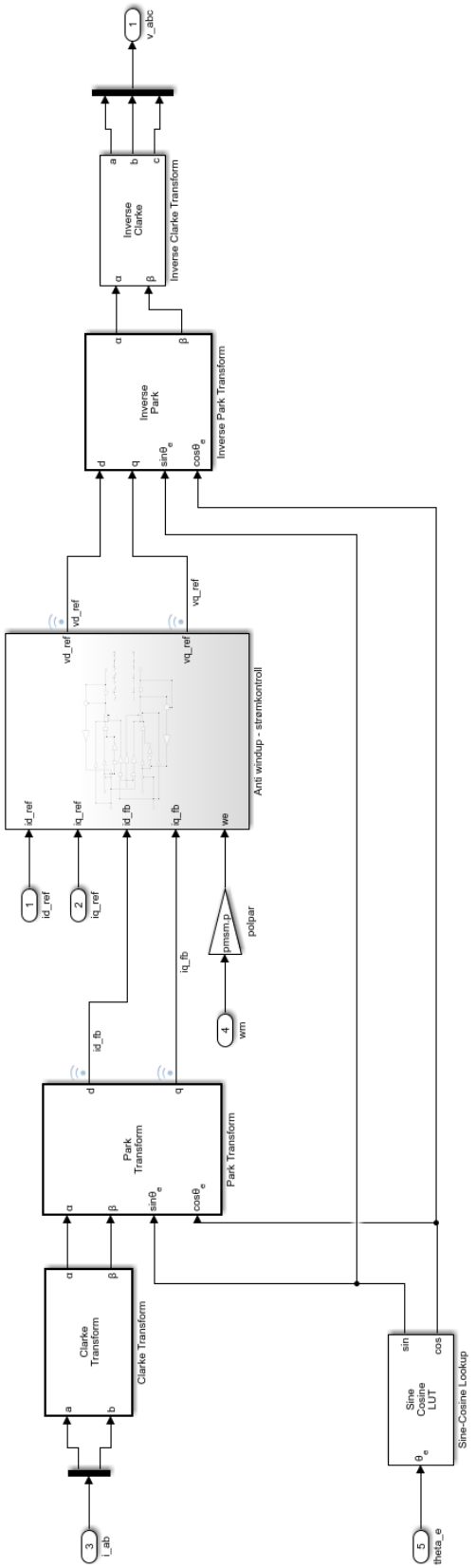
E.2 FOC av PMSM i Simulink-modellen



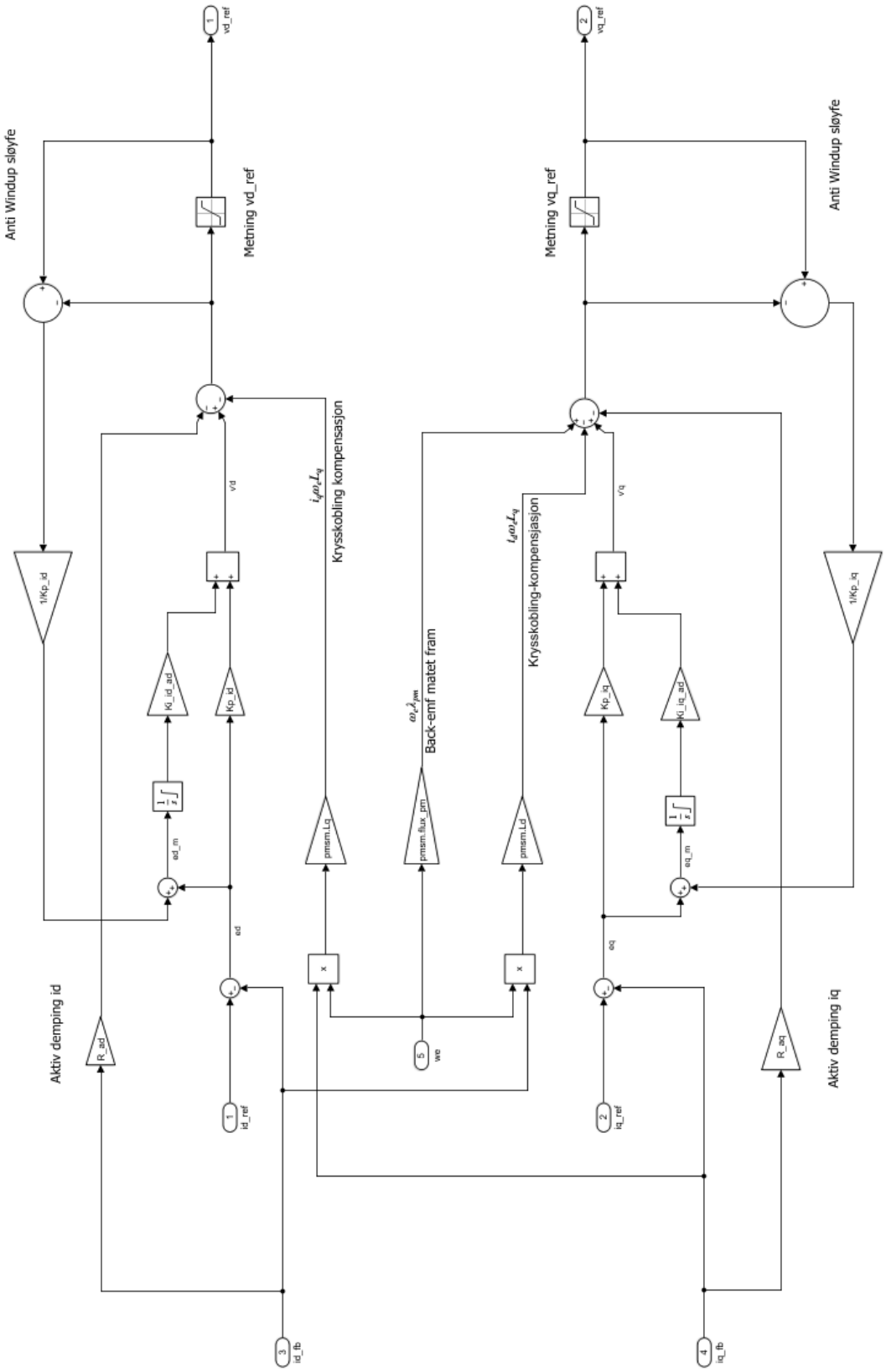
E.3 Hastighetsregulator i Simulink-modellen



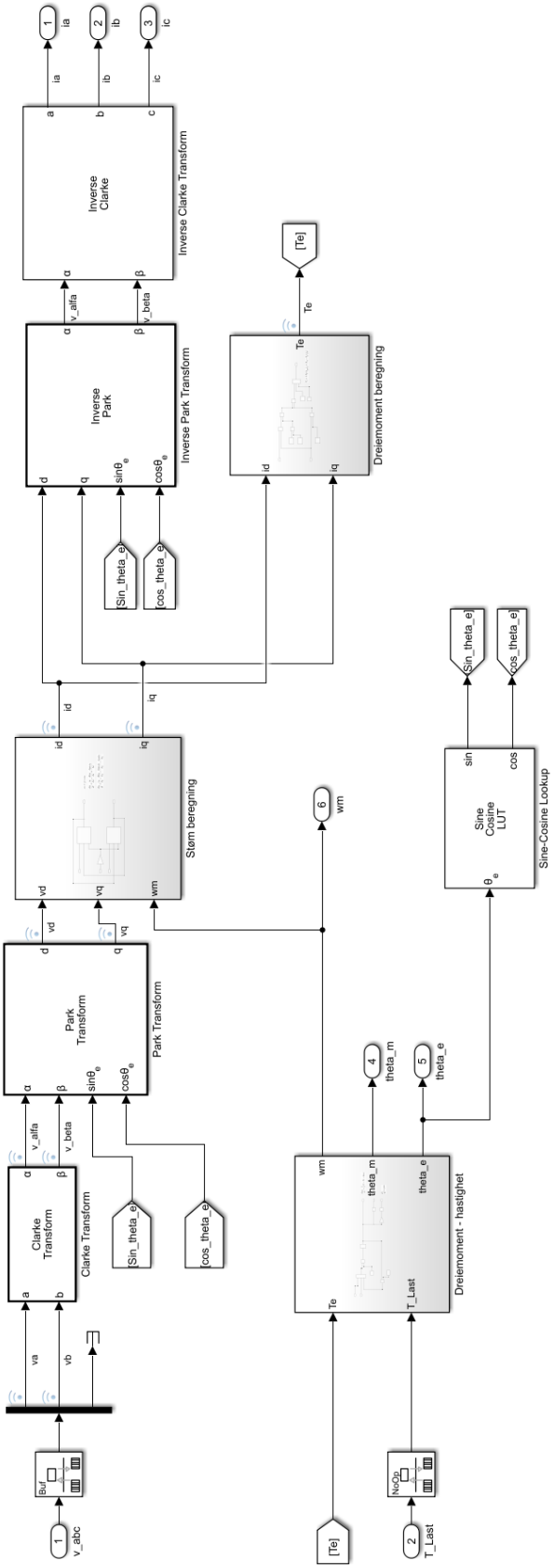
E.4 Strømkontrollsystem i Simulink-modellen



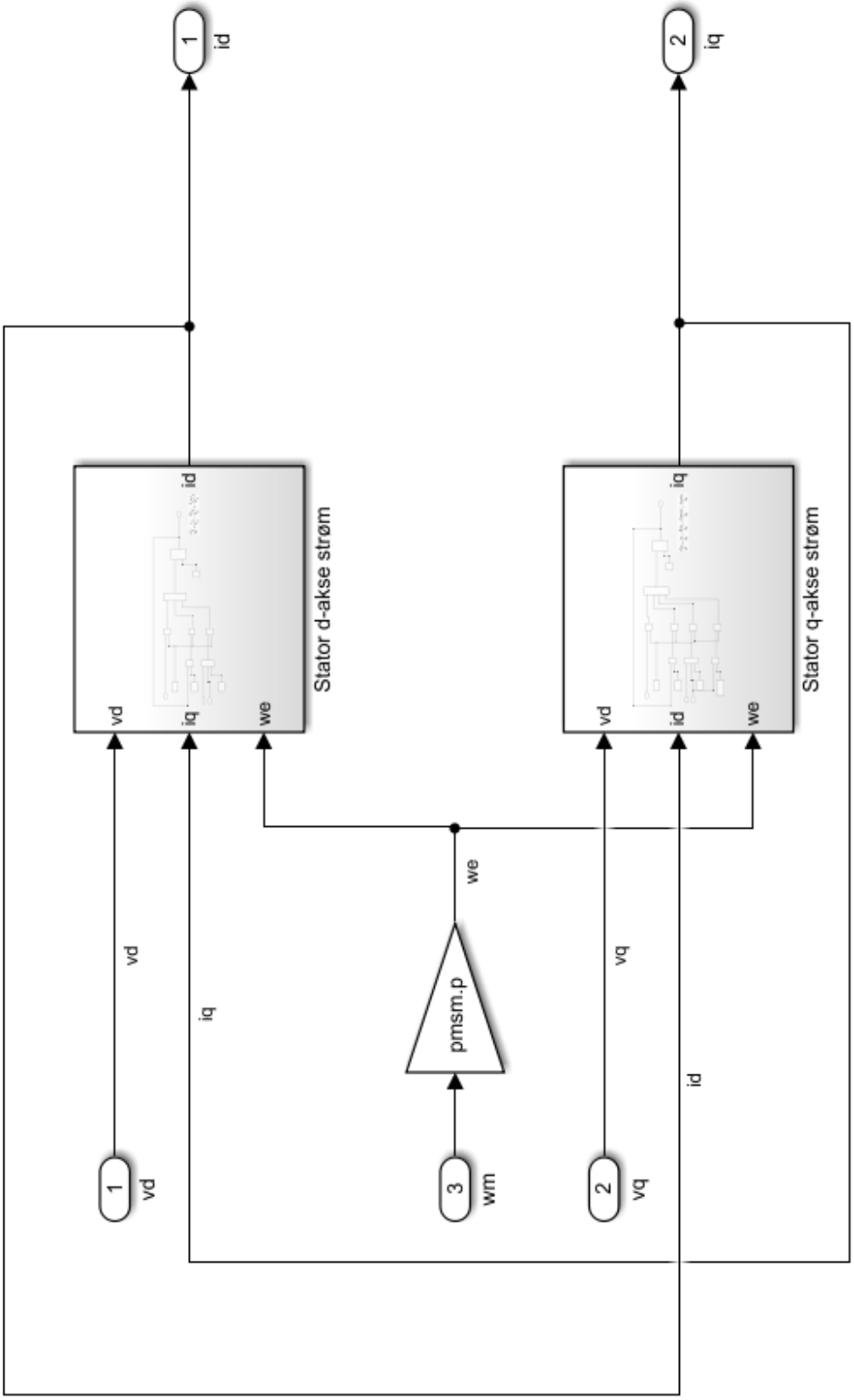
E.5 Strømregulator i Simulink-modellen



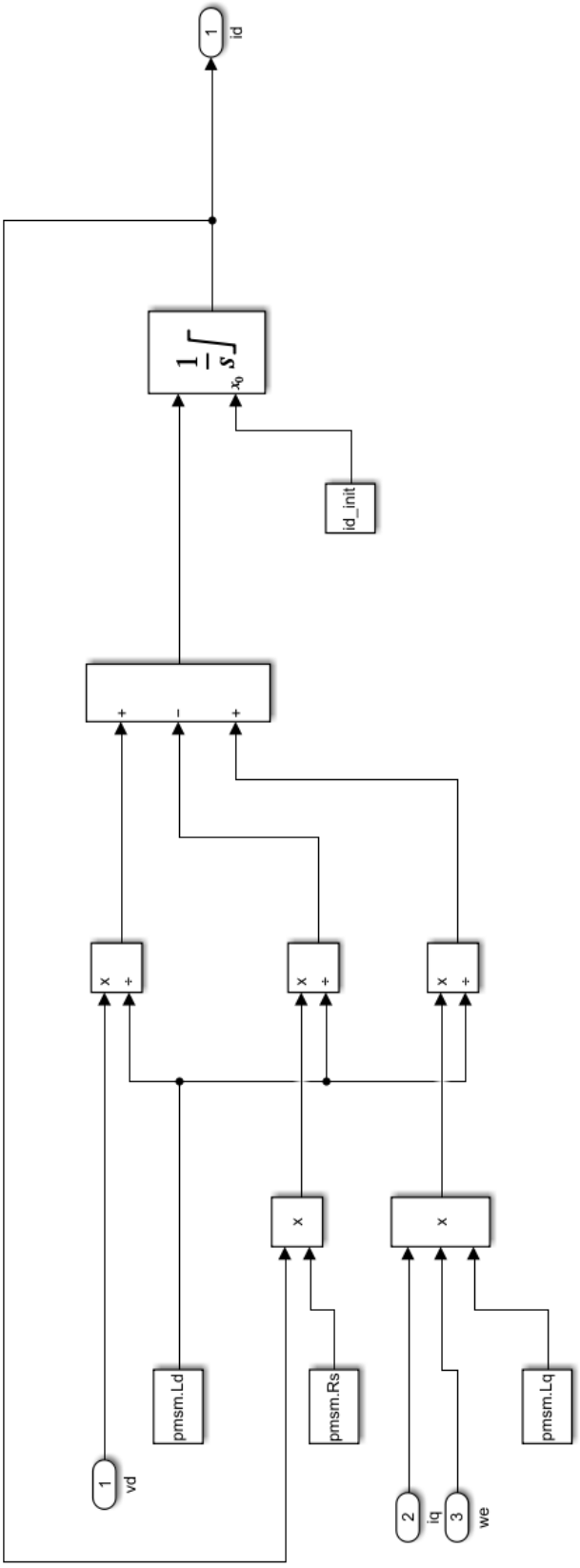
E.6 PMSM-motormodell i Simulink-modellen



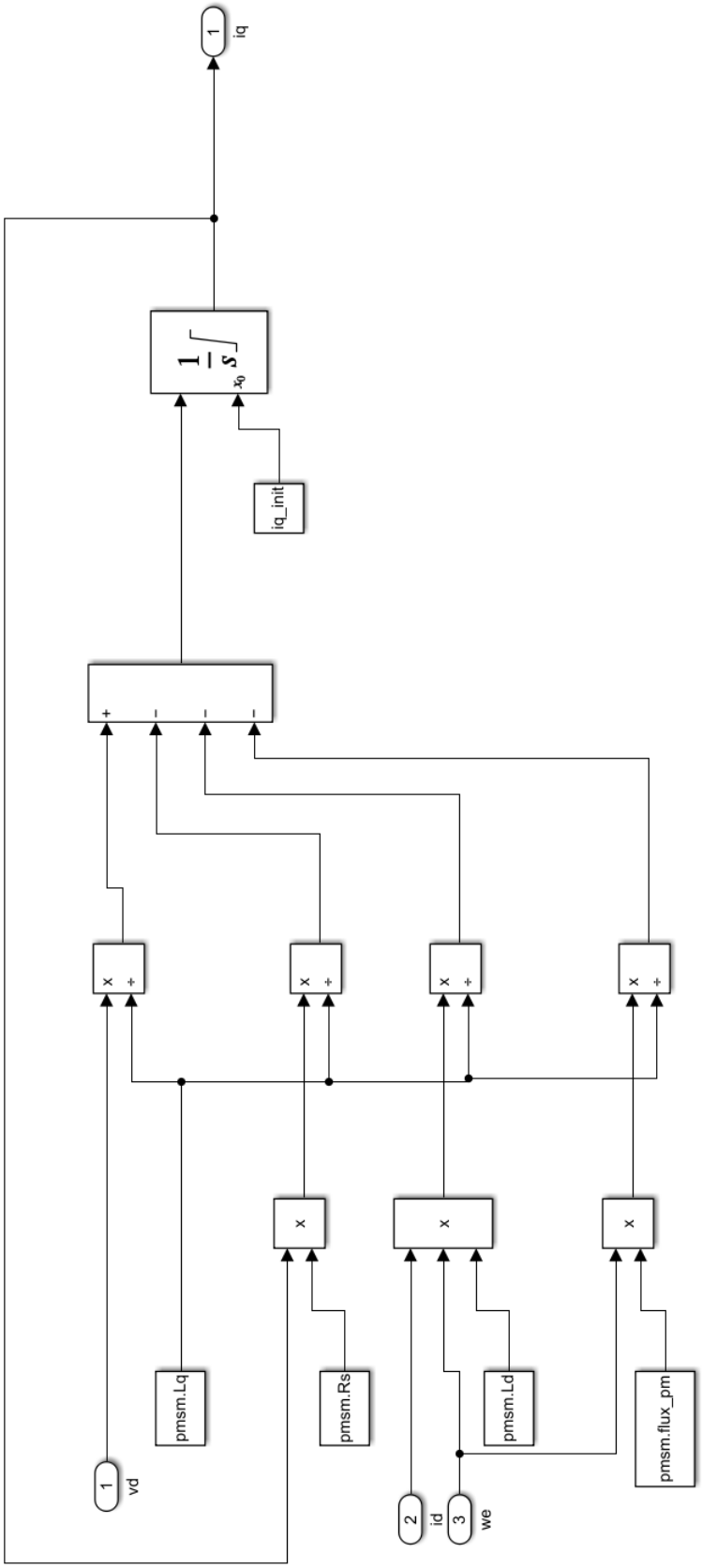
E.7 Beregning av statorstrømmer i Simulink-modellen, utside



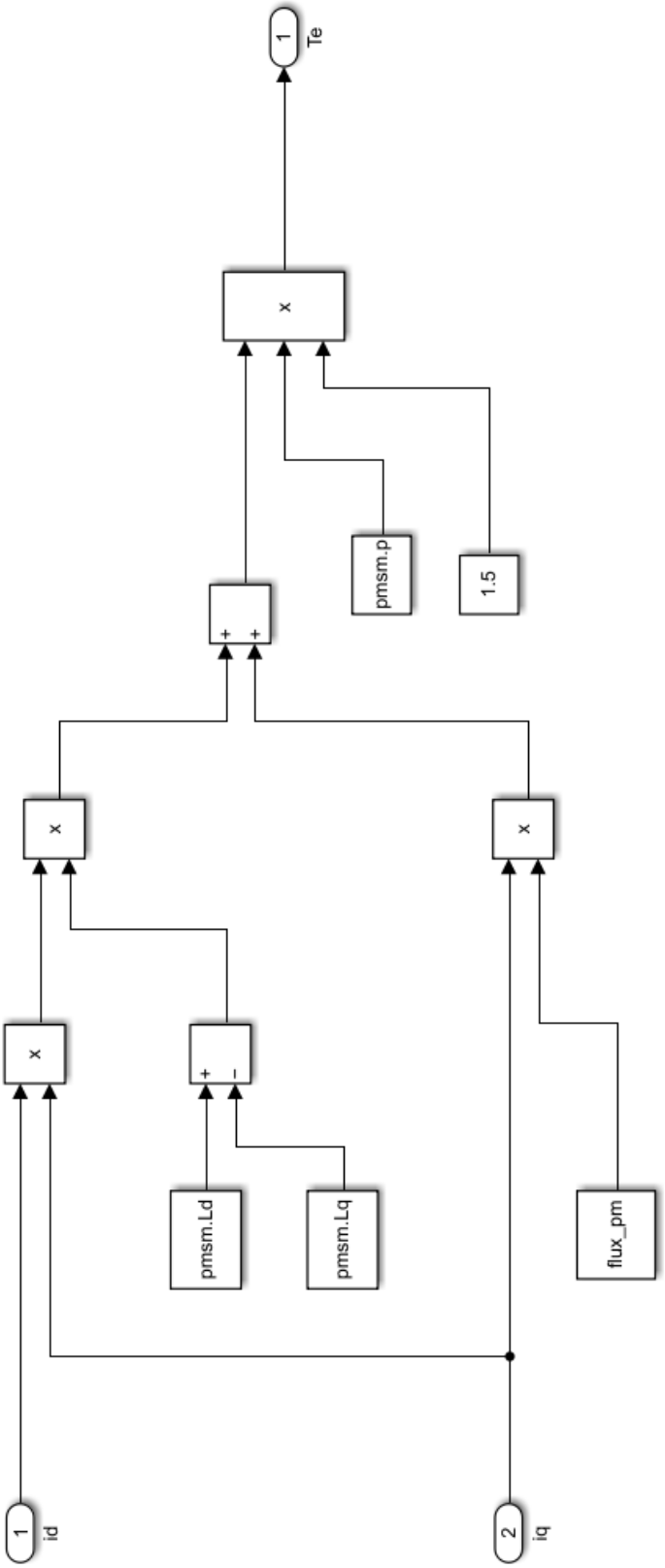
E.8 Beregning av statorstrømmer i Simulink-modellen, d-akse strøm



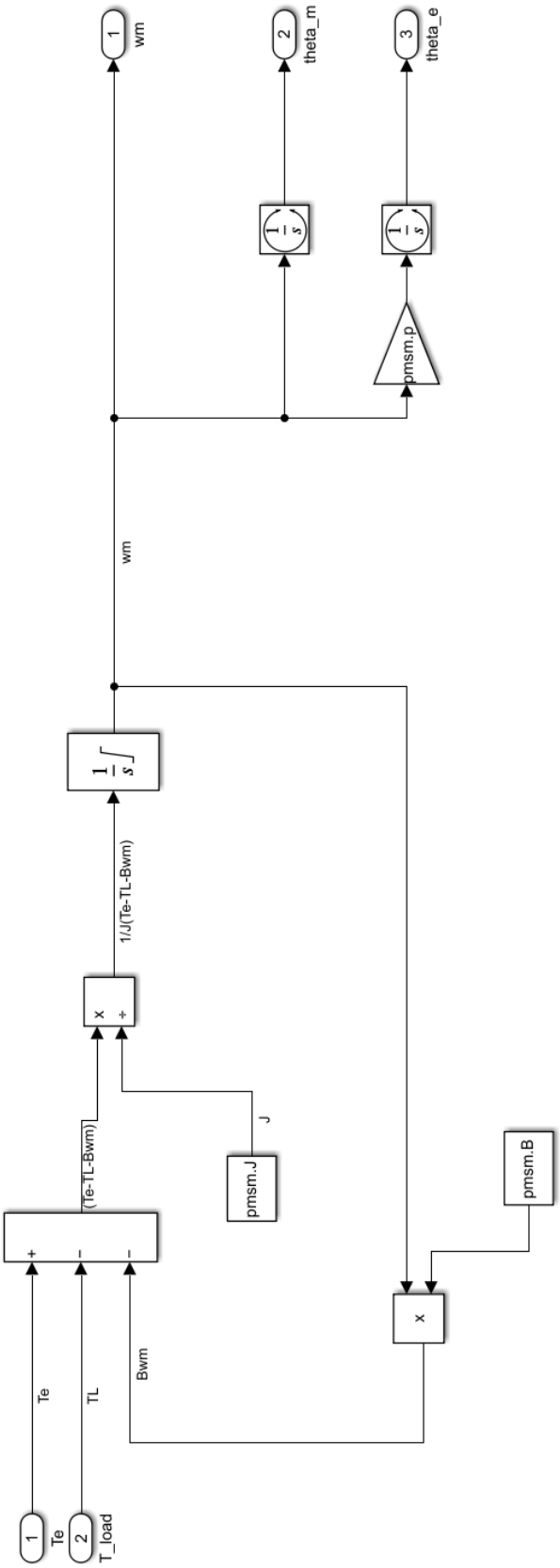
E.9 Beregning av statorstrømmer i Simulink-modellen, q-akse strøm



E.10 Beregning av dreiemomentet T_e i Simulink-modellen



E.11 Beregning av ω_m og θ_e i Simulink-modellen



F Python kode

Navnet på kapitlene er det samme som filnavnene i Python programmet

F.1 main.py

```
from Functions import *
from Initial_Values import *
from PMSM_Parameters import *
import numpy as np


# Beregning av tid
sample_time = float(input('Sample time in seconds: '))  # Hvor mange sekunder
                                                        # som skal simuleres.
sample_points = sample_time / t_sample  # Hvor mange sample-punkter
                                        # simuleringen skal bestå av.

# Åpne dokumentet Results.txt for å kunne skrive ned verdiene for
# hver sekvens av simulasjonen.
f = open('Results.txt', 'w')
f.write('t, Vd, Vq, w, i_d, id_ref, i_q, iq_ref, Theta, T_e, Turtall\n'')

for t in range(int(sample_points)):  # Simuleringsløkke

    # Beregner elektrisk turtall
    w_e = p * w_m

    # PI-regulator for turtall beregner en ny referanse verdi for
    # elektrisk dreiemoment(Te_ref).
    Te_ref, sum_w = PI_w_m(Kp_w, Ki_w, w_ref, w_m, sum_w, t_sample, T_max)

    # Ved hjelp av max torque per ampere så brukes Te_ref til å
    # beregne referanseverider for i_d og i_q.
    id_ref, iq_ref = id_iq_ref_calc(FluxPM, Ld, Lq, I_nom, p, Te_ref, I_max)

    # PI-regulatorer som beregner nye verdier for Vd og Vq.
    Vd, sum_d = PI_Vd(Kp_id, Ki_id, id_ref, i_d, sum_d, t_sample, V_sat_lim,
        w_e, Lq, i_q)
    Vq, sum_q = PI_Vq(Kp_iq, Ki_iq, iq_ref, i_q, sum_q, t_sample, V_sat_lim,
        w_e, Ld, i_d, FluxPM)

    # Beregner verdien for i_d og i_q med å bruke verdiene fra
    # PI-regulatorene for Vd og Vq, og PMSM_Parameters.py filen.
    i_d_temp = id_calc(Vd, Ld, Lq, Rs, i_d, i_q, t_sample, w_m, p)
    i_q_temp = iq_calc(Vq, Ld, Lq, Rs, i_d, i_q, t_sample, w_m, p, FluxPM)

```

```
42
43      # Beregner elektrisk dreiemoment
44      Te_temp = t_e_calc(i_d, i_q, Ld, Lq, p, FluxPM)
45
46      # Beregner rotorhastighet(w_m) med eller uten sprang i lastmomentet
47      if Load_step == 0:  # Beregner w_m med konstant lastmoment
48          w_temp = calc_w_m(t_sample, J, Te, T_l, B, w_m)
49
50      else:  # Beregner w_m med sprang i lastmomentet etter
51             # LS_time/t_sample sekunder.
52          if t < (LS_time / t_sample):
53              T_l = T_l_0
54              w_temp = calc_w_m(t_sample, J, Te, T_l, B, w_m)
55          else:
56              T_l = T_l_1
57              w_temp = calc_w_m(t_sample, J, Te, T_l, B, w_m)
58
59
60      # Skriver ned resultatene fra simuleringen i filen "Results.txt"
61      f.write('%.6f, %.4f, %.4f, %.4f, %.4f, %.4f, %.4f, %.4f, %.4f, %.5f, %.4f\n' %
62              (t * t_sample, Vd, Vq, w_m, i_d, id_ref, i_q, iq_ref, theta_e, Te, rpm))
63
64      # Tildeler variablene de nye verdiene.
65      i_d = i_d_temp
66      i_q = i_q_temp
67      w_m = w_temp
68      Te = Te_temp
69
70  f.close()
```

F.2 Functions.py

```
import numpy as np


# Vinkelhastighetsregulator. Tar inn w_m og lager en referanse
# for elektrisk dreiemoment.
def PI_w_m(Kp, Ki, w_ref, w_m, sum_w, dt, T_max):
    Te_ref = Kp*(w_ref - w_m) + Ki*(w_ref - w_m)*dt + sum_w

    # Metning i regulatoren. Referansen vil aldri bli større
    # eller mindre enn +-T_max.
    if Te_ref > T_max:
       Te_ref = T_max
    elif Te_ref < -T_max:
        Te_ref = -T_max
    sum_w = sum_w + Ki*(w_ref - w_m)*dt
    return Te_ref, sum_w


#Beregner dreiemomentvinkelen Beta
def beta_calc(fluxPM, Ld, Lq, I_rated):
    Beta = np.arccos(-(fluxPM)/(4*(Ld - Lq)*I_rated) - np.sqrt(0.5 + ...
    ((fluxPM)/(4*(Ld - Lq)*I_rated))**2))
    return Beta


# Finner Is referansen som blir brukt i MTPA til å finne i_d,ref
# og i_q,ref ut ifra Te_ref.
def Is_ref_calc(fluxPM, Ld, Lq, I_rated, p, T_ref):
    Beta = beta_calc(fluxPM, Ld, Lq, I_rated)
    Is_ref = -(fluxPM)/(2*(Ld - Lq)*np.cos(Beta)) + np.sqrt(((2)/(3*p*(Ld - Lq)*...
    np.sin(Beta)*np.cos(Beta)))*T_ref + ((fluxPM)/(2*(Ld - Lq)*np.cos(Beta)))**2)

    return Is_ref


# Beregner i_d,ref og i_q,ref ved hjelp av MTPA metode vist i
# kapittel 2.3.5.
def id_iq_ref_calc(fluxPM, Ld, Lq, I_rated, p, T_ref, I_max):
    Beta = beta_calc(fluxPM, Ld, Lq, I_rated)
    Is_ref = Is_ref_calc(fluxPM, Ld, Lq, I_rated, p, T_ref)

    # Metningsgrense. Stømmen blir ikke større enn det vekselretteren kan levere.
    if Is_ref > I_max:
        Is_ref = I_max
    elif Is_ref < -I_max:
        Is_ref = -I_max
```

```
47
48      id_ref = Is_ref*np.cos(Beta)
49      iq_ref = Is_ref*np.sin(Beta)
50      return id_ref, iq_ref
51
52
53  # PI regulator for Vd. Tar inn i_d,ref som referanse
54  def PI_Vd(Kp, Ki, i_dref, i_d, sum_d, dt, V_max, w_e, L_q, i_q):
55      Vd = Kp*(i_dref - i_d) + Ki*(i_dref - i_d)*dt + sum_d + w_e*L_q*i_q
56
57      # Metningsgrense. Spenningen blir ikke større enn det
58      # vekselretteren kan levere.
59      if Vd > V_max:
60          Vd = V_max
61      elif Vd < -V_max:
62          Vd = -V_max
63      sum_d = sum_d + Ki*(i_dref - i_d)*dt
64      return Vd, sum_d
65
66
67  # PI regulator for Vq. Tar inn i_q,ref som referanse
68  def PI_Vq(Kp, Ki, i_qref, i_q, sum_q, dt, V_max, L_d, i_d, fluxPM):
69      Vq = Kp*(i_qref - i_q)+Ki*(i_qref - i_q)*dt + sum_q + w_e*L_d*i_d + w_e*fluxPM
70
71      # Metningsgrense. Spenningen blir ikke større enn det
72      # vekselretteren kan levere.
73      if Vq > V_max:
74          Vq = V_max
75      elif Vq < -V_max:
76          Vq = -V_max
77      sum_q = sum_q + Ki*(i_qref - i_q)*dt
78      return Vq, sum_q
79
80
81  # Beregner i_d med differanselikningen for i_d gitt i kapittel 2.2.5
82  def id_calc(v_d, L_d, L_q, R, i_d, i_q, Dt, w_m, p):
83      diddt = (1 / L_d) * (v_d - R * i_d - L_q * p * w_m * i_q)
84      new_id = i_d + Dt * diddt
85      return new_id
86
87
88  # Beregner i_q med differanselikningen for i_q gitt i kapittel 2.2.5
89  def iq_calc(v_q, L_d, L_q, R, i_d, i_q, Dt, w_m, p, flux_link):
90      diqdt = (1 / L_q) * (v_q - R * i_q - L_d * p * w_m * i_d - p * w_m * flux_link)
91      new_iq = i_q + Dt * diqdt
92      return new_iq
93
94
```

```
95  # Beregner det elektriske dreiemomentet med likningen for T_e
96  # gitt i kapittel 2.2.5.
97  def t_e_calc(i_d, i_q, L_d, L_q, P, flux_link):
98      Te = (3 / 2) * P * (flux_link * i_q + (L_d - L_q) * i_d * i_q)
99      return Te
100
101
102 # Beregner den mekaniske vinkelhastigheten med
103 # differanselikningen for omega_m gitt i kapittel 2.2.5.
104 def calc_w_m(Dt, j, t_e, t_l, b, w_m):
105     if b == 0:
106         dwdt = (1/j)*(t_e - t_l - 0.001*t_l)
107     else:
108         dwdt = (1/j)*(t_e - t_l - b*w_m)
109     w = w_m + Dt*dwdt
```

F.3 PMSM_Parameters.py

```
# Batteri og vekselretter parametere
I_max = 350                     # Maks strøm levert fra vekselretter (A)
V_max = 800                     # Maks spenning levert fra vekselretter (V)
V_sat_lim = (0.95*V_max)/np.sqrt(3)


# Motor parametere
PMSM_model = 'Oswald MFS13.3-6W'
p       = 3                     # Polpar
Rs      = 0.0209                # Stator Resistans (ohms)
Ld      = 0.0012                # D-akse induktans (H)
Lq      = 0.0014                # Q-akse induktans (H)
J       = 0.07                  # Motorans treghetsmoment (Kg-m2)
B       = 0.01                  # Friksjonskoefisien (Kg-m2/s)
FluxPM  = 0.4479                # Permanentmagnet flukskobling (Wb)
I_nom   = 140                   # Nominell strøm Is (A)
N_max   = 4210                  # Nominell fart (RPM)
T_nom   = 378                   # Nominelt dreiemoment (Nm)
T_max   = 780


# Regulator parametere

# Vd regulator parametere
Kp_id = 20.9
Ki_id = 1190.6
# Vq regulator parametere
Kp_iq = 20.9
Ki_iq = 1357.3
# Hastighet regulator parametere
Kp_w = 200
Ki_w = 25
```

F.4 Initial_Values.py

```
from PMSM_Parameters import *
import numpy as np

# Initialverdier for stømmer og spenninger
Vd = 0   # Initialverdien for Vd spenningen
Vq = 0   # Initialverdien for Vq spenningen
i_d = 0  # Initialverdien for id strømmen
i_q = 0  # Initialverdien for iq strømmen

# Referanseverdier
iq_ref = 0                      # Referanse verdi for i_q
id_ref = 0                      # Referanse verdi for i_d
rpm_ref = 2150                  # Referanseverdi for rotorhastighet
                                # omdreininger per minutt.
w_ref = rpm_ref*(2*np.pi)/60    # Referanseverdi for rotorhastighet
                                # i radianer per sekund.

# Initialverdier for summen av integralleddene i PI-regulatorene
sum_d = 0
sum_q = 0
sum_w = 0

# Initialverdier for rotasjonshastigheter og vinkelposisjoner
w_m = 0
w_e = 0
theta_e = 0

# Verdier for dreiemoment
Te = 0  # Elektrisk dreiemoment
T_l = 0*T_nom # Lastmoment

# Betingelser for lastsprang
Load_step = 0        # Hvis Load_step = 0 ingen: sprang i lasten,
                     # hvis Load_step = 1: sprang i lasten.
LS_time = 0.5        # Tiden da spranget skjer i sekunder.
T_l_0 = 0*T_nom      # Lastmoment før spranget.
T_l_1 = 0.5*T_nom    # Lastmoment etter spranget.

t_sample = 1 / 40000 # Tiden for et sample i sekunder.
```

F.5 Plot

Denne koden er ikke i det ferdige produktet, men er tatt med for å vise hvordan grafene er laget i Python.

```
import numpy as np
import matplotlib.pyplot as plt

list_1 = []
list_2 = []

for t in range(int(sample_points)):  # Simulation loop
    ...
    # Linje 21 til 62 i main.py
    ...
    # Bruker i_d og id_ref som eksemplervariabel
    list_1.insert(t, rpm)
    list_2.insert(t, )

    # Linje 65 til 68 i main.py

# Plotting
t_plot = np.arange(0, sample_time, t_sample)
list_1_plot = np.array(list_1)
list_2_plot = np.array(list_2)

plt.rcParams['font.size'] = '15'
plt.plot(t_plot, list_1_plot, linewidth=2, label='i_d')
plt.plot(t_plot, list_2_plot, linewidth=2, label='i_d,ref', linestyle='--')
plt.xlabel('Tid [s]')
plt.ylabel('Strøm [A]') # Må endres etter hva som skal plottes
plt.legend()
plt.tight_layout()
plt.grid(True)
plt.show()
```

G Manual for Python-modell

Innhold

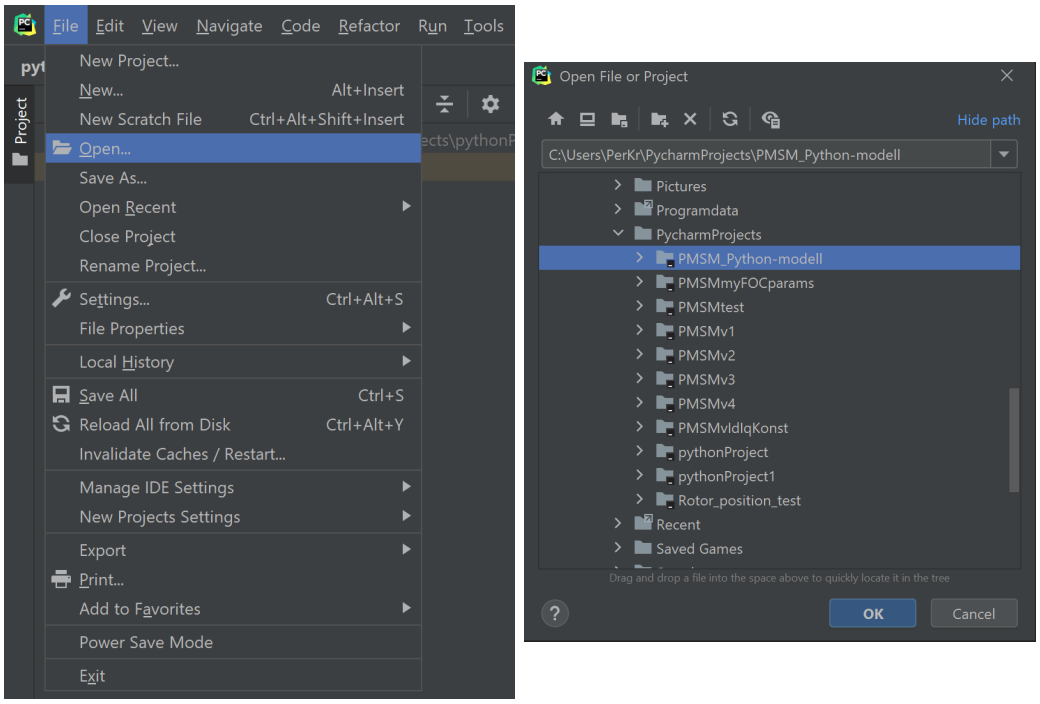
- Generell informasjon
- Installasjon
- Kjøre Simulering
- Se Resultater

Generell informasjon
Det er to versjoner av Python-modellen, en med plotte-funksjon og en uten. Versjonen med plotte-funksjon er kalt "PMSM_Python-modell_plot" og versjonen uten er kalt "PMSM_Python-modell". Plotting av grafer i Python-modellen kan kreve en god del minne, fordi plottefunksjonen bruker Python-lister til å lage grafene. Hvis det ikke er nødvendig å se resultatene på grafisk form så anbefales det å bruke versjonen uten plotte-funksjon. Hvis det er ønsket å se grafer fra resultatene så bruk modellen med plotte-funksjon.

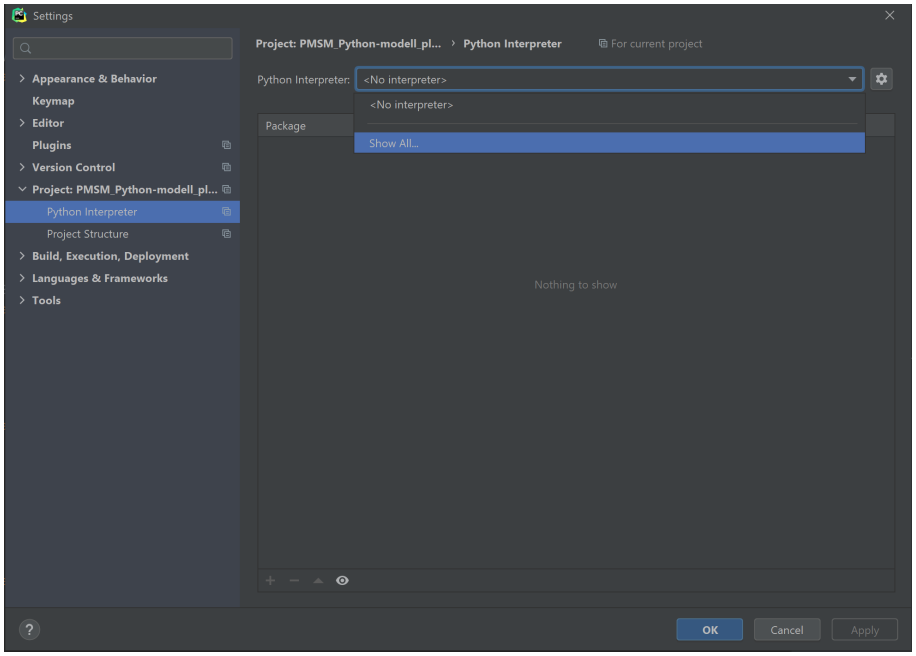
Installasjon
For å kjøre simuleringer med Python-modellen så kreves det en datamaskin med programmet PyCharm installert. For informasjon om installasjon av PyCharm se PyCharm sin hjemmeside: https://www.jetbrains.com/help/pycharm/installation-guide.html. Det er mulig at lignende programmer fungerer, men under prosjektet så har det kun blitt brukt PyCharm.

Python-modellen kommer som en ZIP-fil, som betyr at den må pakkes ut før den kan brukes. Pakk ut filen og plasser den så det er lett å finne den igjen når den skal åpnes i PyCharm. Anbefaler å plassere den i PycharmProjects mappen.

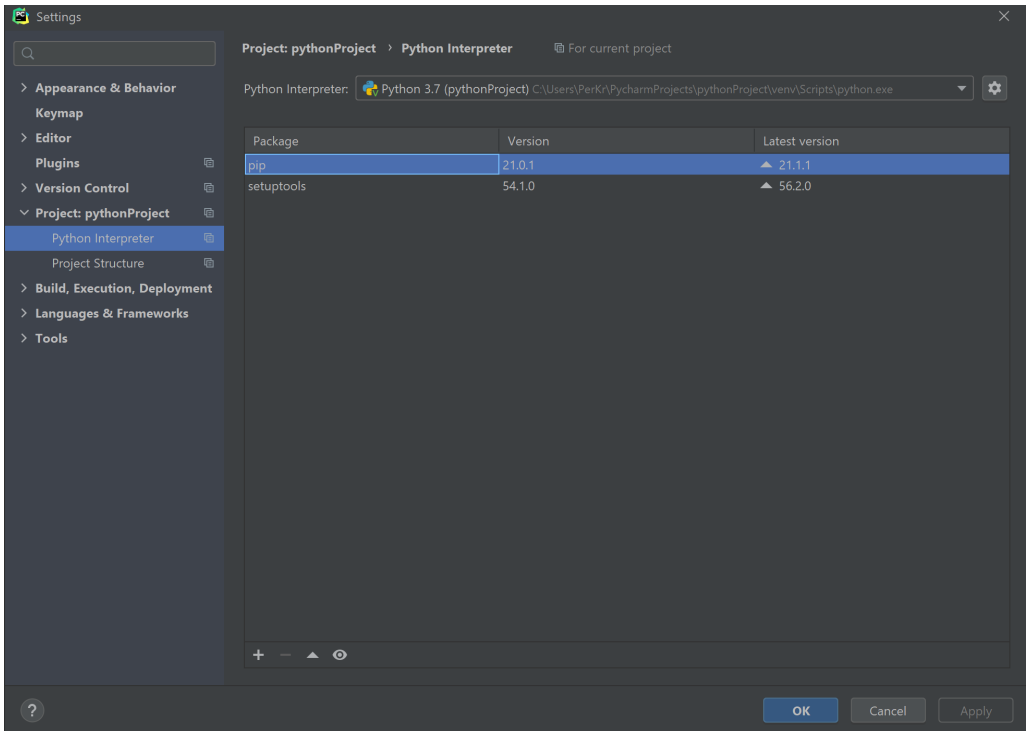
Når PyCharm er installert og ZIP-filen er pakket ut så kan Python-modellen åpnes. Det blir gjort ved å trykke på "file" øverst i venstre hjørne i PyCharm. Trykk på "Open" under file-menyen og velg mappen til Python-modellen og trykk "OK".

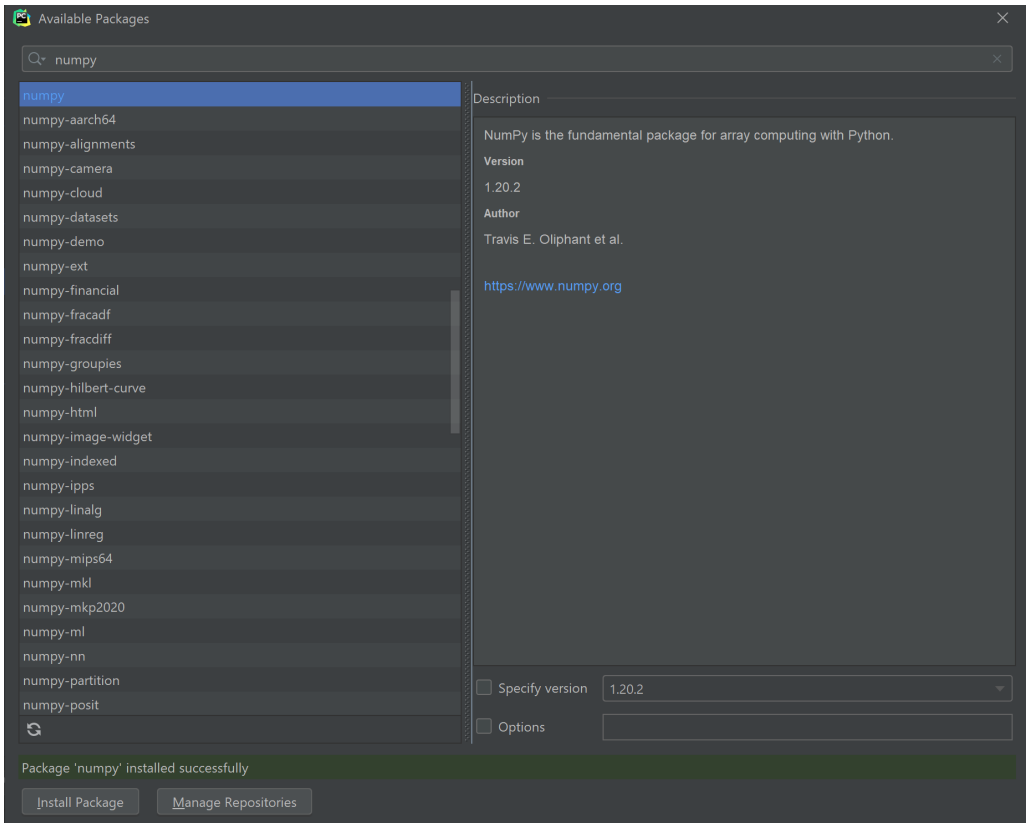


Hvis det er første gang programmet skal bli brukt så kan det være nødvendig å velge en interpreter. For å gjøre det gå under "file" og velg settings. Når settings-vinduet er oppe gå til Project fanen og velg Python Intepreter. Hvis det står ¡No interpreter¿ i menyen øverst så må interpreter bli valgt. Trykk på den menyen og velg show all. Velg nyeste versjon av Python 3 i vinduet som dukker opp og trykk ok.



Når interpreter er valgt så må utvidelsene numpy og matplotlib.pyplot bli installert. Under "file" velg settings. Når settings-vinduet er oppe gå til Project fanen og velg Python Intepreter. Dobbelklikk på "pip" i Python Intepreter-vinduet for å åpne pip. Søk på numpy og matplotlib i pip vinduet, og installer utvidelsene.





Kjøre Simulering
Når alle utvidelsene er installert så kan programmet bli brukt til å kjøre simuleringer. For å kjøre simuleringer så må riktige motor- og simuleringsparametere velges. Motor parameterene som kommer med programmet er fra maskinen "Oswald MFS13.3-6W". Hvis en annen maskin skal bli simulert så må parameterene i filen "PMSM_Parameters.py" endres.

Hastighetsreferanse og betingelser for lastmoment må bli satt før simuleringen kan kjøres. Verdiene blir satt i filen "Initial_Values.py". For å sette en hastighetsreferanse så endres verdien for variabelen "rpm_ref". Enheten for rpm_ref er omdreininger per minutt (rpm). Det er to alternativer for lastmoment i modellen. Konstant lastmoment og steg i lastmoment. For å ha konstant lastmoment så må variabelen "Load_step" bli satt til 0 og lastmomentet settes ved å endre på verdien til variabelen "T_l". For å ha et steg i lastmomentet settes Load_step til 1. Tiden når steget skal skje blir satt med variabelen "LS_time", tiden er gitt i sekunder. Lastmomentet før spranget er variabelen "T_l_0" og lastmomentet etter er variabelen "T_l_1".

Når alle paremeterene er valgt så kan programmet bli startet ved å kjøre filen "main.py". Når filen er blitt kjørt så vil brukeren få beskjed om å velge simuleringstid i sekunder. Velg simuleringstid og trykk enter for å starte simuleringen.

Se Resultater
Det er to alternativer for å se resultatene av simuleringen. Den første er å se verdiene som er blitt skrevet i filen "Results.txt" og den andre er å bruke matplotlib.pyplot for å lage grafer fra simuleringen. Hvis det er ønsket å plotte resultatene, bruk versjonen av modellen med plotte-funksjon. Modellen med plotte-funksjon er kalt "PMSM_Python-modell_plot".

H Testscenarioer

H.1 Test 1: Tomgangstest

Drivsystemet kjører uten eksternt lastmoment. T_{Last} er 0 [Nm]. Drivsystemet skal oppnå og operere på nominelt turtall. Referanse turtall er 2150 [rpm].

Motor paramatere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm

Regulator parametere

Parameter	Python	Simulink
$K_{d,id}$	20.9	1.1906
$K_{i,id}$	1190.6	1190.6
$K_{p,iq}$	20.9	1.3573
$K_{i,iq}$	1357.3	1357.3
$K_{p,\omega}$	200.0	7
$K_{i,\omega}$	25.0	700

Simuleringsparametere

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	0.2	s
T_{Last}	0	Nm
Referansehastighet	2150	rpm

H.2 Test 2: Sprang i last - nominell hastighet

Drivsystemet kjører uten eksternt lastmoment under oppstart. Etter ett sekund skjer det et sprang i lastmoment fra 0 til 189 [Nm] (0.5*T_{nom}). Referanseturtall er 2150 [rpm].

Motor paramatere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm

Regulator parametere

Parameter	Python	Simulink
$K_{d,id}$	20.9	1.1906
$K_{i,id}$	1190.6	1190.6
$K_{p,iq}$	20.9	1.3573
$K_{i,iq}$	1357.3	1357.3
$K_{p,\omega}$	200.0	7
$K_{i,\omega}$	25.0	700

Simuleringsparametere

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	0.4	s
Sprangtid	0.2	s
$T_{Last,1}$	0	Nm
$T_{Last,2}$	$0.5 * T_{nom}$	Nm
Referansehastighet	2150	rpm

H.3 Test 3: Drivsystem uten hastighetskontroll

Drivsystemet bruker konstante i_d og i_q verdier som referanse. Det er ingen hastighetskontroll. Eksternt lastmoment er konstant 189 [Nm] ($0.5*T_{nom}$).

Motor paramatere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm

Regulator parametere

Parameter	Python	Simulink
$K_{d,id}$	70	1.1906
$K_{i,id}$	1	1190.6
$K_{p,iq}$	70	1.3573
$K_{i,iq}$	1	1357.3

Simuleringsparametere

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	1.0	s
T_{Last}	$0.5 * T_{nom}$	Nm
$i_{d,ref}$	-5	A
$i_{q,ref}$	140	A

H.4 Test 4: Stort sprang i last - høy hastighet

Drivsystemet skal oppnå og operere med turtallreferanse som er større enn nominelt turtall. Turtallreferansen er satt til 3000 [rpm]. Ved oppstart er lastmoment 0 [Nm]. Etter ett sekund skjer det et sprang i lastmoment fra 0 til 340.2 [Nm] (0.9*T_{nom}).

Motor paramatere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm

Regulator parametere

Parameter	Python	Simulink
$K_{d,id}$	20.9	1.1906
$K_{i,id}$	1190.6	1190.6
$K_{p,iq}$	20.9	1.3573
$K_{i,iq}$	1357.3	1357.3
$K_{p,\omega}$	200.0	7
$K_{i,\omega}$	25.0	700

Simuleringsparametere

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	0.4	s
Sprangtid	0.2	s
$T_{Last,1}$	0	Nm
$T_{Last,2}$	$0.9 * T_{nom}$	Nm
Referansehastighet	3000	rpm

H.5 Test 5: Varierende turtallsreferanse

Drivsystemet bruker et sinussignal som turtallreferanse. Firkantsignalet har 1000 [rpm] som amplitude og frekvens 0.5 [Hz]. Eksternt lastmoment er 0 [Nm].

Motor paramatere

Variabel	Verdi	Enhet
p	3	-
R_s	0.0209	Ω
L_d	0.0012	H
L_q	0.0014	H
J	0.07	$Kg * m^2$
B_{Py}	0	$Kg * m^2/s$
B_{Sim}	0.0012	$Kg * m^2/s$
λ_{pm}	0.4479	Wb
I_{nom}	140	A
T_{nom}	378	Nm

Regulator parametere

Parameter	Python	Simulink
$K_{d,id}$	20.9	1.1906
$K_{i,id}$	1190.6	1190.6
$K_{p,iq}$	20.9	1.3573
$K_{i,iq}$	1357.3	1357.3
$K_{p,\omega}$	200.0	7
$K_{i,\omega}$	25.0	700

Simuleringsparametere

Variabel	Verdi	Enhet
t_{sample}	$2.5 * 10^{-5}$	s
Simuleringstid	4	s
T_{Last}	0	Nm

Turtallsreferanse		
Bølgeform	Sinus	-
Faseforskyvning	0	-
Amplitude	1000	rpm
Frekvenes	0.5	Hz

I Poster

Modellering og simulering av drivsystem for permanentmagnet synkronmaskin

Per Kristian Engen, Sivert Heidsve og Wilhelm Bergesen

NTNU, Trondheim

Bakgrunn

Transport står for omtrent en tredjedel av totalt klimagassutslipp i Norge, der sjøtransporten utgjør 19% av transportutslippene. Miljøvennlig skipsfart er et av fem prioriterte innsatsområder innenfor dagens klimapolitikk. Med et ønske om et grønt skifte innenfor sjøtransport, utvikler Haf Power Solutions AS en fullelektrisk turistbåt. Båten skal drives av to permanentmagnet synkronmotorer. Et viktig instrument for videreutvikling av drivsystemet er en digital modell av motorene. Ved å bruke den kan en simulere driftssystemet med svært få kostnader.

Oppgaven

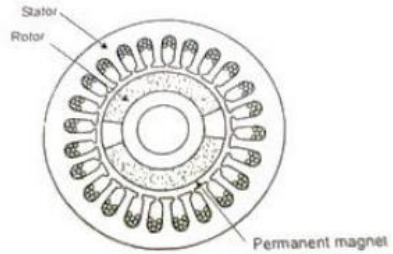
Oppgaven går ut på å lage en digital modell av drivsystemet til en permanentmagnet synkronmaskin (PMSM). Modellen blir grunnlaget for simulering av drivsystemet.

Turistbåten Brim Explorer

Hva er en PMSM

En PMSM er en vekselstrøm synkronmaskin som drives ved hjelp av permanent magneter (PM). Den er bygd opp av en rotor med PM'er som er omsluttet av en stator med viklinger på innsiden. Ved å forsyne statorviklingene med 3-fase vekselstrøm skapes et roterende magnetfelt, som driver rotor i maskinen.
PMSM'er har flere fordeler:
-Høyt dreiemoment over et stort hastighetsområde
-Høy energiffektivitet
-Stillegående
-Lavt treghetsmoment
-Enkel å kontrollere

PMSM Armatur



Oswald MFS 13.3-6. Utgangspunktet for modellene

Metode

De digitale modellene er konstruert, og simulert i Simulink og Python. Simulink-modellen er utviklet for verifisering av Python-modellen.

Digital modell

Drivsystemet i modellene bruker feltorientert kontroll i roterende dq-referanseramme. Drivsystemet består av en hastighetsregulator for kontroll av turtall, og to strømregulatorer som kontrollerer inngangsspenning til motoren. For beregning av referansestrømmer bruker modellene maksimum dreiemoment per ampere.

Matematisk modell

Ligninger som beskriver PMSM:

$$\frac{di_d}{dt} = \frac{v_d}{L_d} - \frac{R_s i_d}{L_d} + \frac{L_q}{L_d} p\omega_m i_q$$

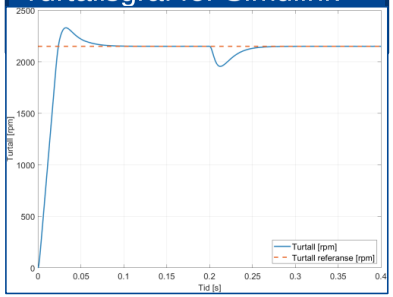
$$\frac{di_q}{dt} = \frac{v_q}{L_q} - \frac{R_s i_q}{L_q} - \frac{L_d}{L_q} p\omega_m i_d - \frac{1}{L_q} p\omega_m \lambda_{pm}$$

$$T_e = 1.5p(\lambda_{pm} i_q + (L_d - L_q) i_d i_q)$$

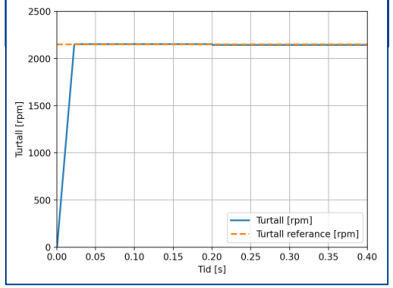
$$\frac{d\omega_m}{dt} = \frac{1}{J}(T_e - T_{Last} - B\omega_m)$$

Resultat:

Turtallsgraf for Simulink



Turtallsgraf for Python



Konklusjon

Det er blitt laget en forenklet modell av drivsystemet til en PMSM. Den kan simulere hastighet, dreiemoment, statorstrøm og statorspenning. Modellen mangler diverse funksjoner som kan videreutvikles som fremtidig arbeid

NTNU | Kunnskap for en bedre verden

HAF POWER SOLUTIONS

Modellering og simulering av drivsystem for permanentmagnet synkronmaskin

NTNU

Kunnskap for en bedre verden