TAMPEREEN TEKNILLINEN YLIOPISTO

HENRI VÄLIMÄKI

SIIRTOLIIKKEEN SÄÄDÖN TOTEUTUS VARASTOROBOTISSA

Diplomityö

Tarkastaja: Karri Palovuori
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-
neuvoston
kokouksessa 08.05.2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
Välimäki, Henri : Siirtoliikkeen säädön toteutus varastorobotissa
Diplomityö, 46 sivua, 6 liitesivua
Toukokuu 2013
Pääaine: Sulautetut järjestelmät
Tarkastajat: Karri Palovuori
Avainsanat: PID, liikkeensäätö, sulautettu järjestelmä

Varastorobotin liikkeeltä vaaditaan tasaista nopeutta ja tarkkaa paikoitusta. Työn tavoitteena on suunnitella ja toteuttaa siirtoliikeen säädin. Liike on toteutettu kahdella paikkatakaisinkytketyllä sähkömoottorilla.

Työ jakautuu siten, että ensin tutustutaan säätöteoriaan ja -tekniikkaan ja erityisesti PID säätöön. Simulaattorissa luodaan suunnitelma siitä minkälainen säädin tullaan toteuttamaan. Lopullinen kehitys tehdään koeajolaitteistolla joka koostuu koeajoradasta ja tiedonkeruujärjestelmästä.

Työn tuloksena on toimiva liikkeensäädin kyseiselle siirtoliikkeelle. Säätimessä on lisäksi seisontajarrujen ohjaus sekä luistonestojärjestelmä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Electrical Engineering
Välimäki, Henri : Implementation of traversing motion controller in warehouse robot
Master of Science Thesis, 46 pages, 6 Appendix pages
May 2013
Major: Embedded systems
Examiner: Karri Palovuori
Keywords: PID, motion control, embedded system

Motion control of a warehouse robot has to be steady in speed and precise in positioning. The goal is to design and implement controller for robots traversal motion. Motion is carried out by two electric motors with position feedback.
The work is organized in such a way that you first learn about the control theory and technology, and in particular the PID control. Then simulator is used to create an outline of what kind of control will be implemented. Major part of development is then carried out with test track and the data acquisition system.
As a result for this thesis there is an effective motion control to the traversing motion. A simple traction control is also implemented.

ALKUSANAT

Työ on tehty Konecranes Oy:ssä. Kiitokset Justus Dahlénille ja Tero Kultaselle neuvoista ja kannustuksesta. Janne Salomäelle kiitos säädinrakenteiden ja Simulinkin perusteiden opastamisesta. Jani Vähä-pesolalle kiitos ohjelmointiin liittyvien ongelmien kanssa auttamisesta.

SISÄLLYS

1. Johdanto 1
2. Säätöteoria ja -tekniikka 2
2.1 Takaisinkytketty järjestelmä 2
2.1.1 Stabiilisuus 3
2.1.2 Järjestelmän suorituskyvyn mittarit 3
2.2 PID-säädin 6
2.2.1 Säätimen esitysmuodot 6
2.2.2 Integraattori 8
2.2.3 Derivaattori 9
2.2.4 Myötäkytkentä 9
2.3 Moniakselikäyttö 10
2.3.1 Isäntä-orja ohjaus 10
2.3.2 Keskitetty rinnanohjaus 11
2.3.3 Hajautettu rinnanohjaus 11
2.4 Virittäminen 13
2.4.1 Ziegler ja Nichols 13
2.4.2 Rele-säätöön perustuva mittaus 15
2.4.3 Iteratiivinen parametrien asettelu 15
3. Diskreetti säädin 17
3.1 Säätösilmukka 17
3.2 Tosiaikavaatimukset 17
3.3 Näytteistys ja uudelleengenereointi 18
3.4 Sisäinen esitystapa 19
3.5 Jatkuva-aikaisen säätimen tietokonetoteutus 19
3.5.1 Eteenpäin differentointi 19
3.5.2 Taaksepäin differentointi 20
4. Ajoliikkeelle asetetut vaatimukset ja toteutusympäristö 21
4.1 Ajoliikkeen vaatimukset 21
4.2 Askelmoottori 22
4.3 Ajoliikkeen mekaniikka 23
4.4 Ohjelmiston alusta 25
4.4.1 Alteran FPGA 25
4.4.2 Moottorinohjausrajapinta 25
5. Ajoliikkeen säädin 28
5.1 Säätimen rakenne 28
5.2 Simuloinnit 28

5.2.1 Nopeussäädin 29
5.2.2 Paikkasäädin 30
5.2.3 Myötäkytkentä (feedforward) 31
5.3 Koeajo 32
5.4 Luistonesto 37
6. Tulokset ja niiden tarkastelu 44
6.1 Koeajojärjestelmä 44
6.2 Säädinparametrit 44
6.3 Luistonesto 45
7. Johtopäätokset 46
Lähteet 47
A.Laitteen teknisiä tietoja 48
B.Säätimen ohjelmakoodi 49
B.1 Paikkasäädin 49
B.2 Nopeussäädin 50
B.3 Momenttisäädin 50
B.4 Liikkeen valmistumisen tarkkailu 51
B.5 Orjamoottorin momentin muodostus 51
B.6 Relemittaus 51

TERMIT JA SYMBOLIT

Akt	Aktuaattori, toimilaite
C	Controller, säädin
e	Erosuure
J	hitausmassa
u	Ohjaussuure
r	Sisääntulo, referenssi
y	Ulostulo
k	Kerroin
P	Prosessi
M	Mittaus
$\hat{}$	Mitattu arvo
*	Ohjearvo
d	Häiriö
n	Kohina
T_d	Derivointiaika
T_i	Integrointiaika
G	Vahvistus
G(s)	Siirtofunktio
ϕ	Kulma
ω	Kulmanopeus
ω_{ff}	Nopeuden myötäkytkentä
$\dot{\omega}_{ff}$	Kiihtyvyyden myötäkytkentä
T	Vääntömomentti

Antiwindup	Integraattorin kasautumisen esto
Avalon	Alteran FPGA:n sisäinen väylä
FPGA	Field Programmable Gate Array, Ohjelmoitava logiikka
IAE	Integrated Absolute Error
ISE	Integrated Squared Error
ITAE	Integrated Time Absolute Error
ITSE	Integrated Time Squared Error
NIOS	Alteran FPGA:lle ohjelmoitava prosessori
PID	Proportional Integrating Derivating
RISC	Reduced Instruction Set Computer
Windup	Integraattorin kasautuminen

1. JOHDANTO

Varastoautomaatilta vaaditaan täsmällistä, turvallista ja nopeaa paketinkuljetusta. Liikkeiden ohjaukselta vaaditaan tarkka paikoitus ja varma toiminta, vaikka kuorma vaihtelee ja liikkuu sukkulan sisällä.

Diplomityössä suunnitellaan ja toteutetaan kiskolla kulkevan varastorobotin siirtoliikkeen säätö. Siirtoliike on toteutettu kahdella vääntömomenttiohjatulla askelmoottorilla, jotka liikuttavat vaunua kumipyörien avulla. Toteutuksessa pitää ottaa mahdollinen pyörien luistaminen ja kaarreajo huomioon. Siirtoliikkeen maksiminopeus on turvallisuussyistä rajoitettu, ja ajonopeuden pitää olla mahdollisimman lähellä tätä nopeutta, sitä kuitenkaan missään tilanteessa ylittämättä. Säädinarkkitehtuuri kehitetään simulaattorin avulla, ja valmiin liikkeenohjauksen toimivuus varmistetaan mittauksin.

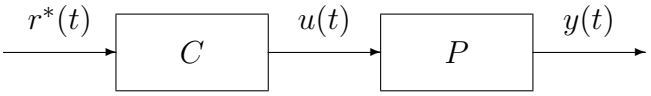
Työ jakaantuu siten, että kappaleessa 2 käsitellään säätöteoriaa, säädinarkkitehtuureja ja säätimen virittämistä. Kappaleessa 3 tarkastellaan säätimen diskreettiä toteutusta. Kappaleessa 4 käsitellään työnä olevan ajoliikkeen vaatimuksia ja toteutusympäristöä. Kappaleessa 5 kehitetään säädin simulaattorin ja koeajolaitteiston avulla. Kappaleessa 6 esitellään saadut tulokset ja kappaleessa 7 vedetään yhteen johtopäätökset.

2. SÄÄTÖTEORIA JA -TEKNIIKKA

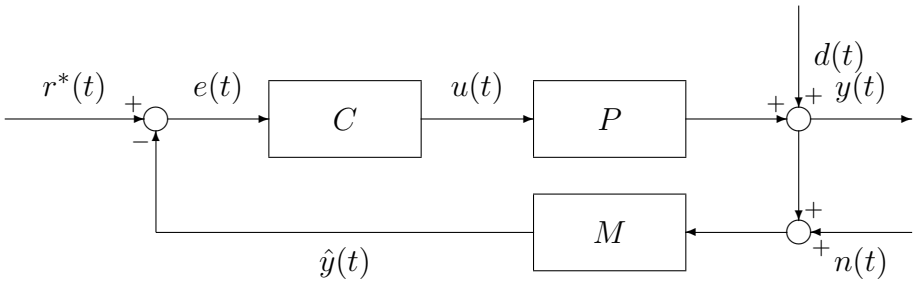
Säätötekniikka on järjestelmän rakenteiden analysointia ja ohjausjärjestelmien suunnittelua siten, että ohjattava prosessi saadaan toteuttamaan sille asetetut tehtävät. Oleellisena osana säädetyssä järjestelmässä on takaisinkytkentä, joka tarkoittaa ulostulon tai ulostulojen mittausta ja tämän tiedon hyväksikäyttöä säätimessä.

2.1 Takaisinkytketty järjestelmä

Kuvan 2.1 järjestelmässä säädin C *(controller)* vaikuttaa ohjaussuureen $u(t)$ avulla prosessin P *(process)* ulostuloon $y(t)$ ilman takaisinkytkentää, eli kyseessä on avoimen silmukan ohjaus. Jotta ulostulo seuraisi ohjausta $r^*(t)$, täytyy prosessi olla hyvin tunnettu ja tuntemattomia ulkopuolisia häiriöitä ei saa olla. Näin ei kuitenkaan välttämättä ole.



Kuva 2.1: Avoimen silmukan ohjausjärjestelmä.



Kuva 2.2: Takaisinkytketty säätöjärjestelmä.

Takaisinkytketty järjestelmä on kuvattu kuvassa 2.2, jossa on lisänä edelliseen takaisinkytkentähaara ja mittaus M *(measurement)*, jonka ulostulona on ulostulon

approksimaatio $\hat{y}(t)$. Ohjauksen ja mitatun ulostulon erotuksesta lasketaan säätimelle menevä erosuure $e(t)$. Säätimelle ei mene ohjaussignaalia, vaan ainoastaan mainittu ohjauksen ja mitatun ulostulon erotus. Nyt säätimen tehtävänä onkin minimoida kyseinen $e(t)$. Lisänä edelliseen avoimen silmukan järjestelmään on mittaus. Mittaus ei yleensä ole täydellinen, vaan sisältää mittavirhettä, viivettä ja kohinaa. Mittauksen virheet on kuvattu $n(t)$:llä.

Takaisinkytkennällä saadaan vähennettyä kuormituksen muuttumisen ja prosessin vaihtelun vaikutusta. Takaisinkytkennällä saadaan myös linearisoitua epälineaarinen järjestelmä ja stabiloitua epästabiili järjestelmä. Takaisinkytkennällä on myös mahdollista tehdä stabiilista järjestelmästä epästabiili.

2.1.1 Stabiilisuus

Takaisinkytketyn järjestelmän stabiilisuus riippuu säätösilmukan viiveestä ja vahvistuksesta. Kun silmukassa on vahvistus yli yhden, ja samanaikaisesti viive aiheuttaa sisäänmenon ja ulostulon välille vaihe-eroa yli 180 astetta, tulee negatiivisesta takaisinkytkennästä positiivinen takaisinkytkentä. [3] Tällöin järjestelmä alkaa oskilloimaan, tai ajautuu saturaatioon.

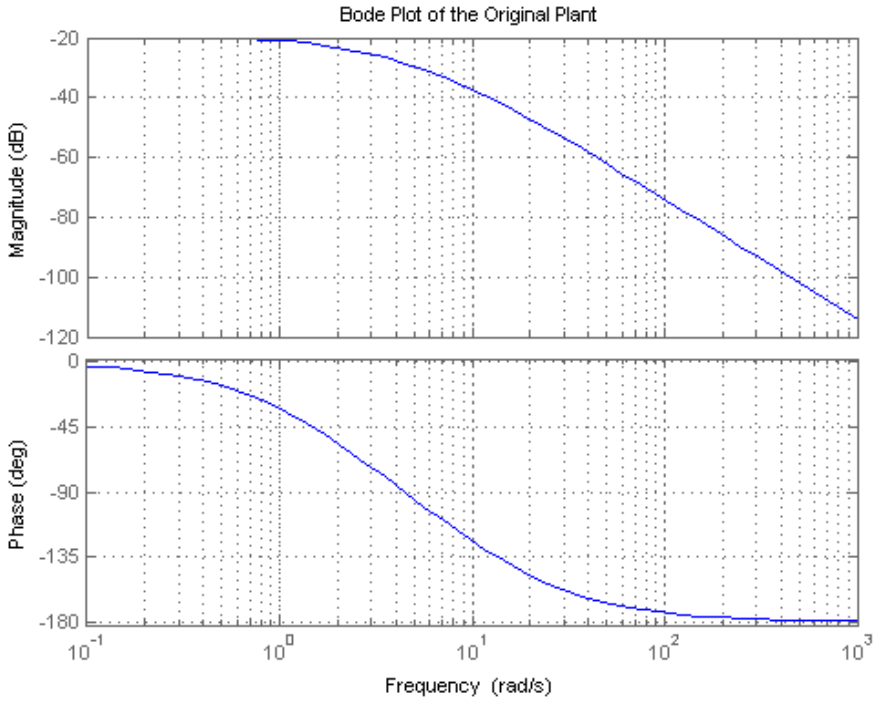
Vahvistus ja viive ovat tyypillisesti taajuuden funktioita. Järjestelmän taajuusvasteen kuvaajaa, jossa on kuvattuna samalla taajuusakselilla järjestelmän vahvistus ja vaihe, kutsutaan Boden kuvaajaksi, kuva 2.3. Jotta järjestelmä sietäisi komponenttien ominaisuuksien vaihtelemista, täytyy vahvistuksessa ja viiveessä olla marginaalia stabiilisuusehdossa. Vahvistusvara tarkoittaa määrää jonka vahvistus voi lisääntyä ennenkuin stabiilisuus menetetään. Tämä nähdään Boden kuvaajasta sen taajuuden kohdalta, jossa vaihesiirto on 180 astetta. Tämä on vahvistuskuvaajan etäisyys yksikkövahvistuksesta eli 0dB:stä. Vastaavasti vaihevara on yksikkövahvistuksen taajuudella vaihekuvaajan etäisyys 180 asteen vaihesiirrosta.

2.1.2 Järjestelmän suorituskyvyn mittarit

Järjestelmän ominaisuuksia saa helposti selville suorittamalla sille askelvastekokeen. Kokeessa syötetään ohjeeksi askelmainen muutos ja tallennetaan siitä aiheutunut ulostulo kuvaajaan, kuten kuvassa 2.4 on tehty. Askelvastekokeen huono puoli on se että, se tehdään avoimelle säätösilmukalle. Tällöin järjestelmä ei ole hallittavissa ja siten askelvastekokeen teko voi olla mahdotonta.

Askelvasteen ominaisuuksia voi käyttää järjestelmän vaatimuksina ja kokeella voidaan vaatimuksien täyttyminen todeta. Vaatimukset ovat järjestelmäkohtaisia ja joitakin liikkeenohjaukseen soveltuvia tunnuslukuja on lueteltu alla.

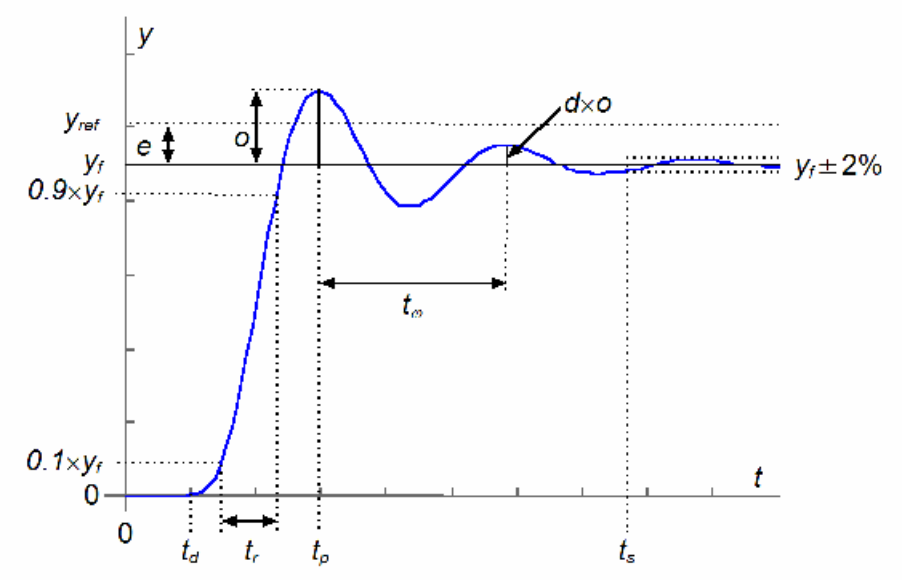
- t_r - nousuaika on aika jossa ulostulo nousee 10%:sta 90%:n



Kuva 2.3: Boden kuvaaja.

- M_p - maksimiylitys, prosentteina
- t_p - huippuaika, aika jollon maksimiylitys tapahtuu
- t_s - asettumisaika, aika jolloin vaste on halutuissa rajoissa (2% tai 5%)
- d - ylitysten vaimenemissuhde

Vasteen virhettä ohjaukseen voidaan myös vertailla laskemalla virheen summa jonkun tunnetun ajon ajalta. Virheen summaamista voidaan käyttää säätimen suunnitteluparametrina tai valmiin säätimen suorituskyvyn mittarina. Eri tavoin painottamalla saadaan mitattua tilanne, joka vastaa säädön vaatimuksia. Yhtälössä (2.1) integroidaan virheen itseisarvoa jonkin halutun ajan yli. Näin saadaan mitattua kuinka hyvin vaste seuraa ohjearvoa. Mittausta voidaan painottaa ottamalla virheestä toinen potenssi kuten yhtälössä (2.2). Näin suuret poikkeamat tulevat paremmin huomioiduiksi. Jos pitkän ajan virheen minimointi on oleellisempaa voidaan edellisiä yhtälöitä painottaa ajan suhteen. Näin on tehty yhtälöissä (2.3) ja (2.4).



Kuva 2.4: Askelvaste.

$$IAE = \int |e(t)|dt \tag{2.1}$$

$$ISE = \int e^2(t)dt \tag{2.2}$$

$$ITAE = \int t \cdot |e(t)|dt \tag{2.3}$$

$$ITSE = \int t \cdot e^2(t)dt \tag{2.4}$$

Huonona puolena edellä mainitussa tavassa mitata järjestelmän suorituskykyä on se, että testiajon täytyy olla joka kerta samanlainen. Mittauksista saatavia tuloksia voi siten vertailla vain keskenään. Lisäksi testiajon valinnalla on merkitystä mittausten hyödyllisyydelle lopullisen käytön kannalta.

2.2 PID-säädin

PID-säädin on yleinen teollisuudessa käytössä oleva säädin. PID-säädin on rakenteeltaan suhteellisen yksinkertainen ja helposti ymmärrettävä sekä sen virittäminen on helpohkoa. Säädin koostuu kolmesta lohkosta; suhteellinen, derivoiva ja integroiva säätölohko. Riippuen säädettävästä prosessista, voidaan lohkoja käyttää kaikkia yhdessä tai jättää esimerkiksi derivoiva säätö pois. Tässä kappaleessa käydään peruslohkojen toimintaperiaatteet läpi, kuten PID-säätimen eri muodot ja esitystavatkin.

Lopuksi tarkastellaan minkälaisia muutoksia koulukirjamuotoon on tehtävä, jotta säädin on käyttökelpoinen.

2.2.1 Säätimen esitysmuodot

PID-säätimen koulukirjamuoto on esitetty yhtälössä (2.5) ja sen siirtofunktiomuoto yhtälössä (2.6).

Yhtälöissä $e(t)$ on asetuspisteen ja takaisinkytkennän erotus ja $u(t)$ on säätimeltä lähtevä ohjaussuure.

Suhteellisen säädön tuottama ohjaussuure on K kertaa erosuure $e(t)$. Suhdesäädölle on ominaista pysyvän tilan virhe.

Integroiva säätö poistaa pysyvän tilan virheen siten, että integraattori summaa virhetermiä ajan yli ja pysyvän tilan virhe ennen pitkää alkaa vaikuttamaan ohjaussuureeseen siten, että virhe menee nollaan. Integrointiaika, T_i, on aika jossa inegrointilohko tuottaa saman ulostulon kuin suhteellinen säätö. Mitä suurempi T_i on sitä vähemmän integrointilohko vaikuttaa säätöön. Pienikin integrointitermin mukana olo kuitenkin poistaa pysyvän tilan virheen kokonaan.

Derivoiva osuus säädöstä ennustaa lineaarisella extrapolaatiolla virheen muutosta. Derivointiaika T_d on aika kuinka pitkälle tulevaisuuteen ennustus tehdään. Ennustavasta luonteesta johtuen derivoiva lohko reagoi virheisiin nopeasti ja siten lisää stabiilisuutta ja mahdollistaa suuremman vahvistuksen käytön. Derivaattori kuitenkin vahvistaa enemmän korkeita taajuuksia, joten mittasignaalin sisältämät korkeataajuiset komponentit voivat olla ongelma.

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{2.5}$$

$$G(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{2.6}$$

Edellisissä yhtälöissä on selkeä yhteys fysikaaliseen maailmaan aikavakioiden kautta. Voidaan kuitenkin puhua myös eri lohkojen vahvistuksista ja erityisesti tietokonetoteutuksessa ei säätösilmukassa kannata tehdä ylimääräisiä jako- ja kertolaskuja. Yhtälössä (2.7) on eri lohkoilla vahvistukset ja eri esitystapojen välillä voidaan vaihdella yhtälöiden (2.8), (2.9) ja (2.10) avulla.

$$G(s) = k + \frac{k_i}{s} + sk_d \tag{2.7}$$

$$k = K \tag{2.8}$$

$$k_i = \frac{K}{T_i} \tag{2.9}$$

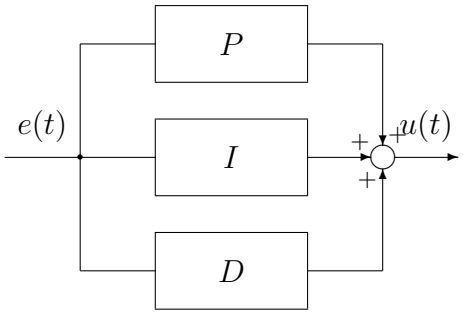
$$k_d = KT_d \tag{2.10}$$

Kuvassa 2.5 on kuvattu edellä mainittu rinnakkaismuotoinen PID-säädin. Tätä muotoa Karl Åström [12] kutsuu vuorovaikutuksettomaksi muodoksi (non-interacting form) sillä lohkojen kertoimien välillä ei ole suoraa vaikutusta. Historiallisista syistä on olemassa myös sarjamuotoinen tai vuorovaikutuksellinen (interacting) muoto jossa kertoimilla on on suora vaikutus toisiinsa. Tietokonetoteutuksissa ei vuorovaikutuksesta ei ole mitään hyötyä, joten puhdas rinnakkainen muoto, jossa eri lohkoilla on vahvistuskertoimet, lienee käyttökelpoisin ja tässä työssä käytetään vain sitä.

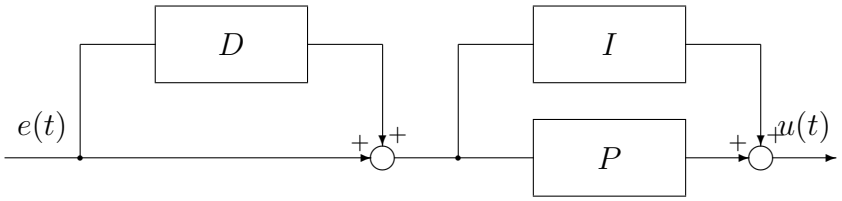
Vuorovaikuteksellista muotoa kuitenkin voi tulla vastaan kirjallisuudessa. Muoto on kuvattu kuvassa 2.6 ja se on siirtofunktiomuodossa yhtälössä (2.11).

$$G(s) = K\left(1 + \frac{1}{sT_i}\right)(1 + sT_d) \tag{2.11}$$

Sarjamuotoinen säädin voidaan aina muuntaa rinnakkaismuotoiseksi yhtälöiden



Kuva 2.5: Rinnakkaismuotoinen PID-säädin.



Kuva 2.6: Sarjamuotoinen PID-säädin.

(2.12) avulla. Yhtälön oikeanpuoleiset termit ovat sarjamuotoisen säätimen kertoimia ja vasemmanpuoleiset rinnakkaismuotoisen. Jos $T_i \geq 4T_d$ voidaan muunnos tehdä myös toisinpäin, mutta sitä ei esitetä tässä.

$$\begin{aligned} K &= K\frac{T_i + T_d}{T_i} \\ T_i &= T_i + T_d \\ T_d &= \frac{T_i T_d}{T_i + T_d} \end{aligned} \tag{2.12}$$

2.2.2 Integraattori

Tilannetta jossa toimilaite ei pysty toteuttamaan pyydettyä ohjetta, ja säätimen integroiva lohko kasvaa rajatta, kutsutaan inegraattorin windupiksi eli kasautumiseksi. Windup-tilanteen jälkeen kun toimilaite taas on toiminnassa, voi säätimen integraattoriin olla kasautunut suurikin summa. Tämä summa purkautuu vain erosuureella, joka on erimerkkinen kuin windup-tilanteen erosuure oli. Järjestelmällä voi siten kestää huomattavan kauan palautua windup-tilanteesta.

Edellä mainuttu ongelma saadaan poistettua siten, että havainnoidaan että pystyykö toimilaite toteuttamaan sille annettua komentoa ja sen mukaan vaikutetaan integraattorilohkon toimintaan. Eräs tapa on estää integroiminen suurilla integraattorin tai erosuureen arvoilla. Toinen tapa on mitata toimilaitteen ulostuloa ja verrata sitä ohjeeseen ja reagoida tämän mukaan. Jos ulostuloa ei pystytä mittaamaan, mutta toimilaitteesta on tiedossa tarkka malli, voidaan saturaatio päätellä sen avulla. Yksinkertaisimmillaan ohjaussuureen menemisestä minimiin tai maksimiin voidaan todeta järjestelmän olevan saturaatiossa. Toimilaite voi tosin olla saturaatiossa vaikka sen ohjaussuure ei olisikaan. Saadun saturaatiotiedon mukaan integraattorin kasvaminen voidaan pysäyttää tai hidastaa. Jossain tilanteessa integraattori voidaan myös nollata. [5]

2.2.3 Derivaattori

Derivaattori parantaa järjestelmän stabiilisuutta ja reagointia häiriöihin. Ongelmana derivaattorin kanssa on askelmaiset asetuspisteen muutokset, jotka aiheuttavat derivaattorin lähtöön voimakkaan piikin. Toinen huono piirre derivaattorissa on korkeiden taajuuksien vahvistaminen. Käytännön derivaattori täytyykin rakentaa siten että edellämainitut ongelmat ratkaistaan tavalla tai toisella. Derivaattorille tuleva signaalin muutosnopeus voidaan rajoittaa tai derivaattorin tulo voidaan muodostaa painotetusti takaisinkytkennästä ja erosuureesta. Korkeiden taajuuksien korostuminen voidaan estää myös rajoittamalla derivaattorin korkeiden taajuuksien vahvistus.

2.2.4 Myötäkytkentä

Myötäkytkennällä *(feedforward)* tarkoitetaan topologiaa, jossa takaisinkytkennän rinnalla on toinen säätörakenne, joka prosessin mallin ja tiedetyn häiriön avulla vaikuttaa ohjaussuureeseen. Myötäkytkentää voidaan käyttää jos prosessin häiriö tiedetään mittaamalla tai laskemalla. Myötäkytkentä ei vaikuta järjestelmän stabiilisuuteen. Ohjaussignaalin ja mallin avulla tuotettun myötäkytkennän avulla saadaan erotettua takaisinkytkentä reagoimaan kuormitushäiriöihin ja myötäkytkentä reagoimaan ohjaussignaalin muutoksiin. Laskennallisen myötäkytkennän täysimääräinen käyttö kuitenkin lisää ylitystä, joten on turvallisempaa käyttää hieman pienempää arvoa, esimerkiksi 60% laskennallisesta maksimivahvistuksesta. [4]

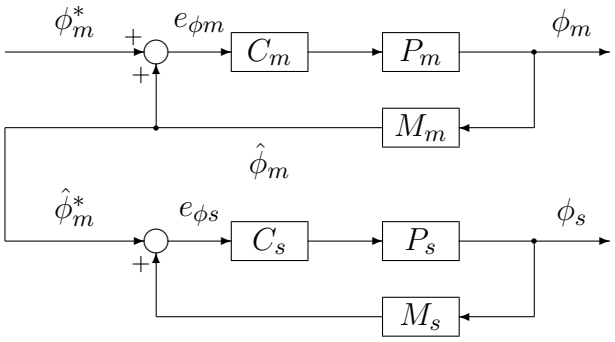
2.3 Moniakselikäyttö

Moniakselikäytössä useampi säädin ja aktuaattori suorittavat liikkeen yhdessä. Liikerata muodostuu usean akselin yhteisestä liikkeestä. Liikerata voi olla moniulotteinen ja mielivaltainen esimerkiksi robotin ja moniakselisen työstökoneen tapauksessa. Yksinkertaisimmillaan liikerata on jokaisella aktuaattorilla sama. Tällainen tilanne on esimerkiksi kun jonkun vaunun pyöriä ohjataan säätimillä ja vaunu kulkee suoraan. Liikkeen vaatimukset asettavat vaatimukset käytettävälle topologialle.

Usean akselin liikkeessä vaaditaan laskentatehoa joko liikkeitä ohjaavalta keskusyksiköltä kuten PC:ltä tai laskentakykyä voi olla hajautettuna ohjaimiin. Lisäksi topologiasta riippuen viestejä välittävälle väylälle tulee lisää nopeus- ja tosiaika vaatimuksia. Viestien välitys laitteiden välillä voi olla analogista virta- tai jänniteviestiä tai laitteet voivat olla liitetty digitaaliseen väylään. Tässä oletetaan että kaikki viestien välitys on digitaalista. Usean toimilaitteen yhteisessä liikkeessä järjestelmällä täytyy olla sama käsitys ajasta eli liikkeet ovat synkronisia. Liikkeiden aloitushetki voi olla synkronoitu tai toimilaitteet voivat synkronoidun väylän avulla pitää aikatasonsa identtisinä.

2.3.1 Isäntä-orja ohjaus

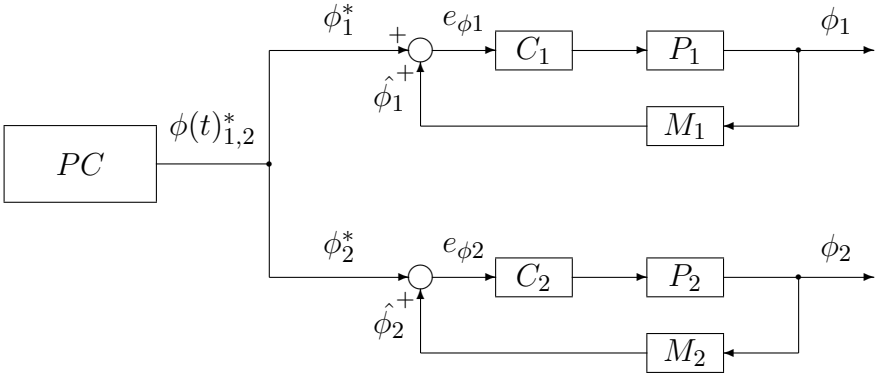
Yksinkertaisimmillaan usean akselin yhteinen liike on niin sanottu isäntä-orja (master-slave) ajo. Siinä yksi akseli on isäntä ja muut akselit käyttävät omana ohjeenaan isännän mitattua paikkaa, nopeutta, momenttia tai jotain näiden funktiota. Orjan ohje on kuitenkin aina mitattu isäntälaitteelta ja siirretty orjalaitteelle väylää pitkin. Näin orjalaite seuraa isännän todellista paikkaa. Huonona puolena on se, että orjan ohjeessa on mukana mittauksen viive, virheet, epätarkkuus ja kohina. Erikoistapauksena isäntä voi olla myös täysin virtuaalinen eli ohjelmallinen simulaatio, jonka tietoja orjalaitteet käyttävät omassa ajossaan. Ohjaus on siten keskitetty ja orjat toimivat kaikki rinnakkain. Paikkaa seuraava isäntä-orja kytkentä on kuvattu periaatetasolla kuvassa 2.7. Isännän paikan mittauksen, M_m, häiriöt ja virheet yhdessä säätimiä yhdistävän väylän kanssa näkyvät suoraan orjan ohjeessa, $\phi^*_{\hat{m}}$.[6]



Kuva 2.7: Isäntä-orja ohjauksen periaate.

2.3.2 Keskitetty rinnanohjaus

Liikkeenohjaus voi olla täysin keskitettyä ja toimilaitteet suorittavat saamaansa ohjetta parhaan kykynsä mukaan. Ohje voi olla rataohje eli paikka joka päivittyy joka säätökierroksella tai yksinkertaisimmillaan pelkkä momenttiohje. Yksittäisen akselin virhe näkyy siten että kokonaisuuteen tulee ratavirhettä.[6] Ratavirhe voidaan minimoida jos ohjaimilta saadaan takaisinkytkentätietoa ohjeita tuottavalle korkeamman tason laitteelle, kuten tietokoneelle. Tämä lisää kuitenkin tiedonsiirtoväylän fyysistä monimutkaisuutta ja nopeusvaatimuksia.



Kuva 2.8: Keskitetyn rinnanohjauksen periaate.

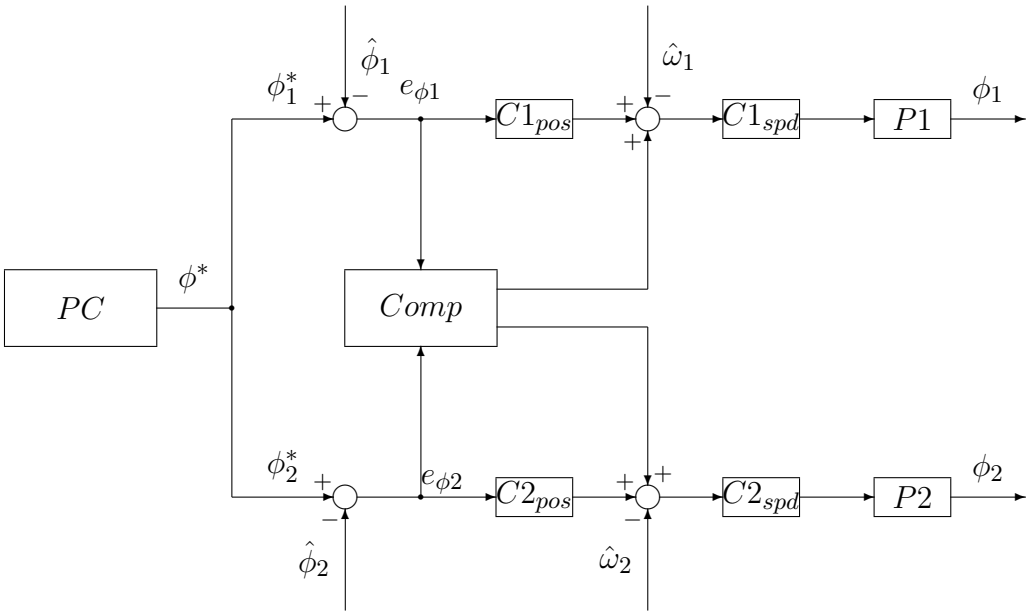
2.3.3 Hajautettu rinnanohjaus

Hajautetussa liikkeenohjauksessa ohjaimet vastaavat itse enemmän liikkeiden suorittamisesta ja koordinoinnista. Moottorinohjaimet laskevat itse ratansa ja synkronoivat liikkeensä toistensa suhteen. Ylemmän tason tietokone antaa moottoriohjaimille liiketehtäviä jotka ohjaimet yhdessä suorittavat.

Hajautetussa ohjauksessa ohjaimilla on oltava keino tietää toistensa tilat, jotta ohjaimet voivat säätää ajoa niin, että ratavirhe minimoituu.[6] Tämä tarkoittaa käytännössä väylää, jota pitkin ohjaimet voivat päivittää jatkuvasti toistensa tiloja tosiaikaisesti. Ohjaimilla voi olla keskenäinen väylä jota pitkin päivittyy tilatiedot ja aika. Näin ei ylemmän tason väylää tarvitse kuormittaa ohjainten välisillä tiedoilla ja ylemmän tason väylän ei tarvitse olla synkronoitu.

Jos akselit ajavat monimutkaisia liikeratoja tulee rinnanohjauksesta monimutkainen ja ohjaimien laskentakykyä vaaditaan entistä enemmän. Työn kohteena olevassa laittessa kaikki monen akselin liikkeet ovat identtisiä siten, että kaksi tai useampi akseli tekee samaa liikettä yhtäaikaa. Tästä seuraa, että ratavirheen kompensointi yksinkertaistuu huomattavasti, sillä ajettavana ratana on aina suora viiva, jota pitkin ajetaan.

Kuvassa 2.9 on havainnollistettu kahden paikkakaskadin rinnanohjaus. Ratavirhettä korjaava kompensointilohko *Comp* on lisätty säätimien rinnalle. Kompensointilohkon tarkempi toteutus riippuu liikkeen vaatimuksista, mutta perusideana on mitata ohjainten paikkavirhettä ja sen mukaan lisätä ohjaimien nopeussäätösilmukkaan korjaustermi, jolla ohjainten paikkavirheiden erotus minimoituu. Radanseurannassa oleellista ei ole yksittäisen akselin paikkavirhe vaan akseleiden yhteisen pisteen pysyminen radalla.



Kuva 2.9: Hajautetun rinnanohjauksen periaate.

2.4 Virittäminen

Säätimen virittämisellä tarkoitetaan säätimen parametrien asettamista siten, että säädettynä prosessi käyttäytyy halutunlaisesti. Jotta säätimen parametrit saadaan asetettua, täytyy prosessista olla riittävällä tarkkuudella oleva malli. Malli saadaan joko laskemalla tai prosessin käyttäytymistä voidaan mitata ja mittausten perusteella päätellään säätimen parametrit. Tässä työssä keskitytään säätimen virittämiseen prosessin mittauksen avulla.

2.4.1 Ziegler ja Nichols

Yleinen viritysmenetelmä on Ziegler:n ja Nichols:n (ZN) kehittämä kokeellisiin mittauksiin perustuva menetelmä. Suunnittelukriteerinä oli, että toisen ja ensimmäisen ylityksen suhde on yhden suhde neljään ja prosessina oli yhtälön (2.13) mukainen prosessi.[12]

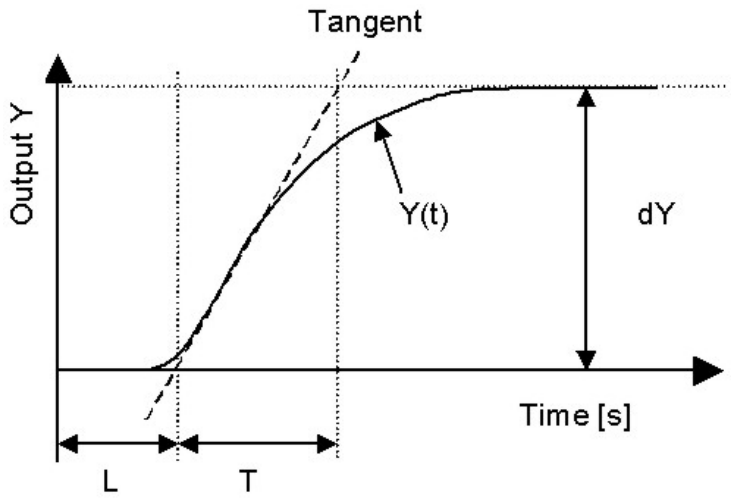
$$G(s) = \frac{b}{s} e^{-sL} \tag{2.13}$$

ZN tutkivat avoimen silmukan askelvasteesta saatavien mittausten perusteella, mitkä säätimen parametrit antavat hyvän säädetyn järjestelmän. Askelvastekoe on kuvattu kuvaajassa 2.10. Kuvaan on piirretty tangentti jyrkimmän nousun kohtaan. Kuvaajassa T on järjestelmän aikavakio ja L on viive. Taulukossa 2.1 parametri a on tangenttisuoran ja y-akselin leikkauskohta. Se saadaan katsottua kuvaajasta tai sijoittamalla $a = L/T$.

Toinen ZN käyttämä prosessin mittaustapa oli niin sanottu suljetun silmukan menetelmä, jossa prosessia ohjataan P-säätimellä ja vahvistusta lisätään, kunnes järjestelmä alkaa värähtelemään. Tämä vahvistus on ZN:n mukaan *Ultimate Gain*, K_u, ja värähtelyn jaksonaika on *Ultimate Period*, T_u. Taulukosta 2.2 voidaan katsoa säätimen parametrit.

Edellä kuvatuilla kokeilla on kuitenkin heikkoutensa. Avoimen silmukan askelvastekoe voi olla mahdoton toteuttaa jos silmukkaa ei saa avattua tai prosessi ei yksinkertaisesti kestä askelkoetta. Toinen ongelma on saadun askelvastekuvaajan tulkinnassa; tangentin asettaminen käyrän jyrkimpään kohtaan ei välttämättä ole helppoa. Tähän Åström ehdottaa parannukseksi että kuvaajasta lasketaan tunnusluvut pinta-alojen mukaan eli integroinnin avulla. [12] Lisäksi Åström suosittelee että askelvasteesta ei tarkastella pysyvään tilaan asettumista vaan 63% saavuttamista.

Resonanssikokeen, eli suljetun silmukan testin, heikkous on siinä että järjestelmä



Kuva 2.10: Ziegler ja Nichols askelvastekoe

Taulukko 2.1: Ziegler-Nichols askelvastekokeesta saatavat parametrit.[12]

Säädin	K	T_i	T_d	T_p
P	1/a			4L
PI	0.9/a	3L		5.7L
PID	1.2/a	2L	L/2	3.4L

joudutaan saattamaan epästabiiliin tilaan. Laite ei ole tällöin hallittavissa ja ohjattavana oleva laite voi rikkoutua.

Ongelmia voi tulla myös ZN ehdottamien säädinparametrien kautta. Saatava järjestelmä on suunniteltu ylittämään asetuspisteensä ja värähtelemään askelmaisen asetuspisteen muutoksen johdosta. Toisaalta ZN parametreilla saatava säädin vastustaa hyvin prosessin häiriöitä. Tämänlainen säädin toimiikin paremmin reguloivana säätimenä kuin asetuspistettä seuraavana servosäätimenä. Säätimen parametreja erilaisille säätimille ja prosesseille on runsaasti. Aidan O'Dwyer on kerännyt niistä kattavan otoksen kirjaan Handbook of PI and PID Controller Tuning Rules.[7]

Taulukko 2.2: Ziegler-Nichols resonanssikokeesta saatavat parametrit.[12]

Säädin	K	T_i	T_d	T_p
P	0.5 K_u			T_u
PI	0.4 K_u	0.8 T_u		1.4 T_u
PID	0.6 K_u	0.5 T_u	0.125 T_u	0.85 T_u

2.4.2 Rele-säätöön perustuva mittaus

Edellisten mittaustapojen ongelmana oli järjestelmän hallitsemattomuus ja mahdollinen rikkoutuminen mittauksen aikana. Releeseen perustuvassa mittauksessa säädin korvataan releellä, joka vaihtaa lähtönsä päinvastaiseksi kuin mitä erosuureen etumerkki on. Nyt releen lähdön suuruus, eli toimilaitteelle menevän ohjaussuureen suuruus, voidaan asettaa sellaiseksi että järjestelmä ei rikkoonnu ja näin saadaan hallittu oskillointi. Erosuureessa on usein mukana kohinaa, joka aiheuttaa releen virheellisen tilanvaihdon. Kohinan vaikutuksen voi minimoida lisäämällä releeseen hystereesiraja, jonka erosuureen täytyy ylittää, jotta rele vaihtaa tilaansa. Mittauksessa hystereesi täytyy asetella niin pieneksi kuin kohina sen sallii ja releen ulostulo niin suureksi kuin prosessi sen sallii. Näin häiriöt tulevat minimoiduiksi. Saadusta kuvaajasta voidaan tulkita jaksonaika ja yhtälöllä (2.14) lasketaan K_u eli *Ultimate Gain*. Yhtälössä d on releen syöttämän ohjaussuureen amplitudi ja a on toimilaitteen ulostulon amplitudi. ϵ on releen hystereesi. [12]

$$K_u(s) = \frac{4d}{\pi\sqrt{a^2 - \epsilon^2}} \tag{2.14}$$

Relemittauksella saatavia parametreja käytetään kuten alkuperäisen ZN metodin resonanssimittauksen parametreja.

2.4.3 Iteratiivinen parametrien asettelu

Taulukoidut säädinparametrit eivät ole joka tilanteessa täydellinen ratkaisu. Taulukoiden parametreillä kuitenkin päästään lähelle toimivaa ratkaisua. Myös ajan mittaan voi prosessin tai mittalaitteiden ominaisuudet muuttua ja säätimen parametreille joudutaan tekemään hienosäätöä käsin. Parametrit vaikuttavat

kuitenkin toinen toisiinsa, joten kun yhtä parametria muutetaan, joudutaan todennäköisesti muuttamaan muitakin ja niin edelleen. Käsin säätäminen onkin siten iteratiivinen prosessi. Taulukossa 2.3 on lueteltu eri parametrien vaikutus järjestelmään. Parametrit ovat rinnakkaismuotoisen säätimen lohkojen vahvistuksia.

Taulukko 2.3: Parametrien vaikutus käsin säädettäessä.

Parametri	Nousuaika	Ylitys	Asetusaika	Pysyvä virhe
K_P	laskee	kasvaa		laskee
K_I	laskee	kasvaa	kasvaa	poistaa
K_D		laskee	laskee	

Esimerkiksi ZN ehdottamat säätimen parametrit tuottavat säätimen joka ylittää asetuspisteen. Taulukosta 2.3 nähdään että P-lohkon vahvistusta pienentämällä ja D-lohkon vahvistusta lisäämällä ylitystä saadaan pienennettyä.

3. DISKREETTI SÄÄDIN

Diskreetti säätöjärjestelmä tarkoittaa järjestelmää, jonka sisääntulot ja lähdöt päivittyvät vain tietyillä ajanhetkillä.

Säätimen diskreetin toteutuksen voi tehdä niin, että suunnittelee jatkuva-aikaisen säätimen ja approksimoi sitä diskreetillä toteutuksella ja varmistaa mittauksin että toteutus on riittävän lähellä alkuperäistä suunnitelmaa. Jotta jatkuva-aikaisen approksimaatiota voidaan käyttää säätimenä, täytyy näytteenottotaajuuden olla suuri suhteessa prosessin aikavakioihin. Tällaista suunnittelutapaa puoltaa se että monet alan teokset käsittelevät jatkuva-aikaista säädintä.

Toinen keino on suunnitella alusta alkaen diskreetti säädin. Jotta järjestelmä säilyy aikainvarianttina, täytyy näytteenottotaajuuden ja ulostulojen päivityksen olla synkronoitu säätösilmukan kiertotaajuuteen. Tällä tavalla saavutetaan diskreetin säätimen kaikki hyödyt. Tässä työssä keskitytään ensimmäiseen suunnittelustrategiaan.

Lisäksi täytyy molemmissa tapauksissa huolehtia että jatkuva-aikainen prosessi toimii oikein säätimen hetkillä jolloin säädin ei päivitä tilaansa, sillä silloin järjestelmä toimii avoimen silmukan ohjauksena.

Molemmilla suunnitteluperiaatteilla päästään hyvään lopputulokseen PID-säätimen tapauksessa. Valintaperusteena voi olla prosessin mallintamisen helppous, suunnittelijan tottumus tai esimerkiksi yrityksen käytäntö.

3.1 Säätösilmukka

Tietokoneella toteutettu säädin on käytännössä jatkuvassa silmukassa pyörivä ohjelma. Silmukan ajon on syytä olla tasatahtista, jotta edellisiin silmukan suorituksiin perustuviin laskuihin ei tule ylimääräistä virhettä. Silmukan suoritus jakautuu tulojen lukemiseen, säätölaskuihin ja lähtöjen päivittämiseen. Kaikki mahdollinen laskeminen kannattaa tehdä silmukan ulkopuolella siten, että aika tulojen päivityksestä lähtöjen päivitykseen minimoituu.

3.2 Tosiaikavaatimukset

Säädinohjelmiston on toimittava toimilaitteen kanssa samassa tahdissa joten kyseessä on tosiaikajärjestelmä. Tosiaikajärjestelmät voidaan jakaa koviin

ja pehmeisiin järjestelmiin. Kovat tosiaikavaatimukset tarkoittavat, että laskenta saadaan varmasti joka kerta suoritettua oikeaan aikaan. Pehmeässä tosiaikajärjestelmässä voidaan joskus epäonnistua määräajassa laskemisessa, ilman että siitä on haittaa järjestelmän toiminnalle. Usein pehmeä tosiaikaisuus riittää, jos kyseessä ei ole turvallisuuskriittinen laite.[9]

Säätimen tehtävät voidaan jakaa jaksollisiin ja satunnaisiin. Näistä jaksolliset toistuvat kerta toisensa jälkeen ja yksinkertaisessa järjestelmässä voidaan jaksolliset tehtävät koodata käsin siten, että tehtäviä vuorotellaan aikaperustaisesti. Normaali säätimen toiminta on luonteeltaan jaksollista. Satunnaisia tehtäviä ei voi vastaavalla tavalla järjestää etukäteen. Satunnaisia tehtäviä ovat esimerkiksi säätimen kommunikointi ulkopuolisten ohjauslaitteiden tai toisten säätimien kanssa. Eräs tapa on varata niitä varten suoritusikkuna jossa pollataan satunnaisten tehtävien jonoa ja suoritetaan tehtäviä. Prosessorin ylikuormitustilanteessa täytyy päättää mitä jätetään tekemättä.[9]

Tehtävien suurin mahdollinen suoritusaika täytyy olla tiedossa että tehtävät voidaan ajoittaa oikein. Voi olla tilanne että tehtävä käyttää jotain yhteistä resurssia ja siten sen suoritusaika voi riippua jostain tuntemattomasta tekijästä. Tällöin ei juuri ole muuta mahdollisuutta kuin mittauksin varmistaa, että tehtävien väliin jää pahimmassakin tapauksessa riittävästi marginaalia. Mittauksen voi tehdä esimerkiksi mittaamalla oskilloskoopilla pinniä, jota ohjaa tehtävien suorituksen mukaan.[9]

3.3 Näytteistys ja uudelleengenereointi

Näytteistyksessä muodostetaan jatkuva-aikaisesta analogisesta signaalista numerojono, joka on kvantisoitunut niin aikatasossa kuin suuruudeltaan. Näytteistyksen taajuus täytyy Shannonin näytteenottoteoreeman mukaan olla yli kaksi kertaa suurempi kuin näytteistettävä signaali. Tämä tarkoittaa käytännössä sitä, että mitattava signaali tulee alipäästösuodattaa analogisella suotimella. Suotimen ominaisuudet vaikuttavat säätimeen ja ne täytyy ottaa huomioon säätöjärjestelmän suunnittelussa.[13]

Jatkuva-aikaisen signaalin muodostaminen diskreeteistä näytteistä on käänteinen tapahtuma edelliselle. Nyt ei ole tiedossa signaalin arvo kuin tietyillä hetkillä ja näiden hetkien väli täytyy jotenkin arvioida. Yksinkertaisin on nollannen asteen pitopiiri, joka pitää signaalia vakiona näytevälien ajan. Suora on siis vaakatasossa oleva viiva. Ensimmäisen asteen pitopiirissä vastaavasti extrapoloidaan kahden edellisen pisteen kautta piirretyllä suoralla jonka kulma-kerroin siis riippuu edellisistä pisteistä. Jos analoginen signaali saa olla näytejakson viiveellä, voidaan

käyttää seuraavan näytepisteen paikkaa suoran kulmakertoimen päättelemiseen tai jos viivettä ei sallita, voidaan kulmakerrointa koittaa ennustaa.[11]

3.4 Sisäinen esitystapa

Säätimen sisäisissä laskuissa numerot voidaan esittää kiinteän tai liukuvan pilkun aritmetiikalla. Sulautetuissa järjestelmissä ei yleensä ole laitteistopohjaista liukulukuaritmetiikkaa, joten liukulukulaskenta vievät huomattavasti prosessoriaikaa ja kooditilaa. Kiinteän pilkun aritmetiikka onnistuu tehokkaasti, mutta ohjelmoijan tulee huolehtia oikeasta pilkun paikasta ja lukujen skaalauksesta. Lisäksi lukualueen ylivuodot täytyy hallita. Liukuluvuilla skaalausongelmia ei ole, mutta toisaalta liukuluvut pyöristyvät sisäisen tallennustavan vuoksi. Pyöristyminen ei sinällään luvun tarkkuuden kannalta yleensä ole ongelma, mutta lukuja vertaillessa toisiinsa täytyy tämä ottaa huomioon.

3.5 Jatkuva-aikaisen säätimen tietokonetoteutus

Jatkuva-aikaista analogista säädintä muutettaessa tietokonepohjaiseksi diskreetiksi säätimeksi täytyy säätöyhtälöt diskretoida. Diskreeteistä yhtälöistä voidaan muodostaa algoritmit, joita ajetaan tietokoneen säätösilmukassa. Tässä esitellään kaksi yksinkertaista ja tässä työssä käytettyä diskretointimenetelmää, joilla aikatason yhtälöt voidaan diskretoida. Edistyneempiä menetelmiä ovat nollannen ja ensimmäisen asteen pitopiiriin tai Tustinin muunnokseen perustuvat menetelmät. [13]

3.5.1 Eteenpäin differentointi

Eteenpäin differentoinnissa jatkuva-aikainen derivaatta korvataan yhtälön (3.1) differentiaalilla. Yhtälössä h on näytteenottoväli.

$$\frac{dx(t)}{dt} = \frac{x(k+1) - x(k)}{h} \tag{3.1}$$

Integraattori yhtälössä (3.2) saadaan diskretoitua siten, että yhtälö derivoidaan puolittain, jolloin saadaan yhtälö (3.3). Derivaattori korvataan aproksimaatiolla ja saadaan yhtälö (3.4). Uudelleenjärjestelemällä saadaan yhtälö (3.5).

$$I(t) = \frac{K}{T_i} \int_0^t e(t)dt \tag{3.2}$$

$$\frac{dI}{dt} = \frac{K}{T_i} e(t) \tag{3.3}$$

$$\frac{I(t_{k+1}) - I(t_k)}{h} = \frac{K}{T_i} e(t_k) \tag{3.4}$$

$$I(t_{k+1}) = I(t_k) + \frac{K * h}{T_i} e(t_k) \tag{3.5}$$

Saatu yhtälö (3.5) on helposti toteutettavissa ohjelmallisesti. $I(t_{k+1})$ on siten summain, johon erosuuretta $e(t_k)$ summataan. Integraattorin kasautumisen esto saadaan summausta rajoittamalla.

3.5.2 Taaksepäin differentointi

Taaksepäin differentointi tapahtuu korvaamalla derivaatta yhtälöllä (3.6).

$$\frac{dx(t)}{dt} = \frac{x(k) - x(k-1)}{h} \tag{3.6}$$

Derivaattorin diskretointi on suoraviivaista korvaamalla yhtälön (3.7) derivaatta sen aproksimaatiolla, jolloin saadaan yhtälö (3.8). Uudelleenjärjestelemällä saadaan yhtälö (3.9), joka on tietokoneella toteutettavissa.[12]

$$D(t) = KT_d \frac{de(t)}{dt} \tag{3.7}$$

$$D(t_k) = KT_d \frac{e(k) - e(k-1)}{h} \tag{3.8}$$

$$D(t_k) = \frac{KT_d}{h} * (e(k) - e(k-1)) \tag{3.9}$$

Derivaattori tällaisenaan ei ole suoraan sovellettavissa käytännön säätimeen, vaan se vaatii suodatusta tai vahvistuksen rajoittamista korkeilla taajuuksilla.

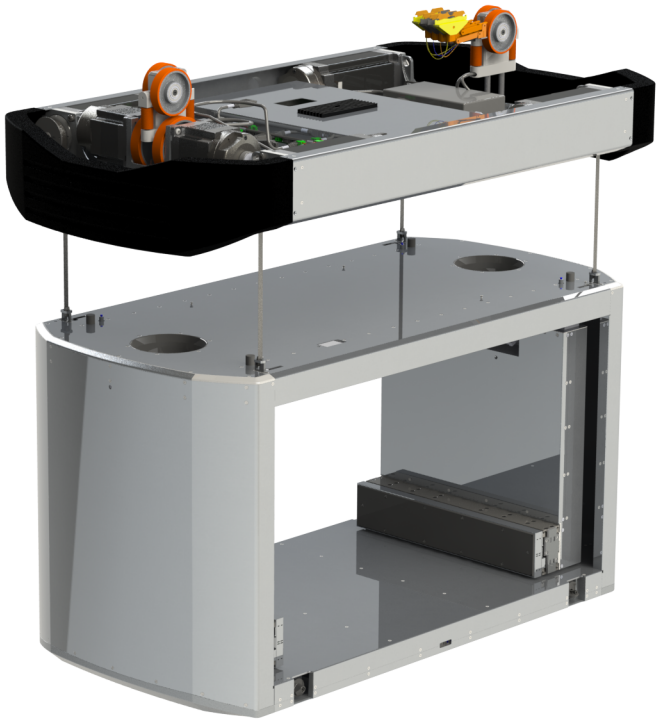
4. AJOLIIKKEELLE ASETETUT VAATIMUKSET JA TOTEUTUSYMPÄRISTÖ

Työn tarkoitus on toteuttaa Agilon-varastoautomaattiin pakettia liikuttavan sukkulan ajoliikkeen säätöjärjestelmä. Varasto on kokonaisuutenaan kuvassa 4.1. Varaston sisällä kulkee sukkula, joka koostuu nostolaitteesta ja kapselista. Nostolaite liikkuu ylhäällä kiskoa pitkin vaakasuunnassa ja nostolaitteesta laskeutuu hyllyjen väliin kapseli, joka liikuttaa paketteja. Sukkula on kuvassa 4.2.

Kuva 4.1: Agilon varastoautomaatti.

4.1 Ajoliikkeen vaatimukset

Ajoliikkeen tulee paikottua ilman ylitystä 2mm tarkkuudella. Ajonopeus tulee olla säädettävissä 4 m/s asti ja ylitystä ei saa tulla yli 5%. Kiihdytys- ja jarrutusrampit



Kuva 4.2: Sukkula koostuu kapselista ja nostolaitteesta.

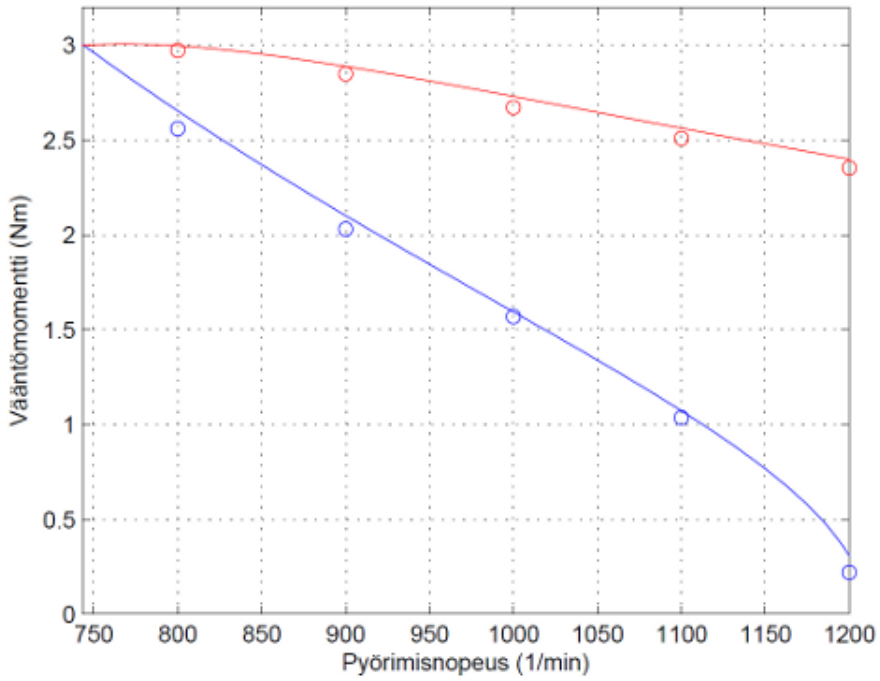
tulee olla säädettävissä ja mahdollisimman nopeita moottoreilta saatavissa olevan momentin puitteissa. Säätimen tulee hallita seisontajarrujen ja virtasäätimien päälle ja pois ohjaus siten, että liikkelle lähtö ja pysähtyminen on sulavaa ja äänetöntä. Säätimen pitää viestittää ylemmälle ohjauslogiikalle ajokomennon suorittamisen valmistumisesta. Liike ei saa rynnätä missään tilanteessa. Toisen pyörän luistaessa pitää toisen vetää täydellä momentilla. Pyörien luistatusta kaarteissa tulee välttää.

4.2 Askelmoottori

Vaunua liikuttaa askelmoottori, jota käytetään asentotakaisinkytkennän avulla kuten servomoottoria, perinteisemmän askeltamisen sijaan. Askelmoottorilla on ominaisuutena maksimimomentti paikallaan ollessaan ja akselilta saatavan momentin laskeminen moottorille ominaisen kierrosnopeuden jälkeen. Ajoliikkeessä käytettävää moottoria on tutkinut Veli-Matti Sainio diplomityössään [10]. Kuvaaja

moottorin akselilta saatavasta teoreettisesta momentista on kuvassa 4.3. Virtasäädin käyttää kentänheikennystä ja kompensointitaulukkoa jolla linearisoidaan akselilta saatavaa momenttia suhteessa momenttiohjeeseen. Virtasäädin toimii $20kHz$ taajuudella.

Liikkeen säätimen tulee toimia 4 m/s ajonopeuteen asti, mikä tarkoittaa 100mm halkaisijalla olevalla ajopyörällä noin 770 $1/min$ pyörimisnopeutta. Kuvasta 4.3 nähdään, että tällä kierrosnopeudella kentänheikennyksen avulla moottorista on saatavissa lähes täysi momentti. Näin moottori voidaan kohtuullisella tarkkuudella mallintaa käytettävälla kierroslukualueella optimaalisella virta-momentti muuntimella jonka muunnoskerroin $k_{T/I} = 3Nm/4.2A = 0.7\ Nm/A$.



Kuva 4.3: Moottorilta saatava teoreettinen maksimimomentti kierrosnopeuden funktiona kentänheikennyksellä (punainen) ja ilman kentänheikennystä (sininen). [10]

Moottoreita on ajoliikkeessä kaksi kappaletta, joten momentiksi voidaan laskea $6Nm$ koko käytettävälle kierroslukualueelle.

4.3 Ajoliikkeen mekaniikka

Varastorobotin ajoliike on toteutettu suoraan moottorin aksellille kytketyillä kumipäällysteisillä alumiinipyörillä. Välissä ei ole vaihteistoa. Kytkentä on siten välyksetön, lukuunottamatta tilanteita joissa pyörät alkavat luistamaan. Pyörimistä

vastustavaa momenttia tulee siitä, että kumipyörää painetaan jousivoimalla kiskoa vasten ja kumipäällyste painuu kasaan.

Ajopyörästö on nivelletty nostolaitteen kelkkaan, jotta laitteella pystytään ajamaan kaarteita. Rakenteet ovat valettua alumiinia ja nivel on kuulalaakeroitu. Rakenteet ovat varsin jäykkiä. Kiskon toisella puolella on vastaavanlainen pyörä vapaapyöränä moottorin ajopyörää vastaan. Tämä on otettu laskuissa huomioon.

Moottorin inertia ajopyörän kanssa J_{mp} on laskettu kohdassa (4.1). Moottorin inertia on saatu valmistajan kotisivuilta [2].

- Ajopyörä: m = 190g, d = 100mm
- Vapaapyörä: m = 140g, d = 75mm
- Moottori: J = 0,00027kgm^2

$$
\begin{aligned}
J_{mp} &= J_m + J_{p1} + J_{p2} \\
&= J_m + \frac{1}{2} m_{p1} r^2 + \frac{1}{2} m_{p2} r^2 \\
&= 0.00027 kgm^2 + \frac{1}{2} \cdot 0.190 kg * (0.05m)^2 + \frac{1}{2} \cdot 0.140 kg \cdot (0.0375m)^2 \\
&= 0.00027 kgm^2 + 0.000238 kgm^2 + 0.0000984 kgm^2 \\
&= 0.00061 kgm^2
\end{aligned}
\tag{4.1}
$$

Edellä laskettu moottorin ja pyörästön inertia koskee yhtä moottoria. Laitteessa moottoreita pyörineen on kaksi kappaletta, joten yhteenlaskettuun inertiaan edellä saatu tulos täytyy kertoa kahdella. Kun oletetaan ajopyörästön ja kelkan yhteys joustamattomaksi, voidaan kelkan massa redusoida moottorin akselilla pyöriväksi hitaudeksi $J_{redusoitu}$. Yhteenlaskettu moottorille näkyvä hitausmassa on J_{tot}. (4.2). Kelkan massaksi arvioidaan 50kg.

$$
\begin{aligned}
J_{tot} &= 2J_{mp} + J_{redusoitu} \\
&= 2J_{mp} + m_{kelkka}r^2 \\
&= 2J_{mp} + 50kg \cdot (0.05m)^2 \\
&= 2 \cdot 0.00061kgm^2 + 0.125kgm^2 \\
&= 0.12622kgm^2 \\
&= 0.13kgm^2
\end{aligned} \tag{4.2}
$$

Laskuista voidaan todeta, että pyörivien osien inertia on huomattavan pieni suhteessa liikuteltavaan massaan.

4.4 Ohjelmiston alusta

Tässä kappaleessa kuvataan piirikortin sähköisiä toimintoja siltä osin kuin liikkeenohjauksen kannalta on välttämätöntä, sekä rajapinnat joiden avulla liikkeensäätö toteutetaan.

4.4.1 Alteran FPGA

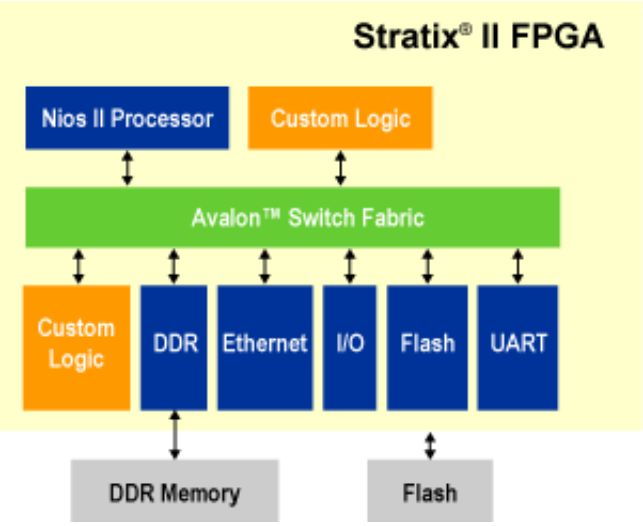
Liikkeensäätimen alustana on Alteran Cyclone 3 FPGA-piiri, kuva 4.4. Piirillä on oma Nios II prosessori säädintä varten, kuva 4.5. Prosessori on 32 bittinen ja RISC-tyyppinen. Prosessori käyttää liukulukulaskennassa laitteistokiihdytystä. Piirin sisällä on väylä (Avalon), jota pitkin prosessori on yhteydessä muihin piirillä oleviin komponentteihin [1]. Kyseessä olevassa järjestelmässä näitä ovat esimerkiksi virtasäädin ja moottorin roottorin asennon anturointi.

4.4.2 Moottorinohjausrajapinta

FPGA:lle on toteutettu logiikalla moottorin virtasäädin ja moottorin asennon- ja nopeudenmittaus. Logiikkalohkot näkyvät prosessorille Avalon väylän yli 32 bittisenä muistina johon voi kirjoittaa ja josta voi lukea.

Virtasäädin toimii 20kHz taajuudella. Liikkeensäätöä tekevä prosessori kirjoittaa virtaohjeen virtasäätimelle, joka H-siltoja ohjaamalla saa käämeihin aikaan halutun virran ja momentin. Koska askelmoottorissa ei ole mekaanista kommutointia, käyttää virtasäädin kulma-anturia kommutointiin.

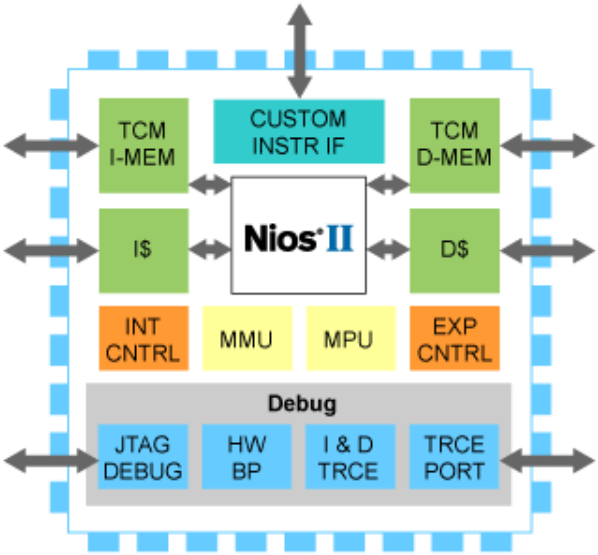
Moottorin asentotieto saadaan kulma-anturilta 12 bittisenä lukuna 6kHz taajuudella. Tästä lasketaan logiikalla nopeus. Näitä tietoja liikkeenohjausprosessori



Kuva 4.4: Nios2 prosessori Avalon väylällä. [1]

käyttää säädössä.

Varastosukkulaa liikutetaan kiskolla aiemmin mainittujen kumipyörien vetämänä. Koska pyörät voivat luistaa, lasketaan vaunun absoluuttinen paikka optisen anturin avulla. Paikka kiskolla päivittyy kun absoluuttianturi saa luettua paikkatiedon ja muuten edetään moottorin kulma-anturin mukaan. Moottorin kulma-anturi antaa paikkatiedon 12 bitin tarkkuudella eli 4096 asemaa kierroksella. Ajopyörän halkaisija on noin 100mm joten kulma-anturin paikoitustarkkuus on teoriassa luokkaa 0.1 mm. Käytännössä pyörän halkaisija ei ole aivan tarkka valetun kumipäällysteen vuoksi ja kumipinnoitteen kokoonpainuminen ja pyörän luistaminen vaikuttaa kierroskohtaiseen etenemään myös.



Kuva 4.5: Nios2 prosessori. [1]

5. AJOLIIKKEEN SÄÄDIN

Koska säätimeltä vaaditaan sekä paikkasäätöä ja nopeussäätöä, on paikka- ja nopeussäätimen kaskadi hyvä lähtökohta säätimen suunnittelulle. Kaskadirakenteeseen on lisäksi mahdollista helposti liittää muita säätörakenteita, joilla monta kaskadisäädintä saadaan tekemään liikettä yhdessä synkronisesti.

Tässä luvussa tehdään ensin Matlabin Simulinkillä kaskadisäädin, jolla paikka- ja nopeussäätimien toiminta varmistetaan. Laitteistoon tehdään tiedonkeruujärjestelmä, jolla saadaan tallennettua jokainen säätimen muuttuja halutulla näytteistystaajuudella. Testialueelle rakennetaan ajorata, jossa laitetta voidaan ajaa turvallisesti. Säätimen kehitystä jatketaan oikella laitteella mahdollisimman aikaisessa vaiheessa.

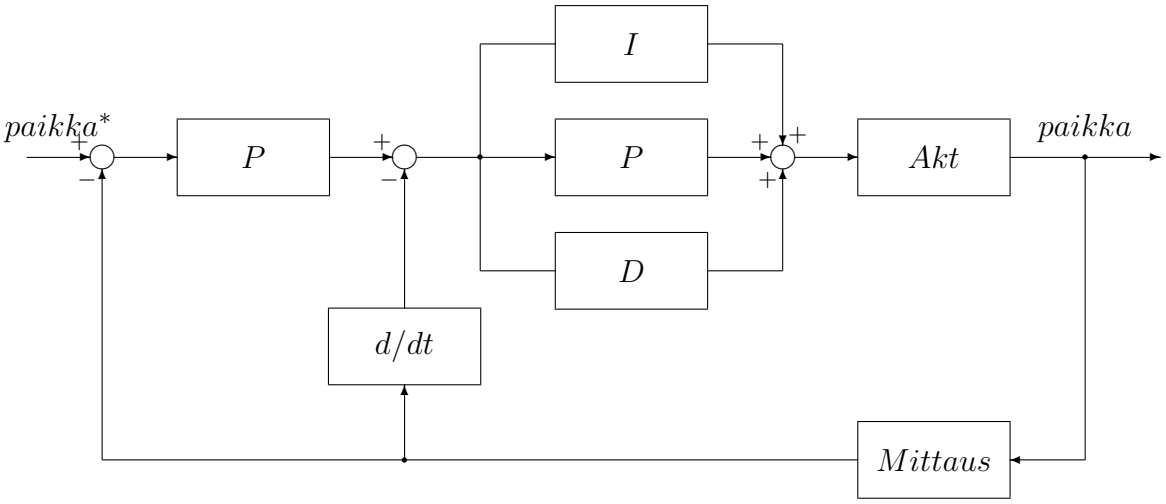
5.1 Säätimen rakenne

Kuvassa 5.1 on kuvattu säätimen periaatekuva. Sisimpänä on nopeussäätösilmukka, jossa on rinnakkainen PID-säädin. Puhtaaseen rinnakkaiseen rakenteeseen päädyttiin yksinkertaisen tietokonetoteutuksen vuoksi. Rinnakkaisesta rakenteesta on helppo kytkeä halutut säätöhaarat käyttöön ja pois yksinkertaisesti vahvistuskertoimella.

5.2 Simuloinnit

Simulointi on suoritettu Matlabin Simulink-työkalulla. Lohkojen kuvaukseen on käytetty s-funktioita, jotta simulaattorissa ajettava koodi olisi täysin samaa kuin mitä lopullisen säätimen koodi on. Näin iteratiivinen kehittäminen todellisen laitteen ja simulaattorin välillä on mahdollisimman helppoa ja simulaatiot säätimen osalta ovat mahdollisimman todenmukaisia. Simuloinneilla haettiin varmistusta säädinrakenteen soveltuvuudelle kyseiseen käyttötapaukseen, joten mallin ja mittauksien epätarkkuuksia ei huomioitu. Liitteessä B on valmis c-kielinen ohjelmakoodi, joka vastaa simulaattorissa ajettua koodia.

Säätimen kehittäminen aloitettiin kaskadin sisemmästä säätösilmukasta eli nopeussäätimestä. Kuormalle luotiin yksinkertainen malli, jossa kaikki massa on redusoitu moottorin akselilla näkyväksi inertiaksi. Kitkakerroin arvioitiin siten että $1m/s$ ajo vie 35% $6Nm$ momentista. Todellisuudessa liikkeellelähtömomentti voi olla



Kuva 5.1: Säätimen periaate.

huomattavankin suuri ja nopeuteen verrannollinen kitka lienee suurilla nopeuksilla korostunut.

5.2.1 Nopeussäädin

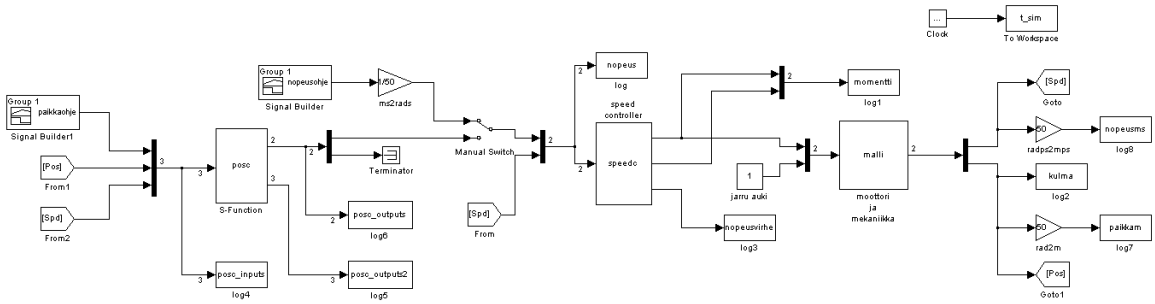
Käytetty simulointimalli on esitetty kuvassa 5.2. Mallissa on kaikki hitausmassa laskettu akselille ja kitka on arvioitu olevan nopeudesta lineaarisesti riippuva. Mallin tarkkuus ei mahdollista säätimen parametrien asettelua, eikä sen avulla voi tarkasti ennustaa todellisen järjestelmän suorituskykyä. Mallin avulla voidaan kuitenkin kehittää säätimen rakenteita sen verran, että kehitystä voidaan jatkaa todellisella laitteistolla.

Nopeussäätimelle syötetty nopeusohje generoitiin Simulinkin funktiogeneraattorilla ja se on kuvaajaan nimetty nopeusohjeeksi.

Säätimelle syötetyn nopeusohjeen kiihtyvyys on alussa liian suuri ja säätimen ulostulona oleva momentti saturoituu. Tästä seuraa kuvassa 5.3 näkyvä voimakas ylitys noin 3 sekunnin kohdalla.

Yksinkertaisimmillaan integraattorin kasautuminen, eli windup, voidaan estää ehdollisella integroinnilla. Tällöin toimilaitteen ohjauksen mennessä ääriarvoonsa, integraattorin toiminta keskeytetään. Kuvassa 5.4 on integraattorin toiminta keskeytetty, kun ulostulo on maksimirajalla.

Pelkästään ulostulon saturoituessa integroinnin estäminen ei välttämättä riitä. Jos toimilaite ei pysty saavuttamaan ohjearvoaan pitkään aikaan, tulee integraattoriin summautuneeksi huomattavan iso summa. Tällaisesta tilanteesta



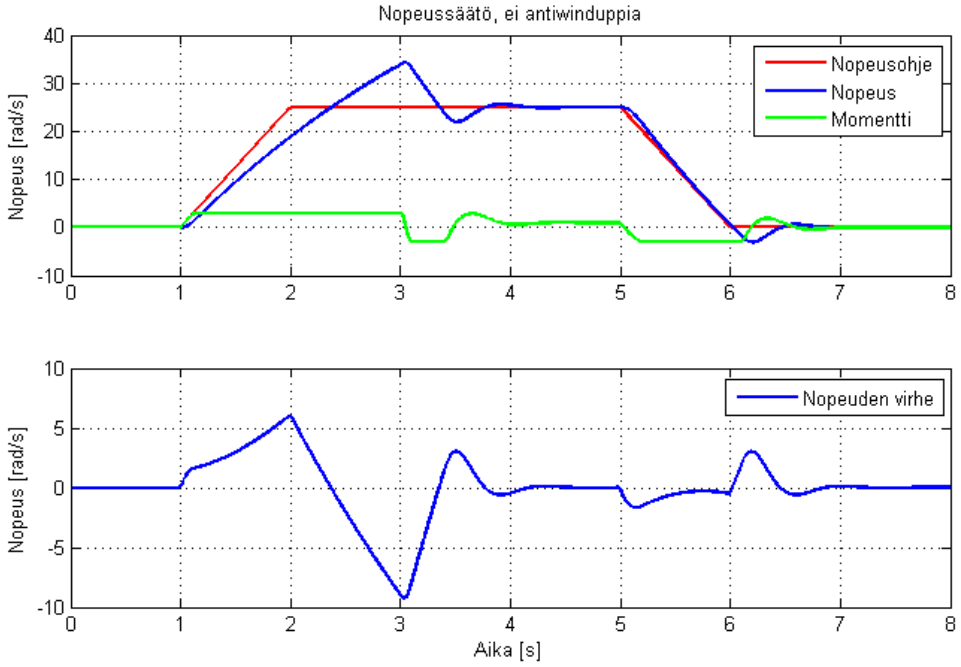
Kuva 5.2: Nopeussäätimen simulointimalli.

toiseen suuntaan lähteminen ilman integraattorin nollaamista on hidasta. Työn kohteena olevassa kiskolla kulkevassa vaunussa tällaista tilannetta normaaliajossa ei tule. Jos vaunu ei pääse etenemään kiskollaan on kyseessä virhetilanne joka vaatii toimia korkeamman tason logiikalta.

5.2.2 Paikkasäädin

Paikkasäätimen tehtävä on paikkatakaisinkytkennän ja paikkakomennon avulla laskea nopeussäätimelle nopeusohje. Nopeussäätimen simuloinnissa käytetyn funktiogeneraattorin ulostulo on se, mitä paikkasäätimen pitää tuottaa. Paikkasäätimen pitää pystyä rajoittamaan nopeusohje haluttuun nopeuteen. Kiihdytyksen ja jarrutuksen suuruus pitää olla myös asetettavissa erikseen sopiviksi. Paikkasäätimeksi riittänee pelkkä P-säädin, sillä nopeussäätimen integraattori riittää pysyvän paikkavirheen poistamiseen.

Kuvassa 5.5 on paikka ajan funtiona, kun paikkasäätimelle on annettu askelmainen paikkaohjeen muutos. Alemmassa kuvassa on paikkasäätimen tuottama nopeusohje, jota nopeussäädin seuraa. Paikkasäädin tuottaa rajoitetun nopeusohjeen ja rajoitetun kiihtyvyyden. Hidastuksessa paikkasäädin tuottaa loivemman hidastumisen kuin rajoitus sallisi. Tämä johtuu pienestä P-kertoimesta paikkasäätimessä. Kasvattamalla kerrointa saadaan hidastusramppia jyrkennettyä kunnes paikassa tapahtuu ylilyöntiä ja viimein oskillointia. Säätimien parametreilla ei ole tarkoitus simuloinneissa päästä parhaaseen mahdolliseen säätöön, vaan tarkoitus on saada halutut ilmiöt näkymään.



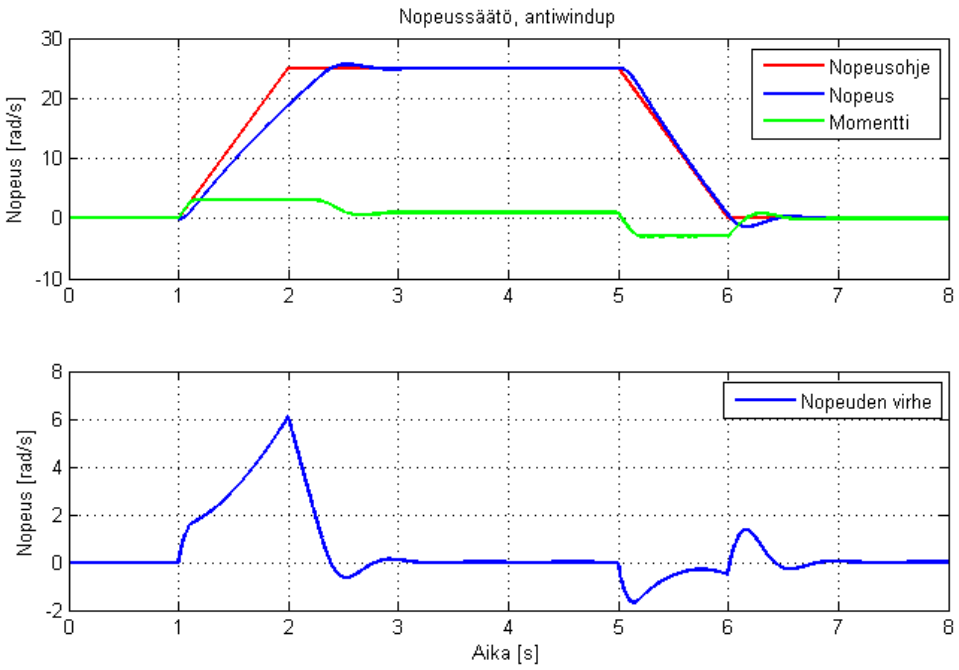
Kuva 5.3: Nopeussäädin ilman antiwinduppia.

5.2.3 Myötäkytkentä (feedforward)

Koska paikkasäätimen ulostulon nopeus ja kiihtyvyys on joka hetki tiedossa, voidaan tätä tietoa käyttää suoraan takaisinkytketyn säätimen rinnalla myötäkytkentänä. Jotta myötäkytkentä toimisi täysin, täytyy kuormasta olla täydellinen malli. Näin on tilanne vain simulaattorissa. Todellisuudessa kitkan aiheuttama voima ei ole lineaarisesti kasvava esimerkiksi nopeuden tai kuorman painon suhteen. Usein liikkeellelähdössä on kitkavoimassa piikki ja kitkavoima ei ole lineaarinen funktio.[8]

Kiihtyvyyden vaatiman momentin kompensointi myötäkytkennällä tuottaa liikkeellelähtöön terävän momentin nousun. Tässä työssä liikkeellelähdössä pehmeys on kuitenkin toivottava ominaisuus, joten kiihtyvyyden kompensointi myötäkytkennällä otetaan käyttöön vain, jos liikkeelle lähdössä tulee jotain ongelmia.

Myötäkytkennän toiminta varmistetaan valmiilla laitteistolla, sillä simulaatiot yksinkertaistetuilla malleilla antavat todennäköisesti liian optimistisen kuvan.



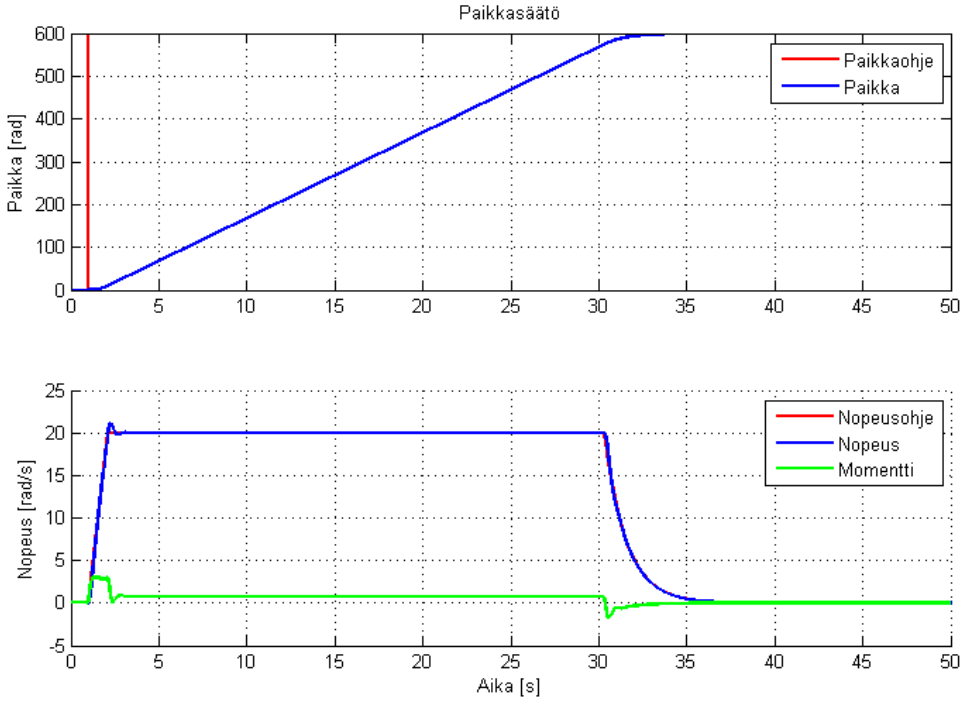
Kuva 5.4: Antiwindupin toiminta.

5.3 Koeajo

Koeajoa varten rakennettiin 6 metriä pitkä koeajorata, jossa kiskolla on pelkkä nostolaite ilman kapselia, kuten kuvassa 5.6 näkyy. Laitetta ohjataan kannettavassa tietokoneessa ajettavalla python-ohjelmalla homeplug-väylän avulla. Nostolaitteen FPGA:lle on tehty oma prosessori tiedon tallentamista varten. Tälle datalogger-prosessorille voidaan antaa seurattavat parametrit ja niiden tallennustaajuus ja aloitus- ja lopetuskäsky. Datalogger käyttää lisämuistia tiedon tallentamiseen ja siirtää tiedon verkon yli kannettavalle tietokoneelle, jossa se voidaan esittää graafisesti. Lisämuistin avulla prosessori voi puskuroida mittadataa ajon aikana ja siirto analysoitavaksi voidaan tehdä ajon jälkeen. Näin verkon siirtonopeus ei vaikuta näytteistystaajuuteen.

Jotta nostolaitteella voidaan ajaa koeajo, täytyy säätimiin saada parametrit joilla laite toimii tyydyttävästi. Laitteelle tehtiin yksinkertainen relekoe ja parametrit valittiin Ziegler:n ja Nichols:n taulukoista. Relekokeeseen päädyttiin askelkokeen sijaan siksi, että näin vältyttiin seinään ajamiselta ja laitteen rikkoontumiselta.

Säätimistä toinen nimettiin isännäksi ja toinen orjaksi. Orjan tehtävä tässä vaiheessa on vain tuottaa sama momentti kuin mitä isäntä tuottaa. Näin veto on symmetristä ja pyörästö ei pyri vääntymään vinoon kiskolla. Isännän säätimen



Kuva 5.5: Paikkasäätimen toiminta.

tilalle laitettiin hystereesillinen ohjelmallinen rele. Relekokeen tulos on kuvassa 5.7. Kuvassa sinisellä on nostolaitteen nopeus kiskolla ja vihreällä moottorien tuottama momentti. Kuvaajassa momentti on 8 bittinen luku siten että 255 vastaa $6Nm$:ä. Relekokeessa tuli ilmi, että kyseisessä yksilössä on jostain syystä huomattava välys, vaikka moottorit ovat suoravetoisia. Tästä johtuen kuvassa näkyy nopeusvärähtelyä suunnan vaihdon kohdalla. Suoritettiin toinen relekoe hyvin pienellä hystereesillä ja vastaava kuvaaja on kuvassa 5.8. Tässä värähtely on käytännössä moottorin vapaata pyörimistä välyksen laidasta toiseen.

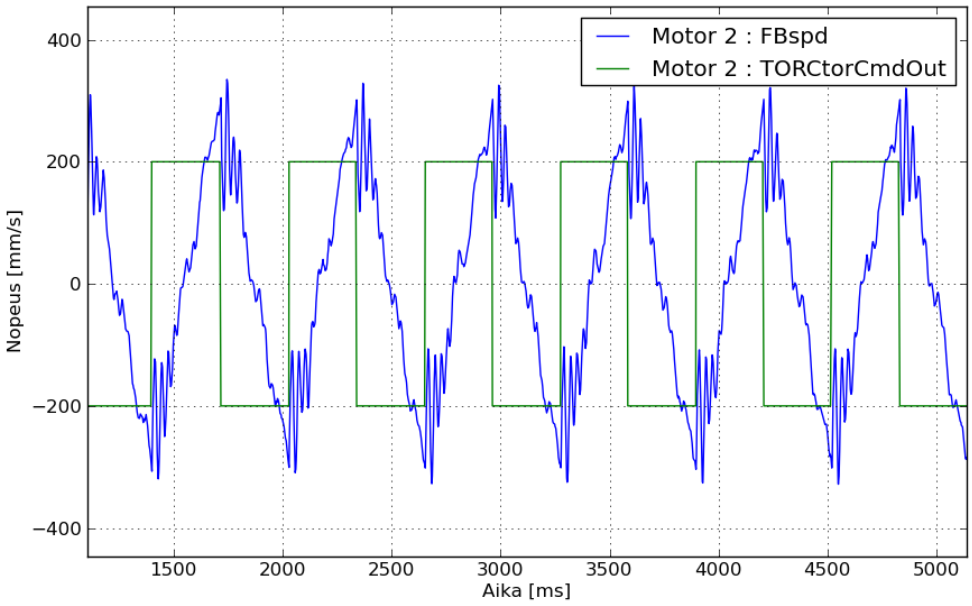
Mittauksien tulokset on kerätty taulukkoon 5.1. Arvot ovat amplitudeja ja d-sarakkeen arvo on laskettu moottorin momenttiohjeesta siten että 255 vastaa $6N$ momenttia. K_u on laskettu yhtälöissä (5.1) ja (5.2) mittauksille 1 ja 2 vastaavasti. T_p , T_i ja k parametrit katsottiin Zigler ja Nicholsin taulukoista 2.2.

Kuva 5.6: Nostolaite koeajoradalla.

$$K_{u1} = \frac{4d_1}{\pi\sqrt{a_1^2 - \epsilon_1^2}} = \frac{4 * 4.8}{\pi\sqrt{0.65^2 - 0.6^2}} = 24.4 \tag{5.1}$$

$$K_{u2} = \frac{4d_2}{\pi\sqrt{a_2^2 - \epsilon_2^2}} = \frac{4 * 1.2}{\pi\sqrt{0.055^2 - 0.01^2}} = 28.3 \tag{5.2}$$

Ensimmäisen kokeen mukaan lasketut parametrit tuottivat voimakasta värähtelyä



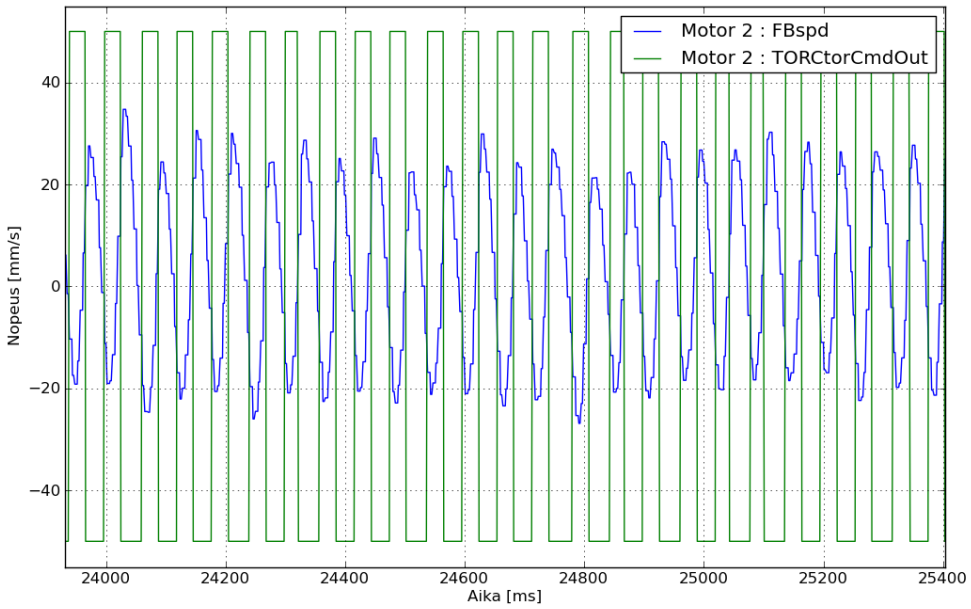
Kuva 5.7: Releajo 300mm/s hystereesillä.

Taulukko 5.1: Relekokeiden tulokset.

Mittaus	d $[N]$	a $[mm/s]$	ϵ $[mm/s]$	T_u $[ms]$	K_u
1	4.8	650	600	625	24.4
2	1.2	55	10	61.7	28.3

aina, kun laite oli muussa kuin kiihdytys- tai jarrutustilanteessa. Toisen kokeen parametrit tuottivat laiskan ajoliikkeen. Parametrit ovat taulukossa 5.2. Taulukossa T_p on suhteellisen haaran vahvistus, T_i on integrointiaika ja k on yhteinen vahvistus. Säätöalgoritmissa yhteinen kerroin on laskettu säätöhaaroihin mukaan ja nämä kertoimet ovat G_i ja G_p.

Toinen koe otettiin lähtökohdaksi säätimien parametreille ja tämä tilanne on kuvattu kuvaajassa 5.9. Nopeussäätimen G_i ja G_p kertoimia kasvattamalla ja paikkasäätimen G_p kertoimen asettamalla sopivaksi, saatiin ajoliike toimimaan suhteellisen hyvin. Isommilla G_i ja G_p kertoimilla tehty ajoliike on kuvattu kuvassa 5.10 ja käytetyt parametrit ovat taulukossa 5.3. Edellä mainituissa kuvissa mitattu nopeus on sininen ja paikkasäätimen tuottama nopeusohje on vihreä käyrä.



Kuva 5.8: Releajo 5mm/s hystereesillä.

Nopeussäätimeltä lähtevä momenttiohje on punainen. Käyrien nimet ovat vastaavat säätimen sisäisten muuttujien nimet.

Nopeudessa ja paikassa ei tapahdu ylitystä, mutta kiihdytys- ja hidastustilanteissa ei nopeus pysty seuraamaan ohjetta. Tämä on hyväksyttävä ominaisuus, joka ei haittaa kun tehtävänä on ajaa pisteestä toiseen, eikä varsinaisesti seurata tarkasti rataa. Kuvasta 5.10 nähdään, että tasaisella nopeudella ajettaessa tarvitaan momenttiohjetta noin 40 kyseisellä asteikolla. Myötäkytkentään voidaan asettaa kerroin joka tuottaa tämän momentin. Lisäksi kiihdytykseen ja jarrutukseen voidaan laittaa pieni myötäkytkentä, kuitenkin kolahtavaa liikkeellelähtöä varoen. Kuvassa 5.11 on ajettu sama matka kuin edellä mutta pelkällä myötäkykennällä eli avoimella säätösilmukalla. Kuvasta nähdään, että myötäkytkentä tuottaa tasaisella ajolla saman momentin kuin säädinkin. Kiihdytyksen ja hidastuksen vaatimaa momenttia ei ole toistaiseksi lisätty myötäkytkentään alkunykäyksen välttämiseksi.

Ajoliike ei saa ylittää sille annettua maksiminopeutta, sillä tämän nopeuden ylittyessä tehdään laitteistopohjainen hätäpysäytys. Kuvassa 5.12 on alussa ajoliikkeen eteneminen estetty ja noin 4 sekunnin kohdalla ajon annettu jatkua. Kuvasta voidaan todeta, että ainakaan tämän kaltainen häiriö ei saa ylinopeutta

Taulukko 5.2: Säätimen parametrit Zieglerin ja Nicholsin mukaan.

Mittaus	T_p	T_i $[ms]$	k	G_i	G_p
1	0.875	500	9.8	4.9	8.75
2	0.086	49.4	11.3	0.56	0.97

Taulukko 5.3: Säätimen parametrit iteroituina

G_i	G_p
1.6	2.0

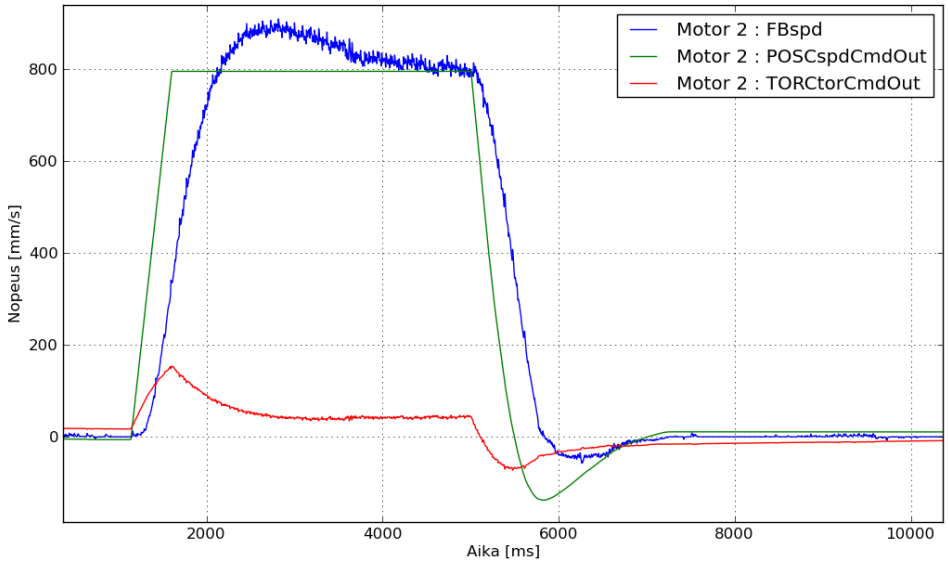
aikaiseksi. Alussa momentti kasvaa maksimiarvoonsa ja kun nostolaite pääsee liikkeelle, se kiihtyy hallitusti oikeaan nopeuteen.

5.4 Luistonesto

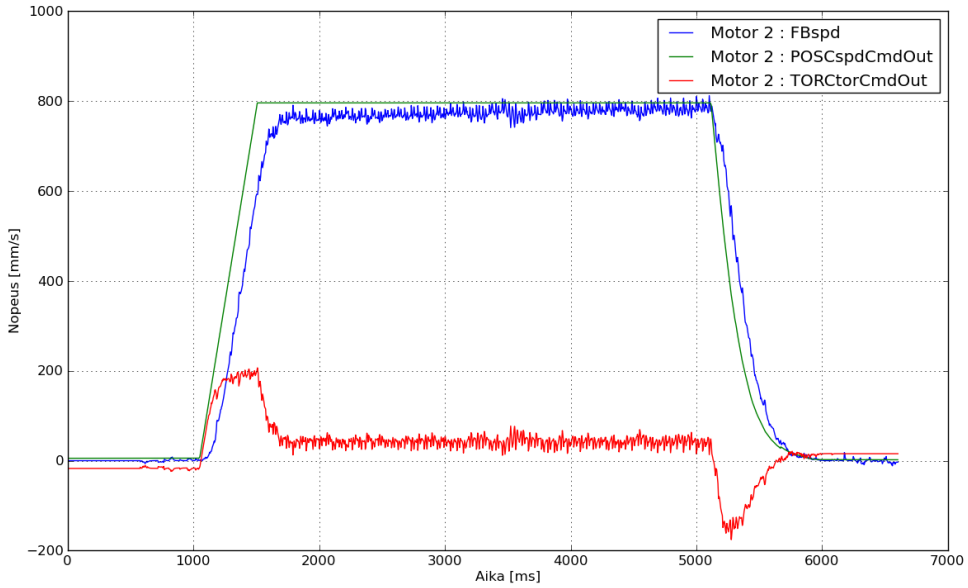
Säädin ohjaa kahta moottoria, jotka liikuttavat nostolaitetta kiskolla kumipyörien avulla. Jos kuorma on kovin epätasapainossa, laite kallistuu kiskolla ja toinen pyörä luistaa.

Kuvassa 5.13 on nostolaitteella ajettu kiskolla ja käsin poikkeutettu sitä siten että nostolaite keikkuu kiskolla. Kuvassa on merkitty sinisellä isännän nopeus ja vihreällä orjan nopeus. Punainen viiva on paikkasäätimen tuottama nopeusohje. Vaaleansininen ja purppura ovat isännän ja orjan momenttiohjeet. Lisäksi kuvaajassa on keltainen viiva, joka on toisella tavalla mitattu nopeus kiskolla. Moottorit ovat pyörästössä vastakkain, joten kuvaajat ovat toistensa peilikuvat vaaka-akselin suhteen. Kuvassa näkyy kuinka orjan luistaessa sen nopeus, eli vihreä viiva, kasvaa suureksi. Kun taas isännän vetopyörä on ilmassa alkaa se värähtelemään ja sen momenttiohje oskilloi äärirajojen välillä. Kun tämä kopioidaan orjamoottorille, alkaa orjakin värähtelemään. Tämä tilanne on kuvattu tarkemmin kuvassa 5.14.

Yksinkertainen ratkaisu on käyttää toista moottoria isäntänä ja toista moottoria orjana. Orjamoottorin tehtävänä on vain tuottaa sama momentti kuin isäntämoottori tuottaa. Näin moottorien eri etenemä kaarteissa ei vaikuta vaunun kulkuun ja normaalitilanteessa symmetrinen momentti molemmissa moottoreissa

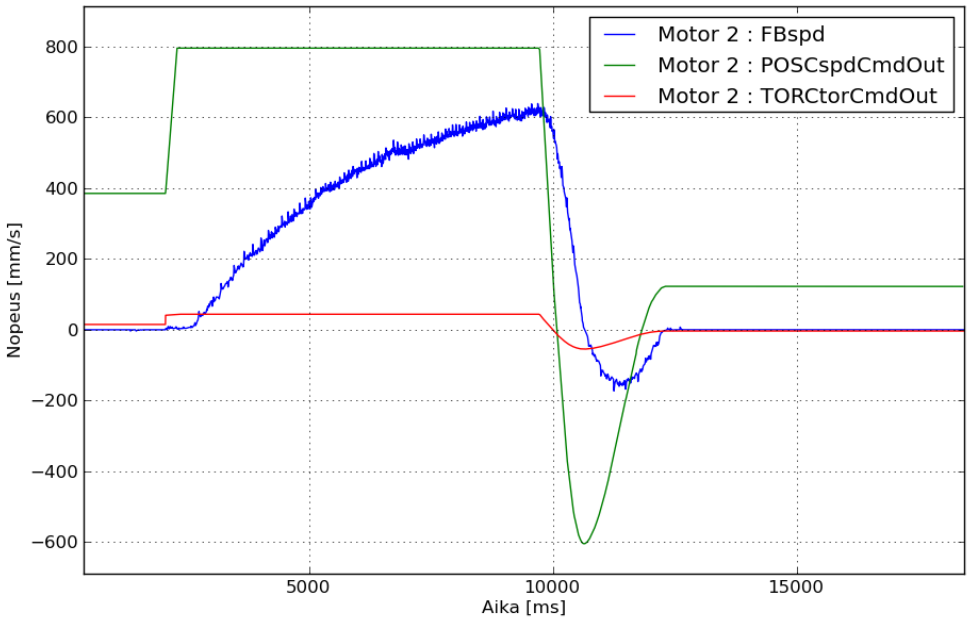


Kuva 5.9: Ensimmäinen koeajo.



Kuva 5.10: Koeajo paremmilla I- ja P-parametreilla.

pitää pyörästön suorassa. Ongelmana on tilanne jossa isäntämoottorin pyörä luistaa.



Kuva 5.11: Koeajo pelkällä myötäkykennällä.

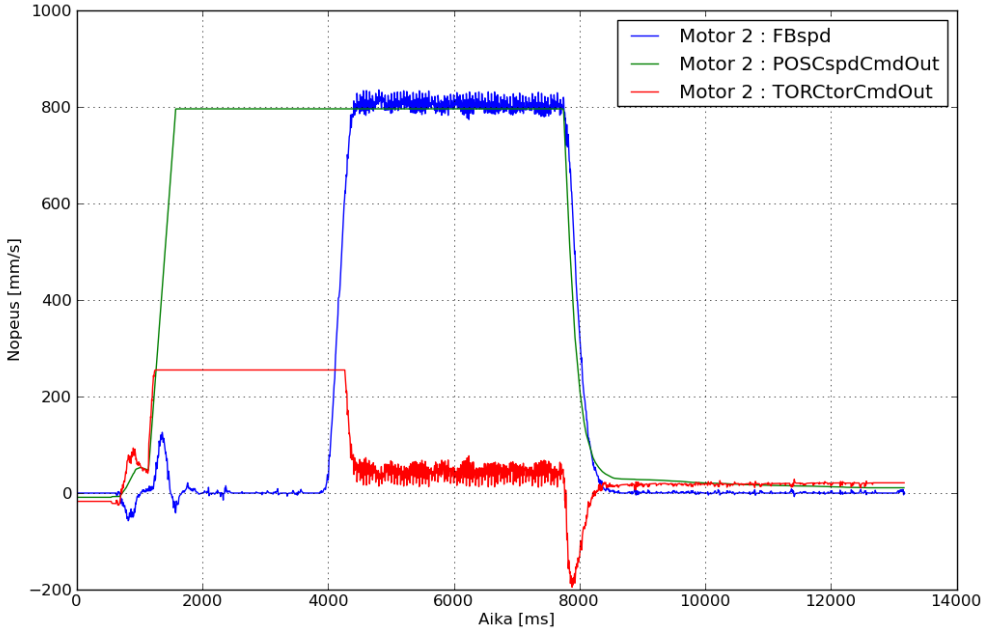
Isäntämoottorin luistaessa on sen momentti pieni tai jos pyörä on täysin ilmassa voi moottori alkaa värähtelemään. Kun tämä isännän momenttiohje kopioidaan orjalle, alkaa sekin värähtelemään tai saa tarpeettoman pienen momenttiohjeen. Näin laite voi jäädä jumiin vaikka momenttireserviä ja pitoa vielä orjalaitteella olisi.

Edellä kuvatussa luistotilanteessa moottoreiden pyörimisnopeudet eivät ole yhtäsuuret. Tätä tietoa voidaan käyttää hyväksi ja laskea akseleiden nopeuserosta orjalle lisättävä momenttiohje. Säätimen kerroin täytyy hakea siten että normaali- ja kaarreajossa ei orjalle tule lisätyksi merkittävää momenttia mutta luistamistilanteessa lisättävä momentti on tarpeeksi normaalia ajoa varten.

$$T_2^* = T_1^* + P_{tc} \cdot (\hat{\omega}_1 - \hat{\omega}_2) \tag{5.3}$$

Kuvassa 5.15 ja yhtälössä (5.3) on kuvattu aikatasossa orja-akselin luistoneston rakenne. T_1^* on isännän momenttiohje ja P_{tc} on luistoneston P-kerroin. $\hat{\omega}_1$ on isännän todellinen nopeus ja $\hat{\omega}_2$ orjan. Kun orjana oleva akseli luistaa ja siten pyörii nopeammin kuin isäntäakseli vähenee yhtälön mukaan orjan momentti.

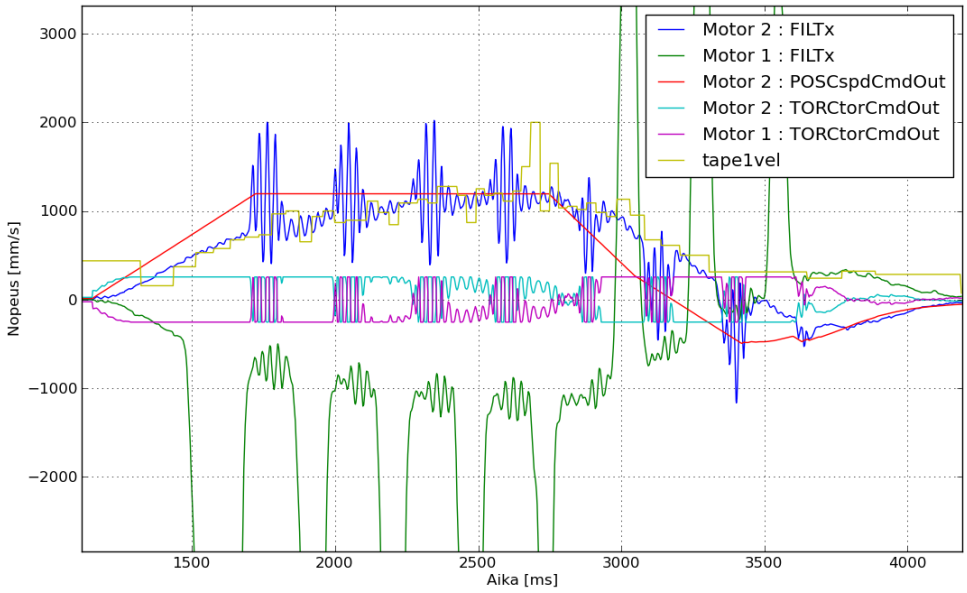
Säätimeen lisättiin edellä kuvattu säädinrakenne isännän ja orjan välille.



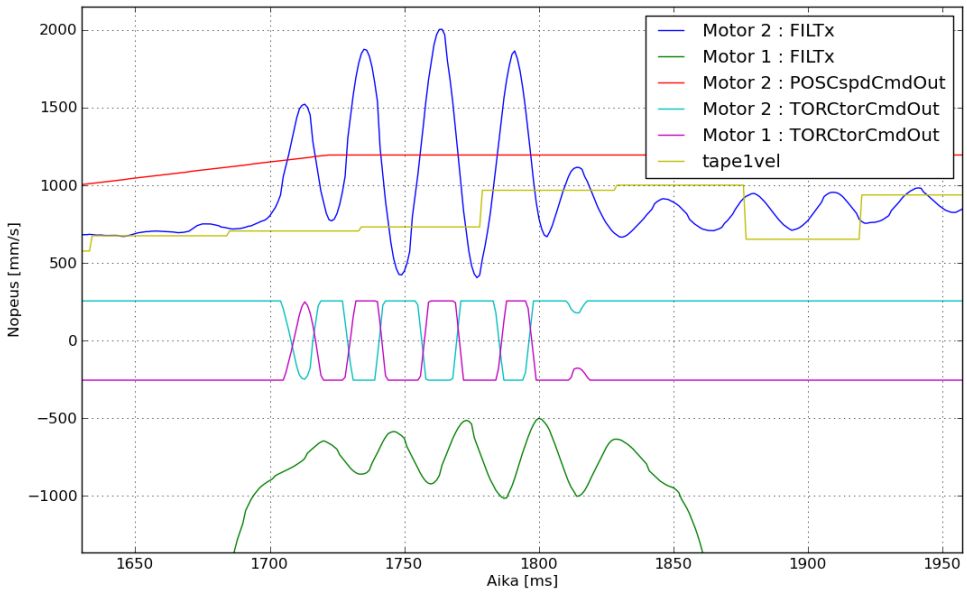
Kuva 5.12: Nostolaite ei ryntää estämisen jälkeen.

Lisäksi nopeuksien erotus alipäästösuodatettiin. Kuvassa 5.16 isäntämoottori luistaa ja värähtelee. Alipäästösuodatettu nopeusero on kuvassa keltainen viiva. Nyt orjamoottorin momenttia on korjattu ja se ei värähtele isännän mukana.

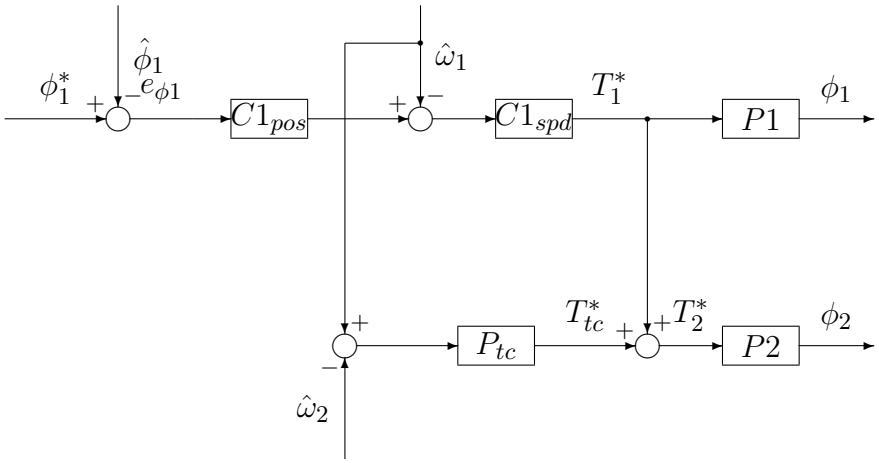
Kuvassa 5.18 on päivitetty säädin joka toteutettiin ajoliikkeen ohjainkortille.



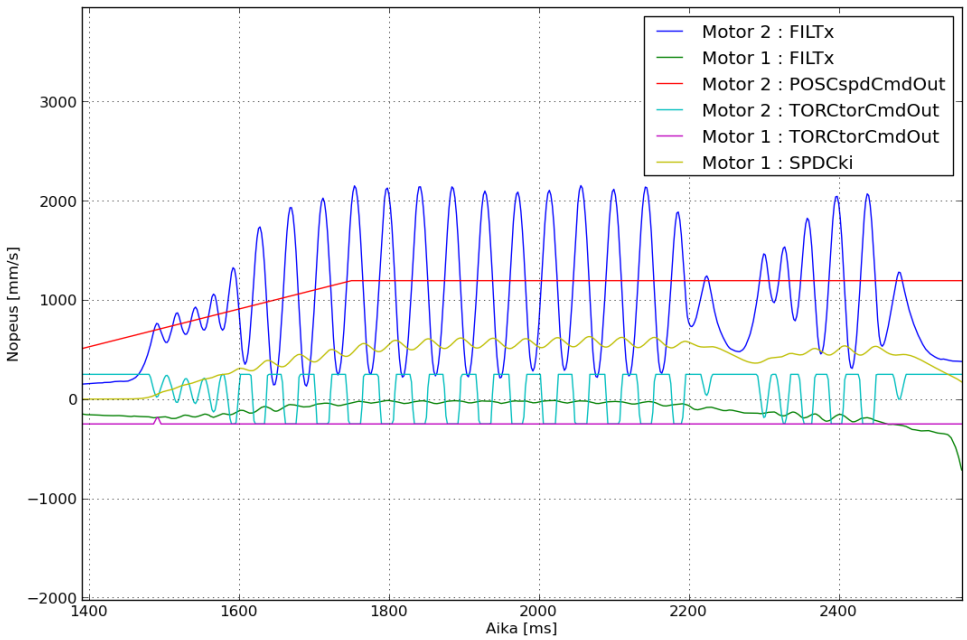
Kuva 5.13: Nostolaite keikkuu kiskolla ja moottorit luistavat vuorotellen.



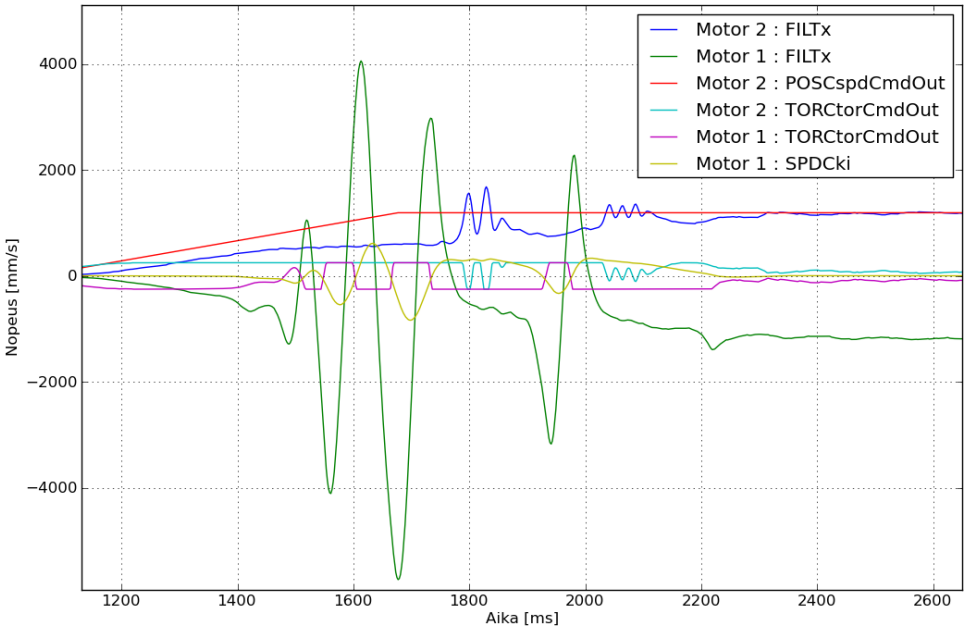
Kuva 5.14: Isäntämoottori luistaa.



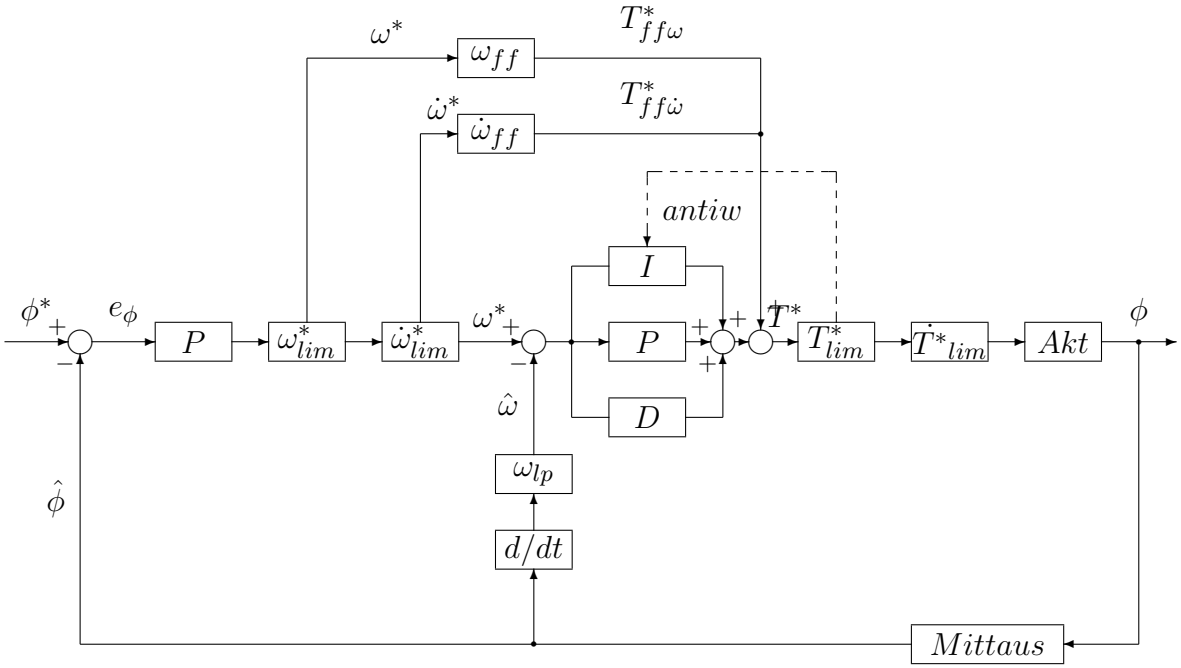
Kuva 5.15: Luistonesto.



Kuva 5.16: Isäntämoottori luistaa ja orja vetää täydellä momentilla.



Kuva 5.17: Orjamoottori luistaa ja isäntämoottori vetää.



Kuva 5.18: Lopullinen toteutettu säädin.

6. TULOKSET JA NIIDEN TARKASTELU

Tässä kappaleessa käydään läpi työn vaiheita ja tulokset, sekä arvioidaan käytettyjen menetelmien soveltuvuutta kyseisen ongelman ratkaisuun. Työn lopputuloksena oleva c-kielinen ohjelmakoodi on liitteessä B.

6.1 Koeajojärjestelmä

Työn kohteena olevasta ajoliikkeestä oli tiedossa paikoitustarkkuuden vaatimukset ja sallittu ajonopeus. Laitteen mekaanisista ominaisuuksista ei ollut paljoa tietoa, eikä siinä vaiheessa ollut vielä paljoa kokemusta asiakkaan tarpeistakaan. Tältä pohjalta päätettiin tehdä simulaattorilla vain säädinrakenteisiin liittyvää peruskehitystä ja tehdä mahdollisimman paljon ja aikaisessa vaiheessa oikealla laitteella.

Lähestymistapa osoittautui oikeaksi, sillä todellisen laitteen kanssa työskennellessä tulee esiin asioita, joita ei muuten tule huomanneeksi. Huonona puolena todellisen laitteen kanssa työskentelyssä on se, että kokeiluihin sopiva laite täytyy tietenkin olla jo olemassa. Laitteiston tiedonkeruujärjestelmä oli mahdollista toteuttaa Alteran fpga:lle softaprosessorina ja tiedonkeruujärjestelmästä pyrittiin tekemään sellainen että sitä voidaan käyttää tulevaisuudessakin. Lisäksi oikealla laitteella kehittäminen vaatii turvallisen ympäristön missä laitetta voidaan ajaa ilman että henkilövahingoista on pelkoa. Tähän tarkoitukseen rakennettiin lattiatasossa kulkeva törmäyssuojin varustettu ajorata. Se on osoittautunut hyväksi työkaluksi säädinkehityksen lisäksi muun muassa erilaisten mekaanisten ratkaisujen tutkimiseen.

6.2 Säädinparametrit

Koska laitteiston kaikkien liikkeiden mallintaminen ja mallin muuttujille numeroarvojen mittaaminen ja näiden avulla simuloiminen tuntui turhan työläältä vaihtoehdolta, täytyi säädinparametrien asettamiselle keksiä joku muu keino. Eri vaihtoehdoista valittiin ajoliikkeen identifiointiin relekoe, koska siinä ei ole ryntäilyn ja törmäilyn vaaraa. Laitteen kaikki liikkeet ovat liikematkaltaan hyvin rajoitettuja. Relekokeella voi siten olla käyttöä muidenkin järjestelmän liikkeiden säätimien virittämisessä.

Säädinparametrit haettiin kohdalleen siten, että alkutilanne katsottiin Zieglerin

ja Nicholsin laskemista taulukoista ja käsin iteroiden parannettiin säätimen suorituskyky halutulle tasolle. Taulukossa 6.1 on parametrit ZN:n mukaan ja tästä lähtökohdasta parannetut parametrit. Lopulliset parametrien asettelut täytyy tehdä todellisessa varastossa paketeilla, jotka ovat mahdollisimman lähellä asiakkaiden käyttämiä paketteja.

Taulukko 6.1: P ja I vahvistukset.

	G_i	G_p
ZN	0.56	0.97
Iteroitu	1.6	2.0

6.3 Luistonesto

Ajoliike koostuu kahdesta moottorista, jotka vetävät sukkulaa kiskolla eteenpäin. Sukkulan kuorman ollessa epätasapainossa sukkula kallistuu ja toinen rengas luistaa tai on jopa ilmassa. Säädinrakenteen tuli olla sellainen että pitävä rengas tästä huolimatta vetäisi normaalisti ja veisi sukkulan kohteeseensa. Työssä tehtiin säätimiin lisäkytkentä, joka moottorien nopeuseron mukaan lisää momenttia pitävälle pyörälle. Tällä järjestelyllä päästään ajamaan kohteeseen, mutta huonona puolena ilmeni ilmassa olevan renkaan äänekäs pyöriminen tai värähtely. Momenttia jakava säädinrakenne on hyvä olla olemassa, mutta renkaan nouseminen kallistuksessa täysin ilmaan on estettävä mekaanisesti.

7. JOHTOPÄÄTOKSET

Työssä tutustuttiin liikkeensäätöön, joka koostuu kahden säätimen kaskadista. Lisäksi tutustuttiin usean säätimen synkroniseen yhteisliikkeeseen. Työn tuloksena saatiin varmatoimiset liikkeenohjaukset varastorobotin kaikkiin liikkeisiin, joista esiteltiin tässä työssä ajoliike.

Säätimen kehittämisestä iso osa tehtiin laitteistolla, josta oli karsittu kaikki ajoliikkeeseen liittymätön pois. Tällä tynkävarastolla robotin kallistumista ja pyörien luistamista oli kätevää tutkia. Karsittu laitteisto aiheutti kuitenkin sen että ei päästy mittaamaan säätimen todellista suorituskykyä todellisilla paketeilla. Kuorman liikkuminen ja pyörien luistaminen erityisesti jarrutuksen aikana ovat asioita, jotka vaikuttavat oleellisesti paikoitustarkkuuteen ja transaktion lopulliseen kokonaisnopeuteen.

Ajoliike on kahden säätimen yhteinen liike. Orjasäädin tuottaa normaalitilanteessa saman momentin kuin isäntäsäädin. Ajoliikkeeseen rakennettiin luistonesto, joka siirtää momenttia sille pyörälle jolla on pitoa. Yleisemmin vastaavankaltaista säätimien ristiinkytkentää voidaan käyttää kahden liikkeen synkronisessa ajossa siten, että liikkeiden paikkojen erosta lasketaan jatkuvasti korjauskerrointa joka syötetään sopivasti suodatettuna kaskadin nopeustakaisinkytkennän summaimeen. Näin saadaan ratavirhe minimoitua.

Ehdollinen integrointi toimii hyvin vain ajoliikkeessä, jossa ei ole liikkeelle esteitä eikä muita yllättäviä pitkäkestoisia ja voimakkaita kuormitushäiriöitä, jotka aiheuttaisivat integraattorin kasautumisen. Muissa järjestelmän säätimissä voi olla syytä rakentaa tarkkailija, joka hidastaa integrointia tilanteissa, jossa ei pystytä seuraamaan komentoa.

Edelleen täytyy tutkia tilannetta jossa yhteistä liikettä tekevät säätimet ovat eri ohjainkorteilla. Tällöin kasvaa korttien välisen väylän tosiaika- ja nopeusvaatimukset. Selvitettävä on mitkä ovat radanseuraamisen tarkkuusvaatimukset kyseisissä tilanteissa ja mikä on yksinkertaisin väyläratkaisu, jolla tarkkuus saavutetaan.

LÄHTEET

[1] Altera Corporation [WWW]. [viitattu 1.10.2012] Saatavissa: http://www.altera.com/devices/fpga/cyclone3/cy3-index.jsp

[2] Changzhou Chuangwei Motor & Electric Apparatus Co., Ltd [WWW]. [viitattu 1.10.2012] Saatavissa: http://www.cw-motor.com/en/ProductShow.asp?ArticleID=6

[3] Dorf, Richard C., Bishop, Robert H. Modern Control Systems. 9th ed. USA 2001. Prentice Hall. 831 s.

[4] Ellis, George. Control system design guide, a practical guide. 3rd ed. Amsterdam Boston 2004, Elsevier Academic Press. 464 s.

[5] Hippe, Peter. Windup in Control, It's Effects and Their Prevention. Lontoo 2006. Springer. 305 s.

[6] Kiong, Tan Kok, Heng, Lee Tong, Sunan, Huang. Precision Motion Control, Design and Implementation. 2nd ed. Lontoo 2008, Springer. 272 s.

[7] O'Dwyer, Aidan. Handbook of PI and PID Controller Tuning Rules. 2nd ed. Lontoo 2006, Imperial College Press. 545 s.

[8] Olsson, H., Åström, K. J., Canudas de Wit, C., Gäfvert, M., Lischinsky, P. Friction Models and Friction Compensation, 1997. 37 s.

[9] Rob, Williams. Real-Time Systems Development. Oxford 2006, Elsevier. 320 s.

[10] Sainio, Veli-Matti. Hybridiaskelmoottorin mallinnus ja ohjaus nosturisovelluksessa. Diplomityö, Espoo 2005. Aalto-yliopisto teknillinen korkeakoulu. Elektroniikan, tietoliikenteen ja automaation tiedekunta. 80 s.

[11] Wescott, Tim. Applied control theory for embedded systems. Burlington 2006. Newnes. 303 s.

[12] Åström, Karl J., Hägglund, Tore. PID Controllers: Theory, design and tuning. 2nd ed. USA 1995, Instrument Society of America. 343 s.

[13] Åström, Karl J., Wittenmark, Björn. Computer Controlled Systems, Theory and design. 3rd ed. 1996 Prentice Hall. 557 s.

A. LAITTEEN TEKNISIÄ TIETOJA

Taulukko A.1: Säädön vaatimuksia.

Ajonopeus, normaali	1.25 m/s
Sallittu nopeuden ylitys	5 %
Paikoitustarkkuus	2 mm
Sallittu paikan ylitys	0 mm

Taulukko A.2: Moottorin tietoja.

Maksimimomentti	3 Nm
Maksimikierrosluku	1000 rpm
Inertia	0.00027 kgm^2

Taulukko A.3: Mekaniikan tietoja.

Ajopyörän halkaisija	100 mm
Ajopyörän inertia	0.000238 kgm^2
Vapaapyörän halkaisija	75 mm
Vapaapyörän inertia	0.0000984 kgm^2
Laitteen massa	50 kg

B. SÄÄTIMEN OHJELMAKOODI

B.1 Paikkasäädin

```
static void MOTORCONTROLLERPositionController(MOTORCONTROLLER* controller)
{
    // sisaisia temppeja
    float varSpdTmp = 0; // temp muuttuja nopeusohjeen laskun P osuudelle
    float varSpdCmd = 0; // temp muuttuja nopeusohjeen P ja I summalle
    // nopeuserosuureen lasku ja P
    //controller->POSCposErr =
     controller->POSCposCmdIn - controller->FBpos; // paikan erosuure
    controller->POSCposErr =
     controller->POSCposCmdSynch - (controller->FBposFixed);
    varSpdTmp = controller->POSCposErr * controller->POSCkp; // P
    varSpdCmd = varSpdTmp; // rajoittamaton valitulos
    // rajoitetetaan nopeusohjetta, sama nopeusrajoitus molempiin suuntiin
    if(varSpdCmd > controller->POSCspdLim)
    {
        varSpdCmd = controller->POSCspdLim;
    }
    else if(varSpdCmd < -controller->POSCspdLim)
    {
        varSpdCmd = -(controller->POSCspdLim);
    }
    // nopeusrajoitettu nopeusohje
    controller->POSCspdCmdOut = varSpdCmd;
    // rajoitetaan nopeusohjeen muutosta eli kiihtyvyytta
    controller->POSCaccCmd = (controller->POSCspdCmdOut -
     controller->POSCspdCmdOutEx) / controller->CNTRLlooptime;
    if((controller->POSCaccCmd > controller->POSCaccLim) &&
     (controller->POSCspdCmdOut > 0)) // kiihd pos nopeutta
    {
        //controller->POSCaccCmd = controller->POSCaccLim;
        controller->POSCspdCmdOut = controller->POSCspdCmdOutEx +
         (controller->POSCaccLim * controller->CNTRLlooptime);
    }
    else if((controller->POSCaccCmd > controller->POSCdecLim) &&
     (controller->POSCspdCmdOut <= 0)) // jarr neg nopeutta
    {
        //a->KiihdytysOhje = a->POSCaccLim * 2;
        controller->POSCspdCmdOut = controller->POSCspdCmdOutEx +
         (controller->POSCdecLim * controller->CNTRLlooptime);
    }
    else if((controller->POSCaccCmd < -controller->POSCdecLim) &&
     (controller->POSCspdCmdOut > 0)) // jarr pos nopeutta
    {
        //a->KiihdytysOhje = -a->POSCaccLim * 2;
        controller->POSCspdCmdOut = controller->POSCspdCmdOutEx -
         (controller->POSCdecLim * controller->CNTRLlooptime);
    }
```

```
    else if((controller->POSCaccCmd < -controller->POSCaccLim) &&
     (controller->POSCspdCmdOut <= 0)) // kiihd neg nopeutta
    {
        //a->KiihdytysOhje = -a->POSCaccLim;
        controller->POSCspdCmdOut = controller->POSCspdCmdOutEx -
         (controller->POSCaccLim * controller->CNTRLlooptime);
        //a->NopeusOhje = a->NopeusOhjeEx + a->KiihdytysOhje;
    }
    controller->POSCspdCmdOutEx = controller->POSCspdCmdOut;
    //controller->POSCaccCmdEx = controller->POSCaccCmd;
} // position controller
```

B.2 Nopeussäädin

```
static void MOTORCONTROLLERVelocityController(MOTORCONTROLLER* controller)
{
    // sisaisia muuttujia
    float varPSpd = 0; // P osuus
    float varDSpd = 0; // D osuus
    controller->SPDCspdErr = ( controller->SPDCspdCmdIn -
     controller->FBspd ) +
     controller->SPDCposErrComp; // nopeuden erosuure
    //controller->SPDCspdLowpass =
    // ( controller->SPDCspdErr - controller->SPDCspdErrEx );
    //controller->SPDCspdLowpass = controller->SPDCSpdLowpass +
    // ((controller->SPDCalpha / controller->CNTRLlooptime) *
    // (controller->SPDCspdErr - controller->SPDCSpdLowpass ) );
    varPSpd = controller->SPDCspdErr * controller->SPDCkp; // P osuus
    controller->SPDCiSum = controller->SPDCiSum +
     controller->CNTRLlooptime *
     (controller->SPDCki * controller->SPDCspdErr *
     controller->SPDCantiwTmp); // I osuus + antiw
    varDSpd = ( controller->SPDCspdErr - controller->SPDCspdErrEx ) *
     controller->SPDCkd; // d
    controller->SPDCtorCmdOut = varPSpd +
     controller->SPDCiSum + varDSpd;
} // velocity controller
```

B.3 Momenttisäädin

```
static void MOTORCONTROLLERTorqueController(MOTORCONTROLLER* controller)
{
    controller->TORCtorCmdOut = controller->TORCtorCmdIn +
        ( controller->POSCspdCmdOut * controller->TORCspdComp ) + // nop
        ( controller->POSCaccCmd * controller->TORCaccComp ) + // kiihd
        ( controller->TORCbias ); // maan vetovoiman kompensointi
    if(controller->TORCtorCmdOut > controller->TORCtorLim)
    {
        controller->TORCtorCmdOut = controller->TORCtorLim;
        controller->SPDCantiwTmp = controller->SPDCantiw; // antiw
    }
    if(controller->TORCtorCmdOut < -controller->TORCtorLim)
    {
        controller->TORCtorCmdOut = -controller->TORCtorLim;
        controller->SPDCantiwTmp = controller->SPDCantiw; // ei antiw
```

```
    }
    if (controller->CNTRLbrakeRelease == 1.0)
{
controller->SPDCantiwTmp = 1;
}
    else
    {
controller->SPDCantiwTmp = 0;
}
}
```

B.4 Liikkeen valmistumisen tarkkailu

```
static void MOTORCONTROLLERIsMovementDone(MOTORCONTROLLER* controller)
{
    // jos ollaan lahella ja pienella nopeudella niin ollaan kohteessa
    // in position
    if ( (controller->STATinPos == 0)
     && ((fabs(controller->POSCposErr) <
     controller->STATinPosDist) && (controller->FILTx <
     controller->STATinPosSpd)) ) // INPOS
    {
        controller->STATinPos = 1;
    }
    // running
    else if ( (fabs(controller->POSCposErr) >
     (controller->STATinPosDist + controller->STATinPosDistHyst)) )
    {
        controller->STATinPos = 0;
    }
    // antiwindup
    if (controller->STATinPos == 1)
    {
        controller->SPDCantiw = 0;
    }
}
```

B.5 Orjamoottorin momentin muodostus

```
// alipaastosuodatettu nopeusero. slaven taulukon parametrit kaytossa
m[DRIVE_SLAVEMOTOR].SPDCspdErr = (m[DRIVE_MASTERMOTOR].FILTx -
        m[DRIVE_SLAVEMOTOR].FILTx); // nopeusvirhe master-slave
m[DRIVE_SLAVEMOTOR].SPDCki = m[DRIVE_SLAVEMOTOR].SPDCki +
    (m[DRIVE_SLAVEMOTOR].SPDCkd * (m[DRIVE_SLAVEMOTOR].SPDCspdErr -
    m[DRIVE_SLAVEMOTOR].SPDCki));
float torq = m[DRIVE_MASTERMOTOR].TORCtorCmdOut +
    (m[DRIVE_SLAVEMOTOR].SPDCki * m[DRIVE_SLAVEMOTOR].SPDCkp);
m[DRIVE_SLAVEMOTOR].TORCtorCmdIn = torq;
MOTORCONTROLLERTorqueController(&m[DRIVE_SLAVEMOTOR]);
MOTORCONTROLLERSetOutputTorque(DRIVE_SLAVEMOTOR,
    m[DRIVE_SLAVEMOTOR].TORCtorCmdOut);
```

B.6 Relemittaus

```
void MOTORCONTROLLERAutoTuner(MOTORCONTROLLER* controller)
{
    if(controller->FILTx >= controller->CNTRLautoTunerHyst)
     {
         controller->TORCtorCmdOut = -(controller->TORCtorLim);
     }
     if(controller->FILTx < -(controller->CNTRLautoTunerHyst))
     {
         controller->TORCtorCmdOut = (controller->TORCtorLim);
     }
}
```