

Oliver Luukka

AUTOMAATTINEN PID-SÄÄDÖN VIRITYS TEOLLISUUSYMPÄRISTÖSSÄ

Diplomityö
Tekniikan ja luonnontieteiden tiedekunta
Tarkastajat: Yliopisto-opettaja Veli-Pekka Pyrhönen
Professori Matti Vilkkö
5/2025

TIIVISTELMÄ

Oliver Luukka: Automaattinen PID-säätimen viritys teollisuusympäristössä
Diplomityö
Tampereen yliopisto
Automaatiotekniikan DI-ohjelma
Toukokuu 2025

Teollisuuden automaattista koneohjausta toteutetaan vaihtelevissa prosessiympäristöissä. Automaatiosuunnittelun keskeisin vaatimus on prosessiymmärrys. Toisaalta säätötehtävä saattaa olla polttokammion puhallinohjaus tai yksinkertainen lämmitysjärjestelmän venttiilin ohjaus. Prosessi voi olla esimerkiksi viiveellinen, integroiva tai aikavakion dominoiva. Automaatiosuunnittelu on aina prosessikohtaista, mutta suunnittelutehtävää voidaan helpottaa yleispätevillä algoritmeilla. Automaattinen PID-säätimen viritys voidaan toteuttaa hyödyntämällä virityssäännöstöjä. Tässä työssä hyödynnetään automaattiviritysalgoritmia, jota voidaan hyödyntää 134 prosessimallia sisältävälle testimallijoukolle askelvastekokeella tunnistetun dynamiikan perusteella.

PID-säädin on teollisuuden yleisin säätömenetelmä. PID-säätimen komponentit ovat proportionaalivahvistin sekä integraatio- että derivaatiohaara. Säätimellä voidaan vaikuttaa prosessin ulostulon käyttäytymiseen prosessikohtaisesti. Teollisuusympäristössä PID-säädin toimii syklipurteisessa logiikkaohjaimessa. Tutkimustyö on toteutettu Beckhoffin TwinCAT -ympäristössä Structured text -pohjaisella lausekielellä.

Suljetun piirin PID-säätö perustuu mittausdatan käsittelyyn ja ohjauksen luomiseen automaatiologiikalla. Automaatiologiikka voi olla ohjelmoitava logiikka tai teollisuustietokone, joka ohjaa toimilaitetta kuten venttiilimootoria tai taajuusmuuttajaa. Ohjaus- ja mittaus tieto välitetään virta- tai jänniteviestinä digitaalisen ja analogisen rajapinnan välillä joko automaatiotähtäimen tai sähköjohtimen kautta.

Ohjelmointitehtävä vaatii ymmärrystä kenttälaitetasolta säätöpiirien matemaattiseen teoriaan. Logiikkaohjelmoinnissa ohjelmointityyli ja ohjelmistoarkkitehtuuri ovat yhä tärkeämpiä. Oikeaoppisella ohjelmointiparadigmojen hallinnalla voidaan tehostaa ohjelmointitehtävää ja tehdä ohjelmistoyksiköistä uusiokäytettäviä.

Tutkimustyössä toteutettu automaattiviritystoimilohko etsii prosessimallin parametrit K vahvistus, L viive, ja T aikavakio askelvastekokeen perusteella, ja tarjoaa rajapintansa kautta neljällä eri virityssäännöstellä lasketut PID-säätimen parametrit. Lambda- ja AMIGO-säännöstellä tehty viritys tarjoaa rauhallisen askelvasteen, siinä missä Ziegler-Nichols ja Cohen-Coon aggressiivisemmän vasteen. Työkalun käyttäjän tulee ymmärtää prosessia, jotta automaattivirittäjän tarjoamista parametreista voidaan valita kuhunkin prosessiin parhaiten sopivat parametrit. PID-säätimen suorituskykyä voidaan lisätä asetusarvopainotuksella, anti-windupilla ja derivaatiohaaran alipäästösuoituksella. Joissakin tapauksissa PI-säädin on PID-säädintä parempi valinta säätötehtävään.

Avainsanat: PID, KLT, Askelvastekoe, The test batch, Ziegler-Nichols, Amigo, Cohen-Coon, Lambda, Beckhoff, TwinCAT, Structured Text, OOP

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

ABSTRACT

Oliver Luukka: Automatic PID control tuning in industrial environments
Master of Science Thesis
Tampere University
Master's Programme in Automation Engineering
May 2025

Automatic machine control in industrial applications is implemented across various process environments. The most critical requirement in automation design is a deep understanding of the process itself. Control tasks may range from fan control in a combustion chamber to the regulation of a simple heating system valve. The process may exhibit characteristics such as delay, integrative behaviour, or dominant time constants. Although automation design is always process-specific, the design task can be supported by universal algorithms. Automatic PID tuning can be done by using tuning rules. The test batch of 134 process models, for which general-purpose tuning algorithms can be applied based on dynamics identified from step response tests is the basis for the tuning algorithm implemented in the work.

The PID controller is the most used control method in industrial automation field. The components of a PID controller include a proportional gain, an integrator, and a derivative operation. The controller influences the output behaviour of the process in a process-specific manner. In industrial environments, PID controllers are typically implemented within cycle-based logic controllers. The research work was conducted using Beckhoff's TwinCAT environment with the Structured Text programming language.

Closed-loop PID control relies on the processing of measurement data and the generation of control commands via automation logic. This logic can be implemented using a programmable logic controller (PLC) or a personal computer that controls an actuator such as a valve motor or a frequency converter. Control and measurement signals are transmitted as current or voltage signals across digital and analog interfaces, either via fieldbus systems or electrical wiring.

Programming such a system requires understanding from the field device level to the mathematical theory of control. In logic programming, the significance of programming style and architecture is increasingly emphasized. Mastery of correct programming paradigms can make programming more efficient and enable the reusability of software components.

In this research, the implemented automatic tuning function block identifies the process model's parameters K gain, L delay, and T time constant based on step response testing and offers PID controller parameters calculated using four different tuning rules. The Lambda and AMIGO methods yield a smoother step response, while Ziegler-Nichols and Cohen-Coon provide a more aggressive response. The user of the tool must understand the process to select the most suitable parameters provided by the auto-tuner for each specific application. PID controller robustness can be improved using setpoint weighting, anti-windup strategies, and low-pass filtering. In some cases, a PI controller is a better choice for a control task than a PID controller.

Keywords: PID, KLT, Step Response Test, The Test Batch, Ziegler-Nichols, Amigo, Cohen-Coon, Lambda, Beckhoff, TwinCAT, Structured Text, OOP

The originality of this thesis has been checked using the Turnitin OriginalityCheck service

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

- Ei
- Kyllä

Ilmoitukseni mukaan olen käyttänyt opinnäytteessäni tutkielmaprosessin aikana seuraavia tekoälysovelluksia: OpenAI GPT 3.5, Google Gemini 2.5 Pro

Käyttötarkoitus: Tekoälyä on käytetty tiedon jäsentelyn sekä omien teorioideni vahvistamisen tai kumoamisen työkaluna. Tekoälyn vastauksiin on jokaisessa kohdassa etsitty varmistava lähde tietokirjallisuudesta ymmärtäen tekoälyn epäluotettavuuden tieteellisenä lähteenä. Lisäksi tekoälyä on käytetty avointen lähteiden kohdalla tiedonhaun työkaluna ja olennaisen tiedon jäsentelyssä tiedonhaun nopeuttamiseksi. Tekoälyä on käytetty lukujen kaksi ja kolme teoriaosuuksien kirjoitusprosessin aikana. Useimmissa kysymyksissä on käytetty kahta tekoälysovellusta, niin että Geminille ja GPT:lle on annettu sama syöte. Geminin ja GPT:n vastaukset olivat usein yhdenmukaisia niissä tapauksissa, joissa tekoälyltä saatu vastaus on todistettu tieteellisistä lähteistä oikeaksi.

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien osat, joissa on hyödynnetty tekoälyä, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

ALKUSANAT

Tämä opinnäytetyö on toteutettu InSolution Oy:lle syksyllä 2024 – keväällä 2025 osana Tampereen yliopiston systeemitekniikan syventäviä opintoja. Tahdon kiittää InSolution Oy:n toimitusjohtaja Juha Katajistoa työni mahdollistamisesta, sekä työni ohjannutta Tampereen yliopiston opettajaa Veli-Pekka Pyrhöstä. Kiitos kuuluu myös esimiehelleni Juha-Pekka Suomalalle, joka mahdollisti diplomityön tekemiselle aikaa töiden lomassa, ja tarkasti työni InSolutionin puolelta.

Pirkkalassa, 20.5.2025

Oliver Luukka

SISÄLLYSLUETTELO

1. JOHDANTO	1
2. PID-SÄÄTIMEN VIRITTÄMINEN VIRITYSSÄÄNNÖILLÄ.....	3
2.1 PID Säädin	3
2.2 Monotonisen prosessin tunnuslukujen löytäminen askelvastekokeella ..	5
2.3 Testimallijoukko	7
2.3.1 Aikavakion dominoiva prosessi	7
2.3.2 Viiveen dominoiva prosessi.....	8
2.3.3 Integroiva prosessi.....	9
2.4 Virityssäännöt.....	9
2.4.1 Ziegler-Nichols	10
2.4.2 AMIGO.....	11
2.4.3 Cohen-Coon	13
2.4.3 Lambda.....	14
2.5 Säädön hyvyyslaskenta	14
3. DIGITAALINEN SÄÄTÖ	16
3.1 Digitaalinen PID-säädin	18
3.2 Asetusarvopainotus	19
3.3 Integraattorin windup-ilmiö ja anti-windup-toiminnot.....	20
3.4 Derivaattorin rajoitus alipäästösuodin	22
3.5 Beckhoff TwinCAT-ohjelmointiympäristö.....	22
3.6 Algoritmit ja arkkitehtuuri	23
3.6.1 Olio-ohjelmointi logiikkaohjelmoinnissa	23
3.6.2 Automaattiviritysalgoritmi	25
4. SÄÄTÖALGORITMIN VALIDOINTI SIMULAATIOYMPÄRISTÖSSÄ.....	27
4.1 TwinCAT-simulaattori	27
4.2 Simulink-simulaattori.....	28
4.3 Simulaatiosäädön tulokset.....	29
4.3.1 Aikavakion dominoiva prosessi	30
4.3.2 Viiveen dominoiva prosessi.....	33
4.3.3 Integroiva prosessi.....	35
4.3.4 Viiveellinen integroiva prosessi	36
5. SÄÄTÖTULOSTEN ANALYYSI	39
5.2 Säättömenetelmien vertailu	39
5.3 Johtopäätelmiä.....	41
6. YHTEENVETO.....	43
LÄHTEET	44
LIITE 1: SIMULAATIOSSA KÄYTETTY SIMULINK-TESTIYMPÄRISTÖ	46

LYHENTEET JA MERKINNÄT

ADC	Analog-to-digital converter muuttaa analogisen signaalin digitaaliseen muotoon digitaaliselle säätimelle.
AMIGO	Approximate MIGO on MIGO-menetelmän yksinkertaistettu versio, jossa säätimen parametrit määritetään approksimoitujen kaavojen avulla, jotta saavutetaan hyvä säätö ilman monimutkaisia laskelmia.
AO/AI	Analog output/input, logiikalta kentälle siirtyvä analoginen signaali, ja toisinpäin. Nimityksiä käytetään yleisimmin IO-moduulien yhteydessä
DAC	Digital-to-analog converter muuttaa digitaalisen signaalin analogiseen muotoon.
FIFO	First-in-first-out viittaa varastointitekniikkaan, jossa ensimmäiseksi varastoon tullut objekti poistuu myös ensimmäisenä.
IAE	Integrated absolute error, säädön hyvyyteen käytetty tunnusluku, joka perustuu virheen aikaintegraalin laskentaan
IEC	International Electrotechnical Commission, sähköalan standardointiorganisaatio.
IO	Input/Output, tiedonsiirtorajapinta kenttä- ja toimilaitteiden välillä
KLT	Vahvistus, viive ja aikavakio, ensimmäisen kertaluokan viiveellisen prosessidynamiikan tunnusluvut
MIGO	M-constrained Integral Gain Optimization. on säätimen viritysmenetelmä integraalivahvistuksen maksimoimiseksi, varmistaen että järjestelmän vaimennussuhde ja robustisuus säilyvät hyväksyttävällä tasolla.
NC	Normally closed on virtapiiri, joka on normaalisti kiinni eli virta kulkee siitä läpi.
NO	Normally open on virtapiiri, joka on normaalisti auki, jolloin sen läpi ei kulje virtaa.
OOP	Object-oriented programming. Ohjelmointiparadigma, jossa sovelus rakennetaan olioista, jotka yhdistävät tietorakenteen ja siihen liittyvät toiminnot.
PC	Personal computer, viittaa yleisesti kaikkiin tietokoneisiin, jotka ovat suoraan käyttäjänsä hallittavia.
PID	Proportional-Integral-Derivate, yleisin teollisuudessa käytetty säädintyyppi.

PLC	Programmable logic controller, Automaatioprosessien ohjaukseen käytettävä ohjelmitava logiikka eli tietokone.
ST	Structured text, logiikkaohjelmointiin tarkoitettu IEC 61131-3 -standardin lausekieli.
XAE/XAR	Extended automation environment/runtime, viittaa bekhoffin ohjelmointi ja suoritusaikaan TwinCAT ympäristössä.
ZOH	Zero-order hold on menetelmä, joka muuttaa diskreetin ohjaussignaalin jatkuva-aikaiseksi.

1. JOHDANTO

Adaptiivinen säätö voidaan mieltää ainakin kahteen yläkategoriaan; itsevirittyvä säätö ja mukautuvasti virittyvä säätö. Tässä työssä adaptiivisena säätimenä tarkastellaan itsevirittyvää säädintä, jossa säätimen parametrit etsitään ensimmäisestä asetusarvomutoksesta. Yhdesti virittyvä säädin on useimmissa sovelluksissa riittävä ratkaisu, kun prosessin dynamiikka ei oleellisesti muutu toiminta-alueellaan. Mukautuvan säädön sovellusympäristö on usein prosessiteollisuudessa useamman kertaluokan prosesseissa.

Tutkimus keskittyy PID-säätimen (proportional-integral-derivate) virittämiseen. PID-säädin on teollisuuden yleisin säädinratkaisu. Tässä tutkimustyössä säätimen viritysparemetrien laskenta perustuu virityssäännöstiihin, ja matemaattinen teoria pitkälti Karl J. Åströmin ja Tore Hägglundin teokseen Advanced PID Control vuodelta 2006 [1]. Åström ja Hägglund esittelevät luvussa 7 134 prosessimallia sisältävän testimallijoukon, joille voidaan hyödyntää yleispäteviä viritysalgoritmeja askelvastekokeella löydettyjen parametrien perusteella. Luvussa 2 esitetään matemaattinen teoria KLT (Vahvistus, viive, aikavakio) -tunnuslukujen löytämisestä ja PID-säätimen virittämisestä virityssäännöstiillä.

Luvussa 3 tarkastellaan digitaalista säätöä kenttälaitteen ja logiikan välisestä kommunikoinnista PID-säätimen digitaaliseen ja ohjelmalliseen toteutukseen. PID-säätimen viritin on toteutettu tutkimustyönä InSolution Oy:lle Beckhoff TwinCAT ympäristöön. InSolution Oy toteuttaa ohjelmistokehitystä, digitaalista koneohjausta ja teollisuusautomaatiota päätoimipaikkana Tampere. Prosessin ymmärtäminen on keskeisin vaatimus automaatiosuunnittelussa. Tutkimustyössä toteutettu algoritmi tarjoaa neljä eri vaihtoehtoista viritystä säätimelle. Luvussa 4 tarkastellaan simulaatiosäädön tuloksia ja simulaattorin toteuttamista TwinCAT- ja Simulink-ympäristössä neljälle eri prosessimallille.

Tämän työn yksi tavoite on toimia myös oppaana digitaalista viritystä tekeville. Digitaalisen säädön päälähdemateriaalina on käytetty oppaita 'Automaatiotekniikan perusteet' (A. K. Kippo & A. Tikka, 2008) ja 'Säätötekniikan perusteita' (J.

Savolainen & R. Vaittinen, 2007). Työssä perehdytään säätimen matemaattiseen teoriaan, mutta teollisuusautomaation osaaminen vaatii myös ymmärrystä kentälaitetason toimintaperiaatteista logiikkaohjelmointiin. Työn pyrkimys on yhdistää säätöteoria ja käytäntö mahdollisimman lukijaystävällisesti. Tästä syystä työssä tarkastellaan säätötehtävää matemaattisesta teoriasta alkaen ohjauksen muodostumiseen logiikalla, ja siitä eteenpäin virtaviestiksi toimilaitteelle, sekä takaisinkytkentätiedoksi anturilta logiikalle.

Luvussa 3 tarkastellaan myös olio-ohjelmoinnin ja ohjelmistoarkkitehtuurin merkitystä logiikkaohjelmoinnissa pääosin teoksen 'Mastering PLC Programming' (M. T. White, 2023) pohjalta. Työssä toteutettu virittäjä voidaan sulauttaa osaksi InSolution Oy:n ohjelmistoja, jossa yleispätevä algoritmi ja joustava ohjelmistoarkkitehtuuri ovat vaatimuksena mahdollistamassa virittäjän käyttämisen muuttuvissa teollisuusprosesseissa. Yksinkertaisuutta pidetään hyveenä ohjelmistoissa, siinä missä monimutkaisen ohjelman sanotaan hajoavan myös monimutkaisesti. Harold Abelson ja Gerald Jay Sussman kirjoittavat teoksessaan Structure and Interpretation of Computer Programs [18, xxii] seuraavasti: "*Programs must be written for people to read, and only incidentally for machines to execute.*", millä viitataan ohjelmointitavan merkitykseen. Ohjelmakoodin tulee olla sellaista, että vielä useammankin vuoden jälkeen se on ymmärrettävää muidenkin kuin toteuttajan näkökulmasta ylläpidon ja modernisoinnin mahdollistamiseksi.

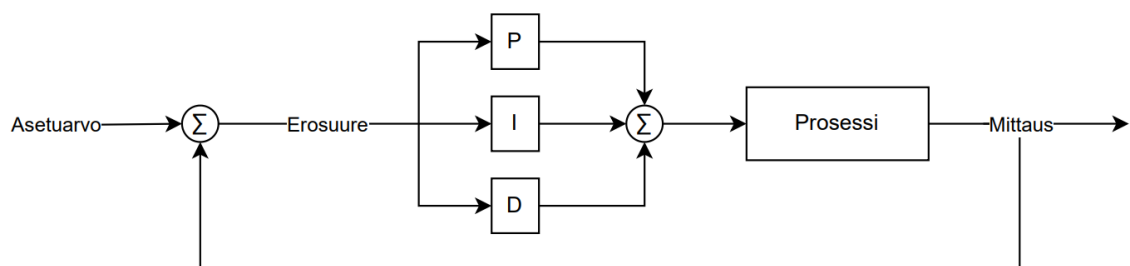
Integraattorin rajoitus, derivaattorin suodatus tai poisjättäminen, sekä asetusarvopainotus ovat työkaluja, joiden avulla voidaan tehdä säädöstä robustimpi yhdessä oikeiden viritysparettien kanssa. Koska virityssäännöt eivät pyri optimaaliseen viritykseen, on erityisen tärkeää tuntea prosessi mahdollisimman hyvin. Luvussa 5 tarkastellaan valittujen säätömenetelmien heikkouksia ja vahvuuksia kootusti. Toisinaan säädöstä halutaan nopeaa, toisinaan tärkeää on pelkkä robustisuus. Ohjelmistotestaamisen tavoite on löytää ohjausjärjestelmän heikkouksia. Tässä työssä viritysparettia on validoitu kahdella eri simulaattorilla. Todellisessa automaatiojärjestelmän käyttöönotossa joudutaan ominaisuuksia testaamaan usein iteratiivisesti, jotta varmistutaan niiden luotettavuudesta.

2. PID-SÄÄTIMEN VIRITTÄMINEN VIRITYSSÄÄNNÖILLÄ

2.1 PID Säädin

Säätötehtävät voidaan jakaa kahteen alaluokkaan. Avoimen piirin ohjaustehtävässä ohjataan laitetta niin, että prosessin ulostulo ei vaikuta säätimen sisääntuloon ainakaan suoraan. Esimerkiksi mikroaaltouunin virittäminen on avoin säätötehtävä, missä käyttäjä pyörittää kelloa arvioiden, mikä aika ruoan tulee olla mikroassa, jotta se on sopivan lämpöistä. Teollisuudessa säätötehtävät ovat usein suljettuja. Suljetun piirin säätötehtävässä hyödynnetään prosessin ulostulon mitausdataa, ja säädin reguloi automaattisesti mittaustuloksen ja asetusarvon välistä eroisuureta. Tästä esimerkkinä voidaan pitää vaikkapa auton vakionopeudensäädintä. [17]

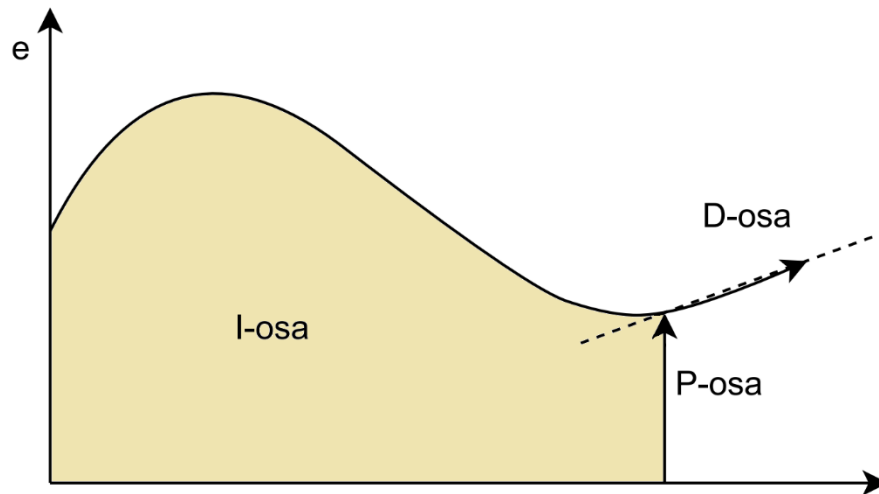
PID-säädintä käytetään perinteisesti suljetuissa säätöpiireissä. P-osa eli proportionaalinen säädin reagoi prosessin nykyiseen virheeseen. P-osa toimii vahvistimena ja ei näin ollen pysty korjaamaan kaikkia virheitä. Säätimessä käytetään usein I- ja/tai D-osa. I-osa (integraalisäädin) korjaa virheitä historiatiedon perusteella. D-osan (derivaattorisäädin) ennustaa tulevia virheitä ja sen tarkoitus on parantaa säädön tarkkuutta. [12]



Kuva 1. Suljetun piirin PID-säätimen alkeislohkokaavio

PID-säätimen parametrien vaikutus prosessin eroisuureeseen eli virheeseen voidaan kuvata aikatasossa seuraavasti. P-osa riippuu suoraan virheen hetkellisestä arvosta e ja reagoi siihen suoraan. I-osaan vaikuttaa virheen aikaintegraali, eli virheen kertymä. I-osan tavoite on poistaa pysyvä virhe. D-osa puolestaan

riippuu virheen muutosnopeudesta. Kuva 2 pyrkii havainnollistamaan miten PID-säätimen eri osat reagoivat reagoi virheeseen. [19, s. 120]



Kuva 2. PID-säätimen komponentit

Säätöparametrien laskennassa tasapainotellaan nopeuden ja luotettavuuden välillä. Kun proportionaalivahvistusta kasvatetaan, säätö nopeutuu, mutta muuttuu epästabiiilimmaksi. Integrointiajan kasvattaminen vastaavasti hidastaa säädintä, mutta parantaa luotettavuutta. Derivointiajan kasvattaminen voi lisätä säätöjärjestelmän vaimennusta ja siten parantaa vakautta, liian suuri derivointiaika voi hidastaa järjestelmän reagointia ja heikentää suorituskykyä. Derivaattori on herkkä komponentti ja nopeat muutokset tai mittauskohina voivat aiheuttaa ohjaukseen suuria muutoksia, kun derivointihaara on käytössä. [19]

Viiveelliset prosessit, joissa säätötoimenpiteet vaikuttavat prosessin lopputulokseen vasta viiveellä, voivat aiheuttaa haasteita säätöjärjestelmän suorituskyvylle. Tällöin PI-säätö (proportionaali-integratiivinen säätö) on usein suositeltavampi vaihtoehto PID-säätöön verrattuna. Viiveen vuoksi PID-säätöön kuuluvan derivaattivisen komponentin käyttö voi johtaa epävakauteen, koska se reagoi nopeasti prosessin muutoksiin, jotka eivät ole vielä vaikuttaneet lopputulokseen. PI-säätö puolestaan on yksinkertaisempi ja vakaampi, mutta mahdollisesti epätarkempi vaihtoehto. D-osa saattaa vahvistaa nopeiden häiriöiden tai muiden häiriöiden vaikutuksia. [1, s.166]

2.2 Monotonisen prosessin tunnuslukujen löytäminen askel-vastekokeella

Monotonisella prosessilla tarkoitetaan yhteen suuntaan kasvavaa tai vähenevää toimintaa. [2, s.268] Mikäli vaste on monotoninen, voidaan järjestelmän tunnuslukuja laskea sen vasteesta, esimerkiksi askelvastekokeella tai värähtelykokeella. Askelvastekokeella tarkoitetaan prosessille syötettävästä askelmuutoksesta syntyvän askelvasteen analysointia ajan funktiona,[3] mikä on käytännöllinen tapa tarkastella prosessin käyttäytymistä silloin kun prosessin matemaattista mallia ei tunneta. [5] Tässä luvussa esitetään erilaisia viritysääntöjä askelvastekokeen perusteella.

Prosessin vaste voidaan määrittää tulosignaalin funktion ja prosessidynamiikan siirtofunktion Laplace-muunnoksella. [2]

$$Y(s) = G(s)U(s), \quad (2.1)$$

missä $G(s)$ on siirtofunktio, $U(s)$ tulosignaalin Laplace-muunnos ja $Y(s)$ prosessin ulostulon Laplace-muunnos. Stabiilien monotonisen askelvasteen prosessien siirtofunktio voidaan esittää yksinkertaisena KLT-mallina [4, s.31]

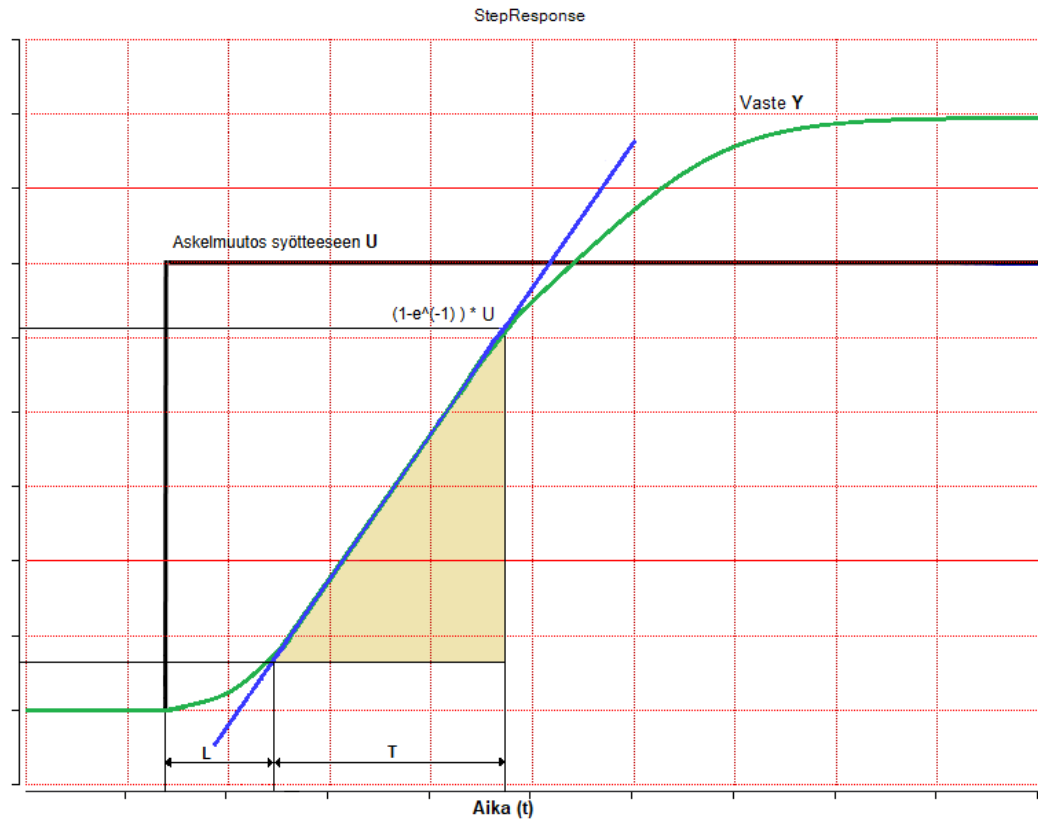
$$G_{\text{KLT}}(s) = \frac{K}{1+sT} e^{-sL}, \quad (2.2)$$

missä staattinen vahvistus $K \neq 0$, viive $L > 0$ ja aikavakio $T > 0$. Askelvastekokeen idea on tunnistaa KLT-parametrit prosessin vasteesta. Parametreja voidaan hyödyntää useissa heuristisissa säätöalgoritmeissa. [4]

Monotonisen ensimmäisen kertaluvun prosessin yksikköaskelvaste voidaan esittää aikatasossa seuraavasti. [3]

$$y(t) = \begin{cases} 0, & t < L \\ K(1 - e^{-t/T}), & t \geq L \end{cases} \quad (2.3)$$

Kuvassa 3 on esitetty KLT-parametrien määrittäminen monotonisesta askelvasteesta.



Kuva 3. *KLT-parametrien tunnistus askelvastekuvaajasta*

Askelvastekuvaajasta voidaan laskea prosessin vahvistus

$$K = \frac{Y(\text{asettunut}) - Y(\text{alussa})}{U(\text{lopus}) - U(\text{alussa})} = \frac{\Delta Y}{\Delta U} \quad (2.4)$$

Kuollut aika L lasketaan askelmuutoksesta siihen hetkeen, kun askelvaste saavuttaa jonkin pienen rajan [1], jossa vaste alkaa selvästi reagoida syötteen askelmuutokseen. Kuolleen ajan laskennan jälkeen määritetään aikavakio. Tarkastelemalla ajanhetkeä $y(T)$, saadaan kaava (2.3) muotoon:

$$y(T) = K(1 - e^{-1})u(t) \quad (2.5)$$

Kun vastesignaali on saavuttanut $1 - e^{-1} \approx 63,2\%$ loppuarvostaan askelmuutoksen jälkeen, saadaan prosessin aikavakion T suuruus selville vähentämällä tästä hetkestä kuollut aika L . [3]

Normalisoitu viive voidaan määritellä seuraavasti [1]

$$\tau = \frac{L}{T_{ar}} = \frac{L}{L+T} \quad (2.6)$$

τ rajautuu välille $[0, 1]$ ja sitä käytetään työkaluna prosessin säädettävyyden helpouden arvioimisessa, sekä onko säätötehtävässä kannattavaa käyttää D-osaa.

Suhteellinen viive voidaan esittää säädettävyyssindeksinä $\frac{L}{T_{ar}}$, missä T_{ar} on keskimääräinen viipymäaika. [1] Askelmainen vaste viittaa staattiseen prosessiin KLT-mallien kohdalla. [7] Säädettävyyseroimen perusteella voidaan päätellä, onko prosessi tasapainossa vai viiveen tai aikavakion dominoiva (ks. 2.2.1 ja 2.2.2). Tarkastellaan säädettävyyden indeksiä tarkemmin viritysäännösten kohdalla alaluvussa 2.4.

2.3 Testimallijoukko

Prosesseille, joille voidaan hyödyntää viritysäännöksiä, oletetaan karakterista olevan olennaisesti monotoninen askelvaste. Prosessit voidaan luokitella monotonisuusindeksin mukaan [1]:

$$\alpha = \frac{\int_0^{\infty} g(t) dt}{\int_0^{\infty} |g(t)| dt}, \quad (2.7)$$

missä g on prosessin impulssivaste. Mikäli $\alpha = 1$, niin systeemin askelvaste on monotoninen. Kun $\alpha > 0.8$ niin systeemin askelvasteen todetaan olevan olennaisesti monotoninen. Luvussa 2.3 esitettyjä viritysääntöjä voidaan hyödyntää 134 eri prosessimallille [1, s. 227]. Testijoukossa on viive- ja aikavakio-dominoivia sekä integroivia prosesseja, mikä soveltuu yleispätevän säätimen automaattivirtäjäen toteuttamiseen muuttuvissa teollisuusprojekteissa.

2.3.1 Aikavakion dominoiva prosessi

Kun prosessin normalisoitu viive on hyvin pieni tai se on nolla, eli prosessin viive on merkittävästi pienempi kuin aikavakio, puhutaan aikavakio-dominoivasta prosessista. Tällöin prosessin vaste määräytyy täysin sen aikavakion perusteella. Aikavakioita voi olla yksi tai useampi. Aikavakio-dominoivat prosessit ovat yleisiä prosessiteollisuudessa. Esimerkiksi suuri säiliö, johon virtaa nestettä sisään ja josta poistuu nestettä ulos, on tyypillinen esimerkki aikavakio-dominoivasta prosessista. Vaikka varsinaista kuollutta aikaa ei esiintyisi, pinnankorkeus muuttuu hitaasti johtuen säiliön suuresta tilavuudesta ja suhteellisen pienestä virtauserosta. Pinnanmuutosnopeus määräytyy sisään- ja ulosvirtauksen välisen erotuksen perusteella, mikä tekee järjestelmästä hitaan reagoimaan ohjausmuutoksiin. [1, s. 29]

Viiveettömän usean navan prosessin siirtofunktio on mallia:

$$G(s) = \frac{K}{(T_1s+1)(T_2s+1)\dots(T_ns+1)}, \quad (2.8)$$

missä n kuvaa aikavakioiden määrän. Mikäli jokin aikavakio on selvästi suurempi kuin muut, niin tällöin pienemmät aikavakiot vaikuttavat vain alkuvaiheen käyttäytymiseen ja voidaan usein jättää huomiotta mallinnuksessa tai säätimen virityksessä. Prosessin siirtofunktiota voidaan approksimoida ensimmäisen kertaluvun mallilla, jossa tämä suurin aikavakio määrää sen dynamiikan.

$$G(s) = \frac{K}{Ts+1}, \quad (2.9)$$

mikä on yhden aikavakion dominoivan prosessin siirtofunktio. Mitä suurempi prosessin aikavakio on, sitä hitaammin se reagoi muutoksiin ja sitä pidempään kestää ennen kuin prosessi saavuttaa uuden tasapainotilan. Kun useamman kertaluokan prosessia approksimoidaan ensimmäisen kertaluvun mallilla, voidaan mallia täydentää lisäämällä siihen viivetermi. Tämä viive huomioi ne dynaamiset vaikutukset, joita yksinkertaistettu malli ei pysty suoraan kuvaamaan, kuten korkeampien kertalukujen aikavakioiden yhteisvaikutuksesta syntyvä viivästynyt vaste.

2.3.2 Viiveen dominoiva prosessi

Viiveellisessä prosessissa on kiinteä aikaviive eli kuollut aika. Viiveen dominoivan prosessista puhutaan, kun aikaviive on merkittävä verrattuna aikavakioon. Viiveen dominoivia prosesseja voivat olla vaikkapa lämmitysjärjestelmät, jossa esimerkiksi huoneen lämpötila nousee vasta jonkin ajan kuluttua lattialämmityksen venttiilin avaamisen jälkeen. Viiveellisen prosessin tyypillinen siirtofunktio on kaavan 2.2 mukainen

$$G(s) = \frac{K}{1+sT} e^{-sL}, \quad (2.10)$$

Suuren viiveen tapauksessa säätötoimenpiteet saattavat tapahtua ennen kuin prosessi ehtii reagoida edellisiin muutoksiin. Esimerkiksi lämpötilan nousu ei tapahdu heti venttiilin avaamisen jälkeen, ja liian nopea säätö johtaa jatkuvaan yliohjaamiseen ja huojuntaan. Suuren viiveen prosesseissa säätimen virityk-

sessä on pienennettävä proportionaali-vahvistusta, hidastettava integraalitoimintaa ja usein derivatiiviosan rajoittamista tai poisjättämistä epävakauden välttämiseksi.

2.3.3 Integroiva prosessi

Integroiva prosessi ei ajaudu avoimen piirin tapauksessa tasapainotilaan, vaan kasvaa jatkuvasti tuloärsykkeestä sisäänmenon ollessa vakio. [1, s.35] Esimerkiksi kiinteistön kiertovesiverkosto voidaan mieltää integroivaksi. Kun kaukolämmön venttiili avataan kiertoveden lämmönsiirtimelle, kasvaa verkoston lämpötila teoriassa rajattomasti ilman ulkoista vaikutinta. Prosessi käyttäytyy integroivasti, vaikka todellisuudessa kiertovesiverkoston lämpötila ei voisi kasvaa paljon suuremmaksi kuin kaukolämpöverkoston lämpötila.

Integroivan prosessin siirtofunktio on muotoa

$$G(s) = \frac{K_v}{s}, \quad (2.11)$$

missä $1/s$ on integraattorin siirtofunktio ja K_v on vahvistus, joka kuvaa, kuinka nopeasti ulostulosignaali kasvaa lineaarisesti sisääntulosignaalin yksikköä kohden. Usein integroivan prosessin siirtofunktioon lisätään aikavakiotermi ja viive-termi kuvaamaan yksinkertaisia järjestelmiä, jolloin siirtofunktio saadaan muotoon. [1]

$$G(s) = \frac{K_v}{s(1 + sT)} e^{-sL}, \quad (2.12)$$

joka on viiveellinen integroiva toisen kertaluvun prosessimalli.

2.4 Vurityssäännöt

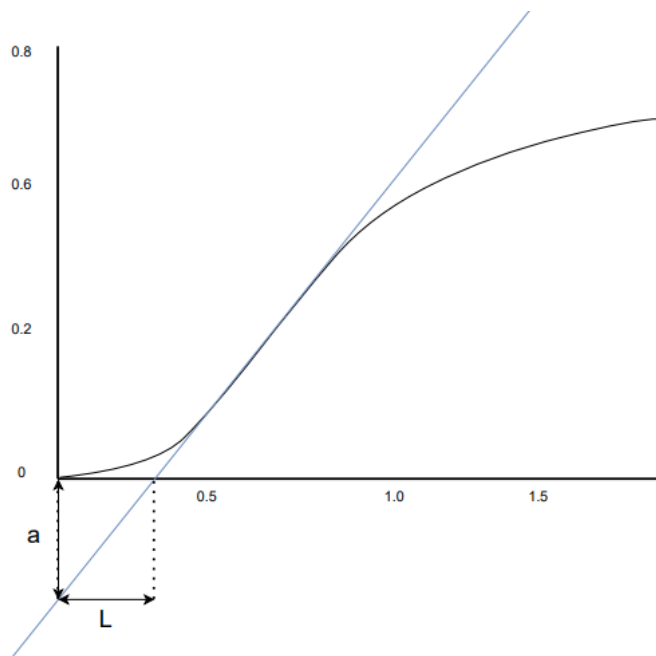
Säätötekniikassa vurityssäännöt ovat systemaattisia menetelmiä, joilla määritetään säätöpiirin parametrit ennalta lasketuilla kertoimilla. Vurityssäännöt eivät sinänsä tuota optimaalisia parametreja, vaan tarjoavat heuristisia ratkaisuja säätöparametrien asettamiseksi siten, että järjestelmä saavuttaa riittävän hyvän suorituskyvyn, jota voidaan tarvittaessa hienosäätää iteratiivisesti kokeilujen ja järjestelmän vasteen analyysin perusteella. Tunnetuimpia vurityssääntöjä ovat Ziegler-Nicholsin ja Cohen-Coonin menetelmät, tässä työssä tarkastellaan lisäksi Amigo ja Lambda -säännöstyöjä.

2.4.1 Ziegler-Nichols

Ziegler-Nichols säätöalgoritmi on yksi heuristinen metodi PID-säätimen virittä-
miseksi. John G. Ziegler ja Nathaniel B. Nichols esittelivät vuonna 1942 *avoimen
ja suljetun piirin viritysmenetit*. [4, s. 52]. Ziegler-Nichols-menetelmän taustalla
on viiveellinen integroiva prosessimalli [1, s. 49]

$$G(s) = \frac{K_v}{s} e^{-sL} = \frac{a}{sL} e^{-sL} \quad (2.13)$$

Prosessin karakteristiset ominaisuudet saadaan sovittamalla tangentti avoimen
piirin systeemin vasteeseen kuvan 4 mukaan. Tangentti asetetaan suurimman
kaltevuuden kohtaan vastekäyrällä, ja kun tangentti piirretään akselistosta yli, voi-
daan laskea tunnusluvut a ja L . [4]



Kuva 4. Avoimen piirin systeemin askelvasteen karakterisointi

Kuvasta 4 voidaan määrittellä a KLT-parametrien avulla [1, s.225]

$$a = \frac{KL}{T}, \quad (2.14)$$

jolloin PID-säädinparametrit voidaan perinteisesti laskea seuraavilla kaavoilla [1, s. 165]

$$K_P = \frac{1.2}{a}$$

$$T_i = 2L \quad (2.15)$$

$$T_d = \frac{2}{L}$$

Vastaavasti PI-säätimen viritysparametrien ehdot ovat

$$K_P = \frac{0.9}{a}$$

$$T_i = 3L, \quad (2.16)$$

missä $a = K_v L$ integroiville prosesseille. [1, s.165] Ziegler-Nichols-menetelmän huono puoli on sen rajoitettu sovellettavuus, sillä se sopii vain tietynlaisille stabiileille prosesseille kriittisen virityksensä takia. [6] Virityssäännöillä tehty säätö ei huomioi koskaan stokastista dynamiikkaa, vaan toimii parhaiten deterministisille prosesseille. Ziegler-Nichols-säännöstö voi kuitenkin antaa joillekin prosesseille riittävän hyvän perussäädön. [5]

Säätötehtävän tulee kuitenkin huomioida esimerkiksi häiriötä, viivettä ja mittauskohinaa. Siinä missä säädöltä halutaan nopeutta ja tarkkuutta, voidaan sen yhdeksi keskeisimmäksi hyveeksi mieltää myös robustisuus, eli häiriökestävyys. Kun virityssäännöissä hyödynnetään enemmän prosessista kerättyä informaatiota kuin Ziegler-Nichols-säätöalgoritmi huomioi, voidaan säätimen robustisuutta kasvattaa [1]

2.4.2 AMIGO

Virityssäännöt eivät pyri tarjoamaan optimaalista suorituskykyä säätimelle, kuitenkin robusti säätö voidaan toteuttaa suorituskyvyn alentamisella. [1] AMIGO (Approximate MIGO) on KLT-parametreille soveltuva säännöstö PID-säätimen virittämiseksi. MIGO (M-constrained Integral Gain Optimization) menetelmä perustuu prosessin keskeisten elementtien yhdistämiseksi teoreettiseksi prosessimalliksi [1, s.217]. Tässä työssä tarkastellaan vain yksinkertaistettua konservatiivista AMIGO-säännöstöä. Konservatiivisen säädön ajatus on pienentää säätimen vahvistusta ja kasvattaa integraaliaikaa, mikä hidastaa säätöä, mutta tekee siitä samalla robustimman. PID-säätimen viritykselle ehdotetaan seuraavaa AMIGO-säännöstöä [1, s. 233]

$$K_P = \frac{1}{K} \left(0,2 + 0,45 \frac{T}{L} \right)$$

$$T_i = \frac{0,45+0,8T}{L+0,1T} L \quad (2.17)$$

$$T_d = \frac{0,5LT}{0,3L + T}$$

Integroivissa prosesseissa AMIGO-säännöstölle ehdotetaan vaihtoehtoisia virityssääntöjä:

$$K_P = 0,45/K_v$$

$$T_i = 8L \quad (2.18)$$

$$T_d = 0.5L$$

PI-säädin voidaan virittää AMIGO-säännöstöllä seuraavasti: [1, s.228]

$$K_P = \frac{0.15}{K} + \left(0,35 \frac{LT}{(L+T)^2}\right) \frac{T}{KL}$$

$$T_i = 0,35L + \frac{13LT^2}{T^2+12LT+7L^2} \quad , \quad (2.19)$$

ja integroivan prosessin ehdotetut PI-viritysparametrit ovat mallia [1, s.229]:

$$K_P = \frac{0,35}{K_v L}$$

$$T_i = 13,4L \quad (2.20)$$

Tarkastelemalla kaavan 2.6 mukaista suhteellista viivettä, voidaan arvioida AMIGO-säännöstön soveltuvuutta säädettävälle prosessille. AMIGO-virityssäännöt asettavat säätimen vahvistuksen K_P varsin tarkasti säätöjärjestelmän suorituskyvyn näkökulmasta, kun normalisoitu viive $\tau > 0.3$, mutta muilla alueilla vahvistus jää usein liian pieneksi. Integraaliajalle laskenta onnistuu puolestaan hyvin, kun $\tau > 0.2$. Derivaatiotoiminnon katsotaan olevan hyödyllinen, kun $\tau > 0.5$ tai $\tau < 0.3$. Sen sijaan alueella $0.3 < \tau < 0.5$ AMIGO-säännöt voivat antaa selvästi liian lyhyen tai pitkän derivaatioajan verrattuna MIGO-menetelmään, ja todellinen optimaalinen T_d voi olla jopa kaksinkertainen AMIGOn antamaan arvoon verrattuna. Jos tällä alueella käytetään AMIGO-säännön mukaisia derivaatioaikoja, säätimen robustisuus heikkenee, ja suorituskyky voi pienentyä. Robustisuus säätöjärjestelmässä ilmaistaan usein maksimiherkkyysfunktioilla (maximum sensitivity, M), joka kuvaa järjestelmän herkkyyttä mallin epävarmuudelle ja häiriöille. Järjestelmän häiriönsietokyky voi laskea noin 15 % tällä alueella lasketuilla derivaatio-

ajoilla. Tästä huolimatta AMIGO tarjoaa yleensä konservatiivisen virityksen kaikille testimallijoukossa esitetyille prosesseille, mikä johtuu pienemmästä säätimen vahvistuksesta ja pidemmästä integraaliajasta. [1]

2.4.3 Cohen-Coon

Cohen-Coon säännöstö on 1953 esitelty viritysmenetelmä. Virityssäännöstö huomioi Ziegler-Nichols-menetelmän kriteerit häiriölle, mutta se eroaa kuitenkin Ziegler-Nichols-menetelmästä, sillä virityssäännöissä on huomioitu kaavan 2.6 suhteellinen viive. [6] Tämä tekee virityksestä paremman viiveellisissä prosesseissa, mikä voi mahdollistaa nopeamman säätövasteen pienemmällä yliojauksella. Cohen-Coon-virityssäännöstö määrittää PID-säätimelle seuraavasti. [1, s. 168]

$$\begin{aligned}
 K_p &= \frac{1.35 \left(1 + \frac{0,18\tau}{1-\tau}\right)}{a} \\
 T_i &= \frac{2,5-2,0\tau}{1-0,39\tau} L \\
 T_d &= \frac{0,37 - 0,37\tau}{1 - 0,81\tau} L
 \end{aligned}
 \tag{2.21}$$

Cohen-Coon-säännöstön mukainen PI-säädin voidaan virittää kaavoilla:

$$\begin{aligned}
 K_p &= \frac{0.9 \left(1 + \frac{0,092\tau}{1-\tau}\right)}{a} \\
 T_i &= \frac{3,3-3,0\tau}{1+1,2\tau} L
 \end{aligned}
 \tag{2.22}$$

Pienillä τ arvoilla Cohen-Coon-säännöstön todetaan tuottavan hyvin samankaltaiset parametrit Ziegler-Nichols-virityssääntöjen kanssa. Kun taas suhteellisen viive τ lähestyy arvoa 1 huomataan vahvistuksen K lähestyvän ääretöntä. Virityksellä on myös heikko kyky vaimentaa asetusarvomuuosvastetta, mikä altistaa järjestelmän värähtelyille ja heikentää sen häiriönsietokykyä. [1, s.168]

Cohen-Coon-viritys perustuu osaksi Ziegler-Nicholsin perustana toimivaan neljännesvaimennukseen, jonka teoreettinen tarkoitus on vaimentaa suljetun piirin askelvastetta riittävän nopeasti mutta ei liian aggressiivisesti. Amplitudi pienenee

jokaisen peräkkäisen ylityksen kohdalla neljäsosaan edellisestä. Neljännesvaimennus ei ole kuitenkaan optimaalinen vaimennuskriteeri, eikä näin ollen tarjoa robustia suljetun piirin säätöä. [1]

2.4.3 Lambda

Lambda-säännöstö on prosessiteollisuudessa yleisesti hyödynnetty viritysmenetelmä, jolle voidaan kirjoittaa viritysparametrit KLT-mallin perusteella seuraavasti:

$$K_P = \frac{1}{K} \frac{L/2 + T}{L/2 + \lambda}$$

$$T_i = T + L/2 \tag{2.23}$$

$$T_d = \frac{TL}{L + 2T}$$

missä λ on tavoiteaikavakio, joka vaikuttaa suoraan prosessin vahvistukseen. [1, s.189] Suurempi λ valinta saa aikaan hitaamman reagoinnin, mutta vakaamman vasteen. Vastaavasti pienempi λ tarkoittaa suurempaa vahvistusta eli nopeampaa reagointia. Lambdan valinta on kriittinen, ja robustille säätimelle suositellaan valintaa $\lambda=T$. Mikäli virityksestä halutaan aggressiivinen, voidaan lambda valita seuraavasti $\lambda=3T$. Viiveen hallitsevissa prosesseissa on joskus suositeltavaa valita T_i arvoksi suurempi arvoista T ja $3L$. Puhtaasti aikaviiveellisissä prosesseissa T valitaan nolaksi. [1, s.187]

Lambda-säätö soveltuu parhaiten hitaasti reagoiville aikavakio-dominoiville prosesseille. Epävakaat, ei-minimivaiheiset prosessit, prosessit voivat vaatia muita säätömenetelmiä. [1]

2.5 Säädön hyvyyslaskenta

Säätöpiirin aikavasteen perusteella voidaan arvioida säädön hyvyttä. IAE (Integrated absolute error) laskee erosuureen itseisarvon aikaintegraalia eli kumulatiivista summaa. IAE voidaan esittää seuraavasti: [1]

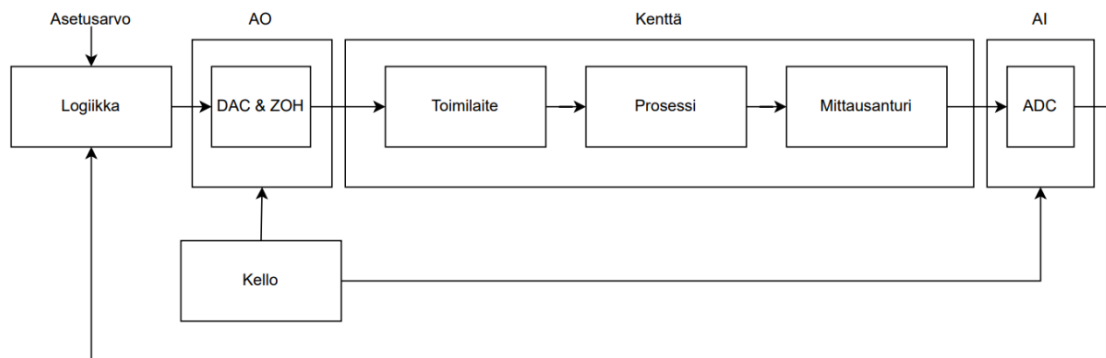
$$IAE = \int_0^{\infty} |e(t)| dt \tag{2.24}$$

Säätö toteutetaan luvussa 4 simulaatiomalleilla ja luvussa 5 tarkastellaan luvussa 2.4 esitettyjen säätöparametrien hyvydestä suhteessa luvussa 2.3 esitettyihin

prosessimalleihin IAE-menetelmällä. IAE-menetelmä ei pelkkänä lukuna ota kantaa säädön onnistuneisuuteen, sillä esimerkiksi hitaat ja stabiilit säädöt poikkeavat asetusarvosta, siinä missä nopea säätö suurella ylityksellä voi erosuureen kumulatiivisesta summaltaan olla pienempi. Kuitenkin hyvin suuret IAE-luvut kertovat aina epäonnistuneesta parametrien virityksestä ja/tai luvuissa 3.2, 3.3 ja 3.4 esiteltyjen työkalujen laiminlyönnistä.

3. DIGITAALINEN SÄÄTÖ

Digitaalisella säädöllä tarkoitetaan ohjelmitavalla logiikalla, mikrokontrollerilla tai teollisuustietokoneella tehtyä säätöä. Tarkastellaan jatkuva-aikaisen prosessin ohjauslogiikkaa, joka on esitetty kuvassa 5. Ohjelmitava logiikka on diskreetti digitaalinen järjestelmä, joka suorittaa komentoja järjestyksessä määrätyllä sykli-ajalla. ZOH (Zero-order hold) tekniikka pitää näytearvon vakiona seuraavaan näytteenottohetkeen asti, ja yhdessä DAC (Digital-to-analog converter) muunnoksen kanssa mahdollistaa portaikkomaisen signaalin lähettämisen analogiseen jatkuva-aikaiseen järjestelmään. ADC (Analog-to-digital converter) puolestaan muuntaa analogisen mittauksen digitaaliselle säätimelle sopivaksi signaaliksi [8, s. X]



Kuva 5. Yksinkertaistettu lohkokaavio digitaalisesta säädöstä

Tietokonesäätö perustuu kello-ohjaukseen. Eli ohjelma suorittaa jokaisella sykli-ajalla järjestyksessä sille määritetyt tehtävät. Yksinkertaistettuna säätötehtävän proseduuri on seuraava [9, s.40].

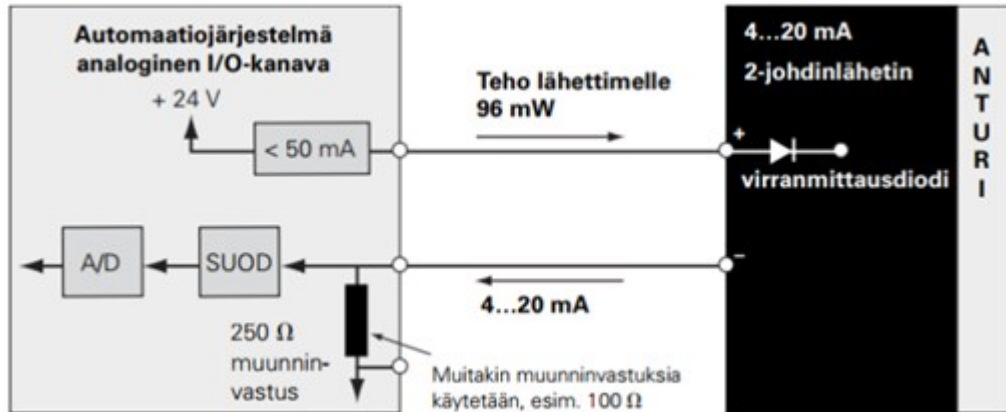
1. Odota kellopulssin liipaisua
2. Tee mittaus ja muuta se digitaaliseen muotoon (ADC)
3. Suorita järjestyksessä säätöalgoritmit ja laske säädölle arvo
4. Muuta ohjaus analogiseen muotoon (DAC & ZOH)
5. Suorita ohjauskomento
6. Palaa kohtaan yksi

Teollisuuden säätötehtävissä IO (Input/Output) moduulit toimivat analogia-digitaalirajapintana. IO-rajapinta käsittää erilaiset logiikan ja kenttälaitteiden välisen keskustelusäännöstön. Kuvassa 5 DAC ja ADC on esitetty AO (Analog output) ja AI (Analog input) rajapintoina. Kuvassa 6 on kuvattu analogisen mittauspiirin 2-johdinkytkentä. Anturilta tuleva virtaviesti muunnetaan analogisessa IO-kanavassa digitaaliseen muotoon.

Pääosin anturi lähettää virtaviestin alueella 4–20 mA, muitakin alueita käytetään kuten 0–20 mA, tai muita erityistapauksia kuten pneumaattinen- tai jänniteviesti. Virta-alue 4–20 mA on kuitenkin yleisin, sillä samalla kaksijohdin kytkennällä voidaan tällöin välittää anturilähettimele virta, ja lukea mittaustieto. Virransyöttö ei onnistuisi 0 mA kytkennällä. Jännitekytkennän heikkous on mahdolliset jännitehäviöt johtuen esimerkiksi pitkän kaapelointimatkan aiheuttamasta resistanssista. [1] Prosessiviestin eläväksi nolaksi kutsutaan alinta mahdollista arvoa (4–20mA tapauksessa 4mA), joka ei ole nolla. Kuollut nolla on vastaavasti 0 mA, joka viittaa häiriötilanteeseen. Elävän nollan hyödyt ovat vikojen tunnistuksessa, tarkkuudessa ja standardoinnissa. Virtaviesti 4–20 mA voidaan laskea kaavan 3.1 mukaan [9]

$$I = 4mA + \frac{B}{A} \cdot 16mA, \quad (3.1)$$

jossa A on mitattavan suureen vaihtelualue ja B mittaussuureen arvo. [9] Virtaviesti skaalataan tietokoneohjelmassa luettavaan ja IO-kortin määräämään kokonaislukumuotoon (integer). Skaalauksen sisääntulon ylä- ja alarajat määräytyvät muuntimen mukaan välille $-2^{N-1} - 2^{N-1} - 1$, missä N on bittien eli tiedon määrä. Esimerkiksi 16-Bittisen integerin suurin mahdollinen arvo on $2^{15} - 1 = 32767$, tällöin viesti rajautuu alueelle $-32768 - 32767$. Vastaavasti etumerkitön kokonaisluku (unsigned integer) rajautuu välille $0 - 2^N - 1$, sillä yhtäkään bittiä ei varata etumerkille [10, s.41–42]. Skaalauksen ulostulo määritellä mittaushetkellä, mikäli mitaus olisi esimerkiksi ilman happipitoisuus, tällöin anturilta luettu $20 \text{ mA} = 32767$ olisi ilman maksimihappipitoisuus 21 % [13], ja $4 \text{ mA} = -32767$ tarkoittaisi 0 % happea.



Kuva 6. Analogisen mittauspiirin 2-johdinkytkentä [9, s.50]

Säätöviesti on vastaavasti ohjauslogiikalta toimilaitteelle välitetty viesti. Ohjauslogiikka keskustelee monissa sovelluksissa väylän kautta, joka muuntaa ohjausarvon moottorikohtaisen taajuuden mukaan. Esimerkiksi venttiilimoottoreita voidaan ohjata suoralla analogialähdöllä, jolloin logiikkaa muuttaa vastaavasti halutun ohjaussuureen virtaviestiksi. Standardiviestiyhteys ohjausvirtapiirissä on myös 4–20 mA. NO (Normally closed) sovelluksissa 0–4 mA vastaa täysin suljettua venttiiliä. Vastaavasti NC (Normally open) eli auki ilman virtaa, jolloin ohjauksen arvolla 4 mA on venttiili auki. [11]

3.1 Digitaalinen PID-säädin

Analogisen PID-säätimen ohjausfunktio aikatasossa voidaan esittää jatkuvassa muodossa:

$$u(t) = K_P \left(e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_D \frac{de(t)}{dt} \right), \quad (3.2)$$

missä $e(t)$ on erosuure. Mikäli PID-säädin halutaan toteuttaa digitaalisessa muodossa, tulee sitä käsitellä aikadiskreetissä epäjatkuvassa muodossa, sillä digitaalinen säätö perustuu näytteenottoihin. Integraaliosuus saadaan epäjatkuvaan muotoon [11]

$$\int_0^t e(t) dt = \sum_{i=0}^n e_i T_A, \quad (3.3)$$

missä jokaisella näytteenottovälillä T_A päivitetään erosuureen kumulatiivista summaa. Epäjatkuva derivaatan laskenta-algoritmi on seuraava. [11]

$$\frac{de(t)}{dt} = \frac{e(k) - e(k-1)}{T_A}, \quad (3.4)$$

jossa tarkastellaan erosuureen muutosta näytteenottovälin aikana. PID-ohjausfunktio saadaan muotoon:

$$u(k) = K_p \left[e(k) + \frac{T_A}{T_i} \sum_{i=0}^{k-1} e(i) - \frac{T_D}{T_A} (e(k) - e(k-1)) \right] \quad (3.5)$$

Integraaliosan summalauseke on kuitenkin algoritmillisesti haastavaa, joten säätöalgoritmia kannattaa yksinkertaistaa. Tarkastellaan edellistä ohjelmasykliä $u(k-1)$, jolloin ohjaussignaali voidaan laskea seuraavalla kaavalla:

$$u(k-1) = K_p \left[e(k-1) + \frac{T_A}{T_i} \sum_{i=0}^{k-2} e(i) - \frac{T_D}{T_A} (e(k-1) - e(k-2)) \right] \quad (3.6)$$

tekemällä vähennys $u(k) - u(k-1)$ saadaan ohjausfunktio $u(k)$ muotoon:

$$u(k) = u(k-1) + K_p \left[\left[1 + \frac{T_D}{T_A} \right] e(k) + \left[-1 + \frac{T_A}{T_i} - \frac{2T_D}{T_A} \right] e(k-1) + \left[\frac{T_D}{T_A} \right] e(k-2) \right], \quad (3.7)$$

jossa $u(k-1)$ on edellinen ohjausarvo. Ohjausarvolaskennan jälkeen tallennetaan $e(k-2)$ vastaavaan muuttujaan edellisen syklin $e(k-1)$, ja vastaavasti $e(k-1)$ vastaavaan muuttujaan edellisen syklin erosuure $e(k)$. Tällöin algoritmillisesti ei tarvita kuin erosuureen $e(k)$ laskenta jokaisella ohjelmasyklillä. [11]

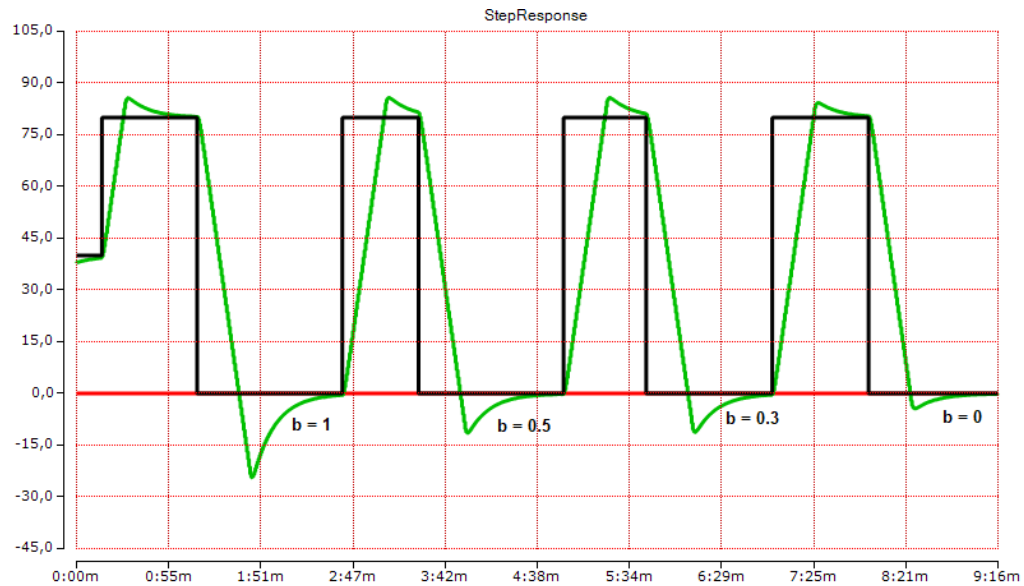
3.2 Asetusarvopainotus

Suljettu ohjauspiiri perustuu virheen regulointiin. Erosuure on asetusarvon ja ulostulon erotus. PID-säädössä voidaan käsitellä P, I ja D-osan virheet erikseen. Tarkastellaan virheitä aikatasossa

$$\begin{aligned} e_p &= b \cdot r(t) - y(t) \\ e_d &= c \cdot r(t) - y(t) \\ e_i &= r(t) - y(t) \end{aligned} \quad (3.8)$$

missä e_p on proportionaaliosan, e_d derivaattaosan, ja e_i integraaliosaan vaikuttava säätövirhe, $r(t)$ asetusarvo ja $y(t)$ prosessin ulostulo. b ja c painokertoimilla voidaan vaikuttaa P- ja D-osan reagointiin asetusarvomuutoksissa. Suuremmilla

b-arvoilla ylitys kasvaa. [1, s. 75] Kuvassa 7 on esitetty b-painokertoimen vaikutus lambda-viritetyn ohjauspiirin ylitykseen.



Kuva 7. Proportionaaliosan painokertoimen vaikutus lambda-viritykseen aikavaikio-dominoivassa prosessissa

Derivaatiohaaran painokerroin c valitaan usein nolaksi, koska derivaatiohaaran tavoitteena on vaimentaa häiriötä eikä reagoida siihen aggressiivisesti. Nopeasti muuttuvat asetusarvomuutokset aiheuttaisivat derivaatiohaarassa suuren ohjauspiikin, mikä johtaisi ylihjaukseen ja mahdollisesti epävakauteen. Kun derivaatta lasketaan ulostulosta, niin derivaattori reagoi vain prosessin todelliseen muutokseen, jolloin säädin toimii rauhallisemmin.

3.3 Integraattorin windup-ilmiö ja anti-windup-toiminnot

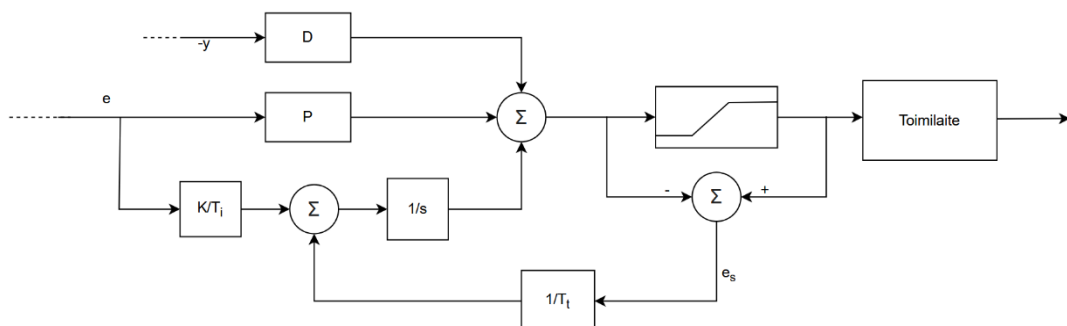
Toimilaitteilla on reaali maailman rajat siinä missä integraali voi kasvaa äärettömästi. Kun prosessin toimilaitte on ääriarajalla, esimerkiksi säätöventtiili on täysin auki, voi ohjaussignaali saavuttaa ylä- tai alarajansa eli mennä saturaatiolle. Tällöin prosessin piiri on avoin, eikä ohjaussignaalin muutos vaikuta prosessin vasteeseen, sillä virhe pysyy vakiona. Integraattoriossa kumuloituu, sillä toimilaitte pysyy ääriasennossaan riippumatta prosessin ulostulosta. Tätä kutsutaan windupiksi eli ylikertymäksi, ja sitä esiintyy tyypillisesti suurissa asetusarvomuutoksissa. [1]

Anti-windup on säätötekniikassa käytetty menetelmä, jolla pyritään estämään säätimen integroivan osuuden ylikertymä, kun ohjaussignaali menee säätöalueen eli toimilaitteen rajojen ulkopuolelle. Yksinkertaisin tapa estää windup on rajoittaa suuria asetusarvomutoksia, mutta tämä ei estä häiriön aiheuttamaa ylikertymää. Parempi tapa estää ylikertymä on niin kutsuttu takaisinlaskenta eli back-calculation, jossa säätimen integraaliosa lasketaan dynaamisesti uudelleen ulostulon mennessä saturaatiolle.

Dynaamiseen laskentaan hyödynnetään seuranta-aikavakiota T_t , joka määrää kuinka nopeasti integraattoria nollataan tai palautetaan. Pienempi T_t tarkoittaa nopeampaa palautusta eli parempaa vastetta, mutta liian pieni seuranta-aikavakio voi aiheuttaa liian herkkää reagointia virheisiin. T_t tulisi valita niin, että se on suurempi kuin derivaatio-osan aikavakio mutta pienempi kuin integraaliaikavakio. Nyrkkisääntönä voidaan pitää, että anti-windupin seuranta-aikavakio valittaisiin seuraavasti: [1]

$$T_t = \sqrt{T_d T_i} \quad (3.9)$$

Kuvassa 8 on esitetty takaisinlaskentaan perustuva anti-windup.



Kuva 8. Takaisinlaskentaan perustuva PID-säätimen anti-windupin alkeislohkokaavio

Toimilaitteen mennessä saturaatiolle, hyödynnetään ylimääräistä takaisinkytkentäsilmukkaa integraalin rajattoman kasvun/vähentymisen välttämiseksi. Säätimen integraaliosa lasketaan uudelleen, siten että integraaliosan ulostulo on saturaatiolla olevan ulostulon suuruinen. Seuranta-aikavakio määrittää palautuksen nopeuden. Mikäli saturaation erosuure e_s on nolla, tällöin ylimääräisellä takaisinkytkennällä ei ole vaikutusta säätimen toimintaan. [1]

3.4 Derivaattorin rajoitus alipäästösuodin

Derivaattorilohko ennakoi, mihin suuntaan järjestelmä on menossa, ja siksi se voi auttaa nopeuttamaan vasteen asettumista ja parantaa vakautta. Ideaalinen derivaattori ei ole kausaalinen, eikä sitä näin ollen voida toteuttaa sellaisenaan, koska se on epäaito. Siksi todellisissa säätimissä käytetään alipäästösuodatusta, jolla derivointi on realisoitavissa ja joka estää korkean taajuuden kohinan voimistumisen. Alipäästösuodin päästää läpi matalat taajuudet ja suodattaa korkeat taajuudet. [18]

Ensimmäisen asteen alipäästösuodin on muotoa

$$H(s) = \frac{1}{s+1} \quad (3.10)$$

Kun derivaattahaaraan lisätään suodatus, voidaan derivaatio-osan siirtofunktio $K_d s$ kirjoittaa muodossa:

$$G_d(s) = \frac{K_d s}{1+sT_f} \quad (3.11)$$

Suodatusaika määritetään kaavalla:

$$T_f = \frac{(K_d/K_p)}{N} = \frac{T_d}{N} \quad (3.12)$$

jossa N on tyypillisesti välillä 2–20. Muuttujan N määrittämisellä valitaan alipäästösuotimen nurkkataajuus. [18]

3.5 Beckhoff TwinCAT-ohjelmointiympäristö

TwinCATissa XAE (extended automation engineering) on Microsoft Visual Studio pohjainen ympäristö, joka tarjoaa IEC 61131-3 -standardin ohjelmointikielet, sekä myös C/C++ että MATLAB/Simulink ohjelmien käyttämisen logiikkaohjauksessa. TwinCAT XAR (extended automation runtime) tarjoaa reaaliaikaisen suoritusajan logiikkaohjaukseen. XAR mahdollistaa minkä tahansa PC:n (Personal computer) käyttämisen logiikkana, mikä mahdollistaa simulaation lokaalissa suoritusympäristössä. [14] Yleisiä logiikkasuorittimia ovat PLC:t (Programmable logic controllerit) ja industrial PC:t (teollisuustietokoneet). [12]

Työssä käytetty säätölogiikka on toteutettu Beckhoffin ohjelmointialustalla ST-ohjelmointikielellä (Structured text), joka on PASCAL pohjainen ylemmän tason lausekieli. Toisin kuin perinteiset tietokoneohjelmat ST toimii syklisesti, eli suorittaa

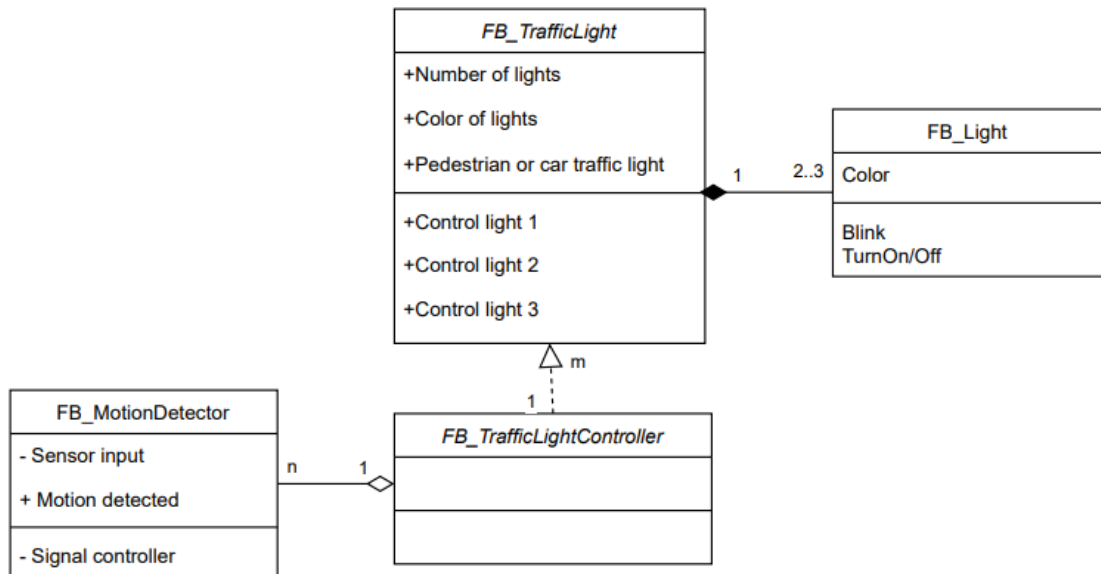
jatkuvasti skannausjaksoja, jossa se lukee tulot, käsittelee logiikan ja päivittää lähdöt. PLC:ssä on watchdog-ajastin, jonka tarkoitus on varmistaa, että skannausjakso ei kestä liian kauan. Esimerkiksi liian pitkä silmukka aiheuttaa PLC:n vikatilaa. [12] TwinCATissa käytetty sykli aika on sovelluskohtainen ja riippuu käytävissä olevasta prosessitehosta. Yleisesti tyypillinen sykli aika PLC:lle on 1–20 ms, mutta se valitaan sovelluksen vaatimusten mukaan, mitä nopeampi ja tarkempi säätö tarvitaan, sitä pienempi sykli aika vaaditaan.

3.6 Algoritmit ja arkkitehtuuri

3.6.1 Olio-ohjelmointi logiikkaohjelmoinnissa

Komplekseissa automaatio-sovelluksissa yleistynyt ohjelmointitapa on OOP (Object-oriented programming) eli olio-ohjelmointi. IEC 61131-3 -standardi on laajentanut PLC-ohjelmointia tuomalla olio-ohjelmoinnin työkalut osaksi standardia, mikä voi helpottaa järjestelmien mallintamista ja uudelleenkäyttöä erityisesti monimutkaisissa sovelluksissa. OOP:n hyöty perustuu neljään peruspilariin, jotka ovat kapselointi, abstraktio, perityminen ja monimuotoisuus. Peruspilareihin nojaten voidaan vähentää tarvittavan ohjelmakoodin määrää ja tehdä ohjelmiston arkkitehtuurista hallittu ja looginen. [15] OOP:n heikkous on tapaukset, jossa koodi pitäisi siirtää toiselle ohjelmointikielelle tai logiikkaympäristöön, joka ei tue olio-ohjelmointia.

Kuvassa 9 on esitetty liikennevalot olio-ohjelmana. Missä yksi logiikka hallitsee yhdestä useampaan määrää liikennevaloja yhden tai useamman liiketunnistimen perusteella. Ohjauslogiikka on luokkakaavion ylimmän tason komponentti, mutta todellisessa sovelluksessa sekin toimisi osana suurempaa kokonaisuutta. Tästä syystä kaavio ei ota kantaa ylimmän tason loogiseen sisältöön. Liikennevalon julkisesta attribuuttirajapinnasta voidaan määrittellä, kuinka monta valoa liikennevalotolpassa on, ja minkä värisiä valot ovat. Lisäksi voitaisiin määrittellä, onko valo-ohjaus kävelijöille vai autoille. Luokkakaavio on hyvin yksinkertaistettu, ja todellisuuden automaatio-sovelluksessa huomioitaisiin enemmän muuttujia, kuten valojen palamisen tai vaihtumisen kestoajat.



Kuva 9. Olio-ohjelmointia havainnollistava luokkakaavio

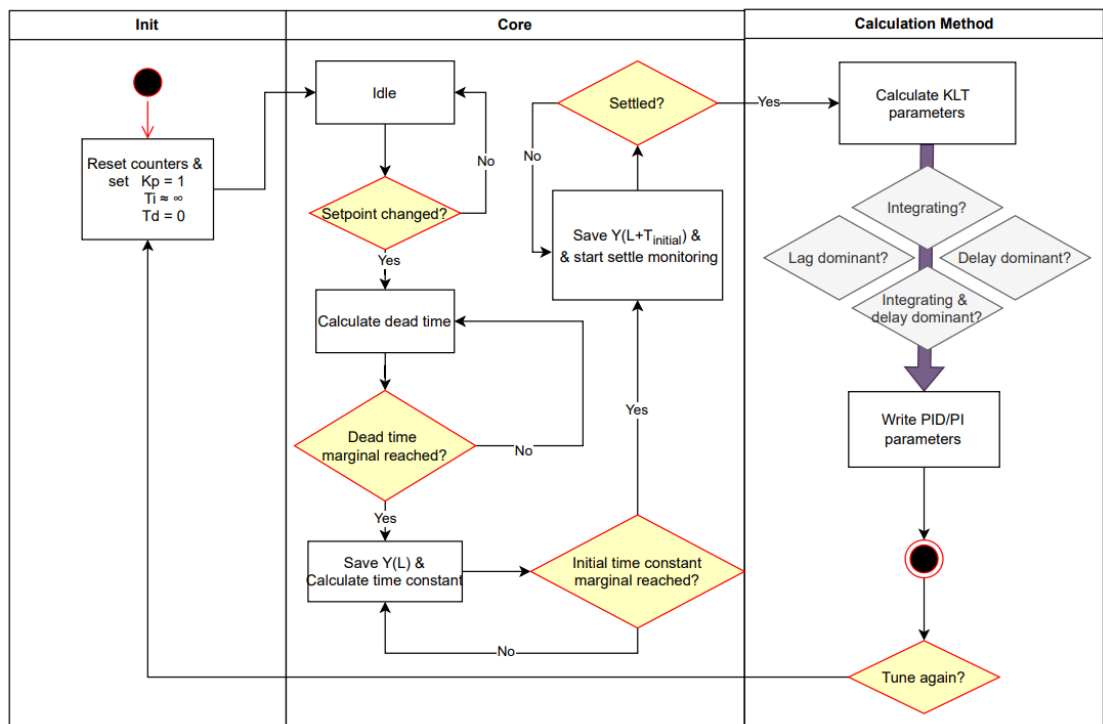
Toimilohkojen eli function blockkien metodit ja attribuutit voivat olla julkisia tai yksityisiä. Liikennevalo esimerkissä liiketunnistin käsittelee signaalin yksityisenä datana ja antaa ulos vain tiedon onko liikettä havaittu. Liikennevalotoimilohko puolestaan tarjoaa avoimen metodirajapinnan valojen ohjaukseen. Todellisessa maailmassa pyrittäisiin piilottamaan mahdollisimman paljon dataa olion sisälle. Tällöin ohjelman ulkopuoliset osat eivät pääse siihen käsiksi, ja virheiden mahdollisuus vähenee, sekä ohjelmakoodin osien hallinta helpottuu. Kun ohjelma tarjoaa ulkopuolisille osille vain välttämättömän piilottaen monimutkaisen toiminnallisuuden, tätä kutsutaan abstraktiksi. Kapseloinnissa puolestaan sidotaan yhteen loogisia tietoja ja toimintoja. Kapselointi ja abstraktio periaatteessa linkittyvät toisiinsa, siinä missä abstraktion tavoite on piilottaa ja kapseloinnissa yhdistää tietoja, niin molempien keskeinen tavoite on tarjota ulos mahdollisimman yksinkertainen rajapinta, ja tehdä ohjelma-arkkitehtuurista näin ollen loogisempaa ja helpompi käyttäjästä. [15]

Ohjelmassa voidaan periyttää toimilohkojen välillä ominaisuuksia. Liikennevaloesimerkissä vilkkuva valo voisi periytyä yksinkertaisemmasta toimilohkosta, joka toteuttaa vain yksinkertaisen vilkkuvan signaalin $0 \rightarrow 1 \rightarrow 0$ ja niin edelleen. Tästä voitaisiin periyttää vilkkuvalo toimilohko, johon lisäksi määriteltäisiin muita haluttuja ominaisuuksia valolle. Periyttämisen idea on vähentää toisteisuutta koodissa, ja yksinkertaistaa olioita, sillä periytetyjä ominaisuuksia ei tarvitse esitellä olion muuttuja-/metodirajapinnassa. Viimeinen tukipilareista on monimuotoisuus.

Aikaisemmin periytetty ominaisuus voidaan määrittellä uudelleen toimilohkossa, jolloin samaan perusluokkaan perustuvat oliot voivat erota toisistaan. Kuvitellaan perusluokaksi eläin, josta on periytetty kaksi oliota lehmä ja leijona. Kun pyydät oliota äänteleämään, voidaan määrittellä ääneksi esimerkiksi karjunta tai ammuminen. [15]

3.6.2 Automaattiviritysalgoritmi

ST-ohjelman aluksi määritellään ohjelmassa käytetyt muuttujat ja toimilohkojen instanssit eli oliot. Tutkimustyönä toteutettu adaptiivinen PID-säätimen viritys on toteutettu yhteen toimilohkoon, joka voidaan mieltää luokaksi. Luokassa on yksi pääohjelmavarkelo, joka määrittelee sen toiminnan. Automaattivirittäjän tapauksessa runkoon on toteutettu kuvan 10 mukainen tilakone. Lisäksi luokassa on kaksi sisäistä metodia. Init-metodi on tyypillisesti ensimmäisellä ajosyklillä kutsuttava nollausfunktio. Toinen metodi on PID-säätimen parametrien laskentaa ja kirjoitusta varten.



Kuva 10. Askelvasteperusteisen automaattiviritystoimilohkon aktiviteettidiagrammi

Viritysolio tarkastelee sille kirjoitettavaa signaalia eli prosessin ulostuloa asetusarvomuutoksen jälkeen. Kun PID-parametrit on löydetty, ne ovat luettavissa olion output-rajapinnasta. Mikäli viritys halutaan tehdä uudestaan oliolle, käsketään uudelleenviritys input-rajapinnasta. Tällöin olio nollaa aikaisemmin lasketut PID-parametrit ja laskee näiden tilalle uudet viritysparametrit. Aina kun lasketaan uusia viritysparametreja, kirjoittaa logiikka integraatio- ja derivaattavahvistuksen nollassi.

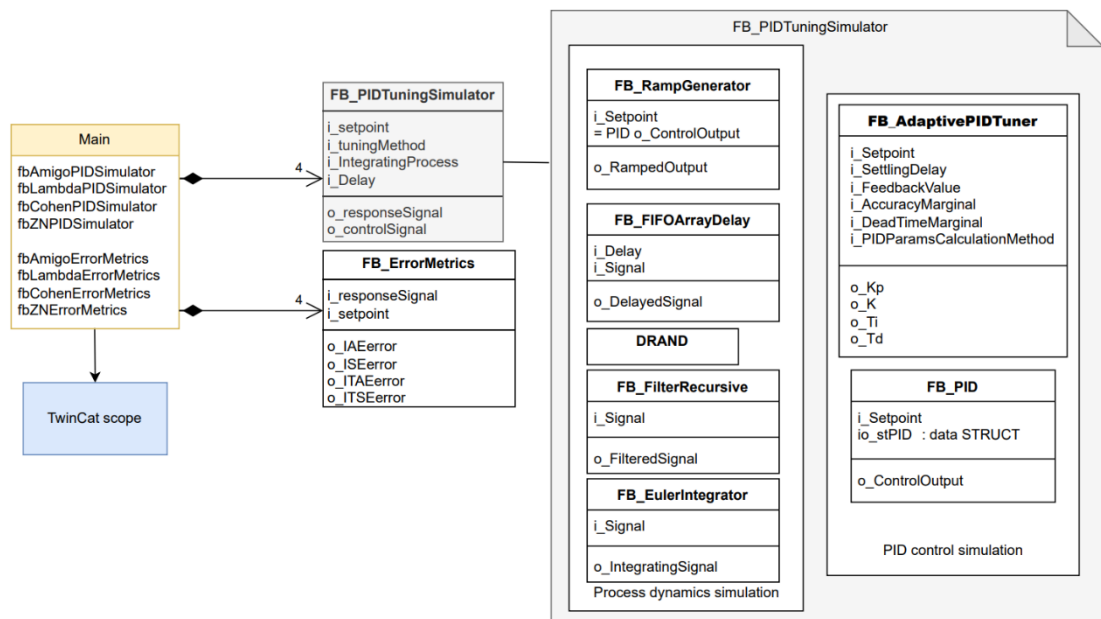
Algoritmin pitää ymmärtää onko kyseessä integroiva prosessi. Integroivissa prosesseissa ei ole staattista vahvistusta, vaan prosessimallin 2.11 mukainen vahvistus lasketaan vasteen nousunopeudesta sisääntulon yksikköä kohden. Algoritmi tarkastelee prosessin vasteen asettumista vasteen muuttumisnopeuden hidastumisen perusteella. Mikäli asettumisaikana prosessin vaste muuttuu aika-askeleen aikana enää jonkin asetettavissa olevan suhteen verran edellisen aika-askeleen muutokseen, todetaan prosessin vasteen asettuneen. Asettumisajan laskenta aloitetaan uudestaan, mikäli prosessin askeleen kasvu kiihtyykin. Algoritmi laskee aikavakion, sillä oletuksella, että prosessin staattinen vahvistus on yksi. Mikäli prosessin vahvistuksen huomataan olevan erisuurta kuin yksi, approksimoidaan aikavakion arvoa kertomalla se vahvistuksella K . Algoritmin on myös tunnistettava, onko prosessi viiveen dominoiva. Normalisoidun viiveen ja integroivan luonteen perusteella algoritmi päättää kirjoitetaanko PID vai PI-parametrit.

Ohjelmistokomponentilla tarkoitetaan itsenäistä ohjelmistoyksikköä, joka on parhaimmillaan helppokäyttöinen ja tarjoaa palveluitaan rajapintojen kautta. Ohjelmistoarkkitehtuuri rakentuu ohjelmistokomponenteista. Arkkitehtuurisuunnittelussa on tärkeää toteuttaa komponenttien välisiä suhteita niin, että jonkin ominaisuuden poistaminen viimeiseksi rikkoo ohjelman toiminnan. Esimerkiksi viritysolikon voi itsenäisenä ohjelmistoyksikkönä lisätä mihin tahansa ohjausjärjestelmään, eikä sen käyttö ole pakollista. Peruseriaatteena voidaan siis todeta, että komponentit eivät koskaan saa olla suoraan riippuvaisia toisistaan, vaan ainoastaan rajapintojen kautta. Mikäli ohjausjärjestelmässä tarvittaisiin PID-säädintä, mutta sille ei haluttaisi laskea parametreja työssä toteutetulla virittäjällä, voidaan viritin olla lisäämättä ohjelmaan. Mikäli viritin olisi PID säätimen toimilohkon sisällä, niin sen pitäminen ohjelmassa ei olisi vapaaehtoista. [20]

4. SÄÄTÖALGORITMIN VALIDOINTI SIMULAA- TIOYMPÄRISTÖSSÄ

4.1 TwinCAT-simulaattori

Kuvassa 11 on esitetty digitaalisen PID-säätimen automaattivirityssimulaattorin ohjelmistokomponentit tässä työssä. Pääohjelmaan on toteutettu instanssit PID-säätimen virittäjälle ja säädön hyvyyslaskennalle jokaista luvussa 2 esitettyä virityssäännöstä kohden.



Kuva 11. Automaattivirityssimulaattorin ohjelmistoarkkitehtuuri

Etuliitteenä *i_* viittaa input-muuttujaan ja *o_* viittaa output-muuttujaan, vastaavasti *io_* viittaa in-out-rajapintaan. Prosessia simuloidaan yksinkertaisella rampifunktiolla, joka kasvattaa asetusarvoa lineaarisesti ajan funktiona. Prosessiin on toteutettu viive FIFO-funktiolla (First-in-first-out), missä signaalia viivästetään siirtämällä datavirtaa rivissä yksi muistialue kerrallaan yhden syklin aikana. Taulukossa olevan datan määrää muuttamalla saadaan viivästyksen suuruus määrättyä, kun sykli aika pidetään vakiona. DRAND-funktio on satunnaislukugeneraattori, jolla signaaliin simuloidaan mittauskohinaa. Luotua mittauskohinaa varten kuitenkin tarvitaan suodatin, jotta simulaatio vastaisi todellista säätötehtävää.

Tutkimustyönä toteutettu 'Adaptive PID tuner' funktio sisältää algoritmit tunnuslukujen laskentaan ja viritysäännöt, joilla PID-säätimen parametrit määritellään. Itse säätö toteutetaan PID-toimilohkossa, jonka ulostulo luetaan ramppifunktiolle asetettavaan asetusarvoon tai integraattorin sisäänmenoksi. PID-virityssimulaattori aliohjelmalle määrätään sisääntulorajapinnassa viiveen suuruus. Integroivan prosessin dynamiikka on toteutettu yksinkertaisella Eulerin menetelmällä, jossa ulostulo kasvaa sisääntuloärsykkeestä loputtomiin ilman säätöä. Tämä mahdollistaa luvuissa 2.2.1, 2.2.2 ja 2.2.3 esitettyjen prosessityyppien simuloinnin. Ohjaustehtäviä on monitoroitu Beckhoffin TwinCAT Measurement -työkalulla [14], joka on kuvassa esitetty Scope-lohkona. Simulaatiosäädön tuloksia tarkastellaan tarkemmin alaluvussa 4.3.

4.2 Simulink-simulaattori

Matlab on MathWorksin kehittämä korkean tason ohjelmointialusta, jonka lisäpaketti Simulink tarjoaa käytännöllisen graafisen käyttöliittymän ohjausjärjestelmien mallintamiseen lohkokaavio-ohjelmana. Simulink-mallilla voidaan simuloida ja analysoida prosessin käyttäytymistä määrättyssä aikaikkunassa. [16]

Vertailun vuoksi TwinCAT-simulaattoria vastaava simulaatiomalli on toteutettu myös liitteen 1 mukaisena Simulink-simulaatiomallina, jossa käytetään TwinCATilla laskettuja KLT-parametreja. Mallissa on vastaavat neljä suljetun piirin simulaattoria. Ulostulosignaalin suodatus on toteutettu C++ koodiblokilla, ja vastaa TwinCATissa käytettyä "recursive filter" suodatinta.

Simulink-simulaattorin avulla voitaisiin todellisessakin säätötehtävässä vertailla eri viritysäännöillä saatujen säätöparametrien sopivuutta. Simulink-simulaatio on nopeampi ja käytännöllisempi kuin TwinCAT-simulaattori.

4.3 Simulaatiosäädön tulokset

Simulaattoreissa prosessit on toteutettu kaavojen 2.9, 2.10, 2.11 mukaisina prosessimalleina, joissa vahvistus on muutettavissa. Integroivaan prosessimalliin voidaan lisätä viivetermi. Aikavakiotermi määräytyy simulaatioissa käytetyn rampifunktion nopeudesta. Vahvistusterminä simulaatioissa käytettiin arvoa $K=1.2$, viiveellisissä prosesseissa L asetettiin arvoon 5.

Testiajoissa aikavakion dominoivassa ja integroivassa prosessissa on käytetty PID-säädintä, pois lukien integroivan prosessin Lambda-säätö, joka on tehty PI-säätimellä. Viiveen dominoivassa prosessissa säätö on tehty PI- ja PID-säätimillä. Integroivaa viiveellistä prosessia on säädetty PI-säätimellä, mutta Ziegler-Nichols ja Cohen-Coon menetelmillä myös PID-säädöllä. AMIGO-viritysparametrien valinnassa on huomioitu kaavan 2.6 mukainen suhteellinen viive. Kaikissa simulaatioissa derivaattorin alipäästösuotimen parametri $N = 10$ ja asetusarvo-painokertoimien alkuarvot $b = 1$ ja $c = 0$.

Simulaatiot tehtiin neljälle eri prosessimalille, joiden automaattivirtäjän löytämät KLT-parametrit on esitetty taulukossa 1.

Taulukko 1. Simulaattorin prosessimallien KLT-parametrit

Prosessi	K / K_v	L	T
Aikavakion dominoiva	$K = 1.19$	0.52	9.91
Viiveen dominoiva	$K = 1.19$	5.43	4.55
Integroiva	$K_v = 0.0478$	0.62	-
Viiveellinen-Integroiva	$K_v = 0.0482$	5.7	-

Viiveettömissä prosesseissa on puolen sekunnin aikaviive, sillä simulaattorin KLT-parametrit ovat laskettu näytteenottoihin perustuvassa logiikkaohjaimessa. Jokaisessa todellisessa prosessissa on aina jonkin verran viivettä tai hidastumista, koska aineen, energian tai informaation siirtyminen vie aina aikaa. Viive voi olla hyvin lyhyt ja käytännössä mitättömän pieni, mutta täydellistä nollaviivettä ei ole.

Taulukkoon 2 on koottu alkuodottamat viritysmenetelmien suoriutumiskyvystä simulaatiosäädössä. Taulukon 2 värikoodaus vastaa alalukujen askelvastekuvajien värikoodausta. Alkuodottamat perustuvat lähteeseen 1, ja teoria on esitetty luvussa 2.

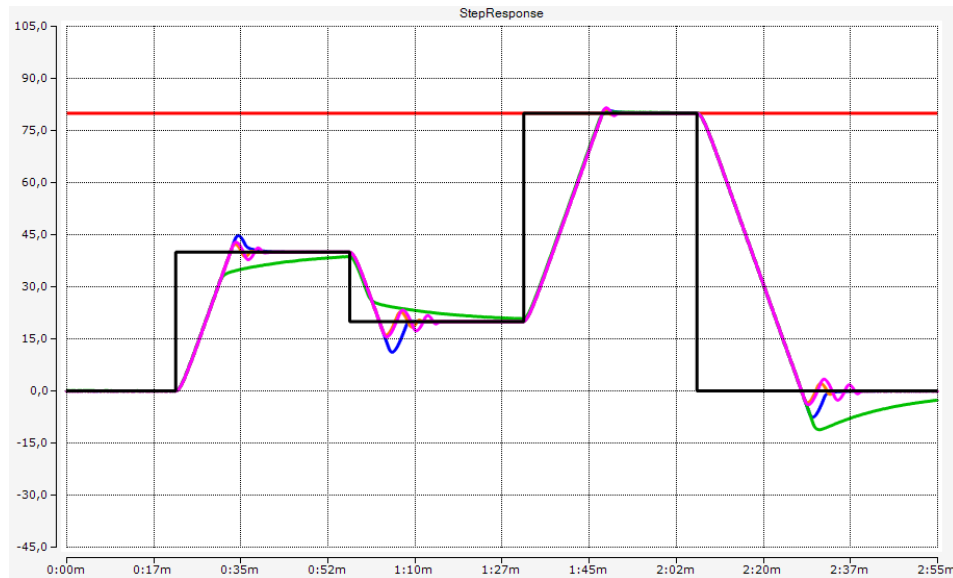
Taulukko 2. Vityssäännösten alkuodottama simulaatiosäädössä

Menetelmä	Tyyppi	Reagointi	Heikkous	Sopii parhaiten
Ziegler-Nichols	Aggressiivinen	Nopea	Kriittinen viritys, jossa järjestelmä toimii lähellä epävakauden tilaa	Viiveellisiin integroiviin prosesseihin
Cohen-Coon	Aggressiivinen	Nopea	Värähtelyn vaimennuskyky	Kriittisiin integroimattomiin prosesseihin, joissa tarkempi viritys on tärkeä
AMIGO	Vakautteen tähtäävä	Hidas	Robustisuutta painottava säädin on hitaampi ja epätarkempi	Hitaat prosessit, pienet säätötarpeet, ei nopeita muutoksia
Lambda	Vaimennettu	Hidas	Epävakaat, ei-minimivaiheiset prosessit, tai pitkän viiven prosessit	Hitaasti muuttuvat aikavakio-dominoivat prosessit

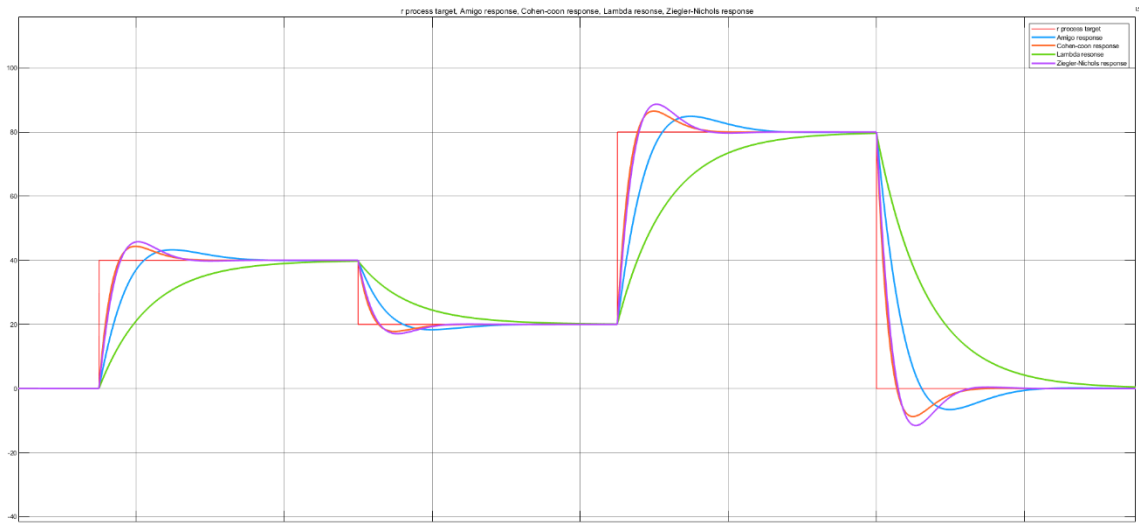
4.3.1 Aikavakion dominoiva prosessi

Kuvissa 12 ja 13 on aikavakio-dominoivan prosessin askelvastekuvajaat. Simulaatioaika on 170 sekuntia. Ziegler-Nichols- ja Cohen-Coon-säännöt suoriutuvat säätötehtävästä ennako-odotuksen mukaan aggressiivisemmin. AMIGO- ja

Lambda-säännöstyillä vaste on rauhallinen etenkin maltillisilla asetusarvomuu-
toksilla. TwinCAT-simulaattorissa aikavakio dynamiikka on toteutettu ramppfunk-
tiolla, joka tekee syötteen muutoksesta jatkuvan ja hitaamman, ja rikkoo yksin-
kertaisen lineaarisen vasteen muodon tehden vasteesta käytännössä epälineaa-
risen ajan funktiona.



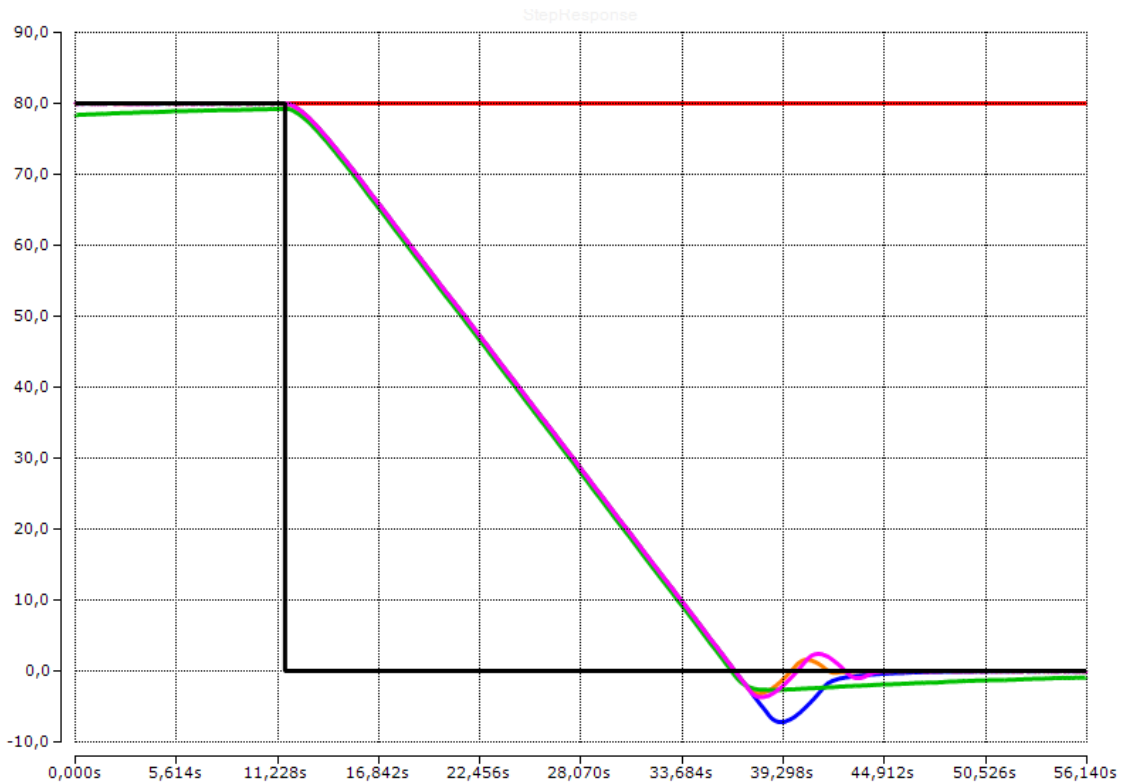
Kuva 12. Aikavakion dominoivan prosessin vasteet TwinCAT-simulaatiossa



Kuva 13. Aikavakion dominoivan prosessin vasteet Simulink-simulaatiossa

TwinCAT-simulaatiossa etenkin Lambda-viritys alittaa asetusarvon suuressa
asetusarvomutoksessa 80:sta nollaan. Aikavakion dominoivassa prosessissa
järjestelmä reagoi hitaasti, mutta säätimen komento on nopea. Simulink-simulaa-
tio on ideaalista, siinä missä TwinCATin laskenta suoritetaan todellisessa ajassa

kiinteällä aikavälillä. Simulaattorin näytteenottoväli on 10 ms, ja suuren muutoksen tapauksessa säätötoimi voi kasvaa aggressiivisesti ennen kuin järjestelmä ehtii korjata itseään seuraavassa näytteessä. Mikäli laskenta-algoritmista halutaan tehokkaampi, voidaan sykliaikaa pienentää, mutta tämä vaatii ohjauslogiikkaa suorittavalta tietokoneelta enemmän resursseja. Kuvassa 14 asetusarvopainokerrointa on muutettu niin, että $b = 0.5$.

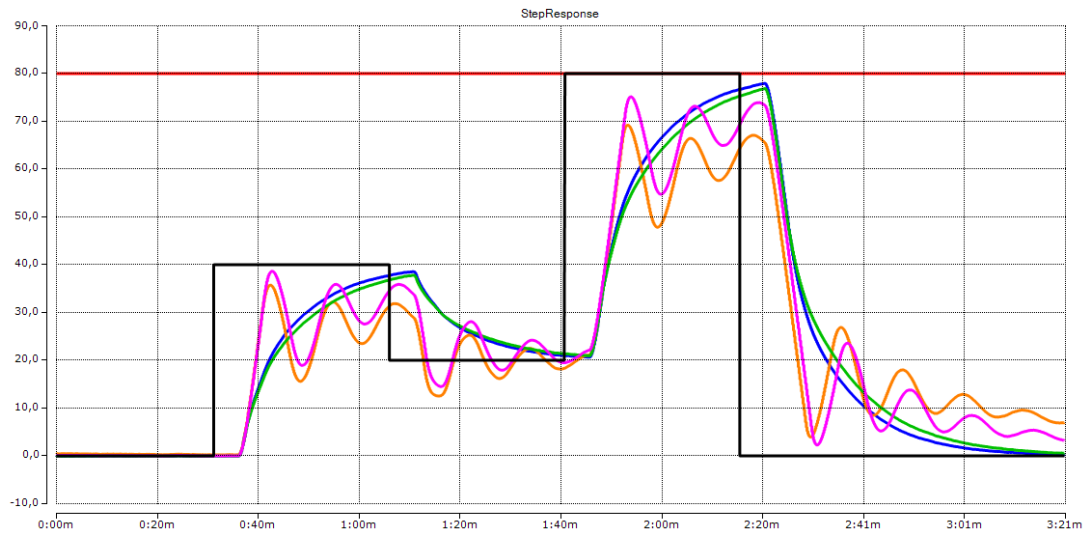


Kuva 14. Asetusarvopainotettu askelvaste aikavakion dominoivalle prosessille TwinCAT-simulaatiossa

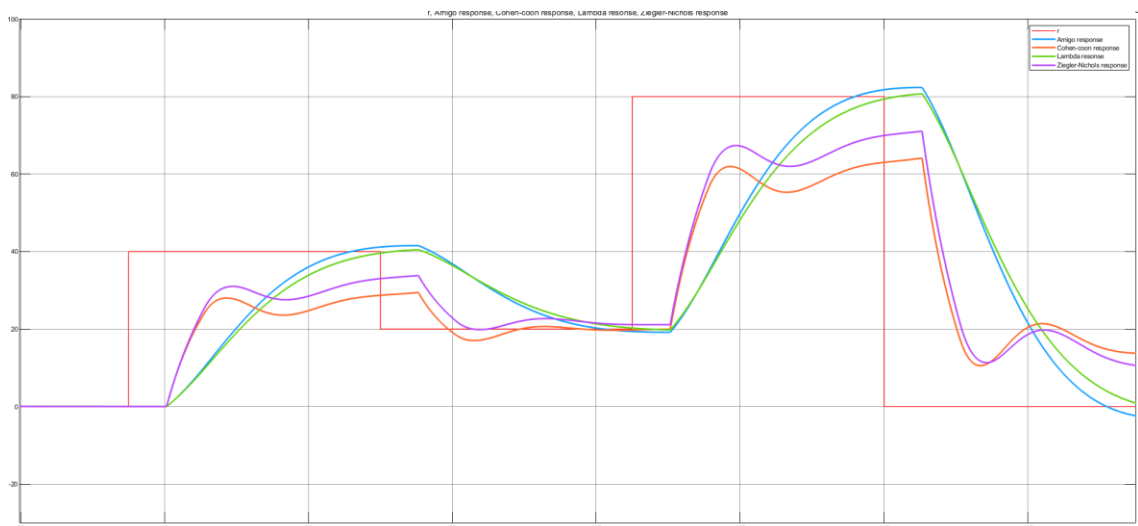
Pienemmän asetusarvopainotuksen kohdalla säädöt rauhoittuvat suuren asetusarvomutoksen tapauksessa hieman. Nyt etenkin Lambda-säädin toteuttaa asetusarvomutoksen vain maltillisella alituksella. Luvussa 2.4.3 mainittiin Ziegler-Nicholsin ja Cohen-Coonin perustana toimiva neljännesvaimennus. Huomataan että aikavakiodominoivan kohdalla neljännesvaimennus toteutuu näennäisesti.

4.3.2 Viiveen dominoiva prosessi

Viiveellisessä prosessissa on kiinteä viiden sekunnin aikaviive, ja tätä pienempi aikavakio. Kuvista 15 ja 16 huomataan että Cohen-Coon- ja Ziegler-Nichols-säännöt reagoivat viiveelliseen prosessin takaisinkytkentään aggressiivisesti ja vaste värähtelee voimakkaasti. AMIGO-säännöstö tuottaa parhaan säätötuloksen, mutta derivaattoriosan puuttuminen vaikuttaa vasteen asettumiseen.



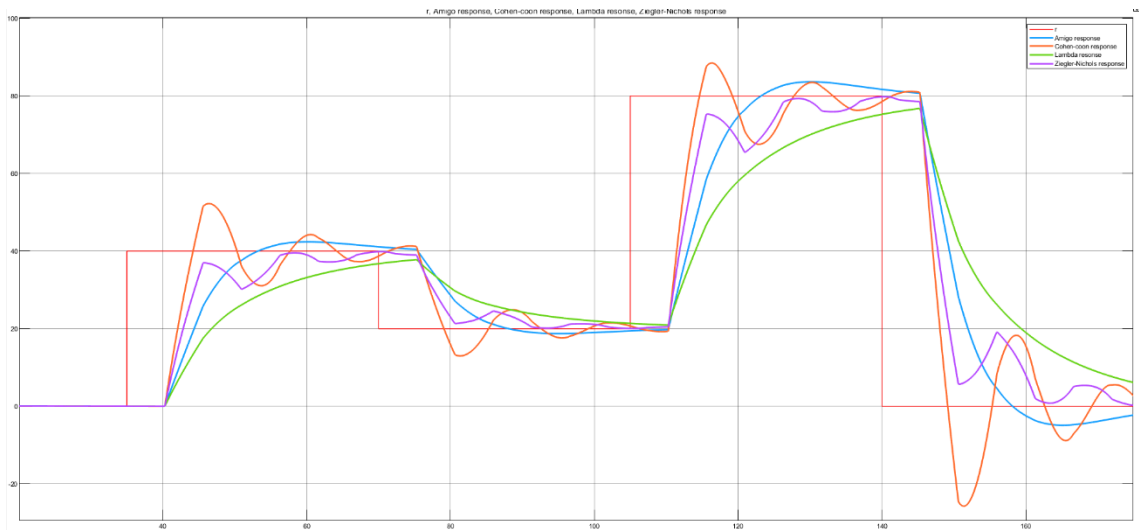
Kuva 15. Viiveen dominoivan prosessin vasteet TwinCAT-simulaatiossa



Kuva 16. Viiveen dominoivan prosessin vasteet Simulink-simulaatiossa

Kuvassa 17 viiveen dominoivan prosessin säädössä on käytetty PID-säätimiä. TwinCAT-simulaattorin aikavakiodynamiikan epälineaarisuuden vuoksi säätö on

tehty vain Simulink-simulaattorilla. Huomataan, että PID-säädöllä saataisiin nopeammat mutta epätarkemmat säätötulokset.

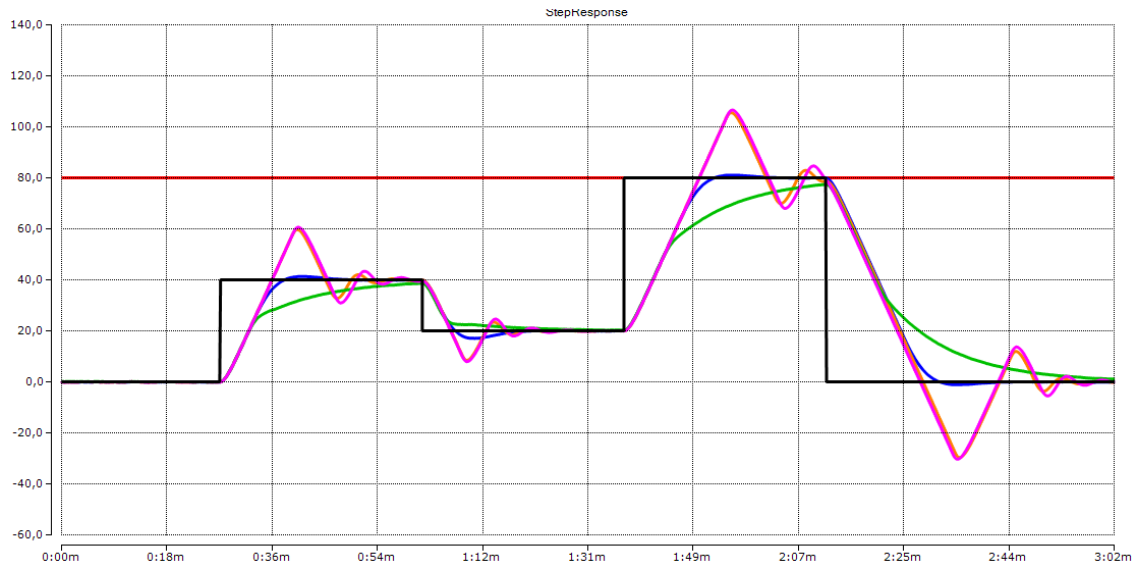


Kuva 17. Viiveen dominoivan prosessin PID-säädetyt vasteet

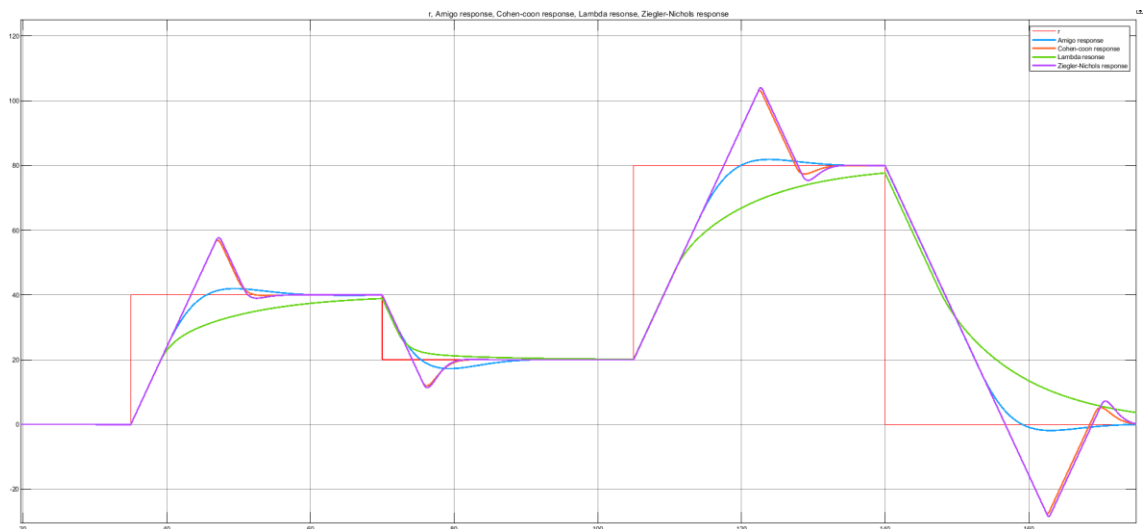
PID-säädössä huomataan viritysäännösten eroavaisuudet paremmin. Lambda- ja AMIGO-säännösten viritys on hitaampi, mutta selvästi toisistaan eroava. Lambda-säännöt tarjoavat ylivoimaisesti hitaimman vasteen. AMIGO on nopeampi, mutta joissain tapauksissa pientäkään ylitystä ei haluta.

4.3.3 Integroiva prosessi

Prosessiin lisätään integraatiotermi ja poistetaan viive- sekä aikavakiotermi. Ku-
vissa 18 ja 19 on esitetty integroivan prosessin vasteet. Huomataan, että Ziegler-
Nichols- ja Cohen-Coon-säännöillä tehty viritys suoriutuu tästä heikoimmin.



Kuva 18. Integroivan prosessin vasteet TwinCAT-simulaatiossa



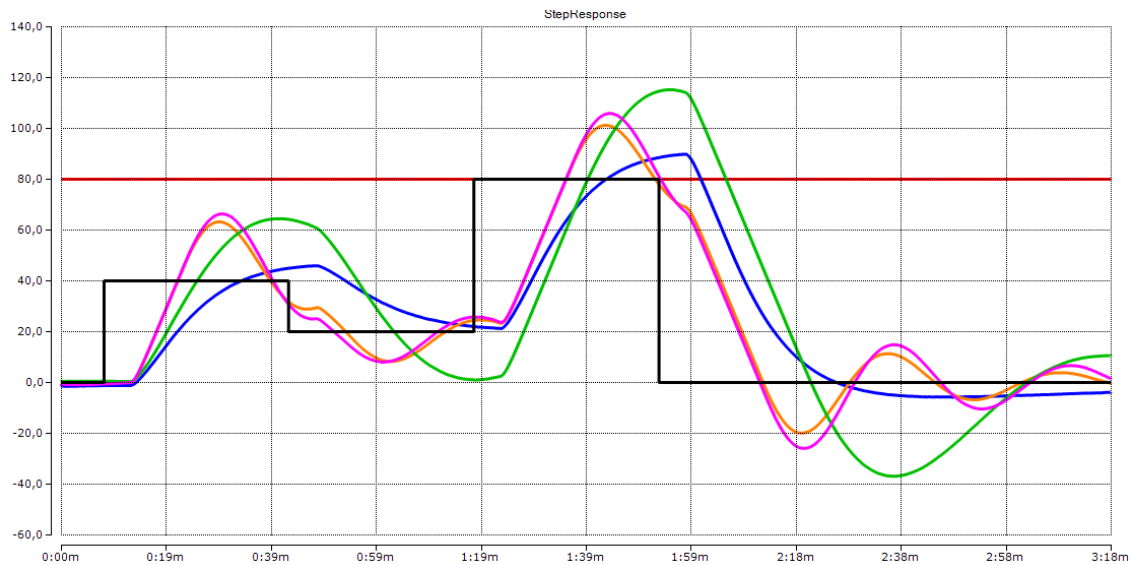
Kuva 19. Integroivan prosessin vasteet Simulink-simulaatiossa

Huomataan myös, että integroivan prosessin kohdalla TwinCAT-simulaattori ja
Simulink-simulaattori toimivat lähes identtisesti, sillä TwinCAT-simulaattorissa ei
ole mukana ramppfunktion epälinearisuutta. Integroivan prosessin kohdalla oh-

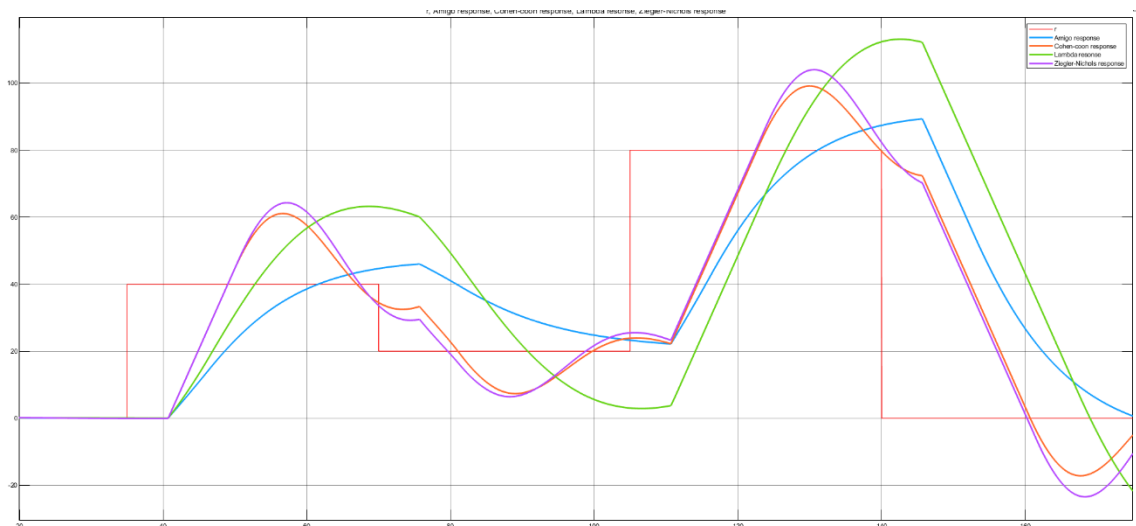
jaussignaali vaikuttaa heti, jolloin etukäteistä ennakointia tai nopeita korjausliikkeitä ei tarvita. Dynamiikka on hidas ja hyvin ennustettava eikä ohjaus saturoi pitkäksi aikaa.

4.3.4 Viiveellinen integroiva prosessi

Viiveelliselle integroivalle prosessille tehdyn simulaatiosäädön tulokset on esitetty kuvissa 20 ja 21. Integroivaan prosessiin lisätään viiden sekunnin aikaviive. Kuvien 20 ja 21 ohjaustehtävät on kontrolloitu PI-säätimellä.



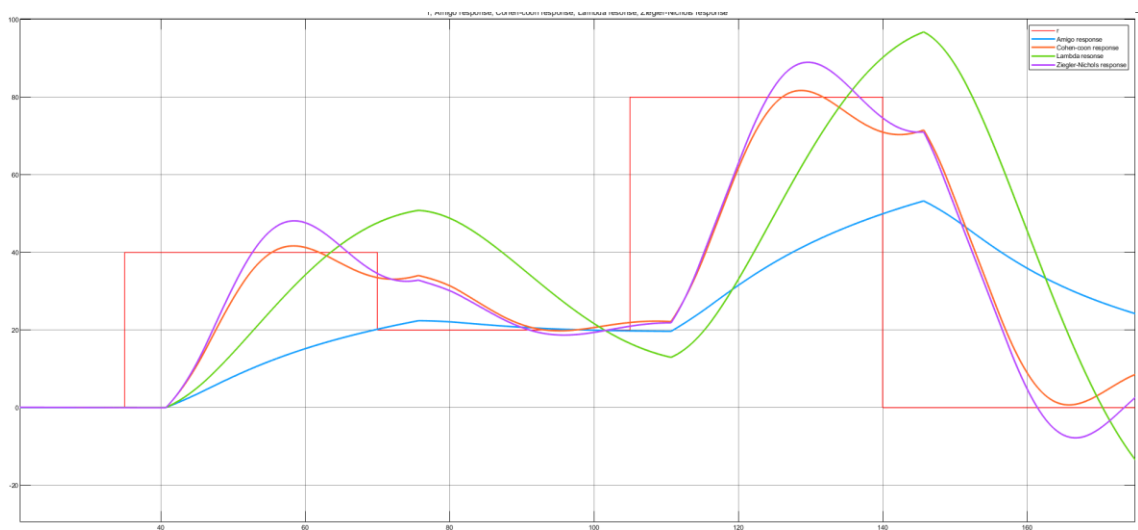
Kuva 20: Viiveellisen integroivan prosessin vasteet TwinCAT-simulaatiossa



Kuva 21. Viiveellisen integroivan prosessin vasteet Simulink-simulaatiossa

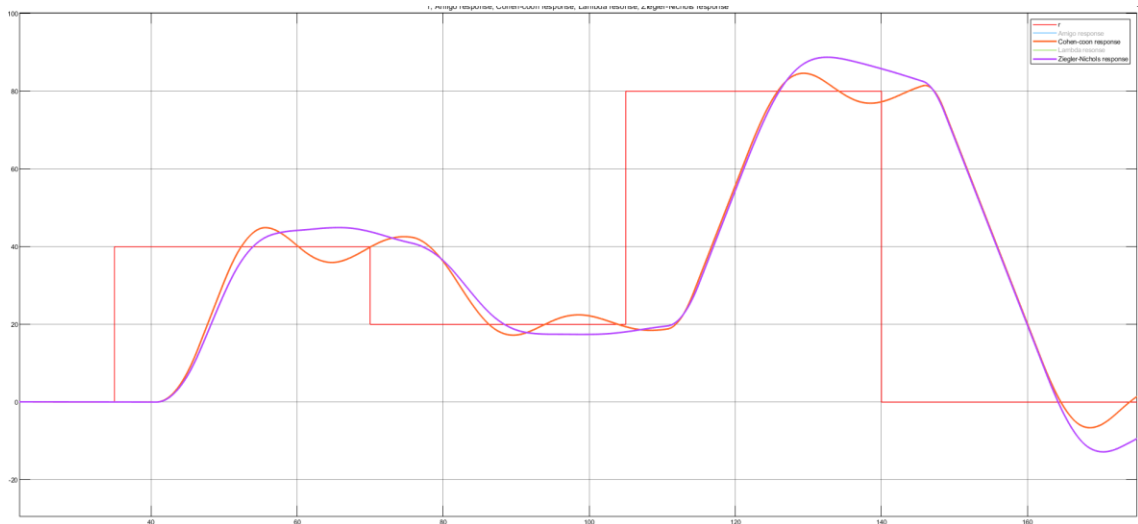
Ziegler-Nichols- ja Cohen-Coon-säännöillä viritetyt prosessit reagoivat viiveeseen hitaasti, ja vasteessa nähdään suuria ylityksiä ja alituksia. AMIGO-säännöllä saadaan robusti säätötulos, vaikkakaan se ei aivan ehdi nopeisiin asetusarvomutoksiin. Viiveellisten integroivien prosessein sovellusalueella ei kuitenkaan usein tarvita nopeita virityksiä (ks. 2.3.2).

Kuvassa 22 asetusarvopainotuksella $b = 0.3$, jolloin Ziegler-Nichols- ja Cohen-Coon-säännöillä saadaan parempia säätötuloksia. Vastaavasti AMIGO-säännöllä tehty viritys ei suoriudu säädöstä enää rajatussa aikaikkunassa pienen vahvistuksen vuoksi.



Kuva 22. Asetusarvopainotettu askelvaste viiveelliselle integroivalle prosessille
Simulink-simulaatiossa

Kuvassa 23 on käytetty PID-säädintä viiveellisen integroivan prosessin säädössä. PID-säätöä voidaan käyttää Ziegler-Nichols- ja Cohen-Coon-säännöillä viiveellisissä integroivissa prosesseissa, sillä niiden säännöt pohjautuvat viiveellisen integroivaan prosessiin. Myös PID-säädössä säädössä on käytetty asetusarvopainotusta $b = 0.3$.



Kuva 23. Ziegler-Nichols ja Cohen-Coon viritteisten PID-säädinten vasteet viiveelliselle integroivalle prosessille

Ziegler-Nichols-säännöillä säädetyn prosessin vaste on rauhallisempi PID-säädöllä kuin PI-säädöllä, mutta derivaattoriosan lisääminen aiheuttaa epätarkkuutta vasteeseen. Vaste jää sitä enemmän asetusarvon yli tai ali mitä suurempi asetusarvomuuotos on. Cohen-Coon-viritys puolestaan toimii vastaavasti kuin PI-säädöllä mutta nopeammin.

5. SÄÄTÖTULOSTEN ANALYYSI

Vaikka ohjaustehtävissä huomattiin myös simulaattorikohtaisia eroja, samat havainnot pystyttiin tekemään molemmilla simulaattoreilla. Simulaattorikohtaiset erot johtuivat epälineaarisuudesta TwinCAT-simulaattorin aikavakiodynamiikan toteutuksessa.

5.2 Säättömenetelmien vertailu

Taulukossa on 3 on esitetty säätötulosten IAE-tunnusluvut simulaatioajoissa, missä prosessikohtaisesti parhaat tulokset on lihavoitu viritysmenetelmän näkökulmasta. Taulukkoon 4 on koottu huomioita säättömenetelmien sopivuudesta eri prosessityyppeihin. Koottujen huomioiden ja taulukon 4 hyvyyslaskennan perusteella voidaan todeta eri säännösten sopivan parhaiten eri prosesseihin. AMIGO-säännöt toimivat kaikissa prosesseissa robustisti. AMIGO-säännöllä tehty viritys oli joissain tapauksissa alkuodottamaa nopeampi. Lambda-säännöllä saadaan hitaimmat viritykset, mutta joissain tapauksissa erityisen hidas ja rauhallinen säätö on oikea valinta, Lambda ei sovellu odotusarvon mukaisesti aikavakion dominoivaan prosessiin. Ziegler-Nichols- ja Cohen-Coon-säännöllä saadaan parhaat tulokset integroivissa viiveellisissä prosesseissa PI- ja PID-säädöllä.

Taulukko 3. IAE ajassa 170 sekuntia

Prosessi	AMIGO	Cohen-Coon	Lambda	Ziegler-Nichols
Viiveeseen dominoiva	3226	2742	3369	2583
Integroiva	1354	1659	1816	1693
Aikavakion dominoiva	644	612	1949	621.8
Viiveellinen integroiva	3412	2924	4782	3054

Taulukko 4. Säättömenetelmien erot simulaatiokokeessa

Menetelmä	<i>Aikavakio-dominoiva prosessi</i>	<i>Viiveen dominoiva prosessi</i>	<i>Integroiva prosessi</i>	<i>Integroiva viiveellinen prosessi</i>
<i>Ziegler–Nichols</i>	Nopea ja tehokas vaste, mutta selvä ylitys	Nopea, mutta heikohko vaimennus	Erittäin aggressiivinen ylitys	Suuri ylitys, mutta rajoitettavissa asetusarvopainotuksella
<i>Cohen–Coon</i>	Hyvin samanlainen vaste kuin ZN, selvä ylitys	Kuten ZN, vielä heikompi vaimennus	Toimii kuten ZN	Paras IAE-tulos ja asetusarvopainotuksella parhaiten soveltuva.
<i>AMIGO</i>	Hitaampi kuin ZN ja CC pieni ylitys	Paras säätö, lievää epätarkkuutta	Hitaampi kuin ZN ja CC, mutta erittäin tarkka	Hidas ja lievä ylitys.
<i>Lambda</i>	Todella hidas, mutta stabiili	Stabiili ja rauhallinen, mutta hitain	Liian hidas aggressiiviseen integroivaan käyttäytymiseen	Hidas ja suuri ylitys

Huomataan, että IAE-tulokset eivät suoraan kerro säättömenetelmän hyvydestä. Nopea säätö etenkin viiveellisissä prosesseissa tuottaa suuren ylityksen, joka voi olla kriittistä. Etenkin Lambda-säännösten hitaus aiheuttaa ulostulon suurta kumulatiivista poikkeamaa erosuureesta. Integroivassa prosessissa Ziegler-Nichols- ja Cohen-Coon-säännöstoillä tehty säätö ylitti ja alitti asetusarvon merkittävästi, mutta sitä ei voi päätellä IAE-tuloksista, sillä säädöt olivat niin nopeita. Samoin viiveen dominoivassa prosessissa Ziegler-Nichols -ja Cohen-Coon-säännösten tarjoama viritys jäi oskilloimaan.

5.3 Johtopäätelmiä

Tutkimustyönä toteutettu adaptiivinen virittäjä laskee PID-säätimen parametrit prosessin vasteesta. Testiympäristössä simulaattori tuottaa aina samanlaisen vasteen. Todellisessa teollisuusympäristössä prosessi voi reagoida vaihtelevasti asetusarvomuutokseen, mikäli prosessiin ei vaikuteta viritetyllä säätimellä. Jotta nykyisellä vitysalgoritmillä saataisiin paras mahdollinen tulos, askelvastekoe voisi olla kannattavaa tehdä useampaan kertaan, ja verrata PID-säätimelle saatuja parametreja. Säädön onnistuneisuus riippuu määrätyistä kriteereistä. Joskus säätimeltä halutaan nopeutta, mutta esimerkiksi ylitystä ei sallita. Jotkin parametrit voivat tarjota säädön, joka on erityisen nopea, mutta siinä on suuret ylitykset, tällöin tulee kunnioittaa molempia kriteerejä ja virittää säädin jollain muulla säännöstöllä.

Tässä työssä kootut päätelmät on tehty simulaatioympäristössä tehtyjen prosessikokeiden perusteella, ja niissä voi esiintyä virheitä. Jotta automaattisen PID-säätimen virittäjän parametrien toimivuudesta voidaan varmistua, tulee virittäjää testata turvallisesti todellisessa teollisuusympäristössä. Automaatiosuunnittelijan tulee tuntea säädettävä prosessi, jotta automaattivirittäjän tarjoaminen PID-säädin parametrien käyttäminen on turvallista. Automaattivirittin tarjoaa neljät toisistaan eroavat parametrit, jotka eroavat käytettävyydeltään prosessikohtaisesti.

Ziegler-Nichols ja Cohen-Coon tarjoavat aggressiivisen virityksen. Viiveellisissä prosesseissa liian nopea säätö aiheuttaa värähtelyä. Mikäli säädettävältä suurelta ei sallita ylitystä, on syytä valita toinen vitysmenetelmä. AMIGO- ja Lambda-säännöstöllä viritetty PID-säädin toimii rauhallisemmin ja soveltuu paremmin viiveelliseen prosessiohjaukseen. Mikäli ylitystä ei haluta, kannattaa PID-säädin virittää AMIGO- ja Lambda-säännöstöillä saaduilla parametreilla. Viiveellisissä prosesseissa on suositeltavaa käyttää PI-säädintä.

PID-säätimen rajoittaminen säädettävän prosessin mukaan on tärkeää. PID-säätimen anti-windup toiminnon ja derivaattorin alipäästösuotimen parametointiin on syytä kiinnittää huomiota, mikäli säätötuloksesta halutaan robusti. Asetusarvopainotuksella voidaan vaikuttaa manuaalisesti säädön nopeuteen riippumatta valitusta vityssäännöstöstä. Integroivien viiveellisten prosessien tapauksessa huo-

mattiin, miten suuri merkitys asetusarvopainotuksella voi olla säätötuloksen kannalta. Simulaatiosäädön perusteella voidaan myös todeta, että joissain prosessimalleissa on kannattavampaa käyttää PID-säätimen sijasta yksinkertaisempaa PI-säädintä.

6. YHTEENVETO

Teollisuuden säätötehtävä voi olla yhden tai useamman aikavakion dominoiva, viivedominoiva, integroiva tai esimerkiksi viiveellinen integroiva. Automaatio-suunnittelijan on erityisen tärkeää tuntea prosessi, jotta prosessiohjauksessa käytetyt säätimet voidaan virittää oikein. Prosessin matemaattinen mallintaminen voi olla kompleksinen tehtävä, ja se ei ole aina mahdollista. Automaattiviritysalgoritmeilla voidaan saada säädettävään prosessiin riittävän hyvät PID-säädin parametrit.

PID-säädin ei sovi kaikkiin teollisuuden prosesseihin. Automaattiviritysalgoritmin pitää tunnistaa, milloin PI-säädin tarjoaa paremman virityksen. Algoritmin käyttäjän vastuulla on kuitenkin ymmärtää prosessin luonne, ja suhtautua kriittisesti automaattivirittäjän tarjoamiin PID/PI-viritysparametreihin. Automaatiosuunnittelu-tehtävässä tulee ymmärtää prosessi kenttälaitetasolta matemaattiseen teoriaan. Työssä luotiin työkalu, jonka tarkoitus on auttaa logiikkaohjelmoijaa virittämään PID-säädin automaattisesti. Tutkimustyön raportti tarkastelee säätötekniikan matemaattisen teorian ja toimilaitteen ohjauksen yhteyttä ohjauslogiikan toteuttajan näkökulmasta.

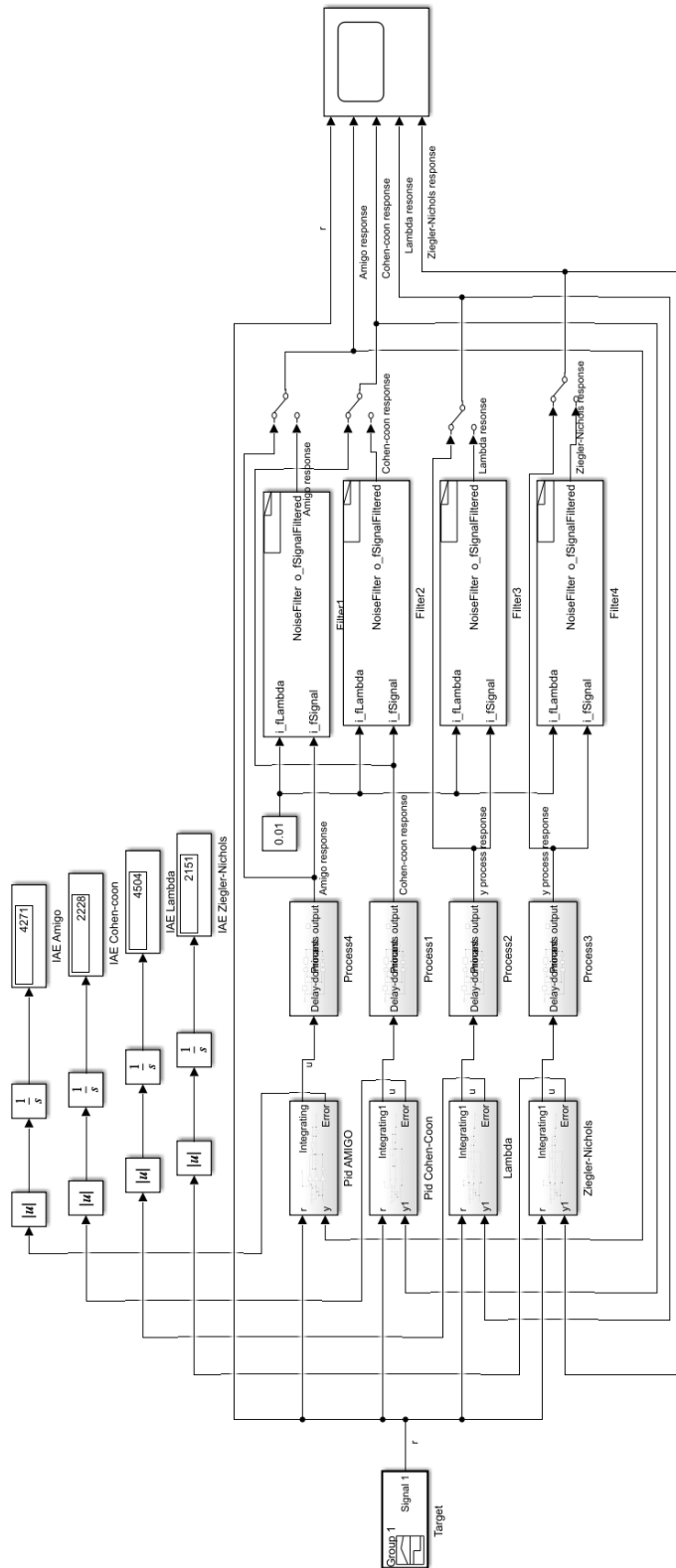
Tutkimuksessa huomattiin, että kaikki askelvastekokeeseen perustuvat algoritmit eivät tarjoa hyvää säätötulosta kaikille prosessimalleille, ja viritysparametrit tulee valita prosessikohtaisesti. Ziegler-Nicholsin- ja Cohen-Coonin-säännöstöihin perustuva viritys ei sovellu hyvin viiveelliseen prosessiin. AMIGO ja Lambda säännöt ovat paremmin määritelty useammalle eri prosessityypille. Ziegler-Nicholsin ja Cohen-Coon-säännöstöjen tarjoamien parametrien huomattiin olevan myös niin samankaltaisia, että suunnittelutyössä olisi ollut kannattavampaa valita Cohen-Coonin sijaan jokin muu säännöstö. Automaattiviritin on toteutettu mahdollisimman geneerisesti niin, että siihen voidaan lisätä myöhemmin tarpeen mukaan muitakin askelvastekokeen perusteella hyödynnettäviä virityssäännöstöjä.

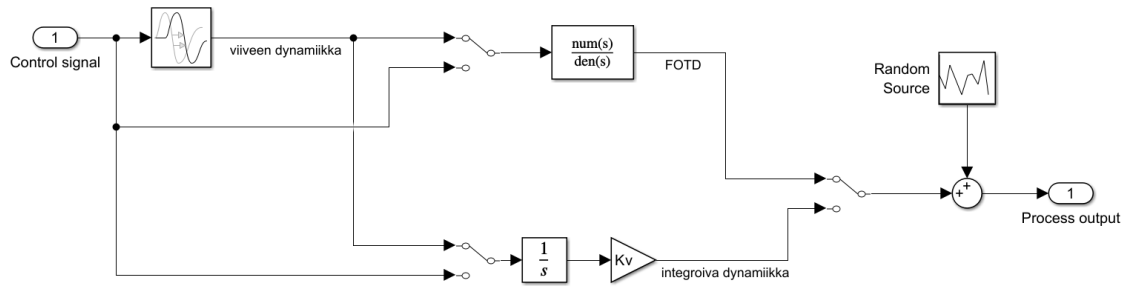
LÄHTEET

- [1] K. J. Åström, T. Hägglund, Advanced PID Control. ISA, 2006.
- [2] J. Thompson, T. Martinson, V. Kauko, Matematiikan käsikirja. 3. p., Tammi, 1994.
- [3] T. Harju, A. Marttinen, Sääntötekniikan koulutusmateriaali. Automaatiosäätö, 2000. Saatavissa: https://www.automaatioseura.fi/site/assets/files/1426/pid_kirja_1-1.pdf
- [4] K. J. Åström, T. Hägglund, Automatic Tuning of PID Controllers. Instrument Society of America, 1988.
- [5] D. Valério, J. S. da Costa, "Tuning of Fractional PID Controllers with Ziegler–Nichols-Type Rules." Signal Processing 86, no. 10 (2006): 2771–84.
- [6] D. Lequesne, Practical PID Handbook. First edition, EDP Sciences, 2022.
- [7] K. J. Åström, T. Hägglund, "Revisiting the Ziegler–Nichols Step Response Method for PID Control." Journal of Process Control, vol. 14, no. 6, 2004, pp. 635–50.
- [8] L. D. Landau, Z. Gianluca, Digital Control Systems: Design, Identification and Implementation. 1st ed., Springer Nature, 2006.
- [9] A. K. Kippo, A. Tikka, Automaatiotekniikan perusteet. Helsinki: Edita publishing Oy, 2008.
- [10] Suomen standardisoimisliitto, Suomen sähköteknillinen standardisoimisyhdistys SESKO. Automaatio, Osa 2, Ohjelmointi Ja Dokumentointi. Suomen standardisoimisliitto, 2014.
- [11] J. Savolainen, R. Vaittinen, Sääntötekniikan perusteita. Suomen robotiikkayhdistys, 2007.
- [12] F. Lamb. Industrial Automation. 1st ed., McGraw-Hill Publishing, 2013.
- [13] H. Karttunen, J. Koistinen, E. Saltikoff, O. Manner, Ilmakehä, sää ja ilmasto. Ursa, 2008.
- [14] Beckhoff, TwinCAT 3: the flexible software solution for PC-based control. Beckhoff automation GmbH & Co. KG, 2021. Saatavissa: https://download.beckhoff.com/download/Document/Catalog/Beckhoff_TwinCAT3_e.pdf
- [15] M. T. White, Mastering PLC Programming: The Software Engineering Survival Guide to Automation Programming. First edition, Packt Publishing Ltd, 2023.
- [16] T. Ashian, Modern Control Design with matlab and Simulink, John Wiley & Sons Ltd, 2002.

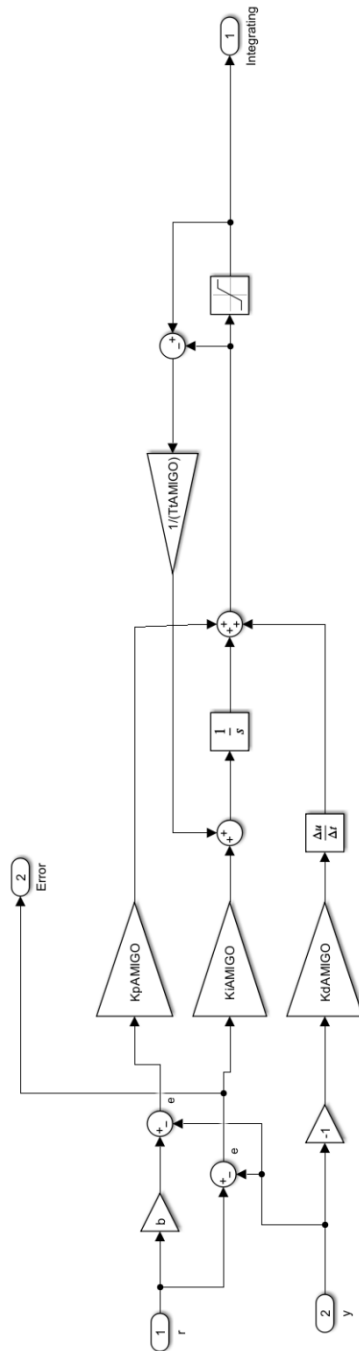
- [17] H Abelson, G. J. Sussman, J. Sussman, Structure and interpretation of computer programs. Second Edition. The Massachusetts Institute of Technology, 1996.
- [18] K. J. Åström, R. M. Murray, Feedback systems: An introduction for Scientists and engineers. Princeton University Press, 2010.
- [19] T. K. Kok Kiong, A. S. Putra, Drives and Control for Industrial Automation. 1. Aufl., Springer Nature, 2010.
- [20] K. Koskimies, T. Mikkonen. Ohjelmistoarkkitehtuurit. Talentum, 2005.

LIITE 1: SIMULAATIOSSA KÄYTETTY SIMULINK-TESTIYMPÄRISTÖ





Prosessilohko



Säädinlohko