Tampereen yliopisto

Toivo Wuoti

KONEOPPIMISALGORITMIEN SOVELTAMINEN PID-SÄÄTIMEN VIRITTÄMISEEN

Kandidaatintyö
Tekniikan ja luonnontieteiden tiedekunta
Tarkastajat: Veli-Pekka Pyrhönen
Maaliskuu 2022

TIIVISTELMÄ

Toivo Wuoti: Koneoppimisalgoritmien soveltaminen PID-säätimen virittämiseen
Kandidaatintyö
Tampereen yliopisto
Automaatiotekniikka
Maaliskuu 2022

Q-oppiminen on malliton vahvistusoppimisalgoritmi, joka oppii tietyn toimen palkkion tietyssä tilassa. Tämän työn tavoitteena oli kehittää Q-oppimisalgoritmi, joka pystyy oppimaan 2DOF-PID-säätimen vahvistusparametrit Quanser Qube-Servo 2 -kääntöheilurijärjestelmälle. Lisäksi tavoitteena oli verrata Q-oppimisella saatavia tuloksia Quanserin tuloksiin ja LQR (Linear-quadratic regulator) -menetelmän tuloksiin. Kääntöheilurijärjestelmä valittiin tutkimuskohteeksi valmiiksi löytyvien tilamallien, sekä fyysisen järjestelmän olemassaolon takia.

Työn teoriaosassa käydään läpi käytettyjen menetelmien teoria, matemaattiset kaavat ja toteutetun simulaatiojärjestelmän rakenne. Työn tulososuudessa vertaillaan eri viritysmenetelmillä saatuja tuloksia, sekä pohditaan reaalimaailman mahdollisia ongelmia.

Q-oppimisalgoritmi toteutettiin Matlab-ohjelmakoodina, joka simuloi kääntöheilurin tilamallia ja sai sen tuloksista vahvistusoppimiseen tarvittavat palkkiot. Q-oppimisalgoritmin Q-taulusta valittiin toimet laskemalla jokaisen toimen eri tilojen palkkiot yhteen ja valitsemalla näistä suurimman positiivisen arvon toimen. Algoritmin laskemia virityksiä vertailtiin perinteisiin menetelmiin simulointien vasteilla. Simuloinneilla tutkittiin, miten eri menetelmät pystyivät tasapainottamaan kääntöheilurin pystyasentoon eri alkutiloista lähdettäessä.

Q-oppimisalgoritmin suurimmaksi haasteeksi osoittautui usean Q-taulun ja laajan tila- ja toimijoukon aiheuttama suuri määrä mahdollisia virityksiä. Hyvä I-haaran viritys saattoi saada negatiivisen palkkion, koska huono P- tai D-haaran viritys aiheutti heilurin kaatumisen.

Q-oppimisalgoritmilla saadut tulokset osoittautuivat melko huonoiksi. Algoritmi ei löytänyt optimisäätöä kaikilla suorituskerroilla, minkä lisäksi sen hyperparametrien optimointiin jouduttiin käyttämään paljon aikaa. Simulaatiossa heilurin säätö onnistui, mutta heilurin vaste oli huomattavasti värähtelevämpi verrattuna tilatakaisinkytkennällä ja valmistajan parametreilla saatuihin vasteisiin. Toisaalta Q-oppimisalgoritmin ohjausjännitteen vaste oli parempi, kuin valmistajan toteuttaman säätöalgoritmin tai LQR-menetelmällä toteutetun säädön vaste.

Avainsanat: koneoppiminen, kääntöheiluri, PID-säädin, säätötekniikka, Q-oppiminen

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

ALKUSANAT

Kiitokset työni ohjaajalle Veli-Pekka Pyrhöselle tarjoamastaan avusta. Sain tehdä kandidaatintyön itse kehittämästäni aiheesta, jonka Veli-Pekka auttoi rajaamaan sopivan kokoiseksi, sekä vastasi aina kysymyksiin nopeasti.

Tampereella, 10. maaliskuuta 2022

Toivo Wuoti

SISÄLLYSLUETTELO

1. Johdanto 1
2. PID-säädin 2
2.1 P-haara 3
2.2 I-haara 3
2.3 D-haara 3
2.4 PID-säätimen asetusarvopainotus 4
3. LQR ja tilamallit 6
3.1 Tilatakaisinkytkentä 7
3.2 LQR 8
4. Q-oppiminen 10
4.1 Alkuarvot 11
5. Kääntöheiluri 12
5.1 QUBE Servo 2 -järjestelmä 12
5.2 Valmistajan 2DOF-PID-säätö 13
5.3 Q-oppiva PID-säätö 13
6. Säätöjärjestelmien simulointi- ja kokeelliset tulokset 18
6.1 Simulointitulokset 18
6.1.1 Q-oppiva 2DOF PID-säätö 18
6.1.2 Valmistajan 2DOF-PID-säätö 20
6.1.3 Valmistajan LQR-säätö 21
6.2 Säätöjen vertailu 23
7. Yhteenveto 25
Lähteet 26
Liite A: Matlab, QUBE:n opetussimulaattori 27
Liite B: Matlab, Q-oppimisalgoritmi 33

LYHENTEET JA MERKINNÄT

1DOF	One-Degree-of-Freedom, Yhden vapausasteen järjestelmä
2DOF	Two-Degree-of-Freedom, Kahden vapausasteen järjestelmä
A	Tilamallin tilamatriisi
α	Oppimistahti
B	Tilamallin ohjausmatriisi
b	P-haaran asetusarvon painokerroin
$C(s)$	Säätimen siirtofunktio
C	Tilamallin mittausmatriisi
c	D-haaran asetusarvon painokerroin
$D(s)$	Suodatetun ideaalisen derivaattorin siirtofunktio
D	Tilamallin suoravaikutusmatriisi
$e(t)$	Erosuure
e_d	D-haaran erosuure
e_p	P-haaran erosuure
K	Tilatakaisinkytkennän vahvistusvektori
K_d	Derivointihaaran vahvistus
K_i	Integrointihaaran vahvistus
K_p	P-haaran vahvistus
k_r	Asetusarvon vahvistuskerroin
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant, lineaarinen aikainvariantti järjestelmä
$m(t)$	Prosessin mitattu ulostulo
N	Alipäästösuotimen rajataajuus
O_b	Havaittavuusmatriisi
O_c	Ohjattavuusmatriisi
PID	Proportional-Integral-Derivative, PID-säädin
Q	Q-taulu

Q_u	Tilojen painoarvomatriisi
R	Ohjausten painoarvomatriisi
s	Laplace-muuttuja
t_0	Aloitusaika
T_d	PID-säätimen derivointiaika
T_i	PID-säätimen integrointiaika
u_b	P-ohjauksen biastermi
u_d	D-haaran ohjaus
u_i	I-haaran ohjaus
u_p	P-haaran ohjaus
$\dot{x}(t)$	Tilojen derivaattavektori
$x(t)$	Tilavektori
x_0	Aloitustilavektori
γ	Vähennyskerroin
$u(t)$	Säätimen ohjaus
$y(t)$	Systeemin ulostulo

1. JOHDANTO

PID-säädin on eräs takaisinkytkentää hyödyntävä säädin, joka on kaikkein yleisin suunnittelutyökalu säätötekniikassa [1, s. xi]. PID-säädin kehitettiin ratkaisemaan päälle/pois-säädinten ongelma, jossa ohjattava suure värähteli asetusarvon läheisyydessä. Tähän ongelmaan käytettiin ratkaisuna suhteellista säätöä (proportional control). Suhteellisen säädön ongelmaksi jäi loppuarvon poikkeaminen halutusta tilasta (säätövirhe), jonka ratkaisemiseksi kehitettiin integroiva säätö, jolla äärellisen pieni säätövirhe aiheuttaa äärellisen ohjauksen äärellisessä ajassa. Ohjauksen parantamisen mahdollisti derivoiva säätö, jonka avulla pystytään kompensoimaan tulevia muutoksia heti kun ne ovat muodostumassa. [1, s. 26]

PID-säätimen monipuolisuus mahdollistaa sen käyttämisen useissa erilaisissa prosesseissa, minkä takia yli 95 % kaikista teollisista säätöongelmista on ratkottu PID-säätimillä [1, s. 24]. PID-säädinten suosion takia niiden virittämiseen on kehitetty useita menetelmiä, jotka soveltuvat dynamiikaltaan monenlaisten järjestelmien takaisinkytkettyyn säätöön. Osa menetelmistä tarvitsevat prosessin dynaamisen mallin, mutta PID-säädin voidaan monelle stabiilille systeemille virittää myös ilman mallia, kunhan prosessista pystytään kokeellisesti selvittämään tietyt ominaisuudet. Tässä työssä viritetään PID-säätimet dynamiikaltaan testijärjestelmälle koneoppimismenetelmällä ja verrataan siitä saatuja tuloksia perinteisillä menetelmillä saatuihin tuloksiin. Testijärjestelmäksi valittiin Quanserin QUBE Servo 2 -heiluri, jonka valmistajan tarjoamiin 2DOF(Two-Degrees of Freedom)-PID- ja LQR-säätöihin Q-oppimisalgoritmia verrataan.

Toisessa luvussa käsitellään PID-säätimen teoriaa ja 2DOF-asetusarvopainotusta. Kolmannessa luvussa esitellään lyhyesti tilamallit ja säätimen virittäminen LQR-menetelmällä. Luvussa 4 esitellään Q-oppiminen ja luvussa 5 esitellään kääntöheiluri yleisesti, sen erityistapaus QUBE Servo 2, sekä QUBE Servo 2:lle toteutetut säädöt. Luvussa 6 esitellään säätöjärjestelmien simuloidut tulokset ja luvussa 7 tehdään yhteenveto.

2. PID-SÄÄDIN

PID-säädin on säätötekniikassa käytetty säätöjärjestelmän osa, joka käyttää prosessin mittaussuuretta ja asetusarvoa prosessin ohjaukseen. PID-säädin toteutetaan tyypillisesti pneumaattisesti, mekaanisesti, sähköisesti tai tietokoneen avulla digitaalisesti. Tavoitellun asetusarvon saavuttaminen ja siinä pysyminen toteutetaan matemaattisilla menetelmillä. [1, s. 23–24] [2, s. 60–70] [3, s. 64–72]

PID-säädin koostuu kolmesta haarasta: P-haara (Proportional), I-haara (Integral) ja D-haara (Derivative). Näiden haarojen rinnan- ja sarjankytkennöillä on mahdollista muodostaa erilaisia säätimiä, kuten PID- tai PI-säädin. On myös mahdollista muodostaa säädin vain yhdestä komponentista, mutta haarojen ominaisuuksien takia vain P-säädin on yleisesti käytetty yksihaaraisista säätimistä. PID-säätimen ohjausfunktio on

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e\left(\tau\right) d\tau + K_{\mathrm{d}} \frac{de(t)}{dt}, \tag{2.1}$$

jossa K_p on proportionaalivahvistus, $u(t)$ on ohjaussignaali, $e(t)$ on erosuure, K_i on integrointivahvistus ja K_d on derivointivahvistus. PID-säätimen ohjaukselle on olemassa myös standardimuoto, jonka yhtälö on

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de(t)}{dt} \right), \tag{2.2}$$

jossa T_i on integrointiaika ja T_d on derivointiaika. Kaavojen (2.1) ja (2.2) avulla saadaan yhteydet integrointi- ja derivointihaarojen vahvistuksille ja integrointi- ja derivointiajoille

$$K_i = \frac{K_p}{T_i} \qquad\qquad K_d = K_p T_d. \tag{2.3}$$

PID-säätimen ohjausfunktiossa oleva erosuure on asetusarvon r ja vasteen mittauksen m erotus:

$$e(t) = r(t) - m(t), \tag{2.4}$$

jossa $r(t)$ on asetusarvo ja $m(t)$ on prosessin mitattu ulostulo. PID-säätimen oppikirjayhtälöstä saadaan Laplace-muunnoksella siirtofunktio

$$C(s) = K_p(1 + \frac{1}{sT_i} + sT_d), \tag{2.5}$$

jossa s on Laplace-muuttuja. PID-säätimen oppikirjasiirtofunktio (2.5) on epäaito, eli se ei ole käytännössä toteutettavissa.

2.1 P-haara

Asettamalla PID-säätimen integrointivahvistuksen K_i ja derivointivahvistuksen K_d arvot nolliksi, saadaan P-säädin:

$$u_p(t) = K_p e(t) + u_b, \tag{2.6}$$

jossa $u_p(t)$ on säätimen ohjaus, K_p on P-haaran vakiovahvistus ja u_b on biastermi. Biastermi tarvitaan, koska pelkällä P-säädöllä ei tyypillisesti saavuteta tarkkaa säätöä tasapainotilassa, mutta tarkka säätö voidaan periaatteessa saavuttaa biastermin u_b oikealla valinnalla.

P-haarassa vakiovahvistus K_p on ainut viritysparametri. Liian pieni vakiovahvistus johtaa hitaaseen prosessivasteeseen, kun taas liian suuri vakiovahvistus johtaa vasteen värähtelyyn ja pahimmillaan järjestelmän epästabiiliuteen.

2.2 I-haara

Integrointihaara huolehtii, että prosessin ulostulolla on sama arvo kuin prosessin asetusarvolla tasapainotilassa. I-haarassa positiivinen virhe johtaa aina kasvavaan ohjaussignaaliin ja negatiivinen virhe pienenevään ohjaussignaaliin. I-haaran ohjauksen kaava on

$$u_i(t) = K_i \int_0^t e(\tau)\, d\tau, \tag{2.7}$$

jossa $u_i(t)$ on I-haaran ohjaus. I-haaran integrointiominaisuuden takia pienikin erosuure aiheuttaa I-haaran muutoksen, jolloin erosuureen nollaantuessa P-haaran ohjaus nollaantuu. I-haara laskee tasapainotilassa biastermille u_b automaattisesti oikean arvon huolimatta siitä, mistä syystä erosuure aiheutui. P-haaraa ja I-haaraa käyttävää säädintä kutsutaan PI-säätimeksi, joka on yleisin käytetty säädin [1, s. 293][3, s. 87].

2.3 D-haara

D-haaran tarkoituksena on nopeuttaa suljetun piirin reaktionopeutta, kun erosuure on nollasta poikkeava. Sillä voidaan vaimentaa askelvasteen värähtelyä ja parantaa piirin stabiiliusvaroja. Tämä nopeampi reagoiminen voidaan toteuttaa ohjauksella, joka hyödyntää ennustettua prosessin ulostuloa. D-haaran ohjauksen kaava on

$$u_d(t) = K_d \frac{de(t)}{dt}. \tag{2.8}$$

D-haaran hyödyllisyys on kuitenkin prosessikohtaista. Kohinaisen erosuureen derivointi voi aiheuttaa toimilaitteiden kulumista ja voi pahimmillaan johtaa epästabiiliuteen. Alipäästösuotimen ja D-haaran virittämisen vaikeuksien takia PID-säätimet eivät ole yleisiä käyttökohteissa, joissa tarvitaan vain hyvää regulointikykyä, kuten hitaissa teollisuuden prosesseissa. D-haara on hyödyllinen prosesseissa, joiden normalisoitu viive on pieni. Tuplaintegraattoria ei voida ohjata PI-säätimellä, koska tuplaintegraattorilla on –180° n vaihesiirto ja PI-säädin lisää negatiivista vaihesiirtoa. [3, s. 87–88] Puhtaan derivaattorin Laplace-siirtofunktio on epäaito, eli se on täten myös epästabiili. D-haara voidaan toteuttaa ideaalisen derivaattorin ja alipäästösuotimen yhdistelmällä., jolloin ensimmäisen kertaluokan suodatettu derivaatta on

$$D(s) = K_d \, \frac{sN}{s+N} = K_d \, \frac{N}{1+N\frac{1}{s}}, \tag{2.9}$$

jossa N on alipäästösuotimen rajataajuus.

2.4 PID-säätimen asetusarvopainotus

PID-säätimen tavoitteena on tasapainottaa vaatimukset asetusarvon seurannalle sekä mittaus- ja kuormitushäiriöiden vasteille. Tyypillinen yhden vapausasteen PID-säädin tekee ohjauspäätökset vain erosuureen perusteella. Yhden vapausasteen järjestelmät riittävät useaan järjestelmään, joissa tarvitaan vain hyvää häiriövastetta, hyvää asetusarvovastetta tai näiden kompromissia. Kuitenkin useassa prosessissa tarvitaan hyvää häiriö- ja asetusarvovastetta, jolloin yksi vapausaste ei riitä säätimen vaatimuksiin. Tähän tehtävään voi soveltaa 2DOF-PID-säädintä, jossa käytetään erillisiä painokertoimia asetusarvolle ja prosessin mitatulle ulostulolle. Asetusarvopainotetun 2DOF-PID-säätimen ohjaussignaalin kaava on

$$u(t) = K_p \left(e_p + \frac{1}{T_i} \int_0^t e(\tau) \, d\tau + T_d \frac{d \, e_d}{dt} \right), \tag{2.10}$$

jossa P-haaran erosuure on

$$e_p = br - y, \tag{2.11}$$

jossa b on P-osan asetusarvon painokerroin. D-haaran erosuure on

$$e_d = cr - y, \tag{2.12}$$

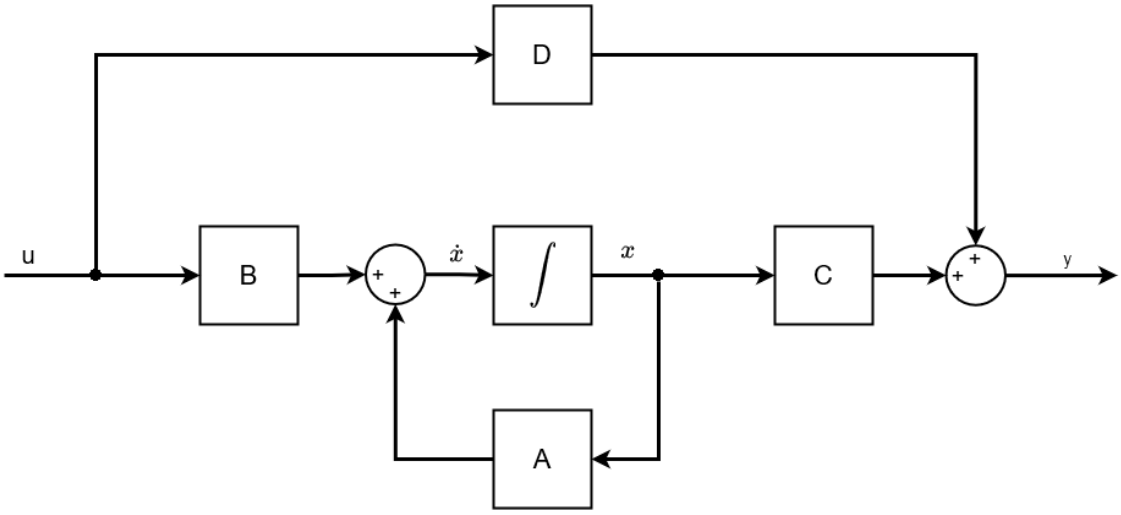
jossa c on D-osan asetusarvon painokerroin. Sijoittamalla kaavat (2.11) ja (2.12) kaavaan (2.10) saadaan ohjaussignaalin kaava muotoon

$$u(t) = K_p \left(br - y + \frac{1}{T_i} \int_0^t e(\tau) \, d\tau + T_d \frac{d(cr-y)}{dt} \right). \tag{2.13}$$

Painokertoimien b ja c avulla voidaan määritellä asetusarvon vaikutus P- tai D-haaran ohjaussignaaliin. Painokerrointen arvoilla $b = 1$ ja $c = 1$ säätimessä ei ole painotusta, jolloin yhtälö yksinkertaistuu kaavaan (2.2). D-haaran painokerroin c valitaan usein nollaksi, koska silloin vältetään äkillisten asetusarvon muutosten ongelmat D-haarassa [3, s. 74]. P-haaran painokerroin b valitaan usein väliltä $0 \leq b \leq 1$.

3. LQR JA TILAMALLIT

Säätötekniikassa fyysisiä järjestelmiä kuvataan matemaattisilla malleilla. Eräs mallityyppi on tilamalli, joka muodostetaan sisääntulo-, ulostulo- ja tilamuuttujista. Nämä muuttujat sidotaan toisiinsa differentiaali- tai differenssiyhtälöillä. Ulostulomuuttujat riippuvat tilamuuttujien arvoista, jotka taas riippuvat tilojen aikaisemmista arvoista ja sisääntulomuuttujien arvoista.



Kuva 3.1. *Tilamallin lohkokaavio*

Tilamuuttujat ovat pienin mahdollinen määrä järjestelmämuuttujia, joiden avulla voidaan laskea järjestelmän tila millä hetkellä tahansa. Tämä määrä on tyypillisesti sama kuin järjestelmää kuvaavan differentiaaliyhtälön aste, eli esimerkiksi kolmannen kertaluokan differentiaaliyhtälön kuvaamiseen tarvitaan kolme ensimmäisen kertaluokan differentiaaliyhtälöä [4, s. 125]. Jos dynaaminen järjestelmä on lineaarinen ja aikainvariantti, voidaan differentiaaliyhtälöt kirjoittaa matriisimuotoon. [4, s. 123–126] LTI-tilamallin yhtälö on:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{3.1}$$

jossa $x(t) \in \mathcal{R}^{n\times 1}$ on tilavektori ja $\dot{x}$ on tilavektorin derivaattavektori:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \qquad \dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix} \tag{3.2}$$

$u(t) \in \mathcal{R}^{m\times 1}$ on sisääntulovektori ja $y(t) \in \mathcal{R}^{p\times 1}$ on ulostulovektori:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \qquad y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{bmatrix} \tag{3.3}$$

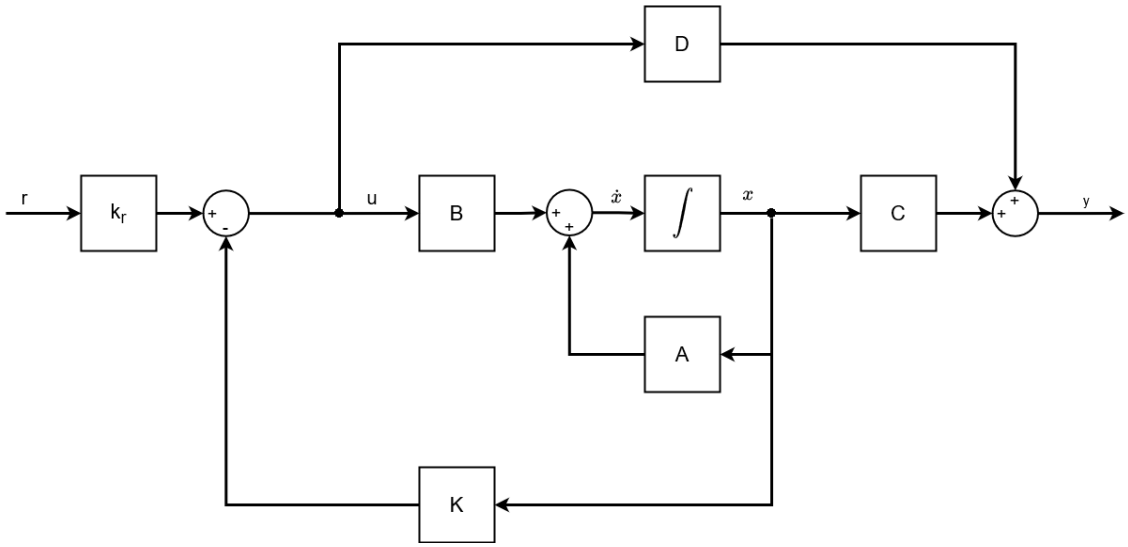
ja $A \in \mathcal{R}^{n\times n}$ on tilamatriisi, $B \in \mathcal{R}^{n\times p}$ on ohjausmatriisi, $C \in \mathcal{R}^{q\times n}$ on mittausmatriisi ja $D \in \mathcal{R}^{q\times p}$ on tilamallin suoravaikutusmatriisi. Tiloille on myös määritelty alkutila x_0, sekä aloitusaika t_0. [5, s.3–5]

3.1 Tilatakaisinkytkentä

Tilamalleille on kaksi säädön kannalta olennaista ominaisuutta: ohjattavuus ja tarkkailtavuus. Ohjattava järjestelmä voidaan ohjata nykyisestä tilasta mihin tahansa muuhun tilaan äärellisellä ohjauksella ja äärellisessä ajassa. Ohjattavuudelle voidaan laskea ohjattavuusmatriisi:

$$C_o = [B \quad AB \quad \dots \quad A^{n-1}B], \tag{3.4}$$

jossa A on tilamatriisi, B on ohjausmatriisi ja n tilojen määrä. Lineaarinen järjestelmä on ohjattava vain, jos ohjattavuusmatriisi C_o on täysiasteinen. [1, s. 167–169]



Kuva 3.2. *Täyden tilan tilatakaisinkytkentä*

Tilatakaisinkytkennässä käytetään tilavektoria $x(t)$ järjestelmän ohjausten laskennassa laskemiseen. Tällöin tilavektori kerrotaan takaisinkytkentämatriisilla K ja negatiivisesti takaisinkytketään järjestelmän asetusarvoon kuvan 3.2 mukaisesti. Asetusarvo kerrotaan vahvistuskertoimella k_r, jolla säätöpiirin DC-vahvistus saadaan 1:ksi.

Edellä mainittu tilatakaisinkytkentä on käytettävissä vain, jos kaikki järjestelmän tilat ovat havaittavissa. Havaittavassa järjestelmässä kaikki tilat voidaan päätellä ulostulosta y ja ohjauksesta u [6, s. 682]. Useissa tilanteissa kaikkia tiloja ei voida mitata, mutta puuttuvat tilat voidaan korvata estimoiduilla tiloilla. Tätä voidaan tutkia ns. havaittavuusmatriisin

$$O_b = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1}, \end{bmatrix} \tag{3.5}$$

avulla, jossa A on tilamatriisi, C on mittausmatriisi ja n on tilojen määrä. Järjestelmä on havaittava jos matriisissa O_b on n kappaletta lineaarisesti riippumattomia sarakkeita. [1, s. 201–211]

3.2 LQR

LQR, eli Linear-quadratic regulator on vaihtoehto säädön suunnittelemiselle suljetun piirin napojen asettelun sijaan. LQR:n tavoitteena on minimoida neliöllinen kustannusfunktio, jonka avulla voidaan laskea viritysparametrien arvot automaattisesti. Jatkuva-aikaisille järjestelmille tilatakaisinkytkentälaki $u = -Kx$ minimoi neliöllisen kustannusfunktion [1,

s. 190–192]

$$J(u) = \int_0^\infty (x^T Q_u x + u^T R u)\, dt, \tag{3.6}$$

jossa Q_u on symmetrinen positiivisesti semidefiniitti tilojen painomatriisi, R on positiivisesti definiitti ohjausten painomatriisi. Tyypillisesti matriiseille Q_u ja R määritetään vain diagonaaliarvot [1, s. 191], jolloin ne ovat muotoa

$$Q_u = \begin{bmatrix} q_1 & & 0 \\ & \ddots & \\ 0 & & q_n \end{bmatrix}, \qquad R = \begin{bmatrix} \rho_1 & & 0 \\ & \ddots & \\ 0 & & \rho_n \end{bmatrix}, \tag{3.7}$$

jossa arvot $q_1 \dots q_n$ ovat tilojen painotukset ja arvot $\rho_1 \dots \rho_n$ ovat ohjauksen painotukset. Nämä painoarvot määrittelevät, miten paljon kunkin tilan pitäisi vaikuttaa kokonaiskustannukseen, eli suhteellisesti suuri tilan q_n arvo tarkoittaa, että säätö priorisoi kyseisen tilan asetusarvoon pääsemistä ja pysymistä. Suuri ohjauksen painoarvo taas aiheuttaa huonomman säätötuloksen, mutta tällöin ohjausta käytetään vähemmän. [1, s. 191][5, s. 358–360]

Tilatakaisinkytkennän ohjaus on

$$u = -Kx, \tag{3.8}$$

jossa x on tilavektori ja K on $1 \times n$ vahvistusvektori. Tämä vahvistusvektori pystytään ratkaisemaan yhtälöstä

$$K = R^{-1} B^T P, \tag{3.9}$$

jossa P on tuntematon, positiivinen definiitti $n \times n$ matriisi, joka voidaan ratkaista numeerisesti Riccatin yhtälöstä

$$A^T P + PA - PBR^{-1}B^T P + Q_u = 0. \tag{3.10}$$

Algoritmi siis ratkaisee ensin P:n yhtälöstä (3.10), jonka jälkeen tämä matriisi P sijoitetaan yhtälöön (3.9). Tämän yhtälön tuloksena saadaan kustannusfunktion J minimoiva optimaalinen tilatakaisinkytkentävahvistus K. [6, s. 793–795]

4. Q-OPPIMINEN

Q-oppiminen on malliton vahvistusoppimisalgoritmi. Vahvistusoppimisalgoritmissa agentti liikkuu tilajoukossa tekemällä jonkin toiminnon. Toiminnon toteuttaminen tilassa antaa agentille reaalilukuisen palkkion. Agentin tehtävänä on maksimoida sen saama palkkio. Tämä saavutetaan laskemalla yhteen tämänhetkiseen tilaan siirtymisestä kertynyt palkkio ja suurin mahdollinen palkkio tulevista tiloista. Agentti on järjestelmässä oppiva päätöstentekijä. Kaikki agentin ulkopuolinen on määritelty ympäristöksi (environment). Agentti ja ympäristö vuorovaikuttavat toisiinsa, agentin tehdessä toimintoja ympäristö reagoi agentin toimiin. [7, s. 840–841] [8] [9, s. 1–4]

Q-oppimisalgoritmin yhtälö:

$$Q^{uusi}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot max\, Q(s_{t+1}, a) - Q(s_t, a_t)), \tag{4.1}$$

jossa $Q^{uusi}(s_t, a_t)$ on Q-taulun uusi arvo tilassa s_t tehtävälle toimelle a_t, $Q(s_t, a_t)$ on Q-taulun vanha arvo, α on oppimistahti (learning rate), r_t on palkkio (reward), γ on vähennyskerroin (discount factor) ja $max\, Q(s_{t+1}, a)$ on maksimipalkkio, joka voidaan saavuttaa uudesta tilasta s_{t+1}

Q-taulu on matriisi, joka koostuu kaikista järjestelmän tila- ja toimintopareista. Q-taulun rivit kuvaavat järjestelmän diskreettejä tiloja ja sarakkeet kuvaavat näissä tiloissa tehtäviä toimija. Q-taulun arvoja päivitetään kaavan (4.1) mukaan. Q-taulun arvoja voidaan koulutuksen jälkeen hyödyntää valitsemalla paras mahdollinen toiminta kyseisessä tilassa.[9]

Agentti pystyy käyttämään Q-taulun arvoja kahdella tavalla, joita kutsutaan tutkimiseksi (exploring) ja hyödyntämiseksi (exploiting). Tutkiessa agentti valitsee toiminnon satunnaisvaraisesti, jolloin agentti tutkii uusia tiloja, joita se ei välttämättä käyttäisi hyödyntämistilassa. Hyödyntämisessä agentti käyttää Q-taulun arvoja valitakseen parhaan palkinnon toteuttavan toiminnon. [8]

Oppimistahti α määrittää uuden informaation vaikutuksen vanhaan informaatioon. Oppimistahdin minimiarvo 0 aiheuttaa sen, että kaava yksinkertaistuu muotoon

$$Q^{uusi}(s_t, a_t) \leftarrow Q(s_t, a_t), \tag{4.2}$$

jolloin algoritmi käyttää vain vanhaa tietoa, eli algoritmi ei ota uuteen tilaan siirtyessä

tapahtuvasta toimesta saatavaa palkkiota huomioon. Oppimistahdin maksimiarvo 1 saa kaavan muotoon

$$Q^{uusi}(s_t, a_t) \leftarrow r_t + \gamma \cdot max\ Q(s_{t+1}, a), \tag{4.3}$$

jolloin algoritmi käyttää vain uutta tietoa.

Vähennyskerroin γ vaikuttaa uuden palkkion suuruuteen. Vähennyskertoimen arvot ovat välillä $0 \leq \gamma \leq 1$, jolloin vähennyskertoimen arvolla 0 algoritmi asettaa ajanhetkellä t tulevan ajanhetken $t + 1$ palkkion nollaksi, eli se ei harkitse tulevia palkkioita. Tällöin kaava (4.1) saadaan muotoon

$$Q^{uusi}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t - Q(s_t, a_t))\,. \tag{4.4}$$

Arvolla 1 algoritmi pyrkii maksimoimaan pitkäaikaisen palkkion. Kuitenkin yhtä pienemmille vähennyskertoimen arvoille tulevan hetken palkkiot muuttuvat jossain vaiheessa merkityksettömiksi. [8, s. 41]

4.1 Alkuarvot

Q-oppimisessa käytetty palkkiomatriisi Q alustetaan ohjelmoijan toimesta tiettyyn arvoon. Yksinkertaisimmillaan matriisin rivit ovat tiloja tilajoukossa ja sarakkeet toimintoja toimijoukossa. Ohjelmoijan määrittelemät palkkiot tilojen välissä siirtymiseen lasketaan uudeksi arvoksi matriisiin kaavan (4.1) mukaan. Suuret alkuarvot palkkiomatriisissa saavat algoritmin tutkimaan eri tilojen välisiä toimintoja, koska matriisin suuret alkuarvot aiheuttavat päivitetyn arvon olevan pienempi kuin muut arvot, jolloin algoritmi kokeilee todennäköisemmin muita vaihtoehtoja kyseisessä tilassa, täten kannustaen eri toimien tutkimista. [7, s. 842–844]

5. KÄÄNTÖHEILURI

Kääntöheiluri on eräs tyypillinen tasapainojärjestelmä. Tasapainojärjestelmässä massakeskipiste sijaitsee nivelkohdan yläpuolella, jonka takia pienikin häiriö voi kaataa tasapainojärjestelmän. Yläasento on epästabiili tasapainopiste. Kääntöheilurin tapauksessa heiluri pysyy teoriassa pystyssä, mikäli siihen ei kohdistu mitään ulkoisia voimia. Kuitenkin erilaiset häiriöt, kuten ilmavirtaukset, alustan tärinä yms. voivat aiheuttaa heilurin kulman muuttumisen, joka johtaa epästabiiliuteen, eli heilurin kaatumiseen. Heilurin stabilointi häiriöistä huolimatta voidaan saavuttaa takaisinkytkennällä. [1, s. 34-37]

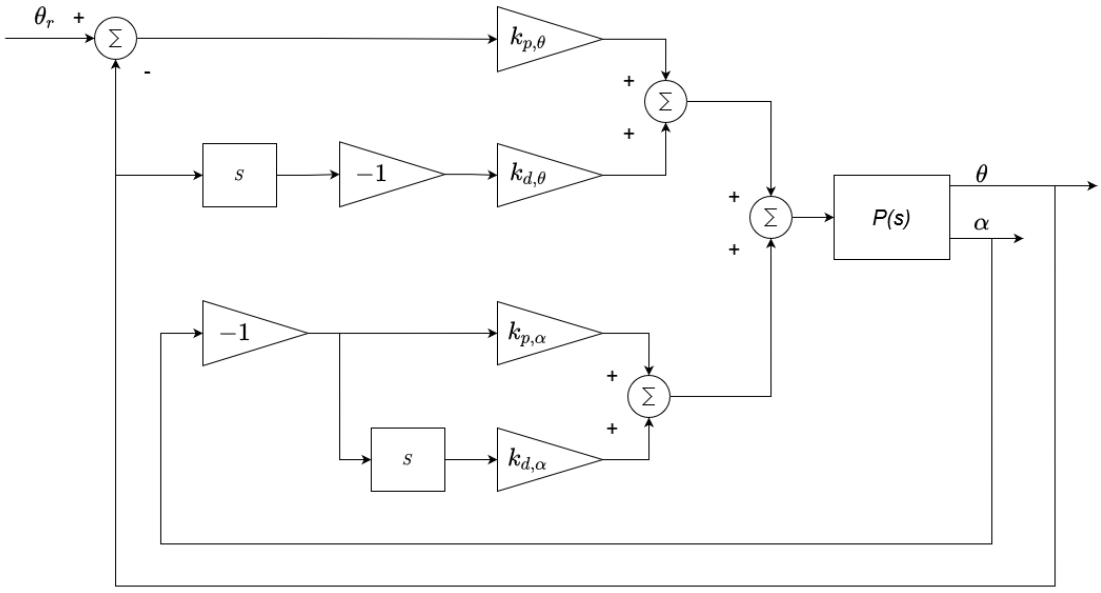
5.1 QUBE Servo 2 -järjestelmä

QUBE Servo 2 on Quanserin tuottama opetustyökalu säätötekniikkaan ja mekatroniikkaan. QUBE Servo 2 koostuu kahdesta osasta, pohjalaatikosta, jossa on magneettinen kiinnityskohta, jonka kulmaa ohjataan moottorilla, sekä magneettiosaan kiinnitettävästä kiekosta, tai kääntöheilurista. QUBE Servo 2 eroaa perinteisestä kärryllä kulkevasta kääntöheilurista, sillä QUBE Servo 2:ta ohjataan heiluria kiertämällä, eikä kärryn edestakaisella liikkeellä. Opetuskäyttöön tarkoitetulla QUBE Servo 2:lle on luotu useita harjoituslaboratoriotöitä, mutta tässä työssä keskitytään heilurin tasapainotukseen. [10]

Alkutilassa heiluri nostetaan pystyasentoon joko käsin tai automaattisesti, minkä jälkeen säädin pyrkii pitämään heilurin pystyasennossa. Säädön laatua voidaan kokeilla aiheuttamalla häiriöitä heilurin asemaan, eli tökkäämällä heiluria sen pyörimisakselin suunnassa tai pyytämällä säädintä siirtämään heiluri pystyasennossa paikasta toiseen kaatumatta. Simulointiin käytettiin Quanserin toimittamaa linearisoitua tilamallia, jossa heilurin nollakulma tarkoittaa heilurin pystyssä pysymistä. [10]

5.2 Valmistajan 2DOF-PID-säätö

Quanser on toteuttanut kääntöheilurille säätöratkaisun, jossa on kaksi PD-säädintä. Toinen PD-säädin säätää kuvan 5.1 mukaisesti moottorin kulmaa θ, ja toinen PD-säädin säätää heilurin kulmaa α. Kummatkin PD-säätimet ovat toteutettu niin, että niiden derivointihaarojen sisäänmeno on mittaus. Valmistajan on toteuttanut säädön kuvan 5.1 mukaisesti, jossa θ_r on moottorin kulman asetusarvo, $k_{p,\theta}$ ja $k_{d,\theta}$ moottorin säätimen P-haaran ja D-haaran vahvistukset, sekä $k_{p,\alpha}$ ja $k_{d,\alpha}$ heilurin säätimen P-haaran ja D-haaran vahvistukset, joille valmistaja on antanut arvot $k_{p,\theta} = -2$, $k_{d,\theta} = -2$, $k_{p,\alpha} = 30$ ja $k_{d,\alpha} = 2.5$. Kuvassa ei ole I-haaroja, sillä niiden vahvistukset ovat $k_{i,\theta} = k_{i,\alpha} = 0$.



Kuva 5.1. *QUBE Servo 2:lle toteutettu säätö kahdella 2DOF-PID-säätimellä*

Valmistajan 2DOF-PID-säätö koostuu kahdesta 2DOF-PID-säätimestä, joista θ-suureilla merkitty säädin ohjaa kiertokulmaa ja α-suureilla merkitty ohjaa heilurin kulmaa. Kiertokulmaa ohjaava säädin on toteutettu asetusarvopainoilla $b = 1$ ja $c = 0$ ja heilurin kulmaa ohjaava säädin on toteutettu asetusarvopainoilla $b = 0$ ja $c = 0$.

5.3 Q-oppiva PID-säätö

Q-oppivan säädön idea kehitettiin lähteiden [11] ja [12] pohjalta. Tämän työn koneoppimisalgoritmin avulla lasketaan kuitenkin vain PID-säädinten vahvistusparametrit, eikä toteuteta adaptiivista PID-säädintä, kuten lähteessä [12] on tehty.

Q-oppiva PID-säätö on toteutettu identtisillä 2DOF-PID-säätimillä, mutta säädinten vahvistusparametrit on laskettu Q-oppimisalgoritmillä. Q-opetusalgoritmi on toteutettu Q-taululla, jonka tilajoukkona on heilurin kulmat, sekä toimijoukkona viritysparametrin arvo. Tilat on

diskretisoitu matriisin riveiksi kaavalla

$$States = \begin{cases} 1, & \text{jos } round(\alpha) < -10^\circ \\ 2, & \text{jos } -10^\circ <= round(\alpha) < -5^\circ \\ 3, & \text{jos } -5^\circ <= round(\alpha) < 0^\circ \\ 4, & \text{jos } round(\alpha) = 0^\circ \\ 5, & \text{jos } 0^\circ < round(\alpha) <= 5^\circ \\ 6, & \text{jos } 5^\circ < round(\alpha) <= 10^\circ \\ 7, & \text{jos } round(\alpha) > 10^\circ, \end{cases} \tag{5.1}$$

jossa pyöristäminen on toteutettu lähimpään kokonaislukuun.

Q-taulujen toimijoukko on valittu väliltä $(0, 50)$. Toimijoukkoon valittiin vain kokonaislukuja Q-taulun arvojen rajoittamiseksi, sekä opetuksen nopeuttamiseksi.

Algoritmin oppimistahdiksi valittiin $\alpha = 0.9$, sekä vähennyskertoimeksi valittiin $\gamma = 0.9$. Algoritmi hyödyntää myös ϵ greedy -menetelmää, jossa algoritmi valitsee joko tutkimisen (satunnainen toiminta) tai tutkimisen (maksimipalkkion antama toiminta tilassa) todennäköisyydellä ϵ. Tämä saa kaavamuodon

$$Actions = \begin{cases} maxQ(a), & \text{todennäköisyydellä } 1 - \epsilon \\ rand(a), & \text{todennäköisyydellä } \epsilon. \end{cases} \tag{5.2}$$

Algoritmiin lisättiin myös ϵ:n heikkenevä määrittely

$$\epsilon(episodi) = max \begin{cases} 1 - \frac{episodi}{maksimiepisodi} \\ 0.05 \end{cases} \tag{5.3}$$

jossa $maksimiepisodi$ on koulutusepisodien maksimimäärä ja $episodi$ nykyinen koulutusepisodi. Koulutusepisodilla tarkoitetaan yhtä tilojen ja toimien sarjaa, mikä päättyy lopputilaan (heilurin kaatuminen) tai simulointiajan päättymiseen. Episodien kasvaessa satunnaisen toimen todennäköisyys pienenee, mutta sen todennäköisyys on aina vähintään 5%. Tämä auttaa välttämään lokaaliin minimiin jumittumista.

Algoritmin palkkiot heilurin kulman ohjaimelle määriteltiin seuraavasti

$$r_{t,\alpha} = \begin{cases} 2, & \text{jos } |\alpha_t| \leq 0 \\ 1, & \text{jos } |\alpha_t| < |\alpha_{t-1}| \text{ ja } |\alpha_t| > 0 \\ -1, & \text{muuten,} \end{cases} \tag{5.4}$$

jossa α_t on heilurin kulman nykyinen tila ja α_{t-1} on vanha tila. Alustan kulman ohjaimelle

on määritelty palkkiot

$$r_{t,\theta} = \begin{cases} 2, & \text{jos } |\alpha_t| < |\alpha_{t-1}| \text{ ja } |\theta_t| < |\theta_{t-1}| \\ 1, & \text{jos } |\theta_t| < |\theta_{t-1}| \\ -1, & \text{muuten,} \end{cases} \tag{5.5}$$

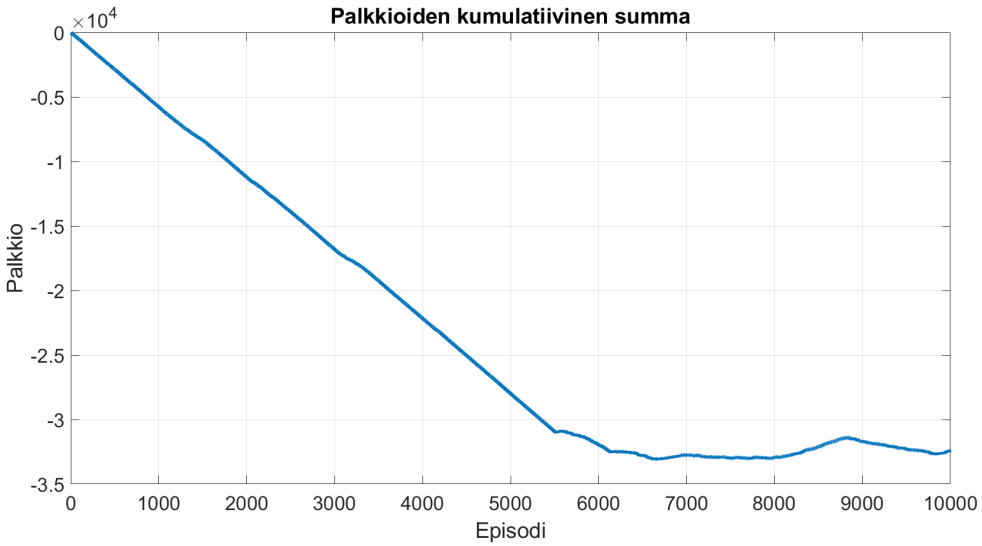
jossa θ_t on alustan kulman nykyinen tila ja θ_{t-1} on vanha tila.

Tilojen välinen aika-askel on määritelty 10 sekunnin pituiseksi. Tämä todettiin kokeilemalla hyväksi, koska lyhyemmät aika-askeleet joko eivät löytäneet stabiilia säätöä, tai vaihtoivat ohjausta liian aggressiivisesti.

Q-oppimisalgoritmin palauttamasta taulusta laskettiin PID-säätimien parametrit summaamalla kaikkien tilojen palkkiot toimille ja valitsemalla näistä suurimman palkkion palauttavan arvon. Täten algoritmi valitsee parametrit, jotka useimmiten johtivat heilurin pääsemisen stabiilin tilaan.

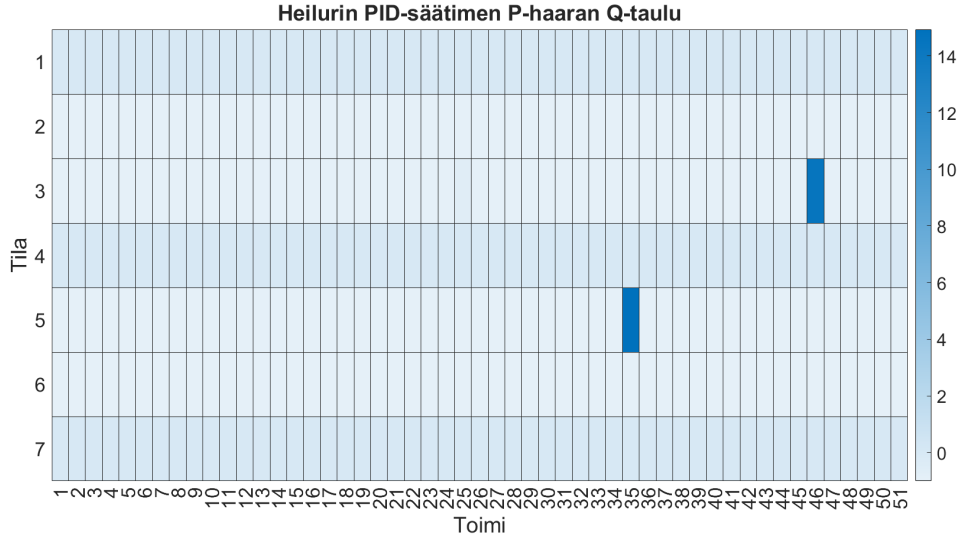
Algoritmin koulutus tapahtui liitteessä 7 esitellyn Simulink-mallin avulla, jota simuloitiin liitteessä 7 esitellyn koodin avulla. Liitteen 7 2DOF-PID-säätimet on toteutettu rinnakkaismuodossa (parallel). Alipäästösuotimen leikkaustaajuus N jätettiin koneoppimisongelman ulkopuolelle ja arvoksi asetettiin oletusarvo 100. Alipäästösuotimelle leikkaustaajuuden arvo 100 on suuri, jonka takia säätimiin kytkeytyy paljon turhaa kohinaa. Oletusarvo 100 valittiin, koska kaikki asetusarvopainot saatiin valmistajan toteutuksesta ja säätimen vahvistusparametrit koneoppimisalgoritmista. Muut parametrit jätettiin vakioarvoisiksi, jotta saataisiin vertailukohta osaamattoman ihmisen toteuttaman LQR-säädön ja 2DOF-PID-säädön välille.

Malliin oli lisätty valkoista kohinaa heilurin ja alustan kulmamittauksiin, jäljittelemään oikean järjestelmän mittauskohinoita. Algoritmin koulutuksessa alustettiin aina uusi simulaatio siten, että heilurin kulma α sai satunnaisen alkuarvon joukosta $\{8, 1, -3, -6\}$. Simulaatioaika oli kymmenen sekuntia, jonka jälkeen Q-oppimisalgoritmi laski palkkiot tauluun. Näitä episodeja tehtiin 10000 kappaletta, joiden yhteinen suoritusaika oli noin 2,5 tuntia.



Kuva 5.2. *Q-oppimisalgoritmin palkkioiden kumulatiivinen summa*

Kuvasta 5.2 voidaan havaita, miten algoritmi saa aluksi paljon negatiivista palkkiota, mutta satunnaisten toimien vähetessä palkkioiden kumulatiivinen summa tasaantuu, koska algoritmi löytää käytettäviä ja toimivia säätöjä osalle tiloista. Kaikille tiloille säädön löytäessä kumulatiivinen summa lähtisi nousemaan episodien kuluessa. Kuvasta 5.3 voidaan havaita, että algoritmi löytää kaksi vaihtoehtoista parametria tiloille 3 ja 5, mutta tiloille 2 ja 6 ei löydy optimiratkaisua. Tästä johtuen algoritmi saa negatiivista palkkiota, jos heilurin aloituskulma kuuluu näihin tiloihin.



Kuva 5.3. *α-kulman säätimen P-haaran Q-taulun lämpökartta*

Lämpökartasta 5.3 voidaan havaita eräs ongelma Q-oppimisessa usean agentin tapauksessa: mikäli agentti löytää sopivan I-haaran virityksen, voi sillä olla kyseisessä tilanteessa epäsopiva P- tai D-haaran viritys, jolloin säätö kokonaisuudessaan päätyy epäs-

tabiiliksi, jolloin myös tämän sopivan I-haaran palkkio kyseiselle arvolle saa negatiivisen palkkion.

Erääksi ongelmaksi kouluttaessa koitui algoritmin tilajoukon kouluttamiseen vaadittu aika. Tässä algoritmissa Q-taulussa oli $7 \times 50 \times 6$ -kokoinen matriisi, joka tarkoitti 2100:aa muuttujaa. Mikäli tilat olisi diskretisoitu asteen välein, ja toimet olisivat 0.1 välein, olisi taulun muuttujien määrä $23 \times 500 \times 6$, jolloin matriisissa olisi jo 69000 muuttujaa. Suuremmasta tilajoukosta oikeiden säätöparien löytäminen muuttuu erittäin hankalaksi ja epätodennäköiseksi. Tässä algoritmissa olisi saattanut olla kannattavaa käyttää vielä pienempää määrää tiloja, koska algoritmi ei löytänyt tiloille 2 ja 6 sopivia säätöparametreja, vaikka nämä tilat olisivat olleet säädettävissä.

6. SÄÄTÖJÄRJESTELMIEN SIMULOINTI- JA KOKEELLISET TULOKSET

Säätöjärjestelmien suorituskyvyn arviointiin voidaan käyttää esimerkiksi Gang-of-Six-siirtofunktiokokoelmaa. Tässä työssä kuitenkin keskitytään vain järjestelmän vasteeseen, eli ulostulojen mittauksiin.

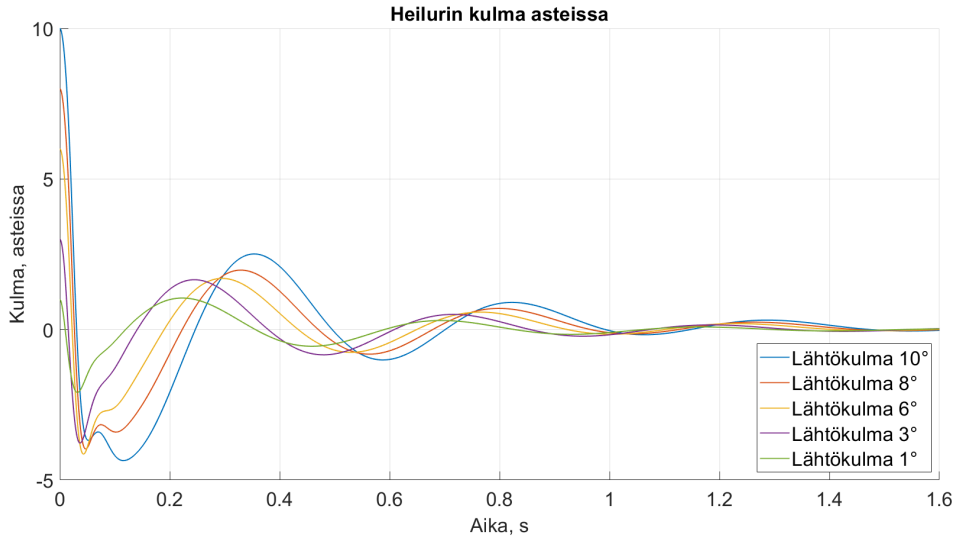
6.1 Simulointitulokset

Simuloidut tulokset saatiin simuloimalla heilurin ja alustan kulman vasteet jokaiselle heilurin kulman aloitusarvolle $\{10°, 8°, 6°, 3°, 1°\}$. Näiden vasteiden tulokset on esitelty seuraavissa alaluvuissa. Simuloidut vasteet on esitetty aika-akseliltaan identtisinä vertailun helpottamiseksi. Simulaattorissa ohjausjännite on rajoitettu välille $[-10, 10]$ volttia.

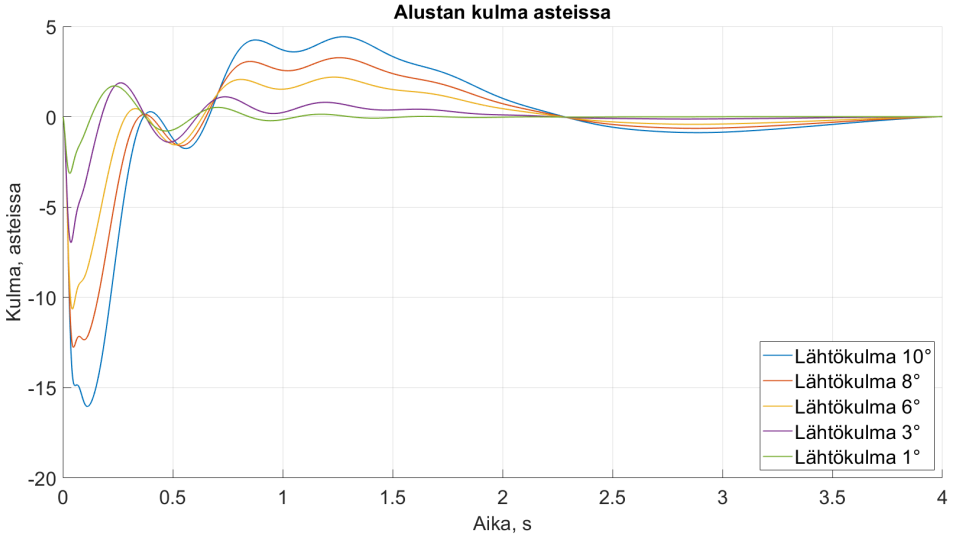
6.1.1 Q-oppiva 2DOF PID-säätö

Q-oppimisalgoritmin määrittelemät säätöparametrit ovat heilurille: $k_{p,\alpha} = 34$, $k_{i,\alpha} = 17$, $k_{d,\alpha} = 6$ ja alustalle: $k_{p,\theta} = -7$, $k_{i,\theta} = -15$, $k_{d,\theta} = -4$.

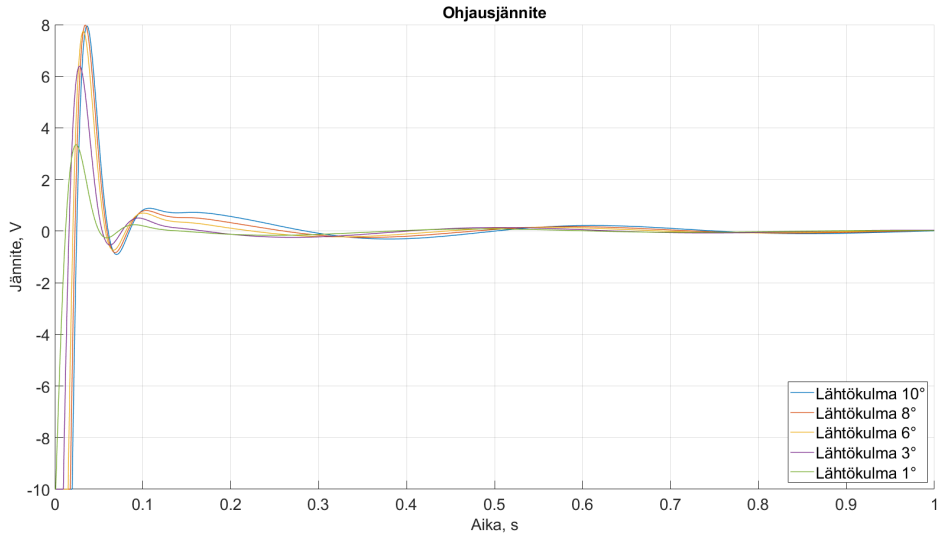
Kuvassa 6.1 on esitelty heilurin kulman vaste, kuvassa 6.2 on esitelty alustan kulman vaste ja kuvassa 6.3 on esitelty moottorin ohjausjännite.



Kuva 6.1. *Heilurin kulma Q-oppimisalgoritmin 2DOF-PID-säädöllä*



Kuva 6.2. *Alustan kulma Q-oppimisalgoritmin 2DOF-PID-säädöllä*

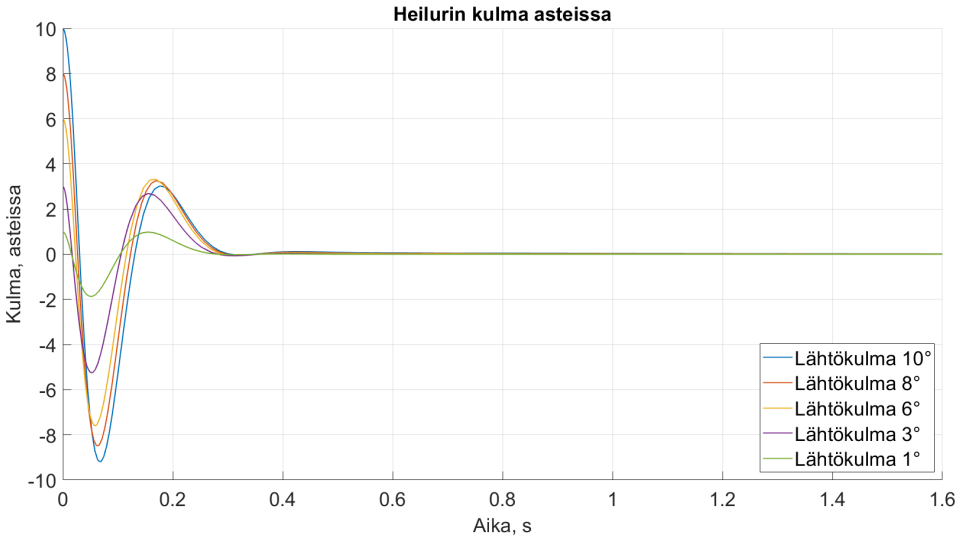


Kuva 6.3. *Ohjausjännite Q-oppimisalgoritmin 2DOF-PID-säädöllä*

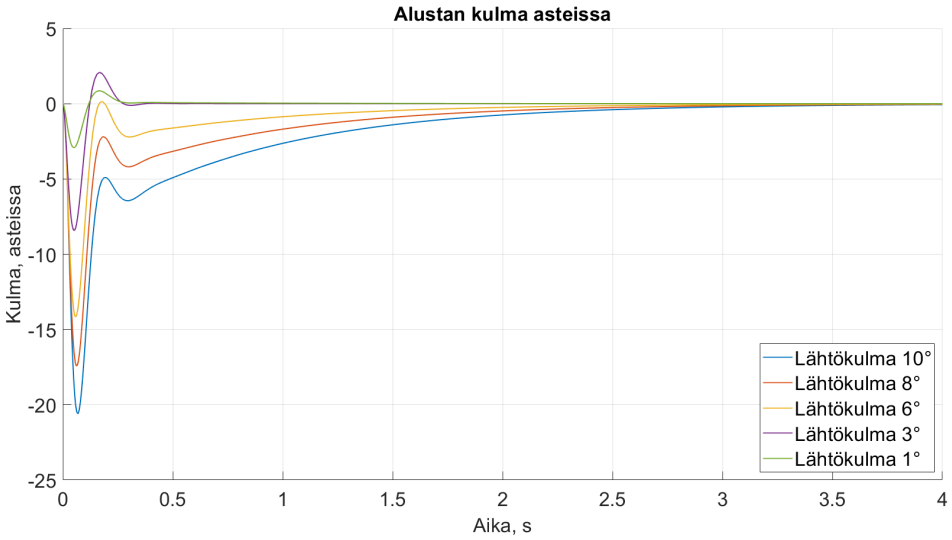
Heilurin kuvaajasta voidaan havaita säädön aiheuttama värähtely heilurin kulman vasteessa, sekä alustan kulman edestakainen värähtely välillä $0.8s - 2s$.

6.1.2 Valmistajan 2DOF-PID-säätö

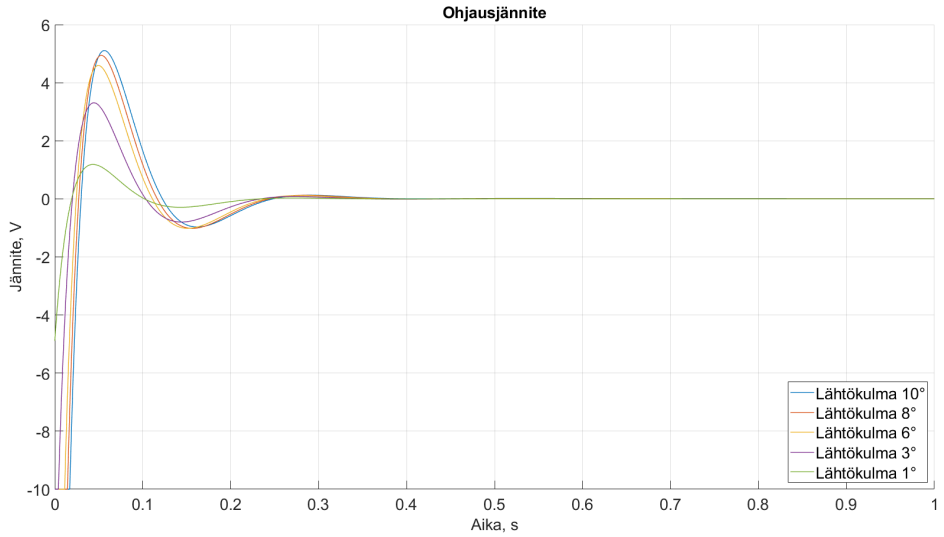
Valmistajan toteuttamille säätöparametreille ei ole määritelty laskumenetelmää, mutta ne ovat kuitenkin hyvä lähtökohta säädön vertailulle. Kuvassa 6.4 on esitelty heilurin kulman vaste, kuvassa 6.5 on esitelty alustan kulman vaste ja kuvassa 6.6 on esitelty moottorin ohjausjännite.



Kuva 6.4. *Heilurin kulma valmistajan 2DOF PID-säädöllä.*



Kuva 6.5. *Alustan kulma valmistajan 2DOF PID-säädöllä.*



Kuva 6.6. *Ohjausjännite valmistajan 2DOF PID-säädöllä*

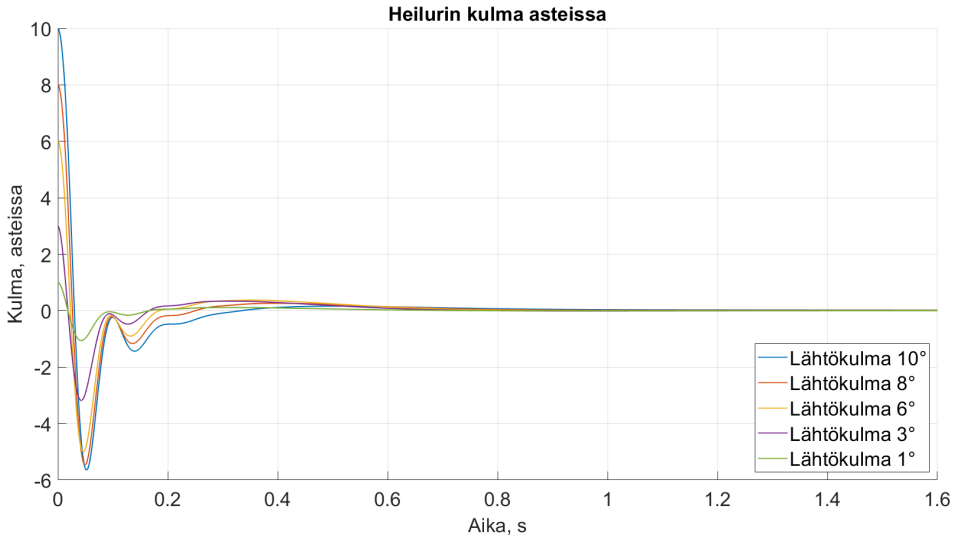
Valmistajan säätö on erittäin mielenkiintoinen verrattava Q-oppivan säädön tuloksiin, sillä ne käyttävät identtistä säädinrakennetta. Valmistajan säätö näyttää olevan huomattavasti robustimpi, sillä se värähtelee vähemmän. Toisaalta valmistajan säädön kulmat käyvät suuremmissa arvoissa, joka saattaisi johtua Q-oppimisalgoritmin painotuksesta kulman pienenemiselle. Erityisesti alustan kulma vaikuttaa paljon järkevämmältä valmistajan säädössä.

6.1.3 Valmistajan LQR-säätö

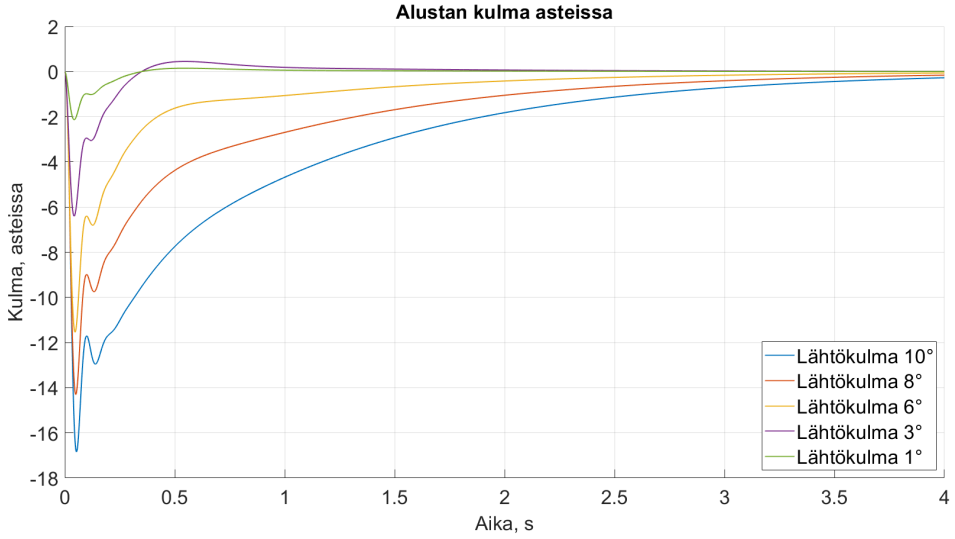
LQR-säädin on viritetty painomatriiseilla $Q = diag\{1, 5, 1, 1\}$ ja $R = 1$, eli tilatakaisinkytkennässä painotetaan järjestelmän toista tilaa, eli heilurin kulmaa. Tilatakaisinvahvistus

on $K = \{-1, 35.1039, -1.4480, 3.0923\}$.

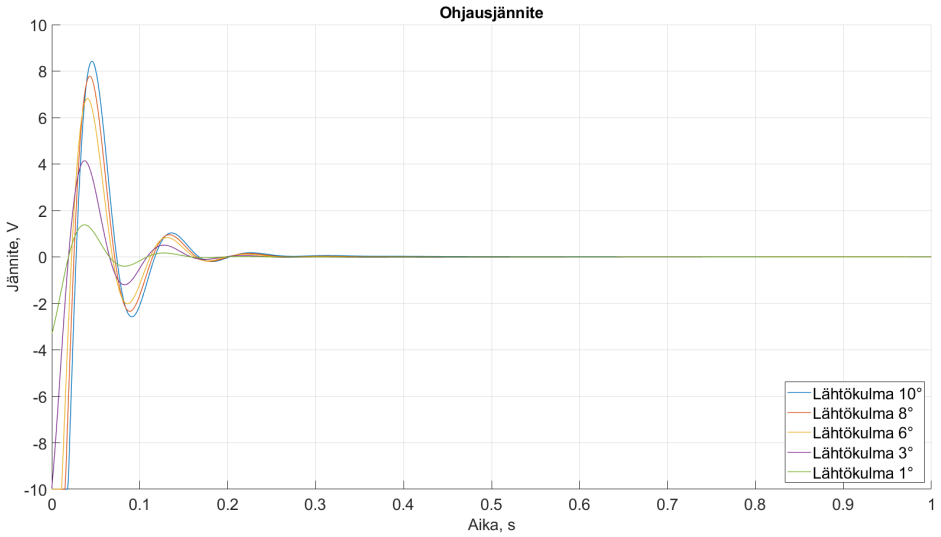
Kuvassa 6.7 on esitelty heilurin kulman vaste, kuvassa 6.8 on esitelty alustan kulman vaste ja kuvassa 6.9 on esitelty alustan ohjausjännite.



Kuva 6.7. *Heilurin kulma tilatakaisinkytkennällä.*



Kuva 6.8. *Alustan kulma tilatakaisinkytkennällä.*



Kuva 6.9. *Ohjausjännite tilatakaisinkytkennällä*

LQR-säädössä pääsee huomattavasti paremmin heilurin asetusarvoon, mutta alustan kulma vaikuttaa saavuttavan asetusarvonsa hitaammin. Toisaalta tämä on asetettujen painojen mukainen tulos.

6.2 Säätöjen vertailu

Q-oppiva säädön kulma värähtelee pidempään kuin valmistajan PID-säätö. Toisaalta Q-oppivan säädön maksimiheilahdus kaikille lähtökulmille on pienempi kuin valmistajan PID-säädön, sekä alustan kulma heilahtaa vähemmän Q-oppivassa säädössä. Parhaan heilurin kulman vasteen antaa kuitenkin LQR-säätö. LQR-säädön alustan kulman stabiloitumisessa kuluu paljon aikaa, mutta toisaalta tämän takia heilurin maksimikulma pysyy pienenä. LQR:n ohjausjännite on myös muodoltaan ja kooltaan melko samanlainen Q-oppivan säädön kanssa, mutta erityisesti 1:n asteen lähtökulmassa LQR-ohjauksen ohjausjännite lähtee -5:stä voltista, kun taas Q-oppivan säädön lähtee suurimmasta mahdollisesta, eli -10:stä voltista. Ohjaukseltaan pienintä jännitettä käyttää valmistajan PID-säätö, sillä sen aloituskulman jälkeisen piikin suurin ohjausjännite on noin 5 volttia verrattuna LQR:n ja Q-oppivan PID-säädön 8:aan volttiin.

Taulukossa 6.1 on koottuna eri säätömenetelmien maksimiheilahdukset ja asettumisajat heilurille. Taulukossa 2DOF-PID tarkoittaa valmistajan toimittamilla parametreillä viritettyä säädintä.

Säätötapa	Maksimiheilahdus	Asettumisaika
Q-oppiminen	-4°	1.6 s
2DOF-PID	-9°	0.4 s
LQR	-6°	0.6 s

Taulukko 6.1. *Heilurin vasteet -10° lähtökulmasta eri säätömenetelmillä.*

Taulukossa 6.2 on taulukon 6.1 tavoin koottu alustan vasteet eri säätömenetelmille. Huomionarvoista on, että Q-oppimissäätö oli ainoa, joka aiheutti myös positiivisen alustan kulman poikkeaman kuvan 6.2 mukaisesti.

Säätötapa	Maksimiheilahdus	Asettumisaika
Q-oppiminen	-16°	4 s
2DOF-PID	-20°	3 s
LQR	-17°	4 s

Taulukko 6.2. *Alustan vasteet -10° lähtökulmasta eri säätömenetelmillä.*

Tämän järjestelmän tapauksessa LQR tuottaa kaksi PD-säädintä, eli se on vain tapa laskea PD-säätimille viritysparametrit.

7. YHTEENVETO

Tässä työssä tutkittiin PID-säädinten, tilatakaisinkytkennän ja Q-oppimisalgoritmin toimintaa, sekä luotiin Q-oppimisalgoritmi kääntöheilurin 2DOF PID-säätimen parametrien virittämiseen. Viritysten suorituskykyä vertailtiin valmistajan valmiiksi toteuttamiin 2DOF PID-säätimen parametreihin, sekä LQR-menetelmällä viritettyyn tilatakaisinkytkentään. Q-oppimisalgoritmilla saatiin toimiva viritys, mutta algoritmi ei löytänyt ratkaisua jokaisella kerralla.

Keskeinen haaste työssä oli säätöteknisesti hankala ongelma yhdistettynä suureen määrään vapaita muuttujia. Tasapainotusongelman luonteen takia suuri osa virityksistä olivat huonoja, mutta algoritmin oli mahdotonta määrittää, mistä muuttujasta ja Q-taulusta tämä johtui, joten tietyn muuttujan hyvätkin viritykset saattoivat jäädä löytymättä. Myös hyperparametrien määrittämiseen kului paljon aikaa, sekä algoritmin vaatima laskenta-aika vaikeutti erilaisten parametrien vaikutusten kokeilemista.

Simulaattorissa Q-oppimisalgoritmin tulokset olivat osittain kilpailukykyisiä muiden viritysten kanssa. Kuitenkin algoritmin tuottamien vahvistusten arvot ovat niin suuret, että niiden toimiminen fyysisessä laitteessa on epätodennäköistä.

LÄHTEET

[1] K. J. Åström and R. M. Murrey, *Feedback systems : an introduction for scientists and engineers*, eng. Princeton, NJ: Princeton University Press, 2008.

[2] K. J. Åstrom and T. Hagglund, *PID controllers: theory, design, and tuning, 2nd edition*. International Society of Automation (ISA), 1995.

[3] K. J. Åström, *Advanced PID control*. Research Triangle Park, NC: ISA, 2006.

[4] N. S. Nise, *Control Systems Engineering*, 6th ed. Wiley, 2010.

[5] R. L. Williams and D. A. Lawrence, *Linear State-Space Control Systems*, eng. 2007.

[6] K. Ogata, *Modern Control Engineering*, 5. painos. Prentice Hal, 2010.

[7] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning", in *Artificial Neural Networks – ICANN 2006*, ser. Lecture Notes in Computer Science, vol. 4131, Springer Berlin Heidelberg, 2006, pp. 840–849.

[8] C. J. C. H. Watkins, "Learning from delayed rewards", English, Ph.D. dissertation, King's College, Oxford, 1989.

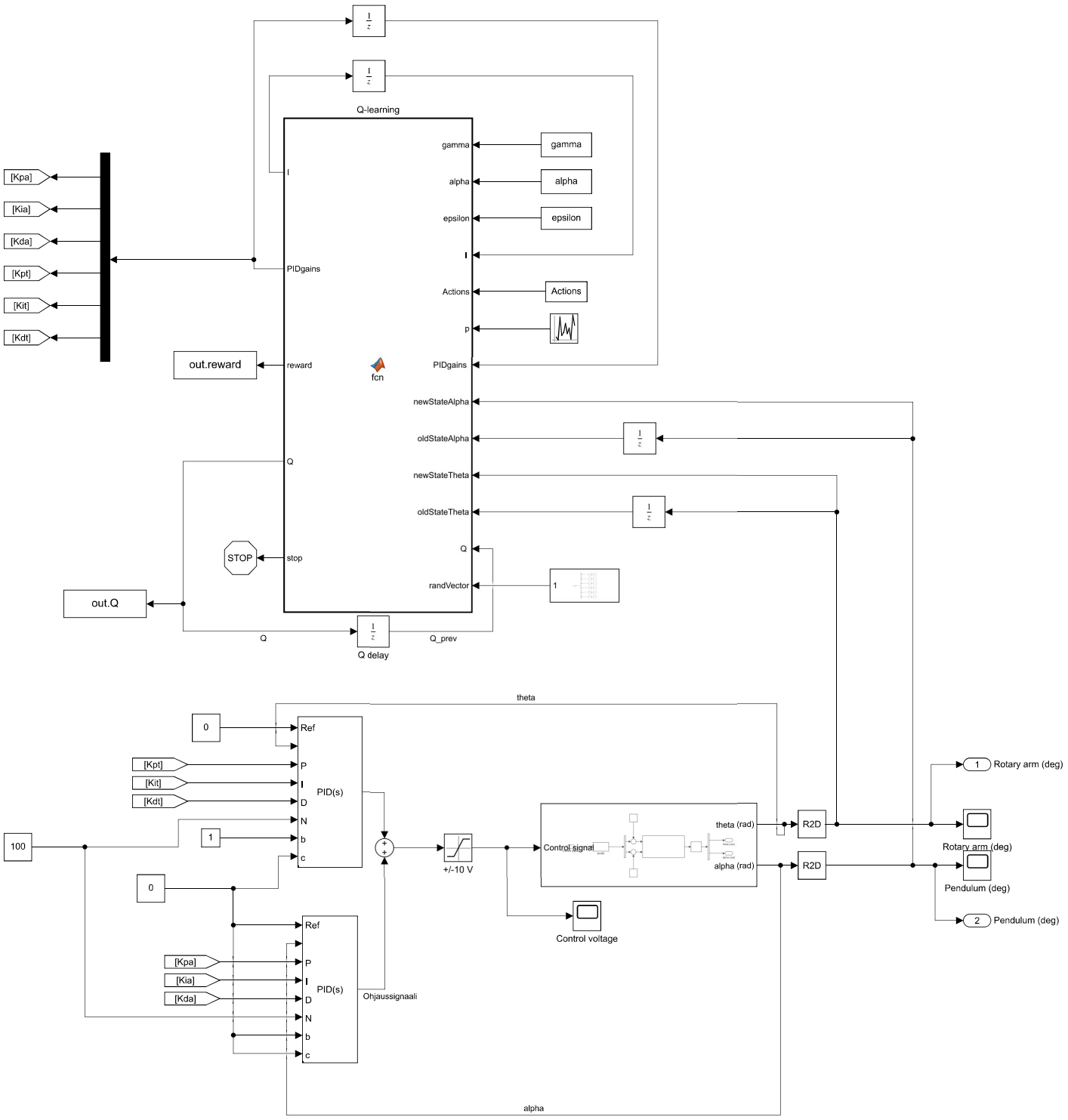
[9] R. S. Sutton, *Reinforcement learning: an introduction*. The MIT Press, 2018.

[10] *QUBE - Servo 2 - Quanser*. url: https://www.quanser.com/products/qube-servo-2/, (viitattu 19.10.2021).

[11] I. Carlucho, M. D. Paula, S. A. Villar, and G. G. Acosta, "Incremental q-learning strategy for adaptive pid control of mobile robots", *Expert Systems with Applications*, vol. 80, pp. 183–199, 2017.

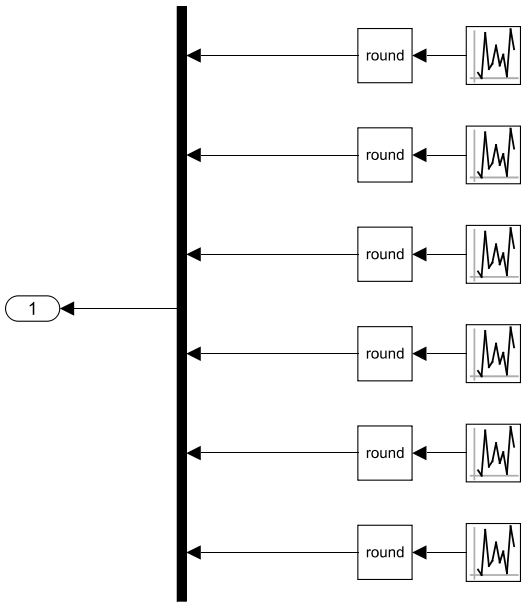
[12] Q. Shi, H.-K. Lam, B. Xiao, and S.-H. Tsai, "Adaptive pid controller based on q-learning algorithm", *CAAI Transactions on Intelligence Technology*, vol. 3, no. 4, pp. 235–244, 2018.

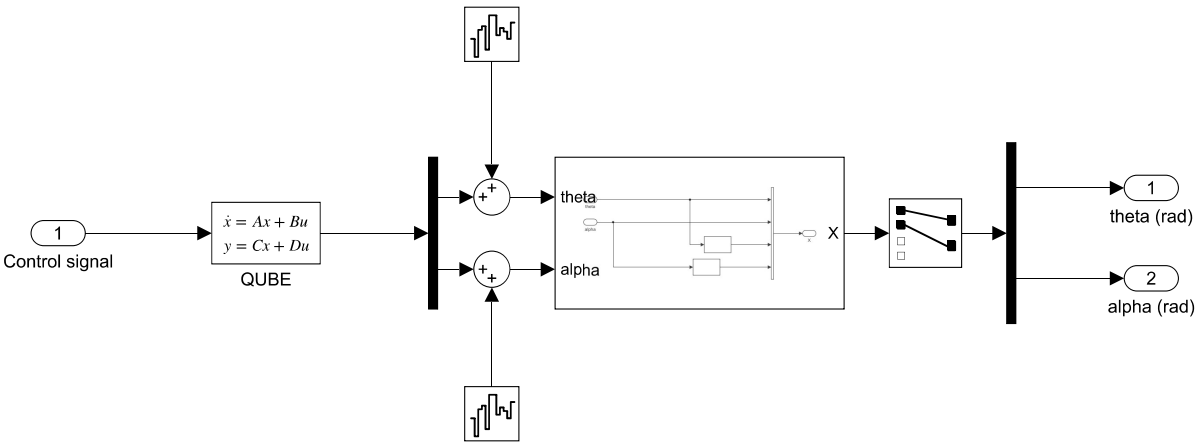
LIITE A: MATLAB, QUBE:N OPETUSSIMULAATTORI

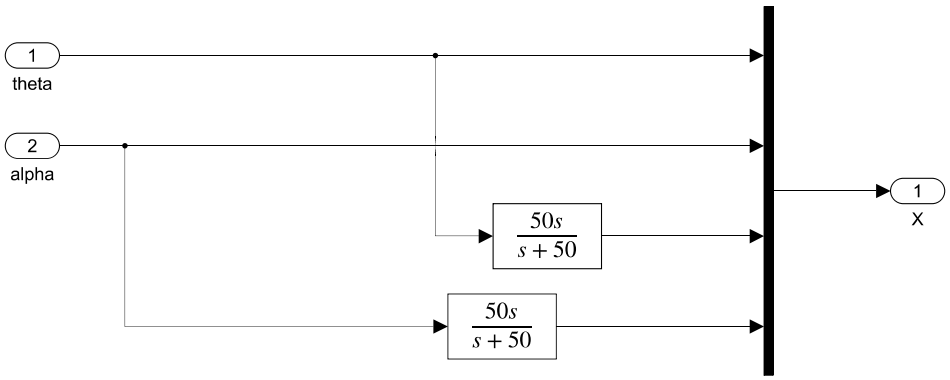


```
function [I, PIDgains, reward, Q, stop]= fcn(gamma, alpha, epsilon, I, Actions, p, PIDgains, newStateAlpha, oldStateAlpha, newStateT
reward = inf;
stop = 0;

% Diskretisöi vanha heilurin kulma
if oldStateAlpha > 10
    tableState = 7;
elseif oldStateAlpha < -10
    tableState = 1;
elseif (oldStateAlpha >= -10) && (oldStateAlpha < -5)
   tableState = 2;
elseif (oldStateAlpha >= -5 && oldStateAlpha < 0)
   tableState = 3;
elseif (oldStateAlpha > 0) && (oldStateAlpha <= 5)
   tableState = 5;
elseif (oldStateAlpha > 5) && (oldStateAlpha <= 10)
   tableState = 6;
else
   tableState = 0;
end

% Diskretisöi uusi heilurin kulma
if newStateAlpha > 10
    newTableState = 7;
elseif newStateAlpha < -10
    newTableState = 1;
elseif (newStateAlpha >= -10) && (newStateAlpha < -5)
   newTableState = 2;
elseif (newStateAlpha >= -5 && newStateAlpha < 0)
   newTableState = 3;
elseif (newStateAlpha > 0) && (newStateAlpha <= 5)
   newTableState = 5;
elseif (newStateAlpha > 5) && (newStateAlpha <= 10)
   newTableState = 6;
else
   newTableState = 0;
end

% Määrittele reward1 heilurin kulmaohjaimelle
if (abs(newStateAlpha) <= 0.1)
    reward1 = 2;
elseif abs(newStateAlpha) < abs(oldStateAlpha)
    reward1 = 1;
else
    reward1 = -1;
end


% Määrittele reward2 kiekon kulmaohjaimelle
if (abs(newStateAlpha) < abs(oldStateAlpha) && abs(newStateTheta) < abs(oldStateTheta))
    reward2 = 2;
elseif (abs(newStateTheta) < abs(oldStateTheta))
    reward2 = 1;
else
    reward2 = -1;
end


rewards = [reward1 reward1 reward1 reward2 reward2 reward2];
ind = sub2ind(size(Q), tableState*ones(1,6), I.', 1:6);
```

```
% Terminal state
% if(abs(newStateAlpha) > 10)
%     Q(ind) = -3;
% Stable state
%elseif (tableState == 0) && (newTableState == 0)
%else

if (PIDgains ~= inf*ones(1,6))
    Q(ind) = Q(ind) + alpha.*(rewards + gamma.*reshape(max(Q(newTableState,:,:)), 1, [])- Q(ind));
    reward = sum(rewards);
end



if abs(newStateAlpha) > 45
    stop = 1;
% elseif abs(newStateAlpha) < 0.1 &&  abs(oldStateAlpha) < 0.1
%     stop = 1;
end


% Valitse uusi toimijoukko
if p < epsilon

    I = randVector;
    %I = randi([1, size(Actions, 2)], 6,1);
    PIDgains = Actions(I).';
    PIDgains(4:end) = -PIDgains(4:end);
else
    [~,It] = max(Q(tableState,:,:), [], 2);
    I = reshape(It, [], 1);
    PIDgains = Actions(I).';
    PIDgains(4:end) = -PIDgains(4:end);
end
% PIDgains = [PIDgains(1) 0 2.5 -2 0 -2];



end
```


round
round
round
round
round
round
1


1
Control signal
$\dot{x} = Ax + Bu$
$y = Cx + Du$
QUBE
theta
alpha
X
1
theta (rad)
2
alpha (rad)


1
theta
2
alpha
$\frac{50s}{s+50}$
$\frac{50s}{s+50}$
1
X

LIITE B: MATLAB, Q-OPPIMISALGORITMI


```
%%
rotpen_ABCD_eqns_ip_LQR

% Tilat diskretisöity
S = (1:7).';
initialStates = [8 1 -3 -6];
% Toimintamatriisi A, jossa on PID -säätimen vahvistukset Kp, Ki ja Kd 1:n
% tarkkuudella
Actions = (0:1:50);

% Q -matriisit jokaiselle vahvistukselle erikseen
initialVal = 0;
Qpa = initialVal*ones([length(S), length(Actions)]);
Qia = initialVal*ones([length(S), length(Actions)]);
Qda = initialVal*ones([length(S), length(Actions)]);

Qpt = initialVal*ones([length(S), length(Actions)]);
Qit = initialVal*ones([length(S), length(Actions)]);
Qdt = initialVal*ones([length(S), length(Actions)]);

Q = cat(3, Qpa, Qia, Qda, Qpt, Qit, Qdt);


%%
rng("shuffle");
episodeLength = 10000;
stepTimes = [10];

gamma = 0.9;
alpha = 0.9;


%%
reward = 0;
Rewards = zeros(1, episodeLength);

model = 'QUBE_balance_oma';
load_system(model)
%%
tstart = tic;

for episode = (1:episodeLength)

    stepTime = stepTimes(randi(length(stepTimes)));

    epsilon = max([1 - episode/episodeLength, 0.05]);
    seed = randi(1e6);
    state = initialStates(randi(length(initialStates)));

    % Set the initial condition of the state space model
    initial_cond = [0 deg2rad(state) 0 0];

    simOut = sim(model);
    Q = simOut.Q.Data;
```

```
    simOut.reward(isinf(simOut.reward)) = [];
    Rewards(episode) = sum(simOut.reward);

    fprintf("Did episode %i with stepTime %.2f \n", episode, stepTime)
    drawnow
end

plot(Rewards);
Qpa = Q(:,:,1);
Qia = Q(:,:,2);
Qda = Q(:,:,3);
Qpt = Q(:,:,4);
Qit = Q(:,:,5);
Qdt = Q(:,:,6);
tend = toc(tstart);

disp(tend)
%%

QnoBal = Q;

% Asettaa tasapainotilan ja epästabiiliustilan arvot pois
 QnoBal(4,:,:) = [];
 QnoBal(1,:,:) = [];
 QnoBal(end,:,:) = [];

[~, tempI] = max(sum(QnoBal,1));
% Hakee indeksit (viritysparametrit), joissa saavutetaan suurin palkkio ja
% läskee tästä viritysparametrit.

Kpa = Actions(tempI(1,1,1))
Kia = Actions(tempI(1,1,2))
Kda = Actions(tempI(1,1,3))
Kpt = -Actions(tempI(1,1,4))
Kit = -Actions(tempI(1,1,5))
Kdt = -Actions(tempI(1,1,6))

%% Default gains

Kpa = 30;
Kia = 0;
Kda = 2.5;

Kpt = -2;
Kit = 0;
Kdt = -2;
```