Tampereen yliopisto

Topias Uusitalo

YKSIULOTTEISEN LÄMMÖNSIIRRON I/PI-SÄÄDÖN SIMULOINTI MATLAB-OHJELMISTOLLA

Kandidaatintutkielma
Tekniikan ja luonnotieteiden tiedekunta
Tarkastaja: Terho Jussila
Heinäkuu 2022

TIIVISTELMÄ

Topias Uusitalo: Yksiulotteisen lämmönsiirron I/PI-säädön simulointi Matlab-ohjelmistolla
Kandidaatintutkielma
Tampereen yliopisto
Tekniisten tieteiden kandidaatin tutkinto-ohjelma, Automaatiotekniikka
Heinäkuu 2022

Tässä dokumentissa simuloidaan yksiulotteisen lämmönsiirron I/PI-säätöä Matlab- ja Simulink-ohjelmistoilla. Tangon lämpötila riippuu sekä ajasta että paikasta, jolloin sitä kuvataan osittaisdifferentiaaliyhtälöllä. Osittaisdifferentiaaliyhtälö diskretoidaan paikan suhteen differenssiapproksimaatiomenetelmällä. Menetelmän avulla voidaan tangon lämpönäytteitä mallintaa tilaesityksellä.

Säätimen parametrit lasketaan Matlab-scriptissä prosessin, anturin ja säätimen taajuusvasteiden ja vaihevaratavoitteen perusteella. PI-säätimen parametrit valitaan tasa-arvokäyrästä, joka rajaan tasosta käyvän alueen. Tasa-arvokäyrä piirretään vaihevaran perusteella ja käyrästä valitaan äärellinen määrä virityskandidaatteja. Virityskandidaateille lasketaan vakioasetusarvolle saatavan säätövirheen IAE-hyvyysluku (Integral of Absolute Error). Myös I-säätimen viritysparametrin valintaan käytetään IAE-hyvyyslukua. Kappaleen lämpöjakaumaa havainnollisetetaan piirtämällä sekä värikuva että säädetyn pisteen ja muutaman aiemman pisteen lämpötilat ajan funktiona. Värikuvan piirrossa sekä ajan että paikan koordinaatit diskretoidaan differenssiapproksimaatiomenetelmällä.

Simulointituloksia tarkistetaan muodostamalla prosessille rationaalinen approksimaatiosiirtofunktio ja laskemalla säätövirheen näytteille approksimaatiot Fourier'in käänteismuunnosta hyödyntäen. Käänteismuunnoksen laskemista varten muodostetaan systeemin säätövirheen Laplace-muunnos. Approksimaatiosiirtofunktio muodostetaan Matlabin funktiolla `invfreqs`.

SISÄLLYSLUETTELO

.

1. JOHDANTO

Lämmönsiirron mekanismien ymmärtäminen on ensiarvoisen tärkeää monessa eri tekniikan sovelluksessa. Lämpö voi siirtyä konvektiivisesti, johtumalla tai säteilemällä kahden eri lämpötilan välillä. Lämpötila ja sen mahdollinen säätötarve on otettava huomioon esimerkiksi turvallisuuden säilyttämiseksi, mukavuuden lisäämiseksi tai prosessin toimivuuden takaamiseksi.

Tässä dokumentissa esitellään Matlab-ohjelma, jolla simuloidaan yksiulotteisen lämmönjohtumisen PI/I-säätö. Lämmön siirtymistä tangossa voi tarkastella yksiulotteisesti, jolloin tangon pituus on huomattavasti suurempi kuin tangon halkaisija. Lämmön siirtyminen tangossa riippuu sekä ajasta että paikasta, jolloin lämmön siirtymisen mallina on sopivaa käyttää osittaisdifferentiaaliyhtälöä. Lämmönjohtuminen riippuu paikan koordinaatin ja ajan lisäksi myös materiaalin termisistä ominaisuuksista, joita ovat ominaislämpö, lämmönsiirtokerroin, tiheys ja näistä muodostuva terminen diffusiviteetti [1, s. 40].

PI-säätimen viritys valitaan tuottaman säätöpiirille haluttu vaihevara ja mahdollisimman pieni virhe asetusarvon ja systeemin ulostulon välille. Myös I-säätimen parametri määritetään vaihevaravaatimuksen pohjalta. Osittaisdifferentiaaliyhtälön ratkaisua approksimoidaan käyttäen differenssiapproksimaatiomenetelmää.

Tässä dokumentissa esitellään ensin säätöpiirin ja säätöparametrien muodostamiseen käytettävät teoriat ja yhtälöt, jonka jälkeen esitellään Matlab-skripti. Lopuksi tutkitaan simuloinnin onnistumista muutamalla eri validointitekniikalla.

2. SÄÄTÖPIIRIN MATEMAATTISIA MALLEJA

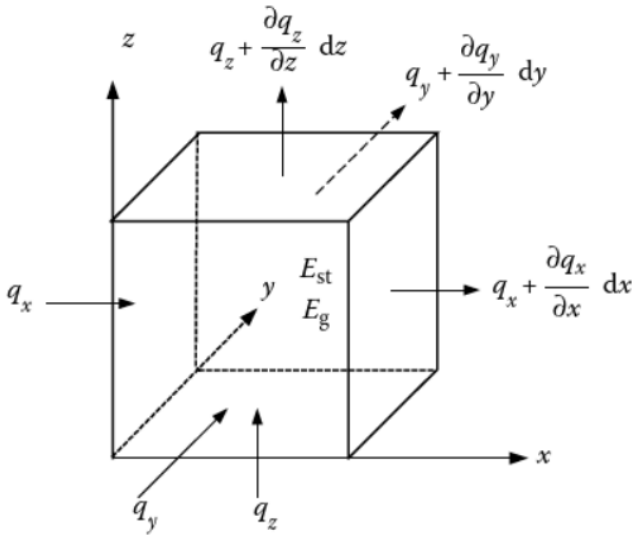
Alla esitellään matemaattinen mallinnus, johon Matlab-ohjelmat perustuvat. Osa kaavojen johdoista esitetään liitteissä.

2.1 Lämmönsiirto

Kolmiulotteisessa lämmön johtumisessa voidaan kiinteään materiaaliin varastoitunutta energiaa mallintaa kuvan 2.1 mukaan. Kuvan energiataseen yhtälö on

$$E_v = E_s - E_u + E_g, \tag{2.1}$$

jossa E_v kuvaa kontrollialueeseen varastoitunutta energiaa, $E_s - E_u$ sisään tulevan ja ulos lähtevän lämpöenergian erotusta ja E_g lämmön generoitumista.



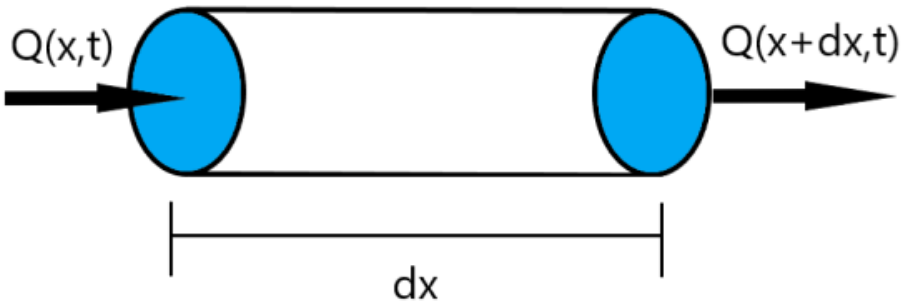
Kuva 2.1. Kolmiulotteinen lämpöenergian varastointi [2, s. 7]

Kaavan (2.1) ja Fourierin johtumislain avulla voidaan kolmiulotteisen lämmön siirtymisen yhtälö johtaa muotoon

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{q}}{k} = \frac{1}{\kappa}\frac{\partial T}{\partial t}\,, \tag{2.2}$$

jossa x, y, z ovat johtumissuunnat, T lämpötila, $\dot{q}$ lämmönlähteen tuottama lämpöenergia, k lämmönjohtumiskerroin, κ terminen diffusiviteetti ja t aika. [2, s. 6–8]

Työssä tarkastellaan yksiulotteista lämmön johtumista sylinterin muotoisessa tangossa olettaen, että lämpöä siirtyy vain kappaleen pitkän sivun suuntaisesti. Tällöin lämmönsiirtoa voidaan mallintaa kuvalla 2.2.



Kuva 2.2. Lämmön johtuminen pitkässä sylinterissä [3, s. 729]

Lämmön johtumisen yhtälö yksiulotteisessa tapauksessa voidaan esittää muodossa

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}, \tag{2.3}$$

jossa

$$\kappa = \frac{k}{\rho c}. \tag{2.4}$$

Yhtälössä (2.3) T on lämpötila, t aika, x paikan koordinaatti, κ materiaalin terminen diffusiviteetti, k lämmönjohtavuuskerroin, c ominaislämpö ja ρ materiaalin tiheys. Yksiulotteista lämmön johtumista voidaan myös mallintaa olettaen, että muut kuin x-akselin suuntaiset sivut ovat äärettömän pitkät, jolloin lämmön johtuminen on mahdollista vain x-akselin suunnassa. [4, s. 45–46]

2.1.1 Siirtofunktio

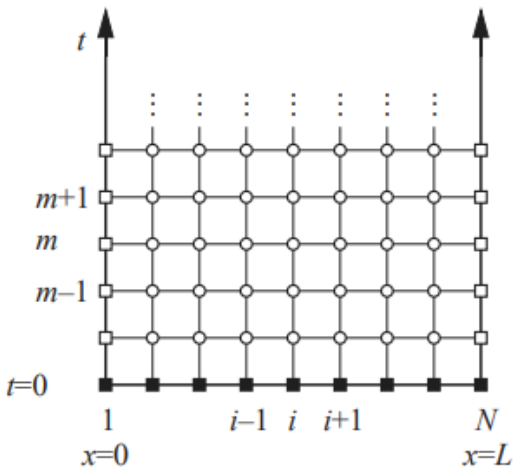
Lämmönsiirron siirtofunktion muodostus on esitetty liitteessä A, jossa on kuvattu kaksi erilaista tapaa siirtofunktion muodostamiselle. Siirtofunktioksi saadaan

$$G_p(\mathrm{s}) = e^{-x\sqrt{bs}}, \tag{2.5}$$

jossa x on mittauspisteen etäisyys lämmityspisteestä, b on termisen diffusiviteetin käänteisarvo ja s on Laplace-muuttuja. Siirtofunktiota käytetään prosessin taajusvasteen muodostamiseen.

2.1.2 Differenssiapproksimaatiomenetelmä

Lämpötilan riippuvuutta ajasta ja paikasta tutkitaan joskus finite difference -metodin avulla. Metodissa muodostetaan differentiaaliyhtälön approksimoitu ratkaisu käyttämällä äärellisen määrän pisteitä sisältävää ruudukkoa. Kuvassa 2.3 on esimerkkikuva ruudukosta. Kuvassa m kuvaa ajan arvoja ja i paikan arvoja.



Kuva 2.3. Differenssiapproksimaatio [6, s. 4]

Approksimaatiossa sijoitetaan derivaattojen paikalle sopivat erotusosamäärät. Niiden käyttö muuttaa differentiaaliyhtälön äärelliseksi määräksi lineaarisia algebrallisia yhtälöitä. [5, s. 43] Lämmönsiirron osittaisdifferentiaaliyhtälöä approksimoidaan diskre-

toimalla sekä ajan että paikan suhteen muodostettuja derivaattoja. Lämmönsiirron yhtälöä diskretoitaessa osittaisdifferentiaaliyhtälö korvataan diskreetillä approksimaatiolla [6, s. 2].

Differenssiapproksimaatiomenetelmää on mahdollista käyttää ennustamaan seuraavaan hetken lämpötilan arvo (eng. forward difference) aiempien ajanhetkien arvojen avulla. Vaihtoehtona on käyttää aikaisempia arvoja, joiden avulla lasketaan nykyhetken arvo (eng. backward difference), tai muodostaa mainittujen approksimaatioiden erotus (eng. central difference). Artikkelissaan Recktenwald [6] käyttää ajan derivaatan approksimointiin forward difference -metodia ja paikan derivaatan approksimointiin central difference -metodia. Lämmönsiirron approksimaatiolle saadaan yhtälö

$$\frac{T_{m,p+1}-T_{m,p}}{\Delta t} = \kappa \frac{T_{m-1,p}-2T_{m,p}+T_{m+1,p}}{\Delta x^2}, \tag{2.6}$$

jossa m on paikkakoordinaatin indeksi, p on ajan indeksi ja T tarkoittaa lämpötilaa kyseisessä pisteessä. Yhtälön (2.6) paikkakoordinaatin muutos määritellään

$$\Delta x = \frac{L}{N-1}, \tag{2.7}$$

jossa L on kappaleen pituus ja N paikkakoordinaattien lukumäärä. Sama pätee ajan muutokselle

$$\Delta t = \frac{t_f}{M-1}, \tag{2.8}$$

jossa t_f on lämmitysaika ja M kuinka moneen osaan aika jaetaan. [5] Paikkainkrementin pienellä arvolla myös aikainkrementin arvon on oltava pieni, jotta vältytään epästabiiliudelta [7, s. 315]. Kun yhtälöstä (2.6) ratkaistaan aikaindeksin $p + 1$ lämpötila, yhtälö saa muodon

$$T_{m,p+1} = \kappa \frac{\Delta t}{\Delta x^2}\left(T_{m-1,p} - 2T_{m,p} + T_{m+1,p}\right) + T_{m,p}. \tag{2.9}$$

Yhtälöstä on mahdollista tehdä selkeämpi järjestelemällä tekijöitä uudelleen:

$$T_{m,p+1} = cT_{m-1,p} + (1-2c)T_{m,p} + cT_{m+1,p}. \tag{2.10}$$

Yhtälöön (2.10) on myös tehty sijoitus [6]

$$c = \kappa \frac{\Delta t}{\Delta x^2}. \tag{2.11}$$

Tätä differenssiapproksimaatiomenetelmän sovellustekniikkaa on käytetty Matlab-ohjelmassa lämpötilajakauman värikuvan piirtämisessä.

Tietyn pisteen lämpötilan säätöä varten käytetään differenssiapproksimaatiomenetelmää vain paikan koordinaateille. Kun paikan koordinaateille käytetään keskeisdifferenssimetodia, yhtälöksi saadaan

$$\frac{DT_m}{dt} = \kappa \frac{T_{m-1,p} - 2T_{m,p} + T_{m+1,p}}{\Delta x^2}. \tag{2.12}$$

Yhtälössä $\frac{DT_m}{dt}$ on tietyn paikkakoordinaatin lämpötilan derivaatta. Tällöin saadaan aikaan tavallinen tiladifferentiaaliyhtälö

$$\frac{dT}{dt} = AT(t) + Bu(t) \tag{2.13}$$

jossa $T(t)$ on tilavektori, joka sisältää lämpötilat ensimmäisestä lämmityspisteen jälkeisestä pisteestä päätepistettä edeltävään pisteeseen. Input-vektorin $u(t)$ avulla kuvataan lämmityksen vaikutus kappaleen lämmitettävään pisteeseen. Arvo T_N on lämpötila kappaleen päädyssä. Se olkoon nolla astetta. Sijoitetaan yhtälöön (2.12)

$$\frac{\kappa}{\Delta x^2} = a. \tag{2.14}$$

Viidelle diskretointipisteelle tiladifferentiaaliyhtälöksi saadaan

$$\dot{T} = \begin{bmatrix} -2a & a & 0 & 0 & 0 \\ a & -2a & a & 0 & 0 \\ 0 & a & -2a & a & 0 \\ 0 & 0 & a & -2a & 2 \\ 0 & 0 & 0 & a & -2a \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} + \begin{bmatrix} a \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) \; . \tag{2.15}$$

Ulostulo voidaan kirjoittaa muodossa

$$y(t) = CT(t), \tag{2.16}$$

jossa C on matriisi. Matriisin C sarakkeiden määrä riippuu diskretointipisteiden N ja M lukumääristä. C-matriisi määrittää ne tilat, joita simuloinnissa tarkkaillaan. [8, 6-3] Tässä työssä ulostulovektori y sisältää tarkkailuun valittujen tilojen lämpötilat. Jos tiloja on 5 kappaletta ja niistä kahta ensimmäistä tarkkaillaan, on matriisi

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}. \quad (2.17)$$

Tarkasteltavista tiloista vain yhtä säädetään, joten tilat eritellään Simulink-lohkokaaviossa Demux-lohkolla.

2.2 Säädin

2.2.1 PI-Säädin

Säätimeksi voidaan valita PI- tai I-säädin. Tässä työssä käytetyn PI-säätimen siirtofunktio on

$$G_s = K_p(1 + \frac{1}{T_i s}). \quad (2.18)$$

Siirtofunktion voidaan esittään

$$G_s = K_i(\frac{1+sT_i}{s}), \quad (2.19)$$

jossa

$$K_i = \frac{K_p}{T_i}. \quad (2.20)$$

Siirtofunktiossa K_p on vahvistus, T_i integrointiaika ja s Laplace-muuttuja. PI-säätimelle piirretään (K_p, K_i) -tason käyrä, jonka alta on valittavissa vaihevaravaatimuksen täyttävät viritykset. Vaihevaran määrityksessä on ensin määritettävä vahvistuksen ylimenokulmataajuus ω_{0dB}. Ylimenokulmataajuus on se kulmataajuus, jolla tarkastellun systeemin amplitudivahvistus on 1. Systeemin vaihevara saadaan määrittämällä kyseisen kulmataajuuden vaihe ja lisäämällä siihen 180 astetta. [8, 10-16] PI-säätimen viritysparametrien K_i ja K_p laskentaa varten on ensin muodostettava taajuusvasteet

kaikille alisysteemeille. Kutsutaan koko avoimen systeemin siirtofunktiota kirjaimella L. Se voidaan muodostaa säätimen, prosessin ja anturin sarjaankytkennästä. Avoimen systeemin siirtofunktio voidaan esittää muodossa

$$L(s) = G_a(s)G_p(s)G_s(s), \tag{2.21}$$

jossa G_a on anturin siirtofunktio, G_p prosessin siirtofunktio ja G_s säätimen siirtofunktio.

Avoimen systeemin taajusvaste luodaan sijoittamalla yhtälöön (2.21) $s = j\omega$. Kulmataajuuden ω amplitudivahvistus on taajuusvasteen itseisarvo ja vaihe on taajuusvasteen argumentti [8, 9-29]. Vahvistuksen ylimenokulmataajuudella ω_{0dB} avoimen systeemin vahvistus on 1. Systeemin vahvistuksen lausekkeesta voidaan ratkaista säätimen vahvistus. Säätimen amplitudivahvistuksen yhtälö on

$$M_{PI} = 1/M_{p,a}, \tag{2.22}$$

jossa M_{PI} on säätimen vahvistus kulmataajuudella ω_{0dB} ja $M_{p,a}$ prosessin ja anturin sarjaankytkennän vahvistus samalla kulmataajuudella. Tässä dokumentissa oletetaan ylimenokulmataajuuden ω_{0dB} olevan suurempi kuin nolla. PI-säätimen vahvistus kulmataajuudella ω_{0dB} on

$$M_{PI} = \left| K_i \frac{1+jT_i\omega_{0dB}}{j\omega_{0dB}} \right|. \tag{2.23}$$

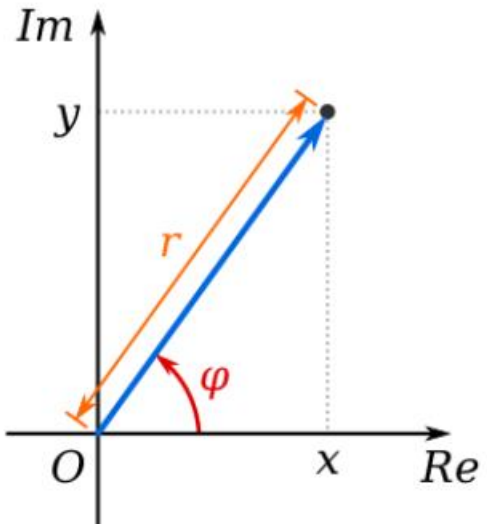
Yhtälöistä (2.22) ja (2.23) voidaan ratkaista parametri K_i kompleksilukujen laskusääntöjä käyttäen:

$$K_i = \frac{\omega_{0dB}}{M_{p,a}\sqrt{1+(T_i\omega_{0dB})^2}}. \tag{2.24}$$

Sijoitetaan K_i:n lauseke yhtälöön $K_p = K_iT_i$, jolloin vahvistuksen K_p yhtälö on

$$K_p = \frac{T_i\omega_{0dB}}{M_{p,a}\sqrt{1+(T_i\omega_{0dB})^2}}. \tag{2.25}$$

Yhtälöissä 2.24 ja 2.25 esiintyvä tekijä $T_i\omega_{0dB}$ on ratkaistavissa vaihevaravaatimuksen avulla. Merkitään haluttua vaihevaraa tunnuksella φ_m. Vaihevaran laskemiseen tarvitaan argumenttia. Kompleksiluvun argumentti tarkoittaa positiivisen reaaliakselin ja origosta tarkasteltavaan kompleksilukupisteeseen vedetyn viivan välistä kulmaa. Argumentin avulla voidaan laskea vaiheen arvoja taajuusvasteista eri kulmataajuuksilla. Argumentin määritelmä on esitetty kuvassa 2.6.



Kuva 2.6. Kompleksilukujen argumentin määritelmä [9]

Vaihevaran laskemiseen voidaan käyttää yhtälöä [8, 10-15]

$$\varphi_m = 180° + \mathrm{Arg}\big(L(j\omega_{0dB})\big). \tag{2.26}$$

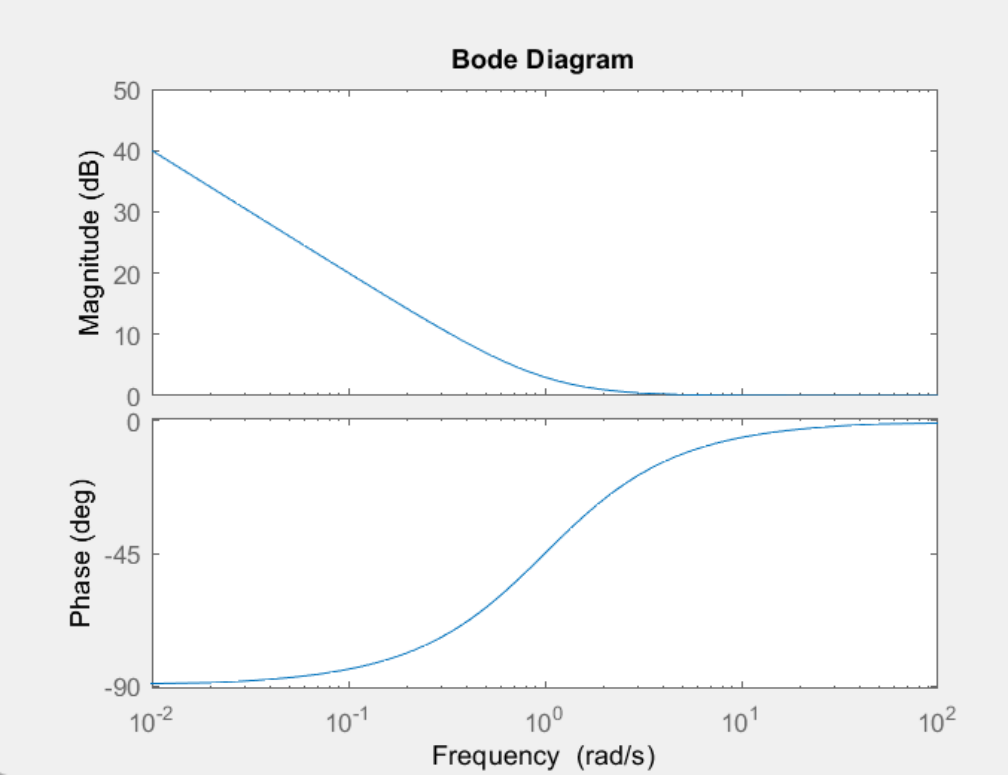
Yhtälössä Arg tarkoittaa standardiargumenttia, joka on määritelty puoliavoimelle välille $(-180°, 180]$ [9]. Taajuusvasteen vaiheesta tämän sovelluksen yhteydessä on kompleksilukujen laskusäännöillä eroteltavissa anturin ja prosessin sarjaankytkennän vaiheensiirto ja säätimen vaihesiirto sääntöä

$$\arg(z_1 z_2) = \arg(z_1) + \arg(z_2) \tag{2.27}$$

käyttäen. Jakolaskuille vastaava sääntö on

$$\arg\left(\frac{z_1}{z_2}\right) = \arg(z_1) - \arg(z_2). \text{ [9]} \tag{2.28}$$

Yhtälöt (2.27) ja (2.28) eivät päde kaikille kompleksiluvuille ja se huomioidaan Matlab-scriptissä poistamalla vaihevektoreista mahdolliset epäjatkuvuudet. Yhtälöissä esiintyvä arg voi yleisesti ottaen poiketa standardiargumentista 360 asteen monikerran verran [9]. Piirtämällä PI-säätimen Bode-kuvaajan, jonka vahvistus $K_p > 0$, säätimen vaiheensiirron voi päätellä olevan välillä $(-90°, 0°)$. Kuvassa 2.5 on PI-säätimen Bode-kuvaaja. Säätimen parametrit ovat 1 ja 1.



Kuva 2.5. PI-säätimen Bode-kuvaaja

Merkitään säätimen vaihetta P_s ja prosessin sekä anturin sarjaankytkennän vaihetta $P_{p,a}$. Yhtälöstä (2.26) voidaan ratkaista säätimen vaihe vahvistuksen ylimenokulmataajuudella

$$P_s(j\omega_{0dB}) = -180° + \varphi_m - P_{p,a}(j\omega_{0dB}). \tag{2.29}$$

Kun tiedetään PI-säätimen vaiheensiirron rajat, voidaan yhtälö (2.29) kirjoittaa epäyhtälönä. Epäyhtälöstä voidaan ratkaista ne prosessin ja anturin sarjaankytkennän vaiheet, joilla on mahdollista saavuttaa toivottu vaihevara. Epäyhtälöksi saadaan

$$\varphi_m - 180° < P_{p,a}(j\omega_{0dB}) < \varphi_m - 90°. \tag{2.30}$$

PI-säätimen vaiheensiirto on

$$\arg\big(G_s(j\omega_{0dB})\big) = \arg(K_i) + \arg\left(\frac{1+j\omega_{0dB}T_i}{j\omega_{0dB}}\right) = \arg(1 + j\omega_{0dB}T_i) - \arg(j\omega_{0dB}). \tag{2.31}$$

Yhtälössä oletuksena on, että K_i on suurempi kuin nolla.

PI-säätimen lopulliseksi vaiheen yhtälöksi kulmataajuudella ω_{0dB} saadaan

$$P_s = \arctan(T_i\omega_{0dB}) - 90°. \tag{2.32}$$

Laskusääntöjä hyödyntämällä ja yhdistämällä yhtälöt (2.26) ja (2.30) voidaan ratkaista puuttuva tekijä $T_i\omega_{0dB}$:

$$T_i\omega_{0dB} = \tan(-90° + \varphi_m - P_{p,a}), \tag{2.33}$$

jossa $P_{p,a}$ on prosessin ja anturin sarjaankytkennän vaiheensiirto kulmataajuudella ω_{0dB}. Näiden tietojen avulla voidaan laskea mahdollisia PI-säätimen viritysparametreja halutulle vaihevaravaatimukselle.

2.2.2 I-säädin

Ei-integroiva prosessi tarvitsee tarkan säädön saavuttamiseksi integroivan säätimen. Integraattorin tarkoitus onkin poistaa prosessin ulostulon ja asetusarvon välinen virhe. Tässä asetusarvo on vakio ja piiri oletetaan stabiiliksi. Tällöin I-säätimen ohjauksen loppuarvon yhtälö on

$$u(\infty) = k_i \int_0^\infty e(t)\, dt, \tag{2.34}$$

jossa k_i on integrointivahvistus ja e on erosuure. [8, 11-4] Integraattorin alkuarvo on tämän sovelluksen yhteydessä 0. I-säätimen siirtofunktioksi saadaan

$$G_c(s) = \frac{K_i}{s}. \tag{2.35}$$

Integraattorin vaiheensiirto on aina $-90°$ [8, 9-30]. Kun säätimen vaiheensiirto tiedetään, voidaan yhtälöstä (2.29) ratkaista prosessin ja anturin sarjaankytkennän vaihe vahvistuksen ylimenokulmataajuudella. Sarjaankytkennän vaiheen kulmataajuudella ω_{0dB} on oltava

$$P_{p,a}(\omega_{0dB}) = \varphi_m - 90°. \tag{2.36}$$

Avoimen systeemin vahvistuksen ylimenokulmataajuutta ω_{0dB} on mahdollista muuttaa vaihtamalla K_i:n arvoja. Ylimenokulmataajuus on valittava siten, että yhtälö (2.36) toteutuu. Vaihevaravaatimus täyttyy myös silloin, kun yhtälön (2.36) vasen puoli on suurempi kuin oikea puoli. Matlabissa valitaan kaikki kulmataajuudet, joilla edellä

mainittu ehto toteutuu. Prosessin ja anturin sarjaankytkennän vaiheensiirron kulmataajuudella ω_{0dB} on siis oltava

$$P_{p,a}(\omega_{0dB}) \geq \varphi_m - 90°. \tag{2.37}$$

Säätimen vahvistus lasketaan samoin kuin PI-säätimelle. Yhtälöstä (2.22) voidaan ratkaista viritysparametri K_i. Viritysparametrin yhtälö on

$$K_i = \frac{\omega_{0dB}}{M_{p,a}}. \tag{2.38}$$

Matlab-ohjelma laskee K_i:n arvot kaikille kulmataajuuksille, joilla yhtälö (2.37) toteutuu.

2.3 Anturi

Matlab-koodissa anturiksi voidaan valita ideaalinen anturi tai ensimmäisen/toisen kertaluvun nollaton ja viiveetön anturi. Ideaalisen anturin siirtofunktio on yksi. Ensimmäisen kertaluvun anturin sisäänmenon u_s ja ulostulon y_s välille voidaan muodostaa yhtälö

$$a_1\dot{y}_s + y_s = u_s, \tag{2.39}$$

jossa a_1 on positiivinen aikavakio. Toisen kertaluvun anturille vastaava yhtälö on

$$a_1\ddot{y}_s + a_2\dot{y}_s + y_s = y_s, \tag{2.40}$$

jossa a_1 ja a_2 ovat positiivisia aikavakioita. Ensimmäisen kertaluvun anturin siirtofunktio on

$$G_a(\mathrm{s}) = \frac{1}{a_1 s+1}. \tag{2.41}$$

Toisen kertaluvun anturin siirtofunktio on

$$G_a(\mathrm{s}) = \frac{1}{a_1 s^2+a_2 s+1}. \tag{2.42}$$

2.4 Fourier-käänteismuunnos

Säätövirhe vakioasetusarvolle I/PI-säädöllä suppenee nollaan, jos säätöpiiri on stabiili. Säätövirheelle johdetaan siirtofunktio, jonka jälkeen sen taajuusvasteelle sovelletaan

Fourier-käänteismuunnosta, joka tuottaa säätövirheen. Sen tulee supeta nollaan. Säätövirheen Laplace-muunnos voidaan kirjoittaa

$$E(s) = R(s) - Y(s), \tag{2.43}$$

jossa $E(s)$ on säätövirheen, $R(s)$ asetusarvon ja $Y(s)$ prosessin ulostulon Laplace-muunnos. Ulostulon Laplace-muunnokselle voidaan kirjoittaa yhtälö

$$Y(s) = G_{tk}(s)R(s), \tag{2.44}$$

jossa $G_{tk}(s)$ on säätöpiirin siirtofunktio. Säätöpiirin siirtofunktio on

$$G_{tk}(s) = \frac{G_p(s)G_s(s)}{1+G_a(s)G_p(s)G_s(s)} \text{ [8, 9-17].} \tag{2.45}$$

Kun yhtälöt (2.43), (2.44) ja (2.45) yhdistetään ja muokataan sopivaan muotoon, tulee säätövirheen siirtofunktiolle lauseke

$$E(s) = (\frac{1-G_p(s)G_s(s)[G_a(s)-1]}{1+G_a(s)G_p(s)G_s(s)})R(s). \tag{2.46}$$

Yhtälö (2.46) voidaan muokata muotoon, jossa nollalla jakamiselta vältytään. Yhtälöt on johdettu liitteessä B. Liittessä johdettujen säätövirheiden kaavoista voidaan muodostaa yleinen yhtälö, jonka avulla Matlab-scriptissä säätövirheen siirtofunktio valitaan riippuen anturin ja säätimen siirtofunktioista. Siirtofunktio on muotoa

$$E_f(s) = \frac{T_{tavoite}(1+G_pKq)}{s+G_aG_pK}. \tag{2.47}$$

Siirtofunktiossa $T_{tavoite}$ on tavoitelämpötila, K säädinoptio ja q anturioptio. PI-säätimellä K on liitteen mukaan K_{PI} ja I-säätimellä K_i. Anturinoptioon viittaava tunnus q on toisen kertaluvun anturia käytettäessä liittessä esitetty q_1 , ensimmäisen kertaluvun anturilla q_2 ja ideaalisella anturilla 1.

Fourier-käänteismuunnoksessa käytetään yhtälöä

$$e(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} E(i\omega)e^{i\omega t}d\omega \text{ [10, s. 109].} \tag{2.48}$$

3. MATLAB-OHJELMA

Alla käydään yksityiskohtaisesti läpi työssä luotua Matlab-skriptiä ja sen käytössä huomioon otettavia asioita. Koodi on kommentoitu englanniksi ja muuttujat on nimetty vastaamaan englanninkielistä sanaa. Pääohjelman koodirivit ovat liitteessä C.

3.1 Aloitus

Säädin valitaan rastittamalla valintalaatikko joko PI- tai I-säätimen kohdalta. Sen jälkeen syötetään mallinnettavan materiaalin termiset ominaisuudet. Termistä diffusiviteettiä merkitään Matlab-skriptissä tunnuksella $kappa$. Mitä suurempi on materiaalin terminen diffusiviteetti, sitä nopeammin se lämpenee ja viilenee [4, s. 46]. Terminen diffusiviteetti on usein suuruusluokaltaan $0.1 - 1000 mm^2/s$, jolloin pienien arvojen myötä yhtälöstä (2.5) saatavat taajuusvasteet ovat myös pieniä [11]. Matlabin tarkkuus heikkenee käsiteltävien arvojen ollessa hyvin pieniä tai suuria. Yhtälön (2.5) taajuusvasteeseen vaikuttaa myös paikkakoordinaatti x ja kulmataajuusvektorin w arvot. Näiden kahden arvon yhteisvaikutus määrittää taajuusvasteiden suuruudet. Pienien taajuusvasteiden kanssa syntyy Matlabissa laskennallisia ongelmia, joten vektorin w kulmataajuuksia on muokattava tarpeen vaatiessa. Pienet kulmataajuudet aiheuttavat epätarkkuuksia, kun simuloinnin onnistumista tarkastellaan.

Seuraavaksi valitaan säädettävän kohdan tavoitelämpötila, jota merkitään koodissa nimellä $Tgoal$. Anturi valitaan syöttämällä anturin siirtofunktion kertoimet $a1$ ja $a2$. Ohjelma päättelee sitten anturin siirtofunktion. Riippuen parametrien arvoista Matlab-ohjelma määrittää suotimen siirtofunktiotyypin. Lopuksi valitaan haluttu vaihevara.

3.2 Taajuusvaste

Taajuusvasteen muodostamiseksi on Matlab-ohjelmassa syötettävä maksimi- ja minimikulmataajuudet $wmin$ ja $wmax$ ja lisättävä niiden välille tasavälein tai logaritmisin välein kulmataajuuksia. Päätöskulmataajuuden $wmax$ mielekäs arvo riippuu termisen diffusiviteetin κ ja paikkakoordinaatin x arvoista. PI-säätimen viritykseen käytetystä (K_p, K_i)-tason kuvaajasta on pääteltävissä, onko $wmax$ valittu oikein. Jos kuvaajassa on vain muutama piste, niin kulmataajuuksien tai niiden lukumäärää on muutettava. Tasaväliset kulmataajuudet toteutetaan `linspace`-komennon avulla ja logaritmiset välit `logspace`-komennolla.

Seuraavaksi muodostetaan anturin taajuusvaste ja siirtofunktio. Matlab-ohjelmassa anturin siirtofunktio on nimetty Gs:ksi ja taajuusvaste Gsf:ksi. Ennen prosessin taajuusvasteen muodostamista on lasketaan parametri b, joka on termisen diffusiviteetin käänteisluku. Prosessin taajuusvastetta merkitään tunnuksella Gpf. Prosessin ja anturin sarjaankytkennän taajuusvaste on niiden taajuusvasteiden tulo. Anturin ja prosessin sarjaankytkennän taajuusvastetta kutsutaan nimellä Gsf_pf.

Anturin ja prosessin sarjaankytkennän amplitudivahvistuksen tunnus on Mag ja se lasketaan `abs`-komennolla. Vaihe lasketaan käyttäen `angle`-komentoa ja vaihetta merkitään nimellä Pha. Vaiheen yksikkö on radiaani. Vaihevasteesta pyritään poistamaan epäjatkuvuuskohdat `unwrap`-komennolla. Se käy läpi vaihevektorin ja löytäessään kaksi peräkkäistä kulmaa, joiden ero on suurempi tai yhtä suuri kuin $\pi\ rad$, se muuttaa eräitä kulmia lisäämällä niihin $\pm 2\pi\ rad$ kunnes ero on pienempi kuin $\pi\ rad$ [12].

Prosessin taajuusvasteesta voidaan muodostaa rationaalinen approksimaatio prosessin siirtofunktiolle. Approksimaation vasteita vertaillaan todellisen prosessin vasteisiin simuloinnin oikeellisuuden tarkistamiseen. Siirtofunktion muodostamiseen käytetään komentoa

```
[num, den] = invfreqs(Gpf, w, n1, n2).
```

`Invfreqs` identifioi taajuusvastedatasta siirtofunktion parametrit [13]. Komennon vasemmalla puolella num (”numerator”) on approksimoidun siirtofunktion osoittajan kerroinvektori ja den (”denominator”) sen nimittäjän kerroinvektori. Ei-negatiiviset kokonaisluvut $n1$ ja $n2$ ovat osoittajapolynomin ja nimittäjäpolynomin asteluvut. Pienien taajuusvasteen arvojen johdosta approksimaation yhteydessä voi Matlabissa ilmaantua varoituksia singulaarisesta tai huonosti skaalatusta matriisista. Varoituksista voi pyrkiä pääsemään eroon muuttamalla joko kulmataajuuksien maksimi- ja minimiarvoja tai niiden lukumäärää. Approksimaatiosiirtofunktio muodostetaan kerroinvektoreista komennolla

```
Gapp = tf(num,den).
```

Approksimaatiota käytetään vertailemaan askelvasteita oikeaan prosessiin.

3.3 State-Space-matriisit

Lämmitettävä kappale jaetaan osiin ja osien lukumäärää merkitään tunnuksella nx. Osavälin pituus dx lasketaa kappaleen pituuden ja osavälien lukumäärän avulla. Simu-

lointi merkitään tunnuksella $tfin$ ja simulointiajan jakovälien lukumäärää tunnuksella nt. Simulointiaika $tfin$ riippuu materiaalin termisestä diffusiviteetistä ja x:n arvosta, jolloin sitä on muutettava, kun tarkasteltava materiaali tai säädettävä kohta materiaalissa muuttuu. Yhden aika-askeleen kestoa merkitään tunnuksella dt.

Matriisi A alustetaan diagonaalimatriisiksi, joka kerrotaan arvolla $-2a$. Matriisin loput arvot asetetaan `for` -loopin avulla. Matriisi B muodostetaan alustamalla se ensin nollamatriisiksi ja sen jälkeen muuntamalla ensimmäinen rivi a:ksi. Matriisissa B on oltava yhtä monta riviä kuin matriisissa A.

Seuraavana valitaan tarkkailtavat tilat. Tiloja kutsutaan tunnuksilla $xnodem$, $xnode0$, $xnode1$ ja $xnode2$. Säädettävän tilan indeksin arvoa vastaa $xnodem$. Säädettävän tilan paikkakoordinaatin arvo metreissä tallennetaan muuttujaan $xcoordinate$. Matriisin C sarakkeiden lukumäärä riippuu tilojen lukumäärästä ja rivien lukumäärä tarkasteltavien tilojen määrästä.

3.4 Säätimen viritys

3.4.1 PI-säätimen viritys

Säätimen parametrien viritystekniikka on esitelty kappaleessa 2.2. Säätimen viritysparametrit K_i ja K_p lasketaan kutsumalla funktiota `GetPiParameters`. Funktio on esitetty kuvassa 3.1.

```
function [Kp, Ki, DM, w0dB] = GetPiParameters(w, M, P, pm)
% This function uses magnitude and phase vectors of the series connection
% between sensor and process to calculate possible Kp, Ki pairs to fulfil
% the chosen phase margin.

% Outputs:
% Kp = proportional gain vector
% Ki = integral gain vector
% DM = delay margin vector
% w0dB = gain crossover frequency vector

% Inputs:
% w = frequency vector
% M = magnitude vector of the series connection
% P = phase vector of the series connection
% pm = phase margin
P = P*180/pi; % Phase of the sensor and process series connection
b = find(P < pm-90 & P > (pm-180)); % Searching for all the possible gain crossover frequencies
w0dB = w(b); % Gain crossover frequencies
P = P(b);
M = M(b);
Ti_W0dB = tan((pi/180)*(-90+pm-P));
d = M.*sqrt(1+Ti_W0dB.^2);
Ki = w0dB./d;
Kp = Ti_W0dB./d;
DM = pm*(pi/180)./w0dB; % seconds
end
```

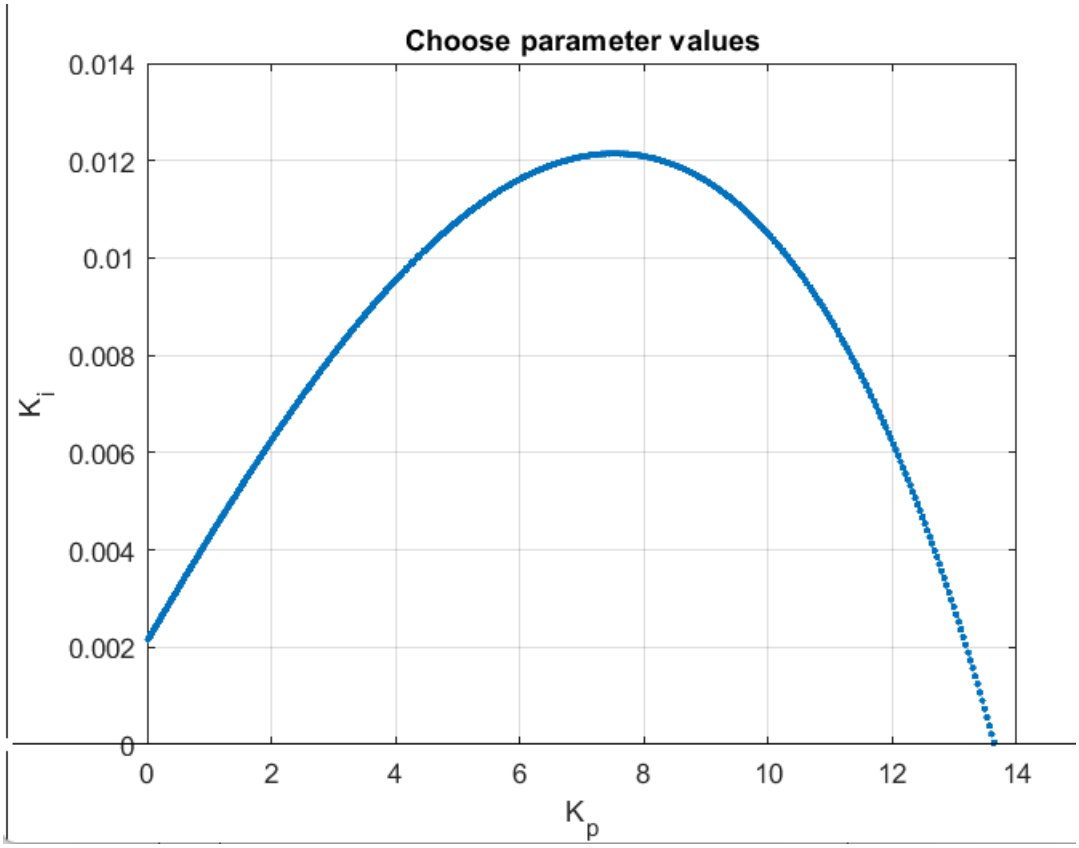
Kuva 3.1. GetPiParameters-funktio

Funktion inputparametrit ovat kulmataajuudet (w), vahvistukset ja vaiheet sisältävät vektorit (Mag ja Pha) sekä haluttu vaihevaraa ($Phase\_margin$). Funktio muuttaa ensin vaiheen asteiksi, jonka jälkeen se etsii vaihevektorista kaikki ne kulmien arvot, jotka täyttävät yhtälön (2.30) mukaisen vaatimuksen. Nämä vektorin indeksit tallennetaan muuttujaan b. Muuttujan b avulla poistetaan ne kulmat ja vahvistukset, jotka eivät täytä haluttua vaihevaravaatimusta. Muokatun vaihevektorin avulla lasketaan yhtälössä (2.33) esitetty muuttuja $T_i\omega_{0dB}$. Funktiossa on lisäksi käytössä apumuuttuja d, jolla korvataan yhtälöiden (2.24) ja (2.25) jakajat. Yhtälöiden (2.24) ja (2.25) avulla lasketaan mahdolliset viritysparametrit, joiden avulla haluttuun vaihevaraan päästään. Ne säilötään vektoreihin Ki ja Kp. Lopuksi funktio laskee vielä kyseisillä parametreillä saadun viivevaran DM. Viivevara voidaan laskea yhtälöllä [14, s. 163]

$$DM = \frac{\varphi_m}{\omega_{0dB}}. \tag{3.1}$$

Yhtälössä (3.1) vaihevara on muutettava yksikköön rad.

Tulosten perusteella voidaan piirtää (K_p, K_i)-tason kuvaaja, josta voidaan poimia mahdollisia viritysparametrejä. Kuvassa 3.2 on esimerkkikuvaaja eräistä viritysparametrien arvoista.



Kuva 3.2. Esimerkkikuva PI-säätimen viritysparametrien valintaan

Parametrien valinta tehdään hiirellä klikkaamalla kuvaajasta haluttu määrä parametrien arvoja. Matlab-ohjelmassa on toteutettu funktio `Choose`, jossa komennon

```
[v,h] = ginput(1)
```

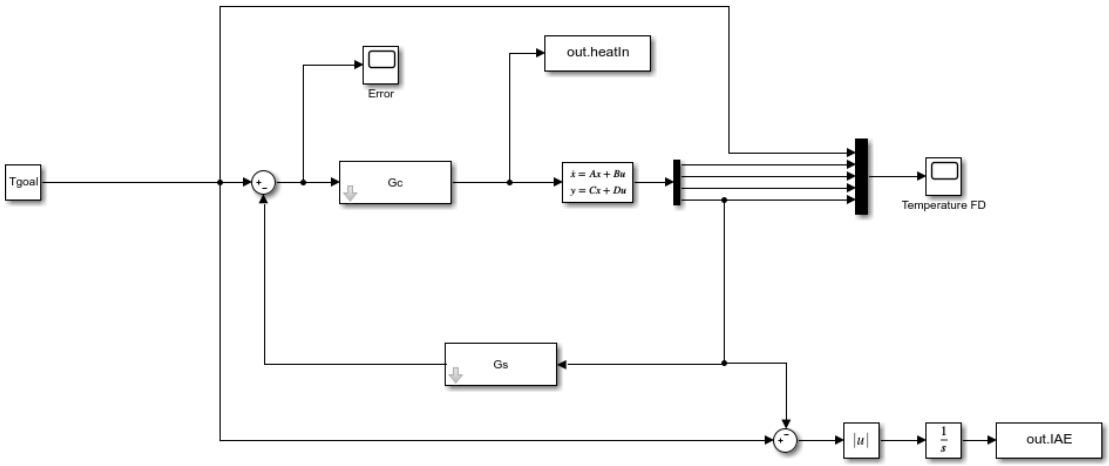
avulla hiirellä valitut parametrit tallentuvat vektoriin. Funktio `Choose` on esitetty kuvassa 3.3.

```
function C = Choose
% This function allows user to choose PI-controller parameters
% from a chart

disp('Choose points by clicking')
disp('Click left of the vertical axis to exit')
C = [];
v = 1;
while v > 0
    [v,h] = ginput(1); % Vertical and horizontal coordinates
    if v >= 0
        plot(v,h,'x')
        A = [v,h];
        C = [C;A];
    end
end
```

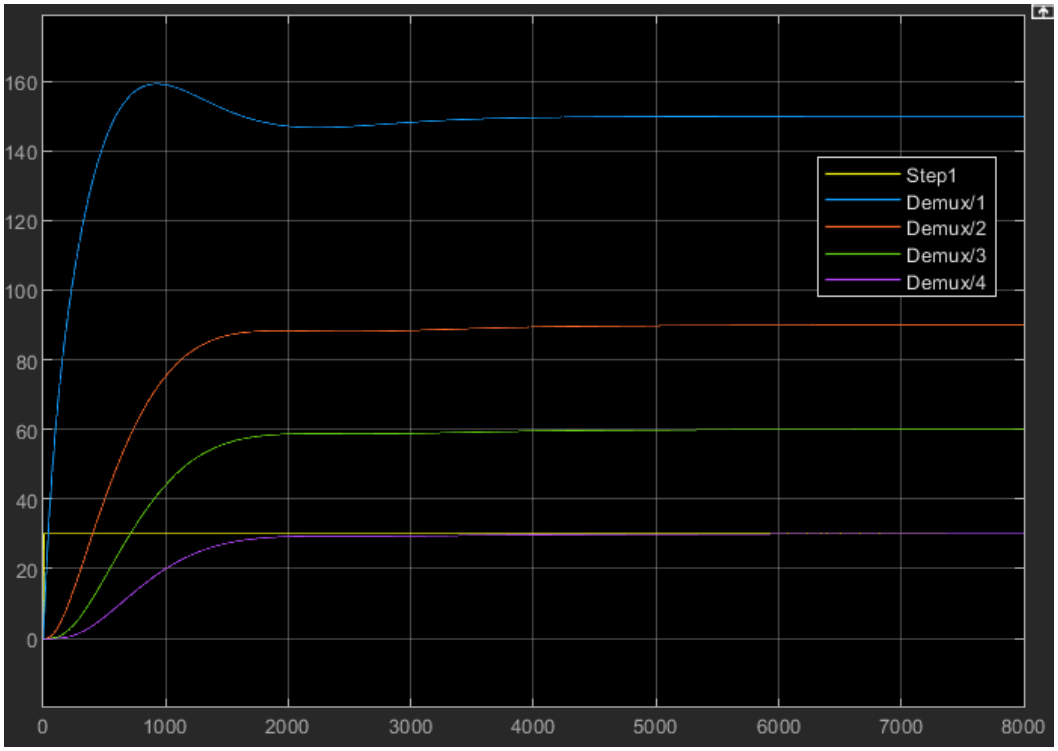
Kuva 3.3. Choose-funktio

Käyrältä valitut parametrit toteuttavat vaihevaravaatimuksen ja käyrän alapuolelta valitut parametrit tuottavat suurempia vaihevaroja. Parametrien valitsemisen voi lopettaa painamalla Ki-akselin vasemmalta puolelta. Tämän jälkeen jokainen valittu parametripari simuloidaan Simulinkin avulla. Simulink-malli simuloi parametrit yksi kerrallaan ja laskee jokaiselle simuloinnille IAE-arvon (”Integral of Absolut Error”) ja valitsee automaattisesti parametrit, joilla pienin IAE toteutuu. IAE lasketaan vähentämällä asetusarvosta prosessin ulostulo, ottamalla itseisarvot erotuksesta ja integroimalla erotus. Kuvassa 3.4 on esitelty Simulink-malli.



Kuva 3.4. Simulink-lohkokaavio prosessin säätöön

IAE arvot tallennetaan Matlabin työtilaan `To Workspace`-lohkolla muuttujaan IAE, josta pienin IAE:n arvo määritetään komennolla `min`. Säätimen siirtofunktion parametrit valitaan pienimmän IAE arvon perusteella. `Temperature FD` -lohkoon piirtyy tarkasteltavien tilojen askelvasteet. Simulink-mallissa `State-Space`-lohkon alkuarvo on $Tinitial$. Kuvassa 3.5 on esimerkkiaskelvaste PI-säätimellä toteutetusta säädöstä.



Kuva 3.5. Säätöpiirin askelvastekuvaaja, $K_i = 0.0066, K_p = 5.4254,$ $\kappa = 13.649 \frac{mm^2}{s}, T_{tavoite} = 30°C, T_{alku} = 0°C$

Kuvaajassa violetti käyrä vastaa säädetyn paikan koordinaatin vastetta ja keltainen asetusarvoa. Kolme muuta käyrää kuvaavat muiden valittujen pisteiden vasteita.

3.4.2 I-säätimen viritys

I-säätimen mahdolliset viritysparametrin K_i arvot lasketaan funktiossa `GetKi`, joka on esitetty kuvassa 3.6.

```
function [Ki, DM, w0dB] = GetKi(w, M, P, pm)
% This function counts values for integral gain Ki that fulfil the chosen phase margin.
% The function uses phase and magnitude vectors of the sensor and process series connection
% and chosen phase margin.

%Outputs:
% Ki = vector of integral gains
% DM = vector of delay margins
% w0dB = vector of possible gain crossover frequencies

%Inputs:
% w = vector frequencies
% M = vector of magnitudes of the sensor and process series connection
% P = vector of phases of the series connections
% pm = phase margin


P = P*180/pi; % rad to deg
b = find(P >= pm-90); % Searching for the possible gain crossover frequencies
w0dB = w(b);
M = M(b);
Ki = w0dB./M;
DM = pm*(pi/180)./w0dB; % seconds
end
```

Kuva 3.6. GetKi-funktio

Funktio ottaa input-arvoinaan kulmataajuusvektorin, prosessin ja anturin sarjaankytkennän taajuusvasteesta muodostetut vaihe- ja vahvistusvektorit sekä vaihevaran. Funktio etsii vaihevektorista kaikki ne vaiheet, joiden kanssa yhtälö (2.37) toteutuu. Kaikki yhtälön toteuttavien vaiheiden indeksit tallennetaan vektoriin, jonka avulla taajuusvasteen vahvistukset sisältävästä vektorista poistetaan kaikki ne komponentit, jotka eivät yhtälöä täytä. Muokatun vahvistusvektorin avulla lasketaan yhtälöstä (2.38) kaikki mahdolliset Ki parametrin arvot. Lopuksi lasketaan vielä viivevara samoin kuten tehtiin PI-säätimelle.

Funktio `GetKi` palauttaa arvoinaan kaikki mahdolliset parametrin Ki arvot, viivevaran Dm ja vahvistuksen ylimenokulmataajuuden $w0dB$. Seuraavaksi ohjelma käy läpi kaikki Ki:n arvot, jonka jälkeen vaihtoehdoista valikoituu automaattisesti se arvo, jonka kanssa IAE-arvo on pienin. Ohjelma tarkastaa molempien säätimien kohdalla, onko toivottu vaihevara mahdollista saavuttaa. Jos haluttu vaihevara on liian suuri, ei ohjelma etene tästä, vaan ilmoittaa, ettei säätimelle löydy viritysparametrejä.

3.5 Lämmönjakautumisen värikuva

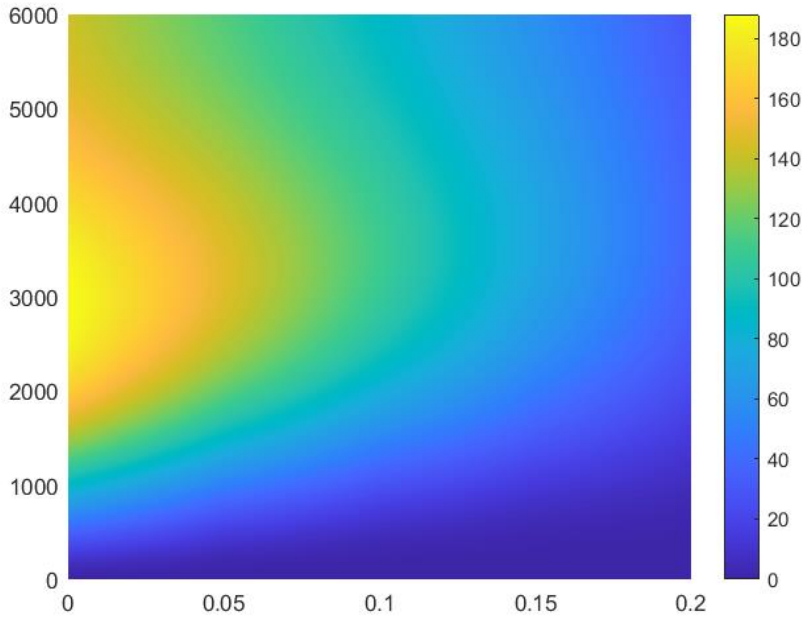
Lämmönjakautumisen piirtoon käytetään yhtälöä (2.10). Kuvaa 2.3 mallinnetaan Matlabissa matriisilla, jossa rivien määrä on nt sarakkeiden määrä on $nx + 2$, sillä sisääntuleva lämpötila sekä ulkolämpötila on otettava huomioon. Matlab-ohjelmassa

lämpötilamatriisi on nimetty $Temp$ ja se alustetaan sopivan kokoiseksi nollamatriisiksi. Matriisin ensimmäinen rivi alustetaan alkulämpötilaan $Tinitial$ ja viimeinen sarake nollaksi ulkolämpötilan mukaan. Kutakin aika-askelta vastaava ohjauksen arvo saadaan kuvassa 3.2 näkyvän `out.heatIn`-lohkon avulla. Lohkon formaatti on Timeseries, jolloin työtilaan tallentuu vektori, jossa on tietyin aikavälein saadut ohjauksen arvot. Näytteenottoväliksi on asetettu dt, jotta vektorin koko olisi oikea. Vektori tallennetaan muuttujaan $Heatin$ ja se asetetaan $Temp$-matriisin ensimmäiseksi sarakkeeksi. Seuraavaksi $Temp$-matriisin arvot päivitetään yhtälön 2.10 mukaan komennoilla `while` ja `for`.

Ennen kuvan piirtoa poistetaan vielä ensimmäinen ja viimeinen sarake, jotta kuva piirtyy oikein. Muuttujia $xbar$ ja t käytetään skaalaamaan kuvaajan akselit oikean kokoisiksi. Itse kuvan piirto tapahtuu komennolla

```
heat = pcolor(xbar,t,Temp).
```

Komento piirtää matriisista värillisen kuvan. Kuvan värit tasoitetaan komennolla `shading interp` ja lopuksi komennolla colorbar saadaan väreille lukuarvot näkyviin. [15] Kuvassa 3.7 on esimerkkikuva, jossa simulointiaika on 6000 s, kappaleen pituus 20 cm ja tavoitelämpötila 30 astetta.

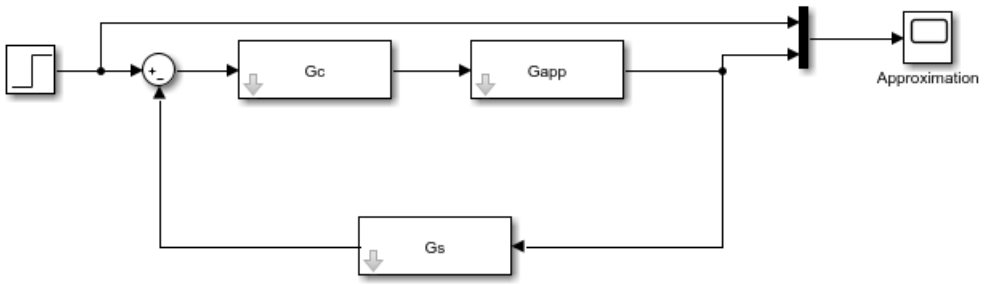


Kuva 3.7. Esimerkkikuva lämmönjakautumisesta

Kuvassa 3.7 pystyakselilla on ajan arvot ja vaaka-akselilla paikan arvot.

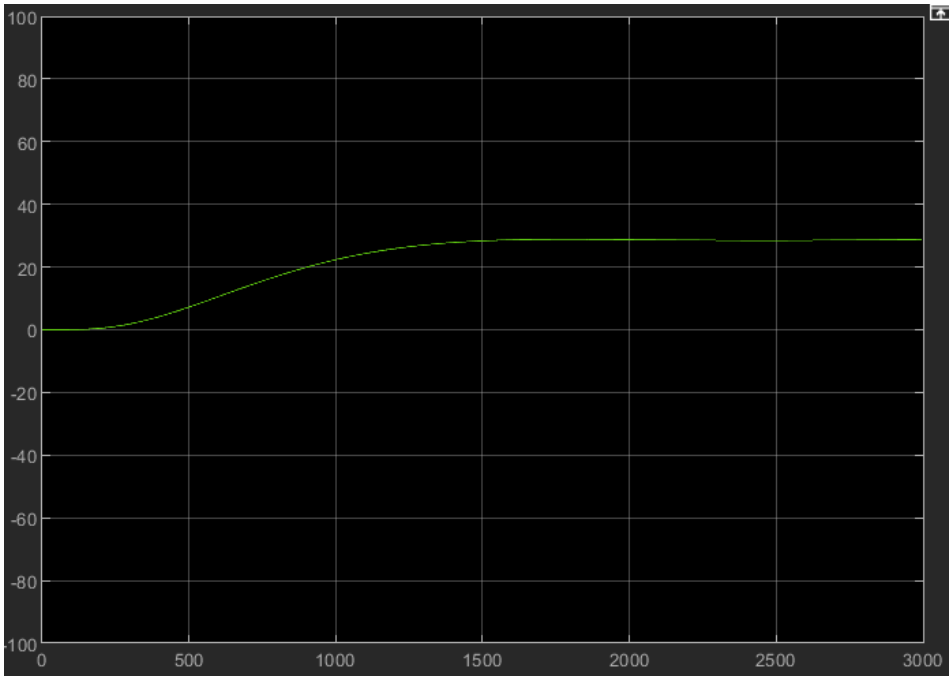
3.6 Validointi

Matlab-ohjelmassa käytetään kahta eri keinoa säädön toimivuuden tarkastamiseen. Ensimmäinen tapa validointiin on verrata approksimaatiosiirtofunktion vasteita oikean prosessin vasteisiin. Prosessin taajuusvasteesta muodostetun siirtofunktion approksimaation $Gapp$ vasteet muodostetaan Matlabissä omalla Simulink-lohkokaaviolla. Lohkokaavio on kuvassa 3.8.

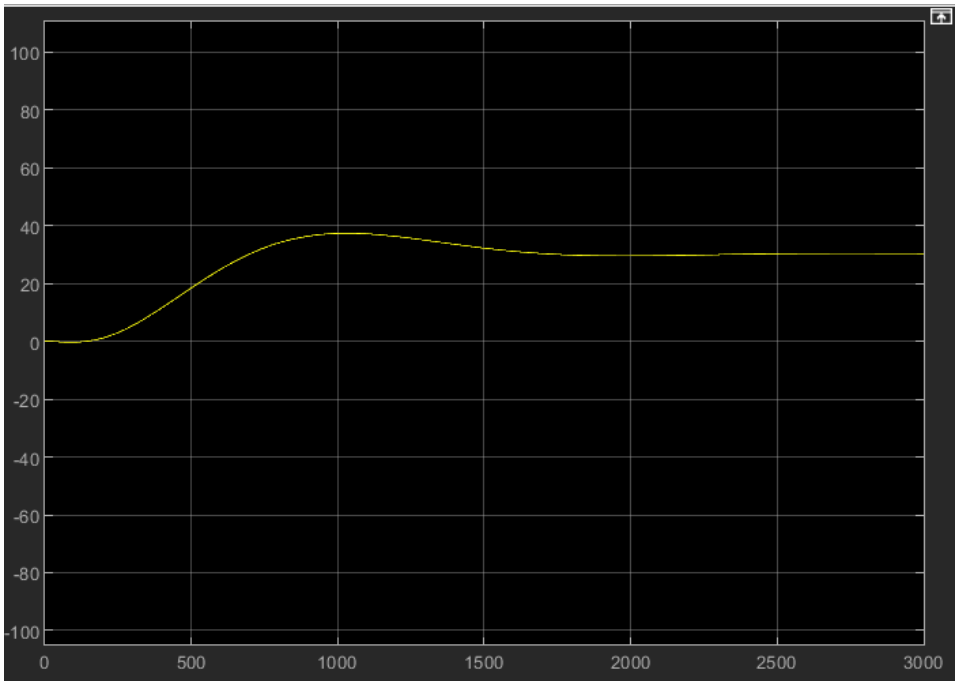


Kuva 3.8. Simulink-lohkokaavio approksimaatiosiirtofunktiolle

Approksimaation kanssa on syytä olla tarkkana, sillä pienien taajuusvasteen arvojen kanssa approksimaatiosta saattaa tulla epätarkka. Kuvassa 3.9 on esitetty todellisen prosessin vaste ja kuvassa 3.10 samoilla säätöparametreilla saatu approksimaation vaste.



Kuva 3.9. Säätöpiirin askelvaste, lämpötila $30°C$



Kuva 3.10. Approksimoidun säätöpiirin ulotustulo, lämpötila $30°C$

Kuvissa pystyakselilla on säädettävän kohdan lämpötila ja vaaka-akselilla kulunut aika. Kuvista on nähtävissä paljon yhtäläisyyksiä dynamiikassa. Asettumisaika on approksimaatiossa hieman pidempi kuin alkuperäisen säätöpiirin ja sen askelvasteessa on pientä ylitystä. Molempien vasteiden alut ovat hyvin samankaltaiset. Ottaen huomioon todellisen prosessin mallinnuksessa käytetty osittaisderivaatan approksimaatio, voidaan päätellä ainakin tämän validointitekniikan näkökulmasta säätöparametrien virityksen onnistuneen.

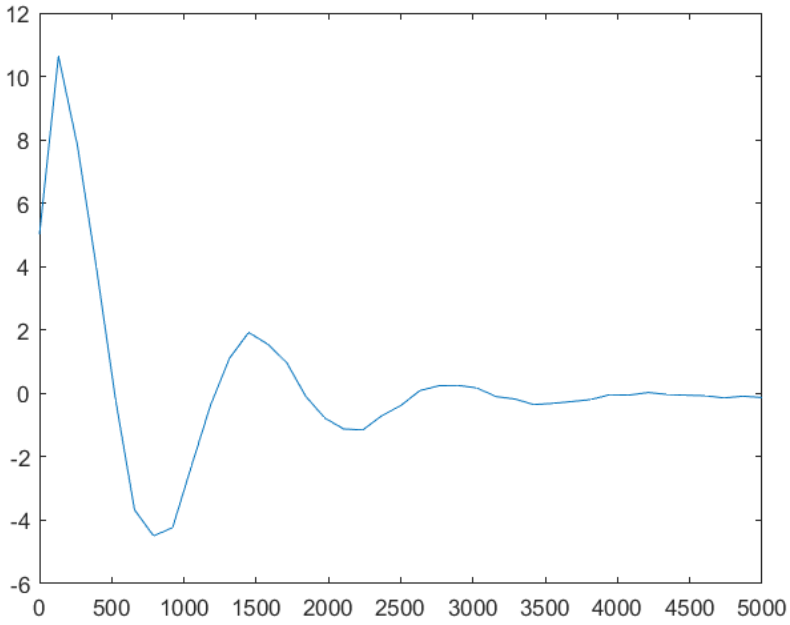
Säätövirheen käyttäytymistä voidaan tutkia Fourier-käänteismuunnoksella. Onnistuneen säädön säätövirheen suppenee nollaan. Validointia varten Matlabissa alustetaan muuttuja wt, jota käytetään integroinnin ylä- ja alarajana äärettömyyden sijasta. Numeerisen integroinnin rajoina ei voi olla ääretön. Ylä- ja alarajan arvot on syytä asettaa suuriksi, jotta numeerinen integrointi vastaa mahdollisimman hyvin yhtälöä (2.48). Muuttujasta wt poistetaan mahdollinen nolla, jolla vältetään mahdollinen nollalla jako. Mitä enemmän muuttujaan wt tallennetaan arvoja, sen tarkempi on Fourier-käänteismunnoksen tulos. Samalla Matlabilta vaaditaan paljon enemmän laskentatehoa.

Asetusarvolle c käytetään vakioarvon Laplace-muunnosta, joka on [3, s. 363]

$$R(s) = \frac{c}{s}. \tag{3.2}$$

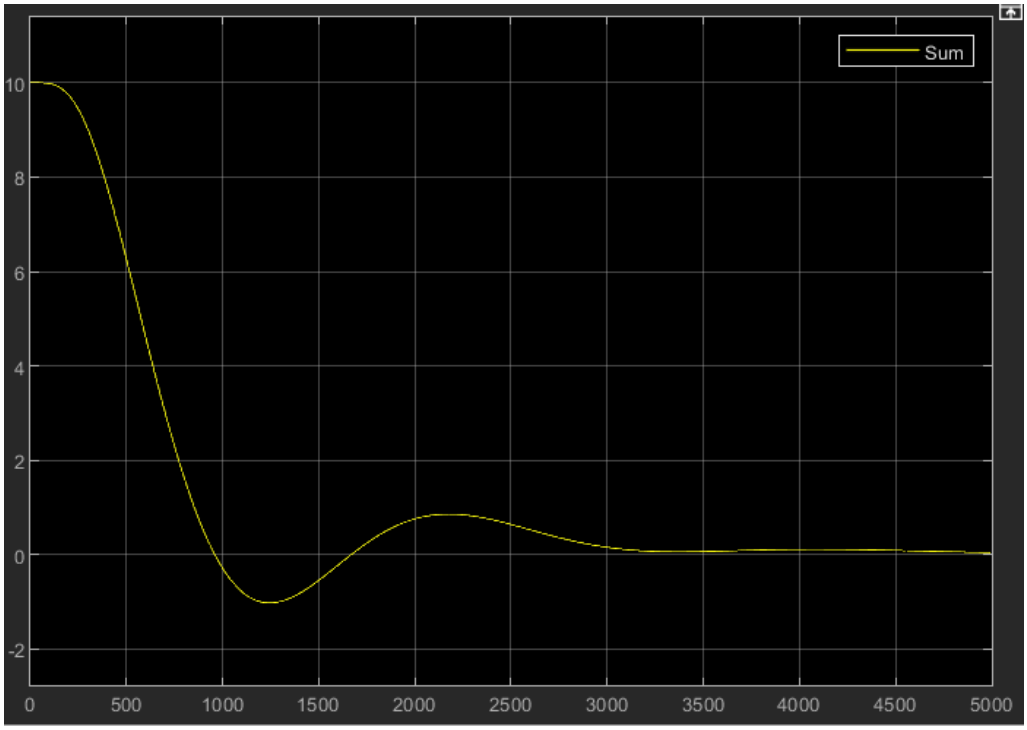
Asetusarvon, prosessin, säätimen ja anturin siirtofunktiot muodostetaan samoin kuin kappaleessa 3.2. Seuraavaksi alustetaan symbolinen muuttuja tt, jota käytetään numeerisessa integroinnissa. Erosuureen taajuusvaste muodostetaan yhtälön (2.46) mukaan.

Siirtofunktioiden monimutkaisuuden takia symbolinen integrointi ei onnistu. Integrointiin käytetään Matlabin komentoa `trapz`. Komennossa ensimmäinen input on ne arvot sisältävä vektori, joiden kohdalta käsiteltävää funktiota integroidaan. Toinen input on integroitava funktio. [16] Lopuksi alustetaan vektori $t2$, jonka avulla piirretään kuvaaja erosuureen käyttäytymisestä. Symbolinen muuttuja tt korvataan vektorilla $t2$ ja erosuureen Fourier-käänteismuunnoksen kuvaaja piirretään komennolla `plot`. Kuvassa 3.11 on Fourier-käänteismuunnoksen tulos säätövirheelle.



Kuva 3.11. Säätövirhe Fourier-käänteismuunnoksesta

Differenssiapproksimaatiomenetelmän avulla muodostetussa Simulink-lohkokaaviossa on $Error$-niminen `Scope`-lohko. Siihen piirtyy differenssiapproksimaatiomenetelmällä muodostettu säätövirhe. Simulink-lohkokaaviosta saatu säätövirhe on esitetty kuvassa 3.12.



Kuva 3.12. Säätövirhe Simulink-lohkokaaviosta

Kuvat 3.11 ja 3.12 ovat saman simulointikerran tuloksia. Tuloksien dynamiikassa on paljon yhtäläisyyksiä. Säätövirheen nollaan suppeneminen tapahtuu molemmissa kuvissa 3000 sekunnin jälkeen. Myös käyrämuodot muistuttavat toisiaan. Fourier-käänteismuunnoksen avulla muodostettu säätövirheen kuvaaja muodostetaan numeerisella integroinnilla, joten integrointipisteiden määrä vaikuttaa lopputulokseen selkeästi. Käyrien yhtäläisyydet antavat kuitenkin viitteittä siitä, että simulointi on onnistunut.

4. YHTEENVETO

Työn tavoitteena oli simuloida yksiulotteisen lämmönsiirron I/PI-säätöä. Simuloinnin tarkkuutta heikentää osittaisdifferentiaaliyhtälölle tehty approksimaatio. Approksimoidun yhtälön askelvasteet vaikuttavat onnistuneen suurimmalta osin hyvin. Matlab-ohjelmalla on mahdollista mallintaa eri materiaalien lämmönjakautumista värillisen jakauma kuvaajan sekä askelvasteiden avulla. Säädinten parametrien valinnalle johdettiin yhtälöt, joiden pohjalta Matlab-scripti laskee mahdollisimman hyvät parametriarvot.

Simuloinnin onnistumisen tarkastamiseen käytettiin kahta eri tekniikkaa. Ensimmäinen tekniikka tuottaa samankaltaista dynamiikkaa kuin prosessikin, joskin pienet taajuusvasteiden arvot saattavat olla ongelma. Fourier-käänteismuunnoksen käyttö säätövirheen mallinnukseen antaa tässä työssä viitteitä siihen, että simulointi on onnistunut, sillä säätövirheelle muodostettu Fourier-käänteismuunnos muistuttaa selvästi differenssiapproksimaation tuottamaa säätövirheen vastetta. Myös askelvasteiden perusteella säätövirhe suppenee ajan myötä nollaan. Matlab-ohjelmaa käytettäessä on syytä ottaa huomioon Matlabin ongelmat hyvin pienten ja suurten lukujen käsittelyssä.

LÄHTEET

[1] D. Annaratone, Engineering heat transfer, Springer Berlin Heidelberg, Berlin, 2010, 329 p.

[2] H. Je-Chin, Analytical heat transfer, Boca Raton, FL: CRC Press, an imprint of Taylor and Francis, 2012, 318 p.

[3] G. James, Advanced modern engineering mathematics, Pearson education limited, Harlow, 2011, 1036 p.

[4] P. Von Böckh, T. Wetzel, Heat transfer basics and practice, Springer Berlin Heidelberg, berlin, 2012, 283 p.

[5] S. Larsson, V. Thomée, Partial differential equations with numerical methods, Springer, Berlin, 2003, 259 p.

[6] G. W. Recktenwald, Finite-difference approximations to the heat equation, Sapienza Universita di Roma, 2004, Saatavissa (viitattu 22.2.2022): http://dma.dima.uniroma1.it/users/lsa_adn/MATERIALE/FDheat.pdf

[7] Y. Pinchover, J. Rubinstein, An introdunction to partial differential equations, Cambridge University Press, Cambridge, 2005, 371 p.

[8] K. J. Åström, R. M. Murray, Feedback systems: an introduction for scientists and engineers, Princeton university press, New Jersey, 2020, 396 p.

[9] Argument (complex analysis), Wikipedia, verkkosivu. Päivitetty 27.11.2021. Saatavissa (viitattu 4.2.2022): https://en.wikipedia.org/wiki/Argument_(complex_analysis)#cite_note-:1-1

[10] W. Wasylkiwskyj, Signals and transforms in linear systems analysis, 1. ed, Springer New York, New York, 2013, 387 p.

[11] Thermal diffusivity, Wikipedia, Päivitetty 25.12.2021. Saatavissa (viitattu 15.4.2022):

https://en.wikipedia.org/wiki/Thermal_diffusivity

[12] Unwrap, Mathworks, verkkosivu. Saatavissa (Viitattu 18.4.2022):

https://se.mathworks.com/help/matlab/ref/unwrap.html

[13] Invfreqs, Mathworks, verkkosivu. Saatavissa (viitattu 18.4.2022):

https://se.mathworks.com/help/signal/ref/invfreqs.html?searchHighlight=invfreqs&s_tid=srchtitle_invfreqs_1

[14] I. D. Díaz-Rodriquez, S. Han, S.P. Bhattackaryya, Analytical design of PID controllers, Springer International Publishing, Vancouver, 2019, 302 p.

[15] Pcolor, Mathworks, verkkosivu. Saatavissa (viitattu 7.5.2022): https://se.mathworks.com/help/matlab/ref/pcolor.html

[16] Trapz, Mathworks, verkkosivu. Saatavissa (viitattu 13.5.2022): https://se.mathworks.com/help/matlab/ref/trapz.html#d123e1529176

[17] Terho Jussila, Private Communication, 27.5.2022

LIITE A LÄMMÖNSIIRRON SIIRTOFUNKTIO

Lämmönsiirron siirtofunktio voidaan johtaa Laplace-muunnoksien (LM) avulla. Lämmönsiirron siirtofunktion johto [17]:

ODY: $$b\frac{\partial}{\partial t}\theta(x,t) = \frac{\partial^2}{\partial x^2}\theta(x,t). \quad (1)$$

Yhtälön (1) LM: $$b[s\Theta(x,s) - \theta(x,0)] = \frac{\partial^2}{\partial x^2}\Theta(x,s) \quad (2)$$

Alkuarvo: $$\theta(x,0) = 0 \quad (3)$$

$$\frac{\partial^2}{\partial x^2}\Theta(x,s) = bs\Theta(x,s) \quad (4)$$

Triviaali ratkaisu: $$\Theta(x,s) = e^{rx}, r = r(s) \quad (5)$$

$$\frac{\partial^2}{\partial x^2}e^{rx} = bse^{rx} \quad (6)$$

$$e^{rx}rr = bse^{rx} \quad (7)$$

$$r^2 = bs \quad (8)$$

$$r = \pm\sqrt{bs} \quad (9)$$

$$\Theta(x,s) = c_1e^{x\sqrt{bs}} + c_2e^{-x\sqrt{bs}} \quad (10)$$

$$\theta(\infty,t) = 0 \rightarrow \Theta(\infty,t) = 0 \rightarrow c_1 = 0 \quad (11)$$

$$\Theta(0,s) = c_2e^{-0\sqrt{bs}} = c_2 \quad (12)$$

$$\theta(0,t) = u(t) \rightarrow \Theta(0,s) = U(s) = c_2 \quad (13)$$

$$\Theta(x,s) = c_2e^{-x\sqrt{bs}} = e^{-x\sqrt{bs}}U(s) \quad (14)$$

$$G(x,s) = e^{-x\sqrt{bs}} \quad (15)$$

LIITE B SÄÄTÖVIRHEEN YHTÄLÖN MUOKKAUS

Tässä liitteessä johdetaan säätövirheen yhtälöt muotoon, jossa nollalla jakamiselta vältytään Laplace-muuttujan ollessa 0. Käytetään asetusarvolle vakioarvoa. Vakioarvon siirtofunktio on

$$R(s) = \frac{T_{tavoite}}{s}. \quad (1)$$

Säätövirheen Laplace-muunnos on

$$E(s) = \left(\frac{1-G_p(s)G_s(s)[G_a(s)-1]}{1+G_a(s)G_p(s)G_s(s)}\right)\frac{T_{tavoite}}{s}. \quad (2)$$

PI-säätimen siirtofunktio on

$$G_s(s) = \left(K_p s + K_i\right)/s. \quad (3)$$

Sijoitetaan yhtälö (3) yhtälöön (2) ja lavennetaan termillä s:

$$E_{PI}(S) = \left(\frac{s-G_p(s)\left(K_p s+K_i\right)[G_a(s)-1]}{s+G_a(s)G_p(s)\left(K_p s+K_i\right)}\right)\frac{T_{tavoite}}{s}. \quad (4)$$

Toisen kertaluvun nollattoman ja viiveettömän anturin siirtofunktio on

$$G_a(s) = \frac{1}{a_1 s^2+a_2 s+1}. \quad (5)$$

Säätövirheen siirtofunktiossa esiintyvä termi voidaan kirjoittaa muodossa

$$G_a(s) - 1 = \frac{1}{a_1 s^2+a_2 s+1} - 1 = \frac{-(a_1 s+a_2)s}{a_1 s^2+a_2 s+1}. \quad (6)$$

Kun yhtälöön (4) sijoitetaan yhtälössä (6) saatu tulos ja Laplace-muuttujat supistetaan, saadaan säätövirheen siirtofunktio muotoon

$$E_1(S) = \frac{T_{tavoite}(1+G_p(K_p s+K_i)q_1}{s+G_a G_p(K_p s+K_i)}, \quad (7)$$

jossa

$$q_1 = \frac{a_1 s+a_2}{a_1 s^2+a_2 s+1} \quad (8)$$

Ensimmäisen kertaluvun nollattoman ja viiveettömän anturin siirtofunktio on

$$G_a(s) = \frac{1}{a_1 s+1}. \quad (9)$$

Yhtälössä (4) esiintyvä termi voidaan kirjoittaa

$$G_a(s) - 1 = \frac{1}{a_1 s+1} - 1 = \frac{-a_1 s}{a_1 s+1}. \quad (10)$$

Kun yhtälöön (4) sijoitetaan yhtälössä (9) muodostettu termi, säätövirheen siirtofunktio on

$$E_2(S) = \frac{T_{tavoite}(1+G_p(K_p s+K_i)q_2}{s+G_a G_p(K_p s+K_i)}, \quad (11)$$

jossa

$$q_2 = \frac{a_1}{a_1 s+1} \quad (12)$$

Ideaalisen anturin siirtofunktio on 1. Tällöin termi $G_a(s) - 1$ supistuu pois automaattisesti. Säätövirheen siirtofunktio on

$$E_3(S) = \frac{T_{tavoite}}{s+G_a G_p(K_p s+K_i)}. \quad (13)$$

Merkitään yhtälöissä (7), (11) ja (12) esiintyvää termiä

$$K_p s + K_i = K_{PI}. \quad (14)$$

I-säätimen siirtofunktio on

$$G_s(s) = \frac{K_i}{s}. \quad (15)$$

Säätövirheen Laplace-muunnos I-säätimellä on

$$E_I(s) = \frac{s-G_p K_i[G_a-1]}{s+G_a G_p K_i} \frac{T_{tavoite}}{s}. \quad (16)$$

Kun yhtälöön (16) sijoitetaan yhtälön (5) mukainen anturin siirtofunktio, on säätövirheen siirtofunktio

$$E_4 = \frac{T_{tavoite}(1+G_p K_i q_1)}{s+G_a G_p K_i}. \quad (17)$$

Ensimmäisen kertaluvun anturilla ja I-säätimellä erosuureen siirtofunktio on

$$E_5 = \frac{T_{tavoite}(1+G_p K_i q_2)}{s+G_a G_p K_i}. \quad (18)$$

Ideaalisella anturilla ja I-säätimellä erosuureen siirtofunktio on

$$E_6 = \frac{T_{tavoite}}{s+G_a G_p K_i}. \quad (19)$$

LIITE C MATLAB PÄÄOHJELMA

```
% This program counts PI/I-controller parameters used in 1-D heat trans-
fer.
% The parameters are calculated using frequency responses (=FR) of the
% subsystems (controller, sensor and process).


clearvars


PI_controller = true;
I_controller = false;


cp = 420; % Specific heat coefficient, J/(kg*K)
rho = 7850; % Density, kg/m^3
k = 45; % Heat conduction coefficient, W/(m*K)
kappa = k/(cp*rho); % thermal diffusivity, m^2/s


x = 0.2; % Spatial coordinate in the transfer function
Tgoal = 10;
Tinitial = 0;


a1 = 1; % Sensor coefficient
a2 = 1; % Sensor coefficient


PhaseMargin = 40;


% frequencies
wmax = 0.01; % rad/s
wmin = 0.0001; % rad/s
n = 1000;
w = linspace(wmin,wmax,n); % frequencies
w = w(:);


s = tf('s');


% Gsf = FR of the sensor, Gs = transfer function of the
% sensor
if a1 == 0 && a2 == 0
    Gsf = 1;
    Gs = 1;
elseif a2 == 0 && a1 ~= 0
    Gsf = 1./(a1*1i*w+1);
```

```
    Gs = 1/(a1*s+1);
elseif a1 ~= 0 && a2 ~= 0
    Gsf = 1./(a1*(1i*w).^2+a2*1i*w+1);
    Gs = 1/(a1*s^2+a2*s+1);
end


b = 1/kappa;


Gpf = exp(-x*sqrt((1i*w)*b)); % FR of the process
Gsf_pf = Gsf.*Gpf; % FR of the process and sensor series connection


Mag = abs(Gsf_pf); % Magnitude of the process and sensor series connec-
tion
Mag = Mag(:);


Pha = angle(Gsf_pf); % Magnitude of the process and sensor series connec-
tion, rad
Pha = unwrap(Pha); % Deleting the points of discontinuities
Pha = Pha(:);


[num, den] = invfreqs(Gpf,w,1,2); % Approximation of the process tf
Gapp = tf(num,den); % Used to validate the controls
```

```


nx = 5; % number of spatial nodes
dx = x/(nx-1); % space between 2 nodes


nt = 6000; % number of time nodes
tfin = 5000; % final simulating time


dt = tfin/(nt-1); % time step


a = kappa/dx^2;


A = -2*a*eye(nx); % Scalar matrix


% Adds "a" values manually in the right places
for m = 1:nx
    if m == 1
        A(m,2) = a;
    elseif m == nx
        A(nx,nx-1) = a;
```

```
    else
        A(m,m-1) = a;
        A(m,m+1) = a;
    end
end


B = zeros(nx,1);
B(1,1) = a;


%xnode is the ordinal number of the node where the sensor is
% attached, xnode has to be smaller or equal to nx
xnodem = 5; % Integer
xcoordinate = dx*xnodem % length from the point 0 to sensor attachment
point, m
xnode0 = 1;
xnode1 = 3;
xnode2 = 4;


C = zeros(4,nx);
C(1,xnode0) = 1;
C(2,xnode1) = 1;
C(3,xnode2) = 1;
C(4,xnodem) = 1;


D = zeros(size(C,1),size(B,2));
```

```
% This section defines the control parameters
% Possible PI-controller parameter pairs can be selected from a graph.
After the
% selection is completed, the script automatically calculates the best
% parameter pair based on integral of absolut error.
open('IaeSimulation.slx')



if PI_controller == 1
    [Kp,Ki,DM,w0dB] = GetPiParameters(w,Mag,Pha,PhaseMargin);
    figure(1)
    plot(Kp,Ki,'.')
    grid
    xlabel('K_p')
    ylabel('K_i')
    title('Choose parameter values')
    hold
    Ch = Choose;
    hold off
    IAEs = ones(size(Ch,1),1);
```

```
    for h = 1:length(IAEs)
        h
        Kp = Ch(h,1);
        Ki = Ch(h,2);
        Gc = Kp+Ki/s;
        OUT = sim ('IaeSimulation.slx');
        IAEs(h) = OUT.IAE(end);
    end

    [IAEvalue, index] = min(IAEs); % Finding the smallest Integral of ab-
solute error
    Kp = Ch(index,1);
    Ki = Ch(index,2);
    Gc = Kp+Ki/s; % Final transfer function of the controller
    w0dB = w0dB(index) % Gain cross over frequency
    delayMargin = DM(index) % Delay margin

else
    [Ki, Dm, w0dB] = GetKi(w, Mag, Pha, PhaseMargin);
    IAEs = ones(length(Ki),1);
    h = 1;
    while h <= length(Ki)
        Gc = Ki(h)/s;
        OUT = sim ('IaeSimulation.slx');
        IAEs(h) = OUT.IAE(end);
        h = h+1;
    end

    [IAEvalue, index] = min(IAEs); % Finding the smallest Integral of ab-
solute error
    IAEvalue
    Ki = Ki(index);
    Gc = Ki/s; % Final transfer function of the controller
    w0dB = w0dB(index) % Gain cross over frequency
    delayMargin = Dm(index) % Delay margin
end



if isempty(IAEvalue)
    disp('No transfer function for this phase margin')
else
    OUT = sim ('IaeSimulation.slx');
end
```

```
% This section draws a coloured graph of the temperature distripution in
the
% heated material
```

```
if ~isempty(IAEvalue)
    c = dt*kappa/dx^2;
    Temp = zeros(nt,nx+2); % nx+2 is for the input and output tempera-
tures
    Temp(1,:) = Tinitial; % Temperature at t = 0
    Temp(:,nx+2) = 0; % output temperature is 0
    p = 1;
    Heatin = OUT.heatIn.Data; % Input temperatures come from IaeSimula-
tion
    Temp(:,1) = Heatin;
    while p < nt
        for m = 2:nx+1
        Temp(p+1,m) = c*Temp(p,m-1)+(1-2*c)*Temp(p,m)+c*Temp(p,m+1);
        end
        p = p+1;
    end
    Temp(:,1) = [];
    Temp(:,nx+1) = [];

    xbar = 0:dx:x; % Horizontal axis of the distribution picture (spa-
tial)
    t = 0:dt:tfin;  % Vertical axis of the distripution (time)
    figure
    heat = pcolor(xbar,t,Temp); % Draws the picture of the matrix
    shading interp; % Blends the colours
    colorbar
    Tx = Temp(nt,nx) %Temperature at the xcoordinate in the end of the
simulation
end
```

```
if ~isempty(IAEvalue)
%Validations:
   %Validation via Gapp:
    open('simulinkValidation.slx');
    sim('simulinkValidation.slx');

    % Validation via inverse fourier transformation:
    wt = linspace(-wmax,wmax,1000);
    ss = 1i*wt;

    % Controller FR
    if PI_controller == 1
        K = Kp*ss+Ki;
    else
        K = Ki;
    end

    Gpp = exp(-x*sqrt((ss)*b));  % Process FR
```

```
    % Sensor FR
    if a1 == 0 && a2 == 0
        Gss = 1;
        q = 0;
    elseif a2 == 0 && a1 ~= 0
        Gss = 1./(a1*ss+1);
        q = a1./(a1*ss+1);
    elseif a1 ~= 0 && a2 ~= 0
        Gss = 1./(a1*ss.^2+a2*ss+1);
        q = (a1*ss+a2)./(a1.*ss.^2+a2.*ss+1);
    end


    Gerror = Tgoal*(1+Gpp.*K.*q)./(ss+Gss.*Gpp.*K); % Error FR

    syms tt


    ft = Gerror.*exp(1i*wt.*tt);
    FT =1/(2*pi)*trapz(wt,ft); % Inverse fourier transform
    t2 = linspace(0,tfin,39);

    FTs = subs(FT,tt,t2);
    plot(t2,FTs)
end
```