Luentorunko kurssiin 477604S

Automaatiotekniikan laskentatyökalut (ALT)

Enso Ikonen 8/2008

Oulun yliopisto, Prosessi- ja ympäristötekniikan osasto, systeemitekniikan laboratorio

Sisältö

Johdanto kurssiin

Säätö- ja systeemitekniikan kirjallisuus on pullollaan algoritmeja: säätöalgoritmeja, identifiointialgoritmeja, optimointialgoritmeja, simulointialgoritmeja, testausalgoritmeja, suunnittelualgoritmeja, jne. Algoritmi on sarja ohjeita tietokoneelle, joiden avulla voidaan ratkaista matemaattinen ongelma. Algoritmeja voidaan toteuttaa konekielisenä, ohjelmointikielten avulla (Fortran, c, lisp, ...), tai interaktiivisten ohjelmistojen ja niihin liittyvien kielten avulla (MATLAB, LABView, Maple, jne). Tässä kurssissa perehdytään MATLAB/Simulink-ohjelmiston ominaisuuksiin ja käyttöön.

*MATLAB*in asema säätötekniikan tutkimuksen alalla on tukeva, suorastaan ylivoimainen. MATLABin perusohjelmisto tarjoaa vankkaa matriisialgebran hallintaa, maustettuna laajalla kirjolla ohjelmointi-, piirto-, yms. ominaisuuksia. *Simulink* on dynaamisten systeemien graafinen simulointi-ohjelmisto, joka on integroitu MATLABin yhteyteen. MATLABin perusohjelmistoa voidaan laajentaa lukuisien MATLAB-kirjastojen eli *Toolbox*ien avulla, joita voi myös tehdä itse.

Ensimmäisessä luvussa tutustutaan MATLABin perustoimintoihin ja funktioihin. Toisessa luvussa keskitytään lineaaristen vakioparametristen (lti, *linear time invariant*) systeemien simulointiin. Nämä ovat niitä joita voidaan näppärästi kuvata siirtofunktioiden ja tilamallien avulla, ja joiden varassa säätötekniikan perusta lepää. Kolmannessa luvussa käsitellään Simulink-ohjelmistoa ja sen käyttöä. Neljännessä luvussa perehdytään säätösuunnitteluun Control System Toolboxin avulla. Viidennessä luvussa tarkastellaan vielä mm. differentiaaliyhtälöiden ja optimointiongelmien ratkaisemistapoja MATLABia käyttäen sekä System Identification Toolboxia.

Kurssiin kuluu 35 luento- ja harjoitustuntia, materiaaleineen. Jotta MATLABista olisi hyötyä myöhemmässä elämässä, on sen käytössä saavutettava ainakin auttava rutiini. Tässä asiassa ei mikään ohjattu opetus voita omaehtoista harrastuneisuutta.

Materiaali perustuu MATLABin versioon 6 (R12) / Simulink 4.0 / Control System Toolbox 5.0 ja Windows XP:hen. Uudempien versioiden päivitettyä opastusmateriaalia löytyy englanninkielisenä MATLABista itsestään, esim. kohdasta "*Help|MATLAB Help|Learning MATLAB*"[1]. Joiltain osin olen lainannut materiaalia tämän kurssin aineistoista aiemmilta vuosilta, tai kehitellyt sitä itse lisää. Säätösuunnittelun työkalujen osalta esimerkkejä löytyy mm. Dorfin oppikirjasta[2].

Ajatuksenani on ollut pyrkimys mahdollistaa kulloisenkin MATLABin version mukana toimitettavien oppaiden suoraviivainen käyttö. En pitänyt tarpeellisena kääntää likikään kaikkia komentojen opasteita suomeksi, koska itse ohjelmakin on englanninkielinen. Keskeisimmät MATLABin 'kielet' ovat kuitenkin matriisialgebra (explisiittisesti) ja c (implisiittisesti) <*imho*>, joiden alkeellinenkin hallitseminen helpottaa MATLABin oppimista huomattavasti. Joka tapauksessa MATLABilla koodaaminen on yleensä intuitiivista ja suoraviivaista.

Materiaali noudattaa aiempaa materiaalia kurssilta "tietokoneavusteinen säätösuunnittelu", pienin korjauksin. Olen parhaani mukaan koettanut välttää rönsyilyä ja pitäytyä perusasiassa, joka meidän tapauksessamme on säätösuunnittelun tekeminen tietokoneen kera niin että tietokoneesta olisi jotain hyötyä. On siitä.

Oulussa, elokuussa 2008

Enso Ikonen
`Enso.Ikonen@oulu.fi`
dosentti, TkT, systeemitekniikan yliassistentti

Systeemitekniikan laboratorio
Prosessi- ja ympäristötekniikan osasto
Oulun yliopisto

[1] Suomenkielinen luettelo keskeisistä MATLAB-komennoista löytyy linkistä http://www.control.hut.fi/Kurssit/MATLAB/, samoin kuin muutakin MATLAB-oppimateriaalia. MATLABin viralliset kotisivut löytyvät osoitteesta www.mathworks.com.

[2] Dorf, R & R Bishop (2005) *Modern Control Systems*, Pearson Prentice Hall (10th ed). www.prenhall.com/dorf (kts. mm. Resources|MATLAB and Simulink Downloads)

1 Johdanto MATLABiin

1.1 Mikä MATLAB on?

MATLAB-nimi tulee sanoista MATrix LABoratory. MATLAB on alunperin fortran-ohjelmointikielellä kirjoitettu matriisienkäsittelyohjelma, jonka tarkoituksena oli olla interaktiivinen ja helppo tapa käyttää LINPACK- ja EISPACK-projektien matriisialgoritmeja. Alkuperäinen MATLAB oli julkisohjelma, jota levitettiin yliopistoille. Nykyisin käytettävissä olevat versiot ovat The MathWorks Inc.:n toimittamia kaupallisia tuotteita.

Valehtelematta voidaan kehua MATLABin olevan..

- mainio matriisien käsittelijä (ml. matriisialgebra),
- yleiskäyttöinen ohjelmointikieli moninaisiin tilanteisiin,
- laajennettavissa hyvin monille erikoisaloille valmiilla Toolboxeilla,
- saatavissa useimmille käyttöjärjestelmille.

Säätötekniikan tutkimuksen alueella MATLABin rooli on suorastaan musertavan ylivoimainen.

MATLABin heikkouksiin kuuluvat..

- kalleus (kaupallinen ohjelma, johon Toolboxit on ostettava erikseen),
- laajuus ja rönsyily uusille alueille (koodi vaikeutuu monipuolistuessaan),
- epävarmuus uusien versioiden yhteensopivuudesta aiemman koodin kanssa (käytännössä mm. Toolboxit on syytä päivittää joka versiolle),
- koodin tehottomuus (esim. vähänkin raskaamman grafiikan pyörittäminen saattaa olla takkuista). Tehottomuus ei kuitenkaan koske matriisien käsittelyä, itse asiassa asiallisesti vektoroitu koodi on erittäin kilpailukykyistä.

MATLAB-ohjelmisto sisältää itsessään kattavan dokumentoinnin esimerkkeineen. Itseopiskelupaketti koostuu kahdesta osasta: lyhyemmästä johdannosta 'Getting Started' sekä varsinaisesta käyttöohjeesta 'Using MATLAB'. Jokaisesta MATLABin funktiosta löytyy kuvaus otsikon 'Reference' alta.

1.2 Suunnitteluympäristö

MATLAB käynnistetään klikkaamalla MATLABin kuvaketta. MATLABista poistutaan sulkemalla ikkuna. MATLABin käyttöliittymä koostuu useista osioista. Näytettävät osiot valitaan valikosta View. Tärkein osio on ns. komentoikkuna (Command Window). Komentoikkunaan syötetään MATLABille annettavat tekstikomennot. Siinä syötetään muuttujat, ja ajetaan funktiot sekä M-tiedostot. Nuolinäppäimien avulla voit selata ja muokata aikaisempia komentoja.

Kirjoittamalla komentoikkunaan `help` *funktion_nimi* (missä *funktion_nimi*-paikalle laitetaan sen funktion nimi josta tietoa halutaan) saat tietoja eri funktioista, niiden käyttötarkoituksesta, syntaksista, ja argumenteista. Kätevä ryhmittely kaikista MATLABin funktiosta löytyy 'Help Navigator' -ikkunan kohdasta 'MATLAB|Reference|MATLAB Function Reference'.

MATLAB käyttää funktioita joita se löytää hakupolkunsa osoittamista paikoista. Hakupolkua hallitaan valikosta 'File|Set path'.

MATLABin työtila (Workspace) koostuu muistiin tallentuneista muuttujista. Lisäät muuttujia työtilaan syöttämällä niitä suoraan, ajamalla funktioita, M-tiedostoja tai lukemalla tallennettuja tietoja. Työtilan sisältöä voit tutkia ikkunasta 'Workspace' tai komennoilla `who` ja `whos`. Muuttujia poistetaan komennolla `clear`.

MATLABin M-tiedostot ovat sinun itse kirjoittamiasi tai jonkun muun tarjoamia koodinpätkiä. M-tiedostot ovat puhdasta ASCII-tekstiä jota voi tuottaa millä tekstinkäsittelyohjelmalla tahansa, mutta MATLABin 'Editor/Debugger' automaattisine väreineen, sisennyksineen ja tarkistuksineen on kätevä "graafinen" käyttöliittymä oman koodin tuottamiseen. 'File|New M-file' avaa editorin.

1.3 Matriisien käsittely

MATLABin matriisi on nelikulmainen numerojoukko. 1x1 matriisit ovat skalaareja, matriisit joissa on vain yksi rivi tai sarake ovat pysty- tai vaakavektoreita. MATLABissa on monia muuttujatyyppejä, mutta aluksi on helpointa ajatella kaikkia muuttujia matriiseina. Matriiseja työstetään MATLABin komennoilla.

1.3.1 Matriisien syöttö ja niihin viittaus

Matriiseja voi syöttää MATLABiin monella tavalla: näppäillä komentolistana, lukea ulkoisesta datatiedostosta, tai generoida sisäisten funktioiden tai M-tiedostojen (funktioiden tai skriptien) avulla.

Komentolistauksessa lista aloitetaan ja suljetaan hakasuluilla, matriisin elementit erotetaan välilyönnillä tai pilkulla, ja puolipiste päättää rivin.

Example 1 *Matriisin* `A` *syöttäminen.*
```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB tallentaa matriisin automaattisesti työtilaan. Siihen voi viitata yksinkertaisesti sen nimellä (esimerkissä `A`)

Komennolla `B=sum(A)` voidaan laskea `A`:n sarakkeiden summat, tuloksena on vaakavektori joka sijoitetaan muuttujaan `B`. Jos ulostulomuuttujaa ei määrätä komennon vasemmalla puolella, (viimeisin) tulos tallentuu muuttujaan `ans`. `A`:n rivien summat voidaan laskea ottamalla matriisista transpoosi ja siitä summa: `sum(A')`. `A`:n diagonaalielementit saadaan komennolla `diag(A)` ja esim. diagonaalielementtien summa komennolla `sum(diag(A))`.

Matriisin `A` elementtiin rivillä i ja sarakkeella j viitataan `A(i,j)`:llä. Kaksoispiste on MATLABin tärkeimpiä operaattoreita. Lauseke `1:10` on rivivektori joka sisältää kokonaisluvut yhdestä kymmeneen. Jos lukujen halutaan kasvavan muun kuin ykkösen verran voidaan muutos määrätä.

Example 2 *Komento* `100:-7:50` *tuottaa vektorin [100 93 86 79 72 65 58 51] jonka elementit vähenevät -7:n verran.*

Matriisin osiin voidaan viitata indeksien avulla, esim `A(1:k,j)` viittaa matriisin `A` j:nnen sarakkeen k:hon ensimmäiseen riviin.

Example 3 *Kaikki seuraavat komentorivit viittaavat 4x4 matriisin* `A` *samoihin elementteihin:*
```
A(:,2)
[A(1,2);A(2,2);A(3,2);A(4,2)]
A(5:8)'
i=[1:4]; A(i,2)
```

Pidempiä matriiseja voidaan yhdistää suuremmiksi ketjuttamalla.

Example 4 *Jos* `A` *on 4x4 matriisi, niin* `B=[A A;A A]` *tuottaa 8x8 matriisin* `B`*.*

Matriisin rivejä ja sarakkeita voi poistaa sijoittamalla niiden arvoksi tyhjän matriisin.

Example 5 *Sijoitukset* `C=B; C(:,2)=[]` *tuottavat 8x7 matriisin* `C`*.*

Exercise 6 *Kokeile ja harjoittele matriisien ja vektoreiden syöttämistä sekä matriiseihin ja niiden osiin viittaamista.*

1.3.2 Lausekkeet

MATLABin lausekkeet käsittelevät kokonaisia matriiseja. Lausekkeiden rakennuspalasia ovat muuttujat, numerot, operaattorit ja funktiot.

MATLABin muuttujien tyyppiä tai kokoa ei tarvitse määritellä, vaan se tapahtuu automaattisesti kun MATLAB havaitsee uuden muuttujan. Jos muuttuja on jo olemassa, se korvataan uusimmalla. Muuttujien nimet alkavat kirjaimella ja jatkuvat kirjaimella, numerolla tai alaviivalla, maksimimitta on 31 merkkiä. Isot ja pienet kirjaimet erotetaan toisistaan. Suomen kielen erikoismerkkejä MATLAB ei tunnista (ä, ö, jne).

Luvut esitetään desimaalilukuna. Imaginääriluvut käyttävät joko i:tä tai j:tä etuliitteenä. Luvut tallennetaan 16-bittisinä liukulukuina.

Operaattoreita ovat mm. ne tavalliset: `+`, `-`, `*`, `/` ja ^ (potenssiin korotus). Matriisialgebran perusoperaatioita ovat yhteen-, vähennys-, kertolasku ja transpoosi, esim. `A+A, A-A, A'*A`. Kaarisuluilla vaikutetaan laskujärjestykseen.

Exercise 7 *Kokeile ja harjoittele matriisialgebran perusoperaatioiden käyttöä. Tutustu imaginäärilukujen syöttämiseen ja esittämiseen MATLABissa.*

MATLAB sisältää suuren määrän matemaattisia funktioita. `help elfun` listaa perusfunktiot, kts. myös `help elmat`. Näistä kannattaa perehtyä ainakin seuraaviin

- `pi`, `inf`, `NaN`,
- `sin`, `cos`, `tan`, `exp`, `log`,
- `zeros`, `ones`, `eye`,
- `rand`, `randn`,
- `sqrt`, `abs`, `real`, `round`,
- `size`, `disp`, `find`.

Exercise 8 *Tutustu* ***`elfun`****-funktioihin* ***`help`*** *komennon avulla. Kokeile funktioiden toimintaa skalaari- ja matriisitapauksissa.*

1.3.3 Lisää matriiseista

Vaikka käytännön kielessä puhutaan yleisesti matriiseista, tarkkaan ottaen matriisit (englanniksi *matrix*) liittyvät lineaarialgebrassa esiintyviin lineaarisiin muunnoksiin, muissa yhteyksissä tulisi puhua kaksiulotteisista "ryhmittelyistä" (englanniksi *array*).

Matriisialgebra on MATLABin vahvaa aluetta. Seuraaviin komentoihin kannattaa tutustua:

- `det`, `inv`, `pinv`, `eig`, ^, `poly`

Exercise 9 *Tutustu matriisialgebran perusoperaatioihin ja -komentoihin* `help` *komennon avulla. Kokeile niiden toimintaa skalaari- ja matriisitapauksissa.*

Lineaarialgebran ulkopuolella ("ryhmittelyiden" tapauksessa) operaatiot tehdään elementti kerrallaan. MATLAB käyttää pistettä merkkinä elementtikohtaisesta operaatioista: `.*`, `./`, `.^`.

Example 10 `A*A` *ei ole sama kuin* `A.*A`.

Monimuuttujatapauksessa MATLAB suorittaa operaatiot sarakkeittain. Kun argumenttina on matriisi, esimerkiksi operaatiot `sum`, `mean` ja `std` palauttavat rivivektorin joka antaa summan, keskiarvon ja keskihajonnan kunkin sarkkeen sisältämälle datalle. `help datafun` listaa data-analyysin työkaluja. Kannattaa tutustua ainakin seuraaviin:

- `max`, `min`, `mean`, `median`, `std`, `var`
- `sort`, `sum`, `cumsum`, `prod`, `diff`

Exercise 11 *Tutustu* `datafun`*-funktioihin* `help` *komennon avulla. Kokeile niiden toimintaa vektori- ja matriisitapauksissa.*

Matriiseja ja skalaareita voi yhdistää monin eri tavoin.

Example 12 *Mm. seuraavat operaatiot ovat sallittuja:*
```
B=1.2*A
B=A-8.5
B(1:2,2:3)=0
```

Loogisia vektoreita (joita syntyy loogisissa ja vertailuoperaatioissa) voidaan myös käyttää matriisien (ja yleisemmin kaikkien ryhmittelyjen) elementteihin viittaamiseen. Jos `X` ja `L` ovat samankokoisia matriiseja, missä `L` on jonkin loogisen operaation tulos, `X(L)` viittaa `X`:n niihin elementteihin joissa `L` on nollasta poikkeava. `find` komennolla taas löydetään indeksit niihin elementteihin jotka täyttävät annetun ehdon.

Example 13 `L=A(:,1)<10` *antaa tulokseksi vektorin [0 1 1 1]'.*
`find(A(:,1)<10)` *antaa tulokseksi vektorin [2 3 4]'*
`A(L)` *antaa tulokseksi [5 9 4]'*

1.3.4 Komentoikkunan i/o

`load` -komento lataa aiemmassa MATLAB sessiossa `save` -komennolla tallennetun binääritiedoston massamuistista.

`format` komennolla vaikutetaan komentoikkunassa näytettävien lukujen ulkoasuun. `format compact` on kätevä. Jos komennon perään lisätään puolipiste, sen tulos jätetään näyttämättä komentoikkunassa. Pitkiä komentoja annettaessa kolme pistettä (`...`) ja niiden jälkeinen 'entteri' (**Enter**, **Return**) jatkavat komentoa seuraavalle riville. `disp` komennolla voi tulostaa lukuja ja tekstiä komentoikkunaan, monipuolisemmin tulostettavaan ulkoasuun pääsee vaikuttamaan `sprintf` komennolla jonka syntaksi muistuttaa c-kielen vastaavia komentoja.

1.4 Kuvien piirtäminen

Peruskuvat piirretään `plot`-funktion avulla, tekstit lisätään `xlabel`, `ylabel` ja `title` komennoilla.

Example 14 *Siniaallon kuvaaja kuvateksteineen piirretään seuraavilla komennoilla:*

```
x = 0:pi/100:2*pi;
y = sin(x);
h = plot(x,y)
xlabel('x = 0:2\pi')
ylabel('sini x:stä')
title('Sinikäyrä','FontSize',12)
```

`plot`-funktion moniin muotoihin kannattaa tutustua. Jos samaan kuvaan piirretään useita käyriä, `legend` komento lisää kuvaan selosteen. Usein on kätevää jakaa kuva useampaan osaan, jolloin piirrettävä osa valitaan `subplot` komennolla. Akseleita säädetään `axis` komennolla. Usein järkevä vaihtoehto `plot`-komennolle on `stairs`-komento.

Jos halutaan kontrolloida yksityiskohtaisesti piirretyn kuvan eri ominaisuuksia, `plot`-funktion ulostuloargumenttina on ns. kahva (engl. *handle*, esimerkin `h`) jonka avulla kunkin piirretyn käyrän ominaisuuksia voidaan tarkastella ja määrittää `get` ja `set` komentojen avulla. Vastaavasti akseleiden ja ikkunan kahvat saadaan komennoilla `gca` ja `gcf`.

Valmis kuva voidaan tallentaa `print` komennolla. *Post script* -formaatti on useimpien tekstinkäsittelyohjelmien tukema hyvälaatuinen vektorigrafiikkakuvaformaatti. MATLABilla voit tehdä eps-värikuvan komennolla `print -depsc` *kuva*, jolloin kuva tallentuu tiedostoon *kuva*.eps.

MATLABin nykyisillä versioilla voi tehdä monenlaista grafiikkaa, esimerkiksi piirtää kolmiulotteista dataa, käsitellä bittikarttoja ja animaatioita, tai tehdä vaikkapa käyttöliittymiä liukukytkimineen ja valikkoineen. Lisätiedon etsimisen voi aloittaa MATLABin 'Help Navigator|Getting Started|Graphics' osiosta.

1.5 MATLABilla ohjelmointi

MATLAB on mainio ohjelmointikieli tutkijalle, johtuen sen tulkattavasta kielestä (ohjelmia ei tarvitse kääntää), vahvasta matriisien käsittelystä (joiden oikeaoppinen käyttö tekee ohjelmakoodeista kompakteja ja nopeita suorittaa) sekä monien kirjastojensa ansiosta (englanniksi *Toolbox*).

1.5.1 if, for, while, ...

MATLABin ohjelmien suoritusjärjestyksen kontrollirakenteet ovat tuttuja muista ohjelmointikielistä.

`if` komento arvioi loogisen lausekkeen totuusarvon, ja suorittaa joukon komentoja jos lauseke on tosi. Komentojen suoritusta voi lisäksi jakaa `else` ja `elseif` komennoilla. `end` päättää `if`-komentojoukon.

Example 15 *Esimerkki if-rakenteesta.*

```
if rand(1)>0.1
 disp('90% todennäköisyys');
else
 disp('Älä veikkaa tätä hevosta');
end
```

Ole tarkkana, sillä MATLABin loogiset vertailut toimivat myös matriiseille. Jos `A` ja `B` ovat sopivankokoisia matriiseja, ekvivalenttisuusrelaatio `A==B` antaa tulokseksi vastaavankokoisen matriisin totuusarvoja. `isequal` vertaa matriisien `A` ja `B` samanlaisuutta ja antaa tulokseksi skalaarin. Kannattaa perehtyä myös funktioihin `all` ja `any`.

`switch` komennossa erotellaan suoritettavat komentojoukot eri argumentin arvoilla (`case`).

`for` silmukka toistaa komentojoukon ennalta määrätyn määrän verran.

Example 16 *Esimerkki kahdesta sisäkkäisestä for-silmukasta.*

```
for i = 1:m
 for j = 1:n
   H(i,j) = 1/(i+j);
 end
end
```

`while` silmukka toistaa komentojoukon niin kauan kuin sen argumentti on tosi (vaikka loputtomiin...). Kuten kaikki muutkin komennot yllä, sekin päättyy `end` komentoon.

`break`-komento siirtää ohjelman suorituksen suoraan ulos `while` tai `for` silmukasta. Esim. FORTRANista ja BASICista tuttua `goto`-käskyä ei MATLABissa ole.

1.5.2 Muita datarakenteita

Tähän asti olemme olettaneet että MATLABin muuttujat ovat kaksiulotteisia matriiseja (tarkkaan ottaen puhumme kaksiulotteisista 'ryhmittelyistä'). Muitakin muuttujatyyppejä on.

Tekstimuuttujat annetaan yksinkertaisten heittomerkkien sisällä, esim `str = 'Moi!'` määrittää muuttujan `str` joka on merkkijono (eikä liukuluku).

MATLAB tukee myös kaksiulotteista moniulotteisempia ryhmittelyjä.

Example 17 *Komento `randn(3,4,5)` luo kolmiulotteisen 3x4x5 matriisin jossa on 60 (normaalijakautunutta) elementtiä.*

Soluryhmittelyt (*cell array*) ovat ryhmittelyjä joiden elementit ovat toisia ryhmittelyjä. Useimmiten niitä käytetään sekalaisten kokoelmien luontiin. Niitä luotaessa ja niihin viitattaessa käytetään aaltosulkuja.

Struktuurit ovat moniulotteisia MATLABin ryhmittelyjä joiden elementteihin viitataan tekstimääritteen avulla. Struktuurit ovat käteviä kun halutaan pitää monentyyppistä dataa järjestyksessä.

Näiden datatyyppien käyttö vaatii hieman harjaannusta. Pienissä koodinpätkissä joissa lasketaan vaikkapa malleja ja säätimiä, pärjää aluksi mainosti ilman syvällisempää perehtymistä näihin tyyppeihin.

1.5.3 Skriptit ja funktiot

MATLABin ohjelma- tai kooditiedostoja kutsutaan M-tiedostoiksi. M-tiedostot ovat tekstitiedostoja jotka sisältävät MATLAB-komentoja. Tiedoston nimeä käytetään koodin ajamiseen, nimi päättyy .m liitteeseen. Siis M-tiedosto koodi.m ajetaan antamalla komentoikkunassa komento `koodi`. M-tiedostoja luodaan tekstieditorilla, ja käytetään kuten mitä tahansa MATLABin funktiota tai komentoa.

Jotta MATLAB löytää luodut M-tiedostot, tulee niiden olla hakupolun päässä. Jos samannimisiä tiedostoja on useita, hakupolun järjestys määrää mikä tiedostoista suoritetaan. Komennolla `type` voit tulostaa M-tiedoston, `which` kertoo mistä se löytyy.

M-tiedostoja on kahdenlaisia:

- Skripteillä ei ole sisäänmeno tai ulostuloargumentteja. Ne operoivat työtilan muistiavaruudessa. Skriptit yksinkertaisesti suorittavat niiden sisältämät komennot yksi kerrallaan.
- Funktioilla voi olla sisäänmeno ja ulostuloargumentteja. Jokaisella funktiolla on oma paikallinen muistiavaruus.

Funktio alkaa määrittelyllä `function`, jota seuraavat ulostuloargumenttien, funktion nimen ja sisäänmenoargumenttien määritykset. Kannattaa noudattaa hyvää ohjelmointitapaa ja lisätä seuraavaksi lyhyt kuvaus ohjelman toiminnasta (jonka kutakin riviä edeltää kommenttimerkki `%`). Tämä kuvaus tulostuu `help` komennolla. Tämän jälkeen seuraa varsinainen koodi.

Example 18 *Esimerkkifunktio.*

```
function [mean,stdev] = stat(x)
%STAT Perusstatistiikkaa.
n = length(x);
mean = sum(x) / n;
stdev = sqrt(sum((x - mean).^2)/n);
```

Exercise 19 *Kirjoita esimerkkifunktion koodi M-tiedostoksi ja tallenna se. Luo sopivat sisäänmenoargumentit ja suorita funktio MATLABin komentoikkunasta.*

Funktion sisällä `nargin` ja `nargout` kertovat käyttäjän funktiolle antamien argumenttien määrän. Funktion sisällä voidaan samassa M-tiedostossa määrittää myös uusia funktioita (käyttämällä `function`-määrittelyä), jolloin ne näkyvät vain kyseisessä M-tiedostossa määritellyille funktioille.

Ohjelmia kirjoitettaessa ovat mahdollisten virheiden etsinnässä hyvänä apuna debuggaus-komennot, kts. `help debug`. Funktion sisällä voit pysähtyä `dbstop`-komennolla, ja poistua debuggaus-tilasta komennolla `dbquit`.

Ohjelmien suoritusnopeuden kannalta hyvä vektorointi on keskeistä. Usein kannattaa tutkia mahdollisuutta korvata `for`-silmukka vastaavalla vektorioperaatiolla. Nopeutta voi myös kasvattaa minimoimalla suoritusaikaisen tulostuksen ja piirtämisen, sekä alustamalla mahdollisia koodin suorituksen myötä suureksi kasvavia muuttujia etukäteen.

1.6 Yhteenveto

- Keskeisiä termejä olivat: MATLABin komentoikkuna, Toolbox, matriisi, funktio, skripti.
- MATLAB sisältää runsaan dokumentoinnin, johon pääsee käsiksi esim. `help`-komennolla.
- Matriisit syötetään hakasulkeiden sisällä, erottimina toimivat välilyönti ja puolipiste. Matriisien elementteihin viitataan indekseillä.
- Laskutoimitukset käsittelevät matriiseja. Perusoperaatioita ovat: `+ - * '`.
- Kuvien piirtämisen peruskomento on `plot`.
- MATLAB-ohjelmat ovat ASCII-koodina kirjoitettuja lausekkeita ja komentoja. Koodi tallennetaan M-tiedostoksi, joka loppuu .m päätteeseen. Funktioiden sisääntulo- ja ulostulotiedot määritellään koodin alussa `function` komennon yhteydessä.
- MATLAB-ohjelmien suoritusjärjestystä ohjataan mm. `for` ja `if` rakenteilla, jotka päättyvät `end`-komentoon.

2 LTI-systeemit

LTI-systeemeillä tarkoitetaan lineaarisia aikainvariantteja systeemejä (*Linear Time-Invariant*). Vakioparametriset jatkuva- ja diskreettiaikaiset siirtofunktioesitykset ja tilamallit kuuluvat näihin lti-systeemeihin.

2.1 Koodaus ilman lti-funktioita

Aloitetaan lti-systeemien käsittely simuloimalla suljettua piiriä joka sisältää ensimmäisen kertaluvun prosessin ja P-säätimen. Rakennetaan ensin systeemi niistä palikoista jotka tähän asti on esitetty, luvussa 2.2 siirrytään käyttämään erityisesti lti-systeemeille kehitettyjä lti-funktioita.

Ensimmäisen kertaluvun prosessi Laplace-tasossa

$$\frac{Y(s)}{U(s)} = \frac{k_\mathrm{p}}{\tau s + 1}$$

voidaan kirjoittaa jatkuva-aikaisessa muodossa

$$\frac{d}{dt} y(t) = \frac{1}{\tau}\left[-y(t) + k_\mathrm{p} u(t)\right],$$

Z-muunnokseksi saadaan (käyttämällä napa-nolla vastaavuutta $z = e^{sh}$)[3] ja mittausväliä h

$$\frac{Y(z)}{U(z)} = \frac{\left(1 - e^{\frac{-h}{\tau}}\right) k_\mathrm{p}}{z - e^{\frac{-h}{\tau}}}$$

Kirjoitetaan yhtälö vielä differenssimuotoon

$$y(k+1) = a y(k) + b u(k)$$

missä $a = e^{\frac{-h}{\tau}}$ ja $b = \left(1 - e^{\frac{-h}{\tau}}\right) k_\mathrm{p}$.

P-säädin on vahvistettu erosuure halutun ja mitatun ulostulon välillä, ja differenssimuodossa saadaan:

$$u(k) = K_\mathrm{P} e(k) = K_\mathrm{P}\left[w(k) - y(k)\right].$$

Differenssimuotoinen systeemi on suoraviivainen simuloitava.

Example 20 *Esimerkki 1. kertaluvun prosessin P-säädetyn systeemin simuloinnista ilman lti-funktioita. (M-tiedosto Ex_lti1.m)*

```
% Ex_lti1-skripti.
%
% Esimerkki 1.kertaluvun diskreetin prosessin
% P-säädetyn systeemin suljetun piirin simuloinnista
```

[3] Systeemin karakteristisesta yhtälöstä saadaan napa: $\tau s + 1 = 0 <=> s = -\frac{1}{\tau}$. Sijoitetaan diskreetin systeemin napa samaan paikkaaan: $z = e^{sh} = e^{-h/\tau}$. Diskreetin systeemin karakteristiseksi yhtälöksi saadaan siis $z - e^{h/\tau} = 0$.

Asetetaan jatkuva-aikaisen ja diskreetin mallin tasapainotilan vahvistukset samoiksi:

$$\lim_{s\to 0} \frac{k_\mathrm{p}}{\tau s + 1} = \lim_{z\to 1} \frac{K}{z - e^{\frac{-h}{\tau}}}$$

josta saadaan ratkaistua diskreetin mallin vahvistus K:

$$K = \left(1 - e^{\frac{-h}{\tau}}\right) k_\mathrm{p}$$

```
% Jatkuva-aikaisen mallin parametrit
% (voit kokeilla eri arvoilla)
k_p = 5; tau = 10; % vahvistus ja aikavakio

% diskreettiaikaisen mallin parametrit
h = 3; % sämpläysväli
a = exp(-h/tau);
b = (1-exp(-h/tau))*k_p;

% P-säädin (Ziegler Nicholsin mukaan)
K_P = tau/(k_p*h);

% asetusarvosekvenssi
w = [zeros(10,1);ones(10,1);-2*ones(10,1)];
% asetusarvojen määrä (=simuloinnin pituus)
lw = length(w);

% prosessin alkutila
y(1) = 0;

% mittaus/säätösilmukka

for k=1:lw
    % erosuure e
    e = w(k) - y(k)
    % prosessin ohjaus u
    u(k) = K_P * e;
    % prosessin ulostulo y
    y(k+1) = a * y(k) + b * u(k);
end


% piirretään kuvaajat ulostulosta ja ohjauksesta

t = h*[1:lw]';

subplot(211);
[x1,y1]=stairs(t,y(1:lw));
[x2,y2]=stairs(t,w(1:lw));
plot(x1,y1,'-',x2,y2,':');
ylabel('y')
ax=axis; axis([0 h*lw ax(3:4)])

subplot(212);
stairs(t,u);
ylabel('u')
xlabel('t')
ax=axis; axis([0 h*lw ax(3:4)])
```

Edelläolevan esimerkin perusteella huomataan, että näinkin yksinkertaisen systeemin simuloinnissa on jo oma työnsä. Ennen kaikkea aikaa kuluu ongelman kirjoittamiseen soveliaaseen muotoon (differenssi- tai tilamallimuotoon).

Monimutkaisempien siirtofunktioiden kanssa pelattaessa polynomin juuret löytyvät `roots` komennolla, toisin päin päästään `poly`-komennolla. Polynomien kertolasku tapahtuu `conv`-funktion avulla. `residue` antaa kahden polynomin suhteen (eli siirtofunktion) osamurtokehitelmän.

2.2 Mallien rakentaminen

Control Systems Toolbox (CST) sisältää mainiot työkalut lti-systeemeille. Tässä luvussa perehdymme systeemien rakentamiseen ja analysointiin, luvussa 4 jatkamme varsinaisilla säätösuunnittelumenetelmillä.

2.2.1 Lineaariset mallit

CST tukee mm. mallin esitystapoja tilamalleina, siirtofunktiomalleina ja napa-nolla-vahvistus-muodossa, niin jatkuva-aikaisina kuin diskreettiaikaisinakin, sekä muunnoksia näiden muotojen välillä. CST käyttää erityisiä datastruktuureja, ns. lti-objekteja, lti-systeemien tallentamiseen.

Usein säätöinsinööri lähtee liikkeelle differentiaaliyhtälömuodosta. Oletetaan nyt että malli on lineaarinen (tai linearisoitu), ja siis muotoa

$$\begin{aligned} \frac{d}{dt}\mathbf{x} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned}$$

Kun malli (ja sen parametrit **A**, **B**, **C** ja **D**) ovat tiedossa, komennolla `ss` voidaan muodostaa tilamalliobjekti.

Example 21 *Jos `A`, `B`, `C` ja `D` matriisit on määritetty, seuraava komento määrittää lti-tilamallin*
`sys = ss(A,B,C,D)`

Vastaavasti malli voidaan muodostaa komennoilla `tf` (siirtofunktio) tai `zpk` (napa-nolla-vahvistus), jos nämä muodot ovat tiedossa. Näihin tulee ehdottomasti tutustua.

Example 22 `sys_tf = tf(num,den)`
`sys_zpk = zpk(z,p,k)`

Kunhan joku muodoista on tunnettu, voidaan muodoista toiseen siirtyä vastaavilla komennoilla.

Example 23 `sys_tf = tf(sys)`
`sys_zpk = zpk(sys)`
`sys_ss = ss(sys_zpk)`
jne.

Diskreettiaikaiset systeemit määritellään vastaavilla tavoilla. Komennotkin ovat samoja, lisättynä ainoastaan argumentilla joka määrittää sämpläysajan h, esim. `tf(num,den,h)`. Siis, jos sämpläysaika on annettu tulkitaan malli diskreettiaikaiseksi. Tämä on kätevää, mutta kannattaa olla tarkkana.

Muunnokset jatkuva-aikaisten ja diskreettiaikaisten mallimuotojen välillä tehdään komennoilla `c2d`, `d2c`. Komento `d2d` sämplää mallin uudella taajuudella.

Systeemimalleihin voi määritellä viiveen. Sisäänmenoviive määritellään ominaisuudella 'InputDelay'.

Example 24 *Diskreettiaikaiselle siirtofunktiomallille viivellä `d` syntaksi on muotoa:*
`tf(num,den,h,'InputDelay',d)`.

LTI-objektien sisältöjä voi tarkastella `get` komennolla, ja manipuloida `set` komennolla. Esimerkiksi edellä mainittu 'InputDelay' on eräs ominaisuuksista, sämpläysajan lti-ominaisuuden nimenä käytetään `Ts`:ää. Muuttujille voi myös määrittää nimet

tätä kautta ominaisuuksilla 'InputName' ja 'OutputName'. `ltiprops` antaa lisää informaatiota näiden objektien sisällöstä.

CST sisältää monenlaisia mallin ominaisuuksista kertovia funktioita. `size` kertoo mallin dimensioista, `pole`, `zero` ja `dcgain` mallin navoista, nollista ja tasapainotilan vahvistuksesta. `minreal` sieventää kumoutuvat navat ja nollat.

Monimuuttujamallien (MIMO, *Multiple-Input Multiple-Output*) luontiin käytetään samoja funktioita kuin SISO (*Single-Input Single Output*) tapaukseen. Erityisesti tilamallit taipuvat luontevasti MIMO tapauksiin.

Exercise 25 *Perehdy käytännössä LTI-muotojen (tilamalli, siirtofunktio, napanolla-vahvistus) syöttötapoihin ja eri muotojen välisiin muunnoksiin jatkuva-aikaisissa ja diskreettiaikaisissa tapauksissa.*

2.2.2 Lineaaristen mallien kytkeminen keskenään

LTI-objekteilla voi suorittaa yksinkertaisia operaatioita, kuten yhteenlasku (rinnakkainen kytkentä, `parallel`) ja kertolasku (sarja- tai kaskadikytkentä, `series`). `feedback`-komennon syntaksi on `feedback(`*sys1*`,`*sys2*`)`, missä *sys1* on eteenpäinkytketty osa ja *sys2* on takaisinpäin kytketyn haaran siirtofunktio. Oletusarvona on negatiivinen takaisinkytkentä.

Example 26 *Muodostetaan P-säätimen ja prosessin muodostama suljetun piirin siirtofunktio kahdella tavalla.*

```
sys_p = tf(5,[10 1]);     % prosessi
sys_c = tf(2/3,1);     % P-säädin
sys_cl = feedback(series(sys_c,sys_p),1);     % tapa 1
sys_cl = (sys_c*sys_p)/(1+sys_c*sys_p);     % tapa 2
```

Operaatioita voi suorittaa, vaikka mallit eivät olisikaan samaa typpiä, tarvittavat muunnokset suoritetaan automaattisesti.

Exercise 27 *Kokeile suljetun piirin muodostamista.*

Valitettavasti `feedback` ei kykene käsittelemään viiveitä. Ongelman voi kiertää tekemällä viiveelle `pade`-approksimoinnin.

Example 28 *Viive määritellään objektin ominaisuudeksi, systeemistä tehdään viiveetön approksimaatio, ja sen suljetun piirin askelvastetta simuloidaan .*

```
sys_p.InputDelay = h
sys_p_apprx = pade(sys_p)
sys_cl = feedback(sys_c*sys_p_apprx,1,-1)
step(sys_cl)
```

2.3 Mallien analysointi

Helpoin tapa analysoida lti-malleja Control System Toolboxissa on käynnistää *LTI Viewer* komennolla `ltiview`. LTI Viewer on interaktiivinen ohjelma joka sisältää omat valikkokomentonsa ja monia hiiren käyttöön perustuvia toimintoja. Sen toimintaan perehtyminen kannattaa varmasti. Voit tarkastella mm. systeemisi...

- askel- ja impulssivastetta (ml. vaikkapa nousu- ja asettumisajat)
- Bode- ja Nyquist-kuvaajia (mm. vaihe- ja vahvistusvarat), sekä
- napa-nolla-karttaa.

Voit myös vertailla useampaa systeemiä yhtä aikaa.

Komentorivitasolla voit käyttää vastaavia komentoja `bode`, `impulse`, `step`, `lsim`, `margin` ja `pzmap`.

Example 29 *Esimerkin 20 simulointiskripti toteutettuna lti-työkalujen avulla.*

```
k_p = 5; tau = 10; h = 3;
sysp = c2d(tf(k_p,[tau 1]),h); % diskretoitu prosessimalli
sysc = tau/(k_p*h); % P-säädin
syscl = feedback(series(sysp,sysc),1); % suljettu piiri
W = [zeros(10,1);ones(10,1);-2*ones(10,1)]; T = [3:3:90];
lsim(sys_sulj_diskr,W,T) % simulointi
```

Exercise 30 *Tutustu* `ltiviewer`*in toimintaan.*

Seuraavassa kerrataan hieman 1. ja 2. kertaluvun lineaaristen systeemien perusteita. Luvussa 4 jatketaan kompensaattorien suunnittelulla. Sitä ennen esitellään Simulink luvussa 3.

2.3.1 Kertausta: systeemin käyttäytyminen s-tasossa

Systeemin suoriutumista voidaan kuvata sen karakteristisen yhtälön juurien sijainnilla s-tasossa (eli Laplace-tasossa). Usein systeemin dynaamisen käyttäytymisen kuvaamisessa keskitytään dominoivaan napapariin. Tämän takia on tärkeää osata hahmottaa luontevasti toisen kertaluvun systeemin napojen ja aikavasteiden välisiä yhteyksiä.

Systeemin siirtofunktio ns. *polynomimuodossa* kirjoitetaan

$$G(s) = \frac{b_1 s^m + b_2 s^{m-1} + \cdots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + \cdots + a_{n+1}}.$$

Ratkaisemalla osoittaja- ja nimittäjäpolynomien juuret, saadaaan ns. *napa-nolla-vahvistus* muoto, jossa systeemin siirtofunktio kirjoitetaan kahden faktoroidun polynomin suhteena

$$G(s) = K \frac{\prod_{i=1}^{m} (s - s_i^{\mathrm{z}})}{\prod_{i=1}^{n} (s - s_i^{\mathrm{p}})}$$

missä s_i^{z} ovat systeemin nollat ja s_i^{p} systeemin navat ja K on systeemin vahvistus. Puhuttaessa navasta $-\sigma$:ssa tarkoitetaan $s^{\mathrm{p}} = -\sigma$, eli $s + \sigma = 0$, jne. Reaaliarvoisille polynomeille kompleksijuuret esiintyvät aina pareittain (ns. kompleksi konjugaatti). Systeemin nollat ovat kohdat (taajuudet) joilla systeemin vahvistus on nolla. Vastaavasti systeemin navat ovat kohdat joilla systeemin vahvistus on ääretön.

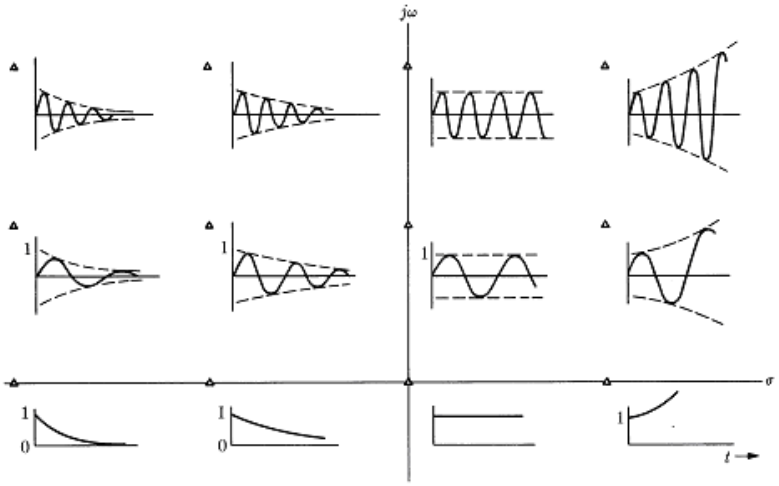
Ensimmäisen kertaluvun systeemi Muistetaan, että ensimmäisen kertaluvun systeemin

$$\frac{k}{\tau s + 1}$$

aikavakio on $\tau = \frac{1}{\sigma}$, navaksi saadaan $s^{\mathrm{p}} = -\sigma$, ja impulssivasteeksi $h(t) = e^{-\sigma t} 1(t)$.

Toisen kertaluvun systeemi Oletetaan toisen kertaluvun systeemille kompleksinavat

$$s^{\mathrm{p}} = -\sigma \pm i\omega_{\mathrm{d}}.$$



Kuva 1: Navat ja impulssivasteet.

Kompleksinavoista saadaan siirtofunktion nimittäjäksi (karakteristiseksi polynomiksi):

$$\begin{aligned} & [s-(-\sigma+i\omega_{\mathrm{d}})]\,[s-(-\sigma-i\omega_{\mathrm{d}})] \\ = \;& [s+\sigma-i\omega_{\mathrm{d}}]\,[s+\sigma+i\omega_{\mathrm{d}}] \\ = \;& (s+\sigma)^2+i\omega_{\mathrm{d}}\,(s+\sigma)-i\omega_{\mathrm{d}}\,(s+\sigma)-i^2\omega_{\mathrm{d}}^2 \\ = \;& (s+\sigma)^2+\omega_{\mathrm{d}}^2 \end{aligned}$$

Usein siirtofunktio kirjoitetaan polynomimuotoon, jolloin nimittäjäksi tulee

$$s^2+2\omega_{\mathrm{n}}\zeta s+\omega_{\mathrm{n}}^2.$$

Vertaamalla muotoja keskenään voidaan kirjoittaa niiden väliset muunnoskaavat:

$$\sigma=\omega_{\mathrm{n}}\zeta,\;\omega_{\mathrm{d}}=\omega_{\mathrm{n}}\sqrt{1-\zeta^2}.$$

Parametria ζ kutsutaan *vaimennussuhteeksi*, ja ω_{n} on *vaimentamaton luonnollinen taajuus*. Kun vaimennussuhde on nolla $\zeta=0$, niin vaimennettu ja vaimentamaton luonnollinen taajuus ovat samat, $\omega_{\mathrm{d}}=\omega_{\mathrm{n}}$. Täysin vaimennetulle systeemille ($\zeta=1$) *vaimennettu luonnollinen taajuus* ω_{d} on nolla, $\omega_{\mathrm{d}}=0$. Toisen kertaluvun (nollattoman) systeemin impulssivasteeksi saadaan

$$h\,(t)=Ke^{-\sigma t}\sin\left(\omega_{\mathrm{d}}t\right)1\,(t)\,.$$

missä $K=\frac{\omega_{\mathrm{n}}}{\sqrt{1-\zeta^2}}$. Toisin sanoen systeemin aikavakio $\tau=\frac{1}{\sigma}$ määrää sen "vaimenemisnopeuden" ja systeemin värähtelyn taajuus määräytyy vaimennetusta luonnollisesta taajuudesta ω_{d}. Huomaa, että nämä parametrit löytyvät kätevästi systeemin navan reaali- ja imaginääriosien kertoimista. Vaimentamaton luonnollinen taajuus ω_{n} on etäisyys navan ja origon välillä, mikä nähdään ratkaisemalla muunnoskaavoista $\omega_{\mathrm{n}}^2=\omega_{\mathrm{d}}^2+\sigma^2$.

Kuva 1 esittää impulssivasteita (aikatasossa) useille systeemin navoille (s-tasossa).

2.4 Yhteenveto

- Keskeisiä termejä olivat: Control System Toolbox, lti-systeemi, lti-objekti.

- Polynomin juuret löytyvät komennolla `roots`, polynomit kerrotaan komennolla `conv`. Siirtofunktion osamurtokehitelmän saa komennolla `residue`.
- Jatkuva-aikaisen siirtofunktion lti-objekti määritellään `tf`-komennolla, jonka argumentteina ovat osoittaja ja nimittäjäpolynomit, tai toinen lti-objekti.
- Diskreettiaikaiselle systeemille annetaan argumenttina myös mittausväli.
- lti-objekteilla voi suorittaa yksinkertaista algebraa, tuloksena on uusi lti-objekti: `* + -`.
- lti-objekteilla voi suorittaa lohkokaavioalgebran peruskomentoja: `series`, `parallel`, `feedback`.
- lti-objektien aikavasteita voi simuloida `impulse` ja `step` -komennoilla. `ltiview`-komento avaa graafisen käyttöliittymän systeemin ominaisuuksien tarkastelua varten.

3 Simulink

3.1 Mikä Simulink on?

Simulink on ohjelmistopaketti joka tarjoaa *graafisen* käyttöliittymän *dynaamisten* systeemien mallintamista, simulointia ja analysointia varten.

Monilta osin valinta Simulinkin ja MATLABin välillä on mielipideasia, molemmilla voi tehdä samoja asioita, ja –itse asiassa– ne käyttävät monin osin ihan samoja funktioita. Allekirjoittanutkin karsastaa laskennan piilottamista graafisten mokkuloiden taakse, koska se samalla 'piilottaa' monet perusoperaatiot ja laskujärjestyksen. Toisaalta mallintaminen Simulinkillä näyttää ihastuttavan helpolta: senkun vetelee valmiiden lohkojen ikonit paikalleen, tarkistaa muutaman lohkon parametriarvot, ja painaa nappia... Asiaan on siis kuitenkin syytä perehtyä.

Simulinkissä mallit voidaan rakentaa hierarkkisina lohkokaavioina. Simulink sisältää mittavan lohkokaaviokirjaston, voit myös itse tehdä lohkoja lisää. Kun malli on määritelty, sitä voidaan simuloida. Haluttuja signaaleja voi seurata simuloinnin edetessä "on-line" tai vaikkapa siirtää MATLABiin myöhempää käsittelyä varten. Koska MATLAB ja Simulink on integroitu yhteen, kaikki MATLABin ja sen Toolboxien työkalut ovat koko ajan käytettävissä.

3.2 Dynaamisten systeemien mallintamisesta

Niinkuin lohkokaaviot yleensäkin, myös Simulinkin lohkokaaviot koostuvat lohkoista ja niiden välisistä kytkennöistä.

Lohko (*block*) koostuu sisäänmenoista, tiloista ja ulostuloista. Lohkon ulostulo on ajan, sisäänmenojen ja tilojen funktio. Lohkojen ulostulojen arvot ovat systeemin signaaleja. Jokaiseen lohkoon liittyy joukko ns. systeemifunktioita, jotka määrittävät sisäänmenojen, tilojen ja ulostulojen väliset aikariippuvuudet. Usein lohkot sisältävät parametrisoituja ominaisuuksia, viritettäviä parametriarvoja voi muuttaa myös simuloinnin aikana.

Monimutkainen systeemi voidan koota *hierarkkisesti* pienemmistä alisysteemeistä, joista jokainen on esitetty omana lohkokaavionaan.

Kukin Simulink malli määrittää sen jatkuvien tilojen aikaderivaatat, muttei itse tilojen arvoja. Systeemiä simuloitaessa Simulink *integroi* tilojen derivaattoja numeerisesti. Integrointi tapahtuu ns. ODE-ratkaisijoiden (Ordinary Differential Equation solver) avulla. Näiden kanssa voi joskus joutua pelaamaan, jotta löytäisi omaan ongelmaan sopivan ratkaisumenetelmän.

3.3 Mallien rakentaminen

Simulink käynnistetään MATLABin komentoikkunassa komennolla `simulink`. Simulinkin kirjastojen selain '*Library Browser*' avautuu, ja näet käytössäsi olevat Simulinkin lohkokirjastot. 'Help Navigator'in 'Simulink|Block Reference| Simulink Block Libraries' sisältää opasteet kaikista Simulinkin lohkoista. Perehdy ainakin seuraaviin:

- Continuous: 'Integrator', 'State-Space', 'Transfer Fcn', 'Transport Delay'
- Discrete: 'Discrete Transfer Fcn', 'Discrete State-Space', 'Unit Delay', 'Zero-Order Hold'
- Functions & Tables: 'Fcn', 'MATLAB Fcn'
- Math: 'Abs', 'Gain', 'Product', 'Sum'
- Nonlinear: 'Saturation'

- Signals and Systems: 'Demux', 'Mux', 'In1', 'Out1'
- Sinks: Scope, 'To Workspace'
- Sources: 'From Workspace', 'Sine Wave', 'Step'
- Control System Toolbox: 'LTI System'

Mallin rakentaminen aloitetaan valitsemalla valikosta 'File|New|Model', tai avaamalla jokin jo olemassa oleva malli. Mallit rakennetaan kopioimalla (hiirellä vetämällä) lohkoja kirjastoista mallin ikkunaan ja tekemällä (taas hiirellä vetämällä) kytkentöjä lohkojen välille.

Exercise 31 *Vedä lohkot 'Step' (Sources-kirjastosta), 'Transfer Fcn' (Continuous-kirjastosta) sekä 'Scope' (Sinks-kirjastosta) tyhjään malli-ikkunaan. Vedä lohkojen välille nuolet siten että ym. järjestys säilyy. Avaa 'Scope'-lohko kaksoisklikkaamalla sitä. Simuloi systeemiä valikkokomennolla 'Simulation|Start'.*

3.3.1 Hiiri- ja näppäimistökomennot

Simulinkin hallinta hiiren ja näppäimistön avulla on samankaltaista monien muiden Windows-ohjelmien kanssa. Yhteeveto hiiri- ja näppäikomennoista lohkojen, viivojen, signaalien nimien ja vapaiden tekstikenttien muokkaamiseksi löytyy 'Help Navigator'in kohdasta '*Simulink|Using Simulink|Creating a Model|Summary of Mouse and Keyboard Actions*' (huh!).

Esimerkiksi oikea hiirinäppäin tuo esille valitun lohkon pikavalikon, ja lohkojen ja kytkentöjen poistaminen tapahtuu **Delete**llä. Lohkon voi irroittaa kytkennöistään painamalla **Shift**, ja vetämällä lohko pois paikaltaan.

Mallilohkojesi ulkonäöksellisiä ominaisuuksia voit muokata valitsemalla lohkon ja tutkimalla '*Edit|Block Properties*' sekä '*Format*' valikon sisältöjä. Vastaavasti signaalin ulkonäköön vaikutetaan valitsemalla signaali ja '*Edit|Signal Properties*'.

Kytkennät lohkojen välillä vedetään ulostuloporteista sisääntuloportteihin. Kytkennän voi haaroittaa painamalla **Ctrl** ja vetämällä haaroituskohdasta uusi kytkentä. Summaus on perinteinen lohkokaavion signaalien yhdistämisoperaatio (Sum-lohko löytyy Math-kirjastosta), ja siitä saa vähennyslaskun muokkaamalla sen parametreja. Kahden lohkon väliseen kytkentään voi lisätä uuden vetämällä uusi lohko kytkennän päälle.

Rakennetun mallin 'syntaksin' voi tarkistaa valitsemalla '*Edit|Update Diagram*'. Tällöin Simulink tarkistaa lohkojen ja signaalien yhteensopivuudet, ja ilmoittaa mahdollisista ongelmista. Simulink tekee tämän myös automaattisesti aina ennen mallin simulointia.

3.3.2 Alisysteemien rakentaminen

Kun mallin koko kasvaa, voi lohkokaavioesitystä yksinkertaistaa yhdistelemällä valittuja lohkoja alisysteemeiksi. Tuloksena saadaan hierarkkinen systeemi missä alisysteemi on yhdellä tasolla, ja alisysteemin muodostavat lohkot toisella.

Alisysteemin rakentaminen tapahtuu valitsemalla mallista halutut lohkot ja suorittamalla '*Edit|Create Subsystem*'. Valittu osa mallistasi korvautuu Subsystem-lohkolla, ja Subsystem lohko sisältää valitut lohkot. Valitsemalla '*View|Model browser options|Model Browser*' pysyt kärryillä mallisi hierarkiasta.

Hyvä ohjelmointitapaa kannattaa Simulinkissäkin noudattaa. Tämä tarkoittaa sopivan hierarkian (vrt. MATLABin funktiorakenne) lisäksi signaalien ja lohkojen osuvaa nimeämistä sekä aputekstien käyttöä (vrt. muuttujien nimeäminen ja kommentointi M-tiedostoissa).

3.3.3 Dynaamisen systeemin mallinnusesimerkki

Simuloidaan P-säädöllä suljettua piiriä kun prosessia kuvaa siirtofunktio

$$\frac{Y(s)}{U(s)} = \frac{k_\mathrm{p}}{\tau s + 1},$$

missä $\tau = 10$ min, $k_\mathrm{p} = 5$ ja P-säätimen vahvistus $K_\mathrm{P} = \frac{2}{3}$.

Example 32 *Haetaan ensin tarvittavat lohkot malli-ikkunaan. Systeemi voidaan esittää käyttämällä:*

- *säätimelle Gain-lohkoa (Math-kirjastosta),*
- *prosessille Transfer Fcn-lohkoa (Continuous-kirjastosta),*
- *erosuureelle Sum-lohko (Math),*
- *sisäänmenoksi Discrete Pulse Generator (Sources) ja*
- *ulostulon piirtämiseen Scope-lohko (Sinks) avulla.*

Seuraavaksi ryhmitellään lohkot nätisti ja vedetään tarvittavat kytkennät. Siirtofunktion parametreiksi asetetaan 'Numerator' [5] ja 'Denominator' [10 1], Sum-lohkon parametreiksi 'list of signs' vaihdetaan `|+-`*, Gain lohkoon vahvistukseksi annetaan 10/(5*3), Discrete Pulse Generatorissa asetetaan 'Period' 20, 'Pulse width' 10 ja 'Sample time' 3.*

Valitaan vielä 'Simulation|Simulation Parameters' valikossa Solver-välilehden alta 'Stop time' 90. Nyt systeemiä voidaan simuloida valikosta 'Simulation|Start'.

3.4 Simulinkin i/o

Malli voidaan tallentaa ja lukea Simulinkkiin File-valikosta. Lohkokaavion voi tulostaa kuten minkä tahansa MATLABin kuvan, `print` komennolla. Simulointitulokset voi halutessaan siirtää MATLABin puolelle 'To Workspace' -lohkon avulla, ja käyttää MATLABin piirto-ominaisuuksia kuvien tuottamiseen.

3.5 Muuta mukavaa

- Simulinkin `linmod` ja `dlinmod` komentojen avulla saadaan graafisesta lohkokaaviomallista lineaarinen tilamallikuvaus, kunhan sisään- ja ulostulot on merkitty Inport ja Outport lohkoilla ('Signals & Systems' -kirjastosta). Tilamalli voidaan sitten muuntaa `ss`-komennolla edelleen lti-objektiksi. Tämä on kätevää, mutta vielä kätevämpää on se että funktiot osaavat *linearisoida* epälineaariset lohkokaaviomallit.
- `trim`-funktio määrittää lohkokaaviomallin tasapainotilan.
- Maskauksen (engl. *masking*) avulla voit muuttaa alisysteemin käyttöliittymää ja ikonia haluamaksesi.
- Simulinkin lohkokaaviomalleja voi ajaa suoraan MATLABin komentoikkunasta `sim`-komennolla.

3.6 Yhteenveto

- Keskeisiä termejä olivat: Simulink, lohkokaavio, lohko, signaali, ODE.
- Simulink tarjoaa graafisen käyttöliittymän dynaamisten systeemien mallintamiseen.
- Simulinkin mallit rakennetaan kopioimalla lohkoja lohkokaaviokirjastosta, ja yhdistelemällä niitä toisiinsa.

4 LTI-säätösuunnittelu

Tässä luvussa jatketaan Control System Toolboxiin (CST) perehtymistä kompensaattorien suunnittelulla. Kompensaattori on säätösysteemiin lisätty komponentti (esim. PID-säädin tai alipäästösuodin) jonka avulla paikataan ("kompensoidaan") puutteellista käyttäytymistä. Säätö*suunnittelu* taas viittaa kompensaattorin parametrien valintaan siten että suljetulle piirille halutut käyttäytymistavoitteet täyttyvät.

Porttina CST-säätösuunnitteluun on graafinen käyttöliittymä, ns. '*SISO Design Tool*'. Sen avulla voidaan...

- manipuloida suljetun piirin dynamiikkaa juuriuratekniikoiden avulla,
- muokata avoimen piirin Bode-kuvaajia,
- muokata vaihe- ja vahvistusvaroja,
- lisätä kompensaattoriin napoja ja nollia,
- lisätä ja virittää johto/jättöpiirejä (engl. *lead*/*lag*), sekä
- tutkia suljetun piirin vasteita ('LTI Viewerin' avulla).

Nämä kaikki kuuluvat 'säätö- ja systeemitekniikan perusteet' -kurssin sisältöön. Ei varmastikaan olisi pahitteeksi kaivaa vanha luentomateriaali tai joku monista hyvistä oppikirjoista esille, ja muistella mistä näissä menetelmissä on kyse.

Koska kukaan ei kuitenkaan tee niin, niin ohessa pikainen muistutus asioista...

4.1 Menetelmien kertausta

Kertausjakson materiaali on pääosin valikoitu ja suomennettu Dorfin oppikirjasta, luvut 5 ja 7–10. Jakson materiaali on varsin tiivistä ja pedagogisesti parempaa oppimateriaalia kannattaakin hakea toisaalta.

4.1.1 Säätösysteemin käyttäytyminen s-tasossa

Takaisinkytketyn systeemin suoriutumista voidaan kuvata sen karakteristisen yhtälön juurien sijannilla s-tasossa (eli Laplace-tasossa).

Example 33 *Prosessin*

$$\frac{\omega_n^2}{s\left(s+2\varsigma\omega_n\right)}$$

negatiivisesti takaisinkytketyn suljetun piirin siirtofunktio on toisen kertaluvun systeemi:

$$\frac{\omega_n^2}{s^2+2\varsigma\omega_n s+\omega_n^2}.$$

Perinteiset aikatason kriteerit Säädön suunnittelun (napojen asettamisen) kannalta napaparin ja askelvasteen perinteisten mittareiden ($\frac{\%}{100}$ *ylitys* M_p, 10% – 90% *nousuaika* t_r ja ±1% *asettumisaika* t_s) väliset yhteydet ovat käteviä käyttää:

$$\begin{aligned} M_\mathrm{p} &\approx e^{\frac{-\pi\varsigma}{\sqrt{1-\varsigma^2}}}, \\ t_\mathrm{r} &\approx \frac{2.16\varsigma+0.60}{\omega_\mathrm{n}} \approx \frac{1.8}{\omega_\mathrm{n}}, \text{ (kun } 0.3 \le \varsigma \le 0.8) \\ t_\mathrm{s} &\approx \frac{4.6}{\omega_\mathrm{n}\zeta}. \end{aligned}$$

Toisen kertaluvun systeemin taajuuskaistalle[4] saadaan approksimaatio $\omega_B = (-1.196\varsigma + 1.85)\,\omega_n$ (kun $0.3 \leq \varsigma \leq 0.8$). Ensimmäisen kertaluvun systeemin taajuuskaista on $\omega_B \approx \frac{1}{\tau}$. Taajuuden yksikkönä näissä käytetään radiaania per sekunti, $\frac{\text{rad}}{\text{s}}$ ($2\pi\frac{\text{rad}}{\text{s}} = 1\text{Hz}$).

Optimipolynomit Yhteyksiä napoihin löytyy myös ns. optimipolynomien kautta. Esimerkiksi askelvasteen *ITAE* (*Integral Time Absolute Error*) -kriteeri

$$ITAE = \int_0^T t\,|e(t)|\,dt$$

minimoituu seuraavilla karakteristilla polynomeilla (nollattomalle n:nnen asteen suljetun piirin siirtofunktiolle, $n = 1$, 2 ja 3):

$$s + \omega_n$$
$$s^2 + 1.4\omega_n s + \omega_n^2$$
$$s^3 + 1.75\omega_n s^2 + 2.15\omega_n^2 s + \omega_n^3.$$

4.1.2 Juuriura (*root locus*)

Takaisinkytketyn systeemin vastetta voidaan muokata muuttamalla yhtä (tai useampaa) systeemin parametrin arvoa. Tällöin on kätevää määrittää se kuinka karakteristisen yhtälön juuret liikkuvat s-tasossa, jonkin parametrin funktiona. Juuriurakuvaaja on graafinen piirros joka muodostuu karakteristisen yhtälön ratkaisuista (eli karakteristisen polynomin juurista, eli suljetun piirin navoista), kun parametrin arvo muuttuu.

Juuriuramuoto Oletetaan että avointa piiriä kuvaa siirtofunktio $G(s)$, ja että se on negatiivisesti takaisinkytketty vahvistuksella K. Suljetun piirin siirtofunktioksi saadaan

$$\frac{Y(s)}{W(s)} = \frac{KG(s)}{1 + KG(s)}.$$

Example 34 *P-säädetylle prosessille avoin piiri $G(s)$ vastaa prosessin mallia, ja vahvistus K vastaa säätimen vahvistusta k_P.*

Example 35 *Oletetaan että prosessia kuvataan siirtofunktiolla*

$$P(s) = \frac{k_p}{s(s+a)}$$

ja että se on negatiivisesti takaisinkytketty vahvistuksella k_P (huomaa alaindeksit: pieni p prosessille ja iso P P-säätimelle). Jos P-säätimen vahvistuksen arvo k_P ajatellaan vakioksi, voidaan parametrin a vaikutusta suljetun piirin vasteeseen tutkia juuriuran avulla, kirjoittamalla karakteristinen yhtälö sopivaan muotoon:

$$1 + KG(s) = 1 + k_P P(s) = 1 + k_P \frac{k_p}{s(s+a)} = 0$$
$$\Leftrightarrow\ 1 + \overbrace{a}^{K}\ \frac{\overbrace{\frac{1}{s}}^{G(s)}}{s^2 + k_P k_p} = 0$$

Yhtälö on nyt "juuriuramuodossa" $1 + KG(s) = 0$ missä K:ta vastaa parametri a ja $G(s)$:ää siirtofunktio $\frac{s}{s^2+k_P k_p}$.

[4] Systeemin *taajuuskaista* ω_B (engl. *bandwidth*) on taajuus jolla taajuusvaste tippuu alle 3 dB sen matalien taajuuksien arvosta. Taajuuskaista kertoo systeemin transienttivasteen nopeudesta ja sen kyvystä seurata sisäänmenosignaalia.

Karakteristinen yhtälö $1 + KG(s)$ voidaan kirjoittaa myös polaarimuodossa

$$|KG(s)| \angle KG(s) = -1 + i0$$

eli karakteristisen yhtälön ratkaisuksi saadaan

$$|KG(s)| = 1 \text{ ja } \angle KG(s) = -180^\circ \pm k360^\circ.$$

Juuriuran avulla suunnittelu *Juuriura* on polku jonka (suljetun piirin) karakteristisen yhtälön juuret kulkevat s-tasossa, kun systeemin parameteria (K) muutetaan (K saa arvoja nollasta äärettömään).

Juuriuran ominaisuuksista mainittakoon...

- juuriura on symmetrinen vaaka-akselin (reaaliakselin) suhteen[5],
- juuriura alkaa avoimen piirin navoista ja päättyy sen nolliin[6].

Juuriurasuunnittelun tavoitteena on siis hakea sopiva parametrin K arvo, siten että suljetun piirin navat siirtyvät haluttuun paikkaan.

4.1.3 Taajuusvaste (*frequency response*)

Systeemin taajuusvaste määritellään systeemin vasteena sinimuotoiseen sisäänmenosignaaliin (tasapainotilassa, eli kun kaikki transienttivasteet ovat hiipuneet). LTI-systeemin vaste sinisignaaliin on toinen sinisignaali. Ulostulo eroaa sisäänmenosta ainoastaan vahvistuksen ja vaiheen osalta.

Example 36 *LTI-systeemin $T(s)$ vaste y sinisignaalille r*

$$r(t) = A \sin \omega t$$

on

$$y(t) = A |T(s)| \sin(\omega t + \phi)$$

missä $\phi = \angle T(s)$ ja kun $t \to \infty$.

Bode-kuvaaja Taajuusvastemenetelmissä tutkitaan siirtofunktiota $G(s)$ kun $s = i\omega$, ja kehitetään graafisia menetelmiä kompleksiluvun $G(i\omega)$ esittämiseksi taajuuden ω muuttuessa.

Systeemin siirtofunktio voidaan esittää taajuustasossa sen reaaliosan ja imaginääriosan avulla: $G(i\omega) = R(\omega) + iX(\omega)$. Vaihtoehtoisesti se voidaan esittää vahvistuksen ja vaiheen avulla $G(i\omega) = |G(\omega)| e^{i\phi(i\omega)} = |G(\omega)| \angle\phi(\omega)$. Graafinen esitys yksinkertaistuu esitettäessä vahvistus logaritmisella asteikolla:

$$\text{logaritminen vahvistus} = 20 \log_{10} |G(\omega)| \Leftrightarrow |G(\omega)| = 10^{\frac{\text{logaritminen vahvistus}}{20}}.$$

Yksiköt ovat desibelejä (dB). *Bode*-kuvaajassa esitetään systeemin logaritminen vahvistus ja sen vaihe ϕ taajuuden ω logaritmin funktiona.

[5] koska kompleksiset juuret esiintyvät aina pareittain.

[6] $G(s)$ voidaan kirjoittaa napa-nolla-vahvistus-muodossa jolloin karakteristiseksi yhtälöksi saadaan

$$1 + K \overbrace{\frac{\prod_i (s + z_i)}{\prod_j (s + p_j)}}^{G(s)} = 0$$

ja edelleen

$$\prod_j (s + p_j) + K \prod_i (s + z_i) = 0.$$

Nyt nähdään että jos $K = 0$, niin karakteristisen yhtälön juuret ovat samat kuin avoimen piirin $G(s)$ navat. Vastaavasti kun $K \to \infty$ niin juuret lähestyvät nollia.

Bode-kuvaajien piirtämissääntöjä Bode-kuvaajat noudattavat muutamaa helposti havainnollistettavaa sääntöä:

- Vakion K logaritminen vahvistus on vakio $20\log_{10}|K|$ dB.
  Positiivisen vahvistuksen vaihe on $0°$, negatiivisen $-180° = -\pi$.
- Reaalisen navan $\frac{1}{\tau}$ leikkaustaajuus (*break frequency*) on $\omega = \frac{1}{\tau}$. Asymptoottien $\omega \to 0$ ja $\omega \to \infty$ johtaminen on suoraviivaista:

$$\lim_{\omega\to 0} \frac{1}{1+i\tau\omega} = 1 \Rightarrow \begin{cases} |1| = 0 \text{ dB} \\ \arg(1) = 0° \end{cases}$$

$$\lim_{\omega\to\infty} \frac{1}{1+i\tau\omega} = \lim_{\omega\to\infty} \frac{1}{i\tau\omega} = \lim_{\omega\to\infty} \frac{-i}{\tau\omega}$$

$$\Rightarrow \begin{cases} \left|\frac{-i}{\tau\omega}\right| = \frac{1}{\tau\omega} = 20\log_{10}\frac{\omega^{-1}}{\tau} \text{ dB} = 20\log_{10}\frac{1}{\tau} \text{ dB} - 20\log_{10}\omega \text{ dB} \\ \arg\left(\frac{-i}{\tau\omega}\right) = -90° \end{cases}$$

  Ts. matalilla taajuuksilla alipäästösuodin lähestyy vakiota 1: Leikkaustaajuuden vasemmalla puolella voidaan approksimoida vahvistukseksi 0 desibeliä ja vaiheeksi 0 astetta. Korkeilla taajuuksilla suodin lähestyy negatiivista imaginääriakselia: Leikkaustaajuuden oikealla puolella vahvistus pienenee kääntäen verrannollisena taajuuteen: logartimisessa asteikossa kulmakertoimella -20 dB/dekadi.
  Vastavasti reaaliselle nollalle saadaan:

$$\lim_{\omega\to 0} 1 + i\tau\omega = 1 \Rightarrow \begin{cases} |1| = 0 \text{ dB} \\ \arg(1) = 0° \end{cases}$$

$$\lim_{\omega\to\infty} 1 + i\tau\omega = \lim_{\omega\to\infty} i\tau\omega$$

$$\Rightarrow \begin{cases} |i\tau\omega| = \tau\omega = 20\log_{10}\tau\omega \text{ dB} = 20\log_{10}\tau + 20\log_{10}\omega \text{ dB} \\ \arg(i\tau\omega) = 90° \end{cases}$$

  Ts. nollan vahvistus leikkaustaajuuden vasemmalla puolella on 1 (0 dB), ja oikealla puolella se kasvaa (+20 dB/dekadi).
  Kannattaa muistaa että leikkaustaajuuden $\frac{1}{\tau}$ kohdalla approksimoinneissa on klappia.

- Bode-kuvaajan logaritmisuuksien ansiosta kertolasku muuttuu yhteenlaskuksi. Tämä on mukavaa, koska siirtofunktiomuodossa keskenään kerrottavien termien vahvistus ja vaihe voidaan päätellä summaamalla yksittäisten termien vaikutukset Bode-kuvaajassa.

Vaihe- ja vahvistusvara Säätösuunnittelussa on tietysti tärkeää että suunniteltava systeemi on stabiili, vaikka systeemin malleissa olisikin pientä klappia, tai systeemeihin vaikuttaisi tuntemattomia häiriöitä. Avoimen piirin Bode-kuvaajasta voidaan helposti määrittää kaksi keskeistä suuretta jotka kertovat systeemin suhteellisesta stabiiliudesta, eli siitä "marginaalista" joka niillä on epästabiiliin käyttäytymiseen.

Muistetaan, että kun systeemin navat siirtyvät s-tason oikealle puolelle, systeemistä tulee epästabiili. Muistetaan edelleen, kuinka negatiivisesti takaisinkytketyn suljetun piirin karakteristisen yhtälön ratkaisu voidaan esittää muodossa

$$|KG(s)| = 1 \text{ ja } \angle KG(s) = -180° \pm k360°,$$

missä $KG(s)$ on avoimen piirin siirtofunktio. Toisin sanoen, voimme tutkia suljetun piirin stabiilisuusominaisuuksia avoimen piirin vahvistuksen ja vaiheen avulla. Stabiilisuuden kannalta kriittinen piste on $-1 + i0$, eli Bode-kuvaajassa vahvistus 0 dB ja vaihe $-180°$.

- *vahvistusvara* (engl. *gain margin*) on kasvuvara systeemin vahvistuksessa, sen ollessa $-180°$ vaiheessa, jonka yli mentäessä systeemistä tulee epästabiili.
  Vahvistusvara [dB] saadaan katsomalla Bode-kuvaajasta taajuus jolla vaihe on $-180°$, lukemalla vastaava vahvistus [dB], ja vähentämällä se nollasta [dB].
- *vaihevara* (engl. *phase margin*) on viivästysvara systeemin vaiheessa, sen vahvistuksen ollessa 1, jonka yli mentäessä systeemistä tulee epästabiili.
  Vaihevara saadaan katsomalla Bode-kuvaajasta taajuus jolla vahvistus on 0 dB, lukemalla vastaava vaihe, ja vähentämällä siitä $-180°$.

Taajuustason säätösuunnittelussa pyritään riittäviin vahvistus- ja vaihevaroihin. Toisen kertaluvun mallin vaimennussuhteen ς ja vaihevaran ϕ_{pm} [°] välillä on likimääräinen yhteys (kun $\varsigma \leq 0.7$)

$$\varsigma = 0.01\phi_{\text{pm}}.$$

4.1.4 Kompensaattorien suunnittelu

Pelkkää vahvistusta muuttamalla voi napoja siirtää vain rajoitetusti, eli valita juuriuralta parhaimmaksi katsomansa pisteen. Lisäämällä kompensaattoriin napoja ja/tai nollia saadaan lisää vapausasteita (ja lisää parametreja). Myötähaaraan liitettyä kompensaattoria kutsutaan kaskadikompensaattoriksi.

Kompensaattorisuunnittelua voidaan lähestyä asettamalla kompensaattorin vahvistus tasapainotilan vaatimukset täyttäväksi, ja virittämällä sitten kompensaattorin navat ja/tai nollat siten että systeemin dynamiikka miellyttää.

Käsitellään ensimmäisen asteen kompensaattoria

$$C(s) = \frac{K(s+z)}{s+p},$$

missä K, z ja p tulisi valita. Kun $|z| < |p|$, systeemiä kutsutaan *vaiheenjohtokompensaattoriksi*. Jos $z \ll |p|$ ja z olisi nollassa, ei navalla olisi merkitystä ja tuloksena olisi derivaattori

$$C(s) \approx \left(\frac{K}{p}\right) s$$

Derivaattorin vaihekulma on $+90°$. Vaiheenjohtokompensaattorin taajuusvasteen leikkaustaajuus on $\sqrt{zp}$. Sen vasemmalla puolella (z:aan asti) vahvistus on nolla [dB], ja nousee sen oikealla puolella (p:hen asti). Vaihekuvaajassa on piikki leikkaustaajuuden kohdalla, mistä nimitys vaiheenjohto.

Vaiheenjättökompensaattorin ($|p| < |z|$) vahvistus- ja vaihekuvaajat ovat päinvastaiset (vahvistus pienenee suurilla taajuuksilla, vaiheessa on pudotus leikkaustaajuuden kohdalla).

Bode Bode-kuvaajaa käyttö vaiheenjohtokompensaattorien suunnittelussa on kätevää, koska kompensaattorin taajuusvaste voidaan yksinkertaisesti lisätä kompensoimattoman systeemin taajuusvasteeseen. Tämä summausominaisuus on Bode-kuvaajien suosion keskeinen syy.

Piirretään ensin avoimen piirin $G(s)$ Bode, jonka jälkeen määrätään sopivat paikat z:lle ja p:lle taajuusvasteen muokkaamiseksi. Napa p sijoitetaan hieman systeemin oikeanpuoleisimman navan oikealle puolelle (ettei ainakaan hidasteta systeemiä), nolla z vastaavasti hitaimman navan paikkeille (kumotaan se). Vahvistus K määrätään siedetyn tasapainotilan virheen perusteella. Tämän jälkeen tarkistetaan ovatko vaihevara ja muut systeemin ominaisuudet riittävät, ja tarvittaessa jatketaan iterointia.

Juuriura Juuriuramenetelmässä tulkitaan ensin säädölle asetetut tavoitteet haluttujen napojen sijaintina. Vaiheenjohtokompensaattorin nolla sijoitetaan välittömästi halutun juurien sijaintipaikan vasemmalle puolelle. Napa sijoitetaan tämän vasemmalle puolelle siten, että suljetun piirin napa tulee haluttuun paikkaan. Tämän jälkeen tarkistetaan vahvistus (tasapainotilan virhe). Tarvittaessa iteroidaan.

4.1.5 Napojensijoittelu

Napojensijoittelu (tai sekä napojen että nollien sijoittelu) on suoraviivainen tapa muokata systeemin vastetta, jossa säädin yksinkertaisesti ratkaistaan siten että suljetun piirin navat *sijoittuvat* haluttuun paikkaan.

Kirjoittajan mielestä napojensijoittelun esittäminen on intuitiivisintä käsiteltäessä suljetun piirin polynomeja. Koska Control System Toolbox kuitenkin tukee tilamalliesitystä sen suoraviivaisen monimuuttujasoveltuvuuden takia, käydään seuraavassa lyhyesti tilamallien napojensijoittelu.

Oletetaan, että prosessia

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned}$$

säädetään tilatakaisinkytkennällä

$$\mathbf{u} = -\mathbf{K}\mathbf{x}.$$

Suljetulle piirille saadaan

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\mathbf{K})\,\mathbf{x},$$

jolloin suljetun piirin navat ovat $\mathbf{A} - \mathbf{B}\mathbf{K}$:n ominaisarvoja (`eig`). Karakteristinen yhtälö kirjoittuu (symbolisesti)

$$\det\left(\lambda\mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K})\right) = 0,$$

josta $\mathbf{K}$:n arvot voidaan ratkaista (käsin) kun haluttu käyttäytyminen esitetään polynomimuodossa $\lambda^n + a_1\lambda^{n-1} + ... + a_n$ ja vaaditaan

$$\det\left(\lambda\mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K})\right) = \lambda^n + a_1\lambda^{n-1} + ... + a_n.$$

Käytännössä käytetään numeerisia menetelmiä vahvistusmatriisin parametrien määrittämiseksi.

4.2 MATLABin SISO-suunnittelutyökalu

SISO Design Tool avautuu komennolla `sisotool`, johon annetaan argumenttina prosessimallin lti-objekti.

Example 37 *`sisotool(sys_p)`*

SISO Design Tool näyttää suljetun piirin juuriuran, sekä avoimen piirin Bode-kuvaajan vahvistus- ja vaihevaroineen. Tools-valikosta voit avata 'LTI Viewer'in suljetulle piirille, jonka avulla voit mm. tarkastella suljetun piirin aikatason vasteita.

'SISO Design Tool'in työkalut löytyvät myös suoraan komentorivitason komentoina: `bode`, `margin`, `rlocus`, `rlocfind`, `pzmap`, `nyquist`.

4.2.1 Suunnittelu Bode-kuvaajan avulla

Kompensaattoreita voidaan suunnitella avoimen piirin Bode-kuvaajien avulla. Suunnittelun välittömänä kohteena ovat vahvistus- ja vaihevarat, taajuuskaista, jne.

Oletusarvoisesti kompensaattori on pelkkä vahvistus K ($K = 1$). Yksinkertaisin tapa systeemin vasteen nopeuttamiseksi on vahvistuksen kasvattaminen. Vahvistuksen muuttaminen onnistuu tarttumalla Bode-kuvaajan (oikealla) vahvistuskäyrään (ylempi kuvaaja) ja siirtämällä sitä ylöspäin. Kulloinenkin vahvistus näkyy kompensaattorin kaavassa (vasen yläkulma), jonne voi myös näpytellä uuden numeroarvon. Systeemin aikavasteita voi tutkia käynnistämällä 'LTI Viewer'in valikosta 'Tools|Loop Responses'. Vahvistus- ja vaihevarat näkyvät Bode-kuvaajissa (vasen alanurkka).

Tasapainotilan virheen poistamiseksi voidaan systeemiin lisätä integraattori pikavalikon 'Add|Integrator' komennolla. Vaiheenjohtokompensaattorin lisääminen löytyy myös pikavalikosta: 'Add|Lead'. Napoja ja nollia voi siirtää tarttumalla niihin ja vetämällä ne haluttuun paikkaan.

4.2.2 Suunnittelu juuriuran avulla

Voit tutkia systeemin vastetta muuttamalla kompensaattorin vahvistusta K juuriurakuvaajassa (vasemmalla). Voit muuttaa vahvistuksen lukuarvoa suoraan (vasen yläkulma), tai nappaamalla punaisista neliöistä ja siirtämällä niitä juuriuraa pitkin. Vastaavat muutokset näkyvät myös Bode-kuvaajissa (oikealla) ja aikavastekuvaajissa. Voit nyt hakea sopivan vahvistuksen arvon interaktiivisesti.

Jos vaste ei tyydytä, voi kompensaattoriin lisätä navan tai nollan. Lisääminen tapahtuu esim. pikavalikon komennolla 'Add|Real Pole' (pikavalikon saat esille klikkaamalla hiiren oikeaa näppäintä juuriurakuvaajan päällä). Klikkaamalla juuriurakuvaajaa määrät kohdan johon napa lisätään. Nollan lisääminen tapahtuu vastaavalla tavalla. Pikavalikon 'Delete Pole/Zero' komennolla voit poistaa kompensaattorin navan tai nollan.

Suljetun piirin vasteen voi nyt suunnitella interaktiivisesti muuttamalla kompensaattorin vahvistusta, sen napojen ja nollien paikkoja, tai määrää. Pikavalikon 'Design Constraints' editorin avulla voit hahmottaa kuinka aikatason kriteerit (asettumisaika, ylitys, vaimennussuhde, luonnollinen taajuus) linkittyvät s-tasoon.

Suunniteltu kompensaattori voidaan palauttaa MATLABin työtilaan 'File|Export' ikkunan avulla. Klikkaamalla kompensaattoria listassa ja valitsemalla 'Export to Workspace' kompensaattori palautetaan lti-objektina työtilaan.

4.2.3 Suunnittelu napojensijoittelun avulla

MATLABin Control System Toolbox sisältää komennon `place`, joka laskee tilatakaisinkytkennän vahvistusmatriisin $\mathbf{K}$ joka siirtää suljetun piirin navat haluttuun paikkaan (kunhan systeemi $\mathbf{A}, \mathbf{B}$ on säädettävä, `ctrb`).

4.3 Yhteenveto

- Keskeisiä termejä olivat: kompensaattori, karakteristinen yhtälö, vaihevara, vahvistusvara, vaiheenjohtokompensaattori.
- Juuriura on polku jonka suljetun piirin navat kulkevat s-tasossa kun vahvistus muuttuu nollasta äärettömään. Juuriura piirretään komennolla `rlocus`.

- Bode-kuvaajassa avoimen piirin (sinimuotoiseen sisäänmenosignaaliin aiheutuva) vahvistus ja vaiheensiirto (ulostulossa) esitetään logaritmisella asteikolla sisäänmenotaajuuden funktiona. Bode-kuvaaja piirretään komennolla `bode`, kuvaajan vaihe- ja vahvistusvaroilla saa komennolla `margin`.
- `sisotool`-komento avaa graafisen käyttöliittymän säätösuunnittelupakettiin.
- säätöpiirin lohkot kuvataan lti-objekteina.

5 Simulointi, optimointi, identifiointi, ...

Tässä vaiheessa kurssia pitäisi lukijalla jo olla auttava rutiini MATLABin itsenäiseen käyttämiseen. Niinpä seuraavat asiat jätetään pitkälti itsetutkiskelun varaan, haluanpahan vain huomauttaa että näissäkin tavallisissa tarpeissa MATLAB on varsin kätevä ja hyödyllinen.

5.1 Differentiaaliyhtälöiden simulointi

Simulinkin simuloinneissa ovat ODE-yhtälöiden integrointialgoritmit keskeisessä asemassa. ODE-yhtälöitä voi ratkaista myös suoraan MATLABissa.

Ratkaistavat yhtälöt kirjoitetaan M-tiedostoon (funktioon), jonka ulostuloargumenttina on tilan muutos $\dot{\mathbf{x}}$, ja sisäänmenoargumentteina aika t ja tila $\mathbf{x}$. Tiedosto sisältää siis koodina differentiaaliyhtälöt:

$$\begin{aligned} \dot{x_1} &= f(\mathbf{x}, \cdot) \\ \dot{x_2} &= f(\mathbf{x}, \cdot) \\ &\vdots \\ \dot{x_n} &= f(\mathbf{x}, \cdot) \end{aligned}$$

missä $\dot{x_1}$, $\dot{x_2}$ jne. ovat vektorin $\dot{\mathbf{x}}$ komponentteja. Hyvä alkuarvaus ratkaisualgoritmiksi on `ode23`.

Säätöongelmissa halutaan usein myös ohjata sämplättyä prosessia. Differentiaaliyhtälöiden ratkaisuun saa ujutettua ohjauksen ylimääräisenä argumenttina (kts. `help ode23`), sämpläyksen saa mukaan esim. simuloimalla prosessia aina yhden askeleen verran kerrallaan ja tallentamalla aikaintervallin lopputuloksen.

'Help Navigator'in 'MATLAB|Using MATLAB|Mathematics|Differential Equations' kertoo laajemmin differentiaaliyhtälöiden ratkaisemisesta.

5.2 Optimointi

Pienimmän neliösumman menetelmän mukaan lineaarisen mallin $\mathbf{Y} = \mathbf{X}\boldsymbol{\theta}^T$ parametriestimaatti lasketaan kaavalla

$$\widehat{\boldsymbol{\theta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

missä $\mathbf{X}$ sisältää sisäänmenodatan sarakkeittain, ja $\mathbf{Y}$ on ulostulot sisältävä pystyvektori. MATLABilla laskennan voi toteuttaa myös jakamalla vasemmalle: $\boldsymbol{\theta} = \mathbf{X}\backslash\mathbf{Y}$.

Epälineaaristen systeemien parametrien estimoinnissa iteratiiviset algoritmit ovat hallitsevia. Pienimmän neliösumman minimointi

$$\widehat{\theta} = \arg\min_{\theta} \sum f(\theta)^2$$

tapahtuu MATLABissa funktiolla `lsqnonlin`. Minimoinnin perustana olevan funktion $f(\theta)$ arvot tuotetaan M-tiedostossa (funktio). Funktion nimi annetaan `lsqnonlin`in argumenttina. Algoritmin optiot sisältävät mm. mainion Levenberg–Marquardt algoritmin.

'Help Navigator'in 'MATLAB|Using MATLAB|Mathematics|Function Functions|Minimizing Functions and Finding Zeros' kertoo laajemmin optimointiongelmien ratkaisemisesta.

5.3 Identifiointi

Levenberg–Marquardt on usein käytetty optimointialgoritmi epälineaaristen systeemien identifioinnin tapauksessa. Lineaaristen aikasarjamallien tapauksessa pääsee helpommalla perehtymällä MATLABin Systems Identification Toolboxiin.

Tyypillisiä säätötekniikassa käytettyjä dynaamisia diskreettiaikaisia mallirakenteita ovat mm. *ARX* (*AutoRegressive eXogenous*) ja *OE* (*Output Error*):

$$\begin{aligned} A\left(q^{-1}\right) y\left(k\right) &= B\left(q^{-1}\right) u\left(k-d\right) + e\left(k\right) \\ y\left(k\right) &= \frac{B\left(q^{-1}\right)}{A\left(q^{-1}\right)} u\left(k-d\right) + e\left(k\right) \end{aligned}$$

missä y ja u ovat systeemin ulostulo- ja sisäänmenosignaalit, A ja B ovat diskreettiaikaisen systeemin polynomeja (taaksepäinsiirto-operaattorin q^{-1} funktiona, $A\left(q^{-1}\right) = 1 + a_1 q^{-1} + ... + a_m q^{-m}$ ja $B\left(q^{-1}\right) = b_0 + b_1 q^{-1} + ... + b_n q^{-n}$) ja d on viive. Esim. ARX-malli voidaan kirjottaa differenssimuodossa:

$$\begin{aligned} y\left(t\right) &= a_1 y\left(t-T\right) + a_2 y\left(t-2T\right) + ... + a_n y\left(t-nT\right) \\ &\quad + b_0 u\left(t\right) + b_1 u\left(t-T\right) + ... + b_m u\left(t-mT\right) + e\left(t\right) \end{aligned}$$

missä T on mittausväli. Signaali e on valkoista kohinaa, eli mallit kuvaavat *stokastisisia* systeemejä.

Näitä mallirakenteita voi identifoida komennoilla `arx` ja `oe`, rekursiivisia versioita ovat `rarx` ja `roe`. Komennot tuottavat lti-objekteja, joita voi näppärästi jatkokäsitellä Control System Toolboxin työkaluilla. CST:n `sisotool`in tapaan tarjolla on myös graafinen käyttöliittymä, komennolla `ident`.

5.4 Harvat matriisit

Suurikokoisten matriisien kanssa pelattaessa tahtoo MATLABista (tietokoneesta) loppua muistitila ja tehotkin kesken. *Harvat matriisit* (engl. *sparse matrices*) ovat suuria matriiseja jotka ovat kuitenkin enimmäkseen tyhjiä, eli sisältävät vain pienen määrän ei-nollia elementtejä. Säätötekniikassa tällaisiin törmää vaikkapa diskreettien Markovin ketjujen yhteydessä.

Harvoja matriiseja voidaan MATLABissa käsitellä tavallisia matriiseja tehokkaammalla tavalla, kunhan matriisi ensin määritellään harvaksi (`sparse`). 'Help Navigator'in MATLAB|Using MATLAB|Mathematics|Sparse Matrices' kertoo enemmän aiheesta.

5.5 Yhteenveto

- Keskeisiä termejä olivat: ODE, System Identification Toolbox, ARX, OE.
- MATLABia voi käyttää mm seuraavien osa-alueiden ongelmien tehokkaassa ratkaisemisessa: differentiaaliyhtälöiden simulointi, optimointitehtävät, ja mallien rakentaminen mittausdatasta (identifiointi).