E-kursuse "Andmeanalüüs ja arvutused MATLAB-iga" materjalid

Aine maht 6 EAP

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

ÕPPEAINE ÜLDANDMED

1	ÕPPEAINE KOOD ja NIMETUS (eesti ja inglise keeles)	Andmeanalüüs ja arvutused MATLAB-iga *Data Analysis and Computational Methods with MATLAB*
2	STRUKTUURIÜKSUS	Tartu Ülikooli tehnoloogiainstituut (LOTI)
3	KOGUMAHT (EAP)	6
4	KESTUS SEMESTRITES	1
5	ÕPPEAINE ON VÕTA RAAMES ASENDATAV *(Jah/ei)*	
6	ÕPPEAINE ÜLDEESMÄRGID (eesti ja inglise keeles)	Õpetada üliõpilastele andmetöötluskeskkonna MATLAB kasutamist tüüpiliste arvutus- ja modelleerimisülesannete lahendamiseks ja tulemuste piltlikuks esitamiseks. The main goal is to teach students the use of MATLAB, the environment of technical computing to solve typical modelling and computational problems, also the visualization of results.
7	ÕPPEAINE ÕPIVÄLJUNDID (eesti ja inglise keeles)	Kursuse läbinud üliõpilased - omavad ülevaadet MATLABi keskkonna ja keele võimalustest; - oskavad töötada massiividega; - tunnevad andmetüüpe ja oskavad kasutada MATLABi andmevahetuse ning –töötluse võimalusi; - omavad arvutile programmeerimise kogemust ning oskust koostada, vormistada ja kasutada M-faile; - tunnevad ja oskavad kasutada MATLABi graafika- ja visualiseerimisvahendeid; - oskavad otsida ja kasutada veebis leiduvaid MATLABi programme ja abivahendeid. The students who pass the course have following skills. - Have overview of the possibilities of MATLAB environment and programming language. - Know how to work with arrays. - Have knowledge about data types and classes and have experience of data exchange, processing and management. - Can write and use different m-files and have passed a first introduction to programming. - Have knowledge and experience in the use of graphical and visualization features of MATLAB. - Know how to use the means available through internet.
8	ÕPPEAINE LÕPPHINDAMINE	Eristav hindamine.

9	SISU LÜHIKIRJELDUS (eesti ja inglise keeles)	Tutvumine rakendustarkvaraga MATLAB. Massiivid. Andmetüübid. Graafika. Andmevahetus. Võrrandid ja võrrandisüsteemid ning nende lahendamine. Funktsioonide lähendamine. Algteadmised harilikest ja osatuletistega diferentsiaalvõrranditest ning nende lahendamise vahenditest. Matemaatiliste mudelite viimine arvutisse keskkonnas MATLAB. Paketi lisad (Toolbox). Simulatsioonid ja numbrilised eksperimendid. Graafiline kasutajaliides. Kursus koosneb loengutest ja harjutustest. Alustatakse elementaarselt tasemelt, milleks eeldusaineid ei nõuta. Kursus võib olla kuulajate esmakordseks kokkupuuteks programmeerimisega. Tutvutakse põhiliste programmeerimisvõtetega, sh blokkskeemide koostamisega ja MATLABi keskkonnaga. jõutakse välja suuremahulise arvutustöö korraldamiseni. Main elements of MATLAB. Arrays, Data types. Graphical features. Data Exchange. Solving of equations and systems. Approximation of functions. Introduction to differential equations, ordinary and partial. The means of finding of solutions of DE. Modelling in MATLAB. Toolboxes. Simulations and numerical experiments. Graphical user interface. The course consists of lectures mixed with accompanying exercises. It starts from elementary level, preliminary courses are not expected:the course could be the first experience of a participant with computer programming. Main elements of programming and the MATLAB programming environment are studied. Finally, use of objects and management of large computations is considered.
10	KOHUSTUSLIKUD EELDUSAINED	Eeldusaineid ei ole.

AINEKAVA (eesti keeles, inglise keeles vajalik täita ainult õppeaine toimumise korral selles keeles)

11	ÕPPEAASTA, SEMESTER, ÕPPEVORM	2013 sügis.
12	OSA, MAHT	Üheosaline, 6 EAP.
13	OSA(DE) LÕPPHINDAMINE/LÕPPHINDAMISED	Lõpphinne kujuneb vastavalt teemade käsitlemisel antud kohustuslike ülesannete lahenduste hinnetele. Lahendused tuleb üles laadida MOODLE'i keskkonnas.
14	VASTUTAV ÕPPEJÕUD (kui õppeainet viib läbi mitu õppejõudu, siis esitada kõigi (külalis) õppejõudude nimed)	Peep Miidla Jüri Vedru
15	OSALEJATE PIIRARV	Ei ole.
16	SIHTRÜHM JA ÕPPEAINES OSALEMISE	Õppeaine ei eelda mingeid spetsiifilisi eelteadmisi ja on mõeldud kõikidele õppeastmetele (Bakalaureuseõpe, Magistriõpe,

	EELTINGIMUSED	Doktoriõpe).
17	SOOVITUSLIKUD EELDUSAINED	Ei ole.
18	ÕPETAMISE KEEL(ED)	Eesti keel.
19	ÕPIVÄLJUNDITE SAAVUTAMISEKS VAJALIKUD TEISED KEELED	Inglise keel abivahendite ja kirjandusega tutvumiseks.
20	ÕPPETÖÖ MAHUD JA VORMID (kontaktõpe: loeng, seminar, praktikum, praktika, individuaaltund; iseseisev töö (sh e-õpe)	Kontaktõpe: loeng ja praktikum (2+2 tundi nädalas, kokku 64 tundi). Iseseisev töö veebivahendiga MOODLE (92 tundi).
21	VEEBIPÕHINE	Osaliselt veebipõhine.
22	TOIMUMISNÄDALAD	1. – 16.
23	AJAKAVA teemade kaupa.	1. Elementaarteadmised arvutitest. Üldine tutvumine programmipaketiga MATLAB. Aknad. Töökausta seadistamine. Paketi abivahendid. 2. Käsuakna kasutamine. Muutujate ja avaldiste sisestamine. Tehted maatriksitega. 3. MATLAB i andmetüübid. Struktuurid. *Cell*. 4. Sisseehitatud funktsioonid. 5. Graafika. Kahemõõtmeliste jooniste tegemine. Jooniseaken, joonise vormindamisaken. 6. Graafika. Kolmedimensionaalsed joonised. 7. M-failid. Toimetiaken. Programmeerimine paketis MATLAB. Blokkskeem. 8. Programmeerimine. Tsüklid. 9. Programmeerimine. Tingimuslikud konstruktsioonid. 10. Andmevahetus. Erinevais formaatides andmete sisestamine ja väljastamine.

		11. Elementaarne statistiline andmetöötlus. 12. Lihtsamate ülesannete lahendamine. Võrrandid ja võrrandite süsteemid. 13. Lihtsamate ülesannete lahendamine. Funktsioonide lähendamine. Andmete silumine. 14. *Optimization Toolbox*. Ülesannete püstitusi ja lahendusmeetodeid. Toolboxi funktsioonide ja kasutajaliidese kasutamine. 15. *Image Processing Toolbox*. Ülesannete püstitusi ja lahendusmeetodeid. Toolboxi funktsioonide ja kasutajaliidese kasutamine. 16. Sissejuhatus blokkmodelleerimise süsteemi SIMULINK. Mudeliväli. Blokkide teek. 17. Blokkmudeli koostamine.
24	ÕPPEAINE KODULEHT	moodle.ut.ee
25	RAAMATUKOGU AINEPAKETT (link)	
26	KOHUSTUSLIKUD ÕPPEMATERJALID	
27	SOOVITUSLIKUD ÕPPEMATERJALID	
28	ISESEISEV TÖÖ (iseseisvate tööde loetelu ja juhised nende tegemiseks)	Iseseisvate tööde loetelu ja juhend nende tegemiseks on üleval veebikeskkonnas MOODLE. Selle kaudu avanevad ka täiendavad konsulteerimisvõimalused.

ÕPIVÄLJUNDITE HINDAMINE

29	HINDAMISMEETOD(ID) *(tasemetest, kontrolltöö, essee, uurimistöö, kirjalik eksam, suuline eksam, kirjalik arvestus, suuline arvestus, kodutöö esitus, ettekanne, esitus, etendus, referaat, kollokvium, kodulugemise vastamine). Kirjeldatakse iga hindamise kohta.*	HINDAMISKRITEERIUMID Hindamise aluseks on keskkonnas MOODLE üles laaditud kohustuslike ülesannete lahendused. Hinded skaalal A – F kujunevad vastavalt esituste tähtajalisusele ja kvaliteedile.

	EKSAMILE/ ARVESTUSELE/ KORDUSEKSAMILE PÄÄSEMISE TINGIMUSED (tähtajad jms)	Üliõpilane, kes esitab hilinemisega kõik tööd, saab positiivse hinde.
	LÕPLIKU HINDAMISE KUJUNEMINE	Lõpliku hindamise aluseks on iseseisvalt lahendatud kohustuslikud ülesanded. Lahendused laaditakse üles keskkonnas MOODLE.
30	VÕLGNEVUSTE LIKVIDEERIMISE VÕIMALUSED	Võlgnevuste likvideerimine toimub jooksvalt veebikeskkonnas MOODLE.
31		
32		
33		

Eesmärk ja õpiväljundid

Kursuse eesmärgiks on üliõpilastele baasteadmiste andmine tänapäevase rakenduspaketi **MATLAB** kasutamise võimalustest, erinevate meetodite tundmaõppimine ja oskuste andmine nende rakendamiseks, samuti iseseisva töö oskuste treenimine. Üldine tutvumine programmipaketiga **MATLAB** (käivitamine, keskkonna ning töölaua (*Desktop*) seadistamine, akende kasutamine jne). Keskkonna **Moodle** kasutamine. Abiakna (*Help*) ja teiste tugimaterjalide ning veebiressursside kasutamine. Sisseehitatud funktsioonide kasutamine. Paketi veebileht http://www.mathworks.com.

The goal of the course is giving basic knowledge to the students in **MATLAB**, modern environment of modelling and simulation and training of skills of individual work.

Kursuse läbinud üliõpilased
- omavad ülevaadet **MATLAB**-i keskkonna ja keele võimalustest;
- oskavad töötada massiividega;
- tunnevad andmetüüpe ja oskavad kasutada **MATLAB**-i andmevahetuse ning –töötluse võimalusi;
- omavad arvutile programmeerimise kogemust ning oskust koostada, vormistada ja kasutada M-faile;
- tunnevad ja oskavad kasutada **MATLAB**-i graafika- ja visualiseerimisvahendeid;
- oskavad otsida ja kasutada veebis leiduvaid **MATLAB**-i programme ja abivahendeid.
- omavad piisavat vilumust iseseisvaks tööks;
- oskavad kasutada õpikeskkonda **MOODLE**.

The students who pass the course have following skills.
- Have overview of the possibilities of **MATLAB** environment and programming language.
- Know how to work with arrays.
- Have knowledge about data types and classes and have experience of data exchange, processing and management.
- Can write and use different m-files and have passed a first introduction to programming.
- Have knowledge and experience in the use of graphical and visualization features of **MATLAB**.
- Know how to use the means available through internet.
- Has good skills of individuaal work;
- Knows how to use **MOODLE** environment.

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Andmeanalüüs ja arvutused MATLAB-iga

Hädavajalikud lõppteadmised. Peab teadma, kuidas

- käivitada paketti **MATLAB**, **MAT**rix **LAB**oratory;
- defineerida muutujaid ja nendega arvutusoperatsioone sooritada;
- kasutada käskude sisestamisel semikoolonit;
- esitada arve erinevates formaatides;
- kirjeldada paketis kasutatavaid andmeklasse;
- kasutada sisseehitatud konstante (**ans**, **eps**, **pi** , **realmax**, **realmin**, **Inf**, **NaN**);
- kasutada paketi erinevaid aknaid;
- saada abi käsuakna kaudu (**help**, **doc**, **lookfor**);
- kasutada abiakent;
- toimetada tööpiirkonnas (**Workspace**, **clear**, **who**, **whos**);
- sisestada vektoreid ja maatrikseid, s.h. kooloni kasutamisega;
- sisestada avaldisi, ka mitut ühel real ning ühte mitmel real;
- eraldada massiividest elemente ja alammassiive indeksite ja kooloni abil;
- kasutada vektorite loomiseks käske **linspace** ja **logspace**;
- luua maatrikseid sisseehitatud käskude abil (**diag**, **eye**, **ones**, **rand**, **zeros**);
- kasutada transponeerimisoperaatorit;
- kasutada tavalisi (*, /, ^) ja elementidekaupa (.*, ./, .^) operatsioone;
- toimetada kompleksarvuliste massiividega (**abs**, **angle**, **exp**, **conj**, **imag**, **real**);
- seadistada töökausta (**path**, **File** → **Set Path**) ja töölaua muid komponente;
- lugeda sisse andmeid failist (**load**, **fopen**, **fscanf**);
- salvestada andmeid (**save**, **fclose**);
- kasutada sisseehitatud funktsioone;
- sisestada stringe ja manipuleerida nendega (**char**, **findstr**, **length**, **num2str**, **str2num**, **strcmp**, **strncmp**, **sprint**);
- luua ja kasutada skriptitüüpi (*script*) m-faile
- tippida m-failidesse ja käskude osadena kommentaare (%);
- töötada polünoomidega (**conv**, **deconv**, **poly**, **polder**, **polyval**, **polyfit**, **roots**);
- teha andmete graafikuid (**plot**, **subplot**, **loglog**, **semilogx**, **semilogy**, **axis**, **grid**, **gtext**, **legend**, **text**, **xlabel**, **ylabel**, **title**);
- teha kahe- ja kolmedimensionaalseid jooniseid (**contour**, **surf**, **plot3** jne.);

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Juhend kohustuslike ülesannete lahenduste vormistamiseks

1. Üldised tingimused.

Hindamise aluseks on kõigi kohustuslike ülesannete lahendused, mis on esitatud õpikeskkonna Moodle kaudu. Kohustuslike ülesannete tekstid ja üleslaadimise kohad on toodud eraldi teemana ning nummerdatud. Ülesannete tekste tuleb tähelepanelikult lugeda ning täita seal toodud nõuded. Nimelt tuleb mõnikord lahendada kõik alamülesanded, mõnel juhul võib teha valiku etteantute seast või ise ülesande leida. Kohustuslikke ülesandeid hinnatakse eristaval skaalal punktidega 0 - 100. lõpphinne kujuneb üksikute hinnete keskmisena. Maksimaalset hinnet alandatakse esituse hilinemise tõttu. On lubatud lahenduste korduvesitamine.

2. Ülesannete lahenduste vormistamine ja esitamine.

Kohustuslike ülesannete lahendused tuleb leida ja vormistada programmipaketi **MATLAB** abil. Lahendused tuleb esitada m-failin(de)a. Faili(de) nime(de)ks tuleb panna üliõpilase perekonnanimi, seejärel ülesande number. Kui ühe teema kohta esitatakse mitu faili, tuleb perenime ja teema numbri järele paigutada allkriips „_“ ja selle järel alamfaili number. See võib osutuda vajalikuks näiteks siis, kui kasutatakse m-failina vormistatud funktsiooni.

Näiteks:

```
Tamm2.m    % See on lahendaja Tamm kohustusliku ülesande nr. 2 lahendamise kohta käiv m-fail.

Tamm2_3.m  % See on lahendaja Tamm kohustusliku ülesande nr. 2 lahendamise kohta käiv m-fail number 3.
```

NB! Kuna paketis MATLAB ei tohi muutujate, sealhulgas programmide nimedes kasutada umlaut- ja täpitähti (õ, ä, ö, ü), siis tuleb tudengil oma perenimi vajadusel modifitseerida. Failid peavad olema vormistatud nii, et neid saab pärast allalaadimist paketi MATLAB käsuaknast käivitada. Kohe faili algul tuleb kommentaaridena kirjutada faili käsitsemisjuhis, autori andmed ja kirjeldada täpselt ülesannet, millist lahendatakse. Kommentaarid tipitakse paketis MATLAB märgi „%“ järele. Esitatavasse faili tuleb kommentaaridena lisada ka kõik vajalikud seletused ülesande lahenduskäigu kohta. Kui seletus lisatakse mõnes teises formaadis vormindatud failina, tuleb see ka põhifailis märkida.

Lühidalt ja veelkord: kohustuslike ülesannete lahendused tuleb varustada täielike kommentaaridega. Nende puudumine võib põhjustada hinde alandamist. Kui esitatakse teistsuguses formaadis (mitte m-failidena) materjale, tuleb nende nimetamisel kinni pidada ülaltoodud reeglitest – seda tuleb teha perekonnanime alusel.

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Teema 1. Paketi MATLAB töölaud ja abivahendid

LOTI.05.019

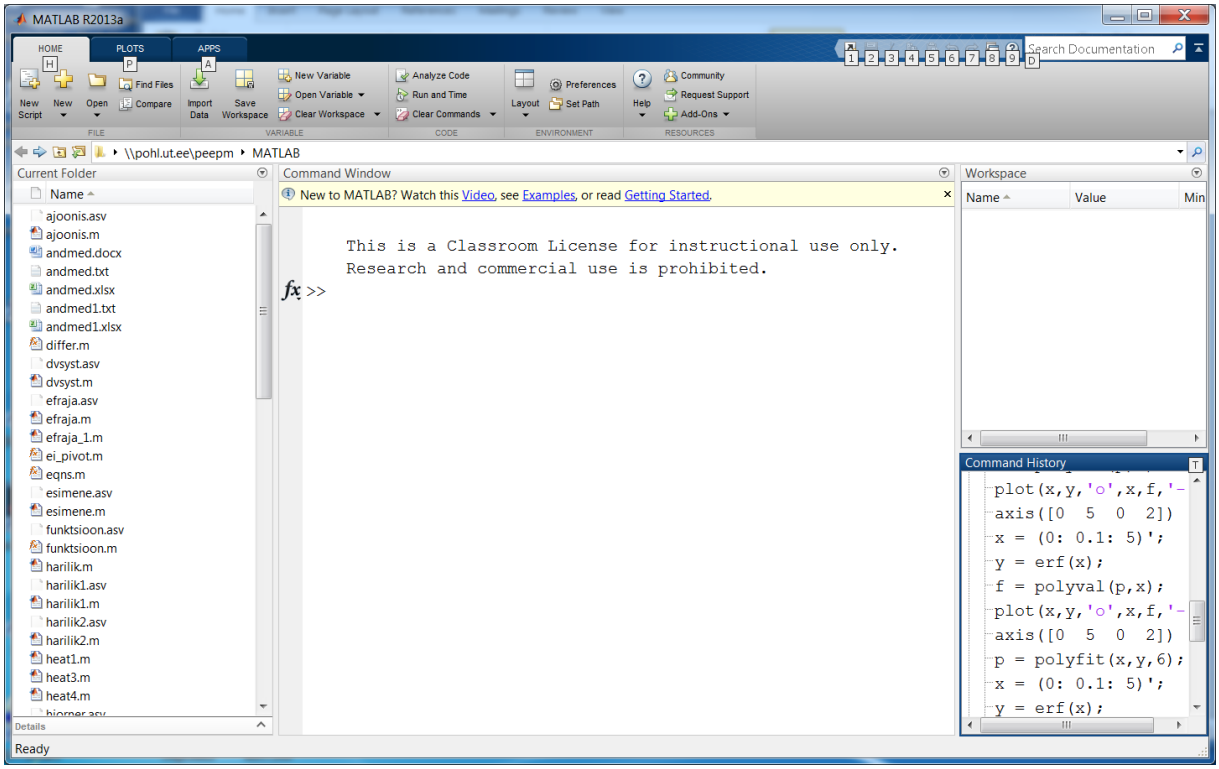
Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

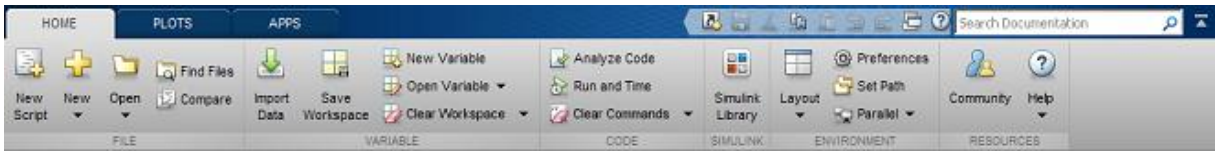
MATLAB käivitub firma MathWorks (http://www.mathworks.se/) logoga ikooni klõpsutamisel. Selle teema raames õpitakse tundma paketi töölaua (desktop) ja töökausta seadistamist. Vaatleme töölaua komponente: käsuaken (*Command Window*), tööriistariba (*Toolstrip*), töökaust (*Current Folder*), käskude ajalugu (*Command History*), mälupiirkond (*Workspace*). Toimetiaknas (*Editor*) saab luua m-faile. Ülevaade paketi abivahenditest.

MATLAB käivitus ja töölaud

MATLAB käivitub firma MathWorks (http://www.mathworks.se/) logoga ikooni klõpsutamisel. Paketi uued versioonid alates 2012 aastast näevad välja erinevad eelkäijatest:

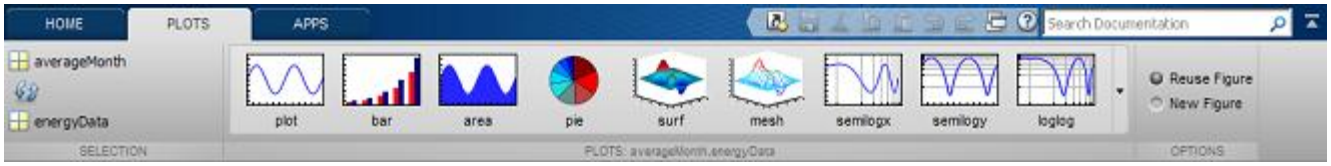


Lisatud on tööriistariba (**MATLAB** *Toolstrip*) päise all:



Tööriistaribale on seatud juurdepääsud mitmetele vajalikele tegevustele, mis varem olid kättesaadavad menüüde ja/või Start – nupu kaudu. Riba saab minimiseerida ja uuesti maksimaalseks muuta paremal ülanurgas asuva noole abil ja see võib olla oluline, kui kuvaripind on napp.
Käivitamise järel näeme tööriistaribal kolme *tab*'i: **HOME**, **PLOTS** ja **APPS**. Need on töölaual kogu aeg nähtavad, sõltumata tegevustest paketiga, seepärast nimetatakse neid ka globaalseteks *tab*'ideks (*global tabs*).
Operatsioonid on rühmitatud riba allservas olevate kategooriate järgi: **FILE** – failidega toimetamiseks, **VARIABLE** – muutujatega tegelemiseks, **CODE** – koodi ehk m-faili muutmiseks, **SIMULINK** – blokkmodelleerimissüsteemi **SIMULINK** kasutajatele, **ENVIRONMENT** – töökeskkonna seadistamiseks ja **RESOURCES** – mitmesuguste abi- ja veebimaterjalidega tegelemiseks.

Järgmine globaalsetest *tab*–idest on graafikavahendite pakkumiseks:



Paketi MATLAB tähtsad komponendid.

- ***Desktop***. Kõik aknad on eraldatavad

- ***Command Window***, käsuaken. Siit käivitatakse paketi MATLAB konstruktsioone (käsud, programmid, funktsioonid).

- ***Command History***, käskude ajalugu. Käsuaknasse sisestatud korralduste logi.

- ***Current Folder***. Näitab töökausta, kuhu salvestatakse ka kasutaja failid ning programmid.

- ***Workspace***. Näitab viimati käivitatud skripti muutujaid ja globaalmuutujaid ning valikut neid iseloomustavaid suurusi. (Erinevalt käsust **whos** ei näita seda programmi töö käigus (isegi mitte käsu **pause** toimimise ajal), vaid ainult siis, kui programmi töö on lõpetatud või katkenud.)

- ***Editor***, toimeti, avaneb ka nuppudega **Open** või **New M-file**. Programmide loomiseks ja redigeerimiseks.

- ***Help***, abiteabe aken, F1. Asendamatu abimees täpse süntaksi alal. Neli alamakent:
Contents, ***Index***, ***Search Results***, ***Demos***.
Demovahenditega tutvumine on väga oluline.

- ***Array Editor***. Muutujate (massiivide) väärtuste kontrollimiseks ja redigeerimiseks.

- ***Figure***, jooniseaken. Koos joonisega avaneb joonise vaatlemise vahendite tööriistariba, millel nupu  alt saab avada ka joonise redigeerimise vahendite menüü.

- ***Toolbox***, spetsialiseeritud lisaprogrammide komplektid vastavate ülesannete lahendamiseks.

MATLAB toimetiaken ja script

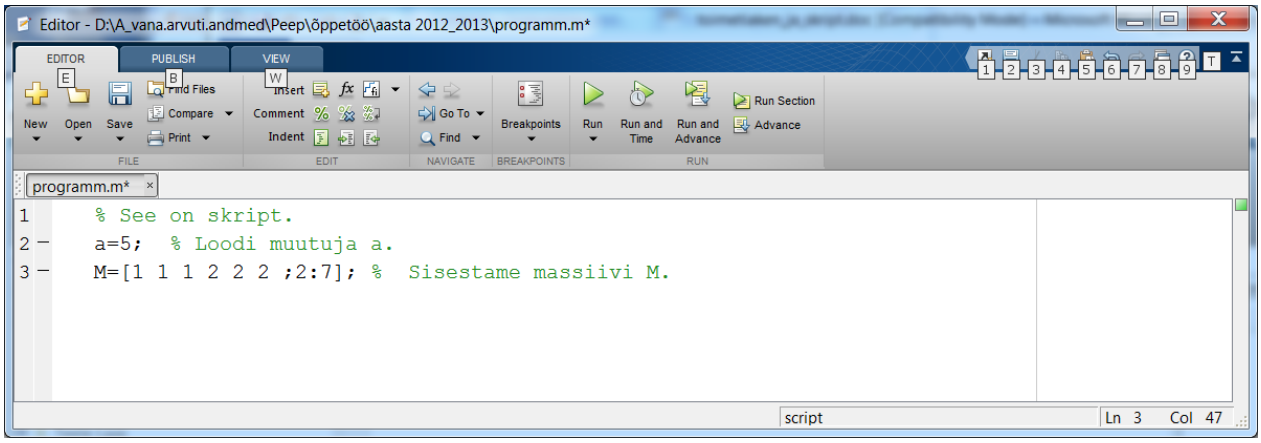
Paketi **MATLAB** üheks oluliseks eeliseks on võimalus kirjutada programme ja neid lihtsal viisil käivitada. Lihtsaim programm, skript (*Script*) kujutab endast üksteise järele kirjutatud sisestusdirektiividest, mis on salvestatud töökausta (*Current Directory*) koostaja poolt valitud nime all laiendiga m. Programmi nimedes pole soovitav kasutada sisseehitatud muutujate või funktsioonide nimesid, keelatud on kasutada täpitähti, tähte õ ning matemaatilisi jt sümboleid. Näiteks, kui salvestate töökausta skripti "**programm.m**", siis selle käivitamiseks käsuaknast või teistest m-failidest väljakutsumiseks tuleb sisestada

>> **programm**;

Skriptide (m-failide) toimetamise aken Editor avaneb kui tööriistaribal klõpsata ikooni või valige tööriistaribalt valikust **New** või tehke parema hiireklahviga klõps käskude ajaloo (*Command History*) mõnel käsul ja valige **Create Script** või… (loodetavasti leiate veel võimalusi).

Skriptid on lihtsaimad programmid, nendel puuduvad sisend- ja väljundargumendid ning muutujad, mida kasutatakse ja genereeritakse on globaalsed, st kättesaadavad mälupiirkonnast (*Workspace*) **MATLAB**-i jooksva seansi ajal. Skriptid on mugavad kasutamiseks algusest peale, kui tahate käsuaknast sisestatud korraldusi korduvalt täita. skriptide muutmiseks või toimetamiseks klõpsutage töökausta aknas (Current Directory) vastaval ikoonil või tippige käsuaknasse

>> **edit failinimi**



Roheline ruuduke sisestuspiirkonna parempoolsel ülanurgal näitab, et sisestusel ei ole süntaksivigu avastatud. Kui „kaart“ on kollane, oleme hoiatatud, kui aga punane, siis teame, et see skript ei käivitu, vead on sees ja need tuleb kõrvaldada. Toimetiaknast näeb, kus midagi viltu on.

Kui selle nimega faili pole, avatakse toimetiaken uue skripti loomiseks. Sisestatud skripti saab käivitada toimetiakna päises asuva klahviga  .

Väga soovitatav, kohustuslike ülesannete lahenduste puhul koguni kohustuslik, on kommentaaride lisamine m-failile. Kommentaarid sisestatakse protsendi märgi (**%**) järgi ja neid võib kirjutada eraldi ridadele või käsuridade lõppu. Kommentaarid on olulised selleks, et teised saaksid autori ideedest aru koodi lugemisel, aga ka autori enese mälu värskendamiseks hiljem. Mugav on ka võimalus kiiresti teada saada, mida tele skript teeb. Kui tipite käsuaknasse

>> **help failinimi**

siis väljastatakse sinnasamasse selle skripti esimesed kommentaariread (kuni esimese käsusisestuseni), nn kommentaariblokid. Selleks tuleb skriptifail muidugi eelnevalt nende kommentaaridega varustada. Kommentaaride osas öeldu kehtib ka funktsioonitüüpi m-failide kohta, mida käsitleme hiljem. Muide, kui tipite käsuaknasse

>> **lookfor 'tekst'**

siis väljastatakse käsuaknasse kõigi nende programmide nimed, millede esimeses kommentaarireas leidub string "tekst". Sisestuse

>> **lookfor 'tekst' –all**

korral vaadatakse läbi kõigi programmide kõik kommentaariblokid.
Kui soovite **MATLAB**-i jooksva seansi jooksul käsuaknasse sisestatud korraldustest moodustada skripti, et seda järgmisel korral kasutada, avage toimetiaken, tähistage käskude ajaloo aknas (*Command History*) vajalikud käsud ning kopeerige need (*Copy – Paste*) toimetiaknasse. Seejärel varustage skript kommentaaridega, korrastage ja salvestage töökausta.

Programmipaketi MATLAB abivahendid

MATLAB-is on väga palju abi- ja näitlikustamisvahendeid. Esimene on abiaken (avaneb ka F1 vajutamisel):

Abiaknas on neli osa. Tavalistele **WINDOWS**- keskkonna sektsioonidele sisukord (**Contents**), indeks (**Index**) ja otsimine (**Search Results**) on lisatud näitlikustamisvahendite (**Demos**) oma.

MATLAB'i funktsioonidele on enamasti võimalik ette anda mitmesuguseid optsioone, mõned neist kohustuslikud, teised mitte. Nende süntaksite meeldejätmine pole alati vajalik. Tõhus võte: leida abiaknast sobiv konstruktsioon üles, teha **copy-paste** (kleepida kas käsuaknasse või toimetiaknasse) ja siis see oma vajadustele vastavaks redigeerida.

Abi saab ka käsurealt. Kasulik oleks proovida järgmisi sisestusi ja uurida väljastatavaid loetelusid:

>>**help ops**

>>**help relop**

>>**help arith**

>>**help slash**

Käsuaknast saab avada ka **Help** akna vastava kirje:

>>**doc plot**

Järgmine sisestus annab teada selle m-faili täieliku nime, mis realiseerib funktsiooni **plot**:

>>**which plot**

Paketis on terve rida olulisi operaatoreid, mille süntaksiks on tavalised kirjamärgid. Näiteks koolon:

>>**help colon**

Hulgaliselt abivahendeid, juhendeid ja näpunäiteid võib leida veebist. Alguseks vaadake paketi tootja **MathWorks** veebilehte:

http://www.mathworks.com/

Edasi „guugeldage“. **MATLAB** on tõhus aga kallis rakenduspakett, siiski riputab suur osa selle kasutajaist avalikult veebi välja omaloodud programme, näiteid, ülesannete lahendusi jne. **MathWorks**, tundub, ei ole väga selle vastu.

Teema 2. Andmed paketis MATLAB

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Reaalarvude esitamine. Digitaalarvuti tegutsemise põhimõtted. Paketis MATLAB saab töötada 15 fundamentaalse andmetüübiga. Arvude esitamise vaikimisi formaadiks on topelttäpsusega ujukoma arv (double-precision floating-point number). Kõikidest andmetüüpidest võib moodustada massiive ja maatrikseid.

Andmeklassid paketis MATLAB

Paketis **MATLAB** saab töötada 15 fundamentaalse andmetüübiga. Kõikidest võib moodustada massiive või maatrikseid.

- Boole´i muutujad, **logical** – loogilised muutujad väärtustega **true** või **false**. Esitatakse reeglina arvuliselt, vastaval 1 ja 0.

Näide.

```
>> t=magic(4)>10     % magic(4) loob 4×4 maagilise ruudu.
```

```
t =

   1    0    0    1
   0    1    0    0
   0    0    0    1
   0    1    1    0
```

- Ujukoma arvud ühekordse täpsusega: **single**.

Näide. (Eelmise jätk.)

```
>> a = magic(4);
>> b = single(a);
>> whos
```

```
  Name      Size            Bytes  Class     Attributes

  a         4x4               128  double
  b         4x4                64  single
  t         4x4                16  logical
```

Näite lõpp.

- Ujukoma arvud kahekordse täpsusega: **double**, vaikimisi kasutusel.

Vt. materjali "**reaalarvud.pdf**".

- Märgiga täisarvud, 8-bitilised, **int8**.
  Väärtuste muutumisvahemik: $-2^7 \ldots 2^7-1$.

- Märgiga täisarvud, 16-bitilised, **int16**.
  Väärtuste muutumisvahemik: $-2^{15} \ldots 2^{15}-1$.

- Märgiga täisarvud, 32-bitilised, **int32**.
  Väärtuste muutumisvahemik: $-2^{31} \ldots 2^{31}-1$.

- Märgiga täisarvud, 64-bitilised, **int64**.
  Väärtuste muutumisvahemik: $-2^{63} \ldots 2^{63}-1$.

- Märgita täisarvud, 8-bitilised, **uint8**.
  Väärtuste muutumisvahemik: 0 … 2^8-1.

- Märgita täisarvud, 16-bitilised, **uint16**.
  Väärtuste muutumisvahemik: 0 … 2^{16}-1.

- Märgita täisarvud, 32-bitilised, **uint32**.
  Väärtuste muutumisvahemik: 0 … 2^{32}-1.

- Märgita täisarvud, 64-bitilised, **uint64**.
  Väärtuste muutumisvahemik: 0 … 2^{64}-1.

- Funktsioonide viit, **function_handle**. Seda võib sisuliselt vaadelda **MATLAB**-i funktsiooni väljakutsutava tähisena, mis võimaldab kasutada funktsiooni teiste funktsioonide argumendina ja pöörduda funktsiooni poole väljastpoolt selle defineerimispiirkonda.

- Tekst, **char**.

Näide.

```
>> tekst = 'Tere hommikust';
>> tekst

tekst =

Tere hommikust

>> int8(tekst)   % Esitame tekstimuutuja täisarvudena.

ans =

  84  101  114  101   32  104  111  109  109  105  107  117  115  116
```

Näite lõpp.

- Heterogeensed andmed, nimepõhine, struktuur, **struct**.

- Heterogeensed andmed, indeksipõhine, **cell**.

Andmevahetus failidega

>>**save** % Salvestatakse mälupiirkonna (*Workspace*) muutujad. Kasutatakse siis, kui tahetakse järgmisel MATLAB-i seansil jätkata "samast kohast", kus hetkel

lõpetatakse. Mälupiirkonna muutujad salvestatakse faili laiendiga .mat, selle vaikimisi nimeks on matlab.mat. Salvestatakse nii muutujate nimed kui ka väärtused.

>>**load** % Loetakse sisse muutujad mälupiirkonda (*Workspace*). Kasutatakse siis, kui tahetakse käsiloleval MATLAB-i seansil jätkata "samast kohast", kus varasemal korral lõpetati. Mälupiirkonna muutujad taastatakse failist laiendiga .mat, vaikimisi nimeks "matlab.mat". Taastatakse nii muutujate nimed kui väärtused.

>>b=**load**('arvud.txt'); % Loetakse sisse andmed failist nimega "arvud.txt", mis eelnevalt peab olema salvestatud töökausta.

>>b=**importdata**('arvud.txt'); % Loetakse sisse andmed failist nimega "arvud.txt", mis eelnevalt peab olema salvestatud töökausta.

>>b=**load**('arvud.xxx','-ascii'); % Loetakse sisse andmed failist nimega "arvud.xxx", mis eelnevalt peab olema salvestatud töökausta. Muutujat b käsitletakse *ascii*-muutujana olenemata sisseloetava faili laiendist ".xxx".

>>b=**load**('fail.xxx','-mat'); % Loetakse sisse andmed failist nimega "fail.xxx", mis eelnevalt peab olema salvestatud töökausta. Muutujat b käsitletakse *mat* - muutujana olenemata sisseloetava faili laiendist ".xxx".

Kui enne eelmist sisestust sisestada järgnev, saab faili "fail.xxx" sisu teada.

>> **disp**('Faili fail.mat sisu on järgmine:') % Väljastab käsuaknasse teksti.
>> **whos** -file fail.mat % Väljastatakse muutujate loetelu.

>>**save** ('muutujab.txt','b','-ASCII'); % Salvestatakse muutuja, näiteks massiiv **b** koos väärtustega (ASCII) faili nimega "muutujab.txt", fail salvestatakse töökausta..

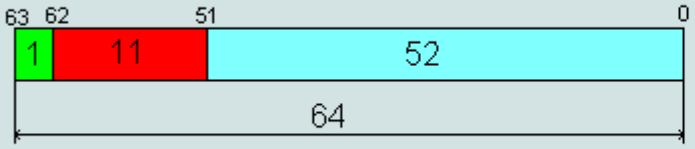
>>a=**xlsread**('Vihik.xls'); % Loetakse sisse andmed EXCELI failist nimega "Vihik.xls", sealne koma teisendatakse punktiks. Andmemassiivi nimeks saab **a**.

>>a=**xlswrite**('Vihik1.xls',a); % Salvestatakse andmed EXCELI faili nimega "Vihik1.xls", **MATLAB**-is kümnendkoha eraldajana kasutatav koma teisendatakse punktiks.

Reaalarvude esitamine

Meenutame (64- bitise) digitaalarvuti tegutsemise põhimõtteid. IEEE (*Institute of Electrical and Electronics Engineers*) on fikseerinud standardi ujukoma arvude aritmeetika realiseerimiseks digitaalarvutis. See standard määratleb, kuidas ühekordse

täpsusega (*single precision*, 32 bit) ja topelttäpsusega (*double precision*, 64 bit) ujukoma arvud esitatakse ja kuidas nendega tehteid tehakse. muuhulgas on fikseeritud ka ümardamisreeglid nn vahemikaritmeetika kaudu. Topelttäpsusega arvuti esitamiseks kasutatakse 64 bitti (8 baiti), ühekordse täpsusega arvude puhul vastavalt 32 bitti ja 4 baiti. Bitid nummerdatakse kokkuleppeliselt vasakult paremale: 63, 62,…, 1, 0.



$$F = (-1)^S \; 2^{(E-1023)} (1+M/2^{52})$$

Ühekordse täpsusega arvu võib ette kujutada 32- bitise sõnana, kahekordse täpsusega arvu – 64- bitise sõnana. Esimene, vasakpoolne bitt näitab arvu märki, kui S = 0, siis on tegemist positiivse arvuga, kui S = 1, siis negatiivsega. Järgmised bitid, "E", tähistavad eksponenti ja viimased, "F" murdosa ehk mantissi.

```
     S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
    31 32    23 22                    0


     S EEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    63 62        52 51
0
```

Paketis **MATLAB** on arvude esitamise vaikimisi formaadiks topelttäpsusega ujukoma arv (*double-precision floating-point number*). Kui aritmeetilises tehtes vähemalt üks operand on topelttäpsusega ujukoma arv (*double*), on seda kogu tehte tulemus. **MATLAB**-i funktsioonid **realmax** ja **realmin** väljastavad vastavalt suurima ja vähima positiivse topelttäpsusega ujukoma arvu, mida paketis saab esitada ja kasutada. Negatiivsete arvude piiride saamiseks on vaja nende funktsioonide ette panna miinus.

Näide.

Sisestades käsuaknast kaks esimest rida, näeme allpool toodud tulemust (**ans**).

```
>> str = 'The range for double is:\n\t%g to %g and\n\t %g to  %g'; % Esimene sisestus

>> sprintf(str, -realmax, -realmin, realmin, realmax)  % Teine sisestuse rida. Allpool vastus.

ans =
```

The range for double is:

-1.79769e+308 to -2.22507e-308 and

2.22507e-308 to 1.79769e+308

Ühekordse täpsusega ujukoma arvude piire näeb järgmiste sisestustega.

```
>> tr = 'The range for single is:\n\t%g to %g and\n\t %g to  %g';
>> sprintf(str, -realmax('single'), -realmin('single'), ...
realmin('single'), realmax('single'))
ans =
The range for double is:
        -3.40282e+038 to -1.17549e-038 and
         1.17549e-038 to  3.40282e+038
```

Näite lõpp.

Eelmises näites kasutati paketi **MATLAB** olulisi konstruktsioone:

- Kommentaarid algavad sümboliga " **%** ". Neid võib lisada igale poole, tähtsad on need m-failides.

- Kui sisestus lõpeb semikooloniga " **;** ", siis selle tulemust käsuaknasse ei väljastata.

- Sisestusrea poolitamiseks ja jätkamiseks järgmiselt realt tuleb enne reavahetust sisestada kolm punkti "**…**".

- Tekstitüüpi muutuja loomiseks (näites **str** ja **tr**) tuleb kasutada stringimärke **' '** (need asuvad tärniga samal klahvil ülemises registris).

Arvudele, mis on suuremad kui **realmax** või väiksemad kui **-realmax** are omistatakse väärtuseks vastavalt positiivne või negatiivne lõpmatus.

Et arvutis saab esitada vaid lõplikku arvu ujukoma arvusid, pole võimalik arvutis esitada kõiki reaalarve. Igas arvutis on kahe järjestikuse reaalarvu vahel väike vahe. Seda saab vaadata paketi MATLAB sisseehitatud funktsiooni **eps** abil.

Näide.

```
>> realmax + .0001e+308
ans =
   Inf
```

```
>> -realmax - .0001e+308

ans =

  -Inf

>> format long

>> eps(5)  % Leiame arvu 5 ja sellele järgneva reaalarvu vahe.

ans =

   8.881784197001252e-016
```

Näite lõpp.

Näite viimasest tulemusest näeme, et arvule 5 järgneb arvutiesituses reaalarv 5 + **eps**(5) ja nende vahel ei ole topelttäpsusega reaalarvu võimalik esitada. Iseseisvalt proovides saab näha, et **eps**(x) sõltub arvust x. Kuidas?

Argumendita funktsioon **eps** väljastab käsuaknasse sama tulemuse nagu **eps**(1). Analoogiliselt saab kontrollida ka ühekordse täpsusega arvudele järgnevaid. siis peab aga funktsiooni **eps** argument olema sisestatud kui ühekordse täpsusega (*single*) muutuja.

Näites kasutati olulisi **MATLAB**-i funktsioone:

- **format long** sisestamise järel väljastatakse arvud käsuaknas pikal kujul.

- **format short** on vaikimisi väljundiformaat, lühike. Väljundiformaadist arvutuste täpsus ei sõltu.

Arvude sellise esituse tõttu võib ette tulla ootamatuid arvutustulemusi. Allpool olevas näites tuleb leida täpsed tulemused ja selgitada, mis valesti "paistab". Mõned konstruktsioonid on etteruttavad, nendest tuleb aru saada vastavalt hiljem.

Näide.

```
>> arv = 1 - 3*(4/3 - 1)

arv =

   2.220446049250313e-016

>> a = 0.0;

for i = 1:10

  a = a + 0.1;

end
```

```
a == 1

ans =

    0

>> b = 1e-16 + 1 - 1e-16;

>> c = 1e-16 - 1e-16 + 1;  % Liidetavate järjekord on oluline mõnikord.

b == c

ans =

    0

>> (2^53 + 1) - 2^53  %  Suurtele arvudele järgnevad arvud suuremate vahedega.

ans =

    0

>> sin(pi)  % Oodatav tulemus oleks null.

ans =

   1.224646799147353e-016

>> sqrt(1e-16 + 1) - 1

ans =

    0
```

Näite lõpp.

http://www.mathworks.se/help/matlab/matlab_prog/floating-point-numbers.html

Teema 3. Käsuakna kasutamine

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Muutujate nimed võivad sisaldada ladina tähestiku tähti ja numbreid. Tähed on tõstutundlikud. Muutujate ja avaldiste sisestamine. Tehted maatriksite ja teiste massiividega. Massiivi elementide indekseerimine. Kui sisestuse lõpus on semikoolon, siis tegevuse tulemust ei kuvata. Kommentaarid tipitakse protsendi märgi järele.

Avaldised paketis **MATLAB**

Arvude esitamine.

- tavalisel, täis- ja reaalarvulisel kümnendkujul: 3; -99; 0.0001; 9.6397238;

- eksponentkujul: 1.60210e-20 = $1.60210*10^{-20}$; 6.02252e23 = $6.02252*10^{23}$;

- kompleksarvud: 1**i** = √(-1); 2 - 3.141**j** = 2 - 3.141*√(-1); 3e5**i** = $3*10^5$ *√(-1).

Esituses käsuaknas kuni 16 kümnendkohta (**format long**), tõkked reaalarvudele: 10^{-308} kuni 10^{+308}, täpsemalt vt **realmax** ja **realmin**.

Tehted.

+ liitmine; - lahutamine; * korrutamine; / jagamine;

\ jagamine vasakult (maatriksite korral); ^ astendamine;

’ kaaskompleksarv (transponeerib ka maatriksi); ( ) tehete järjekord.

Massiivide korral veel elemendikaupa tehted: a.*b, a./ b ja a.^b (vastavalt: elemendikaupa korrutamine, elemendikaupa jagamine ja elemendikaupa astendamine).

Konstandid.

pi = 3.14159265… ;

i imaginaarühik, √(-1) ; **j** sama, mis **i** ;

eps ujukoma arvude suhteline täpsus, ε = 2.2204e-016 = 2^{-52} ;

realmin vähim ujukoma arv, 2.2251e-308 = 2^{-1022} ;

realmax suurim ujukoma arv, 1.7977e+308 = $(2 - \varepsilon)2^{1023}$;

Inf lõpmatus;

NaN ei-ole-number.

Avaldised.

Kirjutatakse käsureale käsuaknas (**Command Window**) või m-failides. Avaldised kirjutatakse ühes reas, tehete järjekorra märkimiseks kasutatakse ümarsulge. Rea jätkumise märgiks on kolm punkti (...), mis tipitakse poolitatavale reale tühikuteta.

```
>> roo = (1+sqrt(5))/2

roo =

1.6180

>> a = abs(3+4i)

a =

5

>> z = sqrt(besselk(4/3,roo-i))

z =

  0.3730 + 0.3214i

>> suur = exp(log(realmax))

suur =

1.7977e+308

>> liigasuur = pi*suur

liigasuur =

Inf
```

Protsendi märgiga (**%**) algavad kommentaarid kas eraldi real või avaldise järel.

Kui avaldise või direktiivi järel on semikoolon (**;**), siis tegevuse tulemust käsuaknas ei kuvata. Seda on oluline tähele panna m-failide kirjutamisel.

Maatriksid ja vektorid paketis MATLAB

Tähtsaim paketi **MATLAB** poolt toetatav andmetüüp on maatriks.

- [1 2 3] või [1, 2, 3] on reavektor;
- [1; 2; 3] on veeruvektor;
- [1 2; 3 4; 5 6] on 3 × 2 maatriks;
- [1:4] on sama mis [1 2 3 4]; koolon tähendab siin „ühest neljani sammuga 1“;
- [pi:-pi/10:-2*pi] tähendab „π-st miinus kahe π-ni sammuga -π/10“;
- A’ annab maatriksi A transponeeritud kaaskomplekssе maatriksi;
- A.’ annab maatriksi A transponeeritud maatriksi;
- **transpose**(A) annab samuti maatriksi A transponeeritud maatriksi;
- A(2, 3) on maatriksi A teise rea kolmanda veeru element;
- A(1, :) on maatriksi A esimene rida vektorina;
- A(2, [1 3]) on vektor [A(2,1), A(2,3)];
- A([1 2], [3 4]) on maatriksi A alammaatriks [A(1,3) A(1,4);A(2,3) A(2,4)] ;
- **ones**(2,3) on ühtedest koosnev 2 × 3 maatriks;
- **zeros**(2,3) on nullidest koosnev 2 × 3 maatriks;
- **eye**(n) on n×n maatriks, mille peadiagonaalil on ühed ja mujal nullid;
- **eye**(n,m) on n×m maatriks, mille diagonaalil on ühed ja mujal nullid;

- **ones**(A) on ühtedest koosnev maatriks, mille dimensioonid võrduvad maatriksi A omadega;

- **diag**(A) - kui A on etteantud maatriks, saame vektori, mille elementideks on maatriksi A peadiagonaali elemendid;

- **diag**(v) - kui v on etteantud vektor, saame maatriksi, mille peadiagonaalil on vektori v elemendid, ülejäänud nullid;

- **diag**(v,k) - kui v on etteantud vektor, saame maatriksi, mille peadiagonaalist k võrra üles (k > 0) või alla (k < 0) nihutatud kõrvaldiagonaalil on vektori v elemendid, ülejäänud nullid;

- A*B on maatriksite A ja B maatrikskorrutis;

- A.*B on maatriksite A ja B korrutis vastavate elementide kaupa;

- A+B on maatriksite A ja B summa;

- A-B on maatriksite A ja B vahe;

- 2*A on maatriks A, mille elementideks on maatriksi A kahekordsed;

- A + 3 liidab maatriksi A igale elemendile arvu 3;

- **sum**(A) annab vektori A elementide summa; kui A on maatriks, siis annab reavektori, milles on A veergude summad;

- **max**(A) annab vektori A maksimaalse elemendi; kui A on maatriks, siis annab reavektori, milles on A veergudes olevad maksimaalsed väärtused;

- **sin**(A) leiab maatriksi A iga elemendi siinuse;

- **inv**(A) on maatriksi A pöördmaatriks;

- **norm**(A,p) annab maatriksi A p-normi;

- A/B on maatriks $A*B^{-1}$ (kui pöördmaatriks B^{-1} eksisteerib);

- B\A on maatriks B^{-1} *A (kui pöördmaatriks B^{-1} eksisteerib);

- A./B on maatriks, mille elementideks on A ja B vastavate elementide jagatised (A ja B peavad olema ühesuguste mõõtmetega);

- B.\A on maatriks, mille elementideks on A ja B vastavate elementide pöördjagatised (A ja B peavad olema ühesuguste mõõtmetega);

- **expm**(A) arvutab maatriksi A eksponendi (eksponentmaatriksi);

- **logm**(A) leiab maatriksi A naturaallogaritmi, A = **expm**(**logm**(A));

- **sqrtm**(A) – ruutjuur maatriksist, A = **sqrt**(A)***sqrt**(A);

Näiteid

Rea elemendid eraldatakse üksteisest komade või tühikutega ning read semikoolonitega. Kümnenderaldajaks on punkt.

Maatriksi elemendid võidakse sisestada ühe tekstireana nurksulgudesse:

```
>>A=[1.4 2.6,3;4,5.5 6;]

 A =

   1.4000   2.6000   3.0000
   4.0000   5.5000   6.0000
```

Maatriksit võib sisestada ka nagu tabelit:

```
>> A=[1 4 7 10
      2 5 8 11
      3 6 9 12]

A =
   1   4   7  10
   2   5   8  11
   3   6   9  12
```

Siit saame näiteks eraldada alammaatriksi:

```
>> B=A([1 2], [3 4])

B =
   7  10
   8  11
```

Suuremaid maatrikseid saab kokku panna väikesematest. Sel juhul peavad osamaatriksite mõõtmed ühendamiseks sobima, nt:

```
>> B=[2,5;3.3 15];
>> C=[A,B]

C =

   1.4000   2.6000   3.0000   2.0000   5.0000
   4.0000   5.5000   6.0000   3.3000  15.0000
```

Juhuslike arvudega täidetud maatriksi genereerimiseks on pakettides käsud **rand**(m,n) (sel juhul jaotuvad elementide väärtused ühtlase jaotuse kohaselt) ja **randn**(m,n) (siis

jaotuvad elementide väärtused normaaljaotuse kohaselt), kus m on maatriksi ridade ja n veergude arv:

```
>>rand(5,2)
ans =

    0.8147    0.0975
    0.9058    0.2785
    0.1270    0.5469
    0.9134    0.9575
    0.6324    0.9649
```

Näiteid paketi **MATLAB** sisseehitatud funktsioonide kasutamise kohta.

```
>>Z = zeros(2,4)
Z =

     0     0     0     0
     0     0     0     0

>>F = 5*ones(3,3)
F =

     5     5     5
     5     5     5
     5     5     5

>>N = fix(10*rand(1,10))
N =

     1     9     9     4     8     1     4     9     7     9

>> B=[ones(2,3),zeros(2,3)]

B =

     1     1     1     0     0     0
     1     1     1     0     0     0
```

Suuri maatrikseid väljastades esitab MATLAB veerunumbrid:

```
>> Q=randn(2,15)

Q =

  Columns 1 through 9

   -1.4916   -1.0616   -0.6156   -0.1924   -0.7648   -1.4224   -0.1774    1.4193    0.1978
   -0.7423    2.3505    0.7481    0.8886   -1.4023    0.4882   -0.1961    0.2916    1.5877
```

Columns 10 through 15

-0.8045	0.8351	0.2157	-1.1480	0.7223	-0.6669
0.6966	-0.2437	-1.1658	0.1049	2.5855	0.1873

Pakett MATLAB, harjutusülesandeid

1. Sisestada vabalt valitud elementidega 4x4 maatriks A ning genereerida käsuga **rand** 4x4 maatriks B.

(i) Leida A*B, A/B, A\B, A .*B, A ./B, A .\B, A^{-1}.

(ii) Uurida saadud maatrikseid käskudega **flipud**, **fliplr**, **rot90**, **flipdim**.

(iii) Eraldada leitud maatriksite 2. veerud ning moodustada neist käsuga **cat** uus maatriks.

(iv) Saadud maatriks transponeerida ja jätta sellest välja kaks vähima reasummaga rida.

2. Sisestada kaks nullidest ja ühtedest koosnevat vektorit, mõlemad näiteks 10-elemendilised. Milliseid võimalusi nende konstrueerimiseks võib leida? Olgu nendeks näiteks u ja v.

(i) Leida **and**(u,v), **or**(u,v), **xor**(u,v), **not**(u), **not**(v). Selgitada tulemusi.

(ii) Leida u > v, u < v, u == v, (u > v) | (v < u), (u > v) & (v < u). Selgitada tulemusi.

(iii) Leida u > 3, v < 3, u((u < 2) | (v >= 1)), v((u < 2) | (u >= 0)). Selgitada tulemusi.

3. Sisestada vabalt valitud vektor pikkusega näiteks 15.

(i) Liita vektori igale elemendile 12.

(ii) Liita paarituarvulise indeksiga elementidele 6.

(iii) Leida ruutjuur igast elemendist ja elementide ruutjuurte summa.

(iv) Leida iga elemendi ruut ja nende ruutude keskmine.

4. Moodustada vektor elementidega

$x_n = (-1)^{n+1}/(2n-1)$.

Alustades vektorist n=3 lisada sellele elemente, kuni vektori pikkuseks on näiteks 10.

Paketi MATLAB käsuakna kasutamisel

Peab teadma, kuidas

- käivitada paketti **MATLAB**, **MAT**rix **LAB**oratory;

- defineerida muutujaid ja nendega arvutusoperatsioone sooritada;

- kasutada käskude sisestamisel semikoolonit;

- esitada arve erinevates formaatides;

- kirjeldada paketis kasutatavaid andmeklasse;

- kasutada sisseehitatud konstante (**ans**, **eps**, **pi** , **realmax**, **realmin**, **Inf**, **NaN**);

- saada abi käsuakna kaudu (**help**, **doc**, **lookfor**);

- toimetada tööpiirkonnas (**Workspace**, **clear**, **who**, **whos**);

- sisestada vektoreid ja maatrikseid, s.h. kooloni kasutamisega;

- sisestada avaldisi, ka mitut ühel real ning ühte mitmel real;

- eraldada massiividest elemente ja alammassiive indeksite ja kooloni abil;

- kasutada vektorite loomiseks käske **linspace** ja **logspace**;

- luua maatrikseid sisseehitatud käskude abil (**diag**, **eye**, **ones**, **rand**, **zeros**);

- kasutada transponeerimisoperaatorit;

- kasutada tavalisi (*, /, ^) ja elementidekaupa (.*, ./, .^) operatsioone;

- toimetada kompleksarvuliste massiividega (**abs**, **angle**, **exp**, **conj**, **imag**, **real**);

- lugeda sisse andmeid failist (**load**, **fopen**);

- salvestada andmeid (**save**, **fclose**);

- kasutada sisseehitatud funktsioone;

- sisestada stringe ja manipuleerida nendega (**char**, **findstr**, **length**, **num2str**, **str2num**, **strncmp**, **sprint**);

- tippida kommentaare (%);

Teema 4. Graafika paketis MATLAB

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Graafika on paketi **MATLAB** üks huvitavamaid ja tähtsamaid vahendeid. Jooniste tegemiseks ja andmete visualiseerimiseks on palju ja erinevaid vahendeid. Kahe- ja kolmemõõtmeliste jooniste tegemine. Jooniseaken. Joonise vormindamisaken. Läbiv teema, uut lisandub kogu kursuse jooksul.

Kahedimensionaalne joonis paketis MATLAB

Joonised tekitatakse eraldi graafikaakendes, millede struktuur on sarnane teiste akende omadega. Paketis **MATLAB** nummerdatakse graafikaaknaid automaatselt alates ühest. Kahedimensionaalseid jooniseid käsuaknast või m-failist tehakse sisseehitatud funktsiooniga **plot**. Selle süntaks on väga mitmekesine ja võimaldab praktiliselt kõik ette vormindada. Samas on sageli mõistlik teha joonise peaosad valmis selle käsuga ning lõplikuks kujundamiseks kasutada spetsiaalseid graafikatoimetamisvahendeid (**Plot Editor**). Käsu **plot** süntaksit saab alati täpsustada abiaknas (**Help**).

Käsu **plot** põhikonstruktsioonid on järgmised.

plot(y) – joonistatakse massiivi y graafik, abstsissteljel massiivi elementide järjekorranumbrid, ordinaattelje skaala määratakse automaatselt vastavalt massiivi y elementidele. Joonis tehakse pideva joonega, kus graafiku punktid ühendatakse sirglõikudega; see on jooniste tegemise vaikimisi seade.

plot(x1,y1,...,xn,yn) – ühte teljestikku joonistatakse massiivide paaride (x1,y1), … (xn,yn) graafikud. Tasandi punktid ühendatakse sirglõikudega.

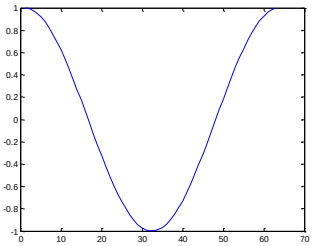
plot(x1,y1,LineSpec,...,xn,yn,LineSpec) – ühte teljestikku joonistatakse kolmikute (x1,y1,LineSpec), … (xn,yn,LineSpec) graafikud. Kolmikute esimeseks ja teiseks komponendiks on andmete massiivid, kolmandaga, LineSpec, määratakse punktide ühendamiseks kasutatavate sirglõikude paksus, andmemarkerite tüüp ja suurus ning värvid.

plot(...,'*PropertyName*',PropertyValue,...) – saab ette anda erinevaid joonise omadusi (omaduse nimi on stringimärkide vahel) ja nende konkreetseid väärtusi, mis enamasti on numbrilised.

Nüüd mõned näited.

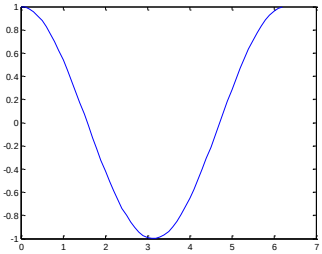
Lihtne kahedimensionaalne graafik.

```
>> t=0:0.1:2*pi;  % Moodustame argumentide massiivi t.
>> plot(cos(t))  %  Joonistame funktsiooni cos(t) graafiku.
```



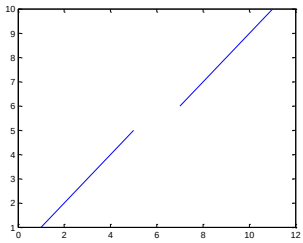
Abstsissteljel on massiivi t elementide järjekorranumbrid, mis on ühtlasi ka massiivi cos(t) järjekorranumbreiks.
Kui teha nii, nagu allpool, saadakse abstsissteljele argumendi väärtused.

```
>> figure  %  Luuakse uus jooniseaken
>> plot(t,cos(t))
```



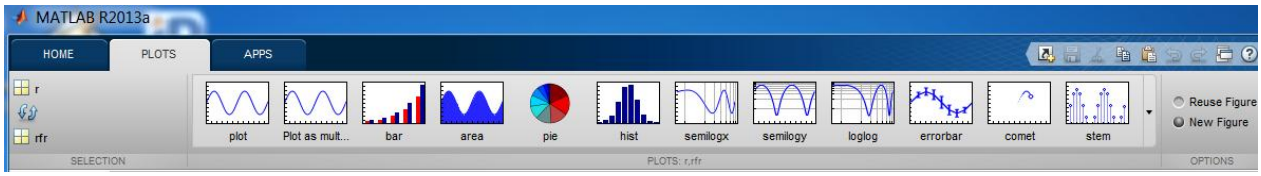
Joonise tegemisel võib kasutada suurusi **NaN** ja **Inf**.

>> **plot**([1:5,NaN,6:10]) % Massiivi [1:10] kuues element jäetakse vahele.



Tööriistariba PLOTS kasutamine

Paketis **MATLAB** on alates 2012. aastast tegevuste lihtsustamiseks olemas tööriistaribad (*Toolstrip*). Need on oluliseks uuenduseks võrreldes paketi varasemate versioonidega. Need asendavad paljuski menüüsid, täielikult aga Start-nuppu. Jooniste tegemiseks mälupiirkonna muutujatest on ette nähtud tööriistariba **Plots**:



Vasakpoolses osas (*Selection*) on näha mälupiirkonnast (*Workspace*) valitud muutujad. Erinevate jooniste valimise osa (*Plots*) on arukas: aktiveeritud on vaid antud muutujate valiku puhul võimalikud graafikuvariandid. Tööriistariba parempoolsel otsal on võimalik valida kas uue jooniseakna või juba aktiivse akna vahel.

Näiteks, kui mälupiirkonnas on massiivid **r** ja **rfr**, mille elemendid peaksid olema vastavalt abstsissteljel ja ordinaatteljel, siis tuleb Ctrl – klahvi all hoides klõpsata esmalt muutuja **r** ees ja siis muutuja **rfr** ees oleval kollasel ikoonil r . Joonis tekib, kui vajutada vastavale valikule jooniste galeriis. Käsuaknasse ja käskude ajaloo aknasse tekib ka vastav käsk, mille saab sealt kopeerida ja vajalikku kohta (m-faili näiteks) kleepida.

Kahe valitud massiivi puhul saab nende osad abstsiss- ja ordinaatteljel vahetada nooltega tööriistariba vasakpoolses osas.

Mitme või mitmedimensionaalse massiivi valikul on palju erinevaid võimalusi.

Kui joonist mingil põhjusel teha ei saa, antakse käsuaknas sellest veatea(de)tega märku.

Klõps paremal pool galerii keskel asuval noolel avab kõikide võimaluste tabeli. Kogu infot erinevate graafikute ja jooniste kohta paketis **MATLAB** näeb ka siin:

http://www.mathworks.se/help/matlab/creating_plots/figures-plots-and-graphs.html

3D joonised MATLAB-is

Peamised vahendid on sellised:

- **3D kõverad**, funktsioon **plot3**.

- **Võrguga pind** (*Mesh Plot*), funktsioonid **mesh**, **meshc**, **meshz**, **waterfall**.

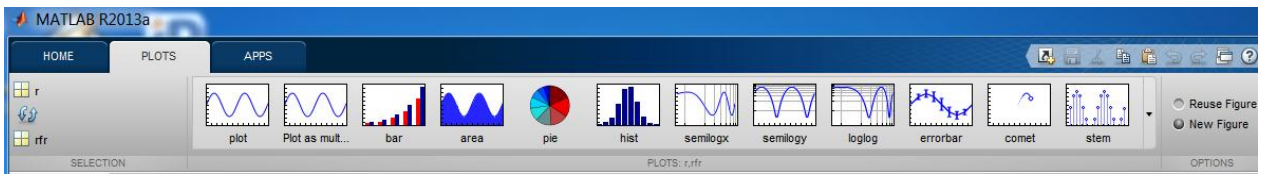
- **Varjutatud pind** (*Surface Plot*), funktsioonid **surf**, **surfc**.

- **Kontuurjoonis**, funktsioonid **contour**, **contour3**, **contourc**, **contourf**.

- **Mahujoonis**, funktsioonid **slice**,

- **Spetsialiseeritud joonised**, funktsioonid **sphere**, **cylinder**, **ribbon**.

Jooniseid saab lihtsasti teha ka globaalse tööriistariba abil:



Tuleb valida mälupiirkonnast (Workspace) muutuja(d) ja siis vastav joonise vorming.

Võrguga pind, **mesh**

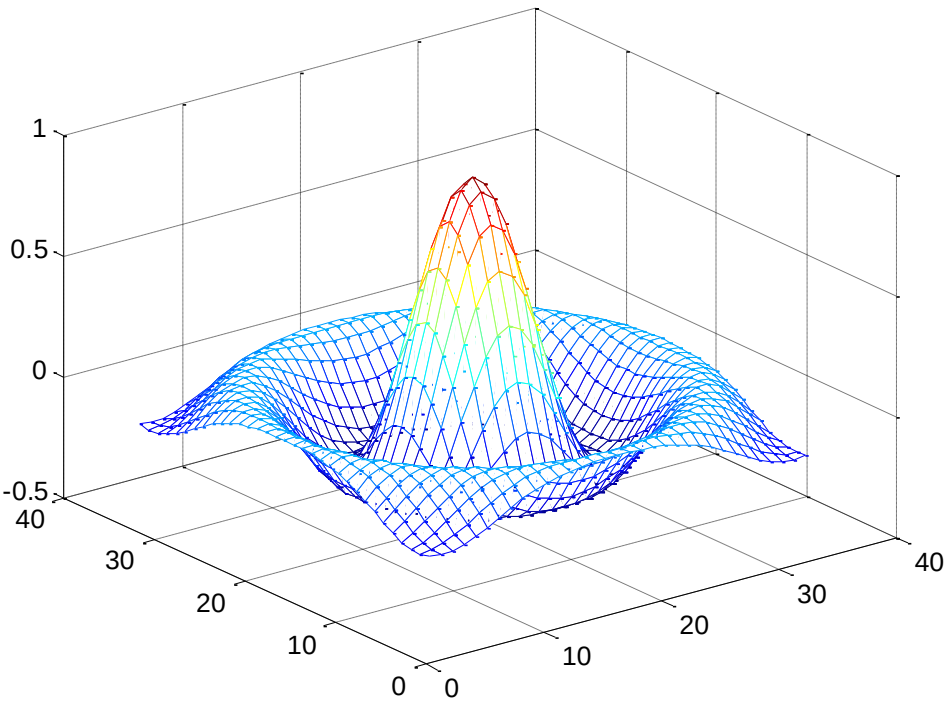
Funktsioon **mesh**(X,Y,Z) joonistab kolmedimensionaalsesse koordinaadistikku pinna, millel on näha võrk. Visualiseeritakse massiivides X, Y ja Z olevate andmete poolt defineeritud punktid. Kui X ja Y on vektorid pikkustega **length**(X) = n ja **length**(Y) = m ning [m,n] = **size**(Z), siis asuvad punktides koordinaatidega (X(k),Y(k),Z(k,j)) pinnavõrgu sõlmed. Kui X ja Y on maatriksid, siis asuvad pinnavõrgu sõlmed punktides koordinaatidega (X(k,j),Y(k,j),Z(k,j)). Andmemassiivide dimensioonid peavad olema kooskõlas.

Pinna värv on vaikimisi määratud proportsionaalselt pinna punktide kõrgustega.

Funktsioon **mesh**(Z) joonistab võrkpinna, kus X = 1:n, Y = 1:m, [m,n] = **size**(Z).

Näide:

```
>> [X,Y] = meshgrid(-8:.5:8);  % Moodustatakse maatriksid X ja Y.
>> R = sqrt(X.^2 + Y.^2) + eps;  % Funktsiooni R väärtused.
>> Z = sin(R)./R;                % Funktsiooni Z väärtused.
>> mesh(Z);                      % Joonis.
```

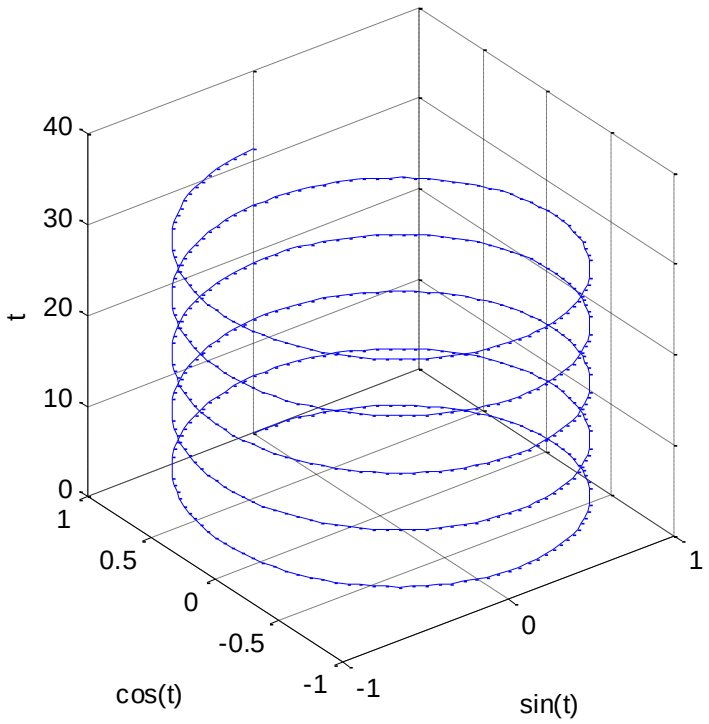


3D kõverad, funktsioon plot3

Funktsioon **plot3**(X,Y,Z) joonistab kolmedimensionaalsesse koordinaadistikku kõvera, visualiseerib massiivides X, Y ja Z olevate andmete poolt defineeritud punktid. Andmemassiivide dimensioonid peavad olema ühesugused.
Kui X, Y ja Z on ühepikkused vektorid, joonistatakse kõver punktidega (X(k),Y(k),Z(k)), kui need on maatriksid, siis võetakse koordinaadid vastavatest veergudest.

Tüüpiline näide, mille teevad läbi kõik **MATLAB**-i õppijad:

```
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t)  % Tekib joonis.
>> xlabel('sin(t)')        % Telje nimi.
>> ylabel('cos(t)')        % Telje nimi
>> zlabel('t')             % Telje nimi
>> grid on       %  Koordinaatide võrk.
>> axis square  % Teljestiku määramine.
```



Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Teema 5. Funktsioonid paketis MATLAB

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Funktsioonid on m-failid, mis sisendparameetrite väärtuste alusel leiavad väljundparameetrite väärtused etteantud algoritmi põhjal. Argumentide väärtused kirjutatakse ümarsulgudesse ja eraldatakse komadega. Õpime tundma paketiga kaasasolevaid, nn sisseehitatud funktsioone (**sin**, **cos**, **exp**, **log** jne) ja kasutaja poolt defineeritud funktsioone. Läbiv teema, uut lisandub kogu kursuse jooksul.

Teema kohta tuleb lahendada kohustuslik ülesanne, mida hinnatakse skaalal F-A (F - 1 punkt ja A 6 punkti). Kohustusliku ülesande tekst on eraldi failis. Teemakohaseks aruteluks kasutame foorumit.

Paketi MATLAB sisseehitatud funktsioonid

Paketiga MATLAB on kaasas palju nn sisseehitatud funktsioone, see tähendab programme või m-faile, mis etteantud sisendiparameetrite või –muutujate väärtuste korral leiavad vastavad väljundparameetrite väärtused ehk lihtsalt väljundväärtused. Kasutaja poolt defineeritavate funktsioonide korral tuleb eristada formaalseid parameetreid ja reaalseid. Funktsiooni sisendparameetrid kirjutatakse funktsiooni nime järele, alati ümarsulgudesse. Väljundparameetrite puhul on erinevaid süntaksivõimalusi. kui on tegemist ühe väljundmuutujaga, siis seda reeglina sulgudesse ei kirjutata.

Lisaks andmesisenditele võib funktsioonide sisendparameetrite hulka kuuluda veel optsioonidele (*Options*) ja variantide valikutele (*Features*) osundavaid parameetreid. On funktsioone, millel sisend- ja väljundparameetrid võivad puududa (**who**, **whos**). Funktsioonide reaalseteks parameetriteks võivad olla ka avaldised.

Avaldistes võib kasutada funktsiooni nime koos konkreetsete sisendparameetrite väärtustega sama moodi, nagu muutujaid või lihtparameetreid. Avaldiste väärtustamisel leitaksegi kõigepealt nende koostisse kuuluvate funktsioonide väärtused ja siis tehakse ülejäänud arvutused vastavalt tehete järjekorrale avaldises.

Loetleme mõned enamkasutatavad sisseehitatud funktsioonid.

cos(x), koosinus; abs(x) absoluutväärtus; **sin**(x), siinus; **sign**(x), signum-funktsioon, muutuja märgi leidmine;**tan**(x), tangens; **max**(x), massiivi maksimaalse väärtusega elemendi väärtuse leidmine; **acos**(x), arkuskoosinus; **min**(x), massiivi minimaalse väärtusega elemendi väärtuse leidmine; **asin**(x), arkussiinus; **ceil**(x), ümardamine üles, **floor**(x), ümardamine alla; **atan**(x), arkustangens; **exp**(x), e aste, eksponent; **round**(x), ümardamine lähima täisarvuni; **sqrt**(x), ruutjuur; **rem**(x), jagatise jääk; **log**(x), naturaallogaritm; **log10**(x), kümnendlogaritm…

Paketi käsuaknas, sisetusviiba all näeme funktsioonide otsimisvahendit. Klõps sellele avab otsinguakna.

Funktsioonide täieliku nimistu leiate siit:
http://www.mathworks.se/help/matlab/functionlist.html.

Kasutaja poolt defineeritud funktsioonid paketis MATLAB

Paketis **MATLAB** saab iga kasutaja ka ise funktsioone defineerida ja see käib päris lihtsalt.
Kui pikk võib olla selle nimi, üldiselt – kasutaja poolt defineeritud muutuja nimi, seda näitab sisseehitatud funktsioon **namelengthmax**. Funktsiooni nime kohta öeldakse **MATLAB**-i terminites *Function Handle*. ja see luuakse märgi @ abil:

```
>> h=@funktsiooninimi
```

Kasutajal on võimalus defineerida oma vajadustele vastav funktsioon kas eraldi m-failina (seda võimalust käsitleme eraldi materjalis), anonüümse funktsioonina või *inline*-funktsioonina.

Anonüümne funktsioon (*Anonymous Function*).

Ei säilitata eraldi failis, aga selle nimi seostatakse andmetüübiga function_handle. Anonüümseid funktsioone kasutatakse nagu tavalisi, kuid need võivad sisaldada vaid üht väärtustatavat avaldist. Üldine süntaks käsureal: funktsiooni nimi (handle), võrdusmärk; @; ümarsulgudes muutuja(te loetelu); avaldis, mis annab funktsiooni väärtuse arvutamise eeskirja.

```
>> ruut=@(x) x.^2;  %  Luuakse anonüümne funktsioon "ruut".

>> a=ruut(3)+ruut(4);  % Kasutamisel kirjutatakse reaalsete argumentide väärtused.

a =
```

Anonüümse funktsiooni defineerimisel võib kasutada mälupiirkonnas defineeritud parameetreid. Sama seansi vältel säilivad nende parameetrite väärtused funktsiooni definitsioonis, isegi kui vastavad parameetrid mälupiirkonnast kustutada.

```
>> a = 1.3; b = .2; c = 30;
>> parabool = @(x) a*x.^2 + b*x + c;

>> clear a b c
>> y = parabool(x)

y =

   31.5000
```

Et kasutada erinevaid kordajaid (parameetreid), tuleb funktsioon uuesti defineerida.

```
>> a = -3.9; b = 52; c = 0;
>> parabool = @(x) a*x.^2 + b*x + c;
>> y = parabool(1)

y =
   48.1000
```

Kui salvestada anonüümne funktsioon **mat**-faili, salvestatakse sinna ka definitsioonis kasutatud parameetrite väärtused ning järgmisel seansil sisselugemise järel see definitsioon taastub.

Anonüümse funktsiooni võib defineerida ka ilma sisendmuutujateta. Kasutamisel on nime järel olevad sulud vajalikud, muidu defineeritakse uus funktsiooninimi.

```
>> t = @() datestr(now);  % Kasutatakse paketti sisseehitatud vahendeid.
d = t()

d =

09-May-2013 20:06:16
```

Anonüümse funktsiooni võib defineerida mitme muutuja funktsioonina, mitme väljundiga funktsioonina.

```
>> yksfunktsioon = @(x,y) (x^2 + y^2 + x*y);

>> x = 1; y = 10;
>> z = yksfunktsioon(x,y)

z =

   111
```

On lubatud ka anonüümsete funktsioonide massiivide defineerimine. Tuleb meeles pidada, et massiivi sisestamisel interpreteeritakse tühikuid reaelementide eraldajatena, seetõttu tuleks üleliigseid tühikuid definitsiooni avaldistes vältida.

```
>> f = {@(x)x.^2;@(y)y+10;@(x,y)x.^2+y+10};

>> x = 1; y = 10;

f{1}(x)
f{2}(y)
f{3}(x,y)

ans =

     1


ans =

    20


ans =

    21
```

http://www.mathworks.se/help/matlab/matlab_prog/anonymous-functions.html.

***Inline*-funktsioon** (*Inline Function*)

Saab kasutada käsurealt loodavate objektide vahendit inline.

```
>> g=inline('2*x.^2 + 3*x + 4')

g =

     Inline function:
     g(x) = 2*x.^2 + 3*x + 4
```

Mälupiirkonnas defineeritud parameetreid ei saa nii kasutada, nagu anonüümse funktsiooni korral see on võimalik. Tundmatud suurused loetakse kõik sisendmuutujateks.

```
>> g=inline('a*x.^2 + b*x + c')

g =

     Inline function:
```

```
g(a,b,c,x) = a*x.^2 + b*x + c
```

Inline-funktsiooni kasutamine on sama, nagu teiste funktsioonide puhul.

```
>> g=inline('2*x.^2 + 3*x + 4');

>> a=g(3) + g(4)

a =

   79
```

Polünoomid paketis MATLAB

Paketis MATLAB saab polünoome lihtsalt luua ja kasutada, selleks on spetsiaalsed sisemised vahendid paketis. Polünoomide all me mõistame siinkohal algebralisi polünoome, ka astmefunktsioone (märgime, et võib rääkida ka näiteks trigonomeetrilistest polünoomidest). Kujul

$$P_n(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0$$

antud funktsiooni nimetamegi n-järku polünoomiks muutuja n suhtes. Suurusi a_n, a_{n-1}, ... , a_1, a_0 nimetatakse polünoomi kordajateks, suurust a_n pealiikme kordajaks. Reeglina oletame, et polünoomi kordajad on reaalarvud ehk reaalsed. Polünoom P_n on oma kordajatega üheselt määratud ja seda on kasutatud ka paketis MATLAB. Nimelt esitatakse ja käsitletakse polünoome nende kordajate vektoritena.

Näiteks polünoom $P_3(x) = x^3 - 2 \cdot x - 5$ esitatakse MATLAB-is vektorina

```
>> p=[1 0 -2 -5];
```

Märgime, et toodud konkreetne polünoom on ajaloolise väärtusega, kuna selle näitel demonstreeriti Prantsuse Akadeemiale esimest korda Newtoni iteratsioonimeetodit funktsiooni nullkohtade leidmiseks.
Polünoomi väärtuse leidmine toimub funktsiooni **polyval** abil.

```
>> polyval(p,3)  % Leitakse polünoomi p väärtus kohal x = 3.

ans =

   16
```

Polünoomide väärtusi saab leida ka maatriksite korral funktsiooniga **polyvalm**.

```
>> X = [2 4 5; -1 0 3; 7 1 5];
>> Y = polyvalm(p,X)

Y =

  377   179   439
  111    81   136
  490   253   639
```

Polünoomid on üheselt määratud ka oma juurtega ehk nullkohtadega. Teatavasti on n-järku polünoomil kompleksarvude korpuses täpselt n nullkohta, kui arvestada nende kordsust. Juuri saab leida MATLAB-i funktsiooniga **roots**. Kokkuvõttes: polünoomid on üheselt määratud nii oma kordajate vektoriga kui ka juurte vektoriga.

```
>> p = [1    -6    11    -6]  % Defineerime polünoomi.

p =

     1    -6    11    -6

>> r=roots(p)  % Leiame selle polünoomi juured.

r =

    3.0000
    2.0000
    1.0000
```

Juurte ehk nullkohtade järgi saab polünoomi kordajad kätte funktsiooniga **poly**.

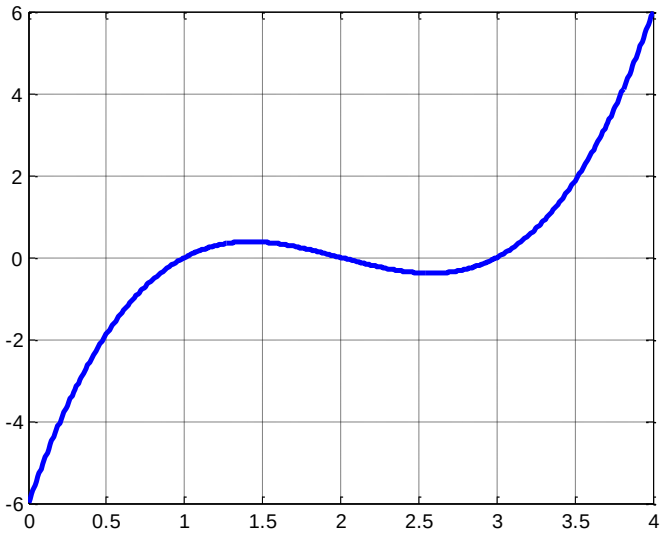
```
>> p1=poly(r)

p1 =

    1.0000   -6.0000   11.0000   -6.0000
```

Teeme ka selle funktsiooni graafiku.

```
>> x=0:0.01:4;
>> y=polyval(p1,x);
>> plot(x,y,'Linewidth',3)
>> grid
```



Funktsioonid eraldi m-failina

Funktsioonide kasutajapoolse defineerimise paindlikem võimalus on luua ja kasutada funktsioonitüüpi m-faile, milledel on sisendparameetrid ja väljundparameetrid. Ning milles väärtustatavate avaldiste hulk pole piiratud. Muutujad sellises m-failis on lokaalsed, st informatsiooni vahendamine mälupiirkonnaga käib vaid sisend- ja väljundparameetrite vüi määratluse **global** kaudu. Sisendparameetrid tuleb funktsiooni poole pöördumisel väärtustada, funktsioon väärtustab väljundparameetri(d).

Funktsiooni päise üldkuju:

```
function [väljund1,väljund2,...] = funktsiooninimi(sisend1,sisend2,...)
%  Esimesed kommentaariread väljastatakse päringuga help funktsiooninimi
%  sisend1,sisend2,... – sisendmuutujate nimede loetelu.
%  väljund1,väljund2,... – väljundmuutujate nimede loetelu.

…  % Siia tuleb tippida täidetavad avaldised.
```

See on nn. primaarne funktsioon (*Primary Function*), mis peab olema salvestatud faili nimega "funktsiooninimi.m". Väljakutsumine (käsurealt või skriptist):

```
>>funktsiooninimi(väärtus1,väärtus2,...) ;
```

Kuna väljaspoolt faili nimega "funktsiooninimi.m" on kättesaadav vaid primaarne funktsioon siis järgmised samasse faili defineeritud funktsioonid on kättesaadavad vaid promaarsele funktsioonile. Need on alamfunktsioonid (*Nested Function, Subfunction*).

```
function u = A(x, y) % Primaarne funktsioon

B(x, y); % Funktsiooni B saab vaid siit välja kutsuda, mitte käsurealt.
D(y); % Funktsiooni D saab vaid siit välja kutsuda, mitte käsurealt.

end % Alamfunktsioonide olemasolul peaks siin end olema.

function v = B(x, y)
% Funktsiooni A alamfunktsiooni B definitsiooni algus.

C(x); % Funktsiooni C saab vaid siit välja kutsuda, mitte käsurealt või A-st.

D(y); % Sama taseme alamfunktsioon, nagu B.

        function t = C(x) % Funktsiooni B alamfunktsioon (nested in B)
D(x);  % Funktsiooni D poole pöördumine.

        end % Käsud end igal pool kohustuslikud; siin lõpeb C.
end % Käsud end igal pool kohustuslikud; siin lõpeb B.

function s = D(x) % Funktsiooni A alamfunktsioon (subfunction).
% Sama taseme alamfunktsioon, nagu B.

E(x); % Funktsiooni E poole pöördumine.

        function q = E(x) % Funktsiooni D alamfunktsioon (nested in D).
        end % Käsk end kohustuslik; siin lõpeb E.
end % Käsk end kohustuslik; siin lõpeb D.
```

Funktsioonitüüpi m-failide poole pöördumine käib nii, nagu teistegi funktsioonide poole pöördumine. Sisendparameetriteks peavad olema tegelike väärtustega muutujad või arvutatavate väärtustega avaldised, väljundparameetreid võib lülitada teistesse avaldistesse.

Funktsioonid kui m-failid peavad olema salvestatud töökausta. Nende poole pöördumine on võimalik kõigi failide poolt, millest on vaadeldavasse töökausta juurdepääs.

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Teema 6. Programmeerimine paketis MATLAB

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Võimalus kirjutada ise programme mitmesuguste ülesannete korduvaks lahendamiseks on paketi üks olulisi eeliseid. Programmide ehk m-failide koostamiseks kasutatakse spetsiaalset toimetiakent. Läbiv fundamentaalne teema, uut lisandub kogu kursuse jooksul.

Teema kohta tuleb lahendada kohustuslik ülesanne, mida hinnatakse skaalal F-A (F - 1 punkt ja A 6 punkti). Kohustusliku ülesande tekst on eraldi failis. Teemakohaseks aruteluks kasutame foorumit.

Programmid ehk m-failid kujutavad endast järjestikku kirjutatud paketi sisestusi, mis salvestatakse eraldi failidena töökaustades ning täidetakse vajadusel korraga. Programme koostatakse ja redigeeritakse eraldi toimetiaknas (*Editor*).

Töökaust (*Current Directory*) seadistatakse seansi alguses vastavas aknas.

Programmifailid salvestatakse töökaustas failidena, mille laiendiks on „.m" ja käivitatakse käsuaknast programmi nimega. Kõik muutujad lihtprogrammis e skriptis on globaalsed, st. kättesaadavad tööpiirkonnas (*Workspace*).

Näiteks, kui töökaustas on programm nimega „programm.m", siis selle käivitamine toimub käsuaknast sisestusega

```
>>programm
```

Kommentaarid, väga vajalikud märkmed programmides ja ka ülesannete lahendustes, algavad protsendimärgiga, st sümboliga „%".

```
>> % Järgnevalt käivitame programmi nimega „programm.m".
>>programm  % Käivitame skripti „programm.m".
```

Programmeerimisel on tihti vaja muutujate väärtusi võrrelda. Võrdlustehete tulemus(t)eks on loogilist tüüpi muutuja väärtustega kas tõene (*true*) või väär (*false*); numbriliselt esitatakse need väärtustena 1 ja 0, vastavalt. Vastavad võrdlusmärgid (võrdlustehted) on paketis **MATLAB** sellised:

a = = b on tõene siis, kui a ja b on võrdsed;
a ~ = b on tõene siis, kui a ja b ei ole võrdsed;
a < b on tõene siis, kui a on väiksem kui b;
a < = b on tõene siis, kui a on väiksem muutujast b või võrdne muutujaga b;
a > b on tõene siis, kui a on suurem kui b;
a > = b on tõene siis, kui a on suurem muutujast b või võrdne muutujaga b;

Komplekssete väärtustega operandide puhul tehakse võrdlustehteid < , >, <= ja >= vaid reaalosade vahel, tehete = = ja ~ = korral arvestatakse ka imaginaarosi.

Võrdlustehteid tehakse ka massiivide vahel. Tulemuseks on vastava dimensiooniga loogilist tüüpi muutujatega massiiv.

Näide.

```
>> X = 5; X >= [1 2 3; 4 5 6; 7 8 10]

ans =

    1    1    1
    1    1    0
    0    0    0

>> Y=all(X)  % Funktsioon all kontrollib, kas kõik elemendid on nullist suuremad.

Y =

    1

>> any(X(:) < 0)  % Funktsioon any kontrollib, kas mõnigi element vastab
tingimusele.

ans =

    0

>> Z= [1 2 3; 4 5 6; 7 8 10];

>> find(Z>5)  % Funktsioon find leiab elementide nn lineaarsed indeksid.

ans =

    3
    6
    8
    9
```

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Tsüklid paketis MATLAB

Programmide puhul on olulisel kohal tsüklite organiseerimine arvutustes. Tsüklites täidetakse mingeid programmi osi korduvalt, kusjuures igal täitmisel võivad arvutuses olla erinevad andmed ja varieeruda võib ka algoritm. Tsüklioperaatoreid on kaks : **for** ja **while**. Paketis **MATLAB** on vajalik on käsu **end** kasutamine tsükli lõpetamiseks. Toome näited koos väljunditega.

```
for k = 1:4  % See on for-tsükkel. k on tsükli täitmiste loendur,
             % k on tsükli täitmiste loendur, tsüklit täidetakse  k(end)=4 korda.

disp(k); % Tsükli ainukeseks korralduseks on tsükliloenduri esitamine käsuaknas.

end      % Tsükli lõpudirektiiv.
```

```
   1

   2

   3

   4
```

```
p=1;  % Tsüklitingimus tuleb algväärtustada.

while p<=4     % See on while-tsükkel.
               % Tsüklit täidetakse siis, kui kehtib p<=4.

disp(k)  % Tsükli esimeseks korralduseks on p esitamine käsuaknas.

p=p+1;  % Tsükli teiseks korralduseks on p suurendamine ühe võrra.

end
```

Tsüklite puhul kasutatakse veel järgmisi juhtimiskorraldusi**.**

break – lõpetab tsükli (nii **for** – kui **while** – tsükli) täitmise.
pause – peatab programmi (seejuures ka nii **for** – kui **while** – tsükli) täitmise. Programmi töö jätkub sisestusklahvile vajutamise järel. Kasutatakse näiteks vahetulemuste vaatamise võimaldamiseks.

Tingimuslikuks direktiiviks on **MATLAB**-is **if**, selle mõjupiirkonna sees võib kasutada veel käske **else** ja **elseif** (viimane – kindlasti ilma tühikuta **if** ees).

Näide – arvu absoluutväärtuse leidmine.

```
x=7;

if x>=0

y=x;

else

y=-x;

end
```

MATLAB-is saab programmi kulgemist seada sõltuvusse mingite tingimuste kompleksist veel käsuga **switch** (lüliti).

NB! Programmifailides on tungivalt soovitatav kasutada kommentaare (juhiseid lugejale, mis kirjeldavad programmi tööd), et programmi kasutajatel, aga ka autoril enesel mõne aja pärast, oleks võimalik programmi loogikast kiiresti aru saada.
Kommentaariks on kõik, mis algab sümboliga **%** ja jääb programmi teksti reas paremale poole seda sümbolit.
Kommentaarides esitatakse

- programmi otstarve;
- funktsiooni puhul selle väljakutsesse kuuluvate muutujate tähendused;
- väljundsuuruste tähendused;
- kommentaaris võidakse selgitada programmi kasutamist;
- antakse programmi suuremate osade pealkirjad;
- seletatakse programmi ridade otstarvet ja tööd
- jms

Kommentaarid on kohustuslikud iseseisvate ülesannete lahenduste esitamisel.

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Võrdlustehted paketis MATLAB

Tihti on vaja muutujate väärtusi võrrelda. Võrdlustehete tulemus(t)eks on loogilist tüüpi muutuja väärtustega kas tõene (*true*) või väär (*false*); numbriliselt esitatakse need vastavalt väärtustena 1 ja 0. Vastavad võrdlusmärgid (võrdlustehted) on paketis **MATLAB** sellised:

a = = b on tõene siis, kui a ja b on võrdsed;
a ~ = b on tõene siis, kui a ja b ei ole võrdsed;
a < b on tõene siis, kui a on väiksem kui b;
a < = b on tõene siis, kui a on väiksem muutujast b või võrdne muutujaga b;
a > b on tõene siis, kui a on suurem kui b;
a > = b on tõene siis, kui a on suurem muutujast b või võrdne muutujaga b;

Komplekssete väärtustega operandide puhul tehakse võrdlustehteid < , >, <= ja >= vaid reaalosade vahel, tehete = = ja ~ = korral arvestatakse ka imaginaarosi.

Võrdlustehteid tehakse ka massiivide vahel. Tulemuseks on vastava dimensiooniga loogilist tüüpi muutujatega massiiv.

Näide.

```
>> X = 5; Y= [1 2 3; 4 5 6; 7 8 10]; Z=X>=Y

Z =

     1     1     1
     1     1     0
     0     0     0

>> all(Z(:))  % Funktsioon all kontrollib, kas kõik elemendid on nullist suuremad.

ans =

     0

>> any(Z < 0)  % Funktsioon any kontrollib, kas mõnigi element vastab tingimusele.

ans =

     0     0     0
```

Pange tähele: skalaari ja vektori puhul teevad **all** ja **any** otsuse kõigi andmeelementide kohta kokku, juba maatriksite puhul aga maatriksi iga veeru kohta eraldi.

```
>> Z= [1 2 3; 4 5 6; 7 8 10];

>> find(Z>5)  % Funktsioon find leiab nn lineaarsed indeksid elementidele,
```

```
        % mis rahuldavad seatud tingimust. Lineaarne indeks on elemendi
    järjenumber
        % vektoris, mis saadakse maatriksi veergude üksteise alla
    paigutamisel.

ans =

    3
    6
    8
    9
```

Võrdlustehteid saab teha ka funktsioonide abil:

eq Võrdumise selgitamine
ge Võrdluse >= selgitamine
gt Võrdluse > selgitamine
le Võrdluse <= selgitamine
lt Võrdluse < selgitamine
ne Mittevõrdumise selgitamine
isequal Massiivide võrdumise selgitamine
isequaln Massiivide võrdumise selgitamine, väärtused NaN loetakse võrdseiks

```
>> Z= [1 2 3; 4 5 6; 7 8 10];

>> eq(5,Z)

ans =

    0    0    0
    0    1    0
    0    0    0
```

NB! Kõnesolevad funktsioonid sobivad vaid arvude võrdlemiseks. Tekstide ja sümbolite võrdlemiseks kasutage funktsioone **strcmp, strcmpi, strncmpi**.

Hädavajalikud teadmised. Peab teadma, kuidas

- luua ja muuta m-faile (skripte ja funktsioone);
- selgitada erinevusi skripti ja funktsiooni vahel;
- kommenteerida programme;
- kasutada käsku **disp**, et väljastada m-failist andmeid;
- kasutada võrdlusoperaatoreid (<, <=, >, >=, ==, ~=);
- kasutada loogilisi operaatoreid (&, |, ~);
- kirjutada **for** ja **while** tsükleid;
- kirjutada tingimuslikke konstruktsioone: **if**...**end**, **if**...**elseif**...**end**, **if**...**elseif**...**else**...**end**, **switch**…**case**…**end**;
- kasutada käske **break** ja **return**;
- asendada tavalised tsüklid võimalusel vektoriseeritutega;
- enne tsükleid reserveerida mälu massiividele, mis tsüklis paisuvad.
- kirjutada programmi heas stiilis – kommentaarid, tühjad read, taane;
- kavandada programmi ja nende hierarhiat blokkskeemi(de) abil;
- siluda programme (**error**, **warning**, **pause**, **keyboard**);
- kasutada interaktiivseid silumisvahendeid.

Paketi MATLAB võimaluste kasutamine

MATLAB on sisuliselt kõrgtaseme programmeerimiskeel paljude sisse ehitatud funktsioonide ja võimalustega. Nagu kõikide selliste keskkondade puhul, nii on ka **MATLAB**'is võimalik kõrgtasemel konstruktsioone asendada madalatasemelistega. Näide selle kohta, vektori elementide ruutu tõstmine.

Madalatasemeline, jõumeetod:

```
>> x = [ 1 2 3 4 5 ]; %  Defineeritakse vektor.
>> y = zeros(size(x)); % Luuakse uus vektor.
>> for i = 1 : numel(x)  %  Algab tsükkel: iga indeksi korral…
>> y(i) = x(i)^2;  %  Arvutatakse vektori x elementide ruudud.
>> end   %  Tsükli lõpp.
```

Sama asi kõrgtasemelises realisatsioonis, vektoriseeritud kujul:

```
>>x = [ 1 2 3 4 5 ]; %  Defineeritakse vektor.
>>y = x.^2;  %  Tõstetakse kõik elemendid ruutu elemendikaupa tehtega.
```

Siiski tuleb iga kord analüüsida, kas kasutada kõrg- või madala tasemega realisatsiooni. Madalatasemelist varianti kasutavad kõik või vähemalt valdav enamus programmeerimiskeeltest ning seetõttu on nende „transportimine“ lihtne. Samuti on lihtsam programmeerijatel ümber kvalifitseeruda.

Kõrgtaseme funktsioonide peamine eesmärk on teha rohkem kui madalatasemelised ekvivalendid. Vastav kood on kompaktsem, lühem, vajab vähem programmeerijatööd ning reeglina on lühema programmi koostamisel ka väiksem võimalus vigu teha. Lühemast programmist saab ka kergemini ülevaate seda uurides. **MATLAB**'is töötavad kõrgtaseme programmid ka kiiremini kui madala tasemega koodid. Tuleb veel silmas pidada, et kõrgtasemel koodid on mõnikord nii lühikesed, et muutuvad arusaamatuteks, obskuurseteks. See eeldab nende programmide kommenteerimist ja dokumenteerimist. Kõrgtaseme koodid nõuavad mõneti teistsuguseid mõtteviise, need arenevad praktiseerides ja ainult nii. Üldine nõuanne kõrg- ja madalatasemelise realisatsioonide vahel valimiseks on: pole mõtet kulutada kolme tundi selleks, et optimiseerida koodi eesmärgiga programmi tööajas minutiline võit saavutada, enne kui pole selge, et olemasoleva programmi jooksutamisest selle kolme tunni vältel vajalikku tulemust ei saavutata.

Teema 7. Lineaaralgebra paketiga MATLAB

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Lineaarse võrrandisüsteemi lahendamine on põhiülesanne, mille juurde matemaatiliste mudelite rakendamine varem või hiljem toob. Vaatleme lineaarsete võrrandisüsteemide ja teise lineaaralgebra ülesannete lahendamist paketiga.

Teema kohta tuleb lahendada kohustuslik ülesanne, mida hinnatakse skaalal F-A (F - 1 punkt ja A 6 punkti). Kohustusliku ülesande tekst on eraldi failis. Teemakohaseks aruteluks kasutame foorumit.

Lineaarsed võrrandisüsteemid

Ülesande üldine püstitus.

Olgu antud n·n maatriks A komponentidega a_{ij}, i, j = 1, 2,…, n ja vektor b = $(b_1, b_2,\ldots,b_n)^T$, vabaliikmete vektor.

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \ldots & a_{3,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & a_{n,3} & \ldots & a_{n,n} \end{bmatrix}, a_{i,j} \in \Re \; b=\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \ldots \\ b_n \end{bmatrix}, b_i \in \Re$$

Olgu tarvis leida vektor x = $(x_1, x_2,\ldots,x_n)^T$, et oleks rahuldatud (vektor)võrdus

$$A\cdot x = b.$$

See seos esitab **lineaarse algebralise võrrandisüsteemi** nn. vektorkujul, st. tegemist on kahe vektori (A·x ja b) vastavate komponentide vaheliste võrdustega.

Maatriksi A **regulaarsus** (st. maatriksi A determinandi erinemine nullist)

$$\det(A)=\begin{vmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \ldots & a_{3,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & a_{n,3} & \ldots & a_{n,n} \end{vmatrix} \neq 0$$

garanteerib võrrandisüsteemi ühese lahenduvuse. Seda üheselt määratud lahendit tähistame edaspidi sümboliga $x^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$. Maatriksit A nimetatakse **võrrandisüsteemi maatriksiks**, ka **kordajate maatriksiks**; vektorit b – **vabaliikmete vektoriks**; x on **tundmatute vektor**.

Lineaarsete võrrandisüsteemide lahendamiseks kasutatakse täpseid meetodeid ja iteratsioonimeetodeid. Tuntuim täpne meetod on Gaussi elimineerimismeetod. See annab tulemuse $n^3/3 \cdot (n^2+3 \cdot n-1)$ tehtega.

Et lahendada võrrandite süsteem $A \cdot x = b$ Gaussi elimineerimismeetodiga , tuleb

- Faktoriseerida A, st viia maatriks A kujule $A = P \cdot L \cdot U$, kus P on permutatsioonimaatriks, L on ühtedest koosneva diagonaaliga alumine kolmnurkne maatriks ja U on mittesingulaarne ülemine kolmnurkmaatriks. Permutatsioonimaatriksiks nimetatakse maatriksit P, mis saadakse ühikmaatriksist ridade vahetamise (permuteerimise) tagajärjel.

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{2,1} & 1 & 0 & \dots & 0 \\ l_{3,1} & l_{3,2} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & 1 \end{bmatrix}, \det L = 1;$$

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ 0 & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ 0 & 0 & u_{3,3} & \dots & u_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & u_{n,n} \end{bmatrix}, \det U \neq 0.$$

- Lahendada süsteem $P \cdot L \cdot U \cdot x = b$ vektori $L \cdot U \cdot x$ suhtes, permuteerides vektori b komponendid.
- Lahendada süsteem $L \cdot U \cdot x = P^{-1} \cdot b$ vektori $U \cdot x$ suhtes,
- $U \cdot x = L^{-1} \cdot (P^{-1} \cdot b)$ (*forward sustitution*).
- Lahendada süsteem $U \cdot x = L^{-1} \cdot (P^{-1} \cdot b)$ vektori x suhtes,
  $x = U^{-1} \cdot (L^{-1} \cdot P^{-1} \cdot b)$ (*backward substitution*).

Kehtivad järgmised teoreetilised tulemused.

Olgu P_1, P_2 ja P permutatsioonimaatriksid (dimensiooniga $n \cdot n$).

Siis

- $P \cdot X$ on maatriks, mis saadakse maatriksist X selle ridade vahetamise teel ja $X \cdot P$ on sama veergude vahetamise järel.
- $P^{-1} = P^T$.
- $\det(P) = \pm 1$.
- $P_1 \cdot P_2$ on samuti permutatsioonimaatriks.

TEOREEM. Kui $n \cdot n$ maatriks A ei ole singulaarne, siis leiduvad permutatsioonimaatriks P (ühikmaatriks, mille read on vahetatud), mittesingulaarne alumine kolmnurkmaatriks L ja mittesingulaarne ülemine kolmnurkmaatriks U sellised, et $A = P \cdot L \cdot U$.

Skemaatiliselt veelkord, süsteemiga $A \cdot x = b$ ekvivalentset süsteemi $P \cdot L \cdot U \cdot x = b$ lahendatakse sellises järjekorras:

$L \cdot U \cdot x = P^{-1} \cdot b = P^T \cdot b$ (vahetatakse b komponendid);
$U \cdot x = L^{-1} \cdot (P^T \cdot b)$ (forward substitution);
$x = U^{-1} \cdot (L^{-1} \cdot P^T \cdot b)$ (back substitution).

Gaussi elimineerimismeetodi realiseerimisel kasutatakse ümardamisvigade vältimiseks peaelementide väljaeraldamist.

Peaelementide väljaeraldamine (*pivoting*).
Osaline peaelementide kasutamine (*partial pivoting*). Kasutatakse suurimat elementi igas veerus; $L \cdot U = P \cdot A$. Seda võtet on kasuttud paketis **MATLAB**.

Täielik peaelementide kasutamine (*complete pivoting*). Leitakse igal elimineerimissammul suurim element kogu maatriksis; $L \cdot U = P \cdot A \cdot Q$.
Peaelemente ei eraldata välja erijuhtudel:

Kui maatriks A on domineeriva peadiagonaaliga:

$$\left|a_{j,j}\right| \geq \sum_{i \neq j} a_{i,j}$$

Kui maatriks A on positiivselt määratud:

$A^T = A$; $x \cdot A \cdot x^T > 0$ iga $x > 0$ korral.

Lineaarsed võrrandisüsteemid
Iteratsioonimeetodid

Ülesande üldine püstitus.

Olgu antud n·n maatriks A komponentidega a_{ij}, i, j = 1, 2,…, n ja vektor b = $(b_1, b_2,\ldots,b_n)^T$, vabaliikmete vektor.

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \ldots & a_{3,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & a_{n,3} & \ldots & a_{n,n} \end{bmatrix}, a_{i,j} \in \Re \; b=\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \ldots \\ b_n \end{bmatrix}, b_i \in \Re$$

Olgu tarvis leida vektor x = $(x_1, x_2,\ldots,x_n)^T$, et oleks rahuldatud (vektor)võrdus

$$A\cdot x = b.$$

Suurte ülesannete korral ümardamisvigade vältimiseks, aga ka täpsuse tõstmiseks juba leitud lahendite korral lineaarsele võrrandisüsteemile rakendatakse iteratsioonimeetodeid e järjestikuste lähendite meetodeid.

Iteratsioonimeetodi idee: valitakse alglähend x^0, sellest lähtuvalt konstrueeritakse lähendite jada x^i , i = 1, 2,… , selline, mis koondub võrrandisüsteemi täpseks lahendiks x^* .

Iteratsioonimeetodi rakendamiseks peab algse kuju A·x = b teisendama vastava iteratsioonimeetodi rakendamiseks sobivale kujule.

Richardsoni iteratsioonimeetod e harilik iteratsioonimeetod. Rakendamiseks sobiv kuju on

$$x = B\cdot x + c.$$

Arvutusskeem:

$$x^{i+1} = B\cdot x^i + c,\ i = 1, 2, \ldots .$$

Hariliku iteratsioonimeetodi realiseerimise üks variant:

$$x^{i+1} = x^i + (b - A\cdot x^i) = (E - A)\cdot x^i + b \quad \text{st.} \quad B = E - A.$$

Kui tähistada $e^i = x - x^i$ saab selle ümber kirjutada nii:

$$x - x^{i+1} = x - x^i - (Ax - Ax^i) \quad \text{ehk} \quad e^{i+1} = e^i - Ae^i = (E - A)e^i$$

$$\|e^{i+1}\| \le \|(E - A)e^i\| \le \|E - A\| \, \|e^i\| \le \|E - A\|^2 \, \|e^{i-1}\| \le \cdots \le \|E - A\|^i \, \|e^0\| .$$

Koondumiseks on tarvis, et ($\|e^i\| \to 0$) ehk et $\|E - A\| < 1$.

Gauss-Seideli iteratsioonimeetod:

$$x^{i+1} = (D - L)^{-1}(Ux^i + b) ,$$

Kus $-L$ on maatriksi A alumine ja $-U$ ülemine kolmnurkne osa; D on diagonaal.

Täpsemalt välja kirjutades:

$$\begin{cases} x_1^{(i+1)} = (b_1 - a_{1,2}x_2^{(i)} - a_{1,3}x_3^{(i)} - a_{1,4}x_4^{(i)} - \ldots - a_{1,n}x_n^{(i)}) / a_{1,1}, \\ x_2^{(i+1)} = (b_2 - a_{2,1}x_1^{(i+1)} - a_{2,3}x_3^{(i)} - a_{2,4}x_4^{(i)} - \ldots - a_{2,n}x_n^{(i)}) / a_{2,2}, \\ x_3^{(i+1)} = (b_3 - a_{3,1}x_1^{(i+1)} - a_{3,2}x_2^{(i+1)} - a_{3,4}x_4^{(i)} - \ldots - a_{3,n}x_n^{(i)}) / a_{3,3}, \\ \ldots \\ x_n^{(i+1)} = (b_n - a_{n,1}x_1^{(i+1)} - a_{n,2}x_2^{(i+1)} - a_{n,3}x_3^{(i+1)} - \ldots - a_{n,n-1}x_{n-1}^{(i+1)}) / a_{n,n}. \end{cases}$$

Näited lineaarse võrrandisüsteemi lahendamise kohta

Näide 1. $\begin{cases} 2x + 3y - z = 9, \\ 2y - z = 2, \\ 3z = 12. \end{cases}$ $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$.

Näide 2. $\begin{cases} 3x - 2y = 2, \\ x + y = 9. \end{cases}$ $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$.

Näide 3. Gaussi elimineerimismeetod, kolmnurksele kujule viimine.

$$\begin{cases} 2x-3y+z=-5, \\ 3x+2y-z=7, \\ x+4y-5z=3. \end{cases} \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}.$$

$$\left(\begin{array}{ccc|c} 2 & -3 & 1 & -5 \\ 3 & 2 & -1 & 7 \\ 1 & 4 & -5 & 3 \end{array}\right) \to \text{R2-3*R3, 2*R3-R1} \to \left(\begin{array}{ccc|c} 2 & -3 & 1 & -5 \\ 0 & -10 & 14 & -2 \\ 0 & 11 & -11 & 11 \end{array}\right) \to$$

$$\text{R3/11, 10*R3+R2} \to \left(\begin{array}{ccc|c} 2 & -3 & 1 & -5 \\ 0 & -10 & 14 & -2 \\ 0 & 0 & 4 & 8 \end{array}\right) \to \text{Süsteem on kolmnurksel kujul, tehakse}$$

tagasisamm.

Näide 4. Gauss-Jordani elimineerimismeetod. Süsteemi maatriks viiakse diagonaalkujule.

$$\begin{cases} 3x+4y+z=6, \\ 2x-y+2z=-5, \\ x+3y-z=9. \end{cases} \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ -4 \end{pmatrix}.$$

$$\left(\begin{array}{ccc|c} 3 & 4 & 1 & 6 \\ 2 & -1 & 2 & -5 \\ 1 & 3 & -1 & 9 \end{array}\right) \to \text{R2-2*R3, 3*R3-R1} \to \left(\begin{array}{ccc|c} 3 & 4 & 1 & 6 \\ 0 & -7 & 4 & -23 \\ 0 & 5 & -4 & 21 \end{array}\right) \to \text{5*R2+7*R3,}$$

R2←→R3

$$\to \left(\begin{array}{ccc|c} 3 & 4 & 1 & 6 \\ 0 & 5 & -4 & 21 \\ 0 & 0 & -8 & 32 \end{array}\right) \to \text{-R3/8, R2+4*R3} \to \left(\begin{array}{ccc|c} 3 & 4 & 1 & 6 \\ 0 & 5 & 0 & 5 \\ 0 & 0 & 1 & -4 \end{array}\right) \to \text{R2/5, R1-4*R2}$$

$$\to \left(\begin{array}{ccc|c} 3 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -4 \end{array}\right) \to \text{R1-R3} \to \left(\begin{array}{ccc|c} 3 & 0 & 0 & 6 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -4 \end{array}\right) \to \text{R1/3} \to \left(\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -4 \end{array}\right).$$

Näide 5. „Halb“ süsteem.

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$ Maatriksi LU- faktoriseering: $$\begin{bmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon & 1 \\ 0 & 1-\frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Teeme läbi Gaussi elimineerimismeetodi peaelementi välja eraldamata.
Esialgu ε väike arv.

$$\begin{bmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, saame... y_1 = 1; y_2 = 1 - \frac{1}{\varepsilon}$$

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, ...saame... x_1 = 0, x_2 = 1$$

Kui võtta ε väga väikeseks, $\varepsilon < 2^{-53}$, siis saame lahendiks $x_1 = x_2 = 1$.

Cholesky faktoriseerimismeetod

Olgu antud sümmeetriline positiivselt määratud maatriks A. Meenutame, et maatriks A on positiivselt defineeritud, kui

$$x^* \cdot A \cdot x > 0 \ \forall x \in \mathbb{C}^n, x \neq 0.$$

Niisuguse maatriksi saab esitada kujul

$$A = R^T \cdot R,$$

kus R on ülemine kolmnurkne maatriks. Sellist esitust nimetatakse Cholesky faktoriseeringuks. Maatriks R kannab maatriksi A Cholesky faktori nimetust.

Maatriksi faktoriseerimine on abiks sellise maatriksiga lineaarse võrrandisüsteemi lahendamisel. Esitame kaks algoritmi Cholesky faktoriseeringu leidmiseks.

Cholesky faktori leidmise algoritm 1: maatriksi R veerukaupa arvutamine.

for j = 1 : n
for i = 1 : j − 1
r(i,j) = (a(i,j) – SUM{k = 1,…,i-1 }r(k,i)r(k,j))/r(i,i);
end
r(j,j) = SQRT(a(j,j) – SUM{k = 1,…,j-1 }r^2(k,j));
end
Algoritm 1, veerukaupa faktoriseerimine annab Cholesky faktorid ka maatriksi A peaalammaatriksitele. Reakaupa faktoriseerimise algoritm pöörab arvutusjärjekorra.

Cholesky faktori leidmise algoritm 2: maatriksi R reakaupa arvutamine.

for i = 1 : n
r(i,i) = SQRT(a(i,i) – SUM{k = 1,…,i-1 }r^2(k,i));
for j = i + 1 : n
r(i,j) = (a(i,j) – SUM{k = 1,…,i-1 }r(k,i)r(k,j))/r(i,i);
end
end

Toodud kaks Cholesky algoritmi versiooni on samaväärsed, st. leiavad ühe ja sama Cholesky faktori.

Andre-Louis Cholesky (1875–1918) oli Prantsuse ohvitser, kes tegeles geodeesiaga ning kes tegi vaatlusi Kreetal ja Põhja- Aafrikas just enne I Maailmasõda. Tema poolt väljatöötatud algoritmi publitseeris temanimelisena postuumselt tema alluv ohvitser Benoit 1924. aastal.

Võrrandisüsteemi A·x = b lahendamiseks Cholesky meetodil tuleb kõigepealt leida maatriksi A Cholesky faktoriseering A = R^T·R, seejärel lahendada R^T·y = b (*forward substitution*) ning siis võrrandisüsteem R·y = y (*back substitution*).

Paketis **MATLAB** leiab Cholesky faktori funktsioon **chol**.

Teisi lineaaralgebra ülesandeid

Determinandi arvutamine.

On antud ruutmatriks A. On vaja leida maatriksi A determinandi väärtus.

Selle ülesande lahendamiseks on võimalik kasutada **Gaussi elimineerimis-meetodit** lineaarsete võrrandisüsteemide lahendamiseks, samuti selle meetodi modifikatsioone.

Paketis **MATLAB** leiab determinandi funktsioon **det**.

Pöördmaatriksi leidmine.

On antud ruutmaatriks A. On vaja leida selle maatriksi pöördmaatriks A^{-1}.

Ka selle ülesande lahendamiseks saab kasutada algoritmi, mis põhineb **Gaussi ellimineerimismeetodil**. Pöördmaatriksi leidmiseks ja täpsustamiseks on kasutatavad ka **iteratsioonimeetodid**.

Paketis **MATLAB** leiab pöördmaatriksi funktsioon **inv**.

Maatriksiga seotud omaväärtusülesanded.

On antud reaalne n–järku ruutmaatriks A. Maatriksi A **omaväärtuseks** nimetatakse sellist reaal– või kompleksarvu λ, mille puhul võrrandisüsteemil

$$A \cdot x = \lambda \cdot x$$

on olemas mittetriviaalne lahend $x \neq 0$. Viimast nimetatakse sellele omaväärtusele vastavaks **omavektoriks**. Teatavasti on maatriksi A omaväärtuseks parajasti **karakteristliku võrrandi** $\det(A - \lambda \cdot I) = 0$ lahendid; siin I on n–järku ühikmaatriks.

Paketis **MATLAB** lahendab omaväärtusülesande funktsioon **eig**.

LOTI.05.019
Andmeanalüüs ja arvutused MATLAB-iga
Data Analysis and Computational Methods with MATLAB

Lineaarsete võrrandisüsteemide lahendamine MATLAB-iga

Olgu tarvis lahendada lineaarne võrrandisüsteem

$$A \cdot x = b.$$

St olgu tarvis leida vektor $x = (x_1, x_2,\ldots,x_n)^T$, et see (vektor)võrdus oleks rahuldatud.

Paketis MATLAB on selleks erinevaid võimalusi. Esimese asjana tuleb mälupiirkonda (Workspace) sisestada (sisse lugeda, näiteks) algandmed, milleks on maatriks A ja vabaliikmete vektor b. Oletame esmalt, et A on n·n ruutmaatriks elementidega a_{ij}, i, j = 1, 2,…, n; vabaliikmete vektor olgu kujul $b = (b_1, b_2,\ldots,b_n)^T$.

Demonstreerime lineaaralgebra vahendeid näite põhjal.

```
>> A=[1 2;3 4]

A =

    1    2
    3    4

>> b=[1;1]

b =

    1
```

```
     1

>> x=b'/A  % Massiivide jagamine, leiab süsteemi x·A = b' lahendi.

x =

   -0.5000    0.5000

>> x1*A

ans =

     1     1

>> x=A\b  % Pöördjagamine, leiab süsteemi A·x = b lahendi.

x =

    -1
     1

>> A*x  % Kontrollime vastust.

ans =

     1
     1

>> C = inv(A)  % Funktsioon inv leiab pöördmaatriksi maatriksile A.

C =

   -2.0000    1.0000
    1.5000   -0.5000

>> x = inv(A)*b  % Leiame süsteemi A·x = b lahendi.

x =

   -1.0000
    1.0000


>> C = A^-1  % Pöördmaatriksi leiab astendamistehe.

C =

   -2.0000    1.0000
    1.5000   -0.5000
```

```
>> x = A^-1*b  % Leiame süsteemi A·x = b lahendi (kolmas variant meil seni).

x =

   -1.0000
    1.0000

>> x = linsolve(A,b)  % Funktsioon linsolve leiab süsteemi A·x = b lahendi.


x =

    -1
     1

>> d=det(A)  %  Funktsioon det leiab maatriksi A determinandi.

d =

    -2


>> lambda = eig(A)  % Funktsioon eig leiab maatriksi A omaväärtused.

lambda =

   -0.3723
    5.3723

>> [V,D] = eig(A)  % Funktsioon eig leiab maatriksi A omaväärtused.

V =

   -0.8246   -0.4160
    0.5658   -0.9094


D =

   -0.3723         0
         0    5.3723
```

Maatriksi V veergudeks on omavektorid, maatriksi D diagonaalil omaväärtused.

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Teema 8. Paketi MATLAB lisad

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Lisaks programmipaketi põhimoodulile on arendajad loonud sadakond lisamoodulit, http://www.mathworks.se/products/index.html?s_tid=hp_fp_viewall Põ himooduli juurde kuuluvaid lisandatud m-failide komplekte spetsiaalsete rakendusülesannete lahendamiseks nimetatakse *Toolbox*'ideks. Nende kasutamisel on enamasti abiks graafilised kasutajaliidesed (GUI, *Graphical User Interface*). graafilisi kasutajaliideseid saab igaüks ka ise teha vastavalt oma vajadustele.

Teema kohta tuleb lahendada kohustuslik ülesanne, mida hinnatakse skaalal F-A (F - 1 punkt ja A 6 punkti). Kohustusliku ülesande tekst on eraldi failis. Teemakohaseks aruteluks kasutame foorumit.

Optimiseerimisülesanne

Optimiseerimisülesande üldine püstitus on järgmine.

On antud hulk $\mathbf{X}$ elementidega $\mathbf{x} \in \mathbf{X}$. Olgu antud funktsioon $\mathbf{f} : \mathbf{X} \to \mathbb{R}$, nõndanimetatud **sihifunktsioon** (*Objective Function*); $\mathbb{R}$ on reaalarvude hulk. Veel olgu antud funktsioonid $\mathbf{g}_i \to \mathbb{R}$, i = 1,2,…,m, mida nimetatakse **kitsendusfunktsioonideks** (*Constraints, Constraint functions*). Hulka $\mathbf{\Omega} = \{x \in \mathbf{X} \mid \mathbf{g}_i(x) \le 0, i = 1,2,\ldots,m\}$ nimetatakse **lubatavate lahendite hulgaks** (*Feasible Set*).

Ülesanne:

Leida min f(x) üle kõigi elementide $x \in \Omega$. (*)

Elementi $\mathbf{x}^* \in \mathbf{X}$, mis realiseerib funktsiooni **f** miinimumi lubatavate lahendite hulgal $\mathbf{\Omega}$, nimetatakse optimiseerimisülesande (*) **lahendiks,** ka **optimaalseks lahendiks**. Kindlasti ei saa ühtki lahendit nimetada kõige optimaalsemaks.

Optimiseerimisülesannetel on sageli rohkem kui üks lahend. Mittelahenduvus tähendab seda, et lubatavate lahendite hulk **Ω** on tühi või tõkestamata.

Sõltuvalt sihifunktsiooni **f**, hulga **X**, ja kitsenduste $\mathbf{g}_i$ (i = 1,2,…,m), st. lubatavate lahendite hulga kujust ja olemusest, nimetatakse ülesannet (*) lineaarseks planeerimisülesandeks, ruutplaneerimise ülesandeks, mittelineaarseks planeerimisülesandeks, poollõpmatuks planeerimisülesandeks, poolmääratud planeerimisülesandeks, mitme sihifunktsiooniga planeerimisülesandeks, diskreetseks planeerimisülesandeks, kombinatoorseks planeerimisülesandeks jne. Optimiseerimisülesannete ja –meetodite valdkond on kiiresti arenev, Paralleelselt modifitseerub ka vastav terminoloogia.

Mõnikord formuleeritakse maksimiseerimisülesanne, milles nõutakse sihifunktsiooni maksimumi leidmist. See on lihtsasti ja üldisust kahandamata viidav kujule (*): kui mingi element **x** minimiseerib funktsiooni **f** väärtuse hulgal **Ω**, siis maksimiseerib seesama element funktsiooni –**f** väärtuse samal hulgal.

Sageli on $\mathbf{X} = \mathbb{R}^n$, see tähendab, et optimiseerimismuutuja **x**, seega ka lubatavad lahendid, on n – mõõtmelised vektorid tavalises mõttes.

Väga oluliseks optimiseerimisülesannete klassiks on lineaarse planeerimise ülesanne:

Minimiseerida $\mathbf{c}^T\mathbf{x}$ kitsendustel $\mathbf{Ax} \le \mathbf{b}$; $\mathbf{Bx} = \mathbf{beq}$, $\mathbf{x}_{min} \le \mathbf{x} \le \mathbf{x}_{max}$.

Sel juhul üldises seades (*) on sihifunktsiooniks $\mathbf{f}(x) = \mathbf{c}^T\mathbf{x}$ (vektorite **c** ja **x** skalaarkorrutis) ja lubatavate lahendite hulgaks $\mathbf{\Omega} = \{\mathbf{x} \in \mathbf{X} = \mathbb{R}^n \mid \mathbf{Ax} \le \mathbf{b}, \mathbf{Bx} = \mathbf{beq}, \mathbf{x}_{min} \le \mathbf{x} \le \mathbf{x}_{max}\}$. Algandmeteks on **A** ja **B** - etteantud maatriksid vastavate dimensioonidega (kindlasti on nendes **n** veergu); **b**, **beq**, ja **c** - etteantud vektorid vastavate pikkustega. Võime kirjutada, et lubatavate lahendite hulga **Ω** määravad kitsendusfunktsioonid $(\mathbf{g}_1(\mathbf{x}),\ldots,\mathbf{g}_m(\mathbf{x}))^T = \mathbf{Ax}$, $(\mathbf{h}_1(\mathbf{x}),\ldots,\mathbf{h}_k(\mathbf{x}))^T = \mathbf{Bx}$.

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Tööriistariba APPS

Üks kolmest nn globaalsest tööriistaribast on paketi **MATLAB** rakenduste käivitamiseks. See asendab *Start*-nuppu vanemates versioonides.

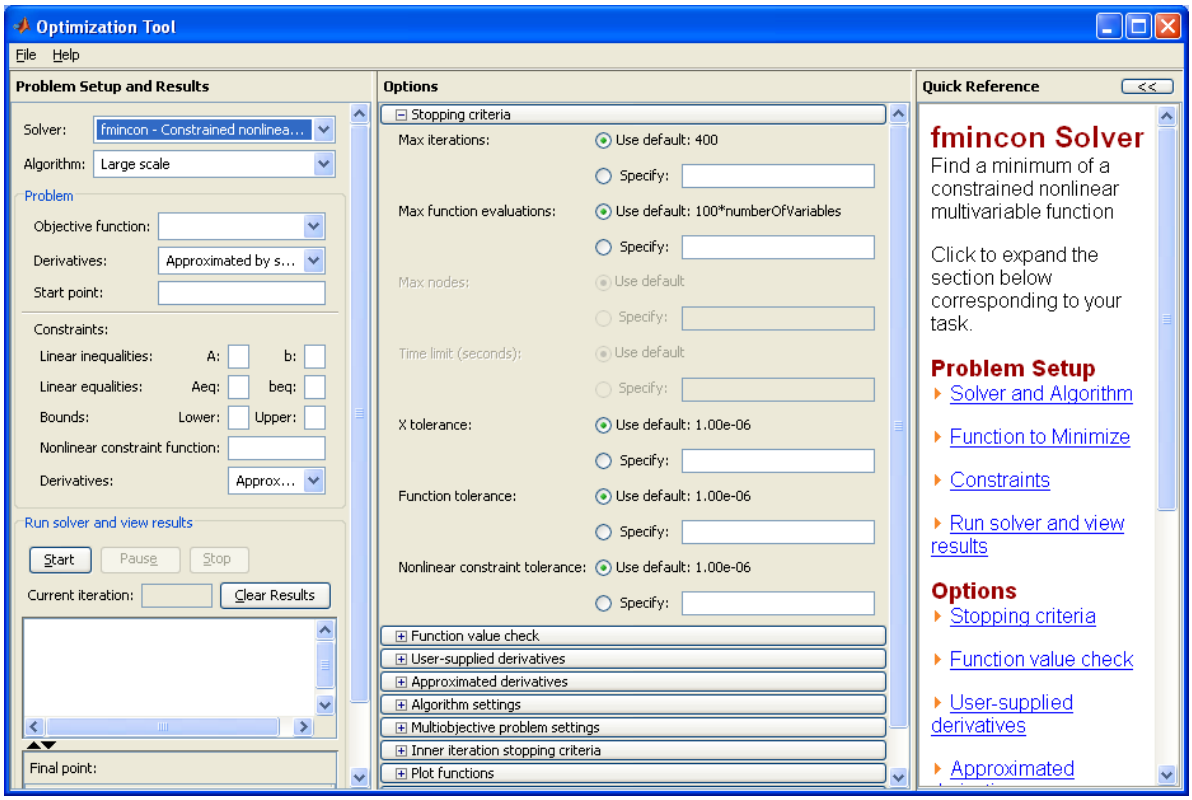
MATLAB, *Optimization Toolbox*

Optimiseerimisvahendi (*Optimization Toolbox*) on graafilise kasutajaliidese tüüpi abiline ja selle avamiseks tuleb esmalt käivitada programmipakett **MATLAB**.

Siis paketi tööriistaribalt (*Apps* Tab):

→ Optimization → Optimization Tool (optimtool)

Aken, mis avaneb, on selline:



Kolmest veerust vasakpoolne on ülesande sisestamiseks (*Problem Setup and Results*), teine mitmesuguste lahendusparameetrite etteandmiseks (*Options*) ja kolmas – abiaken (*Quick Reference*). Olulisim on vasakpoolne, mis määrab ülesande tüübi ja kust sisestatakse algandmed. Optimiseerimismeetodid on paketis **MATLAB** realiseeritud sisseehitatud funktsioonidena ja neid saab valida dialoogiaknas „Solver" sisestusriba paremal äärel oleva noole abil. Need funktsioonid ja seega ka vastavad ülesanded, mida kõnesoleva vahendiga saab lahendada, on järgmised.

Lineaarse ja ruutplaneerimise ülesanded (*Linear and Quadratic Minimization problems*).

linprog - lineaarse planeerimise ülesanne (*Linear programming*).

quadprog - ruutplaneerimise ülesanne (*Quadratic programming*).

Võrrandite ja süsteemide lahendamine (*Nonlinear zero finding, equation solving*).

fzero – skalaarse funktsiooni nullkoha leidmine (*Scalar nonlinear zero finding*).

fsolve – mittelineaarse võrrandisüsteemi lahendamine (*Nonlinear system of equations solve, function solve*).

Lineaarne vähimruutude meetod (*Linear least squares*).

lsqlin - lineaarne vähimruutude meetod lineaarsete kitsendustega (*Linear least squares with linear constraints*).

lsqnonneg - lineaarne vähimruutude meetod lahendite mittenegatiivsuse nõudega (*Linear least squares with nonnegativity constraints*).

Funktsioonide minimiseerimine (*Nonlinear minimization of functions*).

fminbnd – skalaarse tõkestatud funktsiooni miinimumi leidmine (*Scalar bounded nonlinear function minimization*).

fmincon – mitmemõõtmeline kitsendustega minimiseerimine (*Multidimensional constrained nonlinear minimization*).

fminsearch – mitmemõõtmeline kitsendusteta minimiseerimine Melder-Nead'i meetodil (*Multidimensional unconstrained nonlinear minimization, by Nelder-Mead direct search method*).

fminunc - mitmemõõtmeline kitsendusteta minimiseerimine (*Multidimensional unconstrained nonlinear minimization*).

fseminf - mitmemõõtmeline kitsendustega minimiseerimine (*Multidimensional constrained minimization, semi-infinite constraints*).

Mittelineaarne vähimruutude meetod (*Nonlinear least squares*).

lsqcurvefit – mittelineaarsete funktsioonide lähendamine vähimruutude meetodil (*Nonlinear curvefitting via least squares*).

lsqnonlin - mittelineaarne vähimruutude meetod (Nonlinear least squares with upper and lower bounds).

Mitme sihifunktsiooniga miinimumi leidmine (*Nonlinear minimization of multi-objective functions*).

fgoalattain - mitme sihifunktsiooniga miinimumi leidmine (*Multidimensional goal attainment optimization*).

fminimax – mitmemõõtmelised minimax ülesanded (*Multidimensional minimax optimization*).

Abiaken (*Help*), demovahendid (*Demo*) ja paljud muud materjalid aitavad täita ülejäänud lahtreid. Peamised optimiseerimisülesanded leiavad käsitlemist auditoorse töö käigus.

Graafiline kasutajaliides

Graafiline kasutajaliides (GUI, *Graphical User Interface*) on mõeldud paketi MATLAB kasutamise hõlbustamiseks. GUI võimaldab ühelaadseid tegevusi interaktiivselt korduvalt teha. Igaüks võib ise endale sobiva liidese luua paketti sisseehitatud vahenditega.

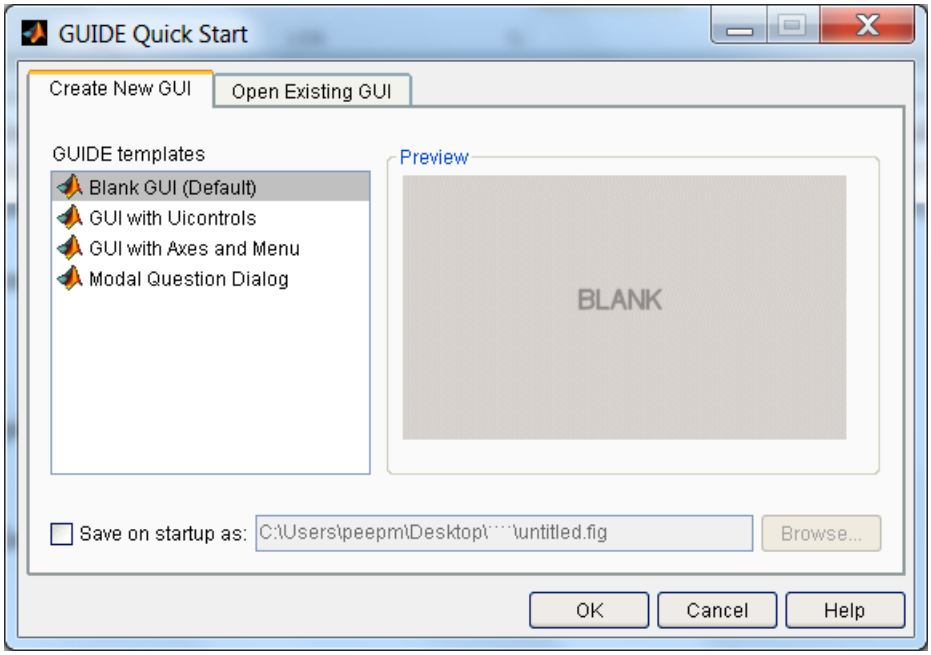
GUI loomiseks avada menüüst

New → GUI .

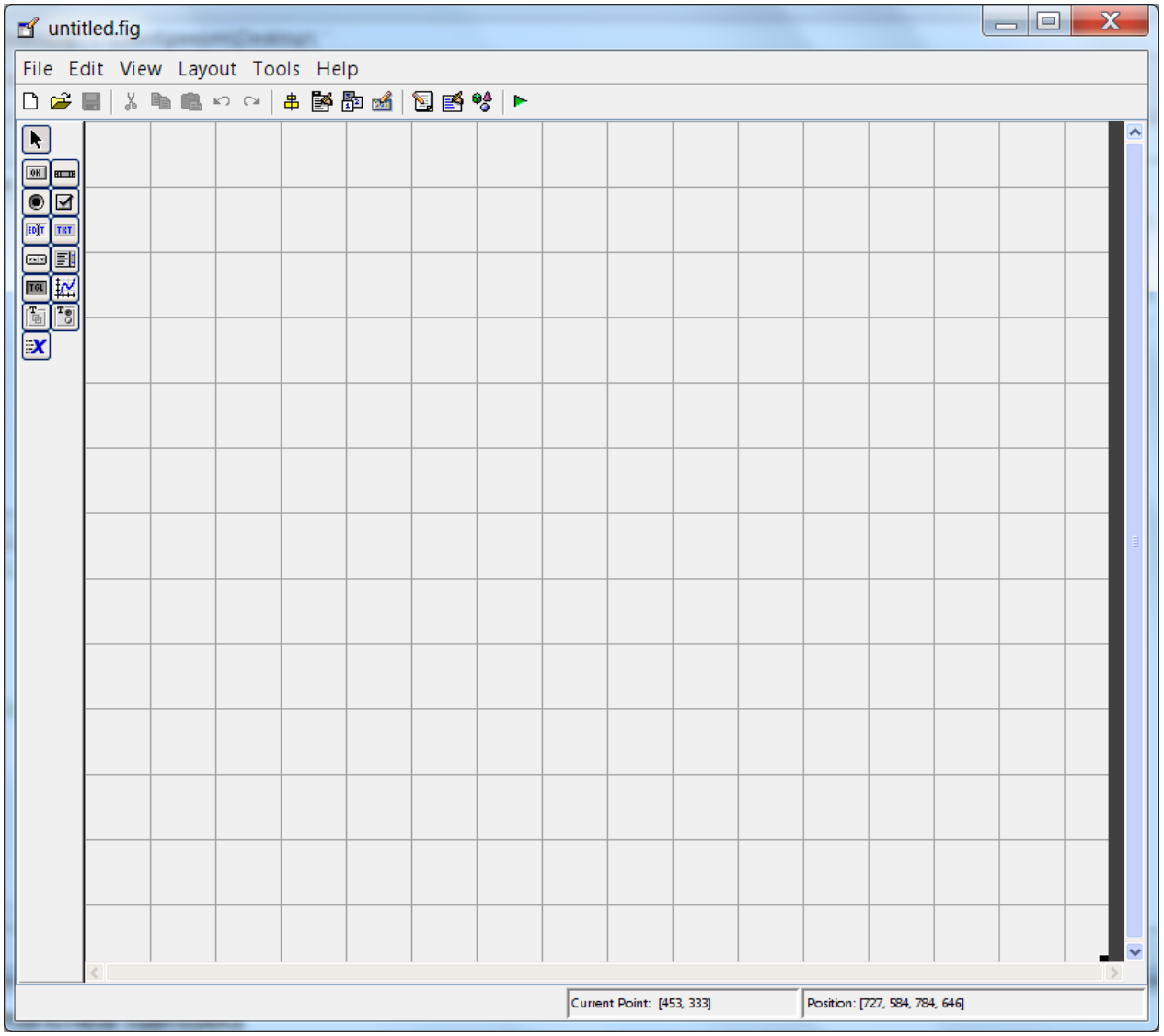
Või tippida käsuaknasse

>> **guide**

Avaneb aken **GUIDE *Quick Start***



Nüüd tuleb valida mall. Näiteks, kui valime **Blank GUI (default)**, avaneb aken



Et oleksid näha **GUI** komponentide nimetused, tuleb valida **File Menu → Preferences** ja märkida linnuke **Show names in component palette** ette.

GUI suuruse seadistamiseks, tuleb klõpsata alumisse paremasse nurka ja lohistada seda ala väiksemaks.

Komponentide lisamiseks tuleb lisada paneel ja klahvid ning lohistada need kavandamispinnale.

Selleks, et komponendid joondada võib kasutada ***Alignment Tool*** 'i.
Valida vajalikud komponendid, seejärel valida ***Tools*** menüüst ***Align Objects***.

Edasiseks seadistamiseks valida ***View*** menüüst ***Property Inspector***.

Name Property – saab **GUI**'le nime anda.
Title Property – antakse pealkiri paneelile. Valida ***Title*** ja sisestada soovitu.

Peep Miidla, Jüri Vedru, Tartu Ülikool, 2013

Teema 9. Blokkmodelleerimise süsteem SIMULINK

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

SIMULINK on programmipaketi **MATLAB** lisakeskkond, blokkmodelleerimise süsteem, mis võimaldab modelleerida, analüüsida ja simulatsioonina jälgida dünaamilisi süsteeme, , s.t süsteeme, millede väljundid muutuvad ajas.

Teema kohta tuleb lahendada kohustuslik ülesanne, mida hinnatakse skaalal F-A (F - 1 punkt ja A 6 punkti). Kohustusliku ülesande tekst on eraldi failis. Teemakohaseks aruteluks kasutame foorumit.

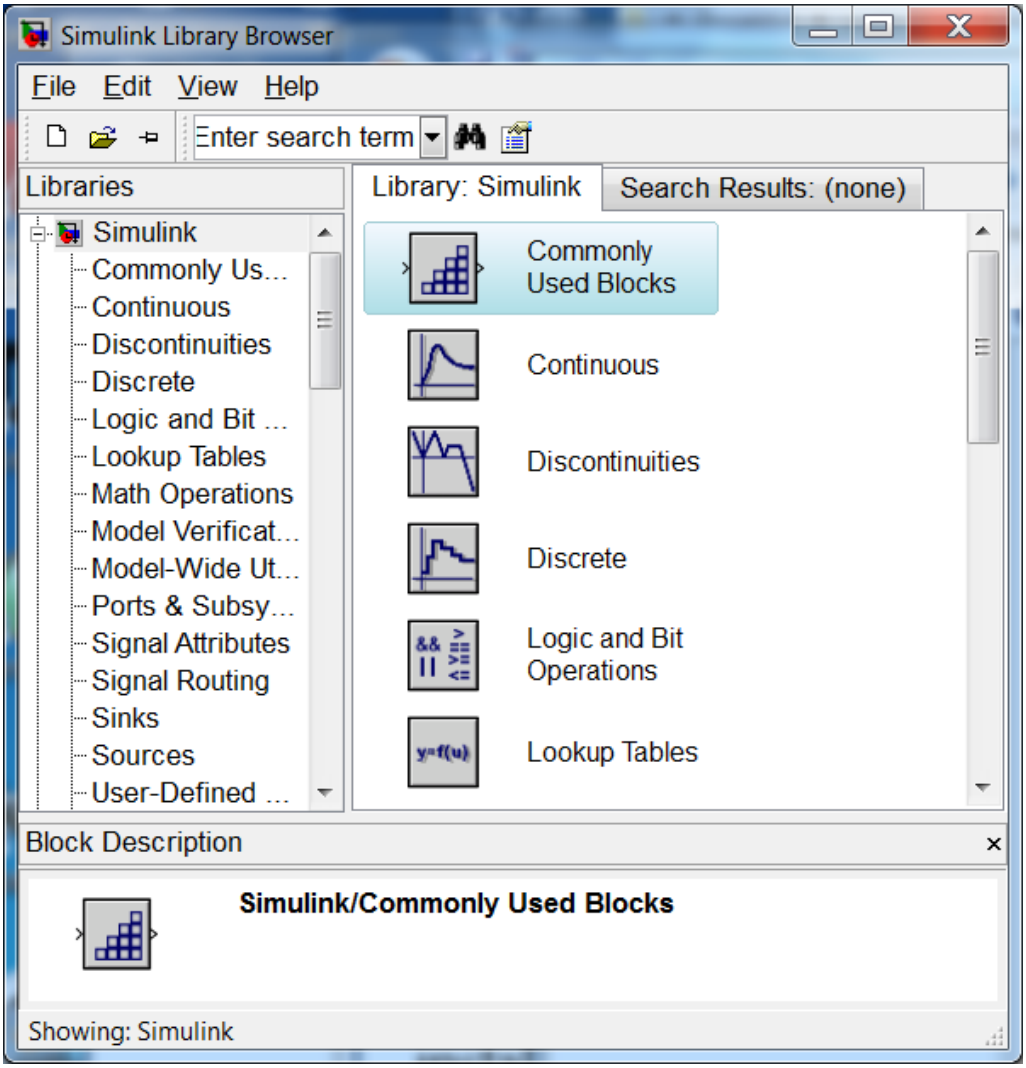
SIMULINK

SIMULINK on programmipaketi **MATLAB** lisakeskkond, blokkmodelleerimise süsteem, mis võimaldab modelleerida, analüüsida ja simulatsioonina jälgida dünaamilisi süsteeme, , s.t süsteeme, millede väljundid muutuvad ajas.

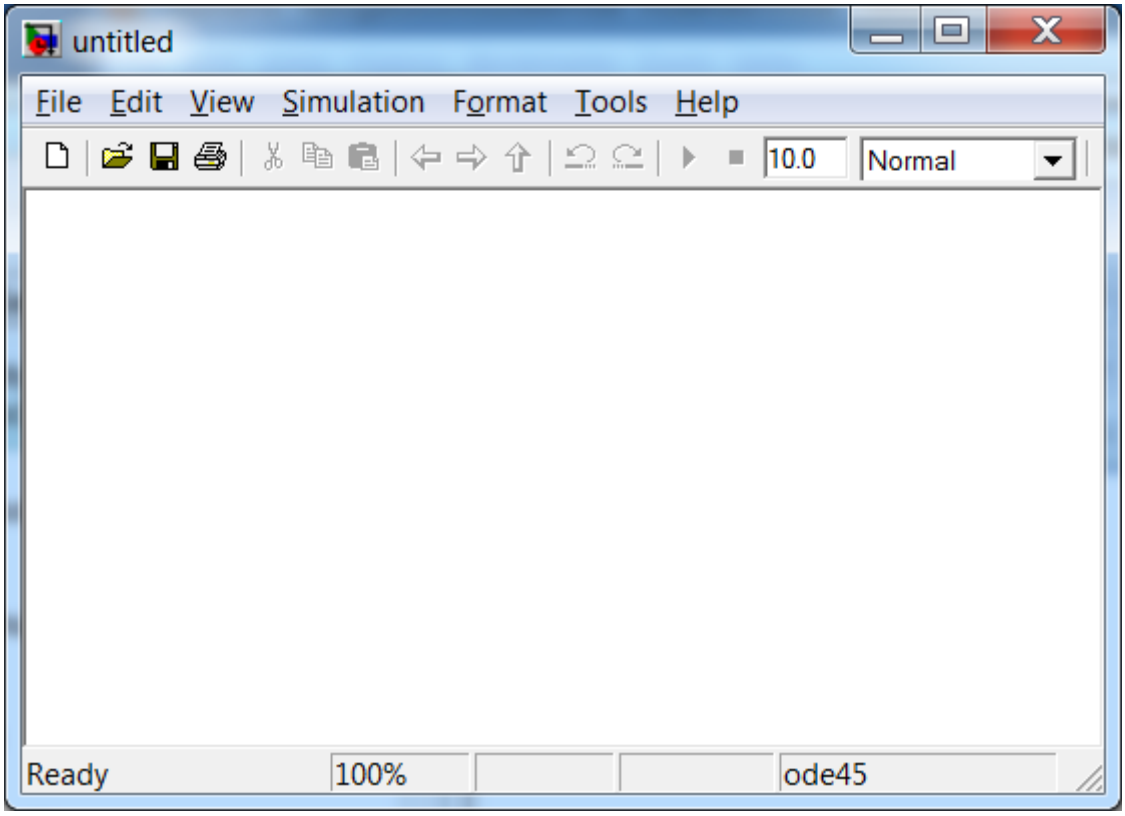
Keskkonna **SIMULINK** käivitamiseks tuleb esmalt käivitada **MATLAB**, seejärel vajutada vastavat nuppu  **MATLAB** paneelil või tippida käsk

```
>> simulink
```

käsuaknasse. Avaneb blokkide teek, ***Library Browser*** :



Sellest leiab kõik mudelite konstrueerimiseks vajalikud blokid, mis on hetkel installeeritud. Neid blokke saab lohistada või kopeerida mudeliaknasse. Mudeliaken avaneb, kui klikkida nupul ***Open*** (saab avada varem salvestatud mudeleid) või ***New*** (avaneb uus mudeliaken).



Koostatud (poolelioleva) mudeli saab salvestada, nagu tavaliselt. Keskkonna **SIMULINK** käske saab sisestada

-mudeliakna menüüribalt;
-kontekstitundliku rippmenüü abil, mis avaneb hiire parempoolse klahvi klõpsuga;
-mudeliakna nupuribalt.

Mudeleid saab printida (***Print***) ning varustada ülevaadetega (***Print details***).

Blokkide komplektid:

Commonly Used Blocks – Sagedamini kasutatavad blokid. Loetelu moodustub kasutajale vastavalt, st on iseõppiv.

Continuous – blokid, mis modelleerivad pidevaid lineaarseid operaatoreid (tuletis, integraal, lineaarne olekusüsteem, ülekandefunktsioon, hilistumine jt).

Discontinuities – mittepidevad funktsioonid ja operaatorid.

Discrete – diskreetsed funktsioonid.

Logic and Bit Operations – loogilised ja bitifunktsioonid.

Look-Up Tables – ülevaatetabelid; väljundi kuju sisendi põhjal.

Math Operations – matemaatiliste operatsioonide blokid.

Model Verification – blokid mudeli automaatseks juhtimiseks.

Model-Wide Utilities – utiliidid.

Ports & Subsystems – alamsüsteemid ja -mudelid.

Signal Attributes - signaali omadused.

Signal Routing - signaali suunamine/ümbersuunamine mudelis.

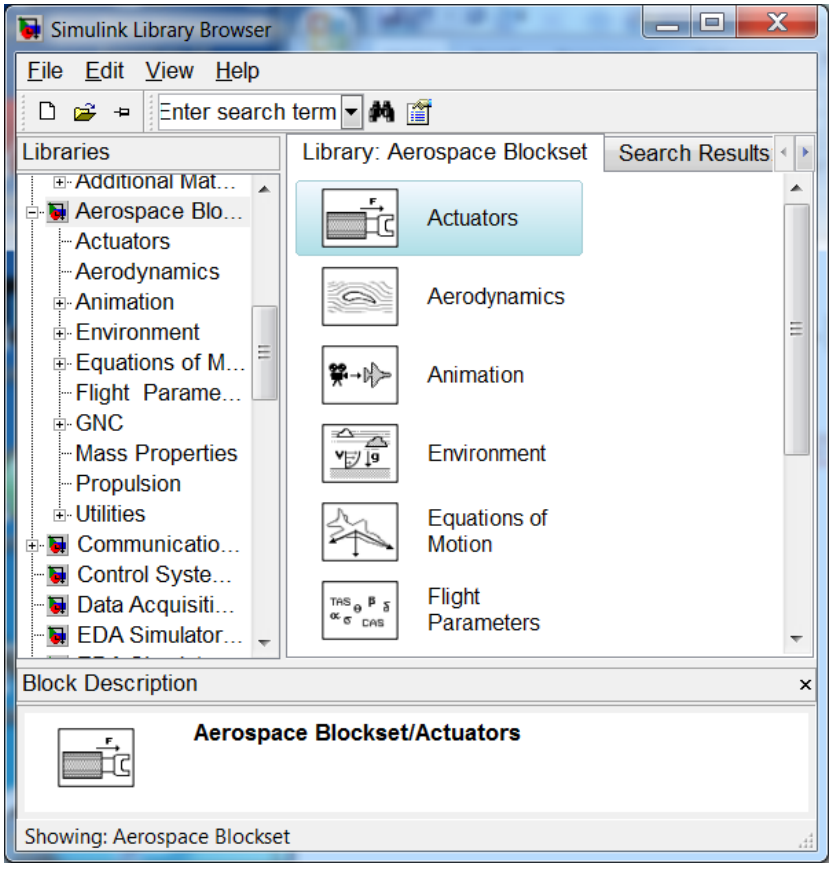
Sinks - väljundblokid.

Sources - allikad, algandmed, signaalide generaatorid.

User-Defined Functions - kasutaja poolt defineeritud funktsioonid.

Additional Math & Discrete – lisafunktsioonid.

Vastavalt paketi **MATLAB** kasutaja huvidele ja võimalustele saab arendajalt juurde osta lisablokkide süsteeme, ***Blockset***, mille abil saab spetsialiseeritumaid mudeleid koostada. Allpool näiteks ***Aerospace Blockset.***



LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

SIMULINK

Mudelite koostamine

Avada mudeli aken. ***New*** avab uue, tühja mudeliakna. Blokke saab mudeliaknasse (**Simulink *Library Browser***-ist) viia lohistamise või kopeerimise teel, samamoodi neid seal liigutada.

Blokkide omavaheline ühendamine.

(i) võib ise joone "vedada", vasaku hiireklahviga lohistades;
(ii) tähistada blokk, kust joon algab ja ***Ctrl***-klahvi hoides klikkida sihtblokile. Saab ka blokkide komplekte korraga ühendada.

Hargnev joon tekib, kui olemasoleva joone punktist hakata vasaku hiireklahviga lohistama, hoides ***Ctrl***- klahvi. Sama: lohistades parema hiireklahviga. Joone segmente ja murdepunkte saab liigutada, lohistades vasaku hiireklahviga. Joonele saab uue bloki paigutada, kui see vajalikule kohale lohistada.

Bloki saab ühendustest (kohalt mudelis) lahti, kui seda lohistada ***Shift***- klahvi all hoides.

Teksti saab mudelile paigutada, kui teha vasaku hiireklahvi topeltklõps vajalikul kohal, tekib tekstikast. Viimast saab ümber paigutada lohistamisega.

Suurte mudelite puhul on otstarbekas kasutada **alamsüsteemi** (***Subsystem***) blokki (vt. ***Ports & Subsystems***). Alamsüsteem koostatakse nii nagu tavalist mudelit, sellel on küll sisend ja väljund. Alamsüsteemi saab tekitada ka olemasolevate blokkide baasil. Need tuleb lohistamisega valida (märgistada ümbritseva kastiga) ning siis valida

Edit --> ***Create Subsystem*.**

Simulatsiooni käivitamine

(i) ***Run*** nupp nupuribal;
(ii) ***Ctrl+T***;
(iii) Menüüst: ***Simulation*** --> ***Start*.**

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

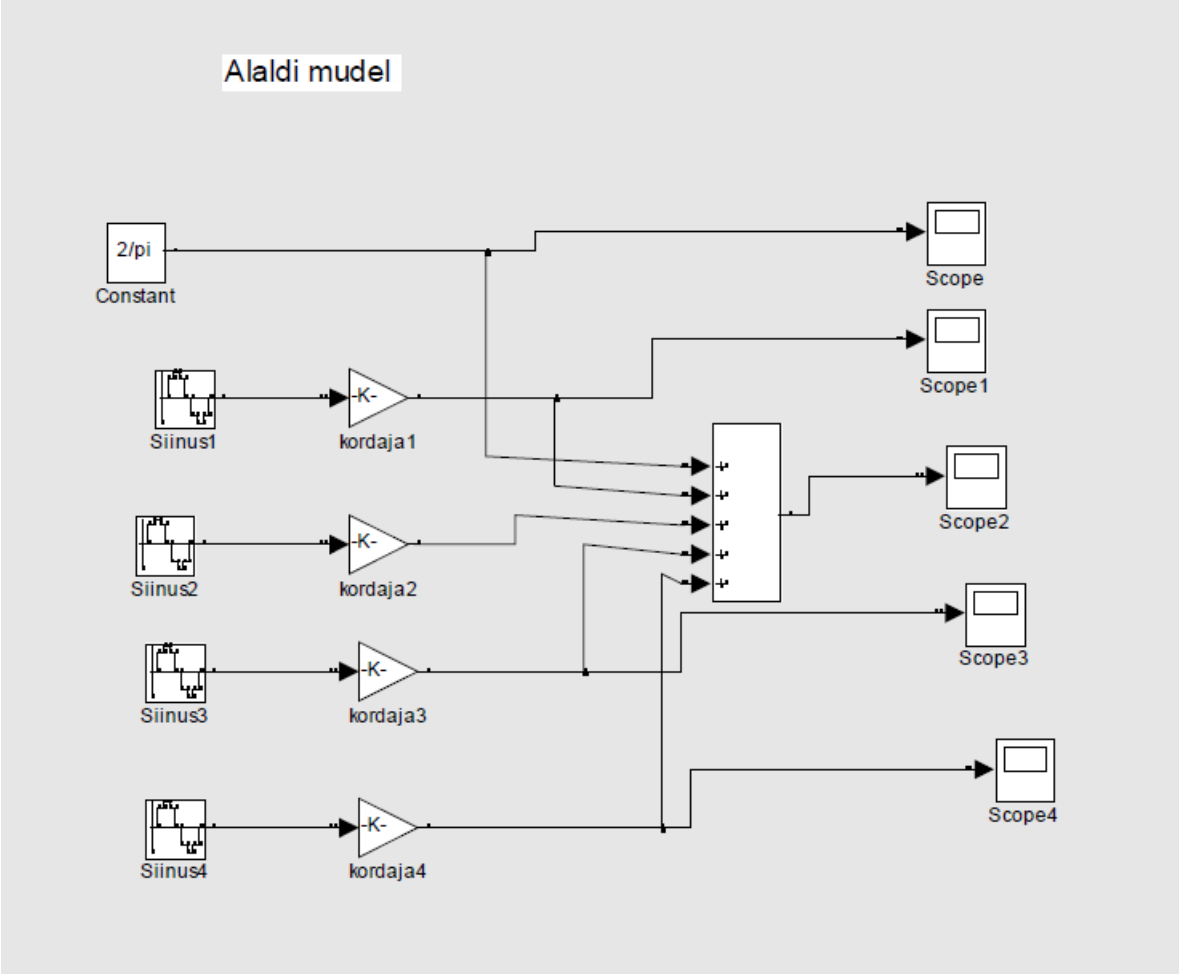
SIMULINK harjutusülesandeid

1. Konstrueerida alaldi mudel, kus sisend on antud funktsiooniga

$$v(t) = \frac{2}{\pi} - \left| \frac{4\cos(2\omega t)}{\pi(1)(3)} - \frac{4\cos(4\omega t)}{\pi(3)(5)} - \frac{4\cos(6\omega t)}{\pi(5)(7)} - \frac{4\cos(8\omega t)}{\pi(7)(9)} \right|$$

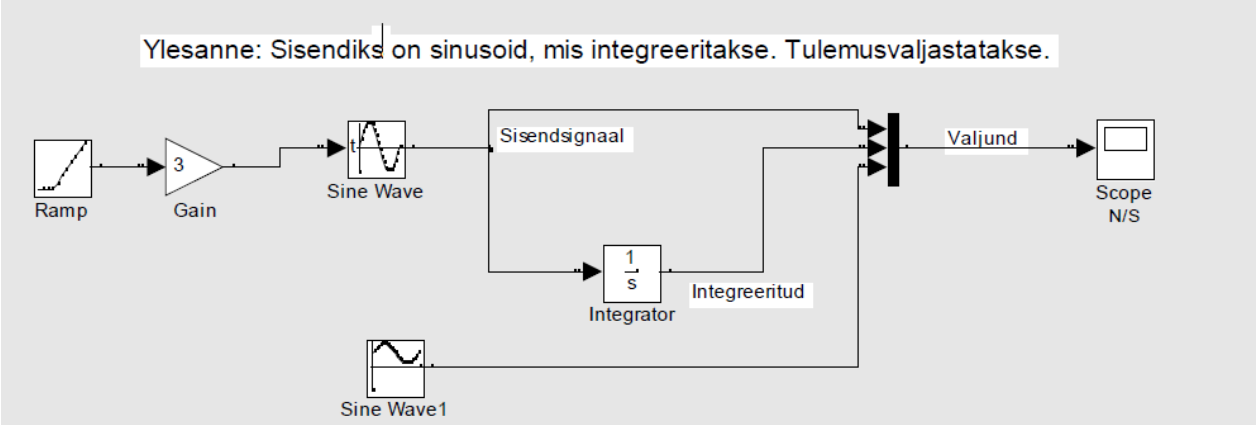
Siin w – sagedus.

Lahendus:



2. Anda ette sisendsignaalina siinus, integreerida see. Esitada erinevalt väljundid, sh. ka **MATLAB**-i tööpiirkonda (*Workspace*).

Lahendus:



3. Modelleerida mass-vedru-amortisaatori süsteem.
Matemaatiliseks mudeliks on diferentsiaalvõrrand

$$mx'' + cx' + kx = f(t),$$

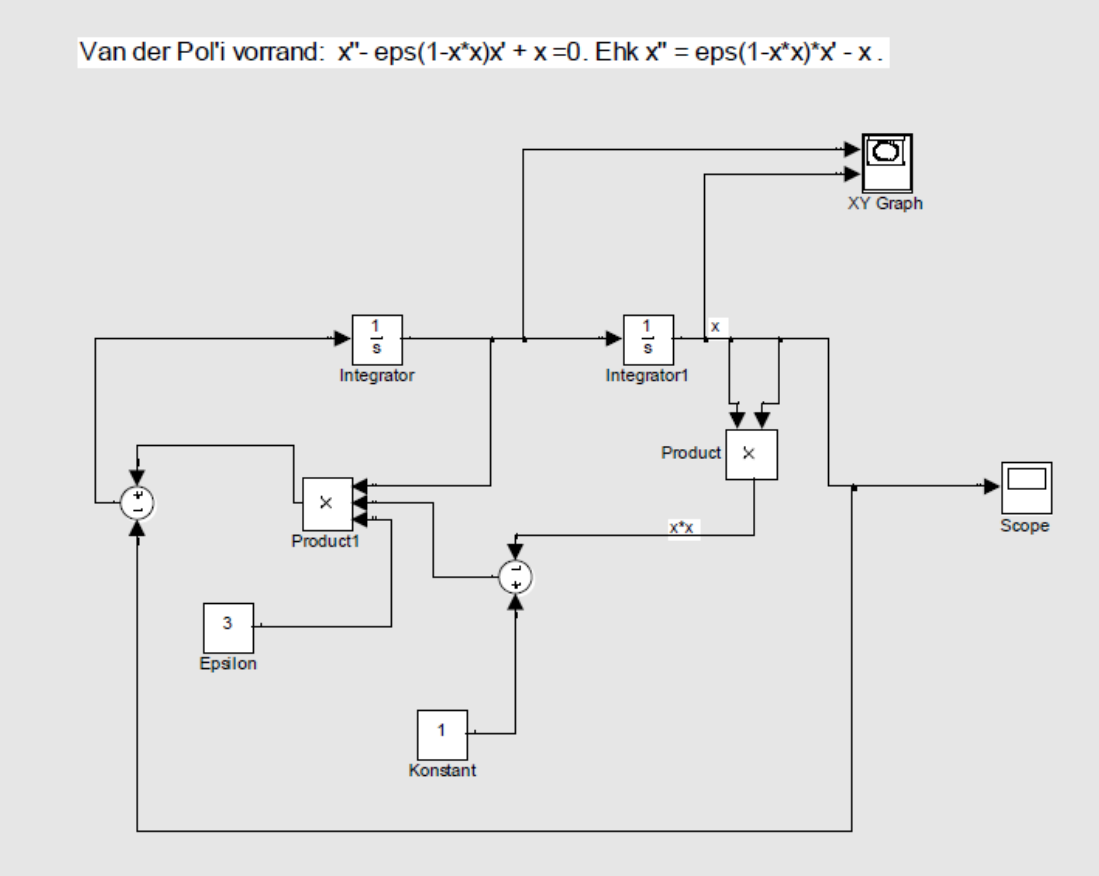
kus m on keha mass, c – hõõrdetegur, k – vedru jäikus, f – väline jõud; nt m = 2; c = 0.7; k=1, f – astmefunktsioon]. Väljastada ka faasikõver. Realiseerida lahendus ülekandefunktsiooni abil.
[*Step* (Sources), *Transfer Function* (1/(ax*x + bx + 1); Continuous), *Scope* (Sinks), *Save File to Workspace* (Sinks).]

4.Uurida Van der Pol'i diferentsiaalvõrrandit

$$x'' – eps(1 – x*x)x' + x = 0.$$

Väljastada faasikõver ja lahend.

Lahendus:


Van der Pol'i vorrand: x''- eps(1-x*x)x' + x =0. Ehk x'' = eps(1-x*x)*x' - x .
XY Graph
1
s
Integrator
1
s
Integrator1
x
Product
Product1
x*x
Scope
3
Epsilon
1
Konstant

Teema 10. Rakendusülesanded ja individuaalsed esitlused

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Mõnede rakendusülesannete lahendamisest paketi MATLAB abil. Lõpliku hinde saamiseks tuleb kõikidel aine kuulajatel teha individuaalne esitlusettekanne mingi ülesande lahenduse või paketi **MATLAB** rakenduse kohta. Selle teemad võib valida toodud nimistust või ise vastavalt oma tööle ja huvidele.

Ettekanne mida hinnatakse skaalal F-A (F - 1 punkt ja A 6 punkti). Ettekanne peab olema sooritatud vähemalt hindele E, et terve kursus positiivselt hinnatud saada. Aruteludeks ja esitluste materjalide üleslaadimiseks ja kaasüliõpilastele kättesaadavaks tegemiseks kasutame foorumit.

Dünaamiline süsteem

Süsteem kõige üldisemalt tähendab omavahel seotud objektide terviklikku kogumit, dünaamiline selle juures näitab, et süsteemi seisund on ajas muutuv ja see muutumine ka jälgitav. Ajalooliselt tähistas termin dünaamiline süsteem (DS) põhiliselt kindla vabadusastmete arvuga mehhaanilist süsteemi. Täna kasutatakse DS mõistet hoopis laiemas kontekstis. Võib öelda, et loodusteadlased peavad DS-ks igasugust liikuvat süsteemi, matemaatikute jaoks on selleks üleskirjutus

$$\mathbf{x}(k+1) = A(k)\mathbf{x}(k) + B(k)\mathbf{u}(k)+\mathbf{w}(k), \qquad (1)$$

kus $\mathbf{x} = (x_1,\ldots,x_n)^T$ on süsteemi olekuvektor (*System State vector*), selle komponendid aga süsteemi olekuparameetrid. Matemaatiliselt on mugav vektoreid kujutada veeruvektoritena, seepärast on transponeerimise märk juures. Indeks k tähendab seda, et süsteemi seisund muutub üleminekul ajahetkelt k hetkele k+1. Siin A on teadaolev süsteemi maatriks ehk üleminekumaatriks mõõtmetega n×n; vektor $\mathbf{u} = (u_1,\ldots,u_m)$ – nn. juhtimisvektor ehk juhis tähistamaks süsteemi talituse reguleerimist võimaldavaid väljastpoolt muudetavaid parameetreid; B – juhtimismaatriks (*Control Matrix*) dimensiooniga n×m; **w** – veavektor pikkusega n, mille kohta Kalmani filtri käsitlemisel eeldatakse, et see on normaaljaotusega juhuslik vektor, mille keskväärtus on null, mille kovariatsioonimaatriks Q on teada ning mille komponendid on nii üksteisest kui ka ajas sõltumatud. Vektor **w** peab olema nn. valge müra, kusjuures $\mathbf{w}(k) \sim N(0, Q(k))$. Maatriksid A, B ja Q võivad ajaindeksist k sõltuda, aga võivad ka mitte sõltuda; lihtsuse mõttes oletamegi allpool, et A ja B on konstantsed.

Dünaamiline süsteem on matemaatilise modelleerimise üks põhimõisteid ning selle defineerimiseks on mitmeid erineva abstraktsioonitasemega võimalusi. Märgime, et seosega (1) esitatakse nn. diskreetne lineaarne DS ja sellest piisab, kui

räägime Kalmani filtrist, sest algne tulemus oli R. E. Kalmani poolt just selle juhu jaoks formuleeritud. Kindlasti tuleb kasuks ettekujutus, et DS on loodusliku süsteemi matemaatiline mudel, milles viimase olekuid kujutatakse ette n- mõõtmelise Eukleidilise ruumi punktidena (olekuvektor **x**) ning millede muutumine toimub fikseeritud reeglite (maatriksi A rakendamine ruumi punktidele) kohaselt. Seda ruumi nimetatakse vaadeldava DS faasiruumiks (*Phase Space*) ning kõige üldisemalt võib DS vabadusastmete arvuks lugeda selle ruumi dimensiooni. Vabadusastmete arv on aga üldjuhul raskesti defineeritav, mistõttu igal konkreetsel juhul tuleb selles ka täpselt veenduda. Näieks laialt levinud Hamiltoni süsteemide korral on vabadusastmete arv hoopis kaks korda väiksem kui süsteemi dimensioon.

Nüüd saame etteruttavalt öelda, mis on Kalmani filter (KF). See on nimelt arvutusmeetod DS olekuvektori **x** täpsustamiseks, olekuparameetrite $x_1,\ldots,x_n$ mingis mõttes paremate väärtuste leidmine nn aprioorse ennustuse

$$\mathbf{x}^-(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \qquad (2)$$

ja loodusliku süsteemi oleku mõõtmise tulemuste alusel. Matemaatilise modelleerimise seisukohalt pole „loodusliku süsteemi" olemasolu sageli üldse probleemiks – see lahendatakse arvutisimulatsiooni abil, kusjuures modelleeritakse ka müra selle teoreetiliste omaduste alusel – selleks kasutatakse juhuslike arvude generaatoreid, mis kaasaegsetele arvutuspakettidele on alati lisatud. Matemaatika seda osa, mis vaatleb kõrvuti nii looduslikke, tegelikkuses ette tulevaid dünaamilisi süsteeme ja nende matemaatilisi mudeleid näiteks kujul (1), nimetatakse tööstusmatemaatikaks (*Industrial Mathematics*). Viimane on arenenud tehnoloogiatega riikides kõrgel tasemel, tööstuse all seejuures mõistetakse kogu maailmas hoopis laiemat rakendusvaldkonda kui see eesti keeles niimoodi väljendab: tootmine, tehnoloogia, toorainete töötlemine, energeetika, transport ja logistika, ökoloogia, seired (atmosfääri, mere, maakoore), juhtimine, finantsmajandus, äri- ja kaubandustegevus, rakendusteadused jne. Igal pool tuleb ette dünaamilisi süsteeme ja nende matemaatilise modelleerimise vajadust. KF on seega tööstusmatemaatika oluline tööriist.

Mõõtmine

KF tagasisidestab reaalse DS mõõtmise tulemused selle süsteemi olekuparameetrite täpsustamiseks. Sõltugu mõõtmistulemus **y** olekuvektorist **x** nii:

$$\mathbf{y}(k+1) = H(k+1)\mathbf{x}(k+1) + \mathbf{v}(k+). \qquad (3)$$

Siin **y** – p-mõõtmeline väljund, mõõtetulemus; H(k+1) on teadaolev maatriks mõõtmetega p×n; **v** - häiritused, vead; jällegi keskväärtusega null valged Gaussi mürad teadaoleva kovariatsioonimaatriksiga R; $\mathbf{v}(k+1) \sim N(0, R(k+1))$. Allpool oletame, et ka H on konstantne, H(k+1) = H. Algolek **x**(0) ja müravektorid **w**(k+1), **v**(k+1) eeldatakse samuti olevat vastastikku sõltumatud. Indeksiks valemis (3) on k+1 rõhutamaks, et mõõtmine on aposterioorne tegevus, sooritatakse pärast seda, kui DS on olekust k juba olekusse k+1 üle läinud.

Mõõtmistulemuse esitamine kujul (3) on õigustatud: reaalsete süsteemide korral on sageli raske kui mitte võimatu mõõta süsteemi parameetrite väärtusi vahetult. Võib juhtuda, et vajalikule mõõtmiskohale on raske ligi pääseda või oleks

tarvis teha nii palju ja nii paljudes kohtades mõõtmisi, et see teeks kogu tulemuse mõttetult kalliks. Sel juhul tuleb „soodsatest kohtadest" saadud näidud kuidagi kas interpoleerida või ekstrapoleerida kogu vajalikku piirkonda. Aga võib ette tulla ka igasuguseid muid olukordi, kus mõõtmisoperaatori H olemasolu on loomulik ja möödapääsmatu.

Asjaolu, et on välja mõeldud ja laialdaselt kasutusele võetud Kalmani filter DS uurimisel, näitab, et teadlased ja praktikud ei usalda täielikult ei teoreetilist mudelit (1) ega ka mõõtmistulemusi (3).

LOTI.05.019

Andmeanalüüs ja arvutused MATLAB-iga

Data Analysis and Computational Methods with MATLAB

Kalmani filter

Antakse ülevaade ühest olulisest matemaatilisest vahendist dünaamiliste süsteemide uurimisel – Kalmani filtrist (KF). KF on meetod dünaamiliste süsteemide olekuparameetrite väärtuste täpsustamiseks üleminekul ühelt ajahetkelt järgmisele. Mõned näited ka.

Ajaloost

Nimi „Kalman" ja sõna „filter" on eraldi võttes väga laialt tuntud. Esimene vähemalt operetihuviliste seas, teine perenaiste, suitsetajate, autojuhtide ja paljude teiste hulgas. Koos kirjutatuna saame termini, mis tähistab üht kaasajal väga laialt kasutatavat arvutusmeetodit dünaamiliste süsteemide olekuparameetrite väärtuste täpsustamiseks. Järgnevas räägitaksegi esmalt dünaamilistest süsteemidest ja selgitatakse nende tähendust. Siis Kalmani filtrist, pisut selle matemaatilisest põhjendusest ning seejärel rakendustest.

Rudolf Emil Kálmán on sündinud 19. mail 1930. aastal Budapestis, hariduse omandas aga juba Ameerika Ühendriikides: bakalaureuse- ja magistrikraadi, elektriinsenerina, muide, Massachusettsi Tehnoloogiainstituudis vastavalt 1953. ja 1954. aastal, doktorikraadi Columbia

Ülikoolis 1957. aastal. Pärast pikki ja viljakaid tööaastaid mitmetes tuntud uurimisasutustes on R. E. Kalman nüüd Florida Ülikooli emeriitprofessor, pärjatud paljude tiitlitega ja auhindadega. Viimaste seas on 1985. aastal saadud Kyoto auhind, mida annab välja Inamori Fond ning mida nimetatakse mõnikord ka Jaapani Nobeli preemiaks.

Kalmani põhjapanev töö ilmus 1960.aastal[1]. Selles käsitleti lineaarse diskreetse süsteemi seisundi optimaalse hindamise ülesannet ning anti sellele ka teoreetiliselt põhjendatud lahendus. Sellest ajast on Kalmani filter olnud teoreetiliste ja rakenduslike uurimuste rakenduste tähelepanu keskmes. Olulisemad kasutusvaldkonnad on navigeerimine ja lennuaparaatide juhtimine, radariseire, sonaarseire, aga ka seismoloogia, meteoroloogia, ökonomeetria ja tuumaenergeetika.

Dünaamiline süsteem

Süsteem kõige üldisemalt tähendab omavahel seotud objektide terviklikku kogumit, dünaamiline selle juures näitab, et süsteemi seisund on ajas muutuv ja see muutumine ka jälgitav. Ajalooliselt tähistas termin dünaamiline süsteem (DS) põhiliselt kindla vabadusastmete arvuga mehhaanilist süsteemi. Täna kasutatakse DS mõistet hoopis laiemas kontekstis. Võib öelda, et loodusteadlased peavad DS-ks igasugust liikuvat süsteemi, matemaatikute jaoks on selleks üleskirjutus

$$\mathbf{x}(k+1) = A(k)\mathbf{x}(k) + B(k)\mathbf{u}(k)+\mathbf{w}(k), \tag{4}$$

kus $\mathbf{x} = (x_1,\ldots,x_n)^T$ on süsteemi olekuvektor (*System State vector*), selle komponendid aga süsteemi olekuparameetrid. Matemaatiliselt on mugav vektoreid kujutada veeruvektoritena, seepärast on transponeerimise märk juures. Indeks k tähendab seda, et süsteemi seisund muutub üleminekul ajahetkelt k hetkele k+1. Siin A on teadaolev süsteemi maatriks ehk üleminekumaatriks mõõtmetega n×n; vektor $\mathbf{u} = (u_1,\ldots,u_m)$ – nn. juhtimisvektor ehk juhis tähistamaks süsteemi talituse reguleerimist võimaldavaid väljastpoolt muudetavaid parameetreid; B – juhtimismaatriks (*Control Matrix*) dimensiooniga n×m; **w** – veavektor pikkusega n, mille kohta Kalmani filtri käsitlemisel eeldatakse, et see on normaaljaotusega juhuslik vektor, mille keskväärtus on null, mille kovariatsioonimaatriks Q on teada ning mille komponendid on nii üksteisest kui ka ajas sõltumatud. Vektor **w** peab olema nn. valge müra, kusjuures $\mathbf{w}(k) \sim N(0, Q(k))$. Maatriksid A, B ja Q võivad ajaindeksist k

[1] Kalman, R. E., A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME-Journal of Basic Engineering, Vol.82, (Series D), P. 35-45, 1960. Vt ka http://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf (viimati väisatud 7. novembril 2007).

sõltuda, aga võivad ka mitte sõltuda; lihtsuse mõttes oletamegi allpool, et A ja B on konstantsed.

Dünaamiline süsteem on matemaatilise modelleerimise üks põhimõisteid ning selle defineerimiseks on mitmeid erineva abstraktsioonitasemega võimalusi. Märgime, et seosega (1) esitatakse nn. diskreetne lineaarne DS ja sellest piisab, kui räägime Kalmani filtrist, sest algne tulemus oli R. E. Kalmani poolt just selle juhu jaoks formuleeritud. Kindlasti tuleb kasuks ettekujutus, et DS on loodusliku süsteemi matemaatiline mudel, milles viimase olekuid kujutatakse ette n-mõõtmelise Eukleidilise ruumi punktidena (olekuvektor **x**) ning millede muutumine toimub fikseeritud reeglite (maatriksi A rakendamine ruumi punktidele) kohaselt. Seda ruumi nimetatakse vaadeldava DS faasiruumiks (*Phase Space*) ning kõige üldisemalt võib DS vabadusastmete arvuks lugeda selle ruumi dimensiooni. Vabadusastmete arv on aga üldjuhul raskesti defineeritav, mistõttu igal konkreetsel juhul tuleb selles ka täpselt veenduda. Näieks laialt levinud Hamiltoni süsteemide korral on vabadusastmete arv hoopis kaks korda väiksem kui süsteemi dimensioon.

Nüüd saame etteruttavalt öelda, mis on Kalmani filter (KF). See on nimelt arvutusmeetod DS olekuvektori **x** täpsustamiseks, olekuparameetrite $x_1,\ldots,x_n$ mingis mõttes paremate väärtuste leidmine nn aprioorse ennustuse

$$\mathbf{x}^{-}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \qquad (5)$$

ja loodusliku süsteemi oleku mõõtmise tulemuste alusel. Matemaatilise modelleerimise seisukohalt pole „loodusliku süsteemi" olemasolu sageli üldse probleemiks – see lahendatakse arvutisimulatsiooni abil, kusjuures modelleeritakse ka müra selle teoreetiliste omaduste alusel – selleks kasutatakse juhuslike arvude generaatoreid, mis kaasaegsetele arvutuspakettidele on alati lisatud. Matemaatika seda osa, mis vaatleb kõrvuti nii looduslikke, tegelikkuses ette tulevaid dünaamilisi süsteeme ja nende matemaatilisi mudeleid näiteks kujul (1), nimetatakse tööstusmatemaatikaks (*Industrial Mathematics*). Viimane on arenenud tehnoloogiatega riikides kõrgel tasemel, tööstuse all seejuures mõistetakse kogu maailmas hoopis laiemat rakendusvaldkonda kui see eesti keeles niimoodi väljendab: tootmine, tehnoloogia, toorainete töötlemine, energeetika, transport ja logistika, ökoloogia, seired (atmosfääri, mere, maakoore), juhtimine, finantsmajandus, äri- ja kaubandustegevus, rakendusteadused jne. Igal pool tuleb ette dünaamilisi süsteeme ja nende matemaatilise modelleerimise vajadust. KF on seega tööstusmatemaatika oluline tööriist.

Mõõtmine

KF tagasisidestab reaalse DS mõõtmise tulemused selle süsteemi olekuparameetrite täpsustamiseks. Sõltugu mõõtmistulemus **y** olekuvektorist **x** nii:

$$\mathbf{y}(k+1) = H(k+1)\mathbf{x}(k+1) + \mathbf{v}(k+). \quad (6)$$

Siin **y** – p-mõõtmeline väljund, mõõtetulemus; H(k+1) on teadaolev maatriks mõõtmetega p×n; **v** - häiritused, vead; jällegi keskväärtusega null valged Gaussi mürad teadaoleva kovariatsioonimaatriksiga R; **v**(k+1) ~ *N*(0, R(k+1)). Allpool oletame, et ka H on konstantne, H(k+1) = H. Algolek **x**(0) ja müravektorid **w**(k+1), **v**(k+1) eeldatakse samuti olevat vastastikku sõltumatud. Indeksiks valemis (3) on k+1 rõhutamaks, et mõõtmine on aposterioorne tegevus, sooritatakse pärast seda, kui DS on olekust k juba olekusse k+1 üle läinud.

Mõõtmistulemuse esitamine kujul (3) on õigustatud: reaalsete süsteemide korral on sageli raske kui mitte võimatu mõõta süsteemi parameetrite väärtusi vahetult. Võib juhtuda, et vajalikule mõõtmiskohale on raske ligi pääseda või oleks tarvis teha nii palju ja nii paljudes kohtades mõõtmisi, et see teeks kogu tulemuse mõttetult kalliks. Sel juhul tuleb „soodsatest kohtadest" saadud näidud kuidagi kas interpoleerida või ekstrapoleerida kogu vajalikku piirkonda. Aga võib ette tulla ka igasuguseid muid olukordi, kus mõõtmisoperaatori H olemasolu on loomulik ja möödapääsmatu.

Asjaolu, et on välja mõeldud ja laialdaselt kasutusele võetud Kalmani filter DS uurimisel, näitab, et teadlased ja praktikud ei usalda täielikult ei teoreetilist mudelit (1) ega ka mõõtmistulemusi (3).

Kalmani filter

Nagu eespool mainisime ja ka näitest paistab, on aprioorsest hinnangust $\mathbf{x}^-$ DS oleku hindamisele vähe ja ka mõõtmised osutuvad ebatäpseiks. Nüüd on aeg rääkida ideest leida selle info alusel uus, nn. aposterioorne hinnang DS olekuvektorile kujul

$$\mathbf{x}^*(k+1) = \mathbf{x}^-(k+1) + \Delta\mathbf{x}(k+1)\ . \quad (4)$$

R. E. Kalman pakkus välja idee, et parandusliige $\mathbf{\Delta x(}k+1\mathbf{)}$ tuleb igal sammul arvutada nii, et uue olekuhinnangu $\mathbf{x}^*\mathbf{(}k+1\mathbf{)}$ matemaatiline ootus võrduks süsteemi oleku matemaatiline ootusega, st. et hinnang oleks nihutamata, ja teiseks, et selle puhul hinnangu vea kovariatsiooni matemaatiline ootus oleks minimaalne üle kõigi hinnangualgoritmide. Matemaatiliselt võib need tingimused kirja panna nii (tähistused valemeis (1) ja (4)):

$$E[\mathbf{x}^*\mathbf{(}k+1\mathbf{)}] = E[\mathbf{x}(k+1)] \; ; \quad E[(\mathbf{x}^*\mathbf{(}k+1\mathbf{)-x}(k+1))(\mathbf{x}^*\mathbf{(}k+1\mathbf{)-x}(k+1))^T] \rightarrow \min .$$

Siin $\mathbf{x}(k+1)$ tähistab DS tegelikku olekut, mida vaatleja ei tea. Toome sisse tähistuse $e(k+1) = \mathbf{x}^*\mathbf{(}k+1\mathbf{)} - \mathbf{x}(k+1)$ uue hinnangu vea jaoks (igaks juhuks veelkord – see hinnang pole teada, sest võrduse paremal poolel lahutatav on tundmatu ning jääbki selleks). R. E. Kalman tegi oma ajaloolises artiklis ettepaneku arvutada uus hinnang (4) DS olekule valemiga

$$\mathbf{(5)} \qquad \mathbf{x}^*\mathbf{(}k+1\mathbf{) = x}^-\mathbf{(}k+1\mathbf{)+} K(k+1)\mathbf{[y(}k+1\mathbf{) –} H\mathbf{x(}k+1\mathbf{)],}$$

kus H on mõõtmiste maatriks valemist (3) ja K – niinimetatud Kalmani kaal, mis saadakse seostest

$$dP(k+1)/dK = 0, \text{ ehk } dP(k+1)/dK = dE\{e(k+1)e^T(k+1)\}/dK = 0.$$

Viimast tingimust rahuldab maatriks K kujul

$$(6) \quad K(k+1) = P^-(k+1)H^T[HP^-(k+1)H^T + R(k+1)]^{-1} ,$$

kus $P^-(k+1) = AP(k)A^T + Q$ on aprioorne hinnang kovariatsioonimaatriksile P. Kokkuvõttes saame DS seisundi uue aposterioorse hinnangu seose (4) alusel:

$$\mathbf{x}^*\mathbf{(}k+1\mathbf{) = x}^-\mathbf{(}k+1\mathbf{) +} K(k+1)[\mathbf{y}(k) - H\mathbf{x}^*\mathbf{(}k\mathbf{)}] =$$
$$= A\mathbf{x}^*\mathbf{(}k\mathbf{)} + B\mathbf{u}(k) + K(k+1)[\mathbf{y}(k) - H\mathbf{x}^*\mathbf{(}k\mathbf{)}] \mathbf{.}$$

Aposterioorne hinnang kovariatsioonimaatriksile P, mida kasutatakse juba järgmisel iteratsioonil, st. üleminekul ajahetkelt k + 1 hetkele k + 2, leitakse nii:

$$P(k+1) = AP(k)A^{\mathbf{T}} + Q - AP(k)H^{\mathbf{T}}R^{-1}HP(k)A^{\mathbf{T}} .$$

Avaldisest (6) Kalmani kaalu leidmiseks näeme, et kui mõõtmisvead lähenevad nullile, st. $R \rightarrow 0$, siis $K \rightarrow H^{-1}$, st. kui mõõtmine muutub täpsemaks, siis usaldatakse seda ka järjest rohkem ning ennustatud mõõtmist $H\mathbf{x}^{*}(k)$ vähem. Ja vastupidi – kui aprioorse olekuhinnangu kovariatsioon $P^{-}(k+1)$ läheneb nullile, usaldatakse mõõtetulemust **y** järjest vähem.

Võtame lõpuks kokku Kalmani filtri kogu arvutuseeskirja skeemis, mille abil seda tavaliselt vastavas kirjanduses tehakse:

„Ennustus" (*Prediction*)

Leida aprioorne olek:

$$\mathbf{x}^{-}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$

Leida aprioorne kovariatsioon:

$$P^{-}(k+1) = AP(k)A^{T} + Q$$

„Parandus" (*Correction*)

Arvutada Kalmani kaal:

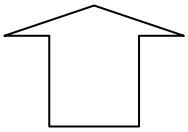
$$K(k+1) = P^{-}(k+1)H^{T}\,(HP^{-}(k+1)\,H^{T} + R)^{-1}$$

Korrigeerida aprioorne hinnang:

$$\mathbf{x}^{*}(k+1) = \mathbf{x}^{-}(k+1)+K(k+1)[\mathbf{y}(k+1)-H\mathbf{x}^{-}(k+1)]$$

Aposterioorne kovariatsioon:

$$P(k+1) = (I - K(k+1)H\,)P^{-}(k+1)$$



Alghinnangud **x**(0) ja P(0)

Näide

Illustreerime teksti lihtsa näitega. Vaatleme dünaamilist süsteemi, milleks on paigalt piki sirgjoont liikuma hakkav auto, millele mõjub konstantne lükkav jõud, st. auto teoreetiline kiirendus on konstantne.

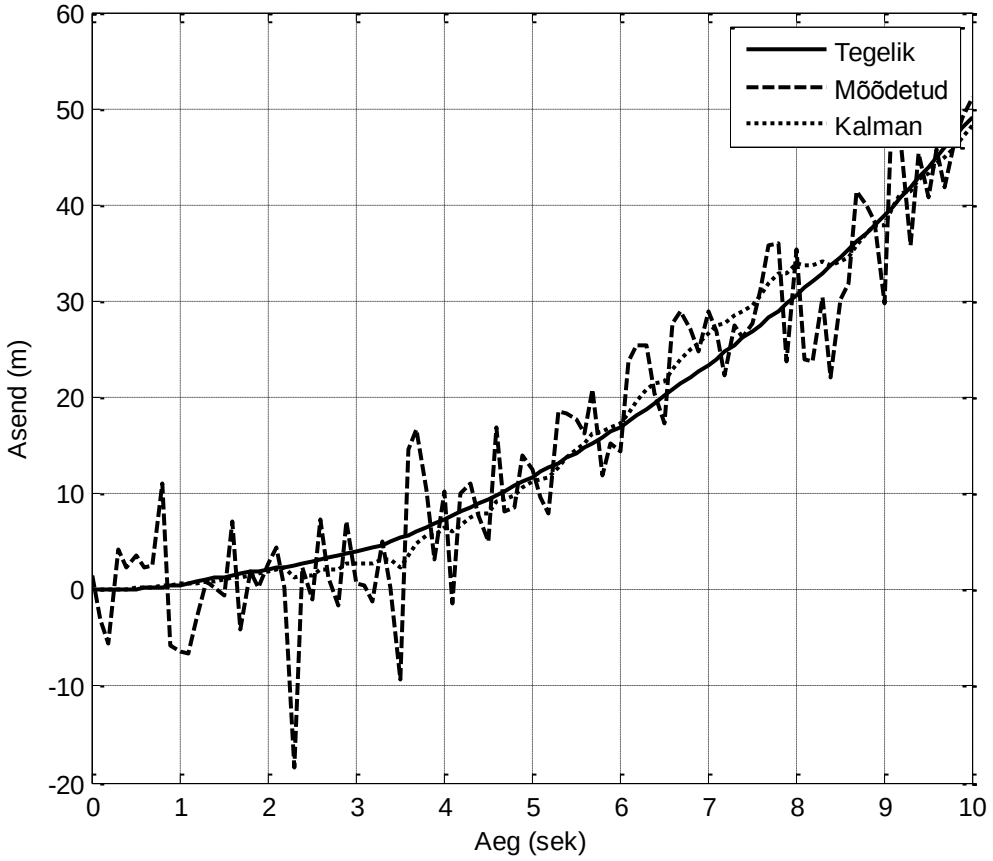
Olekuvektor $\mathbf{x} = [p\ ,\ v]^T$, kus p on auto asend (ühedimensionaalne nihe piki liikumissirget) ja v - kiirus. Süsteemi võrrandid on sellised:

$$\begin{cases} p(k+1) = p(k) + t\,v(k) + (t^2 u(k))/2 + w_1(k), \\ v(k+1) = v(k) + t\,u(k) + w_2(k). \end{cases}$$

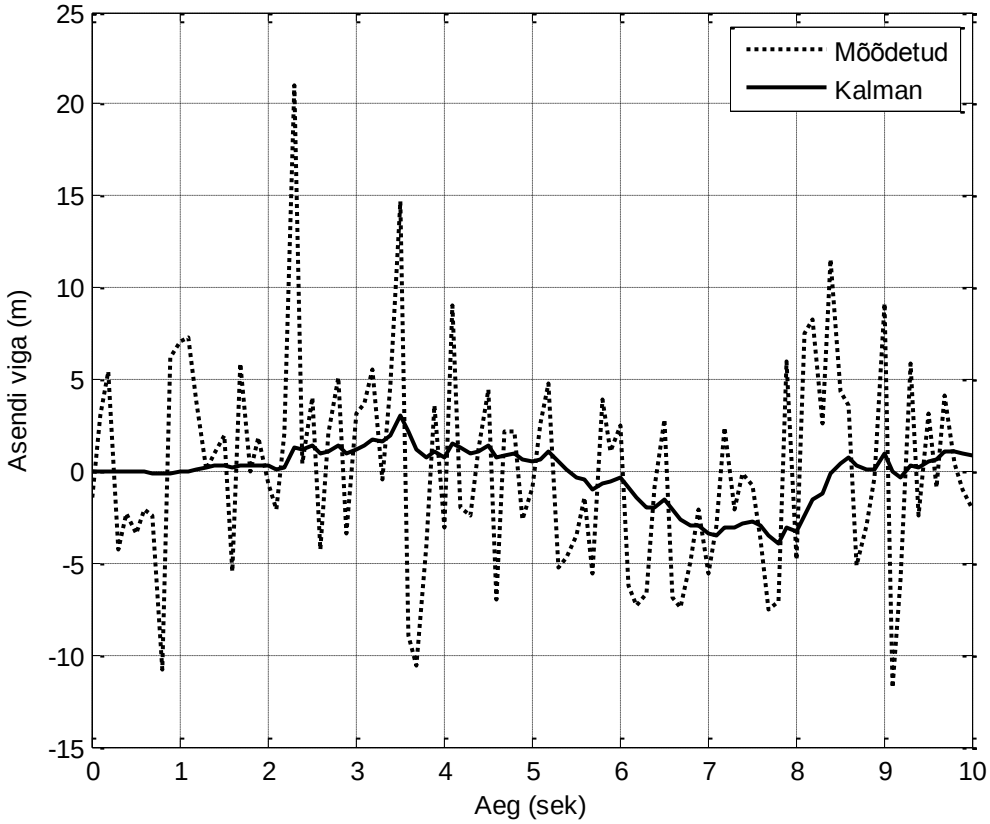
Siin t tähistab ajavahemikku üleminekul ajahetkelt k hetkele k+1; müravektoriks on $[w_1(k)\ ,\ w_2(k)]^T$. Häiritused w_1 ja w_2 tulenevad mitmesugustest välistest asjaoludest (teeolud ja -materjal, tõus või langus jne.), need on meile teadmata ja saame kasutada vaid statistilisi omadusi – tegemist on valge müraga. Esimene võrrand iseloomustab teepikkuse muutust sõltuvalt kiirusest v ja kiirendusest u, teine – kiiruse muutumist suuruse u võrra aja t jooksul.

Joonisel 1 on esitatud arvutisimulatsiooni tulemused: auto tegelik asend, mida vaatleja ei tea, mõõdetud asend ja Kalmani filtriga saadud uus hinnang auto asendile. Joonisel 2 näeme graafiliselt mõõtmisvigu ja Kalmani hinnangu vigu. Lisatud on ka joonis auto kiiruse muutuse kohta: joonisel 3 on kujutatud tegelik (st. arvutisimulatsioonis genereeritud) kiirus ja Kalmani filtriga saadud täpsustus.

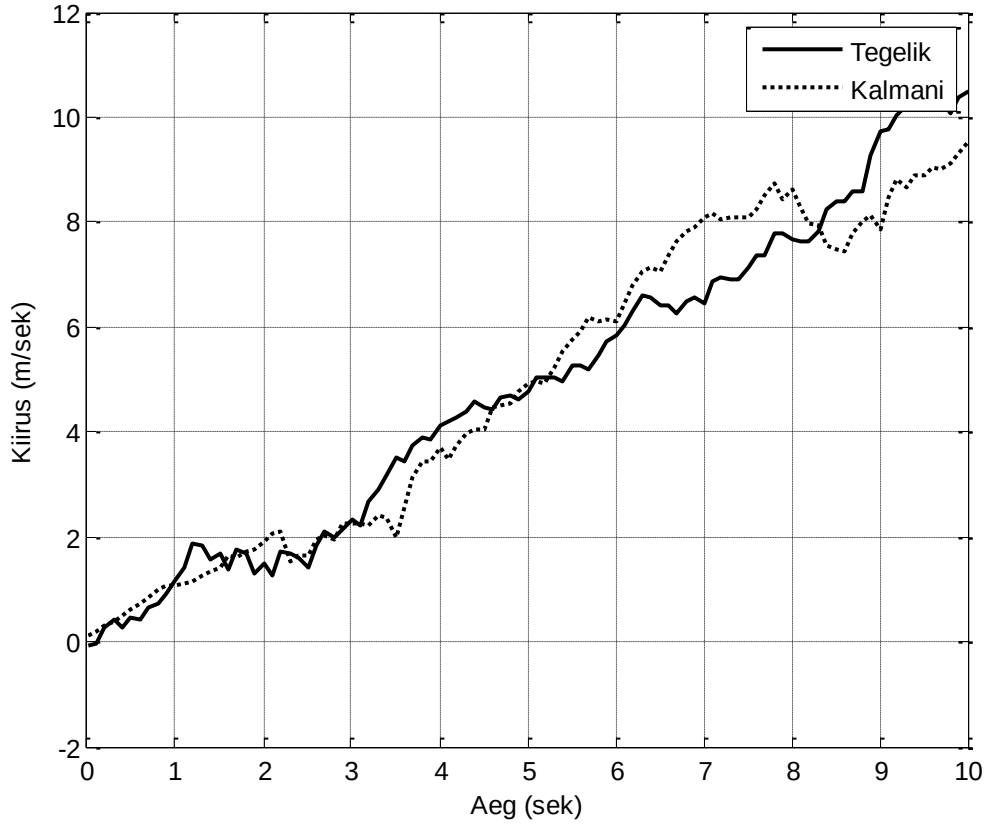
Joonis 1. Tegelik, mõõdetud ja Kalmani asend



Joonis 2. Mõõdetud ja Kalmani asendi vead



Joonis 3. Kiirus (tegelik ja Kalmani)



Alates 1960. aastast on Kalmani filtrit rakendatud väga erinevate DS modelleerimisel ning see tegevus jätkub.

Järgnevalt **MATLAB**-i kood, mis realiseerib ülalkirjeldatud simulatsiooni.

```
function kalman(aeg, dt)
%
% Kalmani filtri simulatsioon.
% Auto liikumine teel.
% SISENDID:
% aeg = simulatsiooni kestvus (sekundites);
% dt = ajasamm (sekundites).
%
%  Pöördumise näide:
```

```
%  >> kalman(10,0.1)
%
measnoise = 1; % Auto asendi mõõtmisviga (meetrites).
accelnoise = 0.05; % Kiirenduse mõõtmisviga
(meetrit/sec^2).
a = [1 dt; 0 1]; % Ülekandemaatriks (teoreetilises
mudelis).
b = [dt^2/2; dt]; % Vabaliige (teoreetilises mudelis).
c = [1 0]; % Mõõtmismaatriks.
x = [0; 0]; % Algasend ([algasend, algkiirus]).
xhat = x; % Algasendi hinnang; langeb kokku algasendiga.
Sz = measnoise^2; % Mõõtmisvea kovariatsioon.
Sw = accelnoise^2 * [dt^4/4 dt^3/2; dt^3/2 dt^2]; %
Kiirenduse
        % mõõtmise vea kovariatsioon.
P = Sw; % Algne aposterioorne kovariatsioon.
% Luuakse massiivid edasiseks visualiseerimiseks,
joonisteks.
pos = []; % Tegelik asend.
poshat = []; % Hinnatud asend, aprioorne.
posmeas = []; % Mõõdetud asend.
vel = []; % Tegelik kiirus.
velhat = []; % Hinnatud aprioorne kiirus.
for t = 0 : dt: aeg,
% Kasutame konstantset kiirenduse muutu 1 meeter/sec^2.
u = 1;
% Aprioorse ennustuse simulatsioon.
ProcessNoise = accelnoise * [(dt^2/2)*randn; dt*randn];
x = a * x + b * u + ProcessNoise;
% Mõõtmise simulatsioon.
MeasNoise = measnoise * randn;
y = c * x + MeasNoise;
%
%  Ekstrapoleerime viimase aposterioorse
```

```
%  asendihinnangu järgmisele ajahetkele, st leiame
aprioorse hinnangu.
%
xhat = a * xhat + b * u;
% Parandusvektor.
Inn = y - c * xhat;
% Paranduse kovariatsioon.
s = c * P * c'+ Sz;
% Kalmani maatriksi leidmine.
K = a * P * c'*inv(s);
% Olekuvektori uuenduse leidmine.
xhat = xhat + K * Inn;
% Uue oleku vea kovariatsioon.
P = a * P * a'- a * P * c'* inv(s) * c * P * a'+ Sw;
% Parameetrite salvestamine edasiseks visualiseerimiseks.
pos = [pos; x(1)]; %#ok<AGROW>
posmeas = [posmeas; y]; %#ok<AGROW>
poshat = [poshat; xhat(1)]; %#ok<AGROW>
vel = [vel; x(2)]; %#ok<AGROW>
velhat = [velhat; xhat(2)]; %#ok<AGROW>
end
% Tulemuste visualiseerimine - graafikud.
close all;
t = 0 : dt : aeg;
figure;
plot(t,pos, t,posmeas, t,poshat);
grid;
xlabel('Aeg (sekund)');
ylabel('Asend (meeter)');
title('Joonis 1 - Auto asend (tegelik, mõõdetud ja
parandatud)')
figure;
plot(t,pos-posmeas, t,pos-poshat);
grid;
```

```
xlabel('Aeg (sekund)');
ylabel('Asendi viga (meeter)');
title('Joonis 2 - Asendi mõõtmisviga ja asendi hinnangu
viga');
figure;
plot(t,vel, t,velhat);
grid;
xlabel('Aeg (sekund)');
ylabel('Kiirus (meeter/sekund)');
title('Joonis 3 - Auto kiirus (tegelik ja
hinnanguline)');
figure;
plot(t,vel-velhat);
grid;
xlabel('Aeg (sekund)');
ylabel('Kiiruse viga (meeter/sekund)');
title('Joonis 4 - Auto kiiruse veahinnang');
```


Iteratsioonimeetodid algebraliste võrrandite lahendamiseks

Iteratsioonimeetodiks nimetatakse teatud võtet võrrandite, võrrandisüsteemide, ekstreemumülesannete jms. ligikaudseks lahendamiseks. Iteratsioonimeetodite idee seisneb järgnevas: ülesandele leitakse mingi alglähend x_0, mille abil moodustatakse lähendite jada

$$x_1 ; x_2 ; x_3 ; \ldots ; x_n ; \ldots .$$

Teatud tingimustel koondub see jada ülesande täpseks lahendiks x^*. Loendaja n järkjärguline suurendamine tähendab iteratsioonisammude tegemist. See lõpetatakse küllalt suure n korral ja viimane lähend x_n loetakse ülesande lahendiks.
Allpool vaatleme võrrandit kujul

$$x = g(x) \quad (1)$$

või kujul

$$f(x) = 0. \quad (2)$$

Uurime nende lahendamist vastavalt hariliku iteratsioonimeetodi ja Newtoni iteratsioonimeetodi abil. Siin **g** ja **f** tähistavad etteantud ühemuutuja funktsiooni.

Harilik iteratsioonimeetod

Olgu meil antud võrrand kujul (1). Pärast alglähendi x_0 fikseerimist moodustatakse lähendite jada järgmise eeskirjaga abil:

$$x_{k+1} = g(x_k) \text{ , } k = 1, 2, \ldots .$$

Jada $x_1; x_2; x_3; \ldots ; x_n; \ldots$ koondub võrrandi (1) täpseks lahendiks x^*, kui mingis jada elemente ja võrrandi täpset lahendit sisaldavas vahemikus on täidetud tingimus

$$| g'(x) | < 1. \quad (3)$$

Näide. Lahendada võrrand $e^{x/4} + x - 8 = 0$ (teha läbi iseseisvalt).

Hariliku iteratsioonimeetodi rakendamiseks tuleb teisendada võrrand kujule (1) . Seda on võimalik teha mitmel eri viisil, kusjuures sellest, kuidas on valitud funktsioon **g**, sõltub lahendite jada koondumise kiirus. Üks võimalus oleks näiteks $8 - e^{x/4} = x$. Seega nüüd $g(x) = 8 - e^{x/4}$.

Newtoni iteratsioonimeetod

Olgu meil antud võrrand kujul (2). Valime alglähendi x_0 ja koostame lähendite jada järgmise eeskirja abil:

$$x_{k+1} = x_k - f(x_k)/f'(x_k) \text{ , } k = 1, 2, \ldots .$$

Newtoni meetodi korral saame lähendite jada järgmise liikme x_{k+1}, kui leiame punktis x_k graafikule tõmmatud puutuja lõikepunkti x-teljega.

Näide. Lahendada võrrand $3^{x-4} - 2^x + 0{,}4 = 0$ (teha läbi iseseisvalt).

Newtoni meetodi korral on järgmise lähendi x_{k+1} arvutamiseks vaja ka funktsiooni $f(x)$ tuletise väärtust $f'(x_k)$ kohal x_k.