FYSIIKAN NUMEERISEN LASKENNAN ALKEET (3 op) J190308

Kevät 2011

Vastuuopettaja: Hannu Laamanen
Puhelin: (013) 251 4654
Sähköposti: hannu.laamanen@uef.fi
Toimipaikka: Itä-Suomen yliopisto, fysiikan ja matematiikan laitos (huone 216)

Kurssin tavoite ja sisältö: Kurssin tavoitteena on antaa perustiedot tietokoneen käytöstä fysiikan numeeristen ongelmien ratkaisemisessa. Kurssilla esimerkkiohjelmistona käytetään Matlabia. Kurssilla käydään läpi perusfunktiot, omien funktioiden tekeminen ja julkaisukelpoisten kuvien tuottaminen. Lisäksi kurssilla käsitellään lyhyesti matriisi- ja vektorilaskennan hyödyntämistä numeeristen ongelmien ratkaisussa.

Suoritustavat: aktiivinen osanotto luennoille ja harjoituksiin ja harjoitustyön tekeminen.

Suositeltavat taustatiedot: matriisilaskennan perusteet.

LUENTO/HARJOITUS AJAT:

- to 13.01. klo 12.15 – 14.00 MP103
- pe 14.01. klo 12.15 – 14.00 MP103
- to 20.01. klo 12.15 – 14.00 MP103
- pe 21.01. klo 12.15 – 14.00 MP103 (???)
- to 27.01. klo 12.15 – 14.00 MP103
- pe 28.01. klo 12.15 – 14.00 MP103
- to 03.02. klo 12.15 – 14.00 MP103
- pe 04.02. klo 12.15 – 14.00 MP103
- to 10.02. klo 12.15 – 14.00 MP102
- pe 11.02. klo 12.15 – 14.00 MP103
- to 17.02. klo 12.15 – 14.00 MP102
- pe 18.02. klo 12.15 – 14.00 MP103
- pe 25.02. klo 12.15 – 14.00 MP103

SISÄLLYSLUETTELO

1. Matlab ympäristö

Matlabin käynnistäminen pikakuvakkeesta tai käynnistä-valikosta avaa Command Window - komentoikkunan, sekä mahdollisesti Command History, Current Directory, Workspace ja Launch Pad - ikkunat.

Komentoikkunan kehotteen >> perään voidaan kirjoittaa komentoja, jotka erotetaan toisistaan joko pilkulla tai jos komennon tuottamaa tulosta ei haluta näytölle käytetään komennon lopussa puolipistettä. Komento toteutetaan painamalla ENTER (↵) painiketta. Aiemmin syötettyjä komentoja voi selata ↑ ja ↓ näppäimillä. Selatessa tietyn käytetyn komennon löytämistä voi helpottaa kirjoittamalla komennon alkukirjaimen tai sen alkuosan ja käyttämällä tämän jälkeen nuolinäppäimiä.

Kun muuttujalle (variable) annetaan jokin arvo, sijoitusoperaattorina käytetään = merkkiä. Kokeile seuraavia komentoja:

```
>> a=1
>> a=1, b=2
>> a=1; b=2;
>> a=1, b=2;
```

Viimeksi käytetyn komennon tulos tallentuu automaattisesti myös muuttujaan **ans**. Lukujen esitystarkkuuteen voidaan vaikuttaa komennolla **format**. Kokeile seuraavia

```
>> pi
>> format long
>> pi
>> format bank
>> pi
>> format short
>> pi
```

Luku π merkitään Matlabissa siis **pi**. Muutamia muita Matlabissa esiintyviä vakioita ovat imaginääriyksikkö **i** tai **j**, ääretön **inf**, ei-luku **NaN**, liukuluvun tarkkuus **eps**, sekä pienin ja suurin käsiteltävissä oleva positiivinen reaaliluku, **realmin** ja **realmax**.

```
>> i                    >> eps
>> j                    >> realmin
>> inf                  >> realmax
>> NaN
```

Komento **who** listaa luotujen muuttujien nimet.

```
>> who
```

kokeile myös komentoa **whos**

```
>> whos
```

Tällöin muuttujan nimen lisäksi näytetään tiedot myös muuttujan koosta (size, bytes) ja luokasta (class).

Luodun muuttujan voi poistaa komennolla **clear + muuttujan nimi**. Pelkkä **clear** poistaa kaikki muistissa olevat muuttujat. Komento **clc** tyhjentää komentoikkunan, mutta ei poista olemassa olevia muuttujia.

```
>> clear a, whos
>> clear, whos
>> clc
```

Tietoa Matlabin komennoista voi hakea komennolla **help**. Pelkkä **help** komento antaa listauksen eri komentoryhmistä ja **help + komentoryhmän nimi** listaa kaikki kyseisen ryhmän komennot. Kokeile seuraavia

```
>> help
>> help general
>> help who
>> help help
```

Komento **pwd** näyttää sen hetkisen työhakemiston ja komennolla **cd** voidaan vaihtaa työhakemistoa. Komento **dir** listaa hakemiston siältämät tiedostot. Komento **what** listaa vain työhakemistossa olevat Matlabin omat .mat-tiedostot.

```
>> pwd
>> cd ..
>> cd C:\
>> dir
```

Komento **path** näyttää käytössä olevat hakemistopolut ja komennolla **addpath** voi luoda sellaisen. Matlab löytää poluilla merkattujen hakemistojen tiedostot suoraan riippumatta siitä, mikä on senhetkinen työhakemisto.

Muistissa olevat muuttujat voi tallettaa työhakemistoon .mat-tiedostoon komennolla **save**.

```
>> a=1; b=2;
>> save testi
>> clear
>> load testi
>> whos
```

```
>> a=1; b=2;
>> save testi.mat a
>> clear
>> load testi
>> whos
```

Työhakemistosta voi poistaa tiedostoja komennolle **delete**.

```
>> delete testi.mat
>> dir
```

Matlab istunto päätetään komennolla **exit** tai **quit**.

```
>> exit
>> quit
```

Sovia tiettyyn aiheeseen liittyviä Matlab komentoja voi etsiä käskyllä **lookfor**. Kokeile esimerkiksi

```
>> lookfor('filtering')
```

2. Matriisit ja niiden peruslaskutoimitukset

2.1 Matriisien muodostaminen

Kaikki Matlabin käyttämät muuttujat määritellään n×m matriiseina, joiden alkiot ovat joko reaali- tai kompleksilukuja tai merkkejä. Jos n = 1 ja m > 1, on kyseessä rivivektori. Jos taas n > 1 ja m = 1 on kyseessä sarakevektori. Molempien, n:n ja m:n, ollessa ykkösiä on kyseessä skalaari.

Vektoreja tai matriiseja muodostettaessa käytetään hakasulkumerkkejä [ ]. Matriisin sarakkeet erotetaan toisistaan pilkuilla ja rivit puolipisteillä. Hakasulkuja voi käyttää myös sisäkkäin rivejä tai sarakkeita muodostettaessa. Matriisitranspoosi määritetään heittomerkillä '. Kokeile seuraavia

```
>> a=[1 2 3]
>> b=[1 2 3]'
>> c=[1 2 3;4 5 6;7 8 9]
>> d=[[1 4 7]',[2 5 8]',[3 6 9]']
>> e='testi'
```

Matriisi voidaan tarvittaessa kirjoittaa myös riveittäin.

```
>> a=[1 2 3
4 5 6
7 8 9
]
```

Matlabista löytyy myös joukko komentoja erilaisten matriisien luomiseksi. Esimerkiksi komento **rand** luo matriisin, jonka alkiot ovat satunnaislukuja väliltä [0,1].
Kokeile seuraavia:

```
>> a=ones(3,4)
>> a=zeros(4,3)
>> a=eye(4,4)
>> b=diag(a)
>> a=diag([1 2 3 4])
>> a=rand(3,3)
>> a=rand(3,3)
```

Matriisin kunkin alkion paikka on määritelty rivi- ja sarakeindeksillä. Jos matriisista halutaan valita tietty useamman alkion käsittävä alue, käytetään kaksoispistettä ja sen molemmin puolin alueen rajaavia indeksejä. Pelkkä kaksoispiste tarkoittaa kaikkia rivi- tai sarakeindeksejä. Kaksoispistettä voidaan käyttää myös luotaessa vektoreita tai matriiseja, joiden alkioiden arvo muuttuu tasaisesti tietyllä porrastuksella; mikäli porrastuksessa käytettävän askeleen suuruutta ei ole erikseen määritelty oletusarvona käytetään ykköstä.

```
>> a=[1 2 3 4;5 6 7 8;9 10 11 12]
>> a(1,3)
>> a(1,3)=5
>> a(1,:)
>> a(:,1:2)
>> b=[0:5]
>> c=[0:1:2;1:1:3;2:1:4]
>> c(:,end)
```

Komento **length** määrittää vektorin alkioiden lukumäärän ja komento **size** tuottaa kaksi lukua, jotka ilmoittavat matriisin rivien ja sarakkeiden lukumäärän.

```
>> k=length(b)
>> [n,m]=size(c)
```

Komentoa **sort** voidaan käyttää matriisin alkioiden järjestämiseksi suuruusjärjestykseen. Komento tuottaa myös vektorin, joka sisältää syötteenä annetun vektorin suuruus-järjestykseen järjestettyjen alkioiden indeksit.
Kokeile

```
>> b=rand(1,5)
>> [n,m]=sort(b)
```

Kokeile myös seuraavia komentoja

```
>> a=[1 2 3;4 5 6]
>> b=rot90(a)
>> c=flipud(a)
>> d=fliplr(a)
```

Tyhjä alkioton matriisi määritellään pelkkien hakasulkujen [ ] avulla. Matriisista voi kätevästi poistaa rivejä tai sarakkeita em. merkinnän avulla, mutta ei kuitenkaan yksittäisiä alkioita, koska lopputulos ei olisi enää matriisi.

```
>> a=[1:4;2:5;0:3]
>> a(:,1)=[ ]
>> a(end,:)=[ ]
>> a(1,2)=[ ]
```

Matlabissa matriisit voivat olla myös kolmi- tai useampi ulotteisia. Esimerkiksi RGB-kuvat ovat kolmiulotteisia matriiseja.

```
>> clear a
>> a(:,:,1)=[1 2;3 4]
>> a(:,:,2)=[5 6;7 8]
>> a
```

Tehtävä: Muodosta matriisi A

$$A=\begin{bmatrix} 1 & 11 & 6 & 2 \\ -4 & 15 & 5 & -4 \\ 3 & -23 & 12 & -4 \\ 8 & 1 & 3 & 6 \end{bmatrix}$$

ja järjestä sen sarakkeet em. komentojen avulla, niin että sarakkeiden alkioiden summat kasvavat oikealta vasemmalle. Sarakkeiden alkioiden summat saat laskettua komennolla **sum**.

2.2 Matriisilaskenta

Matriisien yhteen- ja vähennyslaskussa voidaan käyttää + ja - merkkejä. Tällöin yhteenlaskettavien tai toisistaan vähennettävien matriisien on oltava saman kokoiset. Skalaari voidaan lisätä tai vähentää suoraan matriisin koosta riippumatta.

```
>> a=[1:3;4:6;7:9]
>> a+a
>> a-a
>> a+1
```

Matriisin rivi- ja sarakesummat saadaan lasketuksi komennolla **sum**.

```
>> sum(a)
>> sum(a,1)
>> sum(a,2)
```

Matriisin alkioiden rivi- ja sarakesuuntaiset keskiarvot lasketaan komennolla **mean**. Mikäli tarkoituksena on laskea matriisin kaikkien alkioiden keskiarvo, tarvitaan kaksi sisäkkäistä komentoa.

```
>> mean(a)
>> mean(a,1)
>> mean(a,2)
>> mean(mean(a))
```

Kahden skalaarin tulo voidaan laskea merkin * avulla

```
>> 2*3
>> 3*2
```

Matriisien ja vektoreiden kertolaskussa on lisäksi otettava huomioon keskenään kerrottavien matriisien/vektoreiden oikea koko ja järjestys. Esimerkiksi kahden saman pituisen rivi ja sarakevektorin tulo on joko skalaari tai matriisi, riippuen siitä kummassa järjestyksessä nämä kaksi vektoria kerrotaan keskenään.

```
>> a=[1 2 3]
>> b=[-2 4 3]'
>> a*b
>> b*a
```

Tapausta, jossa kertolaskun tuloksena on skalaari, kutsutaan vektorien sisä-, skalaari- tai pistetuloksi. Jos vektoreiden skalaaritulon tulos on nolla, vektorit ovat toisiaan vastaan kohtisuorassa. Sarakevektoreille **a** ja **b** ovat voimassa seuraavat pistetulon määrittelevät kaavat/merkinnät

$$\mathbf{a}\cdot\mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\|\cos\theta = \mathbf{a}^{\mathrm{T}}\mathbf{b} = \mathbf{b}^{\mathrm{T}}\mathbf{a} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + \cdots + a_n b_n$$

Tulokseksi matriisin tuottavaa kertolaskua kutsutaan vektoreiden ulkotuloksi.

Fyysikoiden usein tarvitsema vektoreiden ristitulo voidaan laskea komennolla **cross**. Tällöin kerrottavien vektoreiden on oltava kolmen alkion pituisia. Sarakevektoreille **a** ja **b** ovat voimassa seuraavat ristitulon määrittelevät kaavat/merkinnät

$$\mathbf{a}\times\mathbf{b}=\hat{\mathbf{n}}\|\mathbf{a}\|\|\mathbf{b}\|\sin\theta=\begin{vmatrix}\hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}}\\ a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\end{vmatrix}=(a_2b_3-b_2a_3)\hat{\mathbf{i}}+(a_3b_1-b_3a_1)\hat{\mathbf{j}}+(a_1b_2-b_1a_2)\hat{\mathbf{k}}$$

```
>> cross([1:3]',[1 0 -2]')
```

Vektorin pistetulo itsensä kanssa on sama kuin vektorin pituuden neliö. Vektorin normi eli pituus voidaan laskea komennolla **norm**. Luvulle voidaan kirjoittaa eksponentti merkillä ^.

```
>> a=[1 1]'
>> a'*a
>> norm(a)
>> ans^2
```

Matriisien $A=(a_{ij})\in\mathrm{R}^{n\times m}$ ja $B=(b_{jk})\in\mathrm{R}^{m\times r}$ tulo on matriisi $AB=C=(c_{ik})\in\mathrm{R}^{n\times r}$, missä $c_{ik}=\sum_{j=1}^{m}a_{ij}b_{jk}$. Matriisissa A on siis oltava sama määrä sarakkeita, kuin matriisissa B on rivejä.

Esimerkkiksi matriisien $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$ ja $B=\begin{bmatrix}7 & 8\\ 9 & 10\\ 11 & 12\end{bmatrix}$ tulomatriisi on

$$C=AB=\begin{bmatrix}1\times7+2\times9+3\times11 & 1\times8+2\times10+3\times12\\ 4\times7+5\times9+6\times11 & 4\times8+5\times10+6\times12\end{bmatrix}=\begin{bmatrix}58 & 64\\ 139 & 154\end{bmatrix}.$$

```
>> A=[1:3;4:6]
>> B=[7:2:11;8:2:12]'
>> C=A*B
>> C=B*A
```

Matriisit voidaan kertoa myös alkioittain; tällöin kertomerkin edessä käytetään pistettä .* ja matriisien on oltava saman kokoiset.

```
>> A.*B'
>> B'.*A
```

Eksponentin ^2 käyttö matriisin tapauksessa tarkoittaa matriisin tuloa itsensä kanssa. Tällöin matriisin on oltava neliömatriisi. Merkintä .^2 tarkoittaa sitä, että matriisin alkiot korotetaan neliöön. Tällöin matriisin ei tarvitse olla neliömatriisi.

```
>> a=[1 2;3 4]
>> a*a
>> a^2
>> a.^2
```

Matriisit voidaan myös jakaa alkioittain. Tällöin käytetään operaatiota ./ Tällöin matriisien on oltava keskenään saman kokoiset.

```
>> A./B'
```

Jakolasku A/B ilman pistettä tuottaa tulokseksi matriisiyhtälön XB=A ratkaisun (X=A/B) ja merkintä A\B matriisiyhtälön AX=B ratkaisun (X=A\B). Matriisiyhtälöiden ratkaisun tarkkuus riippuu matriisien A ja B ominaisuuksista ja dimensioista. Matriisien ei tarvitse olla neliömatriiseja.

```
>> A=[-5 0 7;1 1 3;9 6 2]      >> X=A\B
>> B=[1 4 8;8 2 1;6 6 3]       >> A*X
>> X=A/B
>> X*B
```

Neliömatriisille A voidaan määrittää komennolla **inv** käänteismatriisi A^{-1} siten että kerto-laskut $A^{-1}A$ ja AA^{-1} tuottavat tulokseksi yksikkömatriisin I, jonka kaikki diagonaalialkiot ovat ykkösiä ja muut alkiot nollia. Jos matriisi on singulaarinen, käänteismatriisia ei voida laskea.

```
>> A=[1 2 -3;4 5 6;7 8 9]      >> A=[1 1 1;2 2 2;1 2 3]
>> B=inv(A)                    >> inv(A)
>> A*B
>> B*A
```

Matriisin determinantti lasketaan komennolla **det**. Singulaarisen matriisin determinantti on nolla.

```
>> A=[1 1 1;2 2 2;1 2 3]
>> det(A)
```

Matriisin ominaisarvot ja ominaisvektorit lasketaan komennolla **eig**. Tulokseksi saadaan kaksi matriisia, joista toisen sarakkeet ovat alkuperäisen matriisin ominaisvektoreita ja toisen diagonaalialkiot ominaisarvoja. Ratkaistut ominaisvektorit ovat toisiaan vastaan kohtisuorassa ja niiden pituudet on skaalattu ykkösiksi.

```
>> [V,D]=eig(A)
```

Tehtävä: Ratkaise matriisiyhtälö AXB=C, missä

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\quad B=\begin{bmatrix}3 & 3 & 3 & 4 & -2 & -1 & 3 & -1\\ 2 & 1 & 5 & 2 & 0 & -3 & 2 & -2\\ -2 & -1 & 0 & -4 & -4 & 1 & 0 & 4\\ -2 & 2 & 4 & -5 & 5 & 3 & 1 & -5\\ -2 & 0 & -3 & 4 & 1 & 0 & 3 & 3\\ 0 & -1 & 5 & -3 & -1 & 1 & -4 & 5\\ 2 & 2 & -2 & -2 & 0 & -3 & 1 & 5\\ -2 & 1 & -2 & 2 & -2 & -1 & -4 & 3\end{bmatrix}\quad C=\begin{bmatrix}-526 & -1360\\ 1884 & 4086\\ 3198 & 7534\\ -1039 & -2571\\ -812 & -1634\\ -625 & -1221\\ 569 & 1383\\ 3419 & 7519\end{bmatrix}^{T}$$

Käytä lopuksi komentoa **char** ratkaisemaasi matriisiin X. >> char(X)

2.3 Yhtälöryhmien ratkaiseminen

Edellisessä kappaleessa mainittu matriisien ”jakolasku” operaatio \ soveltuu hyvin lineaaristen yhtälöryhmien ratkaisemiseen. Jos yhtälöitä on enemmän kuin muuttujia, kyseessä on ylimäärätty yhtälöryhmä. Tällöin Matlabin antama ratkaisu on eräänlainen pienimmän neliösumman ratkaisu.

Esimerkiksi jos ratkaistavana on yhtälöryhmä

$$\begin{cases} 2x_1 + x_2 - 3x_3 + x_4 + 4x_5 = 7 \\ 6x_1 + 8x_2 + x_4 - x_5 = 4 \\ 2x_2 + 2x_3 - 3x_4 + x_5 = 8 \\ -x_1 + 7x_2 + 3x_3 - 3x_4 + x_5 = -1 \\ x_1 + x_2 + x_3 + x_4 - 2x_5 = 4 \end{cases}$$

tämä on kirjoitettavissa matriisiyhtälöksi AX=B, missä

kerroinmatriisi $A = \begin{bmatrix} 2 & 1 & -3 & 1 & 4 \\ 6 & 8 & 0 & 1 & -1 \\ 0 & 2 & 2 & -3 & 1 \\ -1 & 7 & 3 & -3 & 1 \\ 1 & 1 & 1 & 1 & -2 \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$ ja $B = \begin{bmatrix} 7 \\ 4 \\ 8 \\ -1 \\ 4 \end{bmatrix}$.

Yhtälöryhmän ratkaisu $A \setminus B = X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 5 \\ -3 \\ 11 \\ 5 \\ 7 \end{bmatrix}$.

Komennolla **rref** yhtälöryhmää vastaava matriisi [A,B] saadaan redusoituun porrasmuotoon. Tästä muodosta on helposti pääteltävissä onko yhtälöryhmällä ratkaisua.

Kirjoita edellisen esimerkin matriisit A ja B, sekä toteuta komento

```
>> rref([A,B])
```

Tehtävä: Muodosta yhtälöryhmää

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 1 \\ -x_1 - x_2 + x_5 = -1 \\ -2x_1 - 2x_2 + x_5 = 1 \\ x_3 + x_4 + x_5 = -1 \\ x_1 + x_2 + 2x_3 + 2x_4 + 2x_5 = 1 \end{cases}$$

vastaavat matriisit A ja B, sekä muuta matriisi [A,B] redusoituun porrasmuotoon. Mitä yhtälöryhmän ratkaisusta on pääteltävissä. Kokeile myös yhtälöryhmän ratkaisemista jakolasku operaation \ avulla.

3. Alkeisfunktiot

3.1 Muutamia alkeisfunktioita

Matlabista löytyy suuri joukko valmiita alkeisfunktioita.

Trigonometriset funktiot

Perusfunktiot:

sin	- sini	
cos	- kosini	
tan	- tangentti	$\tan(x)=\frac{\sin(x)}{\cos(x)}$
cot	- kotangentti	$\cot(x)=\frac{\cos(x)}{\sin(x)}$
sec	- sekantti	$\sec(x)=\frac{1}{\cos(x)}$
csc	- kosekantti	$\csc(x)=\frac{1}{\sin(x)}$

Arkus-funktiot: Arkus-funktiot ovat trigonometristen perusfunktioiden käänteisfunktioita.

asin	- arkussini
acos	- arkuskosini
atan	- arkustangentti
atan2	- arkustangentti (kaikki neljännekset)
acot	- arkuskotangentti
asec	- arkussekantti
acsc	- arkuskosekantti

Hyperboliset funktiot:

sinh	- hyperbelisini	$\sinh(x)=\frac{1}{2}\left(e^{x}-e^{-x}\right)$
cosh	- hyperbelikosini	$\cosh(x)=\frac{1}{2}\left(e^{x}+e^{-x}\right)$
tanh	- hyperbelitangentti	
coth	- hyperbelikotangentti	
sech	- hyperbelisekantti	
csch	- hyperbelikosekantti	

Area-funktiot: Area funktiot ovat hyperbolisten funktioiden käänteisfunktioita.

asinh	- areahyperbelisini
acosh	- areahyperbelikosini
atanh	- areahyperbelitangentti
acoth	- areahyperbelikotangentti
asech	- areahyperbelisekantti
acsch	- areahyperbelikosekantti

Trigonometristen funktioiden arvot ovat aina radiaaneja.

Komento **fplot** piirtää annetun funktion kuvaajan. Syötteeksi komennolla annetaan piirrettävä funktio (heittomerkeissä), sekä funktion muuttujalle piirtoalueen määrittävät rajat. Jokseenkin samaan tapaan toimii komento **ezplot**. Kuvaajien piirtämiseen voi käyttää myös **plot** komentoa. Syötteeksi plot komennolle annetaan pisteiden x ja y arvot sisältävät vektorit tai pelkästään y arvot sisältävä vektori. Grafiikkaan liityvien komentojen tarkempi käsittely luentomonisteen luvussa 5.

Kokeile seuraavia komentoja:

```
#1
>> fplot('sin(x)',[0,2*pi])
>> hold on
>> fplot('cos(x)',[0,2*pi],'r')
>> legend('sin(x)','cos(x)')

#2
>> figure
>> fplot('sinh(x)',[-5,5])
>> hold on
>> fplot('cosh(x)',[-5,5],'g')
>> legend('sinh(x)','cosh(x)')
```

```
#3
>> figure
>> fplot('sec(x)',[0 2*pi])
>> fplot('sec(x)',[0 2*pi -50 50])

#4
>> a=2*pi*[0:.1:1]
>> b=sin(a)
>> plot(a,b)
>> a=2*pi*[0:.001:1];
>> plot(a,sin(a))
```

Eksponentti, logaritmi ja neliöjuuri

exp	- eksponenttifunktio, e^x
log	- luonnollinen logaritmi
log10	- kymmenkantainen logaritmi
sqrt	- neliöjuuri

Kompleksilukuihin liittyvät komennot

abs	- itseisarvo (kompleksiluvun moduli)
angle	- vaihekulma
complex	- kompleksiluvun konstruointi reaali- ja imaginääriosista
conj	- kompleksikonjugaatti
imag	- kompleksiluvun imaginääriosa
real	- kompleksiluvun reaaliosa

```
>> log(exp(5))
>> log(1e5)
>> log10(1e5)
>> exp(i*2*pi)-1
```

```
>> a=2+3*j
>> b=abs(a)
>> c=angle(a)
>> b*exp(c*i)
>> d=conj(a)
>> sqrt(a*d)
```

Lukujen pyöristäminen

fix	- pyöristys nollaa kohti
floor	- pyöristys miinus ääretöntä kohti
ceil	- pyöristys ääretöntä kohti
round	- pyöristys lähintä kokonaislukua kohti

```
>> a=3*rand(1,10)-1.5
>> [a;fix(a);floor(a);ceil(a);round(a)]
```

3.2 Loogiset operaatiot, vertailu operaatiot ja etsintä-toiminto

Loogiset operaatiot

and	&	JA-operaatio tuottaa arvon 1, jos molemmat syötteeksi annetut luvut ovat nollasta poikkeavia, muulloin 0.
or	|	TAI-operaatio tuottaa arvon 0, jos molemmat syötteeksi annetut luvut ovat nollia, muulloin 1.
not	~	EI-operaatio kääntää totuusarvon päinvastaiseksi, $1 \rightarrow 0$, $0 \rightarrow 1$.
xor		JOKO-TAI-operaatio antaa arvon 1, jos vain toinen syötteeksi annetuista luvuista on nollasta poikkeava, muulloin 0.
any		Tuottaa arvon 1, jos ainakin yksi syötteeksi annetun vektorin alkioista on nollasta poikkeava.
all		Tuottaa arvon yksi, jos kaikki syötteeksi annetun verktorin alkiot ovat nollasta poikkeavia.

```
>> a=[0 0 1 1]'
>> b=[0 1 0 1]'
>> [a,b,and(a,b)]
>> [a,b,or(a,b)]
>> [a,b,not(and(a,b))]
>> [a,b,xor(a,b)]
```

```
>> 3&0
>> 3|0
>> any([0 0 1 0])
```

Tehtävä: Luo vektorit $a = \begin{bmatrix}0 & 0 & 1 & 1\end{bmatrix}^T$ ja $b = \begin{bmatrix}0 & 1 & 0 & 1\end{bmatrix}^T$ ja muodosta yllä olevista loogisista operaatioista komento, joka tuottaa vektorin $\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^T$, kun syötteenä käytetään vektoreita *a* ja *b*.

Vertailu operaatiot

eq	==	on yhtäsuuri kuin
ne	~=	on erisuuri kuin
lt	<	on pienempi kuin
gt	>	on suurempi kuin
le	<=	on pienempi tai yhtäsuuri kuin
ge	>=	on suurempi tai yhtäsuuri kuin

```
>> 1>2
>> 2>1
```

Etsintä-toiminto

Komento **find** etsii syötteeksi annetun matriisin tai vektorin nollasta poikkeavien alkioiden indeksit. Komennon voi kätevästi yhdistää edellä esitettyihin loogisiin- ja vertailu operaatioihin.

```
>> a=diag([1,2,3])
>> [n,m]=find(a>1)
>> for i=1:length(n);a(n(i),m(i))=6;end;a
```

```
>> A=[0 1 4 0 7]
>> A(find(A>2))
```

Tehtävä: Kirjoita matriisi $C = \begin{bmatrix} 1 & 0 & 3 & 4 & 5 & 4 & -3 \\ 2 & 1 & 0 & 1 & -2 & 2 & 1 \end{bmatrix}^T$ ja muodosta komento, joka etsii sen rivin indeksin, jonka alkiot ovat 4 ja 1.

3.3 Funktioiden nollakohdat ja ääriarvot

Komennolla **fminbnd** voidaan etsiä (lokaali) minimikohta yhden muuttujan funktiosta, tietyllä määritellyllä alueella. Useamman muuttujan funktion tapauksessa ääriarvojen löytämiseen käytetään komentoa **fminsearch** (tämän komennon käyttö käsitellään luentomonisteen luvussa 6). Komento **fzero** etsii annettua lähtöarvoa lähinnä olevan funktion nollakohdan.

```
>> fplot('x^2-x-1',[-1 2])
>> grid on
>> fminbnd('x^2-x-1',-1,2)
>> fzero('x^2-x-1',0)
>> fzero('x^2-x-1',1)
```

Tehtävä: Syötä Matlab-komentoikkunassa seuraavat skalaarit

$a = 5$, $b = \sqrt{2}$, $c = 1 + 2i$, $d = 10^{10}$, $e = \sin 40°$, $f = \log_{10} 3$ ja $g = \log_3 3$.

Tehtävä: Etsi funktion $y = -x^2 + \pi x - 1$ maksimikohta välillä $x \in [0,2]$.

Tehtävä: Muodosta 100×100 kokoinen satunnaismatriisi A komennolla

```
>> A=10*rand(1,1)*rand(100,100);
```

ja määritä arvoltaan suurinta alkio(i)ta vastaavat rivi- ja sarakeindeksit. Käytä ratkaisuun vain yhtä komentoa (komento voi muodostua useista sisäkkäisistä Matlab-komennoista). Laske myös montako prosenttia matriisin alkioista on arvoltaan suurempia kuin 0.2.

Tehtävä: Ratkaise suoran $y = x - 1$ ja funktion $y = \cos(x)$ leikkauspisteen x- ja y-koordinaatit välillä $x \in [0, \pi]$. Aloita tehtävän ratkaisu piirtämällä kuva. Arvion (ei ole tehtävän ratkaisu) leikkauspisteen koordinaateille saat kuvaajasta käyttäen komentoa
`>> [u,v]=ginput`.

4. Interpolointi ja polynomin sovittaminen mittausaineistoon

4.1 Interpolointi

Matlabissa vektorin tai matriisin yksiulotteiseen interpolointiin käytetään komentoa **interp1**. Syötteeksi komennolle annetaan alkuperäisten datapisteiden x- ja y-koordinaatit vektoreilla X ja Y, sekä vektori XI, joka sisältää uusien interpoloitavien pisteiden x-koordinaatit. Komento antaa tulokseksi uusia x-koordinaatteja vastaavat y-koordinaatit sisältävän vektorin YI. Vakioasetuksena on lineaarinen interpolointi ‘linear’. Muut interpolointitavat voidaan asettaa lisäparametrilla, joka voi olla jokin seuraavista: nearest, linear, spline, pchip ja cubic.

```
>> x=[-3:3]
>> y=6*x.^3-9*x.^2-20*x+1
>> plot(x,y,'k')
>> hold on
>> plot(x,y,'o')
>> x0=-1.5
>> y0=interp1(x,y,x0)
>> plot(x0,y0,'ro')
```

```
>> y1=interp1(x,y,x0,'nearest')
>> plot(x0,y1,'mo')
>> y2=interp1(x,y,x0,'spline')
>> plot(x0,y2,'go')
>> xx=[-3:.01:3];
>> yy=interp1(x,y,xx,'spline');
>> plot(xx,yy,'g')
>> x=xx;y=6*x.^3-9*x.^2-20*x+1;
>> plot(x,y,'k')
```

4.2 Polynomit

Matlabissa polynomia voidaan käsitellä vektorina, jossa on lueteltu kertoimet korkeimmasta muuttujan potenssista alkaen. Esim. p = [1 2 -3] tarkoittaa polynomia $x^2 + 2x - 3$.

Komento **roots** ratkaisee polynomin juuret (vastaavan funktion nollakohdat).

```
>> roots([1 2 -3])
```

Komento **poly** muodostaa polynomin annettujen juurien perusteella ja komento **polyval** puolestaan laskee polynomin arvon annetulla muuttujan arvolla. Komennot roots ja poly ovat toisilleen käänteisiä toimenpiteitä.

```
>> poly([-3 1])
>> polyval([1 2 -3],5)
>> polyval([1 2 -3],1)
```

Tehtävä: Ratkaise polynomin $x^5 - 8x^2 + 3x - 2$ juuret ja polynomin arvo, muuttujan arvolla x = 3. Montako nollakohtaa on funktiolla $y(x) = x^5 - 8x^2 + 3x - 2$?

Tehtävä: Määrää yhtälön $y(x) = Ax^2 + Bx + C$ kertoimet A, B ja C niin, että yhtälöllä on nollakohdat x = -2 ja x = 3 ja että funktio kulkee pisteen (x,y)=(1,-2) kautta. Hyödynnä ratkaisussa edellä esitettyjä Matlab-komentoja.

Tehtävä: Mikä muuttujan x arvo vastaa funktion $y(x) = 2x^3 - 2x + 7$ arvoa y = 8? Ratkaise tehtävä kolmella eri tavalla käyttäen ratkaisuissa komentoja: 1° fplot + ginput 2° polyval + interp1 3° roots. Mikä käytetyistä ratkaisumenetelmistä on tarkin? Tarkista ratkaisut käyttäen polyval komentoa.

4.3 Polynomin sovittaminen pistejoukkoon

Komento **polyfit** sovittaa polynomin annettuihin datapisteisiin. Syötteenä komennolle annetaan datapisteiden x- ja y-koordinaatit kahden vektorin, X ja Y, avulla. Syötteeksi annetaan myös sovitettavan polynomin asteluku.

```
>> x=[1:5]
>> y=[5.5  43.1 128  290.7  498.4]
>> plot(x,y,'ko')
>> grid on
>> hold on
>> p1=polyfit(x,y,1)
>> x2=[1:.01:5];
>> y1=polyval(p1,x2);
>> plot(x2,y1,'r')
>> p2=polyfit(x,y,2)
>> p3=polyfit(x,y,3)
>> y2=polyval(p2,x2);
>> y3=polyval(p3,x2);
>> plot(x2,y2,'g',x2,y3,'b')
```

Komento **corrcoef** laskee kahden vektorin välisen korrelaatiokertoimen. Lasketaan korrelaatiokertoimet edellä lasketuille polynomisovituksille:

```
>> corrcoef(y,polyval(p1,x))
>> corrcoef(y,polyval(p2,x))
>> corrcoef(y,polyval(p3,x))
```

Tehtävä: Tutki mikä pitää olla sovitettavan polynomin asteluku, jotta sovituksen korrelaatiokerroin datapisteisiin olisi parempi kuin 0.99. Datapisteiden x- ja y-koordinaatit ovat

x = [100 150 200 250 300 350 400 450 500 550]
y = [0.3183 0.6685 1.2096 1.4961 1.2732 0.6366 0.9549 1.5915 1.9099 1.7507]

Onko pistejoukolle laskettu hyvän korrelaatiokertoimen (> 0.99) omaava sovitus fysikaalisen ilmiön mallintamisen kannalta mielekäs?

Polynomin (ja myös muiden funktioiden) PNS-sovitus voidaan laskea myös matriisijakolasku operaation \ avulla. Esimerkiksi jos tehtävänä on sovittaa funktio $y = a_2 x^2 + a_1 x + a_0$ pistejoukkoon, jonka koordinaatit ovat

x = [0 0.3 0.8 1.1 1.6 2.3]';
y = [0.5 0.82 1.14 1.25 1.35 1.4]';

Ongelma voidaan esittää muodossa:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \\ x_5^2 & x_5 & 1 \\ x_6^2 & x_6 & 1 \end{bmatrix} \times \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

```
>> x=[0 .3 .8 1.1 1.6 2.3]'
>> y=[.5 .82 1.14 1.25 1.35 1.4]'
>> X=[x.^2 x ones(size(x))]
>> a=X\y
>> plot(x,y,'ko')
>> hold on;grid on
>> xx=[0:.01:2.3];
>> plot(xx,a(1)*xx.^2+a(2)*xx+a(3))
```

Tehtävä: Sovita eksponenttifunktio $y = a_2 x e^{-x} + a_1 x^{-x} + a_0$ edellisen esimerkin piste-joukkoon.

5. Graafiset kuvaajat ja niihin liittyvät komennot

5.1 Funktioiden kuvaajien piirtäminen

Aikaisemmin luentomonisteessa on jo tarvittu funktioiden piirtämiseen käytettyjä komentoja **fplot** ja **plot**. Myös komennot figure ja legend ovat jo ennestään tuttuja. Komento **figure** aukaisee uuden kuvaikkunan ja komento **legend** lisää selitteen piirretyille kuvaajille. Kuvalle voidaan antaa otsikko komennolla **title** ja akselit voidaan nimetä komennoilla **xlabel** ja **ylabel**. Akseleiden aluetta ja skaalausta voidaan säätää komennolla **axis**. Komento **grid on** lisää kuvaan apuviivaston.

```
#1
>> figure
>> x=pi*[0:.01:2]';
>> plot(x,sin(x),x,cos(x))
>> title('Trigonometriset funktiot')
>> ylabel('y')
>> xlabel('x')
>> legend('sin(x)','cos(x)')
>> grid on
```

```
#2
>> axis([0 2*pi -1.2 1.2])
>> axis equal
>> axis square
>> axis off
>> axis on
>> axis normal
```

```
#3
>> figure;
>> plot(x,sin(x),'g.')
>> plot(x,sin(x),'ro')
>> plot(x,sin(x),'md')
>> plot(x,sin(x),'b--','LineWidth',2.5)
>> plot(x,sin(x),'k-.')
```

Havaintopisteet ja niiden virherajat voidaan piirtää komennolla **errorbar**.

```
>> x=[1:10]
>> y=10*sqrt(x)+5*rand(1,10)
>> U=3*rand(1,10)
>> L=2*rand(1,10)
>> errorbar(x,y,L,U,'o')
```

Listaus piirretyn kuvan ominaisuuksista saadaan komennoilla **get(gca)** ja **get(gcf)**. Kuvan useimpia ominaisuuksia voidaan muuttaa komennoilla **set(gca)** ja **set(gcf).**

```
>> get(gca)
>> get(gcf)
>> set(gca,'FontSize',18)
```

Kuvaikkunaan voidaan sijoittaa tekstiä komennoilla **text** ja **gtext**. Ensimmäisen komennon tapauksessa tekstin sijainti määritetään koordinaattiarvojen avulla ja jälkimmäisessä tapauksessa teksti sijoitetaan kuvaan hiiren avulla.

```
>> text(3,0,'\phi','FontSize',25)
>> gtext('\omega','FontSize',25)
```

Kuvaikkuna voidaan jakaa osiin komennon **subplot** avulla.

```
>> x=pi*[0:.01:2];
>> figure;
>> subplot(2,3,1);plot(x,x)
>> subplot(2,3,2);plot(x,x.^2)
>> subplot(2,3,3);plot(x,sin(x))
>> subplot(2,3,4);plot(x,cos(x))
```

Muita piirtokomentoja ovat **semilogx**, **semilogy** ja **loglog**, jotka skaalaavat akselit logaritmisesti.

```
>> x=[0:.1:100]';
>> figure;plot(x,x.^2)
>> figure;semilogy(x,x.^2)
```

Kuvaajan piirtämiseen voidaan käyttää tarvittaessa myös napakoordinaatisto esitystä, komento **polar**. Komennolle annetaan syötteeksi kiertokulma ja säde.

```
>> x=pi*[0:.01:2];
>> polar(x,sin(6*x))
```

Tiettyjä vektorin arvoja vastaavan pylväsdiagrammin voi piirtää komennolla **bar** tai **barh**.

```
>> a=[8 -2 3 4 5]
>> bar(a)
>> barh(a)
```

Jos vektori sisältää useita arvoja tai jos halutaan kuvata arvojen jakaumaa, on pylväsdiagrammi esityksen sijasta havainnollisempaa käyttää histogrammia. Histogrammi saadaan aikaan komennolla **hist**. Histogrammissa vektorin alkioiden arvot ovat vaaka-akselin muuttujana ja pystyakseli esittää kyseisen arvon omaavien alkioiden lukumäärää.

```
>> a=6*rand(1000,1)-3;
>> b=randn(1000,1);
>> figure;bar(a)
>> figure;bar(b)
>> figure;hist(a)
>> figure;hist(b)
>> figure;hist(b,20)
```

5.2 3D-kuvaajat

Komentoa **plot3** voidaan käyttää kolmiulotteisten pistejoukkojen tai –käyrien piirtämiseen normaalin kaksiulotteisen **plot** komennon tapaan; x- ja y-koordinaattien lisäksi komennolle annetaan syötteeksi z-koordinaatit.

```
>> x=pi*[0:.01:2];
>> plot3(x,sin(x),cos(x))
>> grid on
>> axis equal
```

Komennot **legend, xlabel**, **ylabel** ja **axis** toimivat kolmiulotteisilla kuvaajilla samalla tavalla kuin kaksiulotteisten kuvaajien tapauksessa.

Jos halutaan kuvata kolmiulotteisessa koordinaatistossa jokin muotoa z = z(x,y) oleva pinta, käytetään komentoja **surf**, **surfc**, **surfl** ja **mesh**. Pinnan piirtämistä varten luodaan tasavälinen hila (matriisi) komennolla **meshgrid**.

```
>> [x,y]=meshgrid([0:.5:5],[0:5])
>> z=x+y
>> mesh(x,y,z)
>> surf(x,y,z)
```

```
>> [x,y]=meshgrid(pi*[0:.05:2]);
>> z=.5*x+sin(x).*cos(y);
>> surfc(x,y,z)
```

Haluttu tarkastelu- eli kamerakulma voidaan asettaa komennolla **view** tai voidaan säätää hiirellä graafisen käyttöliittymän kautta.

```
>> view(20,20)
```

Komentojen surf, surfc ja mesh tapauksessa käytetään värikarttaa (colormap), jossa piirtoväri indikoi funktion saaman arvon suuruuden. Funktion arvojen ja värin välinen yhteys saadaan näkyville komennolla **colorbar** ja käytettävää värikarttaa voidaan vaihtaa komennolla **colormap**. Kokeile edellisen kuvan tapauksessa komentoja

```
>> colorbar
>> colormap(summer)
>> colormap(hot)
>> colormap(hot)
>> colormap(cool)
>> c=linspace(0,1,64)'*ones(1,3)
>> c(end-4:end,:)=[ones(5,1) zeros(5,1) zeros(5,1)]
>> colormap(c)
>> colormap('default')
```

Tehtävä: Tee oma värikartta ja käytä sitä edellisen esimerkin kuvaajaan. Värikartassa on 64 väriä, jotka ovat määritelty RGB-arvojen avulla (kolme lukuarvoa, punaista, vihreää ja sinistä värikanavaa varten, kukin väliltä 0 - 1).

Tehtävä:

```
>> load topo
>> [x y z]=sphere(50);
>> s=surface(x,y,z,'facecolor','texture','cdata',topo);
>> set(s,'edgecolor','none');
>> axis off vis3d
>> colormap(topomap1)
```

6. Omien Matlab-funktioiden muodostaminen

6.1 Matlab-ohjelmointi

Oma Matlab-funktio/komento ohjelmoidaan Matlabin tekstieditori-ikkunassa, joka avataan valitsemalla valikosta **File** → **New** → **M-File**. Ohjelma talletetaan .m-päätteisenä tekstitiedostona ja sitä kutsutaan Matlabin komentoikkunasta ohjelmatiedoston tallennusnimellä. Monet Matlabin valmiista komennoista ovat .m-tiedostoja.

Matlab-ohjelma voi olla pelkkä joukko peräkkäisiä valmiita Matlab-komentoja (skripti), jolloin ohjelma käyttää Matlabin työtilan muuttujia ja tallettaa myös kaikki ohjelman tarvitsemat muuttujat työtilaan. Ohjelman voi myös määritellä funktioksi, jolloin ohjelman ensimmäinen rivi alkaa avainsanalla **function**. Sanan "function" jälkeen tulee funktion paluuarvo; tämän jälkeen funktion nimi sekä suluissa syötteenä annettavat argumentit. Jos ohjelmaan halutaan lisätä jokin kommentti tai selite, käytetään kyseisen rivin alussa %-merkkiä.

Esimerkki Matlab-funktiosta:

Tekstieditori:

```
function output=summa(input1,input2)
% Tämä ohjelma laskee kahden skalaarin summan
output=input1+input2;
```

Komentoikkuna:

```
>> tulos=summa(1,3)
```

Kun ohjelma määritellään funktioksi, sen tarvitsemien parametrien arvot annetaan ohjelmalle syötteinä, eivätkä tehdyt laskutoimitukset varaa muuttujia työtilasta, vaan ainoastaan funktion lopputuloksena palauttama arvo tallentuu nimettyyn työtilan muuttujaan. Funktion käyttämät muuttujat ovat siis lokaaleja muuttujia. Ohjelmassa välivaiheina tehtävien komentojen tai laskutoimitusten tulokset näkyvät kuitenkin työtilassa, ellei kyseisiä komentoja ole varustettu puolipisteellä. Tarvittaessa tietyt funktion muuttujat voidaan määritellä komennolla **global** globaaleiksi, jolloin ohjelma käyttää niille työtilassa määriteltyjä arvoja. **Huomautus!** jos olemassa olevaa ohjelmaa muutetaan, on ohjelma ensin talletettava, jotta tehdyt muutokset tulisivat voimaan.

Esimerkki:

Komentoikkuna:

```
>> edit summa
```

Tekstieditori:

```
function output=summa
% Tämä ohjelma laskee kahden skalaarin summan
global input1 input2
output=input1+input2;
```

Komentoikkuna:

```
>> global input1 input2
>> input1=2
>> input2=3
>> tulos=summa
```

Tehtävä: Tee Matlab-funktio, joka laskee matriisin (mukaanlukien erikoistapaukset vektori ja skalaari) kaikkien alkioiden keskiarvon. Älä käytä ohjelmasi toteutuksessa valmista keskiarvokomentoa **mean**.

Komennot **nargin** ja **nargout** antavat tulokseksi funktion input- ja output-argumenttien lukumäärän.

6.2 Kontrollirakenteet

Ohjelmoidessa tarvitaan usein hyödyllisiä kontrollirakenteita **for**, **while** ja **if**. Komentojen avulla voidaan ohjelmaan luoda erilaisia silmukoita. Kaikki nämä komennot vaativat komennon **end** käyttöä silmukan/rutiinin lopussa.

Komento **for** toistaa sisällään olevaa rutiinia halutun monta kertaa.

```
>> a=0;for i=1:2:5;i,a=a+i,end
```

Komento **while** puolestaan toistaa silmukkaa niin kauan kuin annettu ehto on voimassa.

```
>> i=1;while i~=5,i=i+1,end;i
```

Tehtävä: Selvitä while komennon avulla mitkä positiiviset kokonaisluvut yhdestä ylöspäin (1, 2, 3…) on kerrottava keskenään, jotta niiden tulo olisi suurempi kuin kymmenen miljardia. Vinkki: Komento **prod** laskee vektorin alkioiden tulon.

If, **elseif** ja **else** komennoilla voidaan asettaa tietty ehto, nuodattaen seuraavanlaista rakennetta:

IF ehto, seuraus
ELSEIF ehto, seuraus
ELSE, seuraus
END

Esimerkiksi

Tekstieditori:

```
function out=testi(in)

if in>=0&in<=2,
    out=1;
elseif in>2,
    out=2;
else out=0;
end
```

Komentoikkuna:

```
>> testi(-1)
>> testi(0)
>> testi(1)
>> testi(2)
>> testi(3)
```

Tehtävä: Tee Matlab-funktio **lotto**, joka arpoo lottorivin, seitsemän numeroa väliltä 1-39 ja sama numero saa esiintyä lottorivissä vain kerran. Ohjelman on annettava arpomansa seitsemän numeroa suuruusjärjestyksessä pienimmästä suurimpaan.

Tehtävä: Muokkaa edellisessä tehtävässä tekemääsi ohjelmaa niin, että ohjelmalle voidaan antaa syötteenä valmiiksi 1-7 numeroa, jotka tulevat osaksi lottoriviä.

6.3 Optimointi fminsearch komennon avulla

Aikaisemmin kurssilla tarkasteltiin optimointitehtävää, jossa ratkaistavana oli yhden muuttujan funktion (lokaali-) ääriarvo tai nollakohta. Komento **fminsearch** soveltuu useamman muutujan funktioihin. Yksinkertaiset optimointitehtävät voidaan ratkaista suoraan komentoikkunassa, mutta monimutkaisemmissa tapauksissa on usein kätevämpää kirjoittaa lyhyt Matlab ohjelma.

Komennolle fminsearch annetaan syötteeksi optimoitavan funktion nimi, sekä alkuarvaus. Lisäksi voidaan asettaa useita komennon toimintaan ja optimointiprosessin tarkkuuteen liittyviä lisäparametreja, kuten Display, TolX, TolFun, MaxFunEvals, and MaxIter. Näiden selitykset löytyvät fminsearch komennon dokumentoinnista » help fminsearch.

Esimerkki: Olkoon tehtävänä ratkaista funktion $f(x,y) = x^2 + y^2$ minimikohta numeerisesti.

Komennon **inline** avulla voidaan määritellä funktio työtilassa. Esimerkiksi mainittu funktio voitaisiin muodostaa komennolla

```
» f=inline('x.^2+y.^2','x','y')
» f(2,1)
```

mutta, jotta funktioon voitaisiin soveltaa fminsearch komentoa, on muuttujat määriteltävä hieman toisin:

```
» f=inline('x(1).^2+x(2).^2','x')
» f([2 1])

» x=fminsearch(f,[1 1])
» f(x)

» x=fminsearch(f,[1 1],optimset('TolX',1e-14))
» f(x)
```

Fminsearch komennon avulla voidaan optimoida myös Matlabin omia valmiita komentoja.

Esimerkki:

```
» g=inline('norm(x)','x')
» x=fminsearch(g,[1 1 1])
```

Fminsearch komennon avulla voidaan myös sovittaa haluttu funktio pistejoukkoon.

Esimerkki: Sovita funktio $y(x) = Ax^2 + Bx + C$ (määrää kertoimet A, B ja C) pistejoukkoon

x = [1 2 3 4 5 6 7 8 9 10]

y = [0.0936 0.1693 1.6477 2.4561 3.5760 6.4882 8.4230 10.8195 15.0956 18.4066]

Ratkaisu:

```
» ABC1=polyfit(x,y,2)
```

tai käytetään fminsearch komentoa ja kirjoitetaan M-tiedostot sovitus.m ja virhe.m.

Tekstieditori:

```
function ABC=sovitus(x,y,arvaus)
ABC=fminsearch('virhe',arvaus,optimset('display','off'),x,y);
```

Tekstieditori:

```
function out=virhe(X,x,y)
y2=X(1)*x.^2+X(2)*x+X(3);
out=sqrt(sum((y-y2).^2));
```

Komentoikkuna:

```
» ABC2=sovitus(x,y,[1 1 1])
```

Tehtävä: Sovita funktio $y(x) = Ae^{Bx} + Cx$ pistejoukkoon

x = [1 2 3 4 5 6 7 8 9 10]

y = [0.3443 0.4984 0.6644 0.8451 1.0437 1.2640 1.5110 1.7906 2.1099 2.4778]

ja piirrä kuva lasketusta sovituksesta.

Saitko ratkaisuksi A = 0.2, B = 0.2 ja C = 0.1? Jos et, niin mikä mahtaisi olla syynä tähän?

7. Signaalinkäsittelyn alkeita

Tässä luvussa tarkastelemme esimerkkien ja harjoitustehtävien avulla muutamia fyysikoiden usein tarvitsemia signaalien eli mittausdatan käsittelyyn liittyviä toimenpiteitä.

#1: ERI SIGNAALIMUOTOJEN KONSTRUOIMINEN

Yleensä analysoitava signaali saadaan suoraan mittalaitteesta, jonkin detektorin tai anturin vasteena, mutta Matlabissa on myös komentoja erilaisten signaalimuotojen luomiseksi lähinnä signaalien teoreettista tarkastelua varten. Esimerkkeinä komennot **sawtooth** ja **square**, joissa arvot vaihtelevat välillä (-1) – 1 ja joissa yhden periodin pituus on 2π.

Esimerkki: Generoidaan kahden sekunnin mittainen kanttiaalto, jonka taajuus on 80Hz ja joka on mitattu 10kHz näytteistystaajuudella. Piirretään signaalista puolen sekunnin mittainen jakso.

```
>> fs=10000       % fs on näytteistystaajuus
>> t=[0:1/fs:2];  % t on aikavektori
>> y=square(2*pi*80*t);
>> plot(t,y);axis([0 .5 -1.1 1.1])
```

Tehtävä: Generoi sinifunktio, jonka taajuus on 2kHz ja jonka vaihe ajanhetkellä t = 0 on $\pi/4$. Näytteistystaajuus on 50kHz ja signaalin kesto/pituus 0.5 sekuntia. Piirrä signaalin lopusta 10ms pituinen pätkä.

KOHINASIGNAALI

Erilaisia kohinamalleja on olemassa monenlaisia ja tässä perehdytään näistä yleisimpään, normaalijakautuneeseen kohinaan, jota syntyy tyypillisesti elektronisissa mittalaitteissa ja järjestelmissä erilaisista ulkoisista ja sisäisistä lähteistä johtuen.

Kohinainen signaali g voi muodostua joko additiivisen kohinan

$$g(t) = f(t) + n(t)$$

tai kerrannaisen kohinan

$$g(t) = f(t) \times n(t)$$

vaikutuksesta, missä f on alkuperäinen signaali ja n tietyn kohinamallin mukainen kohinatermi.

Esimerkki A: Tarkastellaan signaalia, joka koostuu kahdesta sinimuotoisesta signaalista, joiden taajuudet ovat 50Hz ja 120Hz ja joista jälkimmäisen osuuden amplitudi on kaksinkertainen ensimmäisen osuuden amplitudiin verrattuna. Näytteistystaajuus on 1kHz ja signaalin kesto yksi sekunti.

```
>> fs=1000;
>> t=(0:1/fs:1)';
>> y=sin(2*pi*50*t)+2*sin(2*pi*120*t);
```

Lisäksi signaali sisältää normaalijakautunutta additiivista kohinaa.

```
>> n=0.5*randn(size(t));
>> yn=y+n;
>> plot(t(1:100),yn(1:100))
```

#2: SIGNAALIKOHINASUHDE

Mitattaessa reaalimaailman ilmiöitä, mittaussignaaliin summautuu satunnainen kohina-osuus tai kohinasignaali. Tästä syystä eri ajanhetkinä tehdyt mittaukset poikkeavat hieman toisistaan, vaikka itse mittattava ilmiö pysyisikin muuttumattomana.

Signaalikohinasuhde (Signal-to-Noise Ratio, SNR) kuvaa signaalin vahvuutta suhteessa signaalissa esiintyvään taustakohinaan ja se määritellään mittalaitteen vastaanottaman signaalin keskimääräisen tehon ja mittalaitteen vastaanottaman kohinan keskimääräisen tehon välisenä suhteena. Signaalikohinasuhde ilmoitetaan yleensä desibeli asteikolla. Jos esimerkiksi SNR = 20 dB, on kohinan teho sadasosa signaalin tehosta.

$$SNR = 10\log_{10}\left(\frac{P_{signaali}}{P_{kohina}}\right)$$

Tehtävä: Määritä jännitesignaalin yn signaalikohinasuhde edellisen esimerkin tapauksessa. Oletetaan, että jännitesignaalit y ja n on pystytty mittaamaan erikseen saman impedanssin yli. Huomaa, että jännitesignaalin sähköteho on verrannollinen amplitudin neliöön.

Mikäli signaalin alkuperäinen muoto ei ole tiedossa, voi signaali- ja kohinaosan erottaminen toisistaan olla käytännössä vaikeaa. Tällöin tehdään yleensä suuri joukko mittauksia samasta muuttumattomasta mittauskohteesta ja oikeaa signaalimuotoa arvioidaan kaikkien mittausten keskiarvon μ avulla ja kohina lasketaan keskihajonnan σ perusteella,

$$SNR = 10\log_{10}\left(\frac{\mu}{\sigma}\right).$$

Matlabissa keskihajonta (standard deviation) lasketaan komennon **std** avulla.

Tehtävä: Spektrometrin signaalikohinasuhteen määrittämiseksi on tehty 100 toisto-mittausta (matriisin X 100 saraketta). Kukin mitattu spektri kuvaa (muuttumattoman) standardivalonlähteen intensiteettiä aallonpituusalueella 400-700nm, 5nm välein mitattuna. Esitä signaalikohinasuhde aallonpituuden funktiona. Huom! Tässä tapauksessa signaalin amplitudi on suoraan verrannollinen mitattuun valotehoon.

```
>> for n=1:100;x(:,n)=(.5*sin(linspace(2,4.5,61))'.^6+.2).*randn(61,1)+20;end
```

#3: MEDIAANISUODATUS

Mittaussignaalista voidaan poistaa kohinaa esimerkiksi mediaanisuodatuksen avulla. Erilaisia suodatustapoja on useita, mutta tässä tyydymme tarkastelemaan vain mediaanisuodatusta sen yksinkertaisen periaatteen ja helpokäyttöisyyden vuoksi.

Mediaanin laskeminen tietylle lukujoukolle tarkoittaa sitä, että lukuarvot järjestetään suuruusjärjestykseen ja näistä valitaan sitten keskimmäinen. Jos lukujen määrä on parillinen, mediaaniksi lasketaan kahden keskimmäisen luvun keskiarvo.

```
>> a=[7 0 2 2 3]
>> median(a)
>> mean(a)
>> a=[7 0 2 3]
>> median(a)
```

Mediaanisuodatuksessa tarkastellaan vuorollaan yhtä signaalivektorin alkiota ja tiettyä määrää tämän naapurialkioita tarkasteltavan alkion molemmin puolin (esimerkiksi kolme lähintä alkiota tarkasteltavan pisteen molemmin puolin). Nämä alkiot muodostavat ns. ikkunan. Sitten näille lukuarvoille lasketaan mediaani ja tarkasteltavan alkion arvo korvataan mediaanilla. Kaikki signaalivektorin alkiot käydään tällä tavalla järjestyksessä läpi. Mediaanisuodatuksen haittanapuolena on signaalin alku- ja loppupäässä olevien arvojen mahdollinen vääristyminen. Matlabissa yksiulotteinen mediaanisuodatus toteutetaan komennon **medfilt1** avulla. Vaikka mediaanisuodatus ei ole varsinainen taajuussuodin, voidaan ikkunan koolla vaikuttaa siihen minkä kokoiset/taajuiset yksityiskohdat suodattuvat pois.

Tehtävä: Käytä mediaanisuodatusta signaaleihin M1 ja M2.

```
>> M1=sin([0:.01:1]'*pi).*(rand(101,1)+.5);
>> M2=sin([0:.01:2]'*pi).*(rand(201,1)+.5);
```

Suodatuksen jälkeen signaalien tulisi mahdollisimman tarkasti olla muotoa

```
>> S1=sin([0:.01:1]'*pi)
>> S2=sin([0:.01:2]'*pi)
```

Piirrä kuva, jossa näkyvät suodatetut signaalit ja alkuperäiset, kohinattomat signaalit S1 ja S2.

#4: TAAJUUSSPEKTRIN LASKEMINEN

Fourier-muunnosta käytetään tyypillisesti signaalinkäsittelyssä erilaisiin taajuusanalyysia vaativiin tehtäviin. Fourier-muunnos liittyy läheisesti Fourier-sarjoihin ja näiden perusajatukseen funktion esittämisestä sinimuotoisten funktioiden summana. Funktion f Fourier-muunnoksen määrittelee integraali

$$g(\omega) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i \omega t} dt \,,$$

missä ω on kulmataajuus, ja käänteisen Fourier-muunnoksen integraali

$$f(t) = \int_{-\infty}^{\infty} g(\omega) e^{2\pi i \omega t} d\omega .$$

Diskreetin funktion (esim. mittaussignaali) tapauksessa käytetään diskreettiä Fourier-muunnosta (Discrete Fourier Transformaion, DFT), jossa integraali korvautuu summalausekkeella.

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi i n k / N}, n = 0, \ldots, N-1$$

Matlabissa käytetään erityistä nopeaa Fourier-muunnosta (Fast Fourier Transformation, FFT), joka on algoritmi DFT:n laskemiseksi nopeasti ja tehokkaasti.

Seuraavaksi tarkastelemme taajuusspektrin laskemista FFT:n avulla esimerkki A:n tapauksessa.

```
>> fs=1000
>> t=(0:1/fs:1)';
>> y=sin(2*pi*50*t)+2*sin(2*pi*120*t);
>> yn=y+0.5*randn(size(t));
```

Näytteenottotaajuuden fs on oltava riittävän suuri, sillä suurin mitatusta näytteestä havaittavissa oleva taajuus on niin kutsuttu Nyquistin taajuus fn. Kaikki tätä suuremmat taajuudet laskostuvat matalampien taajuuksien päälle.

```
>> fn=.5*fs; % fn on nk. Nyquistin taajuus
>> Y=fft(yn);
>> Y(1)=[]; % ensimmäinen arvo voidaan poistaa tarpeettomana
>> n=length(Y)
>> power=abs(Y(1:n/2)).^2/n; % lasketaan taajuusspektri
>> ff=(1:n/2)/(n/2)*fn; % muuttuja ff määrää taajuusakselin skaalan
>> plot(ff,power)
>> xlabel('Frequency (Hz)')
>> ylabel('Signal Power')
```

Tehtävä: Tarkastellaan mittaussignaalia, joka koostuu Doppler-pursketaajuudesta, signaalikohinasta ja matalataajuisesta verhokäyrästä. Piirrä kuva mittaussignaalista. Poista signaalista matalataajuinen osuus ja piirrä uusi kuva. Määritä tämän jälkeen signaalin Doppler-pursketaajuus.

```
>> t=[0:500e-9:4e-4]';
>> n=.02*abs(randn(1,10))'+1;
>> e=interp1(linspace(0,800,10)',n,[0:800]','spline')+.5;
>> g=.99*exp(-(1.6e4*t-3).^2)+.01;
>> e2=[0 0 0 -.1 -.25 -.5 -.8 -1 -.9 -.7 -.4 -.1 .1 .15 .09 .05 0
0];
>> d=interp1([1:18]',e2',linspace(1,18,801)','cubic');
>> n=.04*randn(801,1);
>> data=100*(cos(4e5*e.*t).*g+d+n);
```

#5: INTERFERENSSIKUVION NÄKYVYYDEN MÄÄRITTÄMINEN

Interferenssikuvioiden ja tyypillisesti myös muiden erilaisten valon intensiteettijakaumien mittaamiseen käytetään CCD-kameraa, jossa kunkin kuvapikselin vaste on suoraan verrannollinen mitatun valon irradianssiin (yksikkö W/m^2). Yleensä mittaamiseen käytetään värittömän harmaasävykuvan tuottamaa kameraa, jonka arvot vaihtelevat tyypillisesti välillä 0-255 (8-bittinen vaste) tai 0-4095 (12 bittinen vaste).

Matlabissa kuvaa käsitellään matriisina, jonka kukin alkio vastaa yhtä kuvapikseliä. Värikuva on puolestaan kolmiulotteinen matriisi, jonka kolme eri kerrosta määrittävät punaisen, vihreän ja sinisen primäärivärin määrät kuvassa. Kuva voidaan esittää Matlabissa **image**, **imagesc** ja **imshow** komentojen avulla. Komento **imagesc** skaalaa kuvan arvot automaattisesti vastaamaan koko harmaasävyasteikkoa, jolloin kuvan yksityiskohdat ovat parhaiten havaittavissa.

```
>> n=20*rand(480,640);
>> image(n);colormap gray
>> imagesc(n);colormap gray
>> imshow(n)
```

Interferenssikuvion näkyvyys (visibility) määritellään kaavalla

$$V = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}},$$

missä $I_{\max}$ ja $I_{\min}$ ovat interferenssikuvion peräkkäisten maksimi- ja minimikohtien irradianssit. Näkyvyyden arvo vaihtelee välillä 0-1.

Tehtävä: Määritä keskimääräinen interferenssikuvion näkyvyys CCD-kameralla kuvatusta interferenssikuviosta.

```
>> kuva=ones(480,1)*(100*sin(linspace(0,100,640))+128)+20*randn(480,640);
>> kuva(find(kuva<0))=0;
>> kuva(find(kuva>255))=255;
```

8. Kertaustehtäviä

1. Muodosta matriisi $A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$

   ja poimi

   a. ensimmäisen rivin neljäs alkio
   b. viimeisen sarakkeen kaksi viimeistä alkiota
   c. kolmas rivi kokonaisuudessaan
   d. kaksi ensimmäistä saraketta
   e. matriisi $\begin{bmatrix} 8 & 9 \\ 13 & 14 \end{bmatrix}$.

2. Muodosta matriisi $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 2 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 2 & 2 \end{bmatrix}$.

   Älä kirjoita matriisia riveittäin, vaan muodosta se osissa erilaisten komentojen avulla.

3. Laske edellisen tehtävän matriisin rivien summat ja keskiarvot.

4. Yhdistä kärkipisteet (0,-1), (3,4), (2,0) ja (5,-2) suorilla viivoilla toisiinsa.

5. Piirrä kolme sisäkkäistä ympyrää, joiden säteet ovat 1, 2 ja 3.

6. Siementen paikat auringonkukan kukinnossa nuodattavat seuraavanlaista napa-koordinaattiesitystä:

   $r_n = \sqrt{n} \qquad \theta_n = \frac{137.51}{180}\pi n$, missä n on kunkin siemenen numero.

   Merkkaa 1000 ensimmäisen siemenen paikka ympyrällä xy-koordinaatistoon.

7. Piirrä funktiot sin(x), cos(x), tan(x) ja cot(x) samaan kuvaan käyttäen eri värejä ja viivatyylejä. Rajaa x-akseli välille [-π, π] ja y-akseli välille [-1.5, 1.5]. Lisää kuvaan selite, joka nimeää piirretyt kuvaajat. Lisää myös otsikko ja nimeä akselit.

8. Ratkaise yhtälöryhmä

$$\begin{cases} x + y + 2z = 9 \\ 2x + 4y - 3z = 1 \\ 3x + 6y - 5z = 0 \end{cases}$$

9. Muodosta seuraavat skalaarit $a = 5$, $b = \tan(40°)$, $c = \sqrt{24}$, $d = \ln(5)$ ja $f = e^{e^{1/3}}$. Talleta muuttuja a tiedostoon data1.mat, muuttuja b tiedostoon data2.mat jne.

10. Tee Matlab-ohjelma, joka lukee edellisessä tehtävässä luodut mat-tiedostot ja sijoittaa tiedostojen sisältämät muuttujat tulosvektorin alkioiksi.

11. Kirjoita Matlab-ohjelma, joka pyöristää ohjelmalle annetun luvun desimaalista 0.6 lähtien ylöspäin.

12. Kirjoita Matlab-ohjelma, jolle annetaan syötteeksi kaksi lukua. Jos lukujen summan itseisarvo on pienempi kuin kaksi (2), ohjelma antaa tulokseksi lukujen tulon. Muutoin tulokseksi annetaan lukujen osamäärä.

9. Harjoitustyöt

Harjoitustyönä tehtävä Matlab-funktio palautetaan sähköpostin liitetiedostona osoitteeseen hannu.laamanen@joensuu.fi

Ohjelmiin on lisättävä riittävä dokumentointi (%-merkeillä), jossa kerrotaan kuinka ohjelma toimii, mitä ovat syötteeksi annettavat parametrit ja millaisia arvoja niille voidaan antaa ja mikä on ohjelman tuottama tulos. Jos kirjoitettu ohjelma on pitkä tai jos siinä on useita välivaiheita, on jokainen ohjelman vaihe syytä selittää erikseen. Lisää ohjelman dokumentointiin myös oma nimesi, opiskelijanumero ja päivämäärä. Mikäli tehtävänantoon liittyy erillisiä kysymyksiä, lisää vastaus sähköpostiviestiisi.

1. Harjoitustyö

Kirjoita funktio, joka muuntaa syötteeksi annetun desimaalijärjestelmän numeron (välillä 1-2500) vastaavaksi roomalaiseksi numeroksi.

Roomalainen luku	Desimaaliluku
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Huom.

Symbolit I, X ja C kirjoitettuna jomman kumman välittömästi suuremman symbolin eteen tulkitaan vähentävästi. Siis IV = 4, IX = 9, XL = 40, XC = 90, CD = 400 ja CM = 900. Siis 19 = XIX, 49 = XLIV, 1999 = MCMXCIX, 1492 = MCDXCII.

2. Harjoitustyö

Kirjoita Matlab-ohjelma, jonka avulla voidaan mallintaa heittoliikettä.

Ohjelmalle annetaan syötteeksi heitettävän kuulan massa, heittokulma ja kulman suuntainen alkunopeus. Ohjelman pitää antaa tulokseksi heittoliikkeen kantaman arvo ja kuulan liike-energia lähtö- tai putoamishetkellä. Ilmanvastusta ei tarvitse huomioida laskuissa. Ohjelman pitää myös piirtää kuvaaja kuulan lentoradasta.

Arvioi ohjelman avulla kuinka paljon energiaa tarvitaan, jotta miesten kuulantyöntö kuula, massa 7,26 kg, saadaan lentämään 20 metriä.

3. Harjoitustyö

Tee Matlab-ohjelma, joka soudattaa ohjelmalle annetuista mittaussignaaleista satunnaiskohinan pois mahdollisimman tehokkaasti. Ohjelman toimivuutta voit testata seuraavilla mittaussignaaleilla:

```
>> M1=sin([0:.01:1]'*pi).*(rand(101,1)+.5);
>> M2=sin([0:.01:2]'*pi).*(rand(201,1)+.5);
```

Suodatuksen jälkeen signaalien tulisi mahdollisimman tarkasti olla muotoa

```
>> M1=sin([0:.01:1]'*pi)
>> M2=sin([0:.01:2]'*pi)
```

Ohjelmaa testataan antamalla syötteeksi jompi kumpi yllä olevista signaaleista, mutta ohjelman tulisi toimia yleisesti.

Lisää ohjelmaan myös syötteeksi annettava parametri, jonka avulla suodatettu mittaussignaali voidaan tarvittaessa invertoida. (parametrin arvo 0 = ei invertointia ja parametrin arvo 1 = invertointi).

4. Harjoitustyö

a) Kirjoita Matlab-ohjelma, joka tutkii onko annettu kokonaisluku parillinen vai pariton. Älä käytä tähän tehtävään tarkoitettuja valmiita Matlab-komentoja. Jos kokonaisluku on pariton, ohjelman pitää antaa tulokseksi nolla (0) ja jos kokonaisluku on parillinen, ohjelman pitää antaa tulokseksi yksi (1).

b) Kirjoita Matlab-ohjelma, jolle annetaan syötteeksi kompleksisia ja/tai reaalilukuja sisältävä rivivektori. Ohjelman on tarkoitus järjestää vektorin alkiot luvun reaaliosan mukaiseen suuruusjäjestykseen suurimmasta pienimpään.

5. Harjoitustyö

Kirjoita Matlab-ohjelma, joka muuttaa Celsius-asteet Fahrenheit-asteiksi ja päivastoin. Se kumpi muunnoksista C° $\rightarrow$ F° vai F° $\rightarrow$ C° on kyseessä, pitää olla valittavissa syötteeksi annettavan lisäparametrin avulla. Parametrille annetaan joko arvo yksi (1) tai nolla (0).

6. Harjoitustyö

Kirjoita Matlab-ohjelma, joka laskee piin likiarvon käyttäen oheista kaavaa

$$\frac{\pi^2 - 8}{16} = \sum_{n=1}^{\infty} \frac{1}{(2n-1)^2 (2n+1)^2}$$

Jos piille laskettu likiarvo pyöristetään kymmenen desimaalin tarkkuuteen, niin mikä n:n pitää vähintään olla, jotta kaikki lasketut kymmenen desimaalia olisivat oikein?

7. Harjoitustyö

Kirjoita Matlab-ohjelma, joka arpoo Lotto-rivin (7 numeroa). Todennäköisyys sille, että jokin numeroista 1-39 tulee valituksi yhdeksi rivin numeroista täytyy olla sama kaikille numeroille. Lisäksi kukin numero voi esiintyä rivissä vain kerran. Numeroiden on oltava ohjelman antamassa Lotto-rivissä suuruusjärjestyksessä. Lisää ohjelmaan syötteeksi vektori argumentti, jolla voidaan antaa 0-7 kpl numeroita, jotka tulevat osaksi lottoriviä.