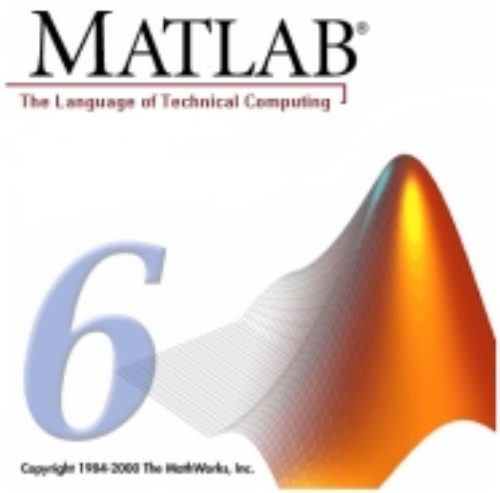
TAMPEREEN AMMATTIKORKEAKOULU
TAMK
TAMPERE POLYTECHNIC

MATLAB 6.0
m-tiedoston ohjelmointiopas

© lähteenmäki.m
2001
www.tpu.fi/~mlahteen/



SISÄLLYSLUETTELO

1 Johdanto

MATLABissa on käytettävissä ohjelmointikieli, jolla voidaan laatia MATLAB-käskyjä sisältäviä tiedostoja. Nämä ns. m-tiedostot voidaan kirjoittaa millä tahansa tavallisella tekstieditorilla ja niille annetaan muotoa `tiedostonimi.m` oleva nimi. m-tiedostoja voidaan sitten ajaa MATLABin komentoikkunasta esimerkiksi antamalla kehotteeseen käsky `tiedostonimi`. Tarkennin `m` on pakollinen ja se tekee tiedostosta MATLABin m-tiedoston. Käytettäessä MATLABia teknilliseen laskentaan ja simulointiin kaikki pysyvään käyttöön tarkoitetut sovellukset ohjelmoidaan m-tiedostoiksi.

m-tiedostoja on kahta tyyppiä. Skriptit sisältävät yksinkertaisesti joukon MATLAB-käskyjä. Funktiot sisältävät myös joukon MATLAB-käskyjä, mutta tämän lisäksi niitä voidaan kutsua annetuilla syöttötiedoilla ja ne voivat palauttaa tulostietoja.

MATLABin mukana tulee kätevä m-tiedosto-editori, jolla nämä tiedostot kannatta kirjoittaa. Editori voidaan käynnistää mm. Windowsin Käynnistä-valikosta tai MATLABin työpöydän menuriviltä valitsemalla File > New > M-File tai Komentoikkunasta kirjoittamalla kehotteeseen käsky `edit tiedostonimi`.

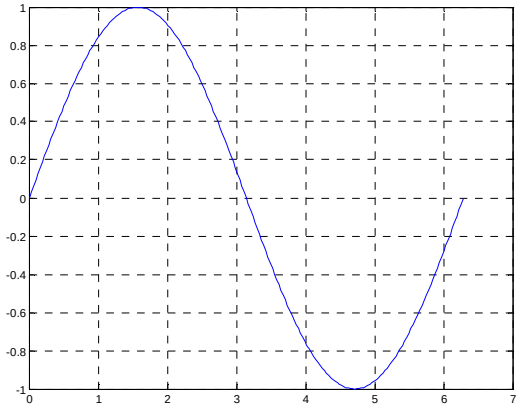
Suurin osa MATLABin ja siihen lisäoptioina hankittujen toolboxien mukana tulevista valmisfunktioista on toteutettu m-tiedostoina, jotka ovat MATLABin asennushakemistossa vapaasti käyttäjän nähtävissä ja editoitavissa. Omat m-tiedostot käyttäjän kannattaa tallentaa asennushakemiston ulkopuolelle konfliktien välttämiseksi tiedostonimissä. MATLABin asennushakemistoon tallennetut omat m-tiedostot häviävät, mikäli MATLAB joudutaan jostain syystä asentamaan uudelleen.

2 Skriptit

Skriptit ovat yksinkertaisia m-tiedostoja, joilla ei ole argumentteja syöttötietojen antamista varten eikä myöskään tietojen palautusta varten. Ne ovat hyödyllisiä automatisoitaessa käskysarjoja, jotka joudutaan usein suorittamaan MATLABin komentoikkunassa. Skriptit voivat operoida avoinna olevan muuttuja-avaruuden muuttujilla, mutta ne voivat myös luoda uusia muuttujia ja operoida niillä. Skriptin luomat muuttujat jäävät muuttuja-avaruuteen sen suorituksen päätyttyä, ja niillä on näin ollen mahdollista operoida edelleen komentoikkunassa.

Seuraavassa on listaus skriptistä `sinif.m`, joka piirtää sinifunktion kuvaajan. Alussa on kaksi %-merkillä alkavaa kommenttiriviä. Kaksi seuraavaa riviä määrittelevät vektorit t ja y, jotka sisältävät 201 argumentin t ja funktion y arvoa. Puolipisteet rivien perässä estävät vektoreiden t ja y tulostumisen komentoikkunaan. Kaksi viimeistä riviä piirtävät käyrän kuvaikkunaan.

```
% Tämä on m-tiedosto, joka piirtää
% sinifunktion kuvaajan jakson matkalta.
t=0:pi/100:2*pi;
y=sin(t);
plot (t,y)
grid on
```



Skripti voidaan ajaa komentoikkunasta kirjoittamalla kehotteeseen käsky `sinif`, jolloin kuvaikkunaan tulostuu viereisen kuvan mukainen käyrä.

3 Funktiot

Funktiot ovat `m`-tiedostoja, joita ajettaessa voidaan antaa syöttötietoina muuttujille arvoja ja jotka palauttavat tarvittaessa arvoja funktion suorituksen päättyessä. Funktiot operoivat oman muuttuja-avaruutensa muuttujilla. Tämä on erillinen komentoikkunassa käytössä olevasta muuttuja-avaruudesta.

Seuraavalla sivulla on listaus funktiosta `rajat.m`, joka laskee ja palauttaa nivelnelikulmion kampikulman ääriarvot, kun syöttötietoina annetaan linkkien pituudet ja runkokulma. Siinä on nähtävissä MATLAB-funktion tyypilliset osat. Ensimmäisenä on funktion esittelyrivi, jonka avainsanasta `function` MATLAB tunnistaa tiedoston funktioksi ja lisäksi määritellään funktion käyttöön liittyvät argumentit.

```
function [alku,loppu]=rajat(r1,r2,r3,r4,theta1)
```

varattu avainsana

palautettavat arvot

funktion nimi

syöttötietoina annettavat arvot

Argumentit, joihin palautettavat arvot sijoitetaan, laitetaan pilkulla erotettuina hakasulkuihin. Jos palautettavien arvojen lukumäärä on yksi tai nolla, sulkujen käyttö ei ole pakollista. Syöttötietoargumentit laitetaan pilkulla erotettuina kaarisulkuihin, mikäli niitä esiintyy, kuten yleensä on asian laita.

Välittömästi funktion määrittelyrivin jälkeen on %-merkillä alkava H1-kommenttirivi:

```
% RAJAT laskee ja palauttaa kampikulman ääriarvot.
```

Tämä on ensimmäinen rivi tekstiä, jonka MATLAB näyttää komentoikkunassa käskyllä `help rajat`. MATLABin etsintäfunktio `lookfor` tutkii ja näyttää funktioiden H1-rivejä, joten tästä rivistä kannattaa tehdä mahdollisimman kuvaava. H1 rivi ei kuitenkaan ole pakollinen.

Välittömästi H1-rivin jälkeen voi sijoittaa esimerkiksi funktion käyttöohjeeksi lisää kommenttirivejä. Funktiossa `rajat.m` näitä on kaksi seuraavaa riviä.

```
% Funktio laskee ja palauttaa nivelnelikulmion kampikulman
% pienimmän ja suurimman arvon. Käyttävä linkki on kampi.
```

MATLAB näyttää komentoikkunassa käskyllä `help rajat` myös nämä rivit kunnes vastaan tulee joko käskyrivi tai tyhjä rivi.

```
function [alku,loppu]=rajat(r1,r2,r3,r4,theta1)
% RAJAT laskee ja palauttaa kampikulman ääriarvot.
% Funktio laskee ja palauttaa nivelnelikulmion kampikulman
% pienimmän ja suurimman arvon. Käyttävä linkki on kampi.

% Muuttujat
%   r1=runkonivelien välimatka
%   r2=kammen pituus
%   r3=kiertokangen pituus
%   r4=keinuvivun pituus
%   theta1=runkokulma
%   alku=kampikulman pienin arvo (asteina)
%   loppu=kampikulman suurin arvo (asteina)

%Laskenta
if (r1+r2)>(r3+r4)&abs(r1-r2)<abs(r3-r4)
     alku=acos((-(r4-r3)^2+r2^2+r1^2)/(2*r1*r2));
     loppu=acos((r2^2-(r4+r3)^2+r1^2)/(2*r1*r2));
end
if (r1+r2)>(r3+r4)&abs(r1-r2)>=abs(r3-r4)
     alku=acos((r2^2-(r4+r3)^2+r1^2)/(2*r1*r2));
     loppu=-alku;
end
if (r1+r2)<=(r3+r4)&abs(r1-r2)>=abs(r3-r4)
     alku=0;
     loppu=2*pi;
end
if (r1+r2)<=(r3+r4)&abs(r1-r2)<abs(r3-r4)
     alku=acos((r2^2-(r4-r3)^2+r1^2)/(2*r1*r2));
     loppu=2*pi-alku;
end
%Palautettavat arvot
alue = loppu-alku;
alku=theta1+(alku+0.0000001*alue)*180/pi;
loppu=theta1+(loppu-0.0000001*alue)*180/pi;
```

Loput funktiosta on sen runko-osaa, jossa varsinainen laskenta ja palautettavien arvojen sijoittaminen argumentteihin tapahtuu. m-tiedostossa ei siis ole erikseen funktion esittelyä ja määrittelyä, vaan määrittelyn ensimmäinen rivi hoitaa samalla esittelyn. Funktion runko-osa voi sisältää kaikkia MATLABissa mahdollisia toimintoja, kuten esimerkiksi muiden funktioiden kutsuja, ohjelmavuon ohjausrakenteita, interaktiivista syöttötietojen antoa, tu-

lostusta, laskentaa, sijoituskäskyjä, kommentteja ja tyhjiä rivejä. Kommentteja voi sijoittaa myös käskyjen perään samalle riville kirjoittamalla ne %-merkin jälkeen.

Funktioiden nimeämisessä on samat rajoitukset kuin muuttujien nimeämisessä. Funktion nimen täytyy alkaa kirjaimella, minkä jälkeen voi olla mielivaltainen määrä kirjaimia, numeroita ja alaviivoja, näistä MATLAB käyttää 31 ensimmäistä merkkiä. Isot ja pienet kirjaimet erotellaan. Tallennettaessa funktiota `m`-tiedostoksi, kannattaa se selvyyden vuoksi nimetä muodossa `funktionimi.m`, vaikka se ei ole pakollista. Funktiota kutsuttaessa on joka tapauksessa käytettävä `m`-tiedostonimeä.

Funktioita voidaan kutsua komentoriviltä tai toisista `m`-tiedostoista. Esimerkiksi komentoriville annettu käsky

```
>> [a,b]=rajat(3,2,1,2,0)
```

kutsuu funktiota `rajat` argumenttien arvoilla `r1=3`, `r2=2`, `r3=1`, `r4=2` ja `theta1=0` palauttaen kampikulman ääriarvot sijoitettuna muuttujiin `a` ja `b`. Muuttujat `a` ja `b` tulostuvat komentoikkunaan, koska kutsun jälkeen ei ole puolipistettä. Komentoriville annetut käskyt

```
>> s=3;t=2;
>> u=1;v=2;
>> w=0;
>> [a,b]=rajat(s,t,u,v,w);
```

aiheuttavat muuten samat toiminnot kuin edellä, mutta nyt muuttujia `a` ja `b` ei tulosteta.

Kohdatessaan uuden nimen MATLAB liittää sen tiettyyn funktioon tutkimalla olemassa olevia nimiä seuraavassa järjestyksessä:

1. Tutkitaan, onko kyseessä muuttuja.
2. Tutkitaan, onko kyseessä kutsuvan funktion kanssa samassa `m`-tiedostossa sijaitseva alifunktio.
3. Tutkitaan, onko kyseessä kutsuvan funktion sijaintihakemiston yksityisessä alihakemistossa sijaitseva yksityinen funktio. Yksityisen funktion määritelmä esitetään luvussa 8.
4. Tutkitaan, onko kyseessä MATLABin hakemistopolussa sijaitseva funktio, jolloin MATLAB päätyy ensimmäisenä löytyneeseen funktioon.

Jos käyttäjällä on useita samannimisiä funktioita, MATLAB suorittaa tiettyä funktiota kutsuttaessa yllä olevien sääntöjen perusteella ensimmäisenä löytyvän funktion. Funktioiden nimiä on tosin mahdollista ylikuormittaa, jolloin sääntöjä tulee lisää.

Kun funktioita kutsutaan komentoriviltä tai `m`-tiedostosta, MATLAB jäsentää funktion pseudokoodiksi ja tallentaa sen muistiin, jolloin jäsennystä ei tarvitse suorittaa uudelleen, mikäli funktiota kutsutaan uudelleen saman istunnon aikana. Pseudokoodi pysyy muistissa niin kauan, kunnes MATLAB suljetaan tai muistia tyhjennetään `clear`-komennon avulla. Komento `clear funktionimi` poistaa muistista funktion `funktionimi` pseudokoodin, `clear functions` poistaa kaikkien funktioiden pseudokoodit ja `clear all` poistaa kaikkien funktioiden pseudokoodit ja lisäksi kaikki muuttujat.

Pseudokoodiksi jäsennettyjä funktioita voidaan myös tallentaa ns. p-tiedostoiksi myöhempiä MATLAB istuntoja varten käyttämällä `pcode`-funktiota. Tästä on harvoin hyötyä, mutta esimerkiksi suuria käyttöliittymiä ohjelmoitaessa se voi nopeuttaa toimintoja. p-tiedostoa voidaan myös käyttää silloin, jos sitä vastaava `m`-tiedosto halutaan salata.

MATLAB siirtää syöttötietona annettavan argumentin funktiolle arvona vain, jos se muuttaa tätä arvoa. Jos funktio ei muuta argumentin arvoa, se siirretään muistin säästämiseksi viittauksena.

Jokaisella `m`-tiedostoon sijoitetulla funktiolla on oma muistialueensa, jossa se operoi ja joka on erillään komentoikkunan ja muiden funktioiden muistialueista. Yleensä vain funktion omassa muistialueessa sijaitsevat muuttujat ovat sen käytettävissä, ts. kuuluvat sen näkyvyysalueeseen. Funktiolle syöttötietoina siirrettävien argumenttien on siis oltava kutsuvan funktion näkyvyysalueessa ja kutsuttava funktio sijoittaa myös palautettavat argumentit sen näkyvyysalueeseen. Muuttuja voidaan kuitenkin tarvittaessa määritellä globaaliksi, jolloin se voi olla samanaikaisesti useamman eri funktion näkyvyysalueessa.

Yleensä funktiota kutsutaan kaikkia sen syöttötietoargumentteja käyttäen ja se palauttaa arvot kaikille palautukseen varatuille argumenteille. MATLABissa on tosin käytettävissä funktiot `nargin` ja `nargout`, joiden avulla funktion runko-osassa voidaan selvittää, montaako argumenttia sen kutsussa on käytetty. Argumenttien lukumäärän perusteella voidaan sitten suorittaa erilaisia toimintoja. Lisäksi on käytettävissä funktiot `varargin` ja `varargout`, jotka tekevät mahdolliseksi kutsua tiettyä funktiota mitä tahansa argumenttimääriä käyttäen.

4 Muuttujat

`m`-tiedostossa käytettävällä muuttujilla on samat ominaisuudet kuin komentoikkunassa käytettävällä muuttujalla. Niiden nimien täytyy alkaa kirjaimella, minkä jälkeen voi olla mielivaltainen määrä kirjaimia, numeroita ja alaviivoja, joista MATLAB käyttää 31 ensimmäistä merkkiä. Isot ja pienet kirjaimet erotellaan.

Muuttujia ei tarvitse esitellä eikä niiden tyyppiä määritellä ennen niiden käyttöönottoa. Kahden muuttujan välisessä sijoituskäskyssä sijoitettavalla muuttujalla (sijoituskäskyn oikealla puolella) täytyy olla arvo. Jokainen operaatio joka sijoittaa muuttujaan arvon, luo samalla tämän muuttujan tai korvaa jo olemassa olevan muuttujan entisen arvon.

MATLABin muuttujat kuuluvat määrättyyn näkyvyysalueeseen, jotka ovat komentoikkunan näkyvyysalue, funktioiden omat näkyvyysalueet sekä komentoikkunan ja/tai tiettyjen funktioiden yhdistetty eli globaali näkyvyysalue. Funktioiden vuon ohjausrakenteissa (esimerkiksi `if`-rakenne) ei ole omia edellistä tasoa laajentavia näkyvyysalueita.

Tavallisesti jokaisella MATLABin funktiolla on omat lokaalit muuttujansa, jotka ovat täysin erillisiä muiden funktioiden ja komentoikkunan muuttujista. Muuttuja voidaan tarvittaessa määritellä globaaliksi komennolla `global`, jolloin useampi funktio ja mahdollisesti komentoikkuna voivat jakaa tämän yhteisen muuttujan. Esimerkiksi komento

```
global MAKSIMI MINIMI
```

määrittelee muuttujat `MAKSIMI` ja `MINIMI` globaaleiksi. Muuttujan määrittely globaaliksi on suoritettava jokaisessa sen jakavassa funktiossa erikseen. Globaalin muuttujan arvoa voidaan muuttaa kaikista sen jakavista funktioista käsin. Globaaliksi määrittely tulee suorittaa ennen muuttujan ensimmäistä esiintymistä funktiossa, joten se on parasta sijoittaa heti `m`-tiedoston alkuun. Usein globaalien muuttujien nimissä käytetään pelkästään isoja kirjaimia, jolloin ne erottuvat paremmin muista muuttujista. Globaaleja muuttujia kannattaa mahdollisimman pitkälle välttää, sillä niiden kanssa syntyneet virhetilanteet ovat yleensä hyvin vaikeita selvittää.

Muuttuja voidaan määritellä myös pysyväksi komennolla `persistent`, mikä tarkoittaa sitä, että sen arvo pysyy samana funktion peräkkäisissä kutsuissa. Näitä muuttujia voidaan käyttää vain funktioissa, ei siis komentoikkunassa. `persistent`-muuttujan arvo säilyy muistissa niin kauan, kunnes vastaava funktio poistetaan `clear`-komennolla muistista tai sitä muutetaan.

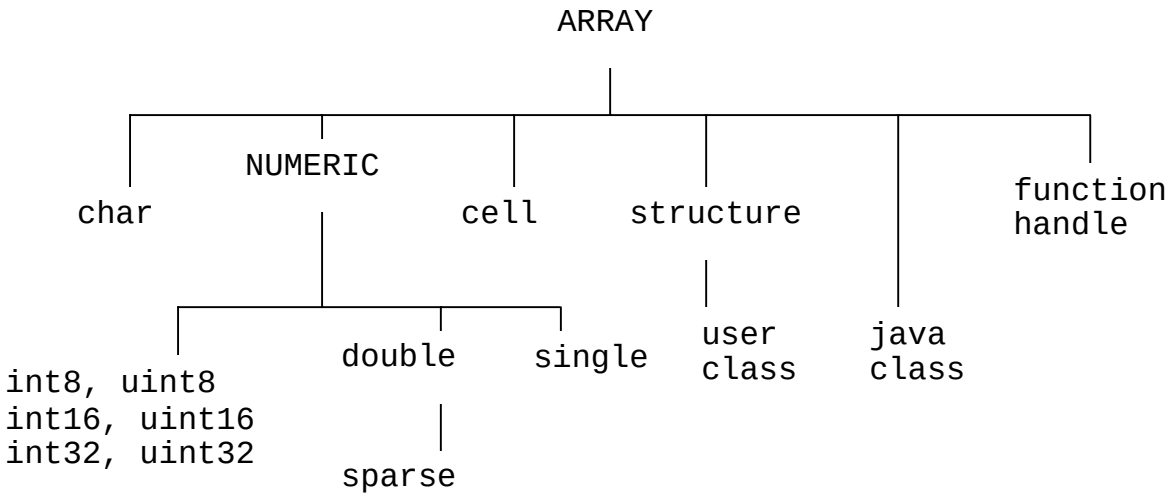
MATLABissa on käytettävissä joitakin usein esiintyviä arvoja palauttavia funktioita, joita voidaan toisinaan hyödyntää `m`-tiedostoissa. Näitä on esitetty seuraavassa taulukossa:

Funktio	Palautettava arvo
`ans`	MATLAB sijoittaa automaattisesti viimeksi lasketun vastauksen muuttujaan `ans`, ellei sitä sijoiteta johonkin muuhun muuttujaan.
`eps`	Liuku-lukulaskennan suhteellinen tarkkuus. Toleranssi, jota MATLAB käyttää numeerisessa laskennassa.
`realmax`	Suurin liukuluku, jonka tietokone pystyy esittämään.
`realmin`	Pienin liukuluku, jonka tietokone pystyy esittämään.
`pi`	3.1415926535897…
`i` tai `j`	Imaginääriyksikkö.
`inf`	Ääretön. Esimerkiksi `n/0=inf`, kun `n`≠`0`.
`NaN`	Ei numeerista arvoa. Esimerkiksi `inf/inf=NaN`. Kaikki aritmeettiset operaatiot, jossa on mukana `NaN`, antavat tuloksen `NaN`.
`computer`	Tietokoneen tyyppi.
`version`	MATLABin version antava merkkijono.

5 Tietotyypit ja tietorakenteet

MATLABissa on käytettävissä 14 erilaista tietotyyppiä ja -rakennetta. Jokainen tietotyyppi on perusluonteeltaan taulukko (array). Nämä taulukot voivat olla n-ulotteisia (n kokonaisluku) ja dimensioiltaan kuinka suuria tahansa (käytettävissä olevan tilan puitteissa). Myös tyhjien taulukkojen käyttö on luvallista (jotkut dimensiot nollia). Kaksiulotteiset taulukot ovat tavanomaisia matriiseja. Tietotyypit ja -rakenteet on esitetty seuraavan sivun kaaviossa pienillä kirjaimilla kirjoitettuna. Lisäksi siinä on typpi `user class`, joka on käyttäjän määrittelemä `structure`-tietotyyppi.

`char`-tietotyyppiin tallennetaan merkkejä Unicode-muodossa. Merkkijono on 1×n-taulukko ja samanpituisia merkkijonoja voi tallentaa esimerkiksi kaksiulotteiseen `char`-tyyppiseen taulukkoon. Eripituisia merkkijonoja voi tallentaa `cell`-tyyppiseen taulukkoon.



Numeerisen tiedon tallennukseen on käytettävissä kokonaislukutyypit (`int8, int16, int32`), etumerkittömät kokonaislukutyypit (`uint8, uint16, uint32`), yksinkertaisen tarkkuuden liukuluku (`single`), kaksinkertaisen tarkkuuden liukuluku (`double`) ja harvojen taulukoiden kaksinkertaisen tarkkuuden liukuluku (`sparse`). MATLAB suorittaa tietotyypistä huolimatta kaiken numeerisen laskennan kaksinkertaisella tarkkuudella. Mikäli kokonaisluku- tai `single`-tyyppisillä taulukoilla suoritetaan matemaattisia operaatioita, on ne ensin muunnettava `double`-tyyppisiksi. Kokonaislukutyyppien ja yksinkertaisen tarkkuuden typpien käytöllä voidaan tietoja tallennettaessa säästää muistitilaa. Kokonaislukutyypeistä kannattaa yleensä käyttää vain `int32`-tyyppiä, koska se on nykyään useimpien tietokoneiden natiivi kokonaislukutyyppi, ja näin ollen sitä käytettäessä laskenta on nopeinta. Muiden kokonaislukutyppien käyttöön pitää olla painavat perusteet, muistin säästäminen ei ole tässä asiassa kovin tärkeää.

`cell`-tietotyyppi on solutaulukko, jonka solut voivat olla muita tietotyyppejä edustavia taulukoita. Tietyn `cell`-tietotyyppiä olevan taulukon jokin solu voi olla esimerkiksi `char`-tyyppinen m×n-taulukko, jokin toinen solu `double`-tyyppinen p×q-taulukko jne. Solutaulukon soluihin viitataan indeksien avulla kuten tavallisen taulukon alkioihin.

`structure`-tietotyyppi on siinä suhteessa samankaltainen `cell`-tietotyypin kanssa, että siihen voidaan tallentaa eri tietotyyppejä edustavia tietoja. `structure`-tietotyyppiä olevan taulukon alkiot ovat tietorakenteita, jotka koostuvat nimetyistä kentistä. Viittaukset `structure`-taulukon alkioon tehdään indekseillä ja alkiona olevassa tietorakenteessa viitataan tiettyyn tietoon kentän nimellä.

Kaikki MATLABin tietotyypit on ohjelmoitu luokkina. Käyttäjä voi ohjelmoida lisää näitä luokkia, jolloin ne periytyvät `structure`-luokasta. Edellä olevassa kaaviossa `user class`-tietotyyppi tarkoittaa käyttäjän ohjelmoimia luokkia.

MATLABissa on liittymä Java-ohjelmointikieleen, mikä tekee mahdolliseksi luoda objekteja (olioita) Java-luokista ja kutsua Java-metodeita näitä objekteja käyttäen. MATLABin mukana tulee joitakin Java-luokkia ja niitä on myös mahdollista tuoda muualta lisää. Kaaviossa `java class`-tietotyyppi tarkoittaa näitä luokkia.

`function handle` on tietotyyppi, joka sisältää informaatiota funktioon viittaamista varten. Kun `function handle`-tyyppiä olevan muuttuja luodaan ja siihen sijoitetaan funktio, MATLAB tallentaa tähän muuttujaan kaiken informaation, joka tarvitaan funktion paikallistamiseksi ja suorittamiseksi eli evaluoimiseksi. Funktio `feval` ottaa syöttötietoargumenttinaan vastaan `function handle`-tyyppiä olevan muuttujan ja evaluoi sitä vastaavan funktion.

6 Operaattorit

MATLABin operaattorit jakaantuvat kolmeen ryhmään. Aritmeettiset operaattorit suorittavat numeerisia laskutoimituksia, vertailuoperaattorit tutkivat operandiensa suuruussuhteita ja kolmannen ryhmän muodostavat loogiset operaattorit. Loogisten operaattoreiden lisäksi on käytettävissä joitakin käteviä loogisia funktioita.

MATLABissa on käytettävissä seuraavan taulukon mukaiset aritmeettiset operaattorit. Niistä `+`, `-`, `.*`, `./`, `.\`, `.^`, `.'` ja `:` ovat alkioittain toimivia taulukko-operaattoreita, jolloin operandien dimensioiden tulee yleensä olla samat. Poikkeuksena ovat skalaarin ja taulukon väliset operaatiot, jotka suoritetaan tavanomaisesti laajentamalla skalaari ensin taulukoksi, jonka kaikki alkiot ovat kyseisen skalaarin suuruisia.

Operaattori	Kuvaus
`+`	Yhteenlasku tai etumerkki alkioittain.
`-`	Vähennyslasku tai etumerkki alkioittain.
`.*`	Kertolasku alkioittain.
`./`	Oikeanpuoleinen jakolasku alkioittain.
`.\`	Vasemmanpuoleinen jakolasku alkioittain.
`.^`	Potenssiin korotus alkioittain.
`.'`	Transpoosi.
`'`	Konjugaatin transpoosi.
`*`	Matriisien kertolasku.
`/`	Matriisien oikeanpuoleinen jakolasku.
`\`	Matriisien vasemmanpuoleinen jakolasku.
`^`	Matriisin potenssiin korotus.
`:`	Kaksoispisteoperaattori. Määrittelee taulukon alueita.

`+`, `-`, `*`, `/`, `\`, `^`, `'` ja `:` ovat normaaleja matriisilaskennan toimintoja, operaattorit `*`, `/` ja `\` toimivat matriisilaskennan sääntöjen mukaan myös toisen operandin ollessa skalaari.

Vertailuoperaattoreita on kuusi kappaletta ja ne on esitetty seuraavassa taulukossa.

Operaattori	Kuvaus
<	Pienempi kuin.
<=	Pienempi tai yhtä suuri kuin.
>	Suurempi kuin.
>=	Suurempi tai yhtä suuri kuin.
==	Yhtä suuri kuin.
~=	Erisuuri kuin.

Vertailuoperaattorit tutkivat taulukoita alkioittain, jolloin vertailtavien taulukoiden dimensioiden tulee yleensä olla samat. Poikkeuksena ovat skalaarin ja taulukon välinen vertailu, jotka suoritetaan tavanomaisesti laajentamalla skalaari ensin taulukoksi, jonka kaikki alkiot ovat kyseisen skalaarin suuruisia.

Loogisia operaattoreita on kolme seuraavan taulukon mukaisesti.

Operaattori	Kuvaus
&	AND
|	OR
~	NOT

Myös loogiset operaattorit käsittelevät taulukoita alkioittain ja niillä on samat säännöt dimensioiden suhteen kuin vertailuoperaattoreilla.
Loogisten operaattoreiden lisäksi MATLABissa on runsaasti hyödyllisiä loogisia funktioita, kuten esimerkiksi funktiot `xor`, `all` ja `any`.

Lausekkeita kirjoitettaessa voidaan operaattoreita yhdistellä halutulla tavalla, jolloin syntyy useita operaatioita sisältäviä lausekkeita. Operaatioiden suoritusjärjestys määrää, missä järjestyksessä lausekkeen arvo määritetään. Operaatioiden suoritusjärjestys on seuraava:

1. Sulut ()
2. Transpoosi (`.'`), potenssi (`.^`), konjugaatin transpoosi (`'`), matriisin potenssi (`^`).
3. Etumerkki plus (+), etumerkki miinus (`-`), looginen NOT (~).
4. Kertolasku (`.*`), oikeanpuoleinen jakolasku (`./`), vasemmanpuoleinen jakolasku (`.\`), matriisien kertolasku (`*`), matriisien oikeanpuoleinen jakolasku (`/`), matriisien vasemmanpuoleinen jakolasku (`\`).
5. Yhteenlasku (+), vähennyslasku (`-`).
6. Kaksoispisteoperaatio (`:`).
7. Pienempi kuin (<), pienempi tai yhtä suuri kuin (<=), suurempi kuin (>), suurempi tai yhtä suuri kuin (>=), yhtä suuri kuin (==), erisuuri kuin (~=).
8. Looginen AND (&).
9. Looginen OR (|).

Suoritusjärjestyksessä samalla tasolla olevat operaatiot suoritetaan vasemmalta oikealle.

7 Vuon ohjaus

MATLABissa on kahdeksan ohjelmavuon ohjausrakennetta, jotka yhdessä `end`-komennon kanssa tekevät mahdolliseksi ohjata tehokkaasti funktiossa suoritettavia toimintoja.

7.1 `if`, `else`, `elseif`

`if`-rakenne evaluoi loogisen lausekkeen ja suorittaa joukon komentoja lausekkeen saaman arvon perusteella. Yksinkertaisin `if`-rakenteen syntaksi on seuraava:

```
if looginen_lauseke
    komentoja
end
```

Jos *looginen_lauseke* saa arvon tosi (=1), MATLAB suorittaa kaikki `if` ja `end` rivien välissä olevat komennot ja jatkaa funktion suoritusta `end` riviä seuraavalta riviltä. Jos *looginen_lauseke* saa arvon epätosi (=0), MATLAB ohittaa kaikki `if` ja `end` rivien välissä olevat komennot ja jatkaa funktion suoritusta `end` riviä seuraavalta riviltä.

`if`-rakenteita voidaan käyttää haluttu määrä sisäkkäin, toisin sanoen *komentoja*-osassa voi olla uusia `if`-rakenteita.

Mikäli *looginen_lauseke* ei evaluoidu skalaariksi (1 tai 0) vaan taulukoksi, tarkoittaa sen arvo tosi kaikkien taulukon alkioiden evaluoitumista arvoksi tosi (=1). Mikäli *looginen_lauseke* evaluoituu tyhjäksi taulukoksi on sen arvo epätosi (=0).

Käyttämällä `else`-lausetta saadaan aikaan mutkikkaampia ehdollisia toimintoja. Syntaksi on tällöin seuraava:

```
if looginen_lauseke
    komentoja1
else
    komentoja2
end
```

Jos *looginen_lauseke* saa arvon tosi (=1), MATLAB suorittaa komennot *komentoja1* ja jatkaa `end` riviä seuraavalta riviltä. Jos *looginen_lauseke* saa arvon epätosi (=0), MATLAB suorittaa komennot *komentoja2* ja jatkaa `end` riviä seuraavalta riviltä.

Vieläkin mutkikkaampia toimintoja voidaan toteuttaa yhden tai useamman `elseif`-lauseen avulla, jolloin syntaksi on seuraava:

```
if looginen_lauseke1
    komentoja1
elseif looginen_lauseke2
    komentoja2
else
    komentoja3
end
```

Jos *looginen_lauseke1* on tosi (=1), MATLAB suorittaa komennot *komentoja1* ja jatkaa suoritusta `end` riviä seuraavalta riviltä. Jos *looginen_lauseke1* on epätosi (=0), MATLAB evaluoi lausekkeen *looginen_lauseke2.* Jos *looginen_lauseke2* on tosi (=1), MATLAB suorittaa komennot *komentoja2* ja jatkaa suoritusta `end` riviä seuraavalta riviltä. Jos *looginen_lauseke2* on epätosi (=0) MATLAB suorittaa komennot *komentoja3* ja jatkaa suoritusta `end` riviä seuraavalta riviltä.

7.2 while

while-rakenne suorittaa käskyjoukkoa toistuvasti niin kauan, kun ohjaava lauseke on tosi (=1). Sen syntaksi on seuraava:

```
while lauseke
    komentoja
end
```

Käskyjä *komentoja* suoritetaan toistuvasti niin kauan kuin lauseke on tosi. Kun lauseke saa arvon epätosi, siirtyy suoritus end-riviä seuraavalle riville. while-rakenteen sisällä voi olla break-lause, joka tietyn ehdon täyttyessä aiheuttaa while-rakenteesta poistumisen, vaikka ohjaava lauseke on tosi.

Jos *lauseke* evaluoituu matriisiksi, täytyy kaikkien sen alkioiden saada arvo 1, jotta suoritus while-rakenteessa jatkuisi. Mikäli *lauseke* evaluoituu tyhjäksi matriisiksi, on sen arvo epätosi (=0).

7.3 switch, case, otherwise

switch-rakenne suorittaa tietyn komentojoukon sen perusteella, minkä arvon jokin muuttuja tai lauseke saa. switch-rakenteen syntaksi on seuraava:

```
switch lauseke
    case arvo1
        komentoja1
    case arvo2
        komentoja2
    .
    .
    .
    case arvoN
        komentojaN
    otherwise
        komentoja_muuten
end
```

switch-rakenteen ensimmäisenä rivinä on avainsana switch ja sitä seuraava evaluoitava *lauseke*, jonka arvon tulee olla skalaari tai merkkijono. Sitä seuraavat case-ryhmät alkavat avainsanalla case, jota seuraa samalla rivillä *lausekkeen* mahdollinen *arvo*. Tämän rivin jälkeen tulee joukko komentoja, jotka suoritetaan, mikäli *lauseke* saa kyseisen arvon. Komennot voivat olla mitä tahansa MATLABin komentoja switch-rakenteet mukaan lukien. case-ryhmän suoritus päättyy heti, kun MATLAB löytää seuraavan case-käskyn tai otherwise-käskyn. MATLAB suorittaa vain sen case-ryhmän, jonka *arvo* ensimmäisenä täsmää *lausekkeen* evaluoituun arvoon. Viimeisenä ryhmänä on otherwise-ryhmä, joka voidaan jättää poiskin. otherwise-käskyn jälkeen tulevat komennot,

jotka suoritetaan, ellei *lausekkeen* evaluoitu arvo täsmää minkään case-ryhmän *arvon* kanssa. otherwise-ryhmän suoritus päättyy end-käskyn tullessa vastaan.

Jos *lauseke* evaluoituu numeeriseen arvoon, täsmääminen case-ryhmän *arvoon* ratkeaa vertailusta arvo==lauseke. Jos *lauseke* evaluoituu merkkijonoksi, täsmääminen case-ryhmän *arvoon* ratkeaa merkkijonoja vertailevan funktion strcmp('arvo','lauseke') palauttamasta arvosta. Funktio strcmp palauttaa arvon tosi (=1) vain, jos merkkijonot arvo ja lauseke ovat täsmälleen samat.

7.4 for

for-rakenne toistaa komentojoukkoa ennalta annetun määrän kertoja. Sen syntaksi on

```
for laskuri = alku:muutos:loppu
    komentoja
end
```

muutos voi olla mikä tahansa positiivinen tai negatiivinen luku. Jos *muutos* on positiivinen, for-rakenteen suoritus päättyy, kun *laskuri* ylittää arvon *loppu*. Jos *muutos* on negatiivinen, for-rakenteen suoritus päättyy, kun *laskuri* alittaa arvon *loppu*. Jos muutos=1, voidaan osa muutos: jättää poiskin. *laskuri* saa lisäyksensä *muuto* silmukan lopussa.

Sisäkkäisten for-rakenteiden käyttö on luonnollisesti sallittua.

7.5 continue

continue-komentoa voidaan käyttää for- tai while-rakenteessa siirtämään suoritus tietyn ehdon täyttyessä meneillään olevan toistokerran loppuun, jolloin tämän toistokerran loput komennot ohitetaan. for-rakenteessa *laskuri* saa lisäyksensä tässäkin tapauksessa toistokerran lopussa. Suoritus siirtyy sitten seuraavan toistokerran alkuun, mikäli silmukka vielä jatkuu. Sisäkkäisissä rakenteissa suoritus siirtyy continue-komennolla (mahdollisen *laskurin* lisäyksen jälkeen) sen rakenteen alkuun, jonka for/while ja end käskyjen välissä continue-komento sijaitsee, mikäli tämä silmukka vielä jatkuu.

7.6 break

break-komentoa voidaan käyttää for- tai while-rakentessa keskeyttämään suoritus tietyn ehdon täyttyessä. break-komennon jälkeen suoritus siirtyy kyseisen rakenteen end-komennon jälkeiselle riville. Sisäkkäisissä rakenteissa suoritus siirtyy break-komennolla sen rakenteen end-komennon jälkeiselle riville, jonka for/while ja end komentojen välissä break-komento sijaitsee.

7.7 try...catch

`try...catch`-rakenne muuttaa suoritusjärjestystä virhetilanteen sattuessa. Syntaksi on

```
try,
    komento1,
    komento2,
    .
    .
    .
    komentoN,
catch,
    komentoN+1,
    komentoN+2,
    .
    .
    .
    komentoN+M,
end
```

`try` ja `catch` komentojen välissä olevia komentoja suoritetaan, kunnes suorituksessa tulee virhe. Funktio `lasterr` palauttaa merkkijonon, joka sisältää syntyneestä virheestä tulleen virheilmoituksen. Virheen sattuessa suoritetaan komentoja `catch` ja `end` komentojen välissä. Näissä komennoissa voidaan suoritusta ohjata funktion `lasterr` palauttaman merkkijonon perusteella. Jos tässä vaiheessa tulee virhe, MATLAB lopettaa suorituksen, paitsi jos ollaan toisen `try...catch`-rakenteen sisällä olevassa `try...catch`-rakenteessa ja virhe saadaan vielä ulommassa rakenteessa kiinni.

7.8 return

`return`-komento päättää meneillään olevan funktion komentojen suorituksen ja palauttaa vuon ohjauksen tätä funktiota kutsuneelle funktiolle tai näppäimistölle. Kutsuttu funktio palauttaa normaalisti ohjauksen kutsuneelle funktiolle suorituksen tullessa funktion loppuun. Kutsutun funktion koodiin sijoitettu `return`-komento voi aiheuttaa tietyn ehdon täyttyessä aikaisemman suorituksen päättymisen ja vuon ohjauksen siirtymisen.

8 Alifunktiot ja yksityiset funktiot

Samassa `m`-tiedostossa voi olla myös useita funktioita. Ensimmäisenä `m`-tiedostossa olevaa funktiota sanotaan primääriksi funktioksi ja sitä kutsutaan `m`-tiedoston nimellä. Primäärin funktion jäljessä olevat funktiot ovat sen alifunktioita. Alifunktiot ovat ainoastaan primäärin funktionsa ja muiden samassa `m`-tiedostossa sijaitsevien alifunktioiden näkyvyysalueessa. Jokainen alifunktio alkaa omalla määrittelyrivillään. Alifunktioiden järjestyksellä ei ole merkitystä, kunhan ne vain ovat primäärin funktion jälkeen.

Kullakin samassa `m`-tiedostossa olevalla funktiolla on omat muuttujansa, jotka eivät ole

muiden funktioiden näkyvyysalueessa, ellei muuttujaa ole määritelty globaaliksi tai sitä siirretä syöttötietoargumenttina toista funktiota kutsuttaessa. MATLABin `help`-funktiolla on pääsy vain primääriin funktioon.

Kutsuttaessa funktiota jostakin `m`-tiedostosta käsin, MATLAB tarkistaa ensin, onko tämä funktio kutsuvan funktion alifunktio. Seuraavaksi MATLAB tutkii, onko kutsuttava funktio kutsuvan funktion yksityinen funktio ja vasta sen jälkeen alkaa etsiä sitä käyttäjän hakupolusta. Tämä tekee mahdolliseksi korvata hakupolussa olemassa olevan funktion suoritus samannimisellä alifunktioilla ja yksityisillä funktioilla. Samassa `m`-tiedostossa olevilla alifunktioilla tulee kuitenkin olla eri nimet.

Yksityiset funktiot ovat `private`-nimisessä alihakemistossa olevia `m`-tiedostoja, jotka ovat ainoastaan niiden funktioiden näkyvyysalueessa, jotka sijaitsevat välittömästi `private`-hakemiston yläpuolella olevassa emohakemistossa. `private`-hakemistoja ei sijoiteta hakupolkuun.

9 Syöttötietojen antaminen suorituksen aikana

MATLABissa on kolme toimintoa, joiden avulla käyttäjä voi antaa syöttötietoja `m`-tiedoston suorituksen aikana.

`m`-tiedoston koodiin sijoitetun `input`-funktion avulla on mahdollista antaa syöttötietoa näppäimistöltä. Sen syntaksi on

```
x=input('kehote-teksti');
```

Funktio näyttää komentoikkunan kehotteen perässä tekstin *`kehote-teksti`* ja jää odottamaan näppäimistöltä annettavaa syöttötietoa. Enterin painamisen jälkeen annettu syöttötieto sijoitetaan muuttujaan `x` ja tiedoston suoritus jatkuu. Jos käyttäjä antaa syöttötietona lausekkeen, `input`-funktio evaluoi sen ennen muuttujaan sijoittamista. Syöttötietona voidaan antaa mikä tahansa MATLABin syntaksin mukainen lauseke, joka voidaan evaluoida komentoikkunan muuttuja-avaruuden muuttujien avulla, esimerkiksi matriiseja ja vektoreita voidaan siis antaa syöttötietoina. Jos ei anneta mitään syöttötietoa vaan painetaan vain Enter näppäintä, palauttaa `input`-funktio tyhjän taulukon.

`input`-funktion avulla voidaan numeerisen syöttötiedon lisäksi antaa näppäimistöltä myös merkkijonoja. Syntaksi on tässä tapauksessa

```
teksti=input('kehote-teksti','s');
```

`input`-funktio on erittäin hyödyllinen yksinkertaisia käyttöliittymiä ohjelmoitaessa.

Toinen keino vaikuttaa `m`-tiedoston suorittamiseen näppäimistöltä on `pause`-funktio, joka pysäyttää ohjelman suorituksen. Ilman argumentteja oleva `pause` pysäyttää tiedoston suorittamisen, kunnes käyttäjä painaa jotakin näppäintä. Komento `pause(n)` aiheuttaa suorituksen pysähtymisen n sekunnin ajaksi.

Mutkikkaiden syöttötietojen antamiseen on mahdollista ohjelmoida graafinen käyttöliittymä MATLABin käyttöliittymäeditorilla. Sillä saadaan aikaan suuresti Windows-käyttöliittymiä muistuttavia käyttöliittymiä.

10 Koodin optimointi ja muistin tehokas käyttö

MATLAB perustuu taulukkomuotoisen datan käsittelyyn, mistä johtuen se suoriutuu erityisen hyvin vektori- ja matriisioperaatioista. Tätä ominaisuutta kannattaa hyödyntää `m`-tiedostoja ohjelmoitaessa.

`m`-tiedoston suoritus nopeutuu, mikäli sinä olevia silmukoita vektoroidaan. Tämä tarkoittaa `for`- ja `while`-ohjausrakenteiden muuttamista samat toiminnot suorittaviksi vektori- ja matriisioperaatioiksi. Esimerkiksi `for`-silmukka

```
i=0;
for x=0:0.001:pi
  i=i+1;
  y(i)=sin(x);
end
```

voidaan vektoroida muotoon `x=0:0.001:pi;  y=sin(x);`

Koodin suoritusta voidaan usein nopeuttaa varaamalla etukäteen taulukot, joihin jatkossa sijoitetaan tietoja. Tällöin taulukon kokoa ei tarvitse jatkossa päivittää. Suurilla taulukoilla vältetään myös tallennettavan tiedon pirstoutuminen muistissa. Taulukoiden varaamiseen voidaan käyttää esimerkiksi funktiota `zeros`.

Muistin käytön tehostamiseen MATLABissa on viisi käyttökelpoista funktiota. `clear` poistaa muuttujan muistista, `pack` siirtää olemassa olevat muuttujat levyltä haettaviksi (ei kannata käyttää silmukoissa), `save` tallentaa halutut muuttujat levylle ja `load` lataa ne sieltä muistiin. `quit` lopettaa MATLABin suorituksen ja vapauttaa sen varaaman muistin.

11 Tiedosto I/O

MATLABissa on käytettävissä joukko I/O-funktioita, jotka muistuttavat suuresti C-kielen vastaavia funktioita, mutta toimivat toisinaan hieman eri tavalla lähinnä MATLABin taulukko-orientoitumisen vuoksi. I/O-toiminnoissa on seuraavat tyypilliset kolme vaihetta:

1. Käsiteltävä tiedosto avataan `fopen`-funktiolla, joka palauttaa tiedoston tunnistimen, jolla tiedostoon jatkossa viitataan.
2. Suoritetaan tiedostossa luku ja kirjoitusoperaatioita, jolloin mahdollisia toimintoja ovat: Luetaan ASCII-dataa `fscanf`-funktiolla. b) Kirjoitetaan ASCII-dataa `fprintf`-funktiolla. c) Luetaan tiedoston rivejä `fgets`- tai `fgetl`-funktiolla. d) Luetaan binääridataa `fread`-funktiolla. e) Kirjoitetaan binääridataa `fwrite`-funktiolla.
3. Suljetaan tiedosto `fclose`-funktiolla.

Avattuaan tiedoston MATLAB ylläpitää tiedosto-osoitinta, jonka avulla voidaan hallita datan käsittelyä. Tiedosto-osoittimen avulla voidaan määrätä, missä tiedoston kohdassa suoritettavat luku- ja kirjoitustoiminnot tapahtuvat. Tiedosto-osoittimen sijaintia ilmoitettaessa yksikkönä on tavu. Funktiolla `fseek` tiedosto-osoitin voidaan siirtää haluttuun paikkaan ja funktio `ftell` palauttaa sen sijainnin. Funktion `fseek` syntaksi on seuraava:

```
tila = fseek(tunniste,poikkeama,origo)
```

Palautettava arvo `tila=0`, jos `fseek` toiminto onnistuu ja `tila=-1`, jos se ei onnistu. Jälkimmäisessä tapauksessa lisää informaatiota virheestä saa `ferror`-funktiolla. `tunniste` on `fopen`-funktion palauttama tiedostotunniste (=kokonaisluku). `origo` on tiedosto-osoittimen referenssikohta, joka voi saada arvot ’bof’ (=-1, beginning of file), ’cof’ (=0, current position in file) ja ’bof’ (=1, end of file). `poikkeama` on tiedosto-osoittimen sijainti referenssikohtaan nähden, jos `poikkeama>0` sijainti on tiedoston loppua kohti ja jos `poikkeama<0` sijainti on tiedoston alkua kohti.

Funktiota `ftell` käytetään muodossa `paikka=ftell(tunniste)`. Palautettava arvo `paikka` on tiedosto-osoittimen sijainti (kokonaisluku, tavujen määrä) tiedoston alusta lukien. Mikäli `ftell`-toiminto epäonnistuu, `paikka=-1`.

11.1 fopen

Ennen tiedoston lukemista tai siihen kirjoittamista se on avattava `fopen` komennolla, jonka perussyntaksi on seuraava:

```
tunniste=fopen(’tiedostonimi’,’oikeudet’)
```

missä merkkijono `tiedostonimi` on avattavan tiedoston nimi ja merkkijono `oikeudet` määrittelee avattavaan tiedostoon syntyvät käyttöoikeudet (esimerkiksi `r` tarkoittaa lukuoikeutta ja on oletusarvona, `w` tarkoittaa tiedoston luonti-/kirjoitusoikeutta, jolloin kirjoitetaan mahdollisen aikaisemman datan päälle, jne.). Palautettava arvo `tunniste` on avaamisen onnistuessa positiivinen kokonaisluku, jolla avattuun tiedostoon jatkossa viitataan. Jos `fopen` toiminto ei onnistu, palautusarvo `tunniste=-1`. Standardi output (=1) ja error (=2) ovat automaattisesti käytettävissä ilman erillistä avaamistakin.

11.2 fscanf

Funktiolla `fscanf` luetaan tiedostosta ASCII-muotoista dataa. Käytettäessä funktiota `fscanf` muodossa

```
A=fscanf(tunniste,format)
```

se lukee matriisiin `A` kaiken datan tiedostosta, johon `tunniste` viittaa ja muuntaa lukemansa datan tallennusmuotoon, jonka merkkijono `format` määrittelee. `format`-merkkijono on muotoa `’määrittely1 märittely2 … määrittelyN’`, missä kukin `määrittely` on muotoa (hakasulut eivät kuulu määrittelyyn)

```
%[S1][S2].[S3][S4]
```

Merkkiä S1 ei tarvita usein, mutta se voi olla esimerkiksi * (=ohita). Luku S2 määrittelee luettavan numeerisen datan kentän pituuden ja S3 desimaalien määrän. S4 on datan tietotyyppiä kuvaava kirjain, esimerkiksi f tarkoittaa liukulukua ja d desimaalilukua. Näistä vain S4 on pakollinen.

Tarkastellaan esimerkkinä tapausta, jossa tiedosto kok.txt sisältää seuraavan datan

```
0.00    1.00000000
0.10    1.10517092
0.20    1.22140276
0.30    1.34985881
0.40    1.49182470
0.50    1.64872127
```

Antamalla komentoikkunassa komennot

```
tunniste=fopen('kok.txt','r');   % avataan lukuoikeuksin
A=fscanf(tunniste,'%g');   % luetaan data matriisiin A
fclose(tunniste);   % suljetaan tiedosto
```

saadaan tiedostossa kok.txt oleva data luetuksi pystyvektoriin A seuraavalla tavalla:

```
A =

  0.00
  1.00000000
  0.10
  1.10517092
  0.20
  1.22140276
  0.30
  1.34985881
  0.40
  1.49182470
  0.50
  1.64872127
```

Edellä olevasta näkyy, että MATLABin fscanf-funktio on vektoroitu, ts. se käyttää formaattia '%g' yhä uudelleen niin kauan kuin siihen sopivaa luettavaa riittää. Vaikka esimerkissä lukukäsky vaihdettaisiin muotoon A=fscanf(tunniste,'%g %g');, saadaan luettua edelleen sama vektori A.

Käytettäessä funktiota fscanf muodossa

```
[A,lukum2]=fscanf(tunniste,format,lukum1)
```

se toimii muuten kuten edellä, mutta lukee vain niin monta alkiota, jonka `lukum1` määrittelee ja palautettu arvo `lukum2` sisältää onnistuneesti luettujen alkioiden lukumäärän. Jos `lukum1=n` (n kokonaisluku), luetaan n alkiota vektoriin `A`, jos `lukum1=inf`, luetaan vektoriin `A` niin monta alkiota kun tiedostossa on ja jos `lukum1=[n,m]`, alkioita luetaan sarakkeittain `nxm`-matriisiin `A`.

Antamalla edellä olevaan esimerkkiin liittyen komentoikkunassa komennot

```
tunniste=fopen('kok.txt','r');   % avataan lukuoikeuksin
[A,lukum2]=fscanf(tunniste,'%g',[2,6]);   % luetaan data
fclose(tunniste);   % suljetaan tiedosto
```

saadaan tiedostossa `kok.txt` oleva data luetuksi matriisiin `A` seuraavalla tavalla:

```
A =

 0.00       0.10       0.20       0.30       0.40       0.50
 1.00000000 1.10517092 1.22140276 1.34985881 1.49182470 1.64872127

A=A';   % transponoidaan
A =

  0.00   1.00000000
  0.10   1.10517092
  0.20   1.22140276
  0.30   1.34985881
  0.40   1.49182470
  0.50   1.64872127
```

Edellä olevasta näkyy selvästi, että `fscanf`-funktio lukee matriisiin `A` dataa sarakkeittain.

11.3 fprintf

Funktio `fprintf` muuntaa dataa merkkijonoiksi ja kirjoittaa sen näytölle tai tiedostoon sille annetun muotoiluformaatin mukaisesti. Syntaksi on seuraavaa muotoa:

```
lukum3=fprintf(tunniste,format,A,...)
```

Kirjoitettava data sijaitsee muuttujassa `A` ja sen jälkeen mahdollisesti olevissa muissa muuttujissa. Palautettava arvo `lukum3` on kirjoitettujen tavujen lukumäärä ja `tunniste` viittaa tiedostoon johon kirjoitetaan tai on standardi output (=1) tai standardi error (=2). Näytölle kirjoitettaessa `tunniste` voi puuttua.

`format`-merkkijono sisältää samantyyppisiä `määrittelyjä` kuin vastaava `fscanf`–funktion `format`-merkkijono. Merkkien `S1` ja `S4` mahdollisissa arvoissa on kuitenkin eroa `fscanf`-funktioon nähden, mutta luvuilla `S2` ja `S3` on sama merkitys. Tässäkin tapauksessa vain `S4` on pakollinen. `Määrittelyjen` lisäksi `fprintf`-funktion `format`-merkkijono voi sisältää halutun valikoiman sellaisenaan tulostuvaa tekstiä ja tulostumattomia merkkejä (esim. rivinsiirto, tabulaattori, jne.).

MATLABin `fprintf`-funktio on myös vektoroitu, ts. se käyttää `format`-merkkijonoa yhä uudelleen niin kauan kuin siihen sopivaa kirjoitettavaa riittää.

Seuraavassa on esimerkki `fprintf`-funktion käytöstä edellä esillä olleen tiedoston `kok.txt` kirjoittamiseen.

```
x=0:0.1:0.5;
y=[x; exp(x)];

y =

 0     0.1         0.2         0.3         0.4         0.5
 1     1.10517092  1.22140276  1.34985881  1.4918247   1.64872127

tunniste=fopen('kok.txt','w');   % avataan kirjoitusoikeuksin
fprintf(tunniste,'%6.2f  %12.8f\n',y);   % kirjoitetaan data
fclose(tunniste);   % suljetaan tiedosto
```

`format`-merkkijono mukaan matriisin y ensimmäinen alkio kirjoitetaan kuuden merkin kenttään kahta desimaalia käyttäen `f`-notaatiolla (fixed point), sitten tulee kaksi välilyöntiä ja toinen alkio kirjoitetaan 12 merkin kenttään kahdeksaa desimaalia käyttäen `f`-notaatiolla, jonka jälkeen tulee rivinsiirto. Kirjoittaminen etenee matriisissa y sarakkeittain ja `format`-merkkijonoa käytetään aina uudelleen sarakkeen alkaessa.

Seuraavassa on vielä matriisin y tulostusesimerkki standardi outputiin 1 (=näyttö) hieman erilaista `format`-merkkijonoa käyttäen:

```
fprintf(1,'x= %6.2f  exp(x)= %12.8f\n',y)

x=   0.00  exp(x)=   1.00000000
x=   0.10  exp(x)=   1.10517092
x=   0.20  exp(x)=   1.22140276
x=   0.30  exp(x)=   1.34985881
x=   0.40  exp(x)=   1.49182470
x=   0.50  exp(x)=   1.64872127
```

11.4 fgets, fgetl

Funktiot `fgets` ja `fgetl` lukevat tekstitiedostosta rivejä ja sijoittavat ne merkkijonoina muuttujaan. `fgets` liittää merkkijonon viimeiseksi merkiksi rivinsiirron, mutta `fgetl` ei liitä. Syntaksi on seuraava

```
rivi=fgets(tunniste)
```

joka lukee `tunniste`-tiedoston seuraavan rivin ja sijoittaa sen muuttujaan `rivi`. Jos luettavan tiedoston tiedosto-osoitin on sen lopussa, palautusarvo `rivi=-1`.

11.5 fread

Funktio `fread` lukee tiedostosta binäärimuotoista dataa ja sijoittaa sen matriisiin. Syntaksi on hyvin pitkälti sama kuin `fscanf`-funktiolla, tallennukseen käytettävän tietotyypin määrittelyssä on kuitenkin selvästi eroa. `fscanf`-funktion syntaksi on esimerkiksi

```
[A,lukum2]=fread(tunniste,lukum1,tietotyyppi)
```

jolloin argumentti `tietotyyppi` määrittelee tallennusmuodon ja voi olla jokin MATLABin tunnistamista tietotyypeistä (`'double'`, `'int32'`, jne.).

11.6 fwrite

Funktio `fwrite` kirjoittaa binäärimuodossa matriisin `A` alkiot sarakkeittain tiedostoon, johon `tunniste` viittaa, halutussa tallennusmuodossa `tietotyyppi`. Syntaksi on tällöin esimerkiksi

```
lukum2 = fwrite(tunniste,A,tietotyyppi)
```

Palautettava arvo `lukum2` sisältää onnistuneesti kirjoitettujen alkioiden lukumäärän.

11.7 fclose

I/O-toimintojen päätyttyä tiedosto suljetaan `fclose` komennolla. Syntaksi on seuraava:

```
tila=fclose(tunniste)
```

jolloin suljetaan tiedosto, johon `tunniste` viittaa ja palautusarvo `tila=0` vastaa onnistunutta sulkemista ja palautusarvo `tila=-1` virhetilannetta. Antamalla tiedostotunnisteeksi `'all'`, `fclose` sulkee kaikki avoinna olevat tiedostot.

Suljettaessa MATLAB kaikki avoinna olevat tiedostot suljetaan automaattisesti, mutta on hyvä käytäntö sulkea itse ne tiedostot, joita ei enää meneillään olevassa istunnossa käytetä.