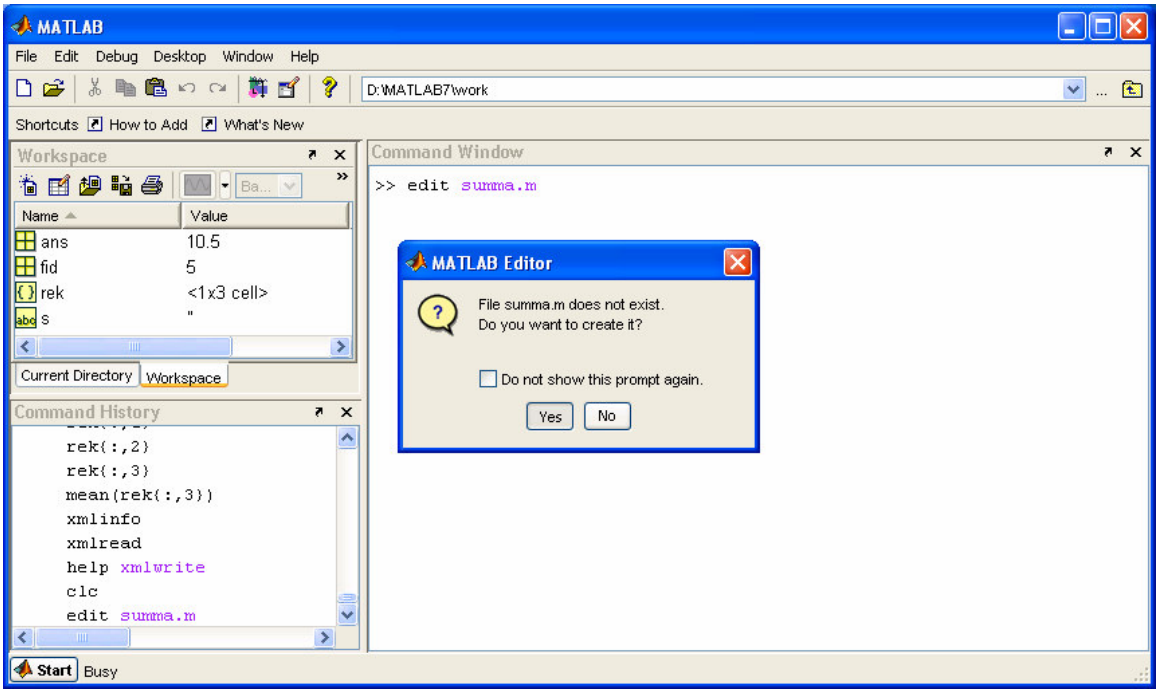
Ohjelmointi

Matlab-komentoja voidaan koota ns. M-tiedostoon. Nimi tulee tiedoston tarkentimesta `.m`. Matlabilla voidaan ohjelmoida kahdella eri tavalla:

- Skriptit eli komentojonot eli makrot
- Funktiot eli aliohjelmat

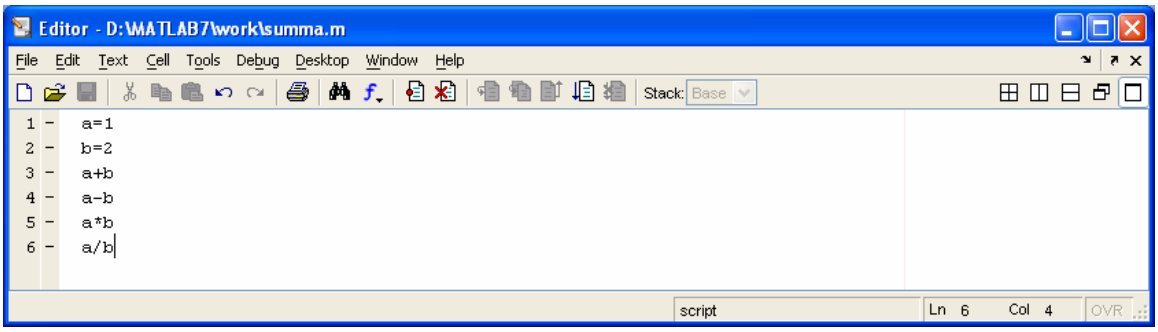
Kaikki tähän asti käsitellyt ”komennot” ovat itse asiassa Matlabin valmisfunktioita.

Molemmat toimivat samalla periaattella. Koodi kirjoitetaan tekstieditorilla ja tallennetaan tekstitiedostoon (plain text). Skriptin tai funktion nimi on oltava sama kuin tiedostonimen alkuosa:



Käynnistetään tekstieditori komentoriviltä kirjoittamalla `edit` ja tiedostonimi. Jos sitä ei ole ennestään olemassa, saadaan oheinen ilmoitus. Vastataan ”Yes”.

Skriptit



Tämä on **skripti**. Editoriin voidaan kirjoittaa rivi kerrallaan haluttuja komentoja ja valmisfunktioiden kutsuja. Skripti ajetaan sen nimellä, jolloin tiedostossa olevat rivit suoritetaan rivi kerrallaan.

```
>> summa
a =
     1
b =
     2
ans =
     3
ans =
    -1
ans =
     2
ans =
    0.5000
>> whos
  Name      Size                    Bytes  Class
  a         1x1                         8  double array
  ans       1x1                         8  double array
  b         1x1                         8  double array
Grand total is 3 elements using 24 bytes
```

Skriptille ei voi antaa parametreja komentoriviltä eikä se voi palauttaa arvoja muuttujiin. Ne siis toimivat tilanteissa, jossa muuttujien arvot eivät muutu (toki niitä voi muuttaa käsin editoimalla m-tiedostoa). Whos-komennon tulostuksesta nähdään, että skriptin käyttämät muuttujat jäävät näkyviin suorituksen jälkeen. Skriptimuuttujat ovat **globaaleja** muuttujia. Jos työmuistissa on toinen samanniminen muuttuja, skripti kirjoittaa sen päälle.

Tehtävä: Kirjoita skripti, joka lataa edellisessä harjoituksessa tutkitun hiilidioksidin määrän eri vuosina sisältävän `carb_dio.txt`-tiedoston Matlabiin, tulostaa kuvaajan sekä lisää siihen akselien nimet ja otsikon.

Funktiot

Matlabin funktioita käytetään samalla periaatteella kuin C-kielen funktioita. Myös funktion tarkennin on `.m`, ja sitä kutsutaan tiedostonimen alkuosalla. Yksi tiedosto sisältää täsmälleen yhden funktion, joka tekee yhden rajatun toiminnon.

Funktiolle voidaan välittää arvoja parametrien avulla ja se voi palauttaa yhden tai useamman arvon (skalaari, vektori, matriisi tai merkkijono) paluuarvonaan.

Funktiotiedoston ensimmäisellä rivillä täytyy olla syntaksiltaan seuraavan kaltainen teksti:

```
function [output_parameter_list] = function_name(input_parameter_list)
                        TAI
function function_name(input_parameter_list)
                        TAI
function function_name()
```

- Määrittely alkaa aina avainsanalla `function`.
- Funktion mahdolliset paluuarvot annetaan hakasuluissa `[ ]`. Paluuarvo voidaan jättää pois jos funktio ei palauta arvoa.
- Jos funktio ei palauta arvoa, jätetään myös ”=”-merkki pois.
- Funktiota kutsutaan sen **nimellä**. Nimi on oltava sama kuin m-tiedoston nimi !
- Lopuksi annetaan suluissa lista funktiolle syötettävistä parametreista. Jos parametreja ei ole, kirjoitetaan tyhjät sulut.
- Parametrilistat erotellaan pilkulla
- Toisin kuin monissa muissa ohjelmointikielissä, **funktio ei saa muuttaa syöttöparametrilistassa funktiolle välitettävien muuttujien arvoja.**
- Return-lause ei ole pakollinen. Funktiosta palataan kutsuvaan ohjelmaan kun se loppuu. Return-lausetta voi käyttää jos halutaan esimerkiksi poistua funktiosta keskeltä koodia jonkin ehdon tarkistuksen seurauksena.

Lisäksi on syytä huomata, että:

- Matlabissa **ei ole** C.n void-tietotyyppiä
- Matlabissa **ei ole** C.n const-tietotyyppiä. Ohjelmoijan on itse huolehdittava siitä, että tietoja joita ei saa muuttaa, ei muuteta funktiossa.
- Matlabissa **ei ole** aaltosulkeita lohkoerottimina. Funktion suoritus loppuu kun tiedostossa ei ole enää suoritettavia rivejä tai kun vastaan tulee `return`-lause.
- Funktiolle syötettävien parametrien tietotyyppejä ei tarvitse määritellä etukäteen. Ei myöskään paluuarvon tyyppejä. Ne voivat vaihdella funktion kutsusta toiseen.
- Kaikki luvut ovat automaattisesti tyyppiä `double`, merkit ja merkkijonot tyyppiä `char`.
- Funktion sisäiset muuttujat ovat **paikallisia**. Ne eivät näy kutsuvaan ohjelmaan.

Esimerkki: Seuraavassa on esitetty yksinkertainen funktio `addtwo.m`, joka ottaa syötteekseen kaksi muuttujaa (skalaari, vektori, matriisi, merkkijono) ja laskee niiden summan.

```
function addtwo(x,y)
%  addtwo(x,y)  Adds two numbers, vectors, whatever, and
%               print the result = x + y
x+y
```

Funktiolla ei ole paluuarvoa, joten hakasulkuja ja =-operaattoria ei tarvita. Summalausekkeen lopussa ei ole puolipistettä, joten summa tulostuu komentoriville. Kutsutaan funktiota eri tietotyypeillä:

```
>> a=1;b=2;
>> addtwo(a,b)
ans =
     3
>> addtwo('a','b')
ans =
   195
>> addtwo([1 3 4],[5 0 3])
ans =
     6     3     7
>> addtwo('yin','yan')
ans =
   242   202   220
```

Esimerkki: Muutetaan funktiota seuraavaksi niin, että se palauttaa arvon:

```
function [z]=addtwo(x,y)
   %  addtwo(x,y)  Adds two numbers, vectors, whatever, and
   %               print the result = x + y
   z=x+y;
```

Kutsutaan funktiota komentoriviltä. Tyhjennetään ensin muuttujat ja katsotaan whos-komennolla ajon jälkeen, mitä muuttujia meillä on:

```
>> clear
>> tulos=addtwo('yin','yan');
>> whos
  Name        Size                     Bytes  Class
  tulos       1x3                         24  double array
Grand total is 3 elements using 24 bytes
```

Siis paikalliset muuttujat x ja y eivät näy komentoriville (sama periaatehan oli C-kielen arvoparametrillisissa funktioissa).

Tehtävä: Tee funktio `[tulos]=seto(x,y)`, joka laskee parametreina syötettyjen lukujen summan, erotuksen, tulon ja osamäärän ja palauttaa laskutoimitusten tulokset vektorissa `tulos`.

Tehtävä: Tee funktio `[r,fii]=sum2phs(x,y)` joka muuntaa syötetyn kompleksiluvun summamuodosta osoitinmuotoon. Tässä vaiheessa ei tarvitse huolehtia syöttötietojen oikeellisuudesta (merkkijonoa esimerkiksi ei voi muuntaa ☺)

Tehtävä: Tee funktio `[x,y]=phs2sum(r,fii)`, joka muuntaa syötetyn kompleksiluvun osoitinmuodosta summamuotoon.

Tehtävä: Tee funktio `[kk,vakio]=pns(tiedostonimi)`, joka lukee datan tiedostosta `tiedostonimi`, tekee dataan pienimmän neliösumman sovituksen ja tulostaa sekä itse datan että sovitetun suoran kuvaajat. Funktio palauttaa kutsuvaan ohjelmaan sovitetun suoran kulmakertoimen ja vakiotermin. Tiedostomuodosta oletetaan, että ensimmäinen rivi on otsikko, sen jälkeen on tyhjä rivi, jonka jälkeen seuraa data kahdessa sarakkeessa (vrt. jousivakion määritys ja `spring.txt`).
`VIHJE 1:data= textscan(fid,'%f %f',-1,'headerlines',2);`
`VIHJE 2:[p,s]=polyfit(x,F,1);`
`VIHJE 3:`:kk=p(1);vakio=p(2);

Ehtolause

Matlabissa on if-lause, jonka avulla voidaan valita joko kahdesta tai useammasta vaihtoehdosta. Seuraavat if-lauseen variantit ovat käytössä:

- `if ... end`
- `if ... else ... end`
- `if ... elseif ... else ... end`

Esimerkki: Yksinkertainen varoitus:

```
d = b^2 - 4*a*c;
if d<0
  disp('Juuret ovat imaginäärisiä !');
end
```

Esimerkki: Valinta useasta vaihtoehdosta:

```
d = b^2 - 4*a*c;
if d<0
  disp('Juuret ovat imaginäärisiä !');
elseif d==0
  disp('Yksi reaalinen juuri');
else
  disp('Kaksi reaalista juurta');
end
```

Muutamia huomioita:

- Ehtolauseen ympärillä ei tarvitse olla sulkuja (mutta ne saa laittaa)
- `elseif`-sana kirjoitetaan yhteen
- `end`-lause on pakollinen
- Käyttäjälle voidaan tulostaa viestejä kommenolla `disp`

If-lauseen kompakti versio voidaan kirjoittaa yhdelle riville:

```
        if x<0, disp('negatiivinen'), end
```

Tässä versiossa pilkku toimii erottimena, joka kertoo mihin if loppuu ja suoritettavat lauseet alkavat.

Tietotyypin tarkistamiseen Matlabissa on käytössä `isa`-funktiot, jotka palauttavat totuusarvon. Näitä voidaan käyttää varmistamaan, että funktiolle on syötetty oikeaa tietoa. Esimerkiksi vuosi ei voi olla liukuluku jne

```
isnumeric, islogical, ischar, iscell, isstruct, isfloat,
isinteger, isobject, isjava, issparse, isreal.
```

```
>> s='merkkejä';
>> if (ischar(s)), disp('On merkkijono');else disp('Ei ole
merkkijono');end
On merkkijono
>> s=45;
>> if (ischar(s)), disp('On merkkijono');else disp('Ei ole
merkkijono');end
Ei ole merkkijono
```

Tehtävä: Karkausvuosi tarkoittaa (paitsi joillekin hamekankaita ja kosintalupia) että helmikuussa on 29 päivää, jolloin myös koko vuonna on 366 päivää tavallisen 365:n sijasta. Vuosi on karkausvuosi, jos se on jaollinen neljällä, mutta ei 100:lla. Poikkeuksena kuitenkin on vuosiluku, joka on jaollinen 400:lla, jolloin vuosi siis on karkausvuosi. Esim. vuosi 1900 ei ollut karkausvuosi kun taas vuosi 2000 oli.

Kirjoita funktio `[logical(v)]=karkausvuosi(vuosi)`, joka tutkii onko käyttäjän syöttämä vuosi karkausvuosi. Funktio palauttaa vastauksen totuusarvona (0 / 1) kutsuvaan ohjelmaan.

Tehtävä: Lisää karkausvuosifunktioon ehto, joka tarkistaa, onko syötetty vuosiluku kokonaisluku.

Toistolauseet

While-silmukka

While-lausetta toistetaan niin kauan kuin ehto on tosi.

```
while ehto
          lauseita
end
```

Esimerkki: Arvotaan satunnaislukuja vektoriin väliltä 0.3 – 0.7.

```
function [luku]=arvonta()
i=1;
luku(i)=rand;
while (luku(i) > 0.3 && luku(i) <0.7)
    i=i+1;
    luku(i)=rand
end
```

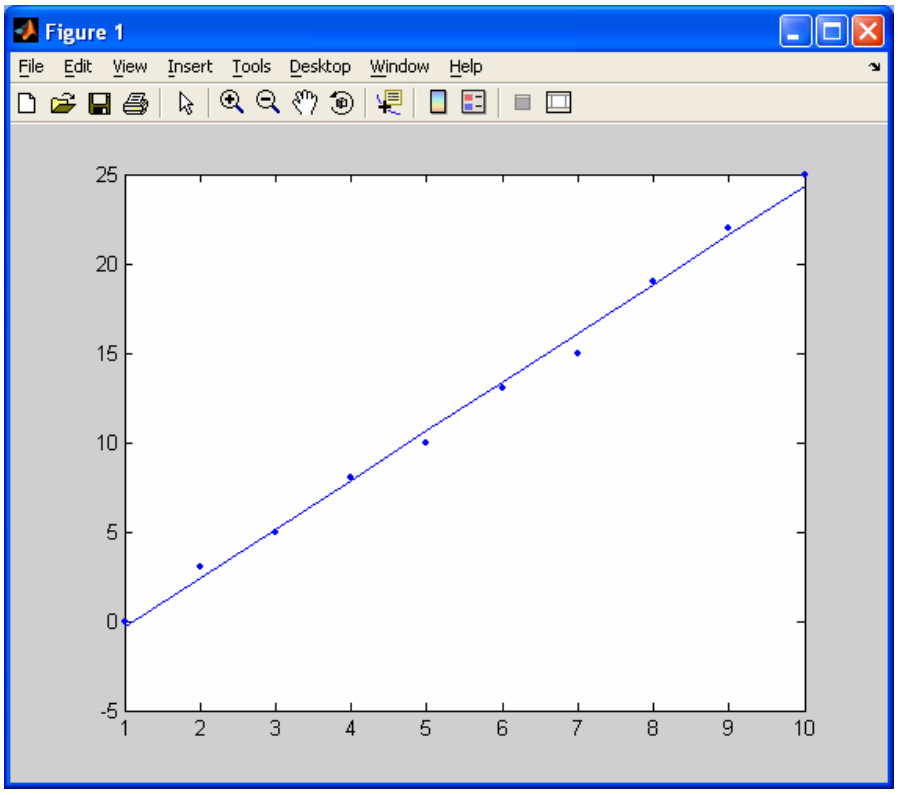
Esimerkki: Luetaan näppäimistöltä 10 lukua vektoriin y ja sovitetaan arvoihin pienimmän neliösumman suora (regressiosuora eli trendiviiva) ja tulostetaan luetut arvot sekä sovitettu suora. Funktio palauttaa sovituksen kulmakertoimen, vakiotermin ja sovituksen virheen (normalized residual) kutsuvaan ohjelmaan.

```
function [kk,vakio,virhe]=fit1()
i=1;
while (i<=10)
    y(i)=input('Luku ?')
    x(i)=i;
    i=i+1;
end
plot(x,y,'.')
[p,s]=polyfit(x,y,1);
hold on
kk=p(1);
vakio=p(2);
virhe=s.normr;
%Suoran yhtälö: y=kx+b
plot(x,kk*x+vakio)
disp('Suoran yhtälö: y=kx+b')
str=sprintf('Kulmakerroin = %f',kk);
disp(str)
str=sprintf('Vakio = %f',vakio);
disp(str)
str=sprintf('Sovituksen virhe = %f', virhe);
disp(str)
```

Ohjelman kutsuminen ja suoritus:

```
>> [kk,vakio,virhe]=fit1();
Luku ?0
y =
     0
Luku ?3
y =
     0     3
Luku ?5
y =
     0     3     5
```

```
Luku ?8
y =
     0     3     5     8
Luku ?10
y =
     0     3     5     8    10
Luku ?13
y =
     0     3     5     8    10    13
Luku ?15
y =
     0     3     5     8    10    13    15
Luku ?19
y =
     0     3     5     8    10    13    15    19
Luku ?22
y =
     0     3     5     8    10    13    15    19    22
Luku ?25
y =
     0     3     5     8    10    13    15    19    22    25
Suoran yhtälö: y=kx+b
Kulmakerroin = 2.739394
Vakio = -3.066667
Sovituksen virhe = 1.702049
```



Tässä esimerkissä tuli muutama huomionarvoinen asia esille:

- Komento `input` periaatteessa yhdistää C:stä tutun `printf`:n ja `scanf`:n (tai C++:n `cout` ja `cin`) yhdeksi. Se tulostaa käyttäjälle merkkijonon ja palauttaa luetun arvon muuttujaan (tässä `luku`)
- Vektoreiden `x` ja `y` koko kasvaa sitä mukaa kun uusia lukuja luetaan. Toisin kuin C:ssä, niiden kokoa ei tarvitse määritellä. Tässä esimerkissä vektorin `y`:n tulostus ei ole välttämätöntä, se on laitettu vain asian havainnollistamiseksi.
- `polyfit` sovittaa suoran eli 1. asteen polynomin havaintoarvoihin ja palauttaa kertoimet vektorissa `p`. Nyt kertoimia on vain kaksi, kulmakerroin ja vakiotermi. Tietue `s` sisältää sovituksen tarkkuuteen liittyviä suureita.
- `disp` tulostaa tekstiä komentoikkunaan.
- `sprintf` tulostaa muotoiltua tekstiä merkkijonomuuttujaan. Sen avulla lukuarvot saadaan muutettua merkkijonoiksi, jotka voidaan tulostaa `disp`:llä

Tehtävä: Kirjoita funktio, joka laskee 100 ensimmäisen positiivisen kokonaisluvun summan käyttäen while-silmukkaa. Tarkista tulos Matlabin sum-funktiolla.

Tehtävä: Matlabin funktio `realmin('double')` palauttaa järjestelmän pienimmän kaksoistarkkuuden liukuluvun. Kirjoita Matlab-koodi, joka tutkii, kuinka monta kertaa luku 1 voidaan jakaa kahdella ennen kuin jakolaskun tulos menee pienemmäksi kuin kyseinen luku.

For-silmukka

For-silmukkaa käytetään kun toistojen lukumäärä tiedetään. Syntaksi:

```
for laskuri=alkuarvo:askel:loppuarvo
     lauseita
end
```

Esimerkki: Tulostetaan luvut 1:stä 10:een for-silmukassa:

```
>>for i=1:10,i,end
```

For-silmukan käyttöä Matlabissa ei suositella. Suurimman osan sillä tehtävistä toimenpiteistä pystyy tekemään suoralla vektorin indeksoinnilla. Esimerkiksi 10 kokonaislukualkion tulostus voidaan Matlabissa tehdä yksinkertaisesti vektorin indeksoinnilla:

```
>> 1:10
>> [1:10]
>>(1:10)
```

Esimerkki: Täytetään 100 000 alkion mittainen vektori for-silmukassa ja suoralla indeksoinnilla.

```
>> t0=clock,for i=1:100000,x(i)=i;end;etime(clock,t0)
ans =
   40.3900
>> t0=clock,y=[1:100000];etime(clock,t0)
ans =
    0.0160
```

Siis for-silmukalla vektorin täyttö kestää yli 40 **sekuntia**. Saman asian tekemiseen suoralla indeksoinnilla menee 16 **millisekuntia** ! Selitys tälle karmealle erolle on, että käytettäessä for-silmukkaa ohjelma varaa jokaisen muistipaikan erikseen. Muistipaikat eivät edes välttämättä sijoitu peräkkäin muistissa. Indeksointiin perustuvassa ratkaisussa koko muistialue varataan yhdellä kertaa ja on taattu että muisti on varattu yhtenä kokonaisuutena.

Tehtävä: Laske 100 ensimmäisen positiivisen kokonaisluvun summa käyttäen for-silmukkaa.

Tehtävä: Piirrä funktioiden `sin(x), sin(2x), sin(3x)` ja `sin(4x)` kuvaajat väliltä $0 – 2\pi$ yhteen figure-ikkunaan käyttäen for-silmukkaa.