Začetni tečaj MATLAB

Avtorja: Poldi Herman, univ. dipl. inž. el.

Andraž Žertek, univ. dipl. inž. el.

April 2007

Vsebina

1. Uvod

Program Matlab je izdelek podjetja MathWorks, Inc. (http://www.mathworks.com/). Matlab je moderno programsko orodje za numerično reševanje problemov. Razvijati so ga pričeli v univerzitetnem okolju in je šele kasneje postal širše dostopen komercialni produkt. Program je napisan tako, da omogoča enostavno izvajanje matričnih operacij, reševanje diferencialnih enačb z numerično integracijo, grafični prikaz rezultatov vključno z animacijami. Osnovni element je matrika; skalarji in vektorji so poseben primer matrik. Osnovna verzija Matlaba zna računati samo numerično. V tako imenovanih m-datotekah lahko v programskem jeziku, ki je zelo podoben progamskemu jeziku Basic, pregledno in hitro zapišemo lasten program. Vsako tako napisano datoteko lahko obravnavamo kot novo funkcijo in jo uporabljamo na enak način kot že vgrajene funkcije. Z združevanjem m-datotek (funkcij) lahko vsakdo ustvari obsežno orodje za namensko uporabo. Tako se je skozi leta (začetki segajo v 70. leta 20. stoletja) od prvih verzij do današnje razvilo veliko število knjižnih funkcij (toolbox), ki so namenjene uporabi na specifičnih področjih.

Osnovni sistem Matlaba lahko razdelimo sledeče:

- **Razvojno okolje**

Nabor orodij in okolij, ki omogočajo uporabo Matlabovih funkcij in datotek.

- **Matlabova matematična funkcijska knjižnica**

Obsežna zbirka računskih algoritmov, ki segajo od enostavnih do zelo kompleksnih.

- **Matlab programski jezik**

Višjenivojski matrični jezik, ki vsebuje funkcije, podatkovne structure in elemente vhodno-izhodnega in objektno orientiranega programiranja.

- **Grafika**

Predstavitev vektorjev in matrik v obliki dvo- in tridimenzionalnih diagramov.

- **Aplikacijski vmesnik (API)**

Omogoča pisanje programov v c-ju in fortranu.

Orodja (toolbox)

Poleg osnovne verzije Matlab-a so na razpolago tudi orodja ('toolbox'), ki jih lahko namestimo poleg osnovnega programa. Obstaja množica orodij iz različnih področij npr.

- orodje za regulacijsko tehniko (Control system),
- orodje za računanje pretokov moči EES (Matpower)
- orodje za numerično identifikacijo sistema (System identification),
- orodje za mehko logiko (Fuzzy logic),
- orodje za simbolično reševanje enačb (Symbolic Math),
- orodje za statistiko (Statistic),
- orodje za delo v realnem času (Real time toolbox),
- orodje za izgradnjo grafičnega vmesnika (GUI) in ostala.

Pogosto najdemo na internetu tudi orodja, ki so jih izdelali posamezniki, kot je na primer orodje za robotiko, Robotics toolbox, (http://www.cat.csiro.au/cmst/staff/pic/robot/) in so v prostem dostopu.
Orodja vsebujejo množico funkcij, ki jih lahko uporabimo v svojih programih.

Pomoč

V Matlab-u je tako za osnovni program, kot za orodja, na voljo obsežna pomoč v obliki *.pdf dokumentov ali spletnih strani. Zraven opisa uporabe razpoložljivih funkcij so običajno podane tudi osnove pripadajoče teorije. Tako se lahko npr. osnov mehke logike in uporabe funkcij v ustreznem Matlab orodju naučimo s pomočjo pomoči za to orodje.

Pomoč v delovnem oknu

- **Splošna pomoč**

V delovnem oknu lahko do pomoči dostopamo tako, da enostavno vtipkamo ukaz *help*. To je najhitrejša in najpomembnejša oblika pomoči. Izpiše se seznam vseh glavnih funckij in vseh nameščenih orodij. Poiščemo kaj nas zanima in ukaz help razširimo v *help topics* .

- **Pomoč pri uporabi ukaza**

Če želimo uporabiti nek ukaz, toda ne vemo kako se ukaz uporabi, vpišemo v delovni prostor ***help ukaz***. Npr. če želimo izvedeti kako s uporabimo ukaz *sin* vtipkamo:

```
help sin

SIN Sine.
SIN(X) is the sine of the elements of X.
```

- **Pomoč pri iskanju ukaza**

Če iščemo nek ukaz toda ne vemo kakšen je, uporabimo v delovnem prostoru iskanje z ***lookfor ključna_besed .*** Če nas npr. zanima kako izračunamo hiperbolični sinus uporabimo npr. ukaz ***lookfor hyperbolic*** . Matlab izpiše seznam vseh funkcij, ki se nanašajo na to ključno besedo.

```
lookfor hyperbolic

ACOSH  Inverse hyperbolic cosine.
ACOTH  Inverse hyperbolic cotangent.
ACSCH  Inverse hyperbolic cosecant.
ASECH  Inverse hyperbolic secant.
ASINH  Inverse hyperbolic sine.
ATANH  Inverse hyperbolic tangent.
COSH   Hyperbolic cosine.
COTH   Hyperbolic cotangent.
CSCH   Hyperbolic cosecant.
SECH   Hyperbolic secant.
SINH   Hyperbolic sine.
TANH   Hyperbolic tangent.
.
.
.
```

- **Pomoč v obliki html strani**

Do pomoči dostopamo tako, da v Matlabovem oknu kliknemo ikoni z vprašajem. Odpre se osnovna stran pomoči, kjer lahko iščemo po indeksu, po ključnih besedah in neposredno po vsebini.

Slika 1: Ikona help.

- **Pomoč v obliki pdf dokumenta**

Na voljo je v direktoriju Matlabov direktorij/Help/pdf_doc.

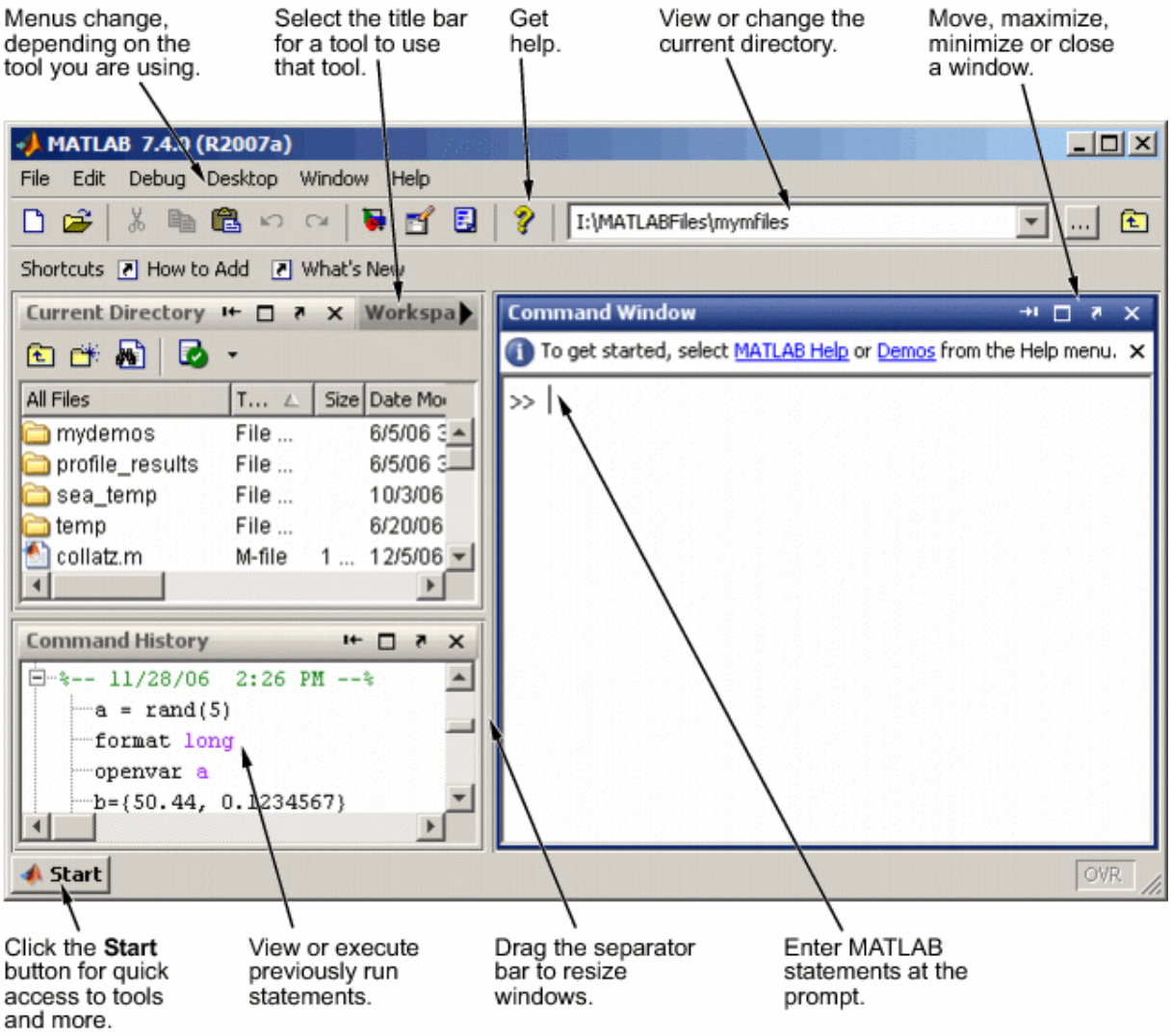
- **Demonstracijski (demo) programi**

Kot dodatek pomoči so Matlab-u dodani demo programi, ki nazorno prikazujejo možnosti, ki jih Matlab ponuja. Ob dodatku vsakega novega orodja se namesti tudi pripadajoči demo program. Demo zaženemo z ukazom ***demo*** v delovnem oknu Matlaba. Odpre se okno kjer

izberemo področje, ki nas zanima. Za vsako področje so prikazani razpoložljivi ukazi, ter način in rezultati uporabe teh ukazov.
Demo si je priporočljivo ogledati preden začnemo z resnim delom z Matlab-om saj nam nudi pregled možnosti, ki jih program ponuja.

Kako pognati program Matlab

Matlab poženemo s klikom na ikono Matlab-a na omizju ali preko razdelka Programi.
Matlab ima ob zagonu lahko različen izgled in ga sestavljajo eno ali več oken. Matlab namreč omogoča da ga priredimo po lastnih potrebah in nato se izbrana nastavitev ohrani tudi ob naslednjem zagonu. Ob zagonu se zato lahko odpre eno ali več oken. Najpogosteje so to okna Command Window, Launch Pad in Command History lahko pa tudi okna Current Directory,Workspace in Help.
Osnovno nastavitev prikličemo s klikom v meniju *View/Desktop Layout/Default*. Novo nastavitev izberemeo tako da v meniju View označimo vsa okna, ki jih želimo imeti odprta.



Slika 2: Osnovna nastavitev oken.

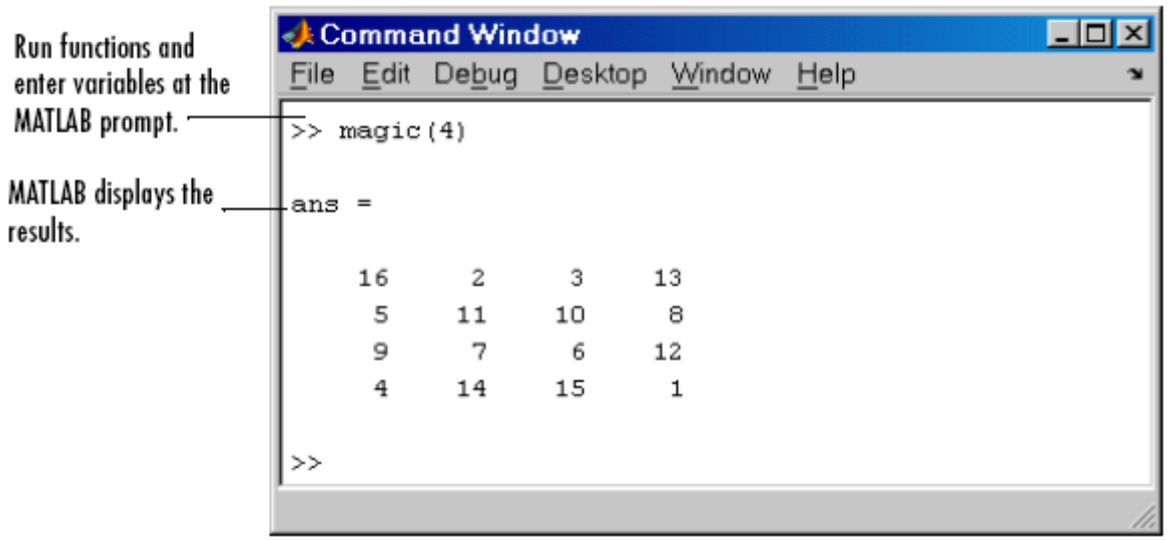
Oglejmo si malce bolj podrobno posamezna okna:

Tabela 1: Matlab okna.

Desktop Tool	**Kratek opis**
Array Editor	Preglejte vsebino delovnega prostora v tabelaričnem formatu.
Command History	Preglejte ukaze, ki ste jih vpisali v Command Window (copy, paste, run, ...).
Command Window	Izvršite MATLAB ukaze.
Current Directory Browser	Preglejte datoteke.
Editor/Debugger	Create, edit, debug, analyze M-files (datoteke, ki vsebujejo MATLAB ukaze).
Figures	Create, modify, view, print MATLAB slike.
File Comparisons	Preglejte vrstico-za-vrstico kodo v dveh dokumentih.
Help Browser	Poiščite in preglejte dokumentacijo za vse MathWork produkte.
Profiler	Izboljšajte učinkovitost vaših M-datotek.
Start Button	Zaženite orodja.
Web Browser	Preglejte HTML in podobne datoteke, ki jih ponuja MATLAB.
Workspace Browser	Preglejte in naredite spremembe v delovnem prostoru.

- **Command Window**

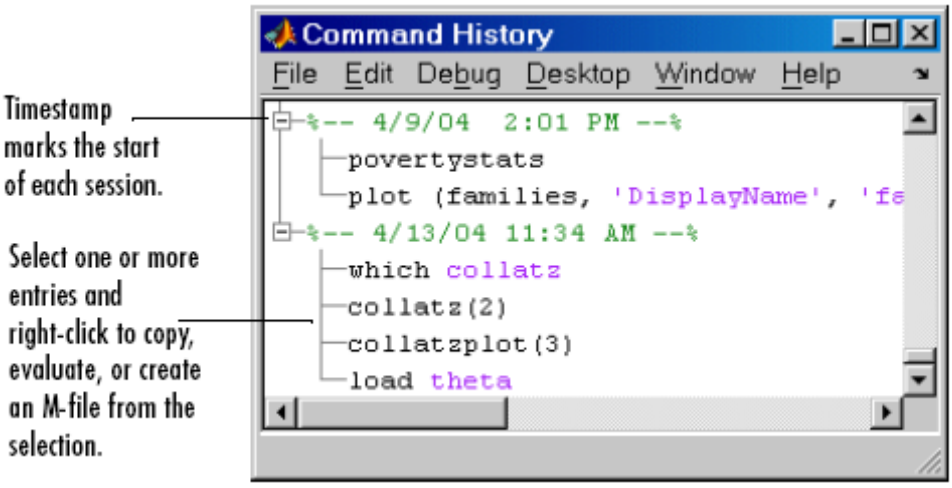
Je glavno okno Matlaba v katerem lahko neposredno vpisujemo ukaze. Delo v njem pa je priporočljivo **le za reševanje enostavnih problemov** in za hitre izračune. Za ponoven izračun, toda na primer z drugačnimi vhodnimi podatki, je namreč potrebno vse ukaze še enkrat napisati.



Slika 3: Okno Command Window

- **Command History**

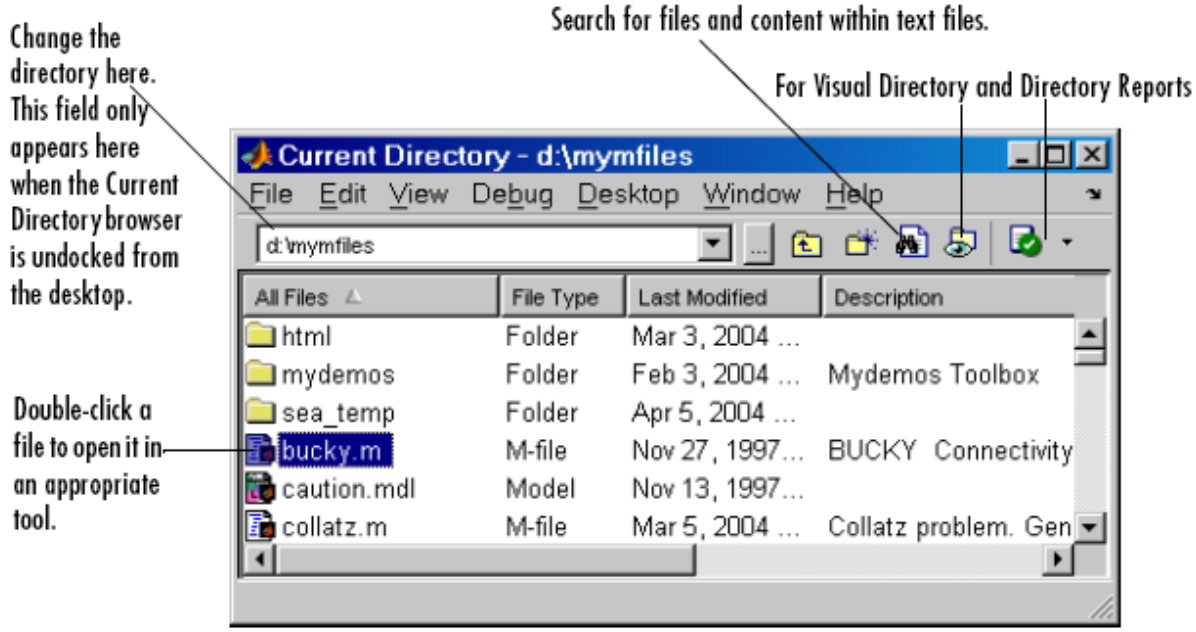
Prikazuje staro vsebino Command Window.



Slika 4: Okno Command history

- **Current Directory**

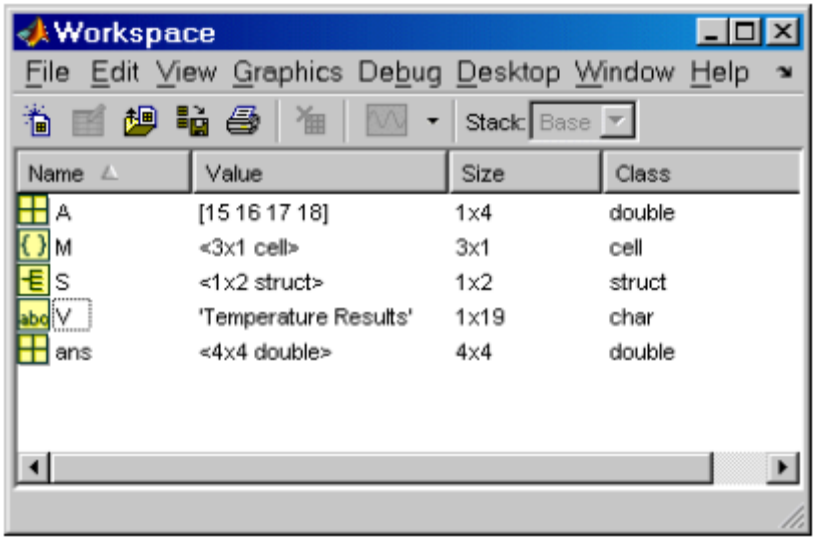
prikaže delovni direktorij s seznamom vseh *.m datotek in *.mdl datotek (Simulink sheme), ki se nahajajo v njem.



Slika 5: Okno Current Directory

- **Workspace**

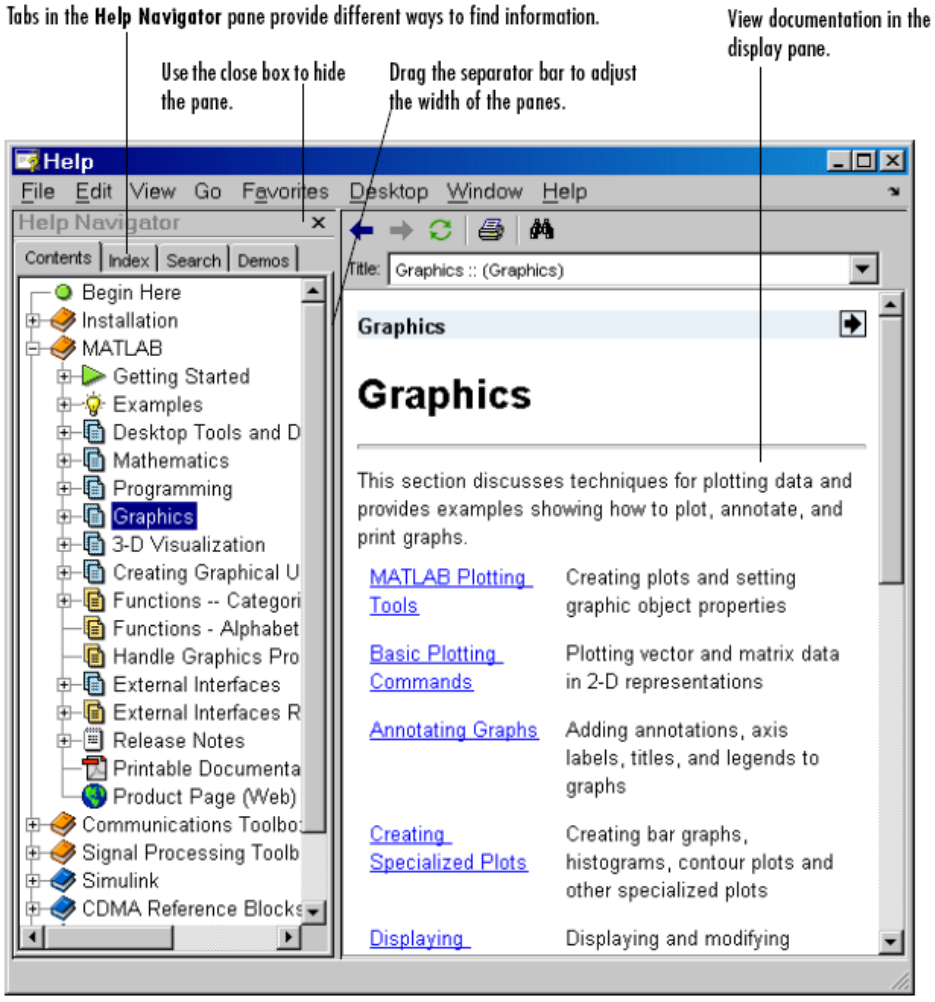
Sezam, tip in dimenzija vseh spremenljiv definiranih v delovnem prostoru.



Slika 6: Okno Workspace.

- **Help**

Je Matlabovo okno s pomočjo.



Slika 7: Okno Help.

M-datoteke

So datoteke v katerih so lahko zapisani uporabniški programi in že izdelani programi (v okviru osnovnega Matlab-a ali njegovih orodij). V M-datoteko torej lahko zapišemo program. Za razliko od neposrednega dela v ukaznem oknu lahko program zapisan v M-datoteki poljubnokrat modificiramo in izvajamo. Pri tem je potrebno uporabljati Matlab-ov programski jezik. Iz m-datoteke lahko kličemo druge M-datoteke.

- **Kako odpreti m-datoteko?**

Novo M-datoteko odpremo v meniju File/New/M-file ali s klikom na ustrezno ikono v orodni vrstici. Odpre se novo okno z Matlab-ovim urejevalnikom besedil v katerega napišemo program. Urejevalniku je dodan tudi razhroščevalnik (Debugger). Za pisanje m-datotek pa lahko uporabimo tudi katerikoli drug editor kot npr. Notepad. Matlab-ov editor lahko odpremo tudi brez zagona Matlab-a.
Že obstoječo datoteko odpremo tako da v Matlab-ovem oknu v meniju izberemo File/Open in nato poiščemo iskano datoteko. Datoteko lahko odpremo tudi s klikom na ustrezno ikono v orodni vrstici.

- **Ime in shranjevanje m-datoteke**

M-datoteko shranimo s končnico *.m*. Če delamo z Matlab-ovim urejevalnikom se ta končnica doda avtomatsko, če pa uporabljamo kakšen drug urejevalnik pa jo je potrebno dodati. Ime datoteke ne sme vsebovati šumnikov in presledkov.

- **Zagon m-datoteke**

Program zapisan v M-datoteki poženemo tako, da napišemo njegovo ime v delovnem oknu in pritisnemo enter. Če smo v programu omogočili izpis rezultatov (to pomeni da se ustrezne vrstice ne zaključijo s podpičjem) vidimo rezultate v delovnem oknu. Program lahko poženemo tudi preko ikone Run, ki se nahaja v orodni vrstici urejevalnika besedil, kjer smo pisali program.

2. Vektorji in matrike

Vpis skalarjev

- **Prireditev vrednosti**

Skalarju priredimo vrednost z npr. *a=10000* . **Decimalna števila se vpisujejo s piko.**

Matlab:
```
>>a=10000

a =

10000

>> b=4.1

b =

4.1000
```

Prireditev vrednosti števila z desetiško potenco
$23{,}123\times10^{-20}$ vpišemo z:

Matlab:
```
>> 23.123*1e-20

ans =

 2.3123e-019
```

- **Najmanjše realno število**

Izpiše se z ukazom ***realmin***.

Matlab:
```
>> realmin

ans =

  2.2251e-308
```

- **Točnost operacij s plavajočo vejico**

Izpiše se z ukazom ***eps***.

```
Matlab:
>> eps

ans =

  2.2204e-016

>> 1-4*eps

ans =

    1.0000

>> a=1+eps

a =

    1.0000
```

```
Matlab:
>> a-1

ans =

  2.2204e-016

>>
a=1+eps/2

a =

     1

>> a-1

ans =

     0
```

- **Največje realno število**

Izpiše se z ukazom ***realmax***.

```
Matlab:
>> realmax

ans =

  1.7977e+308
```

Kompleksna števila

Matlab omogoča tudi delo s kompleksnimi števili. Kompleksno enoto označimo z **i ali j.** S kompleksnimi števili lahko računamo na enak način kot z realnimi števili.

Matlab:

```
>> a=2+4*j

a =

   2.0000 + 4.0000i

>> b=1-j*2

b =

   1.0000 - 2.0000i
```

```
>> a*b

ans =

    10

>> a^b

ans =

 -12.7930 -38.8923i
```

Operatorji in osnovne matematične funkcije

- **Aritmetični operatorji**

Tabela 2: Osnovni aritmetični operatorji.

+, -	seštevanje,odštevanje
*, /	množenje, deljenje
^	potenca
sqrt(x)	kvadratni koren

Matlab:

```
>> a=15;
>> b=25;
>> a+b

ans =

40
```

```
>> a/b

ans =

0.6000

>> sqrt(a/b)

ans =

0.7746
```

Če želimo samo izračunati nek izraz, lahko Matlab uporabljamo na podoben način kot kalkulator. Za določitev vrstnega reda izračunov uporabljamo oklepaje:

Matlab:

```
>> 12.34/5.67*(8e-9*sin(0.5))

ans =

  8.3472e-009
```

Osnovne matematične funkcije

- **Eksponent, logaritemske funkcije, absolutna vrednost**

Tabela 3: Osnovne matematične funkcije (exp, abs, log).

exp(x)	Eksponentna funkcija e^x
log(x)	Naravni logaritem
log10(x)	Desetiški logaritem
abs(x)	Absolutna vrednost

Matlab:
```
>> exp(10)

ans =

  2.2026e+004

>> log(ans)

ans =

   10
```

Matlab:
```
>> log10(10)

ans =

    1

>> abs(-100)

ans =

   100
```

- **Trigonometrične funkcije**

Argumenti (koti) se podajajo **v radianih**. Vrednost konstante π je vgrajena in jo lahko uporabljamo v izrazih:

Tabela 4: Osnovne matematične funkcije (trigonometrične).

cos (x)	Kosinus kota
sin(x)	Sinus kota
tan (x)	Tangens kota
acos(x)	Inverzni kosinus
asin(x)	Inverzni sinus
atan(x)	Inverzni tangens
atan2(y,x)	Inverzni tangens, štirikvadranten
sinh(x)	Hiperbilični sinus
cosh(x)	Hiperbolični kosinus
tanh(x)	Hiperbolični tangens

Matlab:
```
>> cos(pi)

ans =

   -1

>> tan(-1)

ans =

  -1.5574
```

- **Logični operatorji**

Se uporabljajo predvsem kot argumenti v pogojnih stavkih. Primerjamo lahko le matrike enake velikosti. Primerjanje (logična operacija) poteka po komponentah.

Tabela 5: Logični operatorji.

~	negacija
~=	ni enako
= =	ekvivalentno
<	manjše
<=	manjše ali enako
>	večje
>=	večje ali enako
&	logični in
|	logični ali
~	komplement

Podobno lahko uporabljamo tudi naslednje ukaze

Tabela 6: Funkcije za logične operatorje.

and(a,b)	logični in
or(a,b)	logični or
xor(a,b)	ekskluzivni ali

Opomba (!):
Vse navedene funkcije delujejo tako za realna kot tudi za kompleksna števila. Seveda so operacije za kompleksna števila definirane drugače. Npr. ukaz abs za realno število vrne pozitivno vrednost, za kompleksno pa oddaljenost od koordinatnega izhodišča.

Matlab:
```
>> a=1;
>> b=0;
>> a&b
ans =
     0
>> d=~a
d =
     0
```

Matlab:
```
>> a|b
ans =
     1
>> and(1,0)
ans =
     0
```

Zaokroževanje

Za zaokroževanje rezultatov uporabljamo ukaze ***fix, floor, ceil, round.***

Tabela 7: Načini zaokroževanja.

fix(x)	*Z* aokrožitev navzdol na celo število
round(x)	Zaokrožitev na najbližje celo število
floor(x)	Zaokrožitev na najbližje celo število, proti minus neskončno
ceil (x)	Zaokrožitev na najbližje celo število, proti neskončnosti

Format izpisa

Tabela 8: Tipi farmatov izpisa.

format short	Izpis na 5 decimalk.
format long	Izpis na 15 decimalk.
format short e	Izpis potence na 5 decimalk.
format long e	Izpis potence na 15 decimalk.
format +	Izpis predznaka.
format	Enako kot format short.
format compact	Zgosti izpis.
format loose	Dodatna vrstica v izpisu.

Matlab:

```
>> format short e
>> 10/3

ans =

  3.3333e+000

>> format compact
>> 10/3
ans =
  3.3333e+000

>> format loose
>> 10/3

ans =

  3.3333e+000
```

Matlab:

```
>> format short
>> 10/3

ans =

    3.3333

>> format long
>> 10/3

ans =

   3.33333333333333
```

Definicija vektorjev in matrik

Tabela 9: Sintaksa za vpis vektorjev in matrik.

x=[x1, x2, x3 ,..]	Vpis vrstičnega vektorja.
y=[y1; y2; y3; y4; ..]	Vpis stolpičnega vektorja
x=x_zacetni : korak : x_koncni	Avtomatsko generiranje vektorja
A = [A11, A12, A13,.. ; A21, A22, A23,.. ; A31, A32, A33,..;..]	Vpis matrike

- **Vpis vrstičnega vektorja**

Vrstični vektor vpišemo v oglatih oklepajih, člene vektorja med seboj ločimo z vejico ali presledkom ***x = [1,2,3]*** **.**

Matlab:
```
>> x=[1,2,3]

x =

    1    2    3
```

- **Vpis stolpičnega vektorja**

Stolpični vektor vpišemo v oglatih oklepajih, njegove člene pa ločimo s podpičji:

Matlab:
```
>> x=[4;3;2;1]

x =

    4
    3
    2
    1
```

- **Avtomatsko generiranje vektorja z enakomerno razporejenimi členi**

Vektor z enakomerno naraščujočimi/padajočimi členi generiramo na sledeči način:

Matlab:
```
>> x=1:0.2:2

x =

    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

- **Vpis matrik**

Matriko vpišemo tako, da vrstice ločimo s podpičjem, člene v vrstici pa z vejico ali presledkom ***A = [1,2,3;4,5,6;7,8,9].***

Matlab:
```
>> A = [1,2,3;4,5,6;7,8,9]

A =

    1    2    3
    4    5    6
    7    8    9
```

Manipulacije z matrikami in vektorji

Nekaj osnovnih pravil za manipulacije z vektorji in matrikami:

- Z vejico ločimo člene v stolpcu, z dvopičjem preidemo v novo vrstico.

- Pri delu z matrikami, oziroma členi matrik, se prvo število v oklepaju nanaša na vrstico, drugo na stolpec. *A(i,j)* tako pomeni i to vrstico in j-ti člen v vrstici

- Če delamo s celim stolpcem ali vrstico, nadomestimo številko člena z dvopičjem. *A(i,:)* tako pomeni i-to vrstico in je torej vrstični vektor, *A(:,j)* pomeni vse člene v j-tem stolpcu in je stolpični vektor.

- Če bomo priredili vrednost nekega člena matriki, ki še ni definirana, bodo imeli ostali členi matrike vrednost 0. Če npr. uporabimo ukaz C(3,4) = 10 in matrike C še ni, bo generirana matrika C dimenzije 3x4, ki bo imela člen C(3,4) = 10, vsi ostali členi pa bodo nič.

- Pri dodajanju členov matriki je potrebno paziti na dimenzijo. Tako lahko matriki dodamo le stolpec, ki ima toliko členov kot ima matrika vrstic ali vrstico, ki ima toliko členov kot ima matrika stolpcev.

Delo s členi vektorjev

Tabela 10: Manipulacije z vektorji.

xi=x(i)	Branje i-tega člena vektorja
x(i)=xi	Prirejanje vrednosti i-temu členu vektorja
x=[x, a1, a2, a3,..]	Dodajanje členov a1,a2,... vrstičnem vektorju
x=[x; a1; a2; a3,..]	Dodajanje členov a1,a2,... stolpičnem vektorju

- **Branje člena vektorja**

i-ti člen stolpičnega ali vrstičnega vektorja preberemo z ***xi=x(i).***

Matlab:

```
x =

    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000

>> x(4)

ans =

    1.6000
```

- **Prirejanje nove vrednosti členu vektorja**

i-temu členu stolpičnega ali vrstičnega vektorja priredimo novo vrednost z ***x(i)=xi .***

Matlab:

```
x =

     4
     3
     2
     1

>> x(2)=1000

x =

     4
  1000
     2
     1
```

- **Dodajanje členov vrstičnem vektorju**

Člene vrstičnemu vektorju dodamo z ukazom ***x=[x, a1, a2, ..]***

Matlab:

```
>> x=[1,2,3]

x =

    1    2    3

>> y=[x,10,20,30]

y =

    1    2    3   10   20   30
```

Dodajanje členov stolpičnem vektorju poteka analogno kot dodajanje členov vrstičnemu vektorju, le da elemente vektorja ločujemo s podpičji ***x=[x; a1; a2; ..].***

Delo s členi, vrsticami, stolpci in podmatrikami matrik

Tabela 11: Manipulacije z matrikami.

Aij=A(i,j)	Branje člena matrike
a=A(:,j)	Branje j-tega stolpca matrike
a=A(i,:)	Branje i-te vrstice matrike
B=A(i:j,k:l)	Branje podmatrike
A(i,j)=Aij	Prirejanje nove vrednosti členu matrike
A(:,i)=[Ai1; Ai2; Ai3;...]	Prirejanje nove vrednosti stolpcu matrike
A(i,:)=[A1i, A2i, A3i,...]	Prirejanje nove vrednosti vrstici matrike
A(:,j)=[]	Brisanje stolpca iz matrike
A(i,:)=[]	Brisanje vrstice iz matrike
C=[A,x]	Dodajanje stolpca matriki
C=[A;y]	Dodajanje vrstice matriki

- **Branje člena matrike**

Posamezne člene matrike beremo z npr. ***Aij=A(i,j)***. Prvi indeks označuje vrstico, drugi indeks člen v tej vrstici.

Matlab:

```
>> A=[1,2,3;4,5,6;7,8,9]

A =

    1    2    3
    4    5    6
    7    8    9
```

Matlab:

```
>> a22=A(2,2)

a22 =

    5
```

- **Branje stolpca matrike**

j-ti stolpec matrike A preberemo z ukazom *a=A(:,j).* a je stolpični vektor.

Matlab:

```
A =

     1     2     3
     4     5     6
     7     8     9

>> a=A(:,2)

a =

     2
     5
     8
```

- **Branje podmatrike**

Do podmatrike dostopamo tako da določimo kateri členi sestavljajo podmatriko.

Matlab:

```
A =

     1     2     3
     4     5     6
     7     8     9

>> a=A(2:3,1:2)

a =

     4     5
     7     8
```

- **Prirejanje vrednosti vrstici matrike**

i-to vrstico matike A spremenimo z ukazom A*(i,:)=[Ai1; Ai2; Ai3; ,..].*

Matlab:

```
A =

     1     2     3
     4     5     6
     7     8     9

>> A(2,:)=[40,50,60]

A =

     1     2     3
    40    50    60
     7     8     9
```

- **Brisanje stolpca matrike**

i-ti stolpec iz matrike A zbrišemo z *A(:,i)=[]* .

Matlab:

```
A =

     1     2     3
     4     5     6
     7     8     9

>> A(:,1)=[]

A =

     2     3
     5     6
     8     9
```

- **Dodajanje vrstice matriki**

Vrstico dodamo matriki *A* z ukazom *A=[A;x],* kjer je x vrstični vektor enake dolžine kot je število členov vrstice matrike *A* .

Matlab:

```
A =

     1     2     3
     4     5     6
     7     8     9

>> A(:,1)=[]

A =

     2     3
     5     6
     8     9
```

- **Dimenzije vektorjev in matrik**

Tabela 12: Dimenzije vektorje in matrik.

[M,N]=size(A)	Dimenzija vektorja ali matrike A
size(A,1)	Število vrstic
length(x)	Dolžina vektorja

Ukazi za dimenzijo spremenljivk se pogosto uporabljajo v funkcijah, ki preverjajo število podatkov v vhodnem vektrorju. Pri delu v delovnem prostoru pa si dimenzijo vseh definiranih spremenljivk lahko pogledamo tudi z ukazom ***whos***, ki izpiše vse spremenljivke skupaj z njihovo dimenzijo in tipom.

- **Ukaz size**

[M,N]=size(A) priredi spremenljivki M število vrstic in spremenljivki N število stolpcev. Podoben ukaz je *size(A,1)* , ki pa izpiše število vrstic.

Matlab:

```
B =
     1     2     3     4     5     6     7     8     9    10
     1     2     3     4     5     6     7     8     9    10
     1     2     3     4     5     6     7     8     9    10
     1     2     3     4     5     6     7     8     9    10
     1     2     3     4     5     6     7     8     9    10

>> [M,N]=size(B)

M =

     5

N =

    10
```

- **Ukaz length**

length(X) izpiše dolžino vektorja x.

Matlab:
```
>> x=1:1:100;
>> length(x)

ans =

   100
```

Elementarne matrike in vektorji

Nekatere elementarne matrike in vektorje lahko generiramo neposredno z ukazom.

Tabela 13: Ukazi za generiranje elementarnih vektorjev in matrik.

zeros(c,d)	Matrika ničel z c vrsticami in d stolpci
ones (c,d)	Matrika enic z c vrsticami in d stolpci
eye(c)	Enotina matrika z c vrsticami in stolpci
rand(c,d)	Matrika naključnih števil med 0 in 1
diag(a)	Diagonalna matrika s členi vektorja a v diagonali
linspace(x1,x2,N)	Diagonalna matrika s členi vektorja a v diagonali
magic(c)	Matrika dimenzije c x c, ki ima za vsoto vseh vrstic in stolpec enako.

- **Ukaz zeros**

Z ukazom *zeros(c,d)* generiramo matriko ali vektor ničel. Če podamo kot argument ukaza dve število pomeni prvo število (c) število vrstic, drugo (d) pa število stolpcev. Če podamo samo eno število se generira kvadratna matrika ničel s podanim številom stolpcev in vrstic.

Matlab:
```
>> A=zeros(5,5)

A =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

- **Ukaz ones**

Z ukazom *ones (c,d)* generiramo vektor/matriko enic.Če podamo kot argument ukaza dve število pomeni prvo število (c) število vrstic, drugo (d) pa število stolpcev. Če podamo samo eno število se generira kvadratna matrika enic s podanim številom stolpcev in vrstic.

- **Ukaz eye**

Z ukazom *eye(c)* generiramo enotino matriko dimenzije cxc. Ukaz lahko uporabimo tudi v obliki *eye(c,d)*, s tem da se v tem primeru generira matrika, kjer je podmatrika velikosti cxc enotina vrstica, dodanih pa je še d-c stolpcev ničel. Matematično to ni enotina matrika!

Matlab:

```
>> eye(5)

ans =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1

>> eye(5,2)

ans =

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

- **Ukaz rand**

Z ukazom *rand(c,d)* generiramo matriko/vektor naključnih števil med nič in ena. Če podamo kot argument ukaza dve število pomeni prvo število (c) število vrstic, drugo (d) pa število stolpcev. Če podamo samo eno število se generira kvadratna matrika enic s podanim številom stolpcev in vrstic. Območje ki ga zavzemajo nakjučne vrednosti lahko tudi ustrezno premaknemo in razšitrimo.

Matlab:

```
>> rand(3)

ans =

   0.0153   0.9318   0.8462
   0.7468   0.4660   0.5252
   0.4451            0.4186
0.2026>> 2*rand(3,1)+1

ans =
```

Matlab:

```
>> rand(3,1)

ans =

   0.6721
   0.8381
   0.0196
```

- **Ukaz diag**

Z ukazom *diag(a)* generiramo diagonalno matriko, ki ima v diagonali člene, kot so po vrsti navedeni v vektorju, ki ga podamo kot argument ukaza (a).

- **Ukaz linspace**

Ukaz *linspace(x1, x2, N)* generira vektor z N členi, ki so linearno razporejeni med vrednostima x1 in x2.

- **Ukaz magic**

Vsota vseh vrstic in stolpcev matrike dobljene z ukazom *magic(c)* je enaka. c je dimenzija matrike.

Matlab:

```
>> magic(3)

ans =

   8   1   6
   3   5   7
   4   9   2

>> sum(ans)

ans =

   15   15   15
```

Osnovne matematične operacije z vektorji in matrikami

Matlab omogoča dva tipa operacij med vektorji in matrikami. Prvi način je takšen kot ga poznamo z matematike, npr. množenje dveh matrik. Drug način pa je bolj splošen in omogoča tudi operacije po elementih, npr. množenje istoležnih členov dveh matrik. Ta možnost izhaja iz dejstva, da Matlab temelji na uporabi polj (array), matrike pa so samo posebna oblika teh polj s posebej definirami (matričnimi in vektorskimi) operacijami. Oba tipa operacij sta opisana spodaj.

Tabela 14: Matematične operacije nad vektorji in matrikami.

*A (+, -, *, /) a*	Prištevanje /odštevanje/množenje/deljenje vseh členov matrike ali vektorja s skalarjem.
*A*y*	Množenje matrike z vektorjem.
*A (+, -, *, /) B*	Prištevanje /odštevanje/množenje/deljenje matrike z matriko.
A (., . /) B*	Množenje/deljenje matrike z matriko po členih.
*a (+, -, *, /) b*	Prištevanje /odštevanje/množenje/deljenje vektorja z vektorjem.
*a (. *, . /) b*	Množenje/deljenje vektorja z vektorjem po členih.
A'	Transponiranje vektorja ali matike.

- **Prištevanje/odštevanje/množenje/deljenje matike ali vektorja s skalarjem**

Vsakemu členu matike ali vektorja lahko prištejemo/odštejemo vrednost skalarja *a* oziroma jih z *a* delimo/množimo z ukazi *A+a* , *A-a* , *A*a* , *A/a* .

Matlab:

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

>> A+10

ans =

   11   12   13
   14   15   16
   17   18   19

>> A/2

ans =

    0.5000    1.0000    1.5000
    2.0000    2.5000    3.0000
    3.5000    4.0000    4.5000
```

- **Množenje matrike z vektorjem**

Matriko A z vektorjem y pomnožimo z ukazom A*y. y mora biti stolpični vektor dimenzije, ki je enaka številu stolpcev matrike A.

- **Prištevanje/odštevanje/množenje/deljenje matrike z matriko**

Prišteti in odšteti je možno le matrike enakih dimenzij. Operacija se izvrši člen s členom. Pomnožimo (matrično množenje) lahko le matriki, kjer ima prva enako število stolpcev kot druga vrstic. Delimo lahko le matriki, ki imata enako število stolpcev.

Matlab:

```
A =

    1    2    3
    4    5    6

>> B=[9,8,;7,6;5,4]

B =

    9    8
    7    6
    5    4

>> A*B

ans =

   38   32
  101   86

>> A+B'

ans =

   10    9    8
   12   11   10

>> A/B'

ans =

   5.0000   -5.5000
   8.0000   -8.5000
```

- **Množenje/deljenje matrike z matriko po členih**

Operacijo lahko izvajamo samo na matrikah enakih dimenzij. Operacijo po členih označimo s piko pred operatorjem, npr. operator za množenje po členih je (.*). Zmnožijo/delijo se istoležni členi v matrikah (torej ne gre za množenje matrik, kot ga poznamo iz matematike).

Matlab:

```
A =

    1    2    3
    4    5    6

>> B=[9,8,;7,6;5,4]

B =

    9    8
    7    6
    5    4

>> A.*B'

ans =

    9   14   15
   32   30   24
```

Matlab:

```
>> A*B

ans =

   38   32
  101   86

>> A./B'

ans =

   0.1111   0.2857   0.6000
   0.5000   0.8333   1.5000

>> A/B'

ans =

   5.0000   -5.5000
   8.0000   -8.5000
```

- **Prištevanje /odštevanje/množenje/deljenje vektorja z vektorjem.**

Prišteti/odšteti/deliti je možno le vektorje enakih dimenzij. Pri tem se operacija izvede na istoležnih členih.

- **Množenje/deljenje vektorja z vektorjem po členih**

Operacijo po členih označimo s piko pred operatorjem, npr. operator za množenje po členih je (.*). Zmnožijo/delijo se istoležni členi v vektorjih (torej ne gre za množenje vektorjev, kot ga poznamo iz matematike).

- **Transponiranje**

Matriko oziroma vektor transponiramo z ukazom A', oziroma y'.

Vgrajene matrične operacije

Tabela 15: Vgrajene matrične operacije.

sum(x)	Vsota členov vektorja ali stolpca matrike
max(x)	Največji člen vektorja ali stolpca matrike
inv(A)	Inverzna matrika
rank(A)	Rank matrike
det(A)	Determinanta matrike
eig(A)	Lastne vrednosti matrike
poly(A)	Koeficienti karakterističnega polinoma
norm(A)	Norma matrike ali vektorja

- **Ukaz sum**

Če je argument ukaza vektor, z ukazom izračunamo vsoto členov vektorja. Če je argument ukaza matrika z ukazom izračunamo vsoto členov v njenih stolpcih.
Koliko je vsota prvih sto števil?

Matlab:
```
>> A=[1:100];
>> sum(A)
```

- **Ukaz max**

Če je argument ukaza vektor z ukazom določimo največji člen vektorja. Če je argument ukaza matrika z ukazom določimo največji člen v njenih stolpcih.

- **Ukaz inv**

Izračuna inverzno matriko.

- **Ukaz rank**

Določi rank matrike- število linearno neodvisnih vrstic oziroma stolpcev.

- **Ukaz det**

Izračuna determinanto matrike.

- **Ukaz eig**

Izračuna lastne vrednosti kvadratne matrike.

Matlab:

```
A =

    8    1    6
    3    5    7
    4    9    2

>> eig(A)

ans =

   15.0000
    4.8990
   -4.8990
```

- **Ukaz poly**

Izračuna koeficiente karakterističnega polinoma.

- **Ukaz norm**

Izračuna Evklidsko normo matrike ali vektorja. Za vektor je to koren vsote kvadratov členov. Za matriko je to največja singularna vrednost.

Vprašanja v razmislek

1. Kaj je vektor (stolpec/vrstica)?
2. Kaj je matrika?
3. Kako množimo matrike?
4. Kaj je linearni sistem enačb in kako ga rešimo?
5. Kaj je rang matrike (pomen)?
6. Kaj je determinata matrike (pomen)?
7. Matrični inverz.
8. Kaj so lastne vrednosti in kaj lastni vektorji?
9. Pomen matrike lastnih vektorjev.

3. Vizualizacija

2D grafi

Za izris, opremljanje osnovnih 2D grafov in delo z njimi uporabljamo naslednje ukaze:

Tabela 16: Risanje in opremljanje grafov 2D.

figure(i)	Odpre grafično okno stevilka i.
plot(x,y,x1,y1,x2,y2,..)	V okno izriše x,y graf. Vektorja x in y morata biti enake dolžine. Če navedemo več parov podatkov se izriše več grafov v isto okno.
title('Naslov')	Graf opremi z naslovom.
xlabel('Oznaka na X osi')	X os opremi z oznako.
ylabel('Oznaka na Y osi')	Y os opremi z oznako.
grid	V graf nariše mrežo.
axis([Xmin, Xmax, Ymin,Ymax])	Določitev območja X in Y osi grafičnega okna..
loglog(x,y,x1,y1,x2,y2,..)	Enak ukaz kot plot, s tem da se za izris na obeh oseh uporabi logaritemsko merilo.
hold on	Zadrži trenutno sliko v grafičnem oknu, tako da lahko kasneje dodamo še več potekov.
legend ('Prvi graf', 'Druga graf','..)	V grafično okno z več poteki doda legendo.
subplot(2,2,1) , plot(x1,y1)	Grafično okno razdeli v 4 podokna (2x2), V prvo nariše graf (x1,y1)
ginput(s)	Omogoča grafično odčitovanje s točk preko klika z miško.
text(X,Y,'tekst')	Na mesto določeno z X,Y koordinatama v grafičnem oknu napiđše tekst.
gtext('tekst')	Na mesto določeno s klikom miške v grafičnem oknu napiše tekst.

Ostalo

Tabela 17: Dodatni ukazi.

clf	Zbriše vsebino trenutno aktualnega grafičnega okna
close all	Zapre vsa grafična okna

- **Ukaz figure**

Z ukazom figure(i) se odpre i-to grafično okno. Ta ukaz je potrebno uporabiti zmeraj, ko želimo narisati graf v novo okno, to je pred ukazom plot. V primeru ko tega ukaza ne uporabimo se graf nariše v zadnje odprto grafično okno. Stara vsebina tega okna se (razen če smo uporabili ukaz hold on) izbriše. Če še ni odprto nobeno grafično okno in uporabimo ukaz plot, se avtomatsko odpre.

- **Ukaz plot**

XY graf izrišemo torej z ukazom *plot(x,y)* . X in Y morata biti vektorja enake dolžine. V primeru če je eden vektor stolpični drugi pa vrstica je potrebno enega izmed njiju transponirati. Namesto vektorjev *x i* n *y* lahko vpišemo v ukaz plot tudi kakšen izraz, npr. *plot(t,cos(t))* . Če želimo v grafično okno izrisati istočasno več grafov v ukazu plot navedemo x , y pare po vrsti, npr. *plot(x,y,x1,y1,x2,y2).* Vsak potek se izriše s svojo barvo.

- **Ukaz title**

Title(' naslov') doda naslov v grafično okno. V naslovu ne sme biti šumnikov.

- **Ukaza xlabel, ylabel**

xlabel('Oznaka na X osi') doda oznako na X os grafa, ylabel('Oznaka na Y osi') pa oznako na Y os.

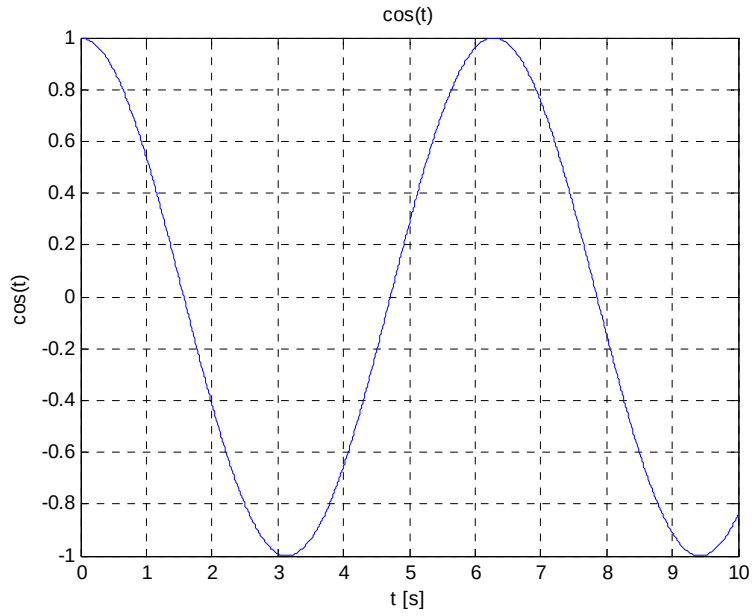
- **Ukaz grid**

Z ukazom *grid* se v graf izriše mreža.

Preizkusimo do sedaj prikazane ukaze na enostavnem primeru risanja funkcije cos(t):

Matlab:

```
>> t=[0:0.01:10];
>>y=cos(t);
>> figure(1), plot(t,y)
>> title('cos(t)')
>> xlabel('t [s]'), ylabel('cos(t)')
>> grid
```

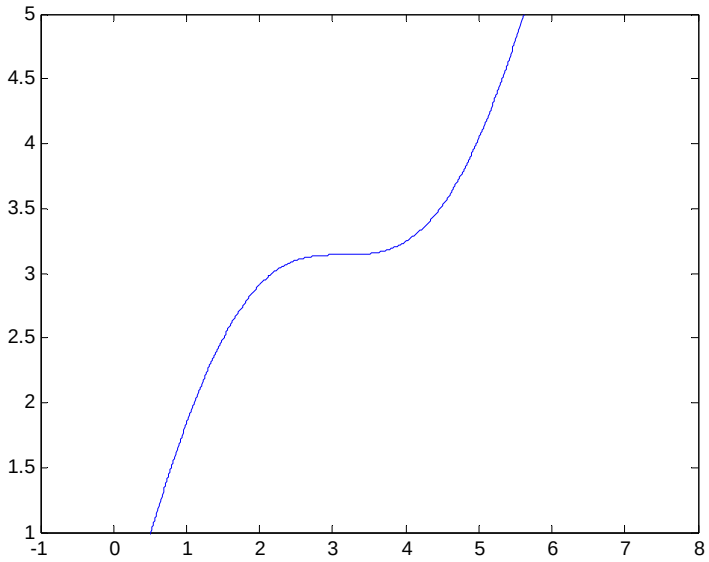


Slika 8: Uporaba funkcije plot.

- **Ukaz axis**

Območje izrisa grafa se v Matlab-u določi tako, da je celoten graf podan s podatki viden na sliki. Če želimo izrisati samo del grafa lahko določimo območje izrisa z ukazom *axis([Xmin, Xmax, Ymin,Ymax])*. Xmin in X max določata območje ki bo prikazano na X osi in Ymin in Ymax območje, ki bo prikazano na Y osi.
Drug način določitve območja izrisa je da ga izberemo neposredno v grafičnem oknu s klikom na gumba v orodni vrstici ('Zoom in' in 'Zoom out').

Matlab:
```
>> x=[0.1:0.01:10];
>> plot(x,cos(x)+log(10*x));
>> axis([-1,8,1,5])
```

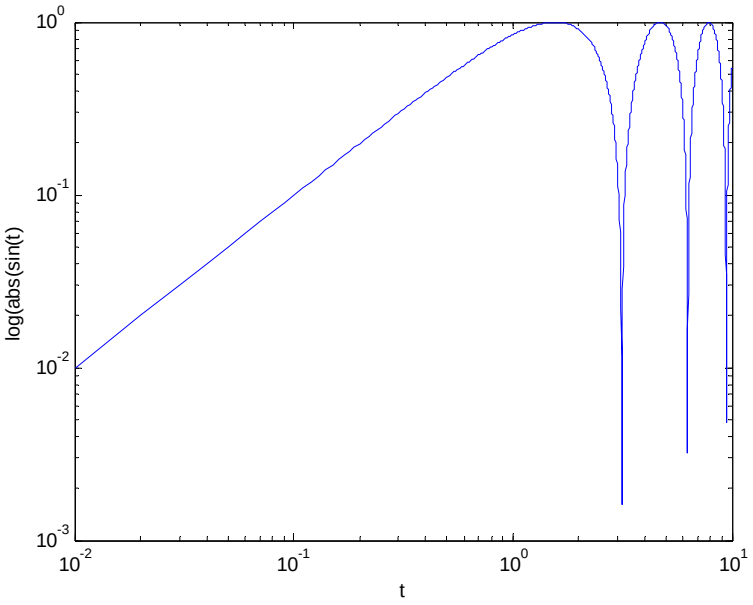


Slika 9: Uporaba ukaza axis.

- **Ukaz logolog**

Z ukazom *loglog(x,y,x1,y1,x2,y2,..)* izrišemo graf, ki ima na obeh oseh logaritemsko merilo. Za opremo grafa in merila prav tako uporabimo ukaze title, xlabel, ylabel, grid in axis.

Matlab:
```
>>x=[-10:0.01:10];
>>loglog(t,t*100)
>>loglog(t,abs(sin(t)))
>>xlabel('t'),ylabel('log(abs(sin(t)')
>>xlabel('t_logaritmicnomerilo'),ylabel('abs(sin(t)_logaritmicno merilo')
```

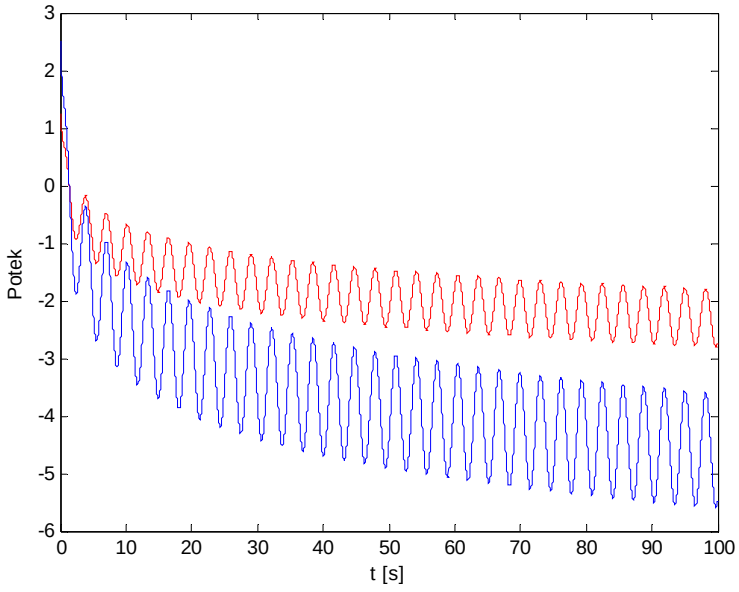


Slika 10: Dvojno logaritmično merilo.

- **Ukaz hold on**

Ukaz hold on zadrži trenutno sliko v grafičnem oknu. Naslednji ukaz plot, ki se nanaša na to okno samo doda nov potek.

Matlab:
```
>>t=([0.1:0.02:100]);
>>plot(t,sin(2*t)-log(t))
>>hold                    on
>> plot(t,0.5*(sin(2*t)-log(t)),'r')
>> xlabel('t [s]'),ylabel('Potek')
```

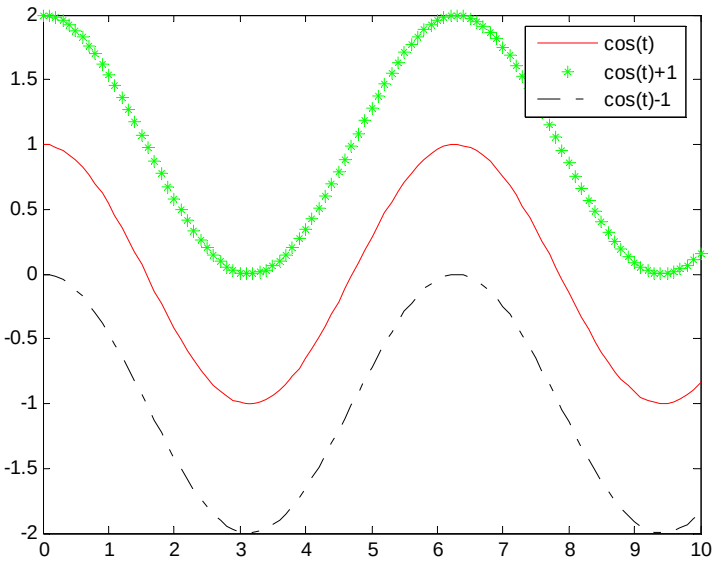


Slika 11: Uporaba ukaza hold.

- **Ukaz legend**

Z ukazom legend ('prvi potek', 'drugi potek') dodamo v grafično okno z več poteki legendo.

Matlab:
```
>> t=[0:0.1:10];
>> figure(1)
>> plot(t,cos(t),'r',t,cos(t)+1,'*g',t,cos(t)-1,'-.k')
>> legend('cos(t)','cos(t)+1','cos(t)-1')
```



Slika 12: Določanje načina prikaza krivulj.

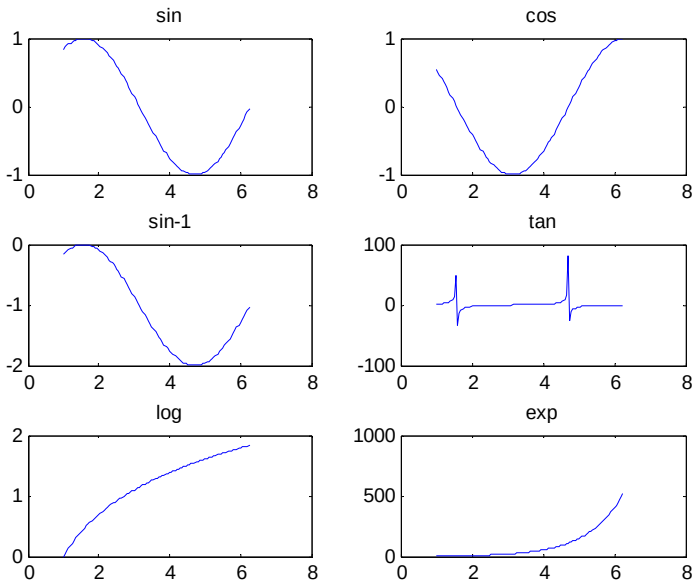
- **Ukaz subplot**

Ukaz subplot(n,m,i) grafično okno razdeli v nxm podoken in postavi i to okno za aktualno. Naslendji plot ukaz tako izriše graf v i-to podokno. Vsak podgraf lahko opremimo z oznakami, naslovi in podobno.

- **Ukaz ginput**

Z miško iz grafa odčitamo XY koordinate npr. 5 točk z ukazom *ginput(5)* . Petkrat kliknemo na graf in v delovnem prostoru se izpišejo koordinate.

Matlab:
```
>>t=1:0.05:2*pi;
>>subplot(3,2,1),plot(t,sin(t)),title('sin')
>>subplot(3,2,2),plot(t,cos(t)),title('cos')
>>subplot(3,2,3),plot(t,sin(t)-1),title('sin-1')
>>subplot(3,2,4),plot(t,tan(t)),title('tan')
>>subplot(3,2,5),plot(t,log(t)),title('log')
>>subplot(3,2,6),plot(t,exp(t)),title('exp')
```



Slika 13: Uporaba ukaza subplot.

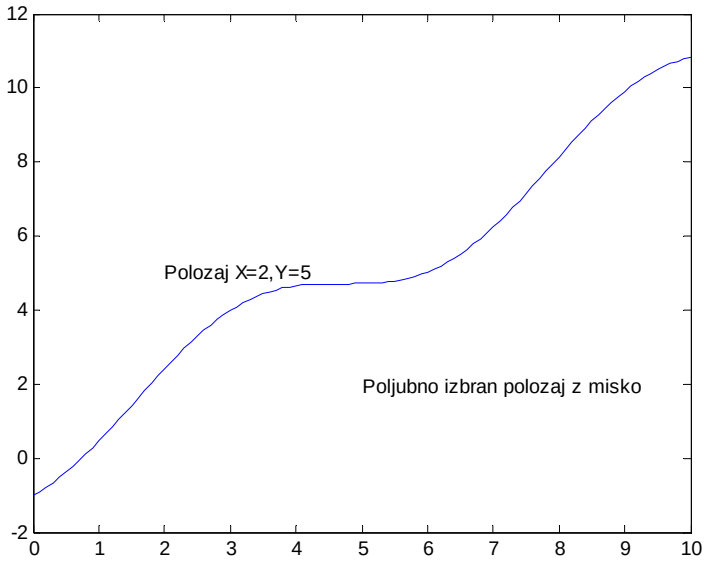
- **Ukaz text**

Z ukazom *text(X,Y,'tekst')* na mesto v tekočem grafičnem oknu določenem z X, Y koordinatami izpišemo tekst. X, Y koordinati sta enaki vrednostim na oseh grafa.
Tekste v grafična okna lahko dodajamo tudi tako, da kliknemo ustrezno ikono v orodni vrstici (ikona A), z miško izberemo ustrezno mesto in vpišemo tekst.

- **Ukaz gtext**

Ukaz *gtext('tekst')* ima enako funkcijo kot ukaz tekst, s tem da se sedaj mesto izpisa določi s klikom miške.
Tekste v grafična okna lahko dodajamo tudi tako, da kliknemo ustrezno ikono v orodni vrstici (ikona A), z miško izberemo ustrezno mesto in vpišemo tekst.

Matlab:
```
>>t=[0:0.1:10];
>> plot(t,t-cos(t))
>> text(5,5,'Polozaj X=5,Y=5')
>> gtext('Poljubno izbran polozaj z misko')
```



Slika 14: Uporaba ukaza gtext.

- **Določitev barve in tipa črt**

Če rišpemo več potekov v isto okno z ukazom plot se vsak potek avtomatsko izriše z drugi barvo, Laghko pa barvo določimo tudi sami. Npr. ukaz *plot(x1, y1, 'r', x2, y2, 'g')* bo izrisal potek x1, y1 z rdečo in x2, y2 z zeleno barvo. Možne barve so: 'c', 'm', 'y', 'r', 'g', 'b', 'w', in 'k'. Prav tako lahko izbiramo tip črte. Ukaz *plot(x1, y1, '--r', x2, y2, '*g')* bo izrisal potek x1, y1 z rdečo prekinjeno črto, x2, y2 pa z zelenimi zvezdicami. Možni tipi črt so: '-', '--', ':', '-.' , '*'.

- **Kopiranje in shranjevanje grafov**

Grafe lahko kopiramo v MS Word ali druge Microsoft programe z opcijo Edit->Copy Figure. Pred tem označimo v Edit->Copy Options opcijo Metafile.
Vsebino grafičnega okna lahko shranimo tudi z Edit->Save. Shranjeno datoteko lahko odpemo le v Matlabu.

Posebni 2D grafi

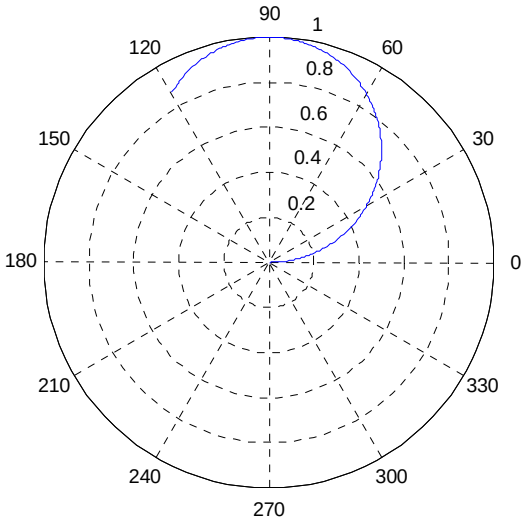
Tabela 18: Posebni 2D grafi.

polar(theta, r)	Izriše graf s podatki podanimi v polarnih koordinatah, theta je kot v radianih, r je polmer
bar (X,Y)	Stolpični graf, X je vektor dolžine *m* , Y matrika velikosti *nxm* .
pie(y)	Izris podatkov v obliki krožnih odsekov
stem(y)	Diskretni izris podatkov.
hist(y)	Prikaže zastopanost in porazdelitev elementov vektorja *y* .
stairs(y)	Stopnični graf.

- **Ukaz polar**

Ukaz *polar(theta, r)* izriše graf v polarnih koordinatah. *Theta* je vektor kotov podanih v radianih, *r* je vektor pripadajočih polmerov. Grafu lahko dodamo naslov z ukazom title('Naslov'). V isto grafično okno ne moremo narisati več potekov.

Matlab:
```
>> theta=[0:0.01:2/3*pi];
>> r=sin(theta);
>> polar(theta,r)
```

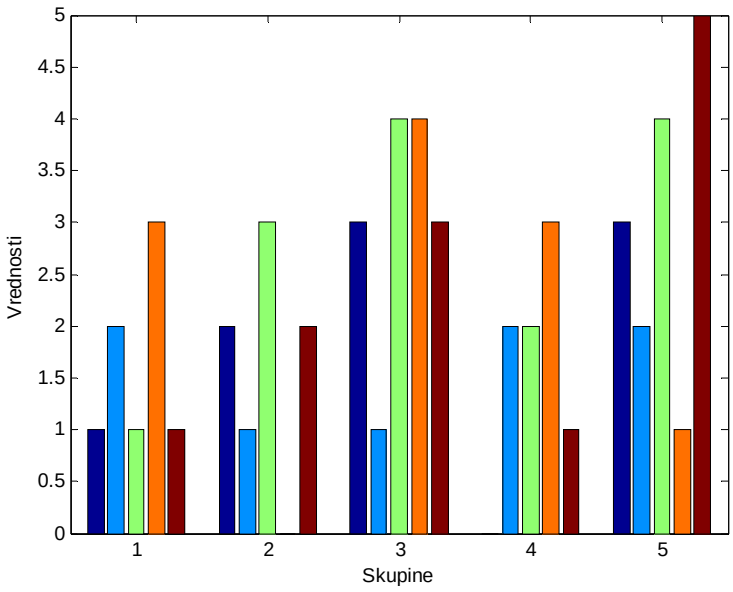


Slika 15: Uporaba ukaza polar.

- **Ukaz bar**

Ukaz *bar (X,Y)* izriše stolpični graf. *X* je vektor dolžine *m* , *Y* matrika velikosti *nxm*. Izrišejo se stolpci matrike *Y* kot M stolpičev na mesta določena s členi vektorja *X* . Členi vektorja *X* morajo biti enakomerno razporejeni.

Matlab:
```
>> y=[1,2,1,3,1;2,1,3,0,2;3,1,4,4,3;0,2,2,3,1;3,2,4,1,5];
>> x=[1,2,3,4,5];
>> bar(x,y)
>> xlabel('Skupine'),ylabel('Vrednosti')
```

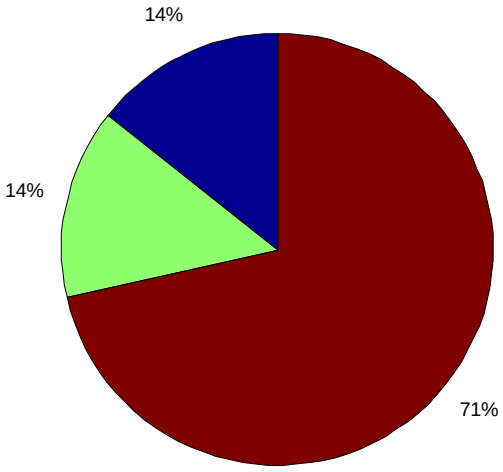


Slika 16: Uporaba ukaza bar.

- **Ukaz pie**

Ukaz *pie(y)* izriše odsekovni krožni graf. Za določitev posamezniih področij se vrednosti *x* normalizirajo po enačbi *y/sum(y)* . Posameznim odsekom lahko dodamo oznake z ukazom *pie(x,{'Prvi','Drugi','Tretji',..}).* Število oznak mora biti enako kot dolžina vektorja *x* .

Matlab:
>> x=[2,2,10];
>> pie(x)

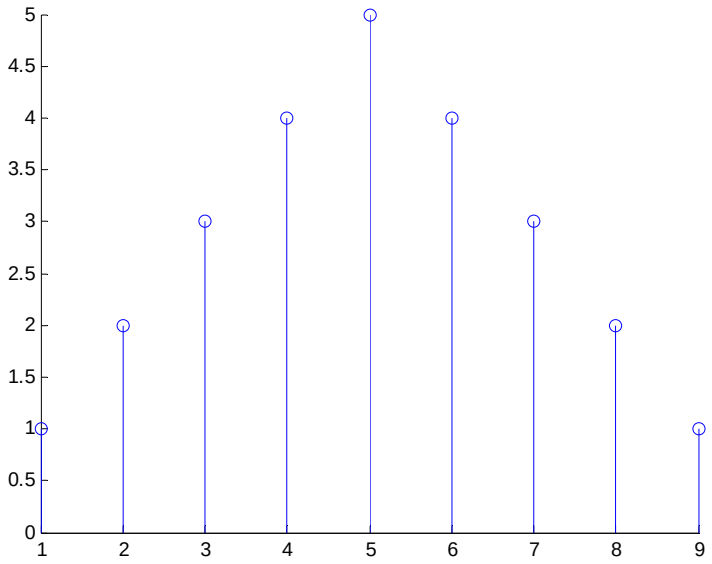


Slika 17: Uporaba ukaza pie.

- **Ukaz stem**

Ukaz *stem (y)* izriše podatke v vektorju *y* kot točke označene s krogcem. Na X osi se uporabi merilo po 1. Če želimo sami določiti merilo tudi na X osi uporabimo ukaz *stem(x,y)* .

Matlab:
```
>> x=[1,2,3,4,5,4,3,2,1];
>> stem(x)
```



Slika 18: Uporaba ukaza stem.

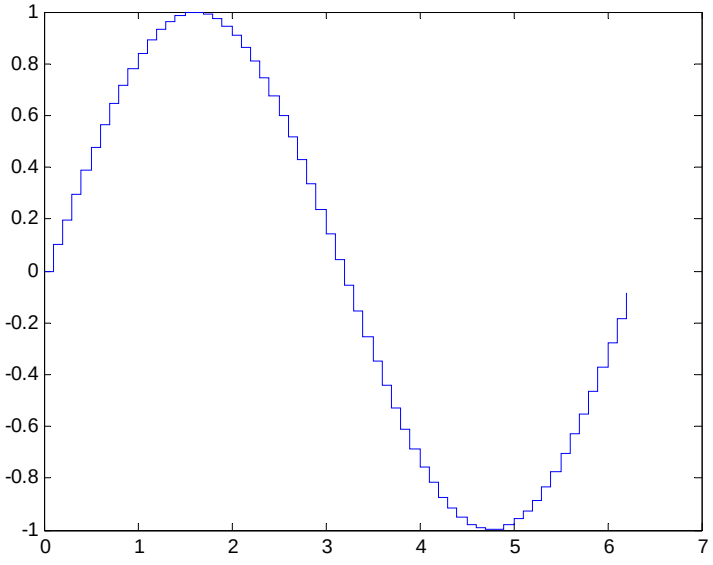
- **Ukaz hist**

Ukaz *hist (y)* razdeli elemente *y* v deset enakih odsekov in določi število elemntov v vsakem odseku in izriše ustrezno utežen stolpec. Če je kot argument podana matrika ukaz deluje po stolpcih.

- **Ukaz stairs**

Ukaz *stairs (y)* izriše stopnični graf ki na Y osi kaže vrednosti *y.* Vrednosti na X osi naraščajo po ena. Ukaz stairs(x,y) izriše graf na enak način, s tem da so tokrat določene vrednosti na obeh oseh.

Matlab:
```
>> x=0:0.1:2*pi;
>> y=sin(x);
>> stairs(x,y)
```

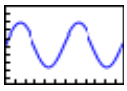
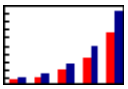
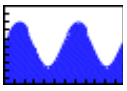
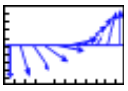
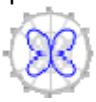
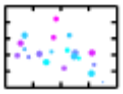
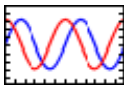
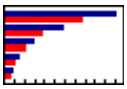
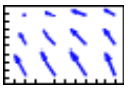
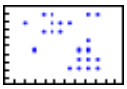
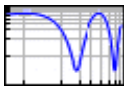
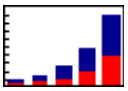
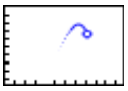
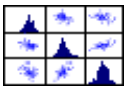
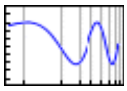
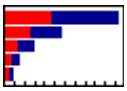
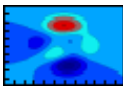
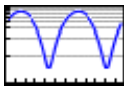
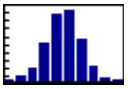
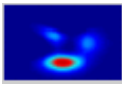
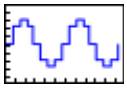
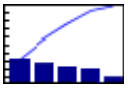
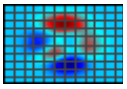
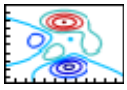
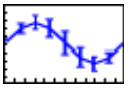
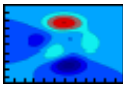
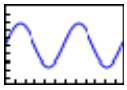
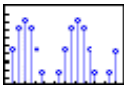
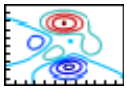


Slika 19:Uporaba ukaza stairs.

- **Pregled vseh 2D grafov**

Spodnja razpredelnica prikazuje vse 2D - plot funkcije, ki so na voljo v Matlabu. Nekatere od funkcij smo opisali tudi v tem poglavju.

Tabela 19: Tabela 2D funkcij.

Line Graphs	Bar Graphs	Area Graphs	Direction Graphs	Radial Graphs	Scatter Graphs
plot 	bar (grouped) 	area 	feather 	polar 	scatter 
plotyy 	barh (grouped) 	pie 	quiver 	rose 	spy 
loglog 	bar (stacked) 	fill 	comet 	compass 	plotmatrix 
semilogx 	barh (stacked) 	contourf 		ezpolar 	
semilogy 	hist 	image 			
stairs 	pareto 	pcolor 			
contour 	errorbar 	ezcontourf 			
ezplot 	stem 				
ezcontour 					

3D grafi

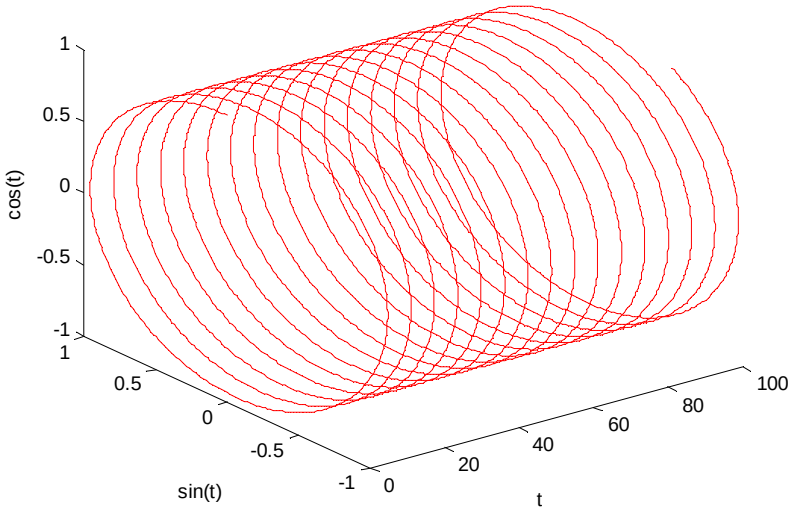
Tabela 20: Ukazi za risanje v 3D.

plot3(x,y,z)	Izriše 3D graf v aktualnem grafičnem oknu.
plot3(X,Y,Z)	X,Y,Z so ematrike enakih dimenzij. Izris poteka po stolpcih matrik.

- **Ukaz plot3(x,y,z)**

Izriše *x,y,z* graf. Vsi vektorji morajo biti enake dolžine. Graf opremimo z oznakami, in naslovom enako kot 2D grafe, torej z uporabo ukazov xlabel, ylabel in title. Z uporabo orodja rotate (ikona v orodni vrstici), lahko graf v 3D tudi rotiramo in pogledamo z drugih zornih kotov.

Matlab:
```
>> t=[0:0.02:100];
>> plot3(t,sin(t),cos(t),'r'), xlabel('t'),
>> ylabel('sin(t)'), zlabel('cos(t)')
```



Slika 20: Uporaba ukaza plot3.

- **Ukaz plot3(X,Y,Z)**

X, Y , Z s o matrike enake dimenzije *nxm* . Na isto grafično okno se izriše *m* #d potekov, vsak predstavlja en stolpec matrik. Z uporabo orodja rotate (ikona v orodni vrstici), lahko graf v 3D tudi rotiramo in pogledamo z drugih zornih kotov.

Posebni 3D grafi

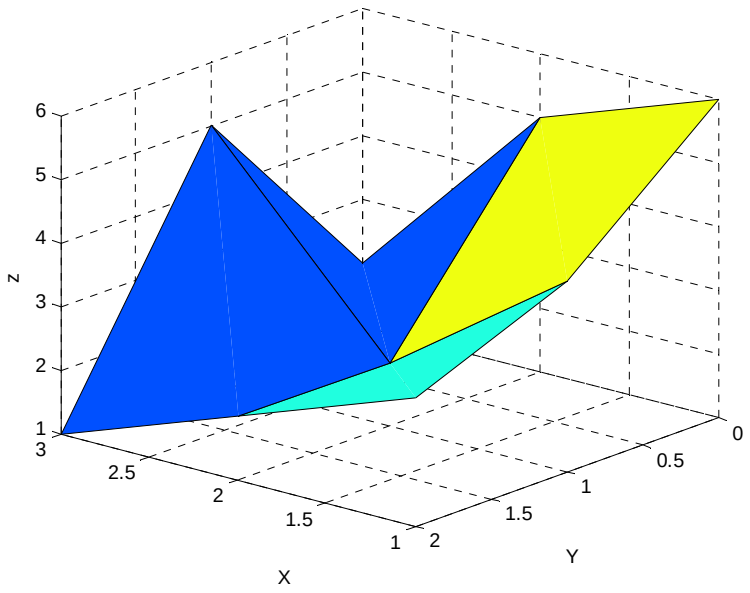
Tabela 21: Posebni ukazi 3D.

surf(x,y,Z)	Izriše barvno ploskev
mesh(x,y,Z)	Izriše barvno mrežo.
waterfall(x,y,Z)	Izriše barvno mrežo.

- **Ukaz surf**

Izriše barvno ploskev glede na podane argumente.Dolžina vektorja *y* je *n* , dolžina vektorja *y* je *m* in velikost matrike Z pa je *mxn* . Površina je določena s trojicami *x(j), y(i), Z(i,j).* Graf opremimo z oznakami, in naslovom enako kot 2D grafe, torej z uporabo ukazov xlabel, ylabel in title. Z uporabo orodja rotate (ikona v orodni vrstici), lahko graf v 3D tudi rotiramo in pogledamo z drugih zornih kotov.

Matlab:
```
>> x=[1,2,3];
>> y=[2,1,0];
>> Z=[3,2,1;4,2,5;6,5,2];
>> surf(x,y,Z)
>> xlabel('X'),ylabel('Y'),zlabel('z')
```



Slika 21: Uporaba ukaza surf.

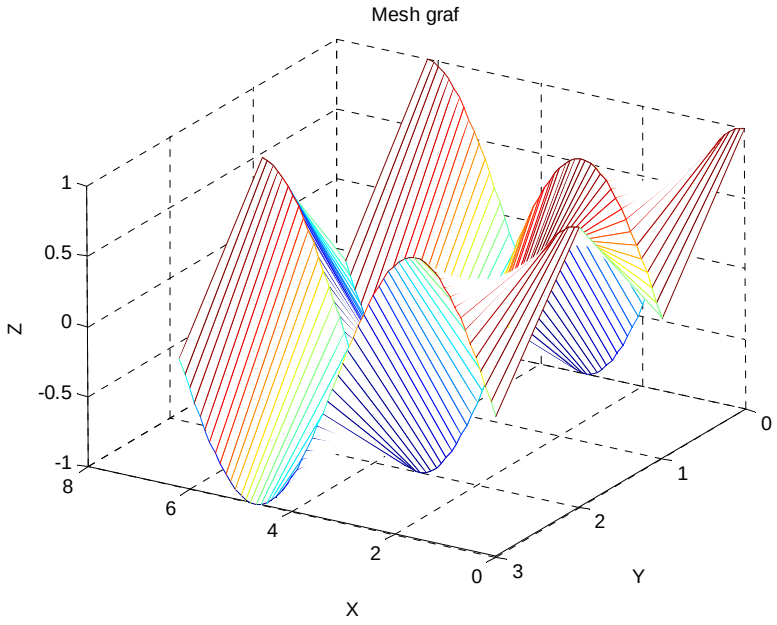
- **Ukaz mesh**

Izriše barvno mrežo glede na podane argumente.Dolžina vektorja *y* je *n* , dolžina vektorja *y* je *m* in velikost matrike *Z* pa je *mxn* . Površina je določena s trojicami *x(j), y(i), Z(i,j)* .Graf opremimo z oznakami, in naslovom enako kot 2D grafe, torej z uporabo ukazov xlabel, ylabel

in title. Z uporabo orodja rotate (ikona v orodni vrstici), lahko graf v 3D tudi rotiramo in pogledamo z drugih zornih kotov.

Matlab:

```
>> x=[0:0.1:2*pi];
>> y=[0,1,2,3];
>> Z=[cos(x);sin(x);cos(x);sin(x)];
>> mesh(x,y,Z)
>> xlabel('X'),ylabel('Y'),zlabel('Z')
```



Slika 22: Uporaba ukaza mesh.

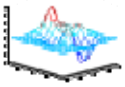
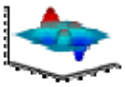
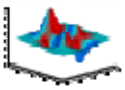
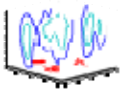
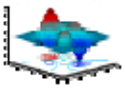
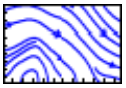
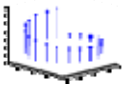
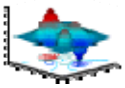
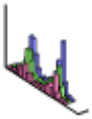
- **Ukaz waterfall**

Enak kot ukaz mesh, s tem da se stolpci mreže ne izrišejo.

- **Pregled vseh 2D grafov**

Spodna tabela prikazuje vse 3D–plot funkcije, ki so na voljo v Matlabu.

Tabela 22: Tabela 3D funkij.

Line Graphs	Mesh Graphs and Bar Graphs	Area Graphs and Constructive Objects	Surface Graphs	Direction Graphs	Volumetric Graphs
plot3 	mesh 	pie3 	surf 	quiver3 	scatter3 
contour3 	meshc 	fill3 	surfl 	comet3 	coneplot 
contourslice 	meshz 	patch 	surfc 	streamslice 	streamline 
ezplot3 	ezmesh 	cylinder 	ezsurf 		streamribbon 
waterfall 	stem3 	ellipsoid 	ezsurfc 		streamtube 
	bar3 	sphere 			
	bar3h 				

4. Programiranje

Števila (realna števila) so v Matlabu predstavljena v pomični piki. Matlab upošteva IEEE standard.

Pogojni stavki

Tabela 23: Pogojni stavki.

if	Pogojni stavek
for zanka	Zanka z določenim številom ponovitev
while zanka	Zanka z logičnim pogojem
switch	Pogojni skoki
break	Skok iz zanke ali pogojnega stavka

- **Ukaz if**

Ukazi napisani v okviru pogojnega stavka se izvedejo če je izpolnjen podan logični pogoj. Splošna oblika if stavka je naslednja:

Matlab:

```
if  pogoj1

 stavki1;

elseif pogoj2

        stavki2;

else

        stavki3;
```

V primeru če je izpolnjen *pogoj1* se torej izvedejo *stavki1* . Sicer se preveri *pogoj2* in če je izpolnjen ta, se izvedejo *stavki2* . Če nista izpolnjena ne *pogoj1* ne *pogoj2* se izvedejo *stavki3* . Seveda lahko v if stavku nastopa še več elseif pogojev, pa tudi else stavek ni nujen. If stavke je možno tudi vgnezditi. Pri pisanju pogojev uporabljamo logične operatorje <, >=, >=, ~=, = =. Njihova uporaba je opisana v poglavju Operatorji in osnovne matematične funkcije.

- **Ukaz for**

Z ukazom *for* generiramo zanko, ki se bo izvedla tolikokrat, kot smo določili. Splošna oblika for zanke je naslednja:

Matlab:
for stevec = zacetni:korak:koncni,

stavki;

end

Število ponovitev je torej določeno z začetno in končno vrednostjo števca ter korakom s katerim se števec povečuje. For zanke lahko tudi vgnezdimo.

- **Ukaz while**

Z ukazom *while* izvajamo določen del programa dokler je izpolnjen nek logični pogoj. Splošna oblika *while* zanke je naslednja:

Matlab:
while pogoj

stavki;

end

Pri pisanju pogojev uporabljamo logične operatorje <, >=, >=, ~=, ==. Ti so podrobno opisani v poglavju Operatorji in osnovne matematične funkcije.

- **Ukaz switch**

Z ukazom *switch* se izvedejo določeni deli programa, ki se izberejo glede na vrednost podanega izraza. Splošna oblika switch stavka je naslednja:

Matlab:
switch izraz

case vrednost1

stavki1;

case vrednost2

stavki2:

end

- **Ukaz break**

Z ukazom *break* se prekine izvajanje for ali while zanke in se nadaljuje izvajanje programa izven trenutne zanke. Če so zanke vgnezdene se prekine samo izvajanje notranje zanke. Če se ukaz uporabi v if ali switch stavku se izvajanje tega stavka prekine.

Matlab funkcije

Funkcije v Matlab-u imajo enako vlogo kot funkcije v drugih programskih jezikih. Prejmejo vhodne in vračajo izhodne parameter. Vse spremenljivke, ki se uporabljajo znotraj funkcije so lokalne in se torej njihova vrednost med posameznimi klici funkcije ohranja. Izjema so spremenljivke, ki so definirane kot globalne. Le izhodni parametric vplivajo na delovno okolje. Tako lahko zapišemo nove funkcije za MATLAB, ki so povsem enakovredne vgrajenim. Pri zapisu funkcij imamo na voljo vse možnosti, ki jih izkoriščajo vgrajene funkcije:

- vgradnjo pomoči, dostopno z ukazom help ime funkcije,
- uporabo privzetih vrednosti za manjkajoče podatke,
- možnost spremenljivega števila vhodnih in izhodnih parametrov.

Struktura funkcijske datoteke:

Tabela 24: Strukturafunkcijske datoteke.

GLAVA	function ime(x). . .procedura function y = ime(x) function [y,z] = ime(x1,x2,x3)
POMOČ	vrstice, ki se pričnejo s % takoj za glavo funkcije, so pomoč dostopne so z ukazom help ime funkcije
TELO	prireditveni ukazi operacije z vhodnimi podatki operacije z lokalnimi količinami zanke prirejanje vrednosti rezultatov

Tabela 25: Ukazi za definicijo funkcije.

function	Matlab funkcija.
nargin	Število vhodnih argumentov Matlab funkcije.
nargout	Število izhodnih argumentov Matlab funkcije.

- **Ukaz function**

Funkcijo zapišemo v obliki samostojne m-datoteke. Pri tem je smiselno m-datoteko shraniti pod enakim imenom kot je ime funkcije. Vhodni podatki v funkcijo se podajo kot argumenti pri njenem klicu. Na začetku funkcije pod komentar zapišemo besedilo, ki se bo izpisalo kot pomoč za uporabo te funkcije (ukaz help ime_funkcije).

Funkcijo, ki nam bo vhodno vrednost pomnožila z dva zapišemo kot:

Matlab:

```
% Pod komentar vpišemo tekst, ki se bo izpisal ko bomo v
% delovnem prostoru napisali ukaz help ime_funkcije.
% Opis delovanja funkcije.
% Ta funkcija množi vhodno vrednost z dva.
% Seznam vhodnih in izhodnih podatkov.
%
% Avtor, datum

function izhod = ime_funkcije(vhod)

izhod = vhod*2;
```

V delovnem prostoru ali iz drugih m-datotek jo lahko kličemo z ukazom ***ime_funkcije(vhodni_podatek)*.**
Funkcije lahko kličemo tudi iz drugih funkcij in iz ostalih m-datotek.

- **Ukaz nargin**

Je ukaz za določitev števila argumentov, ki jih je uporabnik podal ob klicu funkcije. Glede na to koliko vhodov je podanih se tako lahko v funkciji izvajajo različni algoritmi. Primer Matlab funkcije ki pomnoži med seboj dva, tri ali štiri vhodne argumente, pač glede na to koliko jih je podanih, je naslednji:
Primer m-datoteke mnozi.m

Matlab:

```
% Množi dva, tri ali štiri vhode

function izhod = mnozi(vhod1, vhod2, vhod3, vhod4)

if nargin==2
   izhod = vhod1*vhod2;
elseif nargin==3
   izhod=vhod1*vhod2*vhod3;
elseif nargin==4
   izhod=vhod1*vhod2*vhod3*vhod3;
end
```

Datoteko poženemo v delovnem prostoru z različnim številom vhodov. Funkcija deluje za dva, tri ali štiri vhode. Če uporabimo en vhod ali pet in več vhodov Matlab javi napako.

- **Ukaz nargout**

nargout('ime_funkcije') vrne število izhodnih podatkov iz funkcije. Npr. naša funkcija *mnozi* ima samo en izhodni argument:

Matlab:
>>nargout('mnozi')
ans =
1

Lokalne in globalne spremenljivke

Tabela 26: Lokalne in globalne spremenljivke.

global spremenljivka	Definicija globalne spremenljivke
isglobal(spremenljivka)	Ugotavlja ali je spremenljivka globalna
clear global	Zbriše vse globalne spremenljivke

- **Ukaz global**

Vrednosti spremenljivk in izrazov lahko v Matlab-u priredimo na več načinov. Vrednosti lahko priredimo v delovnem prostoru ali v m-datoteki. V m-datoteki lahko uporabljamo tudi spremenljivke, ki smo jim prej priredili vrednost v delovnem prostoru. Prav tako lahko spremenljivke ki smo jim vrednost priredili v m-datoteki uporabljamo v delovnem prostoru, seveda potem, ko datoteko zaženemo.
Ne velja pa to za funkcije, torej m-datoteke, ki se začno z rezervirano besedo *function* . Vrednosti spremenljivk tako lahko v funkcijo iz delovnega prostora preberemo samo, če jih uporabimo kot vhodne podatke, rezultate operacij v funkciji pa lahko preberemo v delovnem prostoru samo, če so to izhodni podatki funkcije. Vse spremenljivke katerim je vrednost prirejena samo znotraj funkcije se lahko uporabljajo samo tam in svoje vrednosti med posameznimi klici funkcije ne ohranijo. Pravimo, da so te spremenljivke **lokalne**.
Včasih pa želimo, da se vrednost lokalne spremenljivke ohrani, oziroma da lahko njeno vrednost uporabimo v več funkcijah. Takrat uporabimo ukaz ***global ime_spremenljivke*** . Če je torej v funkciji spremenljivka definirana kot **globalna** , lahko z njo delamo tudi v delovnem prostoru. **Je pa potrebno spremenljivko definirati kot globalno v vseh funkcijah v katerih jo želimo uporabiti.** Spremenljivko definiramo kot globalno preden ji priredimo vrednost.

- **Ukaz isglobal**

Če za neko spremenljivko nismo prepričani ali je definirana kot globalna uporabimo ukaz *isglobal(ime_spremenljivke)*. Če je spremenljivka definirana kot globalna je odgovor vrednost 1, če pa je lokalna pa 0.

- **Ukaz clear global**

Je verzija ukaza clear. Medtem ko ukaz *clear* izbriše vse definirane spremenljivke, lokalne in globalne, pa ukaz *clear global* izbriše samo vse globalne spremenljivke.

5. Kazalo slik

6. Literatura

[1] The MathWorks, Inc. MATLAB, Using MATLAB, Natick, 1999.

[2] The MathWorks, Inc. MATLAB, Using MATLAB Graphics, Natick, 1999.

[2] MATLAB guide, Higham D. J., 1999.

[3] Matlab na straneh podjetja MathWorks, http://www.mathworks.com/products/matlab/

[4] MATLAB Educational Sites, http://www.eece.maine.edu/mm/matweb.html

[5] http://www.ro.feri.uni-mb.si/predmeti/navodila/Matlab_nav/matlab_65.htm

[5] http://www.math.pitt.edu/~sussmanm/2070Fall06/lab_02/index.html#Exercise2

[5] http://www.facstaff.bucknell.edu/maneval/help211/exercises.html