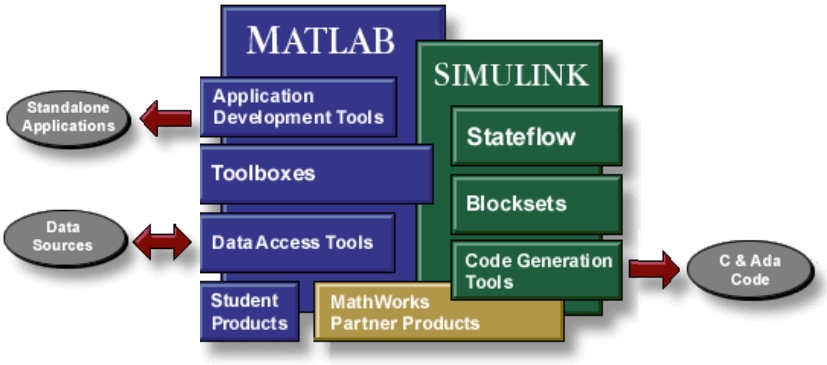


Jezik Matlab

- programski jezik visoke razine
- zapis podataka u obliku *matrica/polja*
- kontrola toka programa, funkcija, strukture podataka, ulaza/izlaza i objektno orijentiranih dijelova programa

Matematičke funkcije

- zbirka matematičkih funkcija od jednostavnih sve do veoma kompleksnih (npr. interpolacija, polinomske funkcije, računanje s matricama, Besselove funkcije, brza Fourierova transformacija, svojstvene vrijednosti itd.)

Radno okružje

- zbirka alata s kojima korisnik izvodi funkcije

Grafički sustav

- prikaz 2D i 3D grafike visoke kvalitete
- obrada slike, animacija, alati za grafičko korisničko sučelje (GUI)

Alati za izradu aplikacija (*Application Development Tools*)

- izrada programa u programskim jezicima C i Fortran koje se mogu uključiti u MATLAB projekte
- pozivanje rutina iz MATLAB ("dinamičko povezivanje")
- pozivanje MATLAB - a kao jedinice za računanje
- alati za izradu i distribuciju samostalnih verzija MATLAB aplikacija

TOOLBOXES

- Mogućnost pisanja vlastitih funkcija u Matlabu omogućila je izradu alata *Toolbox*-ova
- kolekcija gotovih **funkcija prilagođenih za posebne aplikacije** koje daju dodatne mogućnosti MATLAB-u i Simulink-u
- funkcije su ugrađene u MATLAB jezik i mogu se jednostavno pregledavati i modificirati ("otvoreni kod")

Npr.

- Simboličko računanje
- Podešavanje krivulja („fitanje")
- Identificiranje sustava
- Optimiranje procesa

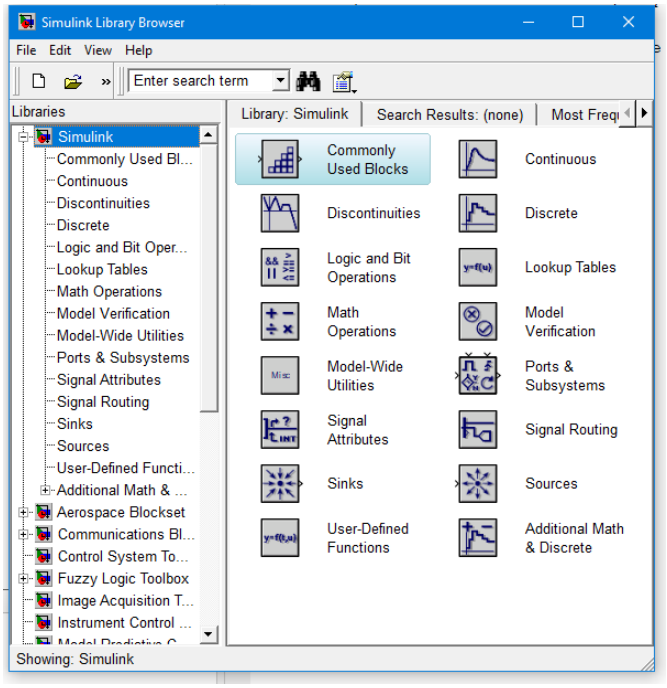
- Statistika
- Neuronske mreže, itd.

Alati za akviziciju i pristup podacima (***Data Access Tools***)

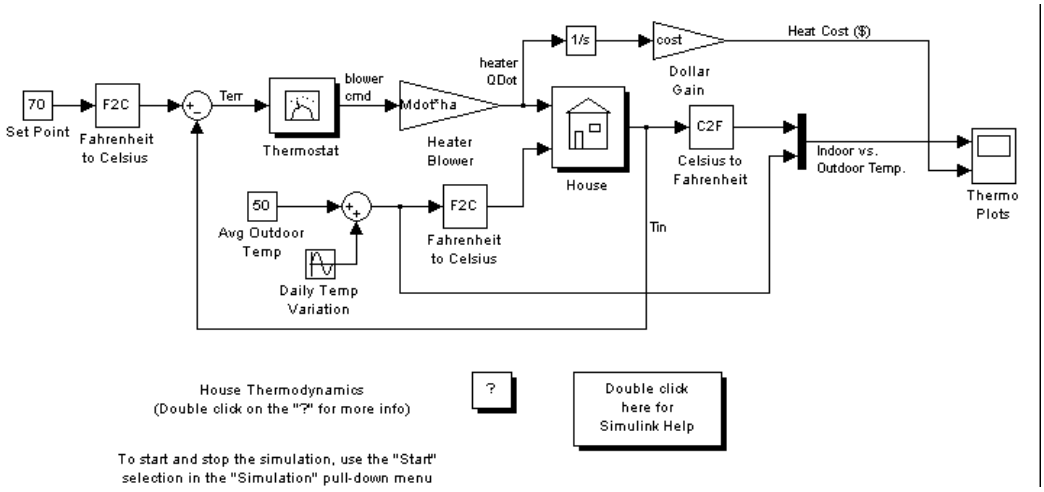
- izravan pristup podacima iz vanjskih jedinica i drugih softverskih paketa
- komunikacija i upravljanje uređajima za akviziciju podataka
- povezivanje s drugim programima (npr. MS Excel)

MATLAB Studentska verzija

SIMULINK



- interaktivni sustav za ***simuliranje*** nelinearnih dinamičkih sustava (isporučuje se zajedno s MATLABom)
- ***grafičko okruženje***, modeliranje i prikaz jednostavnim **blok dijagramima**
- linearni, nelinearni, vremenski kontinuirani, diskretni i viševeličinski sustavi
- vizualni prikaz rezultata simulacije, animacije i zapis u datoteke
- promjena simulacijskih parametara moguća i tijekom same simulacije



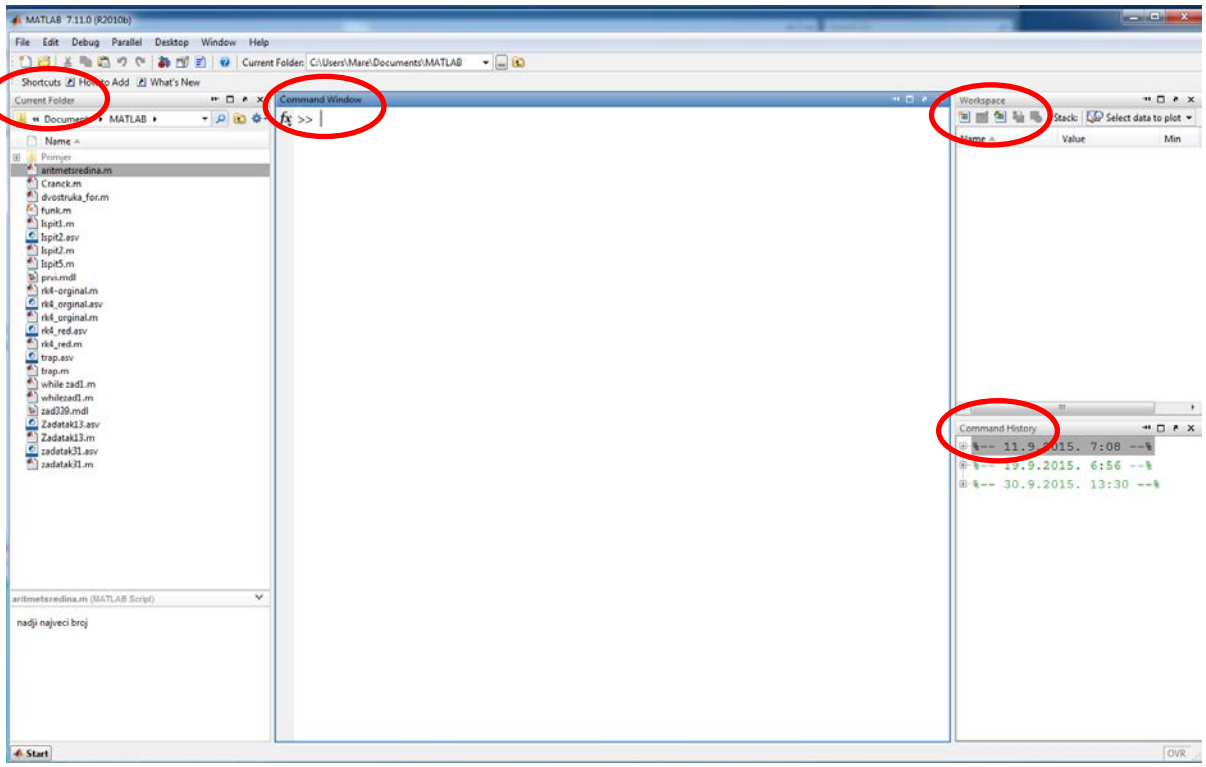
Blocksets

- ugrađeni dodaci Simulink-a koji podržavaju posebne primjene (npr. obrada digitalnih signala, posebne primjene itd.)

Real-Time Workshop

- generiranje C koda iz blok dijagrama i pokretanje aplikacija na različitim sustavima u realnom vremenu

Matlab sučelje



Varijable

Varijable su u osnovi memorijske lokacije čiji se sadržaj može mijenjati tokom izvođenja programa.

Varijable po **tipu** mogu biti cjelobrojne, decimalne, kompleksne, znakovne….

U Matlabu se znak "=" naziva operatorom pridruživanja koji pridružuje vrijednost varijabli.

Njegovo značenje je različito od znaka jednakosti u matematici. Lijevo od operatora pridruživanja može biti samo jedno ime varijable.

var = 10; Definiramo varijablu i pridružimo joj neku vrijednost

var + 10

a = var + 1 Pridružujemo varijabli a vrijednost varijable var uvećanu za 1

var = var + 1 Pridružujemo varijabli var vrijednost varijable var uvećanu za 1

var2 = var+1 Pridružujemo varijabli var2 vrijednost varijable var uvećanu za 1

Matrice

Unos matrica

Matrice se u MATLAB mogu unositi na nekoliko načina:

- direktni unos elemenata
- učitavanje matrica iz vanjske datoteke s podacima
- generiranje matrica pomoću ugrađenih funkcija
- kreiranje matrica s vlastitim M-datotekama

Pri unosu matrice kao liste elemenata pridržavamo se slijedećih pravila:

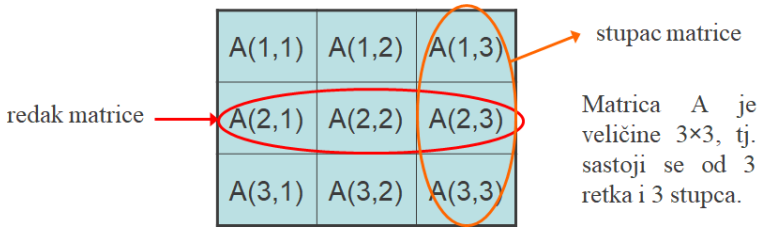
- odjeljujemo elemente reda s prazninama ili zarezom
- kraj svakog reda označavamo sa točkom-zarez `;`
- cijela lista elemenata nalazi se unutar uglatih zagrada, `[ ]`.

Za **unos matrice (4x4)**, jednostavno upišemo:

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB strukturirano prikazuje matricu A:

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```



Elementi matrice (*engl. Subscripts*)

Elementi reda `i` i stupca `j` matrice `A` označavaju se s **`A(i,j)`**. Npr., `A(4,2)` je broj u četvrtom redu i drugom stupcu. U našoj matrici `A(4,2)` je `15`.

```
A(4,2) =
            15
```

Moguće je izračunati sumu elemenata četvrtog stupca od `A` pišući:

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

što daje

```
ans =
     34
```

Operator Colon

Jedan od najvažnijih MATLABovih operatora je dvotočka (engl. *colon*) `:`
Javlja se u nekoliko različitih oblika. Izraz

```
1:10
```

je vektor u obliku retka koji sadrži cijele brojeve od 1 do 10

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

Za dobivanje razmaka/koraka koji nije jediničan, navodi se korak. Npr.,

```
100:-7:50
```

daje

```
100    93    86    79    72    65    58    51
```

i

```
0:pi/4:pi
```

daje

```
0    0.7854    1.5708    2.3562    3.1416
```

Ako se dvotočka navodi u indeksu odnosi se na dio matrice

```
A(1:k,j)
```

je prvih `k` elemenata `j`tog stupca od `A`. Npr.,

```
A(1:3,2)
```

Naredba

```
sum(A(1:4,4))
```

računa sumu četvrtog stupca. Ali, postoji bolji način. Operator Colon sam po sebi poziva *sve* elemente u redu ili stupcu matrice, a ključna riječ `end` odnosi se na zadnji red ili stupac. Prema tome

```
sum(A(:,end))
```

računa sumu svih elemenata u zadnjem stupcu od `A`.

```
ans =
     34
```

Unos matrice iz Excel datoteke

Ako je matrica spremljena u **Excel** datoteci, može se učitati u MATLAB koristeći funkciju **`readmatrix`** ili **`xlsread`** (za starije verzije prije R2019b).

1. Učitavanje cijele matrice iz Excela

Ako Excel datoteka sadrži samo brojeve (bez zaglavlja ili miješanih podataka):

```
B = readmatrix('ime_datoteke.xlsx');
```

2. Učitavanje specifičnog raspona ćelija

Ako se želi učitati samo određeni dio Excela (npr. raspon **B2:E5**):

```
B = readmatrix('ime_datoteke.xlsx', 'Range', 'B2:E5');
```

3. Učitavanje Excela sa funkcijom xlsread

Kod starijih verzija MATLAB-a može se koristiti **`xlsread`**:

```
B = xlsread('ime_datoteke.xlsx');
```

Ako Excel sadrži i tekstualne i numeričke podatke, `xlsread` vraća tri izlaza:

```
[numbers, text, raw] = xlsread('ime_datoteke.xlsx');
```

- `numbers` – samo numerički podaci
- `text` – samo tekstualni podaci
- `raw` – svi podaci (brojevi i tekst)

4. Ako datoteka sadrži zaglavlja ili stringove, može se koristiti:

```
A = readtable('ime_datoteke.xlsx'); % učitava tablicu
```

Generiranje jednostavnih matrica

MATLAB ima četiri osnovne funkcije za stvaranje jednostavne matrice:

zeros	sve nule
ones	sve jedinice
rand	jednoliko raspoređeni slučajni elementi u intervalu [0, 1]
randn	normalno distribuirani slučajni elementi

Nekoliko primjera:

```
Z = zeros(2,4)
Z =
     0     0     0     0
     0     0     0     0

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

N = fix(10*rand(1,10))
N =
     4     9     4     4     8     5     2     6     8
0

R = randn(4,4)
R =
    1.0668    0.2944   -0.6918   -1.4410
    0.0593   -1.3362    0.8580    0.5711
   -0.0956    0.7143    1.2540   -0.3999
   -0.8323    1.6236   -1.5937    0.6900
```

Operatori

Oznaka	**Operacija**	**engl.**
+	Zbrajanje	Addition
-	Oduzimanje	Subtraction
*	Množenje	Multiplication
/	Dijeljenje	Division
\	Recipročno dijeljenje	Left division
^	Potencija	Power
'	Kompleksno konjugirano transponiranje	Complex conjugate transpose
( )	Određivanje redoslijeda izvođenja operacija	Specify evaluation order

Elementarne matematičke funkcije

abs(x)	apsolutna vrijednost
exp(**x**)	e^x
log (**x**)	ln x
log10(**x**)	log x
rem(**x**,**y**)	ostatak (engl. *reminder*) nakon dijeljenja x/y
round(**x**)	zaokružuje na najbliži cijeli broj (integer)
sign(**x**)	daje predznak argumenta (vrijednosti -1, 0, 1)
sqrt(**x**)	kvadratni korijen
a^b	$\mathbf{A^b}$
a:**b**	a/b ili b\a

Sintaksa

- varijable su "*case sensitive*", razlikuju se velika i mala slova, mogu imati do 19 karaktera i moraju započeti slovom. Samo slova engleske abecede, brojke i _
- tekst iza **%** uzima se kao komentar
- na kraju svakog reda stavlja se " **;** " , ako želimo ispis stavlja se " **,** "
- ako je jednadžba preduga piše se " **...**" i pritisne *Enter*, a zatim se nastavlja pisati u novom redu

Ctrl + c kad se pritisne ova kombinacija tipaka prekida se izvođenje programa

Radni prostor

`Radni prostor` (*workspace*) je područje memorije dostupno iz MATLAB komandne linije. Naredbe `who` i `whos`, prikazuju trenutni sadržaj radnog prostora. Naredba `who` daje kratku listu, dok `whos` uz to daje veličinu i neke dodatne informacije.
Pogledajmo što daje naredba `whos` na radnom prostoru koji sadrži rezultate neke od primjera. Vidimo nekoliko različitih struktura podataka.

```
whos
```

```
  Name      Size         Bytes  Class

  A         4x4            128  double array
  M        10x1           3816  cell array
  S         1x3            442  struct array
  h         1x11            22  char array
  n         1x1              8  double array
  s         1x5             10  char array
  v         2x5             20  char array
```

Da bi obrisali sve postojeće varijable iz radnog prostora unesite

```
clear
```

Da bi obrisali sve iz naredbenog prostora unesite

```
clc
```

Save

Naredba `save` čuva sadržaj radnog prostora u MAT-datoteci koja se može pročitati naredbom `load` prilikom slijedećeg rada u MATLAB-u. Npr.

```
save primjer
```

snima cijeli sadržaj radnog prostora u datoteku `primjer.mat`.

Ako je potrebno, mogu se snimiti samo određene varijable navodeći njihova imena nakon imena datoteke. Varijable se spremaju u binarnom formatu koji MATLAB može čitati brzo. Ako je potrebno datotekama pristupiti izvan MATLAB-a, može se naznačiti neki drugi format.

`-ascii`	`8-digit tekst format`
`-ascii -double`	`16-digit tekstualni format`
`-ascii -double -tabs`	`dijeli polje elemenata s tabovima`
`-append`	`dodaje podatke na postojeći MAT-file.`

save ime_datoteke varijable -ascii -tabs (koristi ASCII oblik s 8 znamenki)

Load

Load naredba čita binarnu datoteku koja sadrži matrice stvorene pri prethodnom radu s MATLABom pomoću naredbe *Save* ili čita tekst datoteku koja sadrži numeričke podatke. Tekst datoteka treba biti organizirana kao pravokutna tablica brojeva, odvojena prazninama s jednim redom u liniji i jednakim brojem elemenata u svakom redu. Ovako izgleda tekst datoteka stvorena izvan MATLABa:

```
16.0   3.0    2.0   13.0
 5.0   10.0  11.0    8.0
 9.0    6.0   7.0   12.0
 4.0   15.0  14.0    1.0
```

Spremanjem datoteke pod imenom npr.

```
save magik.dat
```

Zatim naredbom

```
load magik.dat
```

čita se datoteka i stvara varijabla `magik` koja je ustvari gore napisana matrica.

Script File ili M-file

To su jednostavne tekst datoteke koje sadrže MATLAB naredbe, koje se mogu lako otvarati i izvoditi, a pri čemu MATLAB slijedi niz naredbi u datoteci. Izraz *M-file* kaže nam da imena script datoteka moraju završavati s ekstenzijom ".**m**", npr. `primjer.m`

Za stvaranje M-datoteke otvara se *New* iz izbornika *File* i odabire se *M-file*. Na taj način otvara se MATLABov tekst editor u kojeg ćemo upisivati naredbe.

Nakon što takvu datoteku spremimo na disk, jednostavno je pokrećemo utipkavanjem njenog imena u MATLAB promptu, ali pri tom moramo biti u direktoriju gdje je datoteka pohranjena:

```
» primjer
```

Sve varijable iz M-datoteke postaju dio radnog prostora MATLABa.

Primjer: kreirajte matricu A koja sadrži ova četiri redka (studenti sami)

```
A =
    16.0     3.0     2.0    13.0
     5.0    10.0    11.0     8.0
     9.0     6.0     7.0    12.0
     4.0    15.0    14.0     1.0
```

Pohranite datoteku pod imenom `magik.m`.

Nakon toga naredbom `magik` učitajte datoteku s varijablom A koja sadrži ovu matricu.

Jednostavna polja

Unos polja ima slijedeći oblik:

```
varijabla=[ niz brojeva odvojenih zarezom ili razmakom ]
```

```
X = [2, 5, 7, 9, 11]
```

Polja koja sadrže linearno razmaknute elemente mogu se formirati na slijedeći način:

```
first:increment:last
```

(*prvi član: prirast : zadnji član*)

```
Y = 1:1:10
```

a može i pomoću funkcije:

```
linspace(first, last, number_of_points)
```

Funkcije polja primjenjuju funkciju na svaki pojedini element polja.
Osnovne matematičke operacije između skalara i polja primjenjuju se na sve elemente u polju, npr.

```
X-2
```

oduzima dvojku od svih elemenata u `x`.

Osnovne matematičke operacije između polja vrijedi ako su polja iste veličine. Množenje i dijeljenje simbolički se prikazuje kao `.*` i `./` ili `.\`

Polja mogu imati bilo koju pravokutnu formu. Polje s jednim redom obično se naziva retčasti vektor. Polje s jednim stupcem naziva se stupčasti vektor. Polje s više redova i stupaca nazivaju se matrice. Retčasti vektor može se pretvoriti u stupčasti i obratno uporabom operacije točka-apostrof, npr. ako je `x` red, `x.'` je stupac.

Matrica je isto polje:

```
A = [ 6 4 6 3; 5 sqrt(2) 4.2 5; 5 sin(0.5) 4 3; 4 15 14 1.3 ]
```

Točka-apostrof označava transponiranje. Samo apostrof označava kompleksno konjugirano transponiranje.

Format

Naredba `Format` upravlja s numeričkim formatom vrijednosti prikazanih u MATLAB-u. Naredba djeluje samo na prikaz brojeva, a ne i kako će MATLAB računati i spremati ih.

Pogledajmo nekoliko različitih formata, zajedno s izlazom dobivenim za vektor `x` čije su komponente različite veličine:

```
x = [4/3 1.2345e-6]

format short      % Prikazuje brojeve s 4 decimalna mjesta.
   1.3333    0.0000

format short e     % Prikazuje brojeve u eksponencijalnom obliku s
                   4 značajne decimale.
   1.3333e+000  1.2345e-006

format short g      % kombinacija decimalnog i eksponencijalnog
                      zapisa ovisno o veličini broja.

   1.3333  1.2345e-006

format long        % Prikazuje brojeve s 15 značajnih decimala.
   1.33333333333333   0.00000123450000

format long e      % eksponencijalni zapis s 15 decimala
   1.333333333333333e+000    1.234500000000000e-006

format long g         % MATLAB bira između decimalnog i
                      eksponencijalnog zapisa, ali s više decimala
   1.33333333333333              1.2345e-006

format bank       % 2 decimalna mjesta, za financijske izračune
   1.33          0.00
```

Funkcije ispisa i upisa

Funkcija `input`

Funkcija `input` omogućuje unos podataka tijekom izvođenja programa napisanog u skript datoteci. Sastavljena je od ključne riječi `input` i imena varijable čiju ćemo vrijednost upisati.

Sintaksa naredbe `input`:

```
varijabla=input('tekst koji će biti prikazan u naredbenom prozoru');
```

Nakon pokretanja skript datoteke i izvođenja programske linije s funkcijom `input` u naredbenom prozoru se prikazuje poruka nakon koje se pojavljuje treptajući kursor. To je poruka korisniku da putem tipkovnice utipka vrijednost koja će biti pridružena varijabli. Nakon upisivanja vrijednosti potrebno je pritisnuti tipku Enter. Time će varijabli biti pridružena upisana vrijednost.

Funkcija `disp`

Funkcija `disp` ispisuje vrijednosti varijable, bez da se ispisuje njeno ime.

Sintaksa:

`disp` (ime_varijable)

Funkcija `fprintf`

Pomoću funkcije `fprintf` možemo tiskati rezultate (tekst i podatke) na ekranu ili upisati u datoteku.

Prilikom ispisa moguće je formatirati rezultat, npr. tekst i numeričke vrijednosti mogu biti izmiješani i prikazani u istom redu, a moguće je zadati i format brojeva.

Za zajednički prikaz teksta i broja (vrijednosti varijable) rabimo funkciju `fprintf` čija je sintaksa:

```
fprintf('tekst kao niz znakova %5.2f dodatni tekst',ime_varijable)
```

Znak **%** označava mjesto gdje broj treba umetnuti u tekst.

ime_varijable označava ime varijable čija se vrijednost prikazuje.

-5.2f - Elementi formatiranja (definiraju format broja). **f** je znak konverzije (obvezno). **5** ovdje označava širinu polja, **.2** označava preciznost (neobvezno) npr. na 2 decimale.

Primjer: Računanje prosječne vrijednosti koncentracije u dva mjerenja

```
% Unos podataka za mjerenje koncentracije
conc1 = input('Unesi prvo mjerenje koncentracije: ');
conc2 = input('Unesi drugo mjerenje koncentracije: ');
% Računanje prosjeka koncentracije
```

```
prosjek_koncentracije = (conc1 + conc2) / 2;
% Ispis rezultata
fprintf('Prosjek koncentracije je: %5.2f', prosjek_koncentracije);
```

Za vježbu: *Napišite primjer za računanje prosječnog broja bodova postignutih na dva testa.*

Naredbe odluke i ponavljanja

Grananje - struktura `if end`

Izraz u **`if`** naredbi odluke može biti sastavljen od relacijskih (<, <=, >, >=, ==, ~=) i/ili logičkih operatora (&, |, ~). Ako je logički izraz u iskazu **`if`** *istinit* program izvodi niz naredbi ili funkcija između **`if`** i **`end`**. Ako je logički izraz *neistinit*, odnosno, ako uvjet nije ispunjen, izlazi se iz tog dijela programa, tj. program preskače naredbe između iskaza **`if`** i **`end`** i nastavlja izvršavati naredbe iza iskaza **`end`.**

```
if logički_izraz
naredbe ili funkcije;
end
```

Primjer:

```
prvi_broj= 2;
drugi_broj = 3;
if prvi_broj > drugi_broj
disp('prvi je veci')
end
if prvi_broj < drugi_broj
disp('drugi broj je veci')
end
```

Grananje - struktura `if else end`

```
if izraz
naredbe ili funkcije
else
naredbe ili funkcije
end
```

Ako je izraz (uvjet) u **`if`** ispunjen, izvršavaju se naredbe ili funkcije između **`if`** i **`else`**. Ako uvjet nije ispunjen, izvršavaju se naredbe ili funkcije između **`else`** i **`end`**.

Primjer: Varijabli **x** slučajno se dodjeljuje realna vrijednost u rasponu [0,1]. Zatim se ispituje da li je vrijednost varijable **x** manja od 0,5. Ukoliko je vrijednost varijable **x** < 0,5 ispisuje se "pismo", a ukoliko je veća od ili jednaka 0,5 ispisuje se "glava".

```
x = rand
if x < 0.5
disp('pismo')
else
disp('glava')
end
```

Cikličke strukture

Niz algoritamskih koraka u kojem se jedan ili više koraka može izvršiti više od jedanput pri jednom izvršavanju algoritma zadatka tvori **cikličku algoritamsku shemu**.

Ciklička struktura realizira se u obliku programske **petlje** (engl. *loop*). Svako izvršavanje petlje zove se **prolaz** (engl. *pass*). U svakom prolazu se barem jednoj varijabli definiranoj unutar petlje pridružuju nove vrijednosti.

U naredbe ponavljanja ubrajaju se **`for`** petlja i **`while`** petlja. Kod **`for`** petlje najčešće se dio programa ponavlja unaprijed određeni broj puta dok se kod **`while`** petlje najčešće dio programa ponavlja dok se ne zadovolji uvjet. Broj ponavljanja dijela programa unutar **while** petlje nije unaprijed poznat kao što je to kod **for** petlje.

Izvršavanje obje vrste petlji može se prekinuti u svakom trenutku naredbom **`break`**.

`for` - end petlja

U računalnim znanostima `for` petlje omogućuju ponavljanje izvršavanja dijela koda. Obično se koriste za specificiranje unaprijed definiranog broja iteracija (broja ponavljanja odnosno broj prolaza kroz petlju).

U petljama tipa `for-end` izvršavanje naredbi ponavlja se zadani broj puta.

```
for k=f:s:t
...niz Matlab-ovih
...naredbi
end
```

k je kontrolna varijabla petlje, ***brojač*** petlje. Taj dio određuje koliko će se puta dio programa između **`for`** i **`end`** dijela izvršavati.

`f` je vrijednost **k** u prvom prolazu

`s` je ***korak*** povećanja **k** nakon svakog prolaza

`t` je vrijednost **k** u zadnjem prolazu

Petlja mijenja vrijednost brojača od vrijednosti **f** do vrijednosti **t** uz povećavanje vrijednosti određene vrijednošću **s**.

- U prvom prolazu **k**=**f** računalo izvodi naredbe između naredbi `for` i `end`. Zatim se program vraća na naredbu `for` da bi započeo sljedeći prolaz. Varijabla **k** dobiva novu vrijednost, jednaku **k** = **f** + **s**, a zatim se s tom novom vrijednošću **k** izvode naredbe između `for i end`. Postupak se ponavlja dok u posljednjem koraku **k** ne postane jednako **t**. U tom slučaju program se ne vraća na `for`, već nastavlja s naredbama iza naredbe end. Npr., ako je k = 1:2:9, imamo pet prolaza kroz petlju, a vrijednost **k** u svakom prolazu kroz petlju je 1, 3, 5, 7 i 9.
- Korak petlje, **s**, može biti i negativan (npr. k = 30:-5:15 daje četiri prolaza u kojima je k = 30, 25, 20, 15).
- Ako nije zadana vrijednost koraka petlje, **s**, podrazumijeva se 1.
- Svakoj naredbi for u programu mora biti pridružena i odgovarajuća naredba `end`!

- Kada se petlja završi, kontrolna varijabla (**k**) ima vrijednost koja joj je bila posljednja dodijeljena.

Primjer: Izračunajte kvadrate brojeva od 1 do 5 koristeći `for` petlju.

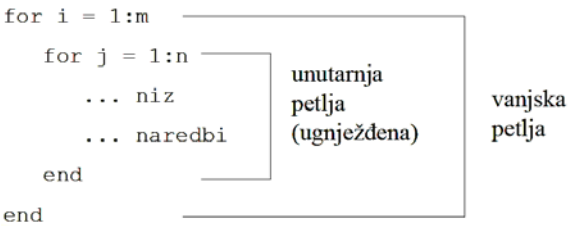
```
for x = 1:5
y = x^2; % kontrolna varijabla se može koristiti za računanje
fprintf('%g\n',y)
end
```

Primjer: Program izračunava zbroj brojeva od 1 do 50. **`for`** petlja se ponavlja 50 puta i pri svakom prolasku kroz petlju varijabli **x** dodaje se vlastita vrijednost iz prijašnjeg koraka i vrijednost brojača petlje **j**.

Na početku se varijabli **x** dodjeljuje vrijednost 0 kako bi zbroj na početku programa bio jednak nuli.

```
x = 0;
for j = 1:50
x = x + j;
end
x
```

Mogu se rabiti i tzv. **ugniježđene `for`** petlje čija je sintaksa sljedeća:



To znači da petlja može započeti (i završiti se) unutar druge petlje.

Primjer: Ugniježđena **for** petlja se sastoji od dvije **for** petlje. Dio programa unutar prve (vanjske) **for** petlje ponavlja se 10 puta, a isto tako i dio programa unutar druge (unutarnje) **for** petlje. Dio programa unutar unutarnje **for** petlje zbog prve se petlje ponavlja 100 puta. Pri svakom prolasku kroz drugu **for** petlju u odgovarajući se element (određen brojačima petlje **m** i **n**) matrice **x** upisuje umnožak prvog i drugog brojača petlje **m*n**.

*Program u matricu **x** dimenzija 10*10 upisuje vrijednosti tablice množenja brojeva od 1 do 10.*

```
for m = 1:10
for n = 1:10
x(m,n) = m * n;
end
end
x
```

`while-end` petlja

`while` petlja je naredba kontrolnog toka koja omogućuje ponavljanje izvršavanja koda na temelju zadanog logičkog uvjeta. Ovakav oblik petlje se rabi kad nije

unaprijed poznat broj ponavljanja petlje (broj prolaza), a petlja se izvršava dok ne bude ispunjen zadani uvjet.

Sintaksa **`while`** petlje ima sljedeći oblik:

```
while uvjet
naredba ili funkcija
naredba ili funkcija
...
end
```

Dio programa između **`while`** i **`end`** izvršavat će se sve dok je ispunjen uvjet. Kada uvjet više nije ispunjen, izvršavanje petlje se prekida.

Primjer: Izračunajte kvadrate brojeva od 1 do 5 koristeći `while` petlju.

```
x = 1;
while x<=5
y = x^2;
fprintf(' %g\n',y)
x = x + 1; % brojač
end
```

Primjer: Varijabli **x** dodjeljuje se vrijednost 100, a varijabli **y** vrijednost 1,5. U **while** petlji ispituje se je li vrijednost varijable **x** veća od 1 **ili** vrijednost varijable **y** manja od 100. Ako je ispunjen taj uvjet, izvršava se dio programa između **while** i **end**, odnosno u ovom slučaju se vrijednost varijable **x** dijeli sa 2, a vrijednost varijable **y** množi sa 3.

```
x = 100
y = 1.5
while (x>1) | (y<100)
x = x / 2
y = y * 3
end
```

*Program unutar **while** petlje izvršio se 7 puta odnosno vrijednost varijable **x** se 7 puta dijelila sa 2, a vrijednost varijable **y** se 7 puta množila sa 3. Prilikom petog prolaska kroz **while** petlju varijabla **y** nije bila manja od 100, ali je varijabla **x** bila veća od 1. Tek kada ni jedan uvjet nije bio ispunjen, izvršavanje programa se prekida.*

M-File funkcije

U MATLAB-u postoji mogućnost kreiranja vlastitih funkcija u formi M-datoteka pohranjenih na računalu. M-file funkcija je slična običnom M-file-u, tekst datoteci s ekstenzijom .m

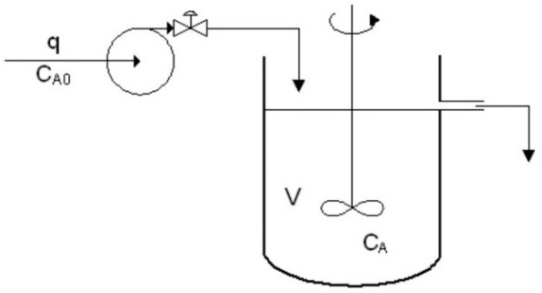
```
function  y = ime funkcije(x1,x2,...xn)
y =. . .
```

Ime funkcije i ime datoteke moraju biti identični! Npr., funkcija `primjer` mora biti pohranjena u datoteci `primjer.m`

Ako se upiše `primjer` u promptu, MATLAB prvo traži varijablu s tim imenom. Ako je nema, onda je traži kao ugrađenu funkciju. Na kraju provjerava u trenutnom direktoriju postoji li `primjer.m`. Ako se ne nalazi ni ovdje provjerava sve MATLAB direktorije da bi našao `primjer.m`

Zadatak:

Odredite prijelazni odziv koncentracije u **protočno kotlastom reaktoru**, c_A ako se koncentracija ulazne struje, c_{A0} trenutačno poveća za 0,92 mol/m^3 .



Zadani podaci:

q = 0,085 m^3/min - protok kroz reaktor
V = 2,1 m^3 - volumen reaktora
$\Delta c_{A,0}$ = 0,92 mol/m^3 - iznos skokomične promjene koncentracije tvari A na ulazu u reaktor
$c_{A,0}$ = 1,85 mol/m^3 - koncentracija tvari A u ulaznoj struji
t_{skok}= 5 min - vrijeme skokomične promjene

Početna koncentracija tvari A u reaktoru jednaka je koncentraciji na ulazu u reaktor
→ $c_{A,poč} = c_{A0} = c_A$ = 1,85 mol/m^3

Iz bilance tvari slijedi: $\frac{dn_A}{dt} = \dot{n}_d - \dot{n}_o$

$$V \cdot \frac{dc_A}{dt} = q \cdot c_{A0} - q \cdot c_A$$

Poznato je analitičko rješenje za odziv procesa prvog reda na skokomičnu promjenu ulazne veličine:

$$C_A = C_{A,poc} + \Delta C_{A,poc} \cdot k \cdot \left(1 - e^{\frac{-t}{\tau}}\right)$$

Pri čemu je k=1, a vremenska konstanta $\tau = V/q$

Program:

```
clear all;                  % brise radni prostor
clc;                        % cisti ekran

%  Zadani procesni podaci
V = 2.1;                    % volumen reaktora [m3]
q = 0.2;                    % protok [m3/min]
% V = input('Unesi vrijednost volumena:');
% q = input('Unesi vrijednost protoka:');
caPoc = 1.85;               % pocetna koncentracija u reaktoru
deltaca0 = 0.92;            % iznos skokomicne promjene

% Podaci za simulaciju
t = 0;                % pocetno vrijeme je nula
tend = 120;           % konačno vrijeme
delt = 0.5;           % korak integracije (iznos jednog vrem. koraka)
n =(tend-t)/delt;     % broj vrem. koraka
```

```
tSkok = 5;          % trenutak kad nastupa promjena

% SIMULACIJA

for cnt = 1:n      % vrem. petlja (od 1 do 120 min, 1 korak 0.5 min)
    t = t + delt;
    ca = caPoc;     % pocetna koncentracija u reaktoru

    % konc. u reaktoru nakon skokomicne promjene konc. na ulazu
    if t >= tSkok
          ca = caPoc + deltaca0*(1 - exp (-(t-tSkok)/(V/q)));
    end
%  Pohrana rezultata za graficki prikaz (u obliku matrica)
    CA(1,cnt) = ca;
    T(1,cnt) = t;
end

%   Graficki prikaz rezultata
plot (T,CA)   % odziv (izlaz)
xlabel ('vrijeme, min')
ylabel ('Ca, mol/m3')
title ('Koncentracija u reaktoru Ca')

fprintf('Koncentracija:\n ')
fprintf('%.2f \n',CA)
```

Crtanje i grafički prikazi

`plot(x,y)`

Naredba `plot` automatski određuje granice osi, označuje pojedine točke i povezuje ih ravnom linijom. Različite opcije naredbe `plot` omogućavaju crtanje više funkcija ili niza podataka na istom grafu, uporabu različitih tipova i boja krivulja, mogućnost označavanja samo točaka bez linija. Osi se mogu imenovati, staviti naslov iznad slike, stavljati crtice na osi itd.

npr.
`plot(x,y,x,z)` - crtanje dviju funkcija `y` i `z` u ovisnosti od `x` na istom grafu
`plot(x,y,x,y, '+')` - crta funkciju `y` dva puta jednom linije, a drugi puta "+"

`grid` - crtanje mreže isprekidanih linija unutar grafa

```
xlabel('tekst ispod osi x')
ylabel('tekst ispod osi y')
title('naslov iznad grafa')
```

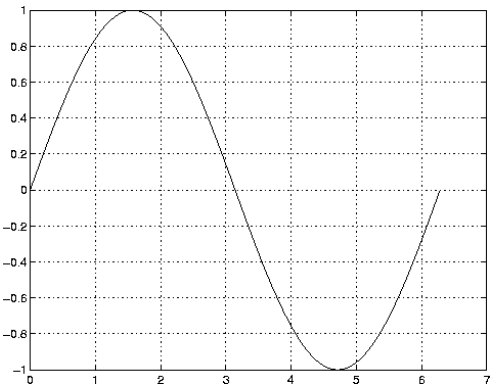
Crtanje grafova

Funkcija `plot` ima različite oblike, što ovise o ulaznom argumentu. Ako je `y` vektor

`plot(y)` daje linearni graf `y` u ovisnosti o indeksu elementa `y` , Ako se postave dva vektora kao argumenti, `plot(x,y)` crta graf `y` vs. `x`.

Da bi nacrtali funkciju sinus na segmentu od 0 do 2. Napisat ćemo:

```
t = 0:pi/100:2*pi;
y = sin(t);
plot(t,y)
```

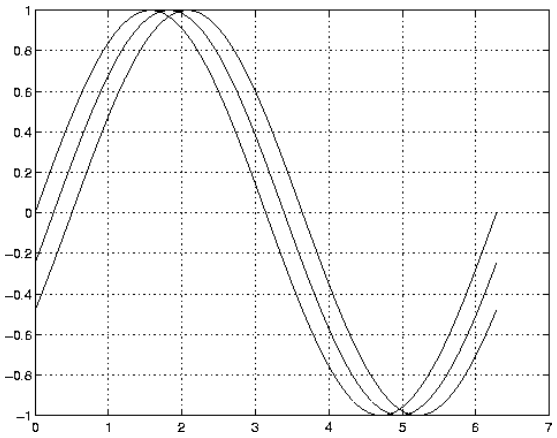


Izrazi koji omogućuju crtanje tri funkcije od `t`, pri čemu je svaka krivulja drugačije boje:

```
y2 = sin(t-.25);
y3 = sin(t-.5);
plot(t,y,t,y2,t,y3)
```

Moguće je odrediti boju, stil linije i markere (npr. kružići ili križići) naredbom

```
plot(x,y,'color_style_marker')
```



`color_style_marker` je 1-, 2- ili 3-znakovni karakter (odvojen s jednostrukim navodnicima) koji se sastoji od boje, stila linije i vrste markera:

* Color = `'c'`, `'m'`, `'y'`, `'r'`, `'g'`, `'b'`, `'w'`, and `'k'`.
(cyan, magenta, yellow, red, green, blue, white, and black)

    * Linestyle = `'-'` puna linija, `'--'` isprekidana, `':'` točkasta, `'-.'` točka-crta, i `'none'` bez crtanja linije

* Uobičajeni markeri su `'+'`, `'o'`, `'*'`, i `'x'`.

Npr., naredba

```
plot(x,y,'y:+')
```

crta žutu točkastu liniju i svaku točku označava križićem, ako je definirani marker a nije linija MATLAB crta samo markere.

Grafički prozor

Funkcija `plot automatski otvara novi prozor s grafičkim prikazom ako već ne postoji otvoreni prozori na ekranu.` Da bi otvorili novi prozor za grafički prikaz i da bi postao tekućim:

```
figure
```

Da bi učinili postojeći prozor s grafičkim prikazom tekućim:

```
figure(n)
```

pri čemu je `n` broj slika na traci s naslovom.

Dodavanje crteža na postojeći grafički prikaz

Naredba `hold` daje mogućnost dodavanja crteže na postojeći grafički prikaz:

```
hold on
```

MATLAB ne uklanja postojeći grafički prikaz, nego dodaje nove podatake na aktivni. Npr. slijedeći niz naredbi prvo stvara konturni prikaz `peaks` funkcije, a zatim dodaje pseudoobojan prikaz iste funkcije:

```
[x,y,z] = peaks;
contour(x,y,z,20,'k')
hold on
pcolor(x,y,z)
shading interp
```

Naredba `hold on` omogućuje da se `pcolor` prikaz kombinira s `contour` prikazom na jednoj slici.

Crtanje 3D grafova

U sljedećem primjeru funkcija `peaks` će biti prikazana najčešćim naredbama za trodimenzionalno crtanje MATLABu.

```
[x,y,z] = peaks;
```

```
plot3(x,y,z)
```

Naredba `plot3` u suštini radi isto što i `plot` funkcija – spaja setove koordinata linijama, ali u ovome slučaju koordinate u tri smijera *x, y* i *z*. Kao i kod plot funkcije bitno je da su sve tri varijable vektori iste dužine.

```
scatter3(x,y,z)
```

Analogno dvodimenzionalnoj inačici, `scatter3` crta točke na specificiranim lokacijama zadanih vektorima *x, y* i *z*.

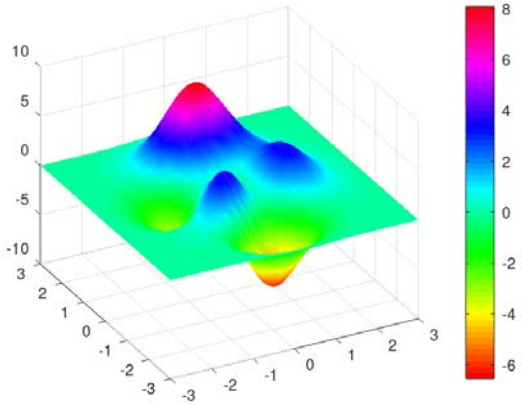
```
mesh(x,y,z)
```

Naredba `mesh` stvara mrežu između točaka tako linijama stvarajući površine - četverokute koji svojim oblikom nagovještavaju trodimenzionalni smijer kretanja funkcije.

```
surf(x,y,z)
```

Naredba `surf` nadogradnja je na prikazanu naredbu `mesh`, gdje su površine obojane te intuitivno prikazuju vrijednosti veće i manje od nule.
Moguće je također dodati drugačiju skalu boja naredbom `colormap`, te odnos brojčane vrijednosti s bojama naredbom `colorbar`. Radi lakšeg prikaza mogu se i maknuti linije naredbom `shading interp`:

```
[x,y,z] = peaks;
surf(x,y,z)
colormap hsv
colorbar
shading interp
```

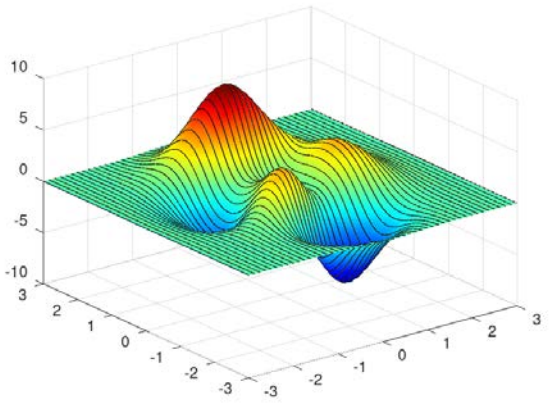


Neke od opcija `colorbar` naredbe: `jet, winter, summer, hsv, turbo, lines, colorcube.`

Za 3D crteže također vrijede navedena pravila dodavanja crteža na postojeći graf.

```
[x,y,z] = peaks;
surf(x,y,z)
colormap jet
shading interp
```

```
hold on
plot3(x,y,z, 'k')
```



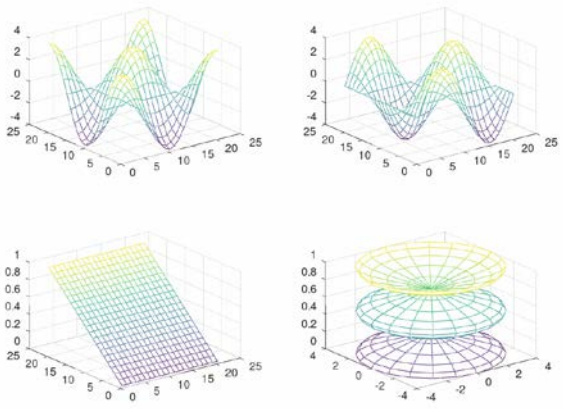
Subplot

Funkcija `subplot` omogućuje prikaz više grafova u istom prozora ili tiskanje takvih grafova na jednom papiru:

```
subplot(m,n,p)
```

prelama sliku u `m` x `n` matricu manjih grafova, a `p` se navodi kao aktivni graf. Grafovi se broje duž prvog reda u prozoru, zatim drugi red itd. Slijedeći primjer pokazuje kako bi nacrtali podatke na četiri grafa unutar prozora:

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1)
mesh(X)
subplot(2,2,2); mesh(Y)
subplot(2,2,3); mesh(Z)
subplot(2,2,4); mesh(X,Y,Z)
```



Još neke mogućnosti:

```
subplot (1,1,1)
```

briše sve osi i vraća na početni postav sa jednim oknom

```
pause
```

privremeno prekida izvođenje programa (pritiskom na bilo koju tipku izvođenje se nastavlja, dok

```
pause(n)
```

zaustavlja prikaz na *n* sekundi pa potom nastavlja.

Kontrola osi

Funkcija `axis` ima brojne opcije za podešavanje skaliranja, orijentacije i aspektni odnos crteža.

MATLAB obično traži maksimume i minimume danih podataka i prema tome bira odgovarajuće područje na osima. Funkcijom `axis` sami ugađamo granice osi:

```
axis([xmin xmax ymin ymax])
```

`axis` isto tako prihvaća ključne riječi za kontrolu osi, npr.

```
axis square
```

postavlja iste duljine za čitave *x i y*-osi, a

```
axis equal
```

postavlja iste duljine crtica i na *x*- i *y*- osi .

Izraz `grid off` isključuje grid lines, dok ih

`grid on` ponovno uključuje.

Literatura

1. D. Grundler, T. Rolich, A. Hursa. **MATLAB i primjena u tekstilnoj tehnologiji**: Sveučilište u Zagrebu, Tekstilno-tehnološki fakultet, Zagreb, 2010.
2. M. Markić, Ž. Ujević Andrijić. Primjena i programiranje računala - materijali s predavanja.