FAKULTET ELEKTROTEHNIKE
STROJARSTVA I BRODOGRADNJE

FACULTY OF ELECTRICAL ENGINEERING,
MECHANICAL ENGINEERING
AND NAVAL ARCHITECTURE

Rudjera Boškovića bb, Split

LABORATORIJ ZA BIOMEHANIKU
AUTOMATIKU I SUSTAVE

LABORATORY FOR BIOMECHANICS
AND AUTOMATIC CONTROL SYSTEMS

KOLEGIJ
SIGNALI I SUSTAVI

OSNOVE MATLABA

Vježba br. 1.

Ak. god. 2008/09.

UVOD

Matlab je interaktivni program za inženjerske i znanstvene proračune. Služi za rješavanje različitih matematičkih problema, te izračunavanja i simulacije vezane uz identifikaciju, upravljanje i regulaciju sustava.

Uz osnovni programski sustav postoje i brojni dodatni programski paketi (toolbox-ovi) koji pokrivaju mnoga područja inženjerske djelatnosti: automatsko upravljanje, identifikaciju sustava, statističke analize, analizu sustava u vremenskoj i frekvencijskoj domeni, simboličku matematiku, obradu signala i slike, 2D i 3D grafičke prikaze i brojne druge. Jedan od značajnijih paketa je Simulink - vizualni alat pomoću kojeg je moguće simulirati kontinuirane i diskretne sustave. Simulink koristi funkcijske blok dijagrame čijim se povezivanjem kreiraju sustavi koje simuliramo. Ovakva primjena programa olakšava rad korisniku jer ne zahtijeva detaljno poznavanje sintakse programskog jezika.

Većina Matlab-ovih naredbi i funkcija, kao i komunikacija između korisnika i programa neovisni su o operacijskom sustavu na kojem pokrećemo Matlab.

U Matlabu je moguće raditi na dva načina. Jedan je direktan, pri čemu se u glavnom prozoru programa upisuje naredba, te program odmah vraća rezultat. Ovaj način prikladan je kada radimo jednostavne operacije bez ponavljanja.

Kod drugog načina, korisnik u tekstualnom editoru (Matlab editor/debbuger) piše program koji se sastoji od slijeda naredbi i pohranjuje ga kao m-datoteku (datoteka s .m ekstenzijom). Kad u komandnom prozoru Matlaba pozovemo ime te datoteke, pokrenut će se izvršavanje programa tj. izvršavanje slijeda naredbi pohranjenih u datoteci.

RAD U MATLAB-u

Matlab sve podatke tretira kao matrice; čak se i skalarne veličine smatraju matricama s dimenzijom 1×1. Matlab podržava realne i kompleksne brojeve i matrice. Nakon pokretanja programa, otvara se glavni prozor u kojem ćemo uočiti znak >> (prompt), koji označava da Matlab očekuje unos nove naredbe. Svaka naredba mora završiti tipkom *Enter* – u nastavku teksta koristit ćemo oznaku <ent>.

U svojoj najjednostavnijoj primjeni, Matlab može služiti kao kalkulator. Za početak, krenimo s nekoliko jednostavnih operacija:

```
>> 5+8          <ent>
ans =
13
>> 3*6-14/7   <ent>
ans =
   16
>>
```

Rezultat se pojavljuje odmah u slijedećem retku (ans = 13; pri čemu je ‘ans’ kratica za answer), a nakon njega slijedi oznaka za unos nove naredbe. Uočimo da Matlab poštuje matematički redoslijed operacija: prvo se izvršava potenciranje, zatim množenje i dijeljenje, te na kraju zbrajanje i oduzimanje.

Osnovne aritmetičke operacije su:

+	zbrajanje
-	oduzimanje
*	množenje
/	"desno" dijeljenje
\	"lijevo" dijeljenje
^	potenciranje
<	manje
<=	manje ili jednako
>	veće
>=	veće ili jednako
= =	jednako
~=	nije jednako

Osnovni logički operatori su:

&	logičko I
|	logičko ILI
~	ligičko NE

Kod logičkih izraza, Matlab za istinit rezultat daje 1, a u suprotnom 0. Npr:

```
>>5<=9
ans=
1
>>9<=5
ans=
0
```

Uočimo razliku između '= =' i '='. Znak '= ='predstavlja znak jednakosti i koristi se u logičkim operacijama, dok '=' predstavlja znak pridruživanja i njime varijabli pridružujemo vrijednost. Npr:

```
>> a=1; b=3;       % pridruživanje vrijednosti varijabli
>> if a= =b        % logički izraz kojim provjeravamo jednakost varijabli a i b
     disp ('a je vece od b')
else
     disp ('b je vece od a')
end

b je vece od a      % rezultat izvršene if naredbe
>>
```

Komentari se u Matlab-u označavaju s %. Ukoliko ne želimo da se rezultat naredbe ispiše neposredno nakon unosa i izvršenja naredbe, koristit ćemo znak ';'. Npr:

```
>> 2*5
ans =
   10
>> 3*8 ;    % zbog ';' rezultat se neće neposredno ispisati.
>>
```

Rezultat posljednje unesene i izvršene naredbe pohranjuje se u varijablu 'ans' koju Matlab automatski generira. Vrijednost varijable ans možemo provjeriti ako unesemo ime varijable:

```
>> ans      <ent>
ans =
   24
>>
```

VARIJABLE I MATRICE

Pogledajmo sad kako se unose matrice. Neka su npr. zadane matrice A i B (dimenzija 3*3 i 3*1)

$$A = \begin{bmatrix} -2 & -5 & -9 \\ 1 & 6 & 0 \\ 0 & 1 & 12 \end{bmatrix}; \qquad B = \begin{bmatrix} 4 \\ 5 \\ 3j \end{bmatrix}$$

Za unošenje matrica koriste se uglate zagrade i točka-zarez, na slijedeći način:

```
>> A=[-2  -5  -9; 1 6  0; 0  1  12]                <ent>
A =
   -2    -5    -9
    1     6     0
    0     1    12

>> B=[4  5  3j]                                     <ent>
B =
  4.0000          5.0000              0 + 3.0000i
>>
```

Točka-zarez ';' razdvaja retke matrice, a razmak (ili zarez) elemente istog retka. Matrice možemo unositi i tako da umjesto znaka ';' idemo u novi red, na način:

```
>> A=[-2  -5  -9     <ent>
 1  6   0             <ent>
 0  1  12]            <ent>

A =
   -2    -5    -9
    1     6     0
    0     1    12
>>
```

Korisnik može sam kreirati varijable i pridruživati im vrijednosti. Dosad smo kreirali varijable tj. matrice A i B. Kreirajmo još nekoliko realnih i kompleksnih varijabli:

```
>> x=35     <ent>
x =
   35

>> y=3i      <ent>
y =
       0 + 3.0000i

>> z=5+9j +y   <ent>
z =
  5.0000 + 12.0000i
```

Imaginarna jedinica označava se slovima i ili j. Matlab razlikuje mala i velika slova, te su stoga, npr. varijable a i A ili x i X međusobno različite. Napomenimo da se naredbe u Matlabu pišu isključivo malim slovima. Imena varijabli mogu se sastojati od kombinacija slova i brojeva s tim da prvi znak mora biti slovo.

NAREDBE *WHOS, CLEAR* I *HELP*

Upoznajmo sada naredbu *whos*. Ova naredba daje nam popis svih varijabli koje se trenutno nalaze u radnom prostoru (workspace-u), njihove dimenzije te koliko memorije zauzimaju. Npr., budući da smo dosad kreirali varijable A, B, x, y i z, rezultat naredbe who biti će:

```
>> whos   <ent>
  Name  Size         Bytes         Class

  A       3x3           72         double array
  B       1x3           48         double array (complex)
  x       1x1            8         double array
  y       1x1           16         double array (complex)
  z       1x1           16         double array (complex)
```

Želimo li izbrisati neku varijablu iz radnog prostora, koristit ćemo naredbu *clear*. Npr:

```
>> clear A     <ent>                % briše varijablu A
>> clear all    <ent>               % briše sve varijable iz radnog prostora
```

Ako želimo ispisati vrijednost neke postojeće varijable, unijet ćemo njeno ime (bez znaka ;):

```
>> B              <ent>
B =
  4.0000          5.0000              0 + 3.0000i
```

Vrlo korisna u radu bit će nam naredba *help*. Ova naredba ispisuje detaljne upute za korištenje pojedine naredbe, kao i argumente koje naredba prihvaća. Npr:
>> help clear

>> help whos

Matlab pamti i određeni broj prethodno izvršenih naredbi. Do prethodnih naredbi dolazi se pritiskom na tipku ↑. Ako znamo da izraz kojeg smo već prije izveli započinje s npr. 'a', možemo napisati nekoliko slova kojima izraz započinje i zatim koristiti tipku ↑. U tom slučaju Matlab ispisuje samo one naredbe koje započinju s već napisanim znakovima. Npr:

```
>> a                          <↑>
>> a=[1 2 3 4 5]              <↑>
>> atan(30*pi/180)            <↑>
>> a=conv([1 3 3],[1 -4])     <↑>      %  itd. Kad dođemo do željene naredbe, editiramo je
                                       %   i izvršimo
```

KONSTANTE

U Matlab-u je definirano više konstanti tj. varijabli s predefiniranom vrijednošću:

pi	3.14159265
i	imaginarna jedinica
j	imaginarna jedinica
realmax	najveći pozitivan realni broj
realmin	najmanji pozitivni realni broj
inf	infinity (rezultat djeljenja s nulom)
nan	not-a-number (npr. rezultat dijeljenja 0/0 ili inf/inf)

NAREDBA *FORMAT*

Matlab računske operacije izvodi u dvostrukoj preciznosti (15 značajnih znamenki), a način ispisivanja određuje naredba *format*.

format short (ili samo format)	4 decimalna mjesta
format short e	eksponencijalni prikaz sa 4 decimalna mjesta
format long	14 decimalnih mjesta
format long e	eksponencijalni prikaz sa 14 decimalnih mjesta
format rat	aproksimacija brojeva razlomkom
format compact	ispis bez praznih redova

Naredba *help format* navest će sve moguće načine ispisa.

VEKTORI

Veoma često ćemo u radu koristiti matrice s jednim retkom ili stupcem koje nazivamo redni i stupčasti vektori. Vektore unosimo kao klasične matrice. Npr:

```
>> redni_vektor=[1 4 7 2]
redni_vektor =
    1    4    7    2

>> stupcasti_vektor=[3; 6; 9; 7]
```

```
stupcasti_vektor =

    3
    6
    9
    7
>>
```

Često će nam vektor predstavljati rastući ili padajući niz brojeva s jednolikim razmakom između susjeda. Takav vektor zadajemo pomoću znaka dvotočke (:). Npr:

```
>> t=3:0.5:6
t =
   3.0000    3.5000    4.0000    4.5000    5.0000    5.5000    6.0000
```

Prvi broj u definiciji vektora (3) je početna vrijednost, treći je završna vrijednost niza (6), dok drugi broj predstavlja korak tj. razmak između susjeda (0.5). Padajući niz zadaje se negativnom vrijednošću koraka. Ako korak nije naveden, podrazumijeva se vrijednost 1.

```
>> padajuci_niz=100:-10:20
padajuci_niz =
  100    90    80    70    60    50    40    30    20
```

PRISTUP ELEMENTIMA MATRICE

Pojedinim elementima matrice pristupa se korištenjem okruglih zagrada:

```
>>  a = [1  5  9; 7  9  5; 9  45  8]      <ent>
a =
    1     5     9
    7     9     5
    9    45     8

>>  a(3,2)       <ent>     % ispis elementa matrice a u trećem retku i drugom stupcu
ans =
45
```

Pored pojedinog elementa, iz matrice je moguće izdvojiti pojedini redak, stupac ili čitavu podmatricu na slijedeći način:

```
>> drugi_redak=a(2,:)   <ent>    %  spremanje drugog retka matrice a u novu matricu pod
                                  %   nazivom drugi_redak
drugi_redak =
    7     9     5

>> treci_stupac=a(:,3)   <ent>    %  spremanje trećeg retka matrice a u novu matricu pod
                                  %   nazivom treci_stupac
treci_stupac =
    9
```

```
     5
     8
>> podmatrica=a(1:2,:)    <ent>  %  kreiranje varijable podmatrica koja se sastoji od prvog i
                                  %  drugog retka matrice (1:2) a i svih stupaca (označeno
                                  %  dvotočkom : )
podmatrica =
     1     5     9
     7     9     5
```

Operator ':' bez zadane početne i završne vrijednosti izdvaja sve elemente nekog retka ili stupca.

OSNOVNE MATEMATIČKE OPERACIJE S MATRICAMA

Primijenimo na matricama osnovne matematičke operacije +, -, * i /. Kod zbrajanja i oduzimanja matrice moraju biti istih dimenzija:

```
>> a=[1 2 3; 4 5 6];
>> b=[7 8 9; 10 11 12];
>> c=a+b
c =
     8    10    12
    14    16    18
```

U slučaju nepoklapanja dimenzija matrica, Matlab će javiti pogrešku:

```
>> d=[1 2; 3 4];
>> e=a+d
??? Error using ==> +
Matrix dimensions must agree.
>>
```

Kod množenja matrica broj stupaca lijeve matrice mora biti jednak broju redaka desne matrice:

```
>> a=[1 2 3; 4 5 6];              % dva retka i tri stupca
>> b=[7 8; 9 10; 11 12];          % tri retka i dva stupca
>> a*b
ans =                             % rezultat ima dva retka i stupca
    58    64
   139   154

>> b*a                        % podsjetimo se, množenje matrica nije komutativno (a*b ≠ b*a)
ans =
    39    54    69
    49    68    87
```

Množenje matrice sa skalarom djeluje tako da svaki element matrice biva pomnožen sa skalarom:

```
>> a=[1 2 3; 4 5 6];
>> 8*a
ans =
    8   16   24
   32   40   48
```

Također, kod zbrajanja sa skalarom, skalar se dodaje svakom elementu matrice:

```
>> a+3
ans =
    4    5    6
    7    8    9
```

Determinanta matrice nalazi se naredbom *det()*:

```
>> x=[4 7 2; 4 8 23; 3 5 7];          % matrica mora biti kvadratna
>> det(x)
ans =
   43
```

Inverzna matrica nalazi se naredbom *inv()*:

```
>> inv(x)
ans =
  -1.3721   -0.9070    3.3721
   0.9535    0.5116   -1.9535
  -0.0930    0.0233    0.0930
```

Za inverzne matrice vrijedi: x*inv(x) = inv(x)*x = I (jedinična matrica)
Kod traženja inverza, matrica ne smije biti singularna, tj. det(x) ne smije biti nula.
Transpozicija matrice vrši se operatorom ' na način:

```
>> x =
    4    7    2
    4    8   23
    3    5    7

>> x_transp=x'
x_transp =
    4    4    3
    7    8    5
    2   23    7
```

Potenciranje matrice cijelim brojem odgovara uzastopnom množenju matrice same sa sobom.
Pri tome matrica mora biti kvadratna:

```
>> x^3          % odgovara slijedecem:
>>x*x*x
```

OPERACIJE PO ELEMENTIMA

Matematičke operacije možemo izvoditi i između pojedinačnih elemenata matrica, po načelu "svaki element prve matrice s korespondirajućim elementom druge matrice". Tada prije željenog matematičkog operatora stavljamo točku . (.*, ./, .^, itd.):

```
>> A=[1 5 10];
>> B=[1 2 3];
>> C=A .*B                  %   točka prije operatora množenja
C =
    1    10    30
```

Uočimo da je ovakvo množenje različito od matričnog množenja. Još nekoliko primjera operacija po elementima:

```
>> A./B                              % rezultat daje kvocijente 1/1 5/2 10/3
ans =
   1.0000    2.5000    3.3333
>> A.^B                               % rezultat daje potencije 1^1, 5^2, 10^3
ans =
         1        25      1000
>> A.^3                              % rezultat daje potencije sa skalarom 1^3, 5^3, 10^3
ans =
         1       125      1000
```

POSEBNE MATRICE

Naredba *zeros(m,n)* vraća matricu sa m redaka i n stupaca popunjenu nulama, dok funkcija *ones(m,n)* vraća matricu istih dimenzija popunjenu jedinicama. Naredba *eye(n)* kreirat će kvadratnu matricu dimenzija n×n s jedinicama po dijagonali (jedinična matrica). Matricu sa slučajno odabranim brojevima (u intervalu od 0 do 1) kreirat će naredba *rand(n,m)*:

```
>> zeros(3,2)
ans =
    0    0
    0    0
    0    0

>> ones(2,6)
ans =
    1    1    1    1    1    1
    1    1    1    1    1    1

>> eye(4,4)
ans =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
```

```
>> rand(3,2)
ans =
   0.8913    0.0185
   0.7621    0.8214
   0.4565    0.4447
```

Uočimo iz prethodnog primjera da ako Matlab naredba ima više argumenata, oni se međusobno odvajaju zarezom.

JOŠ NEKE OPERACIJE NAD MATRICAMA

Naredba *max(a)* kao rezultat daje redni vektor čiji elementi odgovaraju maksimalnim vrijednostima stupaca matrice a. Analogno vrijedi za naredbu *min(a)*:

```
>> a=rand(4,5)
a =
   0.6154    0.1763    0.4103    0.8132    0.1987
   0.7919    0.4057    0.8936    0.0099    0.6038
   0.9218    0.9355    0.0579    0.1389    0.2722
   0.7382    0.9169    0.3529    0.2028    0.1988

>> max(a)
ans =
   0.9218    0.9355    0.8936    0.8132    0.6038

>> min(a)
ans =
   0.6154    0.1763    0.0579    0.0099    0.1987
```

U slučaju vektora, naredbe *max()* i *min()* daju najveću tj. najmanju vrijednost vektora:

```
>> redni_v=[1 4 23 78 -45 0]
redni_v =
    1     4    23    78   -45     0
>> max(redni_v)
ans =
   78
```

Kod vektora, naredba *sum()* računa sumu svih elemenata vektora. Kod matrica, rezultat naredbe biti će redni vektor čiji će svaki element predstavljati sumu pojedinog stupca matrice, npr:

```
>> vektor=[1 2 3 4 5];
>> sum(vektor)
ans =
   15

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

```
>> sum(matrica)
ans =

   12    15    18
```

Naredba *size()* za rezultat daje dimenziju matrice, a naredba *length()* daje duljinu tj. broj elemenata vektora:

```
>> [br_redaka, br_stupaca]=size(matrica)        <ent>
br_redaka =
    3
br_stupaca =
    3

>> length(vektor)                                <ent>
ans =
    5
```

MATEMATIČKE FUNKCIJE

Matlab ima biblioteku nekih matematičkih funkcija:

IME FUNKCIJE	ZNAČENJE
sin ( )	sinus
cos ( )	kosinus
tan ( )	tangens
asin ( )	arkus sinus
acos ( )	arkus kosinus
atan ( )	arkus tangens
sinh ( )	hiperbolni sinus
cosh ( )	hipebolni kosinus
tanh ( )	hiperbolni tangens
asinh ( )	area sinus hiperbolni
acosh ( )	area kosinus hiperbolni
atanh ( )	area tangens hiperbolni
abs ( )	apsolutna vrijednost
sqrt ( )	kvadratni korijen
real ( )	realni dio kompleksnog broja
imag ( )	imaginarni dio kompleksnog broja
conj ( )	kompleksno konjugiranje
exp ( )	exponencijalna funkcija (baza e)
log ( )	logaritamska funkcija (baza e)
log10 ( )	dekadski logaritam

Argument ovih funkcija može biti skalar ili matrica. U slučaju matrice, funkcija se izvršava na svakom elementu posebno, npr:

```
>> a=[-3  4  -5;  6  -8  9];
>> abs(a)
ans =
    3    4    5
    6    8    9
```

Napomena: trigonometrijske funkcije kao argument uzimaju kut u radijanima. Npr:

```
>> cos (pi)
ans =
   -1

>> cos (180)
ans =
  -0.5985

>> cos (180*pi/180)
ans =
   -1
```

Detaljan popis funkcija ispisat će se naredbom *help elfun.*

POLINOMI

U teoriji sustava i automatici prijenosne funkcije sustava definiraju se razlomcima s polinomima u brojniku i nazivniku. Stoga ćemo pokazati kako se u Matlabu definiraju polinomi. Neka je zadan polinom:

$p=s^5 + 3s^2 – 4s + 21$

Polinom se zadaje u obliku matrice tako da se u matricu zapišu koeficijenti polinoma, od koeficijenta uz najveću potenciju prema najmanjoj:

```
>>  polinom=[1 0 0 3 –4 21]          %   uočiti, koeficijenti uz potencije 4 i 3 su 0
```

Korijeni polinoma traže se naredbom *roots()*:

```
>> r=roots(polinom)
r =
  -2.1217
   1.3843 + 1.1254i
   1.3843 - 1.1254i
  -0.3235 + 1.7335i
  -0.3235 - 1.7335i
```

Naredba *poly()* računa koeficijente polinoma na temelju zadanih korijena. Pritom se i korijeni zadaju u matrici. Npr, nađimo polinom čiji su korijeni: 1, 2 i 0:

```
>> korijeni=[1 2 0];
```

```
>> polinom = poly (korijeni)

polinom =
    1    -3     2     0
```

Dakle, polinom će biti: polinom=s^3-3s^2+2s.

Naredba *polyval()* računa vrijednosti polinoma u zadanim točkama na slijedeći način:

```
>> % opći oblik naredbe:
>> % y= polyval (p, x)
>> % y vrijednosti polinoma u točkama sadržanim u matrici x
>> % matrica p sadrži koeficijente polinoma
>> % primjer:

>> polinom=[1 -3 2 0]            % polinom iz prethodnog primjera
>> vrijednosti_polinoma= polyval (polinom, [1 2 3 4 5])

vrijednosti_polinoma =
    0     0     6    24    60          % vrijednosti polinoma u točkama 1,2,3,4 i 5
```

Polinomi se množe naredbom *conv()* :

```
>> p1=[1 3 2];
>> p2=[4 2];

>> umnozak=conv(p1,p2)         % istovremeno se mogu pomnoziti samo dva polinoma
umnozak =
    4    14    14     4
```

Za dijeljenje polinoma služi naredba *deconv()*:

```
>> [n,r]=deconv(p1,p2)                %   n je kvocijent, a r ostatak dijeljenja

n =
   0.2500    0.6250
r =
         0         0    0.7500
```

pri čemu je n kvocijent, a r ostatak dijeljenja, tj. vrijedi: $p_1/p_2 = n + r/p_2$

UČITAVANJE I SPREMANJE PODATAKA

Matlab omogućava spremanje varijabli u datoteke, kao i njihovo učitavanje. Najjednostavnija naredba za pohranjivanje varijabli iz radnog prostora (work space-a) jest *save*. Ovom naredbom sve varijable trenutno prisutne u radnom prostoru pohranit će se u datoteku koju će Matlab nazvati *matlab.mat* i smjestiti je u work direktorij (ili u direktorij u kojemu se trenutno nalazimo).

Često varijable želimo pohraniti u datoteku pod željenim imenom, npr:

```
>>   save ime_datoteke      %  sada će Matlab kreirati datoteku pod imenom
                            %   ime_datoteke.mat i u nju pohraniti sadržaj radnog prostora
```

Učitavanje sadržaja datoteke u radni rostor izvršava se naredbom *load*:

```
>>   clear all                %  oslobađa se radni prostor
>>   load ime_datoteke        %  u radni prostor se učitavaju se sve varijable sadržane u
                              %  datoteci ime_datoteke.mat
```

U datoteke se mogu pohranjivati i točno određene varijable, umjesto čitavog sadržaja radnog prostora. Npr, kreirajmo neke dvije varijable a i b te ih pohranimo u datoteku:

```
>> a=eye(3,7);
>> b=cos(3*pi/2);
>>  save  moja_datoteka  a  b                  %  pohranjujemo  varijable  a  i  b  u  datoteku
moja_datoteka.mat
```

Želimo li vidjeti sadržaj datoteke, bez njena učitavanja u radni prostor, koristit će nam naredba *whos -file ime_datoteke*. Npr:

```
clear all
>> whos -file moja_datoteka
  Name      Size            Bytes  Class
  a         3x7               168  double array
  b         1x1                 8  double array
Grand total is 22 elements using 176 bytes
```

Želimo li dodati još neku naredbu u već postojeću datoteku, upotrijebit ćemo naredbu *save ime_datoteke ime_varijable –append*. Npr:

```
>> c=[1 2 3 4 5];
>> save moja_datoteka c -append      %  u datoteku moja_datoteka.mat pohranili smo još i
                                     %  varijablu c.
```

Provjerimo sad sadržaj datoteke:

```
>> whos -file moja_datoteka            % datoteka će imati varijable a, b, c:
  Name      Size            Bytes  Class
  a         3x7               168  double array
  b         1x1                 8  double array
  c         1x5                40  double array
```

Također, iz datoteke možemo u radni prostor učitati samo pojedine željene varijable.
Npr, iz datoteke moja_datoteka želimo učitati samo varijable c i b:

```
>> clear all
>> load moja_datoteka c b
```

Ako želimo, pored spremanja varijabli u datoteke, spremiti i bilješke onoga što smo radili (tako da naknadno možemo pratiti tijek interaktivnog rada u Matlabu), koristit će nam naredba *diary*. Naredba će u tekstualnu datoteku pohraniti čitavi tekst koji se ispisao u glavnom prozoru Matlaba za vrijeme našeg rada, počevši od izvršenja naredbe pa sve do izvršenja naredbe *diary off*:

```
>> diary danasnji_rad    <ent>

….. slijedi rad čiji će zapis biti pohranjen u datoteku danasnji_rad.txt …..

>> diary off
```

GRAFIČKI PRIKAZ REZULTATA

Za crtanje tj. grafički prikaz najčešće se koristi naredba *plot()*. Matlab grafičke prikaze crta u grafičkim prozorima. Novi grafički prozor otvara se naredbom *figure*. U slučaju da ne postoji aktivni grafički prozor, Matlab će jedan otvoriti. Možemo imati više otvorenih grafičkih prozora, pri čemu Matlab crta u trenutno aktivnom prozoru. Želimo li crtati u neki određeni prozor, potrebno ga je prije crtanja odabrati, tj. učiniti aktivnim; zato svaki grafički prozor ima svoj identifikacijski broj kojeg naredba figure koristi prilikom odabira aktivnog prozora, npr:

```
>> figure                  % kreira prozor po rednim brojem 1
>> figure (2)              % kreira prozor po rednim brojem 2
>> figure (3)              % kreira prozor po rednim brojem 3
```

Naredba *plot(x)* crta vektor x na način da se na *x*-os nanose indeksi vektora x, a na *y*-osi nalaze se vrijednosti vektora. Zadane točke međusobno se spajaju ravnim linijama:

```
>> x=rand(1,10)            % generirajmo i nacrtajmo vektor s 10 slučajnih brojeva
x =
  Columns 1 through 7
    0.6154    0.7919    0.9218    0.7382    0.1763    0.4057    0.9355
  Columns 8 through 10
    0.9169    0.4103    0.8936

>> figure (2); plot(x)     % aktivirajmo npr. prozor pod rednim brojem 2 i nacrtajmo vektor
```

Grafički prikaz dan je na slici 1. Točke vektora i linije koje ih povezuju mogu biti definirani na više različitih načina (oblik točke, boja, tip i debljina linije). Npr, želimo isti graf nacrtati u crvenoj boji, točke prikazati kružićima, a liniju isprekidanom crtom (slika 2.) :

```
>> plot(x,'o:r')    % znakovi u navodnicima znače: o - točke označene kružićima
                    %                              :  -  isprekidana crta
                    %                              r  -  korištena crvena boja (r- red)
```

ili npr (slika 3):

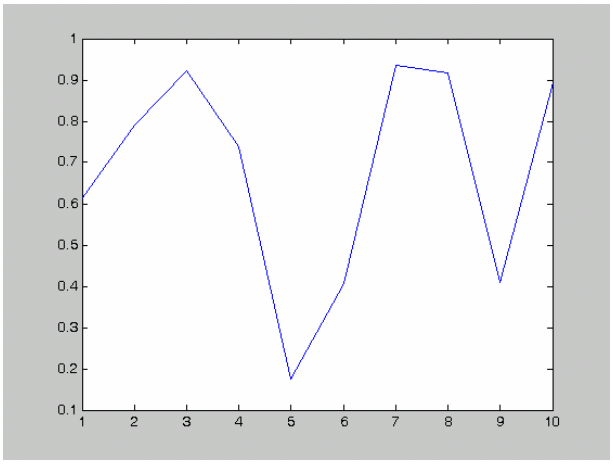
```
>> plot(x, 'g', ‘LineWidth’,3)    %   ‘g’ – korištena boja zelena (g-green)
                                  %   ‘LineWidth’,3 – debljina linije je 3 (inače, po
                                  %    defaultu je 0.5)
```

Za sve mogućnosti naredbe plot proučiti naredbu *help plot*.

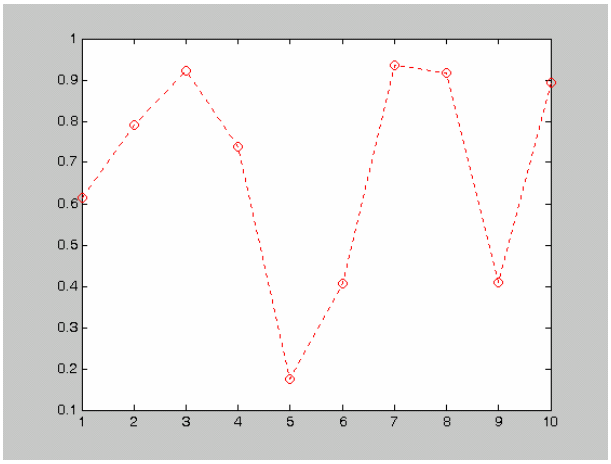
Slici možemo dodati i naslov (naredba *title*), te nazive osiju (naredbe *xlabel* i *ylabel*), npr (slika 4):

```
>> >> plot(x, 'pm', 'LineWidth',2)  % ‘p’ - točke označene pentagramima (zvjezdicama)
                                    %   ‘m’ – korištena boja je magenta
>> title('naslov: zvjezdice')        %   naslov će biti tekst između navodnika
>> xlabel('x-os')
>> ylabel('y-os')
```

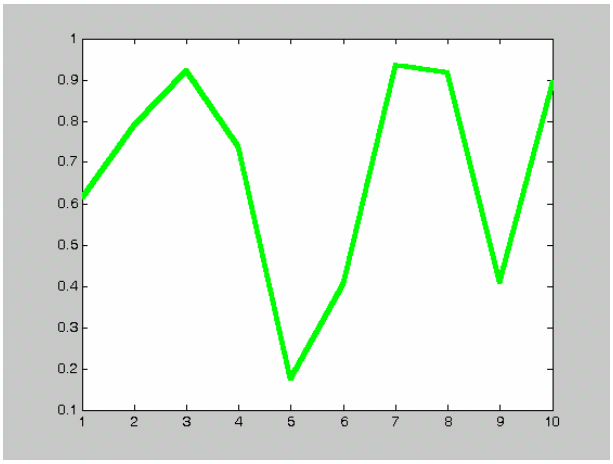
Sve karakteristike slike možemo podešavati i tako da aktiviramo editiranje slike (Tools-Edit Figure) te 2-put kliknemo na sliku nakon čega će se otvoriti prozor za editiranje.



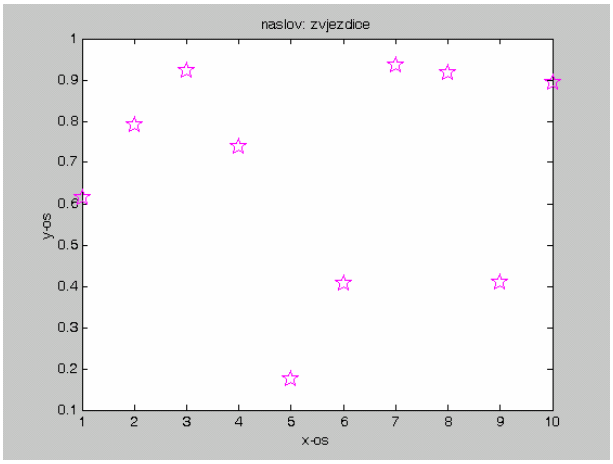
Sl. 1



Sl. 2



Sl. 3



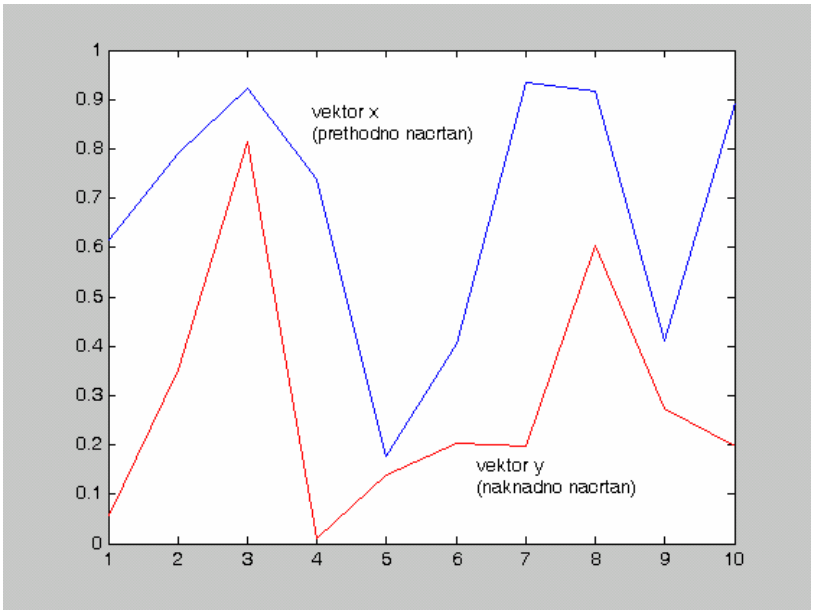
Sl. 4

Naredba plot i općenito svaka naredba za crtanje u neki grafičku prozor briše prethodni sadržaj prozora (prethodni graf). Ukoliko želimo sačuvati postojeće i crtati preko već

nacrtanog, koristit ćemo naredbu *hold on* koja zadržava zadnji aktivni prozor i u njega crta novi graf, npr:

```
>> plot (x)
>> y=rand(1,10)
y =
  Columns 1 through 7
    0.0579    0.3529    0.8132    0.0099    0.1389    0.2028    0.1987
  Columns 8 through 10
    0.6038    0.2722    0.1988

>> hold on                                % zadržava postojeći prozor
>> plot(y,'r')
>> title ('dva grafa na istoj slici')
```

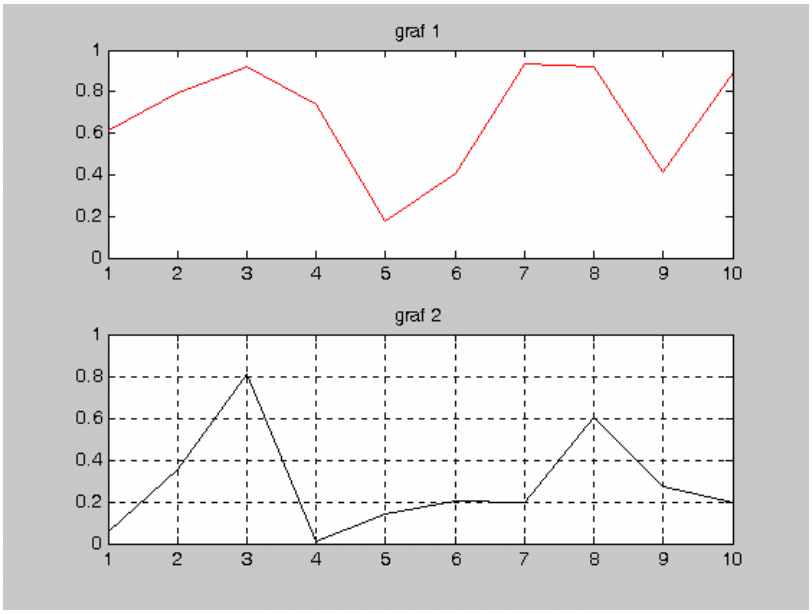
Rezultat je prikazan na slici 5. Zadržavanje postojećeg stanja grafičkog prozora isključuje se naredbom *hold off*.



Sl. 5: Primjena naredbe *hold on*

Zanimljiva je i naredba *subplot()* kojom u okviru istog grafičkog prozora prikazujemo više različitih grafova, i to svaki na zasebnoj slici. Naredba *subplot(m,n,p)* razbija grafički prozor na m*n slika (m redci, n stupci) te aktivira p-tu sliku u koju se crta npr (Sl.6):

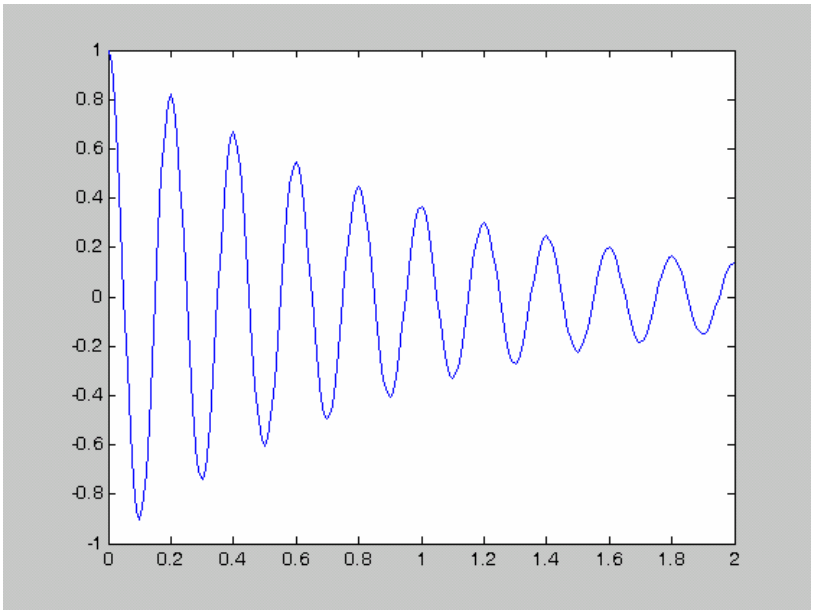
```
>> subplot(2,1,1);        %  grafički prozor dijeli se na 2 retka, 1 stupac i slijedeći graf crta se
                          %  na mjestu p=1
>> plot(x,'r');  title ('graf 1')

>> subplot(2,1,2);        %  slijedeći graf crta se  na mjestu p=2 (brojanje slika ide slijeva
                          %  nadesno i od prvog retka prema dole)
>> plot (y,'k');  title ('graf 2');

>> grid on                %  uočimo i ovu naredbu koja na posljednje nacrtani graf ucrtava
                          %  mrežu (grid)
```



Sl. 6: Primjena naredbe *subplot*

Osim prikaza jednog vektora u zavisnosti o indeksu možemo nacrtati i parove točaka (x, y) koje su prethodno zadane. Pritom naredba plot kao argument prihvaća dva vektora od kojih prvi sadrži x-koordinate dok drugi sadrži y-koordinate (oba vektora moraju imati jednak broj elemenata). Ovakav prikaz najčešće se koristi. Npr. crtaju se vremenski ovisne funkcije, kod kojih vrijeme definiramo kao vektor. Primjer (slika 7):

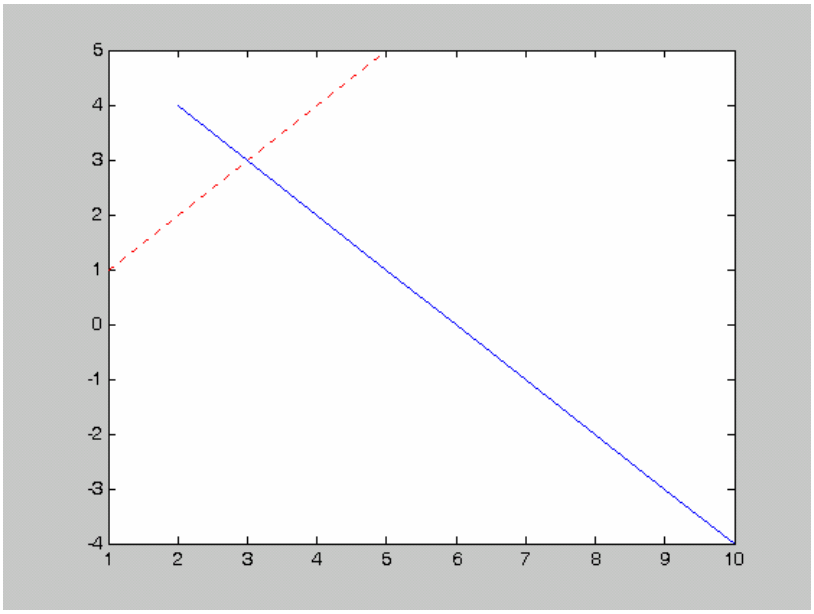
```
>> t=0:0.01:2;                % zadali smo vremenski vektor
>> x=exp(-t).*cos(10*pi*t);   %  obratimo pozornost na '.' ispred operatora množenja
                              %  (jer množimo međusobno odgovarajuće elemente
                              %  obje matrice)
>> plot (t,x)                 % crtamo funkciju x u ovisnosti o vremenu
```



Sl. 7: Funkcija x u ovisnosti o vremenu

Možemo istodobno nacrtati i više različitih parova signala. Općenito naredba plot očekuje ulazne podatke koji su redom *x*-koordinate, *y*-koordinate, način crtanja pa zatim opet *x*-k ordinate, *y*-koordinate, način crtanja itd. Pri tome način crtanja može biti izostavljen. Primjer (slika 8):

```
>> x1=1:5;
>> x2=2:10;
>> y1=x1;
>> y2=6-x2;
>> plot(x1,y1,'r:',x2,y2)
```

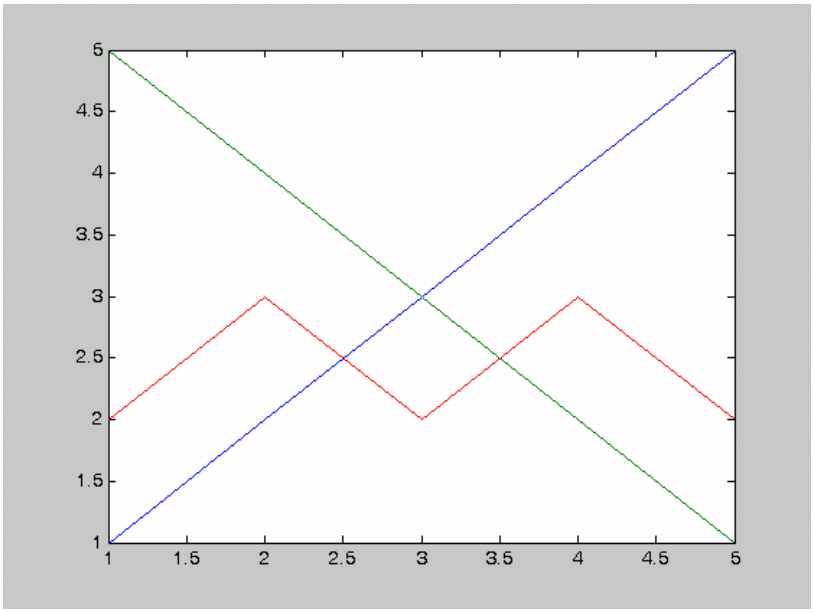


Sl. 8.

Kao što smo vidjeli iz navedenog primjera, pri ovakvom crtanju se različite krivulje mogu razlikovati u broju točaka, tj. krivulja određena parovima (x_1, y_1) ne mora imati jednak broj točaka kao krivulja određena parovima (x_2, y_2).

Naredba *plot()* može se koristiti i za crtanje matrica. Pritom se matrica promatra po stupcima, kao i kod naredbe *max*, tj. pretpostavlja se da svaki stupac predstavlja pojedinačni signal tj. vektor. Primjer (Slika 9):

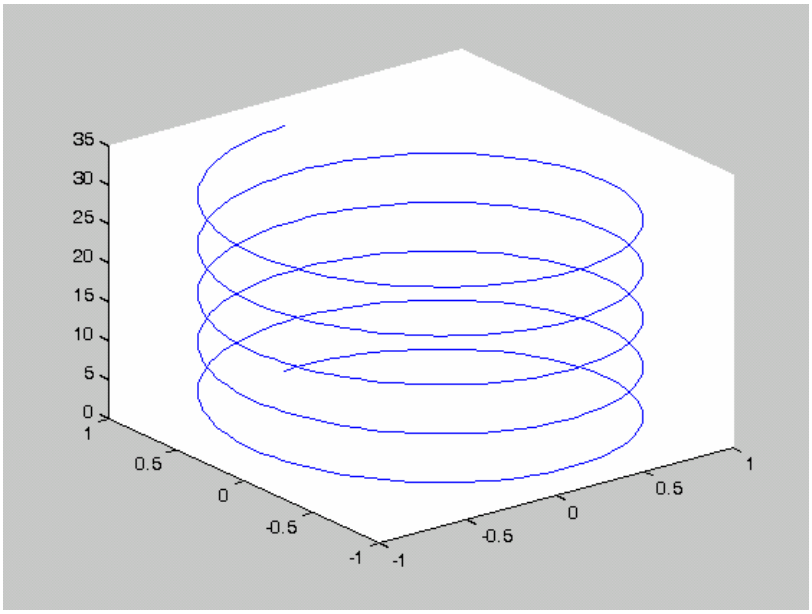
```
>> matrica=[1 5 2; 2 4 3; 3 3 2; 4 2 3; 5 1 2]   %  stupci matrice smatrat
                                                  %  će se kao nezavisni vektori
matrica =
    1    5    2
    2    4    3
    3    3    2
    4    2    3
    5    1    2

>> plot(matrica)          %  naredba plot(matrica) crta vektore [1 2 3 4 5]', [5 4 3 2 1].'
                          %  i [2 3 2 3 2]' u različitim boje, dok se na x-osi
                          %  nalaze indeksi elemenata po recima [1 2 3 4 5]
```



Sl. 9: Crtanje matrica

Matlab podržava i trodimenzionalno crtanje. U tu svrhu koristimo naredbu *plot3(x,y,z)* gdje su x, y i z vektori ili matrice istih dimenzija, npr (Slika 10):

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t);
```



Sl. 10: 3D crtanje

PROGRAMIRANJE U MATLABU

Do sada smo opisali interaktivni rad u Matlabu – upisivanje pojedinačnih naredbi i njihovo neposredno izvršavanje. Sada ćemo opisati kako se u Matlabu pišu programi tj. skripte. Kao što smo već naveli u uvodu, program se sastoji od niza naredbi, piše se u tekstualnom editoru (Matlab editor/debbuger) i pohranjuje kao m-datoteka (datoteka s .m ekstenzijom). Kad u

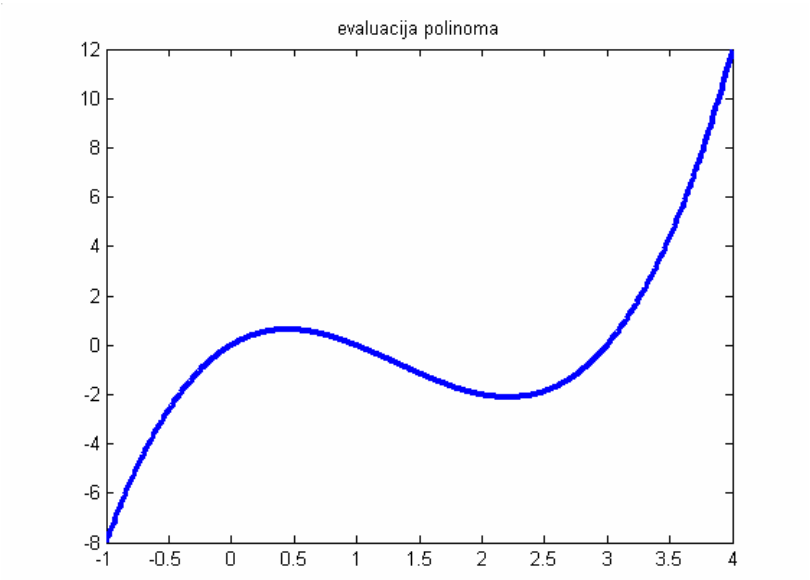
komandnom prozoru Matlaba pozovemo ime te datoteke, i nakon pritiska tipke <ent>, pokrenut će se izvršavanje programa.
Kao primjer napisat ćemo program koji kao ulaznu varijablu prihvaća korijene polinoma, na temelju njih računa koeficijente polinoma te crta polinom u funkciji vremena. Program ćemo nazvati *crtanje_polinoma.m*. Napomena: za imena m-datoteka ne možemo koristiti imena Matlabovih naredbi (npr. *plot.m*, *max.m*, itd.). Tekstualni editor otvaramo iz komandnog prozora na način: File-Open→New→M-File (ili, jednostavnije, u komandnom prozoru izvršimo naredbu *edit*). U editoru pišemo kod programa:

```
% program za crtanje polinoma
clear all
disp('     ')                  %  naredba za ispis poruke na ekran: ispisuje se poruka ispisana u
                               %  navodnicima, u ovom slucaju prazan redak
                               % u slučaju: disp(ime_varijable) naredba disp ispisuje
                               % sadržaj varijable ime_varijable
disp('program za crtanje polinoma na temelju poznatih korijena')
disp('korijene polinoma unijeti u obliku matrice korijeni=[k1 k2...kn]')
disp('za nastavak pritisni bilo koju tipku')
pause                          % Matlab ceka na reakciju korisnika (pritisak na bilo koju
                               % tipku tipkovnice)
disp ('    ')
korijeni=input ('unesi korijene polinoma:  korijeni=')   % naredba za unos podataka pomocu
                                    %  tipkovnice: podatak kojeg unosi korisnik
                                    %  pridruzuje se varijabli korijeni, uz ispis poruke koja
                                    %  se nalazi u navodnicima
disp('koeficijenti polinoma su:')
koef_polinoma=poly(korijeni)  %  racunanje koeficijenata polinoma na temelju korijena

disp('     ')
disp('za nastavak pritisni bilo koju tipku')
pause

disp('unijeti vremenski interval za crtanje polinoma')
t_pocetno=input('unesi pocetno vrijeme, t_pocetno=')
t_zavrsno=input('unesi zavrsno vrijeme, t_zavrsno=')

t=t_pocetno:0.01:t_zavrsno;        % kreiranje vremenskog vektora

x=polyval(koef_polinoma,t);       %  naredba kojom se racunaju vrijednosti polinoma u svim
                                   % tockama vektora t (evaluacija polinoma)
plot(t,x, 'LineWidth',3)
title('evaluacija polinoma')
% kraj m-datoteke
```

Po završetku pisanja koda, program pohranjujemo pod nazivom *crtanje_polinoma.m* i pozivamo ga u glavnom prozoru Matlaba.
Primjerice, želimo nacrtati polinom s korijenima 0,1 i 3 u intervalu od –1 do 4. U glavnom prozoru Matlaba pokrenemo izvršavanje programa:

```
>> crtanje_polinoma                    <ent>

program za crtanje polinoma na temelju poznatih korijena
korijene polinoma unijeti u obliku matrice: korijeni=[k1 k2...kn]
za nastavak pritisni bilo koju tipku

unesi korijene polinoma:  korijeni=[0 1 3]          (korisnik unosi korijene [0 1 3])

kor =
     0     1     3
koeficijenti polinoma su:
koef_polinoma =
     1    -4     3     0


za nastavak pritisni bilo koju tipku
unijeti vremenski interval za crtanje polinoma
unesi pocetno vrijeme, t_pocetno= -1          (korisnik unosi vrijeme –1)

t_pocetno =
    -1

unesi zavrsno vrijeme, t_zavrsno=4            (korisnik unosi vrijeme 4)

t_zavrsno =
     4
```

Kao rezultat, Matlab crta sliku:



Sl. 11

Napomena: Obratiti pozornost na nove naredbe s kojima smo se sreli u ovom programu (*disp, pause, input, polyval*)!

FUNKCIJE U MATLABU

U m-datotekama mogu se definirati i funkcije s jednim ili više ulaznih i izlaznih argumenata. Takve datoteke se nazivaju funkcijske m-datoteke. Kod pisanja funkcije, prvi red mora sadržavati ključnu riječ *function*, ime funkcije te ulazne i izlazne varijable. Funkcija se pohranjuje pod nazivom *ime_funkcije.m*. Funkcija će općenito izgledati ovako:

```
function  [izlaz1, izlaz2,…] = ime_funkcije(ulaz1, ulaz2…)
% komentar
tijelo funkcije
```

Funkciju pozivamo kao i skriptu, jednostavnim navođenjem imena funkcije, s tim da kod funkcije moramo zadati i argumente tj. ulazne varijable.
Kao primjer napišimo funkciju koja će prihvaćati 2 broja te računati i kao rezultat vraćati njihov zbroj i razliku:

```
function [zbr,raz] = zbroj_razlika(x, y)
% zbroj_razlika: zbrajanje i oduzimanje dva broja
% [zbr,raz] = zbroj_razlika(x, y)
% ulazne velicine: x ,y
% izlazne velicine: zbr (x+y), raz (x-y)

%kod funkcije:
zbr=x+y;
raz=x-y;
```

Funkciju pohranjujemo pod nazivom *zbroj_razlika.m* i pozivamo je u komandnom prozoru na način:

```
>> [zbroj, razlika]=zbroj_razlika(23,87)  %   pozivamo funkciju i rezultate spremamo
                                          %   u varijable zbroj i razlika
zbroj =
   110
razlika =
   -64
```

U slučaju da pri pozivu funkcije ne navedemo ulazne argumente, Matlab će javiti pogrešku:

```
>> [zbroj, razlika]=zbroj_razlika
??? Input argument 'x' is undefined.

Error in ==> C:\MATLAB6p1\work\zbroj_razlika.m
On line 9  ==> zbr=x+y;
```

Naredbu *help* možemo koristiti i kod funkcija definiranih od strane korisnika. Na poziv naredbe *help ime_funkcije* ispisat će se komentar ispod ključne riječi *function*, sve do prve prazne linije, npr:

```
>> help zbroj_razlika

  zbroj_razlika: zbrajanje i oduzimanje dva broja
  [zbr,raz] = zbroj_razlika(x, y)
  ulazne velicine: x,y
  izlazne velicine: zbr (x+y), raz (x-y)
```

NAREDBE ZA KONTROLU TOKA PROGRAMA

Matlab može izvršavati osnovne programske petlje (for, while) te uvjetna i bezuvjetna grananja (if, switch, break) kojima se kontrolira tok programa.

FOR PETLJA

For petlja izvršava se na način da brojač unutar petlje poprima sve vrijednosti matrice (po stupcima).

```
>> matrica=[1 2 3;4 5 6];
>> for i=matrica      %  za svaki stupac matrice X
    disp(i)           %  izvrši tijelo petlje tj. ispiši vrijednost brojača i
    end               % svaki for mora završiti end-om

% rezultat izvršenja naredbe:
    1
    4

    2
    5

    3
    6
```

Ako je riječ o rednom vektoru, brojač poprima sve vrijednosti tog retka:

```
>> for i=[1 5 2 8]
     disp(i*10)
     end
  % rezultat izvršenja naredbe:
    10
    50
    20
    80
```

Međutim, najčešće se koristi operator dvotočka (:) koji omogućuje jednostavno postavljanje početne i konačne vrijednosti te koraka:

```
>> X = zeros(1,8);          % inicijalizacija vektora X
>> for i=1:8                %  za svaki i od 1 do 8 …
X(i)=log10(1000*i);         %  …na i-to mjesto vektora X ubaci vrijednost log10(1000*i)
```

```
end
```

Možemo koristiti i ‘for petlju unutar for petlje’:

```
>> X = zeros(3,8);              % inicijalizacija matrice X
>> for i=1:3                    %  za svaki i od 1 do 3 …
       for j=1:8                %   … te za svaki j od 1 do 8 …
         X(i,j)=i+j;            % …na (i,j)-to mjesto matrice X ubaci vrijednost i+j
       end                      %  kraj druge for petlje
   end                          %  kraj prve for petlje
```

WHILE PETLJA

While petlja izvršava se sve dok je ispunjen logički uvjet petlje:

```
>> i=0;                  % inicijalizacija varijable i
>> while i<6             % ponavljaj sve dok je i manji od 6
      disp(i)               % naredbe unutar while petlje
      i=i+1;                % naredbe unutar while petlje
   end                   % kraj while naredbe
% rezultat naredbe:
   0
   1
   2
   3
   4
   5
```

IF NAREDBA

If naredba omogućuje uvjetna izvršavanja koda.

```
A=input(‘A=’)
B=input(‘B=’)
    if  A<B
        disp (‘A je manje od B’)
    elseif A>B
        disp (‘A je vece od B’)
    else
        disp (‘A je jednako B’)
    end
```

NAREDBA SWITCH

Za kompliciranija uvjetna izvršavanja obično koristimo naredbu switch:

```
% razvrstavanje cijene voca po klasama

klasa_voca=input('unesi klasu voca (1,2,3)...')
```

```
switch klasa_voca
  case 1
     disp('cijena prve klase voca iznosi 20kn')
  case 2
     disp('cijena druge klase voca iznosi 15kn')
  case 3
     disp('cijena trece klase voca iznosi 10kn')
  otherwise
     disp('voce nije kategorizirano po klasama')
end                                                    %  kraj switch naredbe
```

ZADACI NA VJEŽBI:

1	Definirati dvije matrice sa po 3 retka i 2 stupca te naći njihov zbroj i razliku.
2	Definirati dvije regularne matrice: $A = \begin{bmatrix} 1 & 4 & 6 \\ 3 & 2 & 5 \\ 9 & 2 & -6 \end{bmatrix}$; $B = \begin{bmatrix} 9 & 23 & 87 \\ -12 & 45 & 4 \\ 2 & 3 & 4 \end{bmatrix}$ Odrediti rezultat matričnih operacija +,-,*.
3	Za matrice iz prethodnog zadatka odrediti također rezultat množenja element-po-element (tj. .*). U čemu je razlika u odnosu na matrično množenje?
4	Odrediti determinante i inverze matrica A i B (naredbe *det()* i *inv()*).
5	Generirati matricu C koja se sastoji od prvog retka matrice A. Pribrojiti matricu C matrici odgovarajućih dimenzija koja sadrži sve jedinice. Koristiti naredbu *ones*().
6	Korištenjem operatora dvotočka definirati vremenski vektor t koji započinje u nuli i završava u trenutku t=15. Neka je korak 0,05. Izračunaj vrijednosti funkcije $y=\cos(2\pi ft)$ za sve elemente vremenskog vektora. f=1.5Hz. Nacrtati dobivenu funkciju (naredba *plot()*).
7	Nacrtati polinom y kojem su zadani korijeni: k1=-30, k2=50, k3=15+3j, k4=15-3j. Polinom nacrtati u intervalu [-30,50] s korakom 0.01. Na istom grafu nacrtati zrcalnu sliku polinoma s obzirom na os apscisa. Polinom y nacrtati žutom, linijom debljine 2, a zrcalnu funkciju isprekidanom crvenom linijom debljine 0.5. Označiti x i y os te napisati naslov. naredbe: *poly(), plot(), hold on, xlabel(), ylabel(), title()*
8	Nacrtati graf funkcije: $Y=200e^{-0.05x}\sin(x)$, za x u intervalu [0,20], korak 0.01. Napomena: kod množenja matrica $e^{-0.05x}$ i sin(x) koristiti množenje element-po-element (.*). naredbe: *exp(), plot().*
9	Napisati program (m-datoteku) koji unosi matricu $A = \begin{bmatrix} 1 & 2 & 8 & 9 \\ 4 & 5 & 1 & 0 \end{bmatrix}$. Na temelju učitane matrice program generira dva polinoma na način da su koeficijenti prvog polinoma sadržani u prvom retku matrice te analogno za drugi polinom. Na isti graf različitim bojama nacrtati 2 polinoma za vremenski period t=[0,20], korak 0.1. Umetnuti naslov te označiti x i y os.
10	Napisati program koji učitava 2. stupčaste matrice (svaka ima po 8 elemenata) i ispisuje sumu veće matrice.
11	Napisati funkciju koja kao ulazni argument prima stupčastu matricu i računa standardnu devijaciju učitane matrice. U glavnom programu učitati matricu i ispisati standardnu

	devijaciju pozivom na funkciju. Koristiti formulu: $st\_dev = \sqrt{\frac{1}{N-1}\sum_i (x_i - \bar{x})^2}$ gdje je N broj elemenata matrice, a $\bar{x}$ srednja vrijednost svih elemenata matrice.
12	Napisati program koji na dva zasebna grafa u okviru iste slike crta crta slijedeće funkcije: $y_1 = x\sin(3x)$ i $y_2 = 5y_1 - x$, pri čemu je x=[0.20], korak 0.1. Označiti x i y osi i napisati naslove. y_1 nacrtati crnom linijom, a y_2 crvenom isprekidanom linijom. Koristiti naredbu *subplot().*
13	Napisati program koji žutom isprekidanom linijom crta graf funkcije $y=5xe^{-3x}$. Na graf napisati naslov te označiti x i y os. x=[-3,10], korak 0.05.
14	Napisati program koji učitava elemente stupčaste matrice sve dok se ne učita element manji od 0. Sortirati učitane elemente matrice po rastućem nizu te ih ispisati. Koristiti naredbe *while* i *sort* (za sortiranje matrice)