Sadržaj

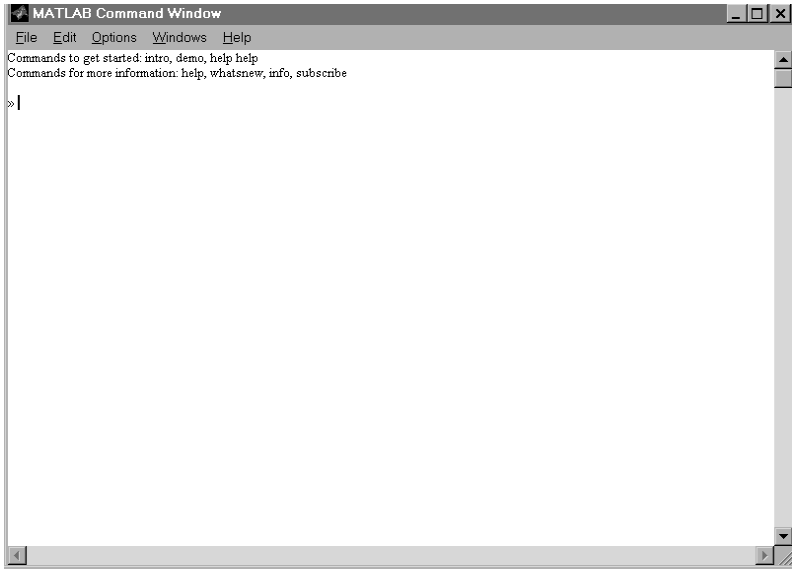
Uvod:

Prva, izvorna verzija MATLAB-a, napisana je kasnih sedamdesetih, na univerzitetu New Mexico i Stanford Univerzitetu, sa osnovnom namjenom da služi kao pomoćno sredstvo na kursevima iz linerarne algebre, i numeričke analize. Zamišljeno je da ovaj paket bude nadgradnja FORTRAN-a koja bi koristila gotove potprograme FORTRAN-a. Današnje mogućnosti MATLAB-a daleko prevazilaze tadašnji originalni "Matrix Laboratory". Ogroman broj naučnih i tehničkih disciplina neizostavno zahtijevaju korištenje MATLAB-a. MATLAB pored svojih prvenstveno razvojnih i programabilnih mogućnosti posjeduje još jednu zaista moćnu alatku koja je jedna od osnovnih odlika ovog paketa. To su *toolboxovi*. Naime, vrlo jednostavno se u MATLAB-u mogu kreirati sopstvene funkcije koje daju rješenja na postavljene zahtjeve. Skup ovako kreiranih funkcija (*m-fajlova*) objedinjenih u jednu cjelinu predstavlja osnovnu strukturu toolboxa. Toolboxovi dakako predstavljaju mnogo više od kolekcije upotrebljivih fajlova, jer je u njima objedinjen trud velikih svjetskih istražitelja u raznim područjima nauke.

OSNOVNI PRINCIPI RADA U MATLAB-u

Uvodne napomene:

Nakon što ste startovali MATLAB na vašem monitoru se pojavljuje okvir prikazan na sljedećoj slici:



Ovaj okvir predstavlja osnovni prostor za rad sa MATLAB-om. Prompt "»" označava da je MATLAB spreman da odgovori na vaše pitanje.

Osnovne matematičke operacije:

OPERACIJA	SIMBOL	PRIMJER
sabiranje	+	5+3
oduzimanje	-	23-11
množenje	*	3.14*0.85
dijeljenje	/ ili \	56/8 = 8\56
stepenovanje	^	5^2

Evo jednog jednostavnog primjera:
Neka je potrebno izračunati: 2+4+6. Da bi dobili rezultat dovoljno je otkucati i pritisnuti Enter :

```
» 2 + 4 + 6
ans =
   12
```

nakon čega MATLAB daje rezultat (ans je skraćenica od engl. riječi answer). Ako je potrebno izračunati sledeći izraz: (4*25)+(6*22)+(2*99) kucamo niz naredbi:

```
» 4*25+6*22+2*99
ans = 430
```

Zgodno je ova dva primjera rješiti uvođenjem pomoćnih varijabli A, B i C. Tada bi se rješenje ova dva primjera dobilo kucanjem sljedećeg niza naredbi u MATLAB-u:

```
» A=2

A =
     2
» B=4;

» C=6

C =
     6
» D=A+B+C

D =
    12
» E=B*25+C*22+A*99

E =
   430
```

Ovdje su kreirane konstante A, B, C sa vrijednostima 2, 4, 6. Treba primjetiti da simbol ";" u liniji prihvata rezultat bez njegovog ispisa na ekranu.

Osvrnimo se sada ne neke osnovne osobine MATLAB-a. Prilikom rada u komandnom prostoru, MATLAB pamti sve naredbe koje su unešene kao i vrijednosti svih varijabli koje se u programu koriste. Ove naredbe ili vrijednosti varijabli mogu se vrlo jednostavno provjeriti. Na primjer, da bi provjerili vrijednost varijable C potrebno je da to zatražite od MATLAB-a kucajući ime varijable nakon prompta.

```
» C
C =
     6
```

Ukoliko postoji potreba da se provjere imena neke od varijabli ili u krajnjem slučaju sve varijable potrebno je od MATLAB-a zatražiti listu varijabli koristeći naredbu *who*.

```
»who
Your variables are:
A        B        C
```

Odavde je vidljivo da programski paket MATLAB ne daje vrijednosti ovih varijabli već samo njihova imena. Da bi dobili vrijednost neke od varijabli potrebno je nakon prompta otkucati njeno ime. Kod nekih verzija MATLAB-a posljednja linija *who* naredbe daje podatak o slobodnom prostoru koji zavisi od raspoložive memorije računara koji koristite.

Formati za prikazivanje numeričkih podataka

Pri radu sa numeričkim podacima u MATLAB-u važe sljedeća pravila. Ako je rezultat integer MATLAB ga prikazuje kao integer. Isto tako, ako je podatak realan broj MATLAB ga prikazuje kao realan broj sa četiri decimalna mjesta. U sljedećoj tabeli su dati formati koji se mogu koristiti u MATLAB-u.

format long	35.833333333333334	16 cifara
format short e	3.5833e+01	5 cifara plus eksponent
format long e	3.583333333333334e+01	16 cifara plus eksponent
format hex	4041eaaaaaaaaaab	hexadecimalni
format bank	35.83	dva decimalna mjesta
format +	+	pozitivan, negativan ili nula
format rat	215/6	rac.aproksimacija
format short	35.8333	uobičajen format

Treba primjetiti da MATLAB ne mjenja unutrašnju konvenciju o zapisu broja kada se koriste različiti formati, već samo prikazuje broj u formatu koji je izabran. Kao i u bilo kojem drugom programskom jeziku i u MATLAB-u postoje izvjesna pravila pri kreiranju imena varijabli. Kao prvo, ime varijable mora biti jedinstvena riječ bez praznih mijesta. U MATLAB-u nije svejedno da li se u imenu varijable pojavljuju mala ili velika slova. Tako na primjer, fruit, Fruit, FrUit i FRUIT su različite varijable. Nadalje, dopuštena dužina varijabli je najviše 19 karaktera. Svi karakteri nakon 19-og biće ignorisani. Obavezno je da varijabla počinje slovom, tj. da prvi karakter bude slovo iza kojeg mogu da slijede slova, brojevi ili simbol "_". U MATLAB-u postoji nekoliko *specijalnih varijabli*:

ans	varijabla u koju se smješta rezultat nakon izvršene operacije
pi	broj pi
eps	najmanji broj koji dodan jedinici daje broj sa pokretnim zarezom koji je veći od jedan
inf	beskonačna vrijednost $(1/0)$
NaN	0/0 nedefinisana vrijednost (Not-a-Number)
i	imaginarna jedinica
realmin	najmanji pozitivan realan broj
realmax	najveći pozitivan realan broj

Nakon definisanja varijable u MATLAB-u može se javiti potreba da se posmatrana varijabla redefiniše, odnosno da joj se dodijeli neka druga vrijednost. Razmotrimo ovdje sljedeći jednostavan primjer:

```
»A=2;
»B=4;
»C=6;
»D=A+B+C
D=
  12
```

Ako sada redefinišemo npr. varijablu A i zadamo joj vrijednost 6:

```
»A=6;
»D
D=
   12
```

vidimo da se rezultat nije promjenio tj. da MATLAB nije prihvatio unešenu promjenu. Da bi unešenu promjenu MATLAB prihvatio potrebno je koristiti neku od odgovarajućih naredbi MATLAB-a i zahtjevati ponovo izračunavanje zadanog algoritma. Kasnije, kada budu objašnjeni m-fajlovi, biće objašnjen jednostavan način da se ponovo izvrši redefinisanje grupe MATLAB-ovih varijabli. Sve ovo odnosi se i na ranije navedene specijalne varijable. Nakon startovanja MATLAB-a ovim varijablama su dodjeljene vrijednosti navedene u prethodnoj tabeli. Ukoliko redefinišete ove varijable, novounešene vrijednosti ostaju sve dok se ove varijable ne izbrišu ili MATLAB ponovo ne startuje. Zbog svega ovoga se ne preporučuje redefinisanje specijalnih varijabli ukoliko to nije neophodno.

Varijable u MATLAB-ovom okruženju se mogu nepovratno izbrisati korištenjem naredbe *clear*. Tako, na primjer naredba: *clear A* briše varijablu A. Naredba: *clear B C* briše u isto vrijeme varijable B i C, dok naredba: *clear* briše sve varijable iz okruženja. Ovdje treba biti posebno pažljiv, jer MATLAB ne zahtjeva da potvrdite ovu naredbu, nego je odmah izvršava, tj. briše sve varijable bez mogućnosti da se obrisane naredbe na bilo koji način ponovo "ožive".

Ako je u nizu MATLAB-ovih naredbi potrebno ubaciti komentar u svrhu lakšeg praćenja toka programa onda se iza naredbe MATLAB-a ili odmah iza prompta “»” navodi simbol postotka (%) iza koga se piše komentar.

```
» n=10          %  n određuje dužinu niza
n =
   10
```

MATLAB dopušta da u jednoj liniji bude napisano više komandi uz uslov da budu odvojene simbolom "," ili ";".

```
» A=2 ,B=4 ;C=6
A =
    2
C =
    6
```

U ovom slučaju su varijable A i C prikazane u obliku rezultata, jer iza njh nije naveden simbol ";".

MATLAB možete u svakom trenutku prekinuti pritiskom *Ctrl+C* na tastaturi. Ako ste završili sa radom otkucate *quit* i izlazite iz MATLAB-a.

Matematičke funkcije MATLAB-a

MATLAB nudi mnoštvo matematičkih funkcija koje se koriste pri rješavanju problema iz raznih oblasti nauke. Lista matematičkih funkcija koje podržava MATLAB data

je u sljedećoj tabeli. Velika olakšica pri radu sa ovim programskim paketom je sličnost sa standardnom matematičkom notacijom.

Lista matematičkih funkcija:

abs (x)	- apsolutna vrijednost	floor (x)	- zaokruživanje prema $-\infty$
acos (x)	- arkus kosinus	imag (x)	- imaginarni dio
acosh (x)	- arkus kosinus hiperbolički	log (x)	- prirodni logaritam
angle (x)	- ugao	log10 (x)	- logaritam sa osnovom 10
asin (x)	- arkus sinus	real (x)	- realni dio
asinh (x)	- arkus sinus hiperbolički	rem (x,y)	- ostatak pri djeljenju
atan (x)	- arkus tangens	round (x)	- najveće cijelo
atan2 (x,y)	- arkus tangens	sign (x)	- signum funkcija
atanh (x)	- arkus tangens hiperbolički	sin (x)	- sinus
ceil (x)	- zaokruživanje prema ∞	sinh (x)	- sinus hiperbolički
conj (x)	- konjugovani broj	sqrt (x)	- kvadratni korjen
cos (x)	- kosinus	tan (x)	- tangens
cosh (x)	- kosinus hiperbolički	tanh (x)	- tangens hiperbolički
exp (x)	- e^x		
fix (x)	- zaokruživanje ka nuli		

Još jedna velika prednost MATLAB-a jeste mogućnost rada sa *kompleksnim brojevima*. Da bismo to ilustrovali razmotrićemo ovdje kvadratnu jednačinu poznatog oblika:

$$ax^2 + bx + c = 0$$

Rješenja ove jednačine su također poznata:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Za a=1, b=5 i c=6, rješenje postavljene jednačine dobija se na sljedeći način:

```
»a=1; b=5; c=6;
»x1=(-b+sqrt(b^2-4*a*c))/(2*a)
x1 =
  -2
»x2=(-b-sqrt(b^2-4*a*c))/(2*a)
x2 =
  -3
»a*x1^2+b*x1+c       %provjera rezultata
ans =
    0
»a*x2^2+b*x2+c       %provjera rezultata
ans =
    0
```

Posljednje dvije linije u ovom algoritmu očigledno služe za provjeru rezultata. Kod kvadratne jednačine sa ovako izabranim koeficijentima diskriminanta je bila veća od nule, te

su rješenja bila realna i različita. Izaberimo sada za koeficijente jednačine sljedeće vrijednosti: a=1, b=4 i c=13. Tada će rješenje jednačine biti:

```
»a=1; b=4; c=13;
»x1=(-b+sqrt(b^2-4*a*c))/(2*a)
x1 =
  -2.0000+3.0000i
»x2=(-b-sqrt(b^2-4*a*c))/(2*a)
x2 =
  -2.0000-3.0000i
```

Ovdje su x1 i x2 dati u kompleksnom obliku, odnosno u obliku z=a+bi, gdje su a i b realni i imaginarni dijelovi kompleksnog broja z, respektivno.
Evo nekoliko jednostavnih primjera kompleksnih brojeva i operacija nad njima:

```
» c1=1-2i
c1 =
  1.0000 - 2.0000i
» c1=1-2j
c1 =
  1.0000 - 2.0000i
» c2= 3*( 2-sqrt ( -1 ) *3 )
c2 =
  6.0000 - 9.0000i
» c3=sqrt ( -2 )
c3 =
      0 + 1.4142i
» c4=6+sin( .5 )*i
c4 =
  6.0000 + 0.4794i
» c5=6+sin( .5 )*j
c5 =
  6.0000 + 0.4794i
```

Iz ovih primjera se vidi da MATLAB ne pravi razliku u označavanju imaginarne jedinice, tj. svejedno mu je da li smo je označili sa "i" ili sa "j". U nekim programskim jezicima operacije nad kompleksnim brojevima zadaju dosta muke programerima i rad sa kompleksnim brojevima je prilično mukotrpan. Sa MATLAB-om to nije slučaj, jer se svaki kompleksan broj unosi na sličan način kao i realan. U sljedećem primjeru je ilustrovan rad sa kompleksnim brojevima pri izvršavanju nekih jednostavnih matematičkih operacija:

```
» check_it_out=i^2              %  varijabla check_it_out
check_it_out =
                -1.0000 + 0.0000i
» check_it_out=real (check_it_out )
check_it_out =
                 -1
```

U ovom je primjeru pokazano korištenje naredbe *real* koja kao rezultat vraća realni dio kompleksnog broja. Kao posljednji primjer, razmotrimo ovdje Ojlerov obrazac.

$$a + ib = Me^{j\theta}$$

gdje su M-moduo, a θ -argument dati sa:

$$M = \sqrt{a^2 + b^2}$$
$$\theta = arctg(\frac{b}{a})$$
$$a = M\ cos\,\theta$$
$$b = M\ sin\,\theta$$

Prelaz sa pravougaonih na polarne koordinate i obrnuto vrši se pomoću naredbi *abs, angle, real* i *imag* :

```
»c1=1-2i;
»mag_c1=abs(c1)       %moduo kompleksnog broja
mag_c1 =
          2.2361
»angle_c1=angle(c1)  %argument kompleksnog broja
angle_c1 =
          -1.1071
»deg_c1=angle_c1*180/pi
deg_c1 =
         -63.4349
»real_c1=real(c1)
real_c1 =
          1
»imag_c1=imag(c1)
imag_c1 =
          -2
```

Korištenje HELP-a

Programiranje u MATLAB-u kao i u drugim programskim jezicima zahtijeva poznavanje čitavog niza naredbi ovog programskog paketa. Da biste došli do određene naredbe, MATLAB vam može pomoći na tri načina:

- koristeći naredbu help,
- koristeći naredbu lookfor,
- koristeći help interaktivno iz bar menija.

Komanda help

Korištenje ove komande je najjednostavniji način da dobijete informaciju o naredbi koja vam je potrebna. Npr.:

```
» help sqrt

 SQRT Square root.
       SQRT(X) is the square root of the elements of X. Complex
```

results are produced if X is not positive.

See also SQRTM.

Ali, na upit

```
» help cows
cows not found.
```

MATLAB jednostavno odgovara da o "cows" ne zna ništa. Naredba help odlično funkcioniše ukoliko ste MATLAB-u eksplicitno zadali temu ili naredbu koja vas interesuje. HELP također nudi i neke smjernice koje mogu korisno da posluže u potrazi za potrebnom naredbom. Za ovo je dovoljno otkucati:

```
» help
HELP topics:
toolbox\local          - Local function library.
matlab\datafun         - Data analysis and Fourier transform functions.
matlab\elfun           - Elementary math functions.
matlab\elmat           - Elementary matrices and matrix manipulation.
matlab\funfun          - Function functions - nonlinear numerical methods.
matlab\general         - General purpose commands.
matlab\color           - Color control and lighting model functions.
matlab\graphics        - General purpose graphics functions.
matlab\iofun           - Low-level file I/O functions.
matlab\lang            - Language constructs and debugging.
matlab\matfun          - Matrix functions - numerical linear algebra.
matlab\ops             - Operators and special characters.
matlab\plotxy          - Two dimensional graphics.
matlab\plotxyz         - Three dimensional graphics.
matlab\polyfun         - Polynomial and interpolation functions.
matlab\sounds          - Sound processing functions.
matlab\sparfun         - Sparse matrix functions.
matlab\specfun         - Specialized math functions.
matlab\specmat         - Specialized matrices.
matlab\strfun          - Character string functions.
matlab\dde             - DDE Toolbox.
matlab\demos           - The MATLAB Expo and other demonstrations.
toolbox\codegen        - Real-Time Workshop
toolbox\control        - Control System Toolbox.
toolbox\dspblks        - DSP Blockset.
toolbox\uitools        - User Interface Utilities.
fuzzy\fuzzy            - Fuzzy Logic Toolbox.
fuzzy\fuzdemos         - Fuzzy Logic Toolbox Demos.
toolbox\images         - Image Processing Toolbox.
nnet\nnet              - Neural Network Toolbox.
nnet\nndemos           - Neural Network Demonstrations and Applications.
toolbox\signal         - Signal Processing Toolbox.
simulink\simulink      - SIMULINK model analysis and construction functions.
simulink\simdemos      - SIMULINK demonstrations and samples.
simulink\blocks        - SIMULINK block library.
simulink:2sl           - SystemBuild 3.0 model import into SIMULINK.
```

wavelet\wavelet - Wavelet Toolbox.
wavelet\wavedemo - Wavelet Toolbox Demos.

For more help on directory/topic, type "help topic".

Lista na vašem računaru može se unekoliko razlikovati od ove. U svakom slučaju, gornja lista vam daje mogućnost da vrlo brzo dođete do komande koja vas interesuje. Korištenjem direktno help komande postoji mogućnost dobijanja povratne informacije samo ako ste potpuno sigurni koja naredba vam je potrebna. Ukoliko to niste preostala dva načina mogu biti od velike pomoći.

Lookfor komanda

Ova komanda pretražuje sve linije help-a i vraća samo one koje sadrže ključnu riječ koju navedete. Npr.

```
» lookfor complex
CPLXPAIR Sort numbers into complex conjugate pairs.
CONJ  Complex conjugate.
IMAG Complex imaginary part.
REAL  Complex real part.
CDF2RDF Complex diagonal form to real block diagonal form.
RSF2CSF Real block diagonal form to complex diagonal form.
CPLXDEMO Maps of functions of a complex variable.
CPLXGRID Polar coordinate complex grid.
CPLXMAP Plot a function of a complex variable.
GRAFCPLX Demonstrates complex function plots in MATLAB.
DSORT        Sort complex discrete eigenvalues in descending order.
ESORT        Sort complex continuous eigenvalues in descending order
LOGM2        LOGM2(X)  is the matrix natural logarithm of  X .    Complex
 CPLXICON  Complex Icon function for SIMULINK Complex blocks.
SCPLXWKS  DSP Blockset S-Function to store complex SIMULINK data in
workspace.
CCEPS        Complex cepstrum.
```

Naredba *complex* nije MATLAB-ova naredba, ali je nađena u deskripciji nekoliko MATLAB-ovih naredbi. Ili na primjer:

```
» help conj

 CONJ Complex conjugate.
       CONJ(X) is the complex conjugate of X.
```

Na PC-u, help je dostupan u liniji menija sa opcijama *Table of Contents* ili *Index*. Na ovaj način otvarate *MATLAB Help* gdje dvostrukim klikom miša birate opciju ili naredbu koja vas interesuje.

Operacije sa poljima

Sva dosadašnja razmatranja uključivala su rad sa brojevima, odnosno skalarima i operacije nad njima. Operacije nad skalarima su osnovne matematičke operacije. Sa druge strane, ako je potrebno jednu te istu operaciju izvršiti nad više od jednog skalara ponovljene operacije bile bi povezane sa izvjesnim teškoćama. Upravo iz tih razloga se došlo do pojma

polja (matrica). MATLAB sve varijable posmatra kao matrice i sve operacije izvršava kao matrične operacije. Skalari su samo specijalan slučaj matrica dimenzija 1x1.

Prosta polja

Razmatrajmo ovdje primjer sračunavanja vrijednosti funkcije sinusa na polovini perioda. Jasno je da je nemoguće izračunati vrijednosti ove funkcije u svim tačkama datog intervala (kojih ima beskonačno mnogo), već se ograničavamo na konačan broj tačaka iz datog intervala. Uzmimo na primjer da je korak diskretizacije 0.1π. Mi bismo do rezultata mogli doći kucanjem posebno svake vrijednosti argumenta i tražiti rezultat. Ovo se međutim mnogo brže rješava korištenjem polja. Definisanje polja je jednostavno:

```
» x= [ 0  .1*pi  .2*pi  .3*pi  .4*pi  .5*pi  .6*pi  .7*pi  .8*pi  .9*pi  pi ]
x =
  Columns 1 through 7
        0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850
  Columns 8 through 11
    2.1991    2.5133    2.8274    3.1416
» y=sin (x)
y =
  Columns 1 through 7
        0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
  Columns 8 through 11
    0.8090    0.5878    0.3090    0.0000
```

Dovoljno je otvoriti srednju zagradu, unijeti vrijednosti željenih tačaka, odvojenih međusobno razmacima ili zarezima, te zatvoriti srednju zagradu. MATLAB sada "razumije" da želite da vam izračuna sinus svakog elementa iz x i da rezultat treba da bude smješten u polje y. Ovo je fundamentalna razlika između MATLAB-a i ostalih programskih jezika. Ako se kao element polja unosi kompleksan broj onda se on mora unijeti kao jedinstven, bez razmaka ili objedinjen u maloj zagradi. Npr. polje [1 -2i 3 4 5+6i] sadrži pet elemenata dok polja [(1 -2i) 3 4 5+6i] ili [1-2i 3 4 5+6i] sadrže po četiri elementa.

Pristup elementima polja

U gornjem primjeru je polje x sadržavalo ukupno 11 elemenata. To je u stvari matrica vrsta, tj. matrica sa jednom vrstom i 11 kolona ili jednostavno, prosto polje dužine 11. Pristup svakom elementu polja je jednostavan. Dovoljno je otkucati ime polja i indeks elementa.

```
» x ( 3 )              %  treci element polja x
ans =
   0.6283
» y ( 5 )              %  peti element polja y
ans =
   0.9511
```

Pristup nizu elemenata polja može se uraditi na sledeći način:

```
» x ( 1:5 )
ans =
        0    0.3142    0.6283    0.9425    1.2566
» y ( 3: -1 : 1)
```

```
ans =
   0.5878    0.3090         0
```

U prethodna dva primjera, drugi broj u maloj zagradi očigledno označava korak, s tim što negativan predznak znači da se elementi uzimaju unazad.

```
» x ( 2  :  2  :  7 )
ans =
   0.3142    0.9425    1.5708
```

Na ovaj način smo dobili drugi, četvrti i šesti element polja.

```
» y ( [ 8  2  9  1] )
ans =
   0.8090    0.3090    0.5878         0
```

U ovom primjeru smo koristili polje [8 2 9 1] da bismo dobili elemente polja y u redosljedu koji mi želimo. Preporuka je da ovdje uradite sami nekoliko primjera, te da pokušate pozvati petnaesti element (koji ne postoji), pa da vidite šta će MATLAB uraditi.

Definisanje polja

U jednom od ranijih primjera polje smo definisali prosto unošenjem pojedinih vrijednosti elemenata polja. Taj primjer je sadržavao svega 11 vrijednosti. Međutim, šta da je bilo potrebno unijeti 111 vrijednosti? Evo dva načina kako ovo jednostavno uraditi;

```
» x = ( 0 : 0.1 : 1 )*pi
x =
  Columns 1 through 7
         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850
  Columns 8 through 11
    2.1991    2.5133    2.8274    3.1416
```

ili

```
» x = linspace ( 0, pi, 11 )
x =
  Columns 1 through 7
         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850
  Columns 8 through 11
    2.1991    2.5133    2.8274    3.1416
```

U prvom primjeru definisano je polje koje počinje od nule uvećava se za 0.1 i zaustavlja se kada poprimi vrijednost 1. Svaki element polja je pomnožen sa *pi* da bi se dobola željena vrijednost argumenta. U drugom slučaju, korištena je MATLAB-ova naredba *linspace* sa sljedećom sintaksom:

linspace (početna_vrijednost, korak, krajnja_vrijednost)

U oba prethodna primjera, definisana su polja kod kojih su elementi linearno uvećavani za *pi*. Ako je potrebno da skala elemenata bude logaritamska onda se koristi MATLAB-ova naredba *logspace*:

```
» logspace ( 0, 2, 11 )
ans =
  Columns 1 through 7
```

```
  1.0000   1.5849   2.5119   3.9811   6.3096  10.0000  15.8489
 Columns 8 through 11
  25.1189  39.8107  63.0957 100.0000
```

U ovom primjeru je definisano polje čiji je početni element 10^0, a krajnji 10^2. Sintaksa ove naredbe je:

logspace (početni_eksponent, posljednji_eksponent, broj_elemenata)

U slučaju da je redosljed elemenata polja proizvoljan, definišu se dva ili koliko je potrebno polja koja se potom spajaju u jedinstveno polje. Ovo se može uraditi na sljedeći način:

```
» a=1 : 5, b=1 : 2 : 9
a =
   1   2   3   4   5
b =
   1   3   5   7   9
```

Na ovaj način su definisana dva polja a i b. Treba primjetiti da su dvije linije sažete u jednu tako što su naredbe međusobno odvojene zarezom. Spajanje ova dva polja u jedno vrši se na sljedeći način:

```
» c = [ b a ]
c =
   1   3   5   7   9   1   2   3   4   5
```

Naredbom:

```
» d=[ a( 1:2:5 ) 1 0 1]
d =
   1   3   5   1   0   1
```

definisano je polje d koje sadrži prvi, treći i peti element polja a i elemente 1,0 i 1. Matematičke operacije sabiranja, oduzimanja, množenja i dijeljenja polja sa skalarom primjenjuju se posebno na svaki element polja. Tako na primjer:

```
» a-2
ans =
  -1   0   1   2   3
```

Ovom naredbom je svaki član polja umanjen za dva.

```
» 2*a-1
ans =
   1   3   5   7   9
```

ovom naredbom je svaki element pomnožen sa dva i od svakog elementa oduzeta jedinica.

Operacije sa poljima

Matematičke operacije sa poljima su nešto komplikovanije od prethodno objašnjenih operacija između skalara i polja, pogotovo ako su polja različite dužine. Ako su polja iste

dužine, operacije sabiranja, oduzimanja, množenja i dijeljenja obavljaju se između elemenata na istim pozicijama u polju. Tako, na primjer:

```
»  a, b
a =
    1    2    3    4    5
b =
    1    3    5    7    9
```

daje pregled polja a i b, a potom

```
» a+b
ans =
    2    5    8   11   14
```

sabira odgovarajuće elemente i smješta u *ans*.

```
» ans-b
ans =
    1    2    3    4    5
```

u ovom primjeru je očigledno *ans=a*.

```
» 2*a-b
ans =
    1    1    1    1    1
```

U ovom slučaju je svaki element pomnožen sa dva i od svakog elementa je oduzeta vrijednost odgovarajućeg elementa polja b. Množenje odgovarajućih elemenata ova dva polja vrši se korištenjem sljedeće naredbe:

```
» a .* b
ans =
    1    6   15   28   45
```

Elementi polja a su pomnoženi sa elementima polja b koji imaju iste indekse. Za ovo množenje je korišten operator ".*" Množenje bez tačke odnosi se na matrično množenje. U ovom konkretnim primjeru matrično množenje (bez tačke uz operator "*") nema smisla.

```
» a*b
??? Error using ==> *
Inner matrix dimensions must agree.
```

Oprator dijeljenja elemenata jednog polja sa odgovarajućim elementima drugog polja također zahtjeva korištenje tačke.

```
» a ./b
ans =
   1.0000    0.6667    0.6000    0.5714    0.5556
» b .\a
ans =
   1.0000    0.6667    0.6000    0.5714    0.5556
```

Često je pri radu sa poljima ovakvog tipa potrebno elemente polja kvadrirati ili ih stepenovati proizvoljnim eksponentom. I u ovom slučaju se pored operatora stepenovanja

mora koristiti i simbol "." da bi zadani operator stepenovanja bio primjenjen na svaki element polja posebno. Primjer:

```
» a .^ 2
ans =
    1    4    9   16   25
```

Ovdje je očigledno svaki član polja a kvadriran.

```
» 2 .^ a
ans =
    2    4    8   16   32
```

Ovom naredbom je broj dva stepenovan redom sa svakim elementom polja a. Evo još nekih primjera:

```
» b .^ a
ans =
         1         9       125      2401     59049
» b .^ ( a - 3 )
ans =
   1.0000    0.3333    1.0000    7.0000   81.0000
```

U prethodnim primjerima, polja koja su definisana sadržavala su jednu vrstu i nekoliko kolona, tj. u vektor-vrstu. Rezultat zadane operacije također je bio smješten u vektor-vrstu. MATLAB dopušta da se polje definiše i kao vektor kolona tj. polje koje bi sadržavalo jednu kolonu i nekoliko vrsta. Kod definisanja i zadavanja polja na ovaj način sva pravila ostaju na snazi, samo što su polje i rezultat prikazani kao vektor-kolone. Evo nekih primjera u kojima su definisani vektor-kolone.

```
» c = [ 1 ; 2 ; 3 ; 4 ; 5 ]
c =
    1
    2
    3
    4
    5
```

Ovdje treba primjetiti da elementi razdvojeni razmacima ili zarezima definišu elemente jedne vrste, a više kolona, dok elementi razdvojeni simbolom ";" definišu polje koje čini jedna kolona i nekoliko vrsta.

```
» a = 1 : 5
a =
    1    2    3    4    5
» b =a'
b =
    1
    2
    3
    4
```

```
     5
```

U posljednja dva primjera je korišten simbol " ' " za transponovanje vektor-kolone u vektor-vrstu i obratno.

Rad sa matricama

Pošto je matrica osnovni element MATLAB-a postoji mnogo načina za manipulisanje i rad sa matricama. Kada se matrica definiše, odnosno unese u program, MATLAB omogućuje čitav niz postupaka kojima se unesena matrica po volji može mijenjati. Ovo je u stvari ključ za efikasno korištenje MATLAB-a. Ove teme smo se malo dotakli u primjerima gdje je pokazan rad sa jednostavnim poljima. Da bi ovo ilustrovali prelazimo odmah na primjere:

```
» A=[1 2 3;4 5 6; 7 8 9]

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

Definisana je matrica A. U slučaju da je potrebno promjeniti neki od elemenata definisane matrice postupamo na sljedeći način:

```
» A(3,3)=0

A =
   1   2   3
   4   5   6
   7   8   0
```

gdje je očigledno element treće vrste i treće kolone pretvoren u nulu.

```
» A(2,6)=1

A =
   1   2   3   0   0   0
   4   5   6   0   0   1
   7   8   0   0   0   0
```

U ovom je primjeru je na presjeku druge vrste i šeste kolone generisana jedinica. Pošto zadana matrica nema ovu dimenziju preostala mjesta su popunjena nulama, tako da je matrica i dalje ostala pravougaona.

```
» A=[1 2 3;4 5 6; 7 8 9] ;

» B=A(3: -1: 1 , 1 : 3)

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

Na ovaj način je kreirana matrica B tako što su vrste matrice A uzete u obrnutom redoslijedu.

```
» B=A(3: -1: 1 ,  : )

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

Efekt ove naredbe je isti kao i prethodne. Simbol ":" označava da se naredba odnosi na sve kolone. To je, u stvari skraćena notacija za 1:3.

```
» C=[A B(: , [1 3] ) ]

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

Ovom naredbom je zadana matrica C tako što je desna strana matrice A dopunjena prvom i trećom kolonom matrice B.

```
» B=A( 1 : 2 , 2 : 3 )

B =
    2    3
    5    6
```

Ovdje je definisana matrica B, ali tako što je iz matrice A izbačena treća vrsta i prva kolona.

```
» C=[1 3]

C =
    1    3

» B=A(C , C)

B =
    1    3
    7    9
```

U ovom primjeru je iskorišteno polje C za adresiranje matrice A. Ovdje je formirana matrica B tako što je iz matrica A izbačena druga vrsta i druga kolona, odnosno matrica formirana od prve i treće vrsta te prve i treće kolone matrice A.

```
» B=A( : )

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

Ovom naredbom matrica A pretvorena u vektor kolonu.

```
» B=B'

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

Ovdje je izvršeno transponovanje matrice B.

```
» B=A

B =
    1    2    3
    4    5    6
    7    8    9

» B(  :  , 2  )=[ ]

B =
    1    3
    4    6
    7    9
```

U ovom primjeru je redefinisana matrica B tako što je iz zadane matrice B izbačena druga kolona. Kada se označi da je nešto jednako praznoj matrici ono biva obrisano, narušavajući ujedno i dimenzije matrice.

```
» B=B'
B =
    1    4    7
    3    6    9

» B( 2 , : ) = [ ]
B =
    1    4    7
```

Ovdje je očigledno izbačena druga vrsta matrice B.

```
» A( 2 , : ) = B

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

Ovdje je u drugu vrstu matrice A ubačena matrica B.

```
» B=A(  :  , [2 2 2 2] )

B =
    2    2    2    2
    4    4    4    4
    8    8    8    8
```

Sada je matrica B kreirana četverostrukim dupliranjem svih vrsta drugom kolonom matrice A.

```
» A( 2 , 2 )=[ ]
???  In an assignment  A(matrix,matrix) = B, the number of rows in B
and the number of elements in the A row index matrix must be the same.
```

Ova naredba nije dozvoljena. Dimenzije matrica sa obe strane jednakosti se moraju podudarati. U našem slučaju sa lijeve strane jednakosti je skalar (matrica dimenzija 1x1), a sa desne strane prazan niz (matrica dimenzija 0x0) te MATLAB javlja grešku.

```
» B=A( 4 , : )
???  Index exceeds matrix dimensions.
```

Ni ova naredba nije dozvoljena, jer matrica A nema četiri vrste.

```
» C( 1: 6 )=A( : , 2 : 3 )

C =
    2    4    8    3    7    9
```

Ovdje je kreiran vektor C ekstrakcijom i transponovanjem druge i treće kolone matrice A.

U sledećih nekoliko primjera biće pokazano korištenje relacionih operatora pri radu sa poljima i matricama.

```
» x=-3:3
x =
   -3   -2   -1    0    1    2    3

» abs(x)>1
ans =
    1    1    0    0    0    1    1
```

U ovom primjeru je definisan vektor x a kao rezultat su vraćene jedinice na svim mjestima gdje je apsolutna vrijednost pojedinog člana veća od jedinice.

```
» y=x(abs(x)>1)
y =
   -3    -2     2     3
```

U prethodnom primjeru je definisano polje y u koje su smješteni svi elementi iz x čija je apsolutna vrijednost veća od jedinice.

```
» y=x( [ 1 1 1 1 0 0 0 ] )
y =
   -3    -2    -1     0
```

Ovom naredbom su selektovana prva četiri elementa, dok su ostali zanemareni. Ako ovu naredbu usporedimo sa sledećom:

```
» y=x( [ 1 1 1 1 ] )
y =
   -3    -3    -3    -3
```

onda se vidi da je ovdje kreiran y tako što je četiri puta ponovljen prvi član iz x.

```
» y=x( [ 1 0 1 0 ] )
???  Index into matrix is negative or zero.
```

Ova naredba nije dopuštena jer elementi polja u uglatoj zagradi predstavljaju indekse za vektor x, a indeksi ne mogu biti negativni ili nule.

```
» x(abs(x)>1)=[ ]
x =
   -1     0     1
```

Rezultat ove naredbe je da su iz polja x izbačeni svi elementi čija je apsolutna vrijednost veća od jedinice. Ovaj primjer pokazuje da se relacioni operator može koristiti i na lijevoj strani gornje jednakosti. Sve što je dosad navedeno važi i pri radu sa matricama.

```
» b=[5 -3 ; 2 -4 ]
b =
    5    -3
    2    -4
» x=abs(b)>2
x =
    1     1
    0     1
» y=b(abs(b)>2)
y =
    5
   -3
   -4
```

Rezultat posljednje naredbe je smješten u vektor kolonu, jer ne postoji način da se formira matrica sa tri elementa. Ovdje ima još jedan pristup a to je korištenje naredbe *find*.

```
» x=-3:3
x =
   -3    -2    -1     0     1     2     3

» k=find(abs(x)>1)
k =
    1     2     6     7
```

U ovom slučaju je od naredbe *find* zatraženo da pronađe indekse elemenata čija je apsolutna vrijednost veća od jedan.

Naredba *find* također funkcioniše i kod matrica.

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

» [i , j]=find(A>5)
i =
    3
    3
    2
    3
j =
    1
    2
    3
    3
```

Indeksi *i* i *j* su pomoćne varijable čijim se kombinovanjem dolazi do pozicije u matrici tj. indeksa elementa koji je zadovoljio nejednakost postavljenu u gornjoj naredbi. Tako npr. prve vrijednosti indeksa *i* i *j* daju poziciju prvog elementa koji je veći od pet, a to je element ne presjeku treće vrste i prve kolone.

Ako nam dimenzije matrice, odnosno vektora nisu poznate, MATLAB nudi naredbe *size* i *length* koje nam daju informaciju o dimenzijama matrice.

```
» A=[1 2 3 4;5 6 7 8]
A =
    1     2     3     4
    5     6     7     8

» B=pi:0.01:2*pi;
» s=size(A)

s =
   2     4
```

Ovdje je vrijednost fukcije *size* definisana jednom varijablom, pa je kao rješenje vraćen vektor vrsta, gdje prvi član daje podatak o broju vrsta a drugi o broju kolona. Osim ovog načina moguće je vrijednost funkcije *size* definisati kao u sljedećem primjeru.

```
» [r,c]=size(A)
r =
    2
c =
    4
```

Dvije izlazne varijable su smještene u vektor kolonu. Elementi imaju isto značenje kao i maloprije.

Naredba *length* kao rezultat vraća broj kolona ili broj vrsta u zavisnosti od toga da li matrica ima veći broj vrsta ili kolona.

```
» length(A)
ans =
    4

» size(B)
ans =
      1   315

» length(B)
ans =
  315
```

Pošto vektor ima samo jednu dimenziju u ovom slučaju funkcija *length* kao rezultat vraća broj elemenata polja B.

Specijalne matrice

U MATLAB-u je moguće definisati i neke specijalne matrice, koje su od interesa u raznim oblastima nauke i tehnike.

```
» zeros(3)
ans =
    0    0    0
    0    0    0
    0    0    0
```

Ova naredba definiše nula matricu.

```
» ones(2,4)
ans =
    1    1    1    1
    1    1    1    1
```

Ovom naredbom je definisana matrica jedinica.

```
» ones(3)*pi
ans =
   3.1416   3.1416   3.1416
   3.1416   3.1416   3.1416
   3.1416   3.1416   3.1416
» rand(3,1)
ans =
   0.2190
   0.0470
   0.6789
```

Ovim primjerom je definisana matrica čiji su elementi slučajni brojevi sa uniformnom raspodjelom.

```
» randn(2)
ans =
   1.1650   0.0751
   0.6268   0.3516
```

Elementi ove matrice su slučajni brojevi sa normalnom raspodjelom.

```
» eye(3)
ans =
    1    0    0
    0    1    0
    0    0    1
» eye(3,2)
ans =
    1    0
    0    1
    0    0
```

Ovom naredbom je definisana jedinična matrica.
Djeljenje i množenje matrica je pokazano u sledećim primjerima

```
» a=[1 3 -3;2 5 -1];
» b=[2 1;-6 2;1 4];
» a*b
ans =
  -19   -5
  -27    8
» b*a
ans =
    4   11   -7
   -2   -8   16
    9   23   -7
» a/b
??? Error using ==> /
Matrix dimensions must agree.
```

Djeljenje matrica a i b nije bilo moguće izvršiti jer se dimenzije matrica ne slažu.

```
» c=[2 1;3 2];
» d=[3 2;4 3];
» c/d
ans =
   2.0000   -1.0000
   1.0000    0.0000
» e=[3 5 2;1 3 2];
» f=[2 1 4;-4 -3 1];
» e*f
??? Error using ==> *
Inner matrix dimensions must agree.
```

Množenje matrica nije bilo moguće izvršiti jer se dimenzije matrica ne slažu.

Prvobitna svrha MATLAB-a je bila da pojednostavi rad u mnogim oblastima u kojima se primjenjuju osnove linearne algebre. Ovo se posebno odnosi na sisteme linearnih jednačina. Da bi ilustrovali ovaj poseban problem posmatrajmo primjer:

$$1x_1+2x_2+3x_3=366$$
$$4x_1+5x_2+6x_3=804$$
$$7x_1+8x_2+0x_3=351$$

Ovaj sistem se može predstaviti i u matričnom obliku (matričnoj jednačini) na sledeći način:

$$\mathbf{A}x=b$$

Ova matrična jednačina definiše proizvod matrice A i vektora x koji je jednak vektoru b. Egzistencija rješenja ovog sistema i uopšte sistema jednačina je fundamentalno pitanje linearne algebre. Ukoliko je sistem rješiv postoji mnoštvo pristupa kojima se to rješenje nalazi. Analitički, rješenje ovog sistema bi bilo:

$$x=\mathbf{A}^{-1}b$$

Rješenje ovog sistema dobija se kucanjem sljedećeg niza naredbi:

```
» A=[1 2 3;4 5 6;7 8 0]
A =
   1   2   3
   4   5   6
   7   8   0
» b=[366 ; 804 ; 351]
b =
  366
  804
  351
```

Kao što je ranije objašnjeno, matrica A se može unijeti na nekoliko načina. Za prelazak u novu vrstu može se koristiti simbol ";" ili prelaz u novi red. Vektor b je vektor kolona jer svaki simbol ";" označava početak nove vrste. Može se pokazati da ovaj problem ima jedinstveno rješenje ako je determinanta matrice A različita od nule. U ovom je primjeru det (A)=27 te će rješenje sistema Ax=b biti moguće naći. Jedan od načina jeste da se izračuna inverzna matrica A^{-1} :

```
» x=inv (A)*b
x =
  25.0000
  22.0000
```

```
   99.0000
```

Ovo rješenje se moglo dobiti i na sljedeći način:

```
» x=A\b
x =
   25.0000
   22.0000
   99.0000
```

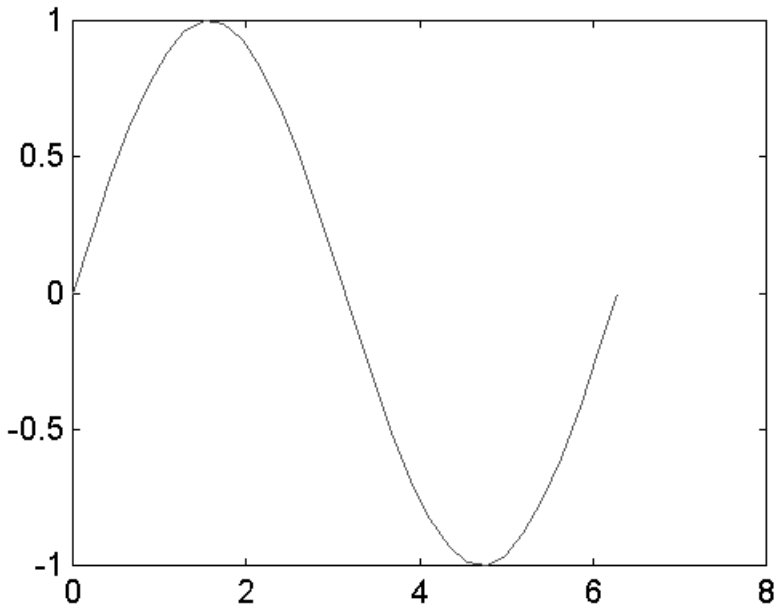
Drugo rješenje tj. korištenje operatora "\" preporučljivije je od prethodnog. Ovaj način rješavanja zahtjeva manji broj unutrašnjih operacija množenja i dijeljenja, te je samim tim ovaj postupak brži. Pogotovo se preporučuje za sisteme sa većim brojem nepoznatih. U slučaju da MATLAB ne može naći rješenje korektno, on javlja grešku.

Grafički prikaz podataka

Jedna od možda najboljih osobina MATLAB-a je ta što ima veoma velike grafičke mogućnosti. U MATLAB-u postoji velik broj naredbi pomoću kojih se grafičko prikazivanje podataka može prikazati u dvije ili tri dimenzije. Urađeni grafici mogu biti sačuvani u nekom od standardnih formata i kao takvi biti korišteni u nekim drugim programskim paketima. Najjednostavnija naredba za prikazivanje podataka je naredba plot. Prikažimo ovdje funkciju: y=sin(x) za x od 0 do 2π. Prvo je potrebno definisati interval sa konačnim brojem tačaka između 0 i 2π.

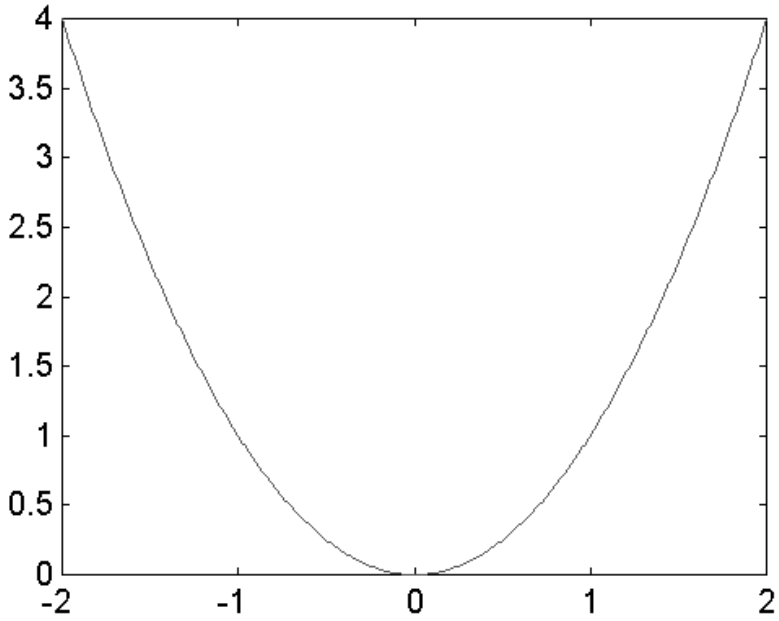
```
» x = linspace ( 0, 2*pi, 30 );
» y=sin ( x );
» plot ( x, y )
```

nakon čega slijedi grafik:



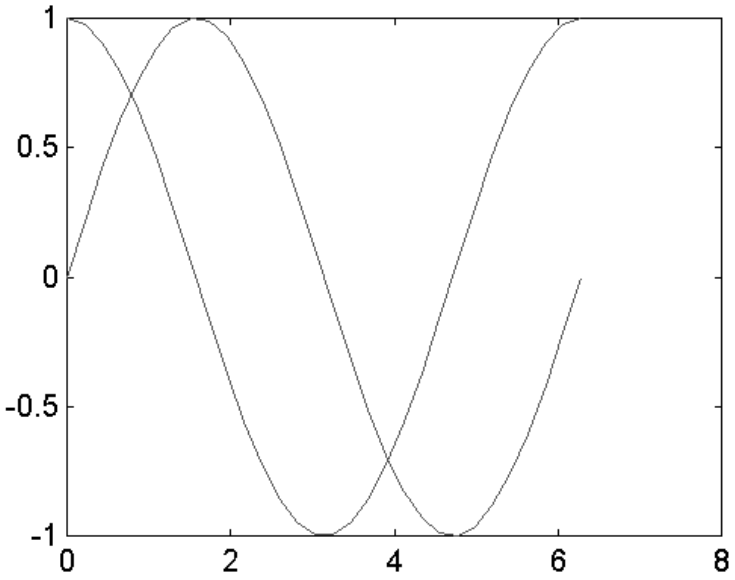
Ili na primjer funkcija: $y=x^2$ u intervalu od -2 do 2.

```
» x =-2 : 0.01 : 2 ;
» y=x.^2 ;
» plot ( x , y )
```



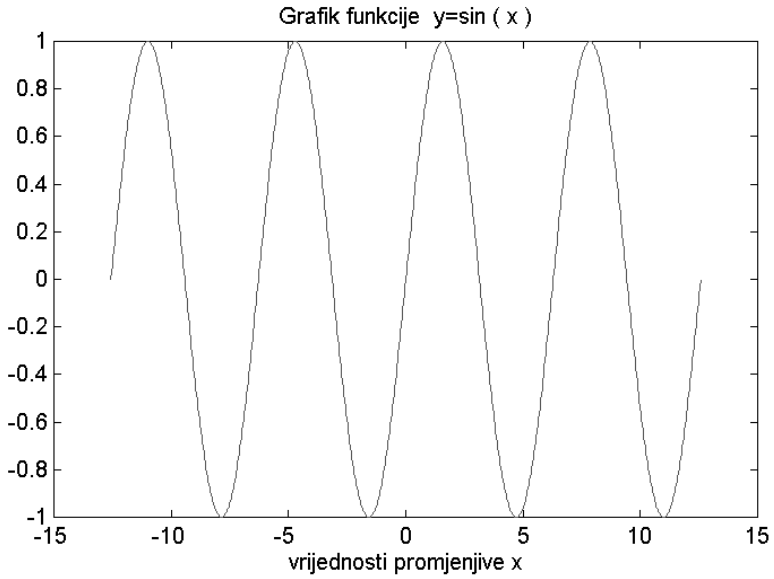
Koristeći naredbu plot moguće je i više grafika prikazati odjednom.

```
» x = linspace ( 0, 2*pi, 30 );
» y=sin ( x );
» z=cos ( x );
» plot ( x, y, x, z )
```



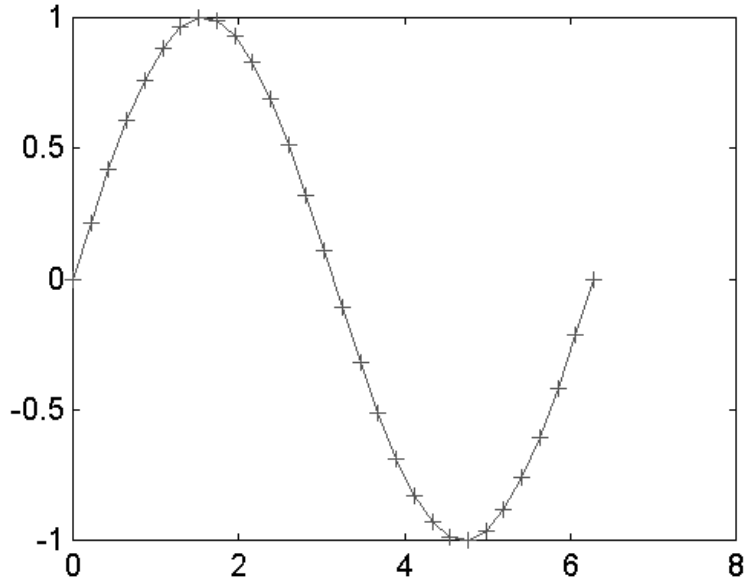
Označavanje grafika i osa u MATLAB-u se postiže naredbama *title*, *xlabel* i *ylabel.* Evo jednog primjera gdje su ilustrovane ove naredbe:

```
» x=-4*pi : pi/100 : 4*pi ;
» y=sin ( x );
» plot ( x , y )
» title ( ' Grafik funkcije  y=sin ( x ) ' )
» xlabel ( ' vrijednosti promjenjive x ' )
```



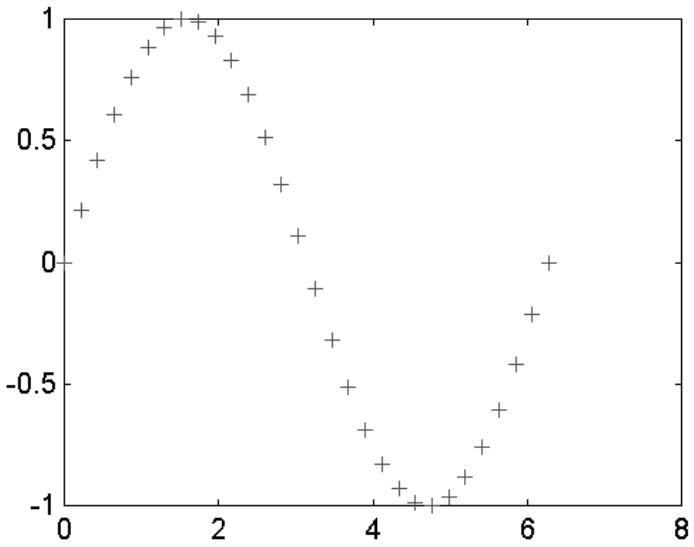
Tačke na koje je podjeljen interval u kome se funkcija prikazuje, mogu se posebno označiti.

```
» x = linspace ( 0, 2*pi, 30 );
» y=sin ( x );
» plot ( x , y , x , y , '+'  )
```



U ovom primjeru je funkcija y=sin(x) crtana dva puta. Prvi put je dat standardan prikaz date funkcije, nakon čega su vrijednosti funkcije na intervalu u kome se posmatra funkcija posebno označene. Da je u naredbi funkcije *plot* bio samo jedan argument, to bi značilo da će se vrijednosti nanositi samo na ordinatu. U tom slučaju, na apsisu se nanose indeksi elemenata (tačaka) na koje je podjeljen interval u kome se posmatra zadana funkcija. Dodatni argument "+" definiše kako će se vršiti crtanje. Ako je ovaj argument izostavljen, grafik se crta tako što se vrši linearna interpolacija između zadanih tačaka.

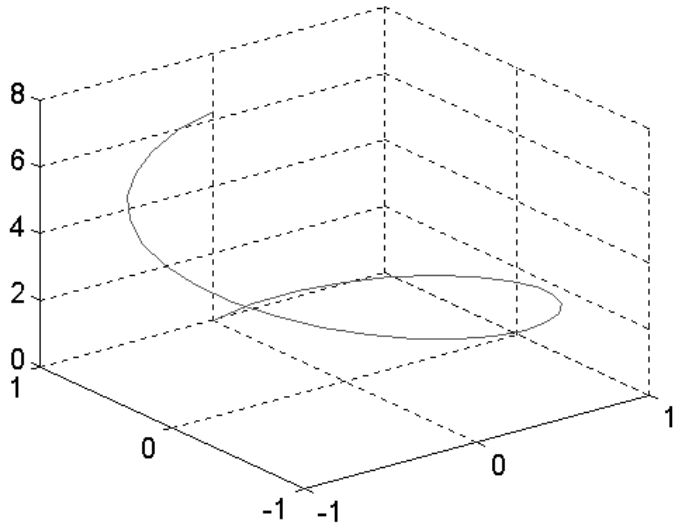
```
» x = linspace ( 0, 2*pi, 30 );
» y=sin ( x );
» plot ( x , y  , '+'  )
```



Ako se nakon otkucanih naredbi u MATLAB-u grafik ne pojavi na ekranu računara, treba na dnu ekrana kliknuti na ikonu "Figure No. 1", nakon čega se grafik pojavljuje na ekranu.

Kao što je rečeno, MATLAB može grafik da prikaže i u trodimenzionalnom obliku. Evo jednog primjera:

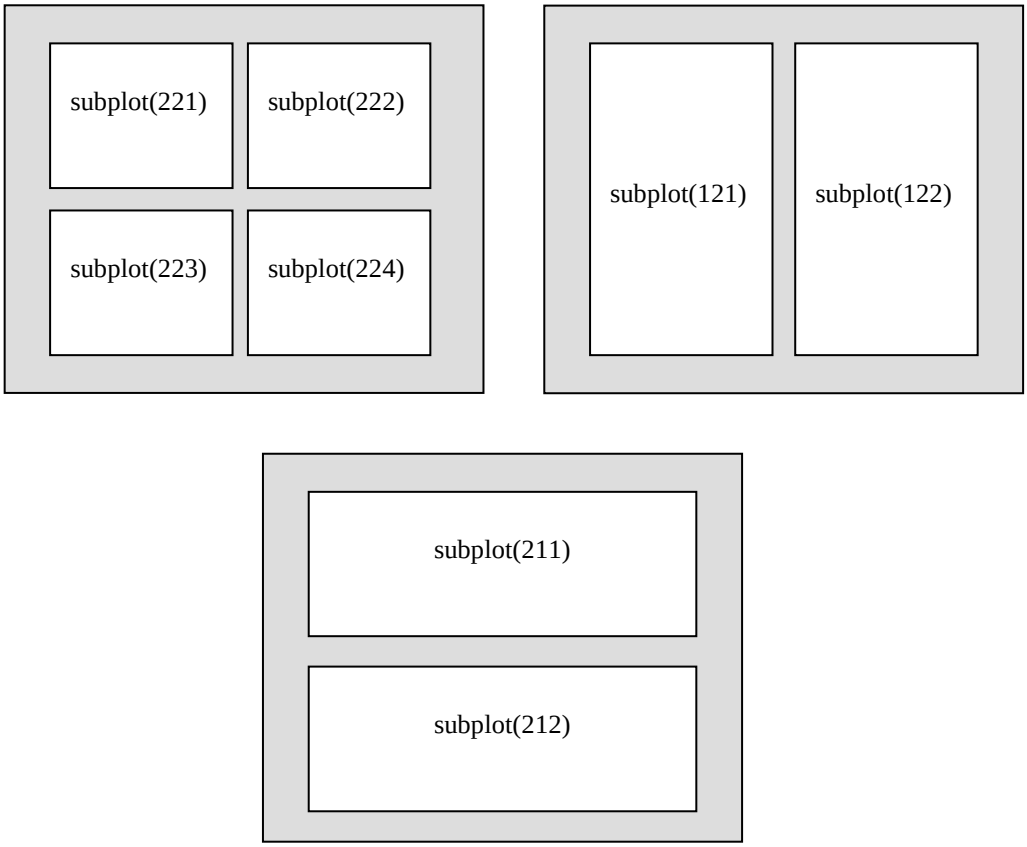
```
» plot3 ( y , z  , x  ) , grid
```



U dosadašnjim primjerima grafici su prikazivani preko cijelog ekrana. U MATLAB-u postoji mogućnost i da se nacrta više različitih grafika na jednom ekranu (maksimalno 4), za šta se koristi naredba *subplot* koja se navodi prije naredbe za crtanje. Opšti oblik ove naredbe je:

subplot(mnp)

što znači da će se ekran podjeliti na *m* dijelova po horizontali, *n* dijelova po vertikali i tekući grafik će se nacrtati u *p*-tom dijelu. Na sledećoj slici je ilustrovano na koji se način koristi *subplot* da bi se ostvarile različite podjele ekrana.



Naredbe za rad sa tekstom

Pored MATLAB-ove prvenstveno matematičke orjentacije, on posjeduje i neke naredbe za manipulisanje tekstom. U MATLAB-u se tekst jednostavno naziva stringovima. Rad sa ovakvim stringovima je sličan radu sa prostim poljima.

```
» t='How about this character string?'
t =
How about this character string?
```

Tekst se unosi između navodnih znakova. Ako je potrebno izdvojiti dio teksta unutar stringa onda se to radi sljedećom naredbom:

```
» u=t(16:24)
u =
character
```

Na ovaj način je iz teksta izdvojena riječ *character*. Ovu riječ je moguće dobiti i u obrnutom redoslijedu:

```
» u=t(24:-1:16)
u =
retcarahc
```

Kao i kod matrica tekst se može sastojati i iz nekoliko vrsta.

```
» v=[ 'Character strings having more  than  '
      'one row must have the same number '
      'of coloumns just like matrices!         ' ]
v=
Character strings having more  than
one row must have the same number
of coloumns just like matrices!
```

pri čemu broj kolona mora biti jednak.

Nad stringovima je također moguće vršiti i matematičke operacije kao i predstaviti ih u ASCII kodu. Da bi došli do ASCII koda stringa koristimo apsolutnu vrijednost:

```
» s='ABCDEFG'
s =
ABCDEFG

» m=abs( s )
m =
   65    66    67    68    69    70    71

» m=s+0
m =
   65    66    67    68    69    70    71
```

Povratak iz ASCII koda vrši se na sljedeći način:

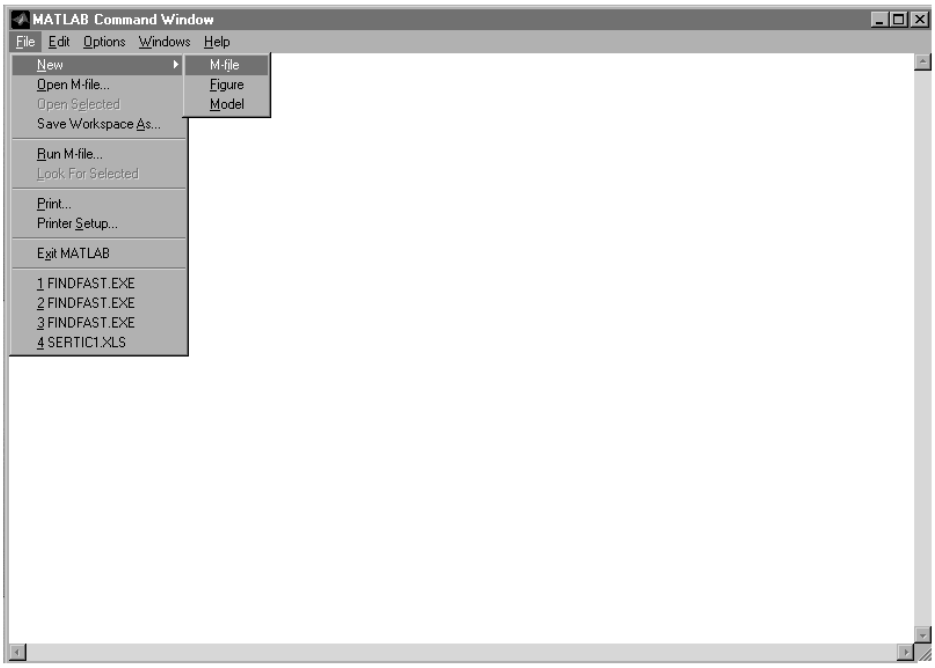
```
» setstr (m)
ans =
ABCDEFG

» n=s+5
n =
   70    71    72    73    74    75    76

» setstr (n)
ans =
FGHIJKL
```

m- fajlovi

Iz dosadašnjeg izlaganja se vidi da MATLAB odgovara na vaš postavljeni zadatak onim redoslijedom kako je to od njega zatraženo. Na neke jednostavne probleme MATLAB efikasno odgovara nakon što otkucamo određenu sekvencu naredbi. U slučaju većeg broja ovakvih naredbi, pogotovo ako je neko izračunavanje potrebno nekoliko puta vršiti za različite vrijednosti parametara programa, onda ovakav posao postaje mukotrpan. Zbog ovoga se u MATLAB-u koriste m-fajlovi. Postupak je jednostavan. Programska sekvenca koja bi se inače unosila nakon MATLAB-ovog prompta, sada se unese u m-fajl i kao takav sačuva. Da bi se kreirao m-fajl bira se opcija *New* iz *File* menija i selektuje *M-file*.



M-fajlovi su u stvari fajlovi sa ekstenzijom “m”. Nakon što je unešen i sačuvan fajl se jednostavno izvršava kucanjem njegovog imena iza matlabovog prompta.

» ime.m

Nakon ovoga MATLAB prateći linije programa vrši izračunavanje, u zavisnosti od parametara koji su unešeni u program. Prilikom dodjeljivanja imena fajlu treba paziti da ime ne bude identično nekoj MATLAB-ovoj naredbi, jer ona ima prioritet nad imenom m fajla. Ako je potrebno imati pregled izvršavanja toka programa onda se naredbom *echo on* to i omogućuje. Ovaj tip fajlova je također pogodan za unošenje polja većih dimenzija, npr. rezultata laboratorijskih mjerenja. Na ovaj način su podaci sačuvani, a jednostavno mogu i da se promjene. MATLAB posjeduje nekoliko naredbi koje nam daju informaciju o broju ovih fajlova, te naredbe za brisanje, prelazak u neki drugi direktorij, itd...

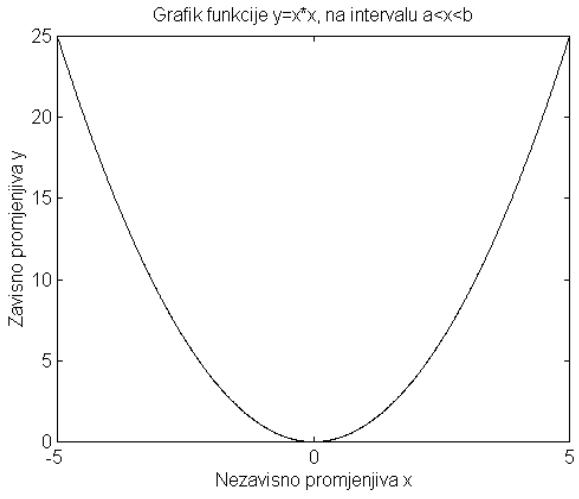
Pisanje funkcija u MATLAB-u

Prilikom korištenja MATLAB-ovih funkcija kao što su *abs*, *angle*, *sqrt*, MATLAB je koristio varijable koje smo sami zadavali nakon čega je slijedio rezultat. Osim ovoga mi možemo sami da kreiramo funkciju, pohranimo je kao m-fajl i nakon toga je pozivamo i izvršavamo.

```
function y=fakt(n)
if n<0
  disp('Greška. Faktorijel negativnog broja nije definisan')
  y=[ ]
  return
end
y=1;
ind=n;
while (ind>0),
   y=y*ind;
   ind=ind-1
end
```

Ovaj fajl se zatim sačuva kao fakt.m i poziva se kucanjem imena (fakt). Neophodno je da ime m-fajla bude identično imenu funkcije. Ako je potrebno neke vrijednosti unijeti sa tastature onda se koristi naredba *input*. Korišćenje ove naredbe najlakše je objasniti na sledećem primjeru: pretpostavimo da je potrebno nacrtati funkciju $y = x^2$ na proizvoljnom intervalu $[a,b]$ sa proizvoljnim korakom. Vrednosti a, b kao i vrednost koraka unose se sa tastature u toku izvršavanja programa. Rješenje u MATLAB-u je:

```
%program za crtanje funkcije y=x*x na proizvoljnom intervalu
a=input('Pocetak intervala (a): ');          %unosi se a sa tastature
b=input('Kraj intervala (b): ');             %unosi se b sa tastature
korak=input('korak iznosi: ');               %unosi se korak sa tastature
x=a:korak:b;
y=x.^2;
plot(x,y)
title('Grafik funkcije y=x*x, na intervalu a<x<b'),
xlabel('Nezavisno promjenjiva x'),ylabel('Zavisno promjenjiva y')
```

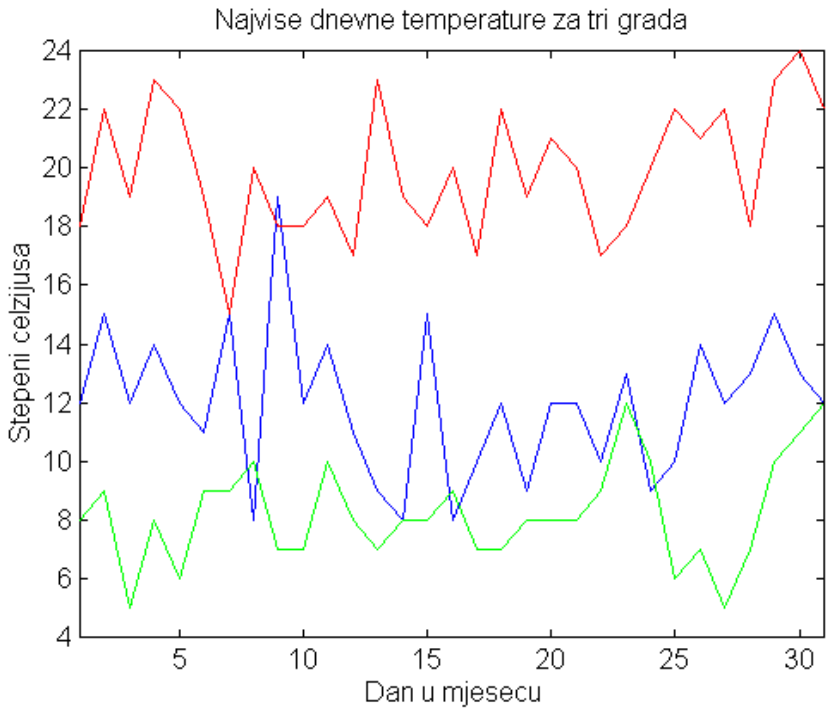


a=-5
b=5
korak=0.01

MATLAB se može koristiti i pri obradi podataka kao i setova podataka. Po svojoj unutarnjoj konvenciji ovi setovi su smješteni u matrice-kolone. Svaka kolona predstavlja različite vrijednosti mjerene varijable. Npr. pretpostavimo da su dnevne temperature za tri različita grada date u m-fajlu *temps*. Startovanjem ovog fajla dobijamo temperature u MATLAB-ovom okruženju.

```
temps =
    12     8    18
    15     9    22
    12     5    19
    14     8    23
    12     6    22
    11     9    19
    15     9    15
     8    10    20
    19     7    18
    12     7    18
    14    10    19
    11     8    17
     9     7    23
     8     8    19
    15     8    18
     8     9    20
    10     7    17
    12     7    22
     9     8    19
    12     8    21
    12     8    20
    10     9    17
    13    12    18
     9    10    20
    10     6    22
    14     7    21
    12     5    22
    13     7    18
    15    10    23
    13    11    24
    12    12    22

» d=1:31;
» plot(d,temps)
» xlabel('Dan u mjesecu'),ylabel('Stepeni celzijusa')
» title('Najvise dnevne temperature za tri grada')
```



plot naredba u ovm primjeru je kao argument imala matricu *temps*. Rezultat ovoga je grafički prikaz vrijednosti temperatura svake kolone posebno.

Relacioni i logički operatori

Ovi operatori su često korišteni u oblastima programiranja. Oni koji su se više bavili programiranjem upoznati su sa ovim. Svrha ovih operatora je u stvari da odgovore na pitanje da li je nešto tačno ili netačno. Česta upotreba ovih operatora jeste u petljama, o kojima će biti rječi nešto kasnije. Izlazna vrijednost ovih operatora jeste "1", ako je izraz tačan, odnosno "0", ako izraz nije tačan.

Relacioni operatori
Relacioni operatori koji se koriste u MATLAB-u dati su u slijedećoj tabeli

<	- manji od
>=	-manji ili jednak
>	- veći od
>=	- veći ili jednak
==	- jednak
~=	- različit od

Relacioni operatori mogu se koristiti za upoređivanje dvaju polja iste dužine, kao i za upoređivanje polja skalarom. U ovom slučaju se svaki element polja upoređuje sa skalarom i kao rezultat se dobija polje iste dužine.

```
» A=1:9,B=9-A
A =
   1   2   3   4   5   6   7   8   9
B =
   8   7   6   5   4   3   2   1   0
```

```
» tf=A>4
tf =
   0    0    0    0    1    1    1    1    1
```

U posljednjem primjeru su mjesta na kojima je zadovoljena nejednakost prikazana jedinicama.

```
» tf=A==B
tf =
   0    0    0    0    0    0    0    0    0
```

Traženi elementi polja A su ekvivalentni elementima polja B. Treba uočiti razliku između znaka jednakosti i dvostrukog znaka jednakosti. Dvostruki znak jednakosti upoređuje dvije varijable i kao rezultat vraća jedinicu ako su one jednake ili nulu ako su različite.

```
» tf=B-(A>2)
tf =
   8    7    5    4    3    2    1    0   -1
```

Vidimo da se operatori mogu koristiti i pri matematičkim operacijama. Elemente polja je moguće zamijenjiti specijalnim MATLAB-ovim brojem eps, koji iznosi aproksimativno 2.2exp(-16). Ova konkretna vrijednost je ponekad korisna da bi se izbjeglo djeljenje sa nulom.

```
» B=B+(B==0)*eps
B =
  Columns 1 through 7
   8.0000    7.0000    6.0000    5.0000    4.0000    3.0000    2.0000
  Columns 8 through 9
   1.0000    0.0000
```

Sledeći primjer izračunava vrijednosti funkcije sinc(x) uz upozorenje da je peti broj nula.

```
» x=(-3:3)/3
x =
  -1.0000   -0.6667   -0.3333         0    0.3333    0.6667    1.0000
» sin(x)./x
Warning: Divide by zero
ans =
   0.8415    0.9276    0.9816       NaN    0.9816    0.9276    0.8415
```

Pošto izraz 0/0 nije definisan dobijamo upozorenje *NaN*, sa značenjem Not a number. Ako se sada nula zamjeni sa *eps*

```
» x=x+(x==0)*eps;
» sin(x)./x
ans =
   0.8415    0.9276    0.9816    1.0000    0.9816    0.9276    0.8415
```

rezultat je korektan.

Logički operatori
Logički operatori se koriste za kombinovanje sa relacionim ili za njihovo negiranje.

& - logičko "i"
| - logičko "ili"
.~ - negacija

Evo nekoliko primjera

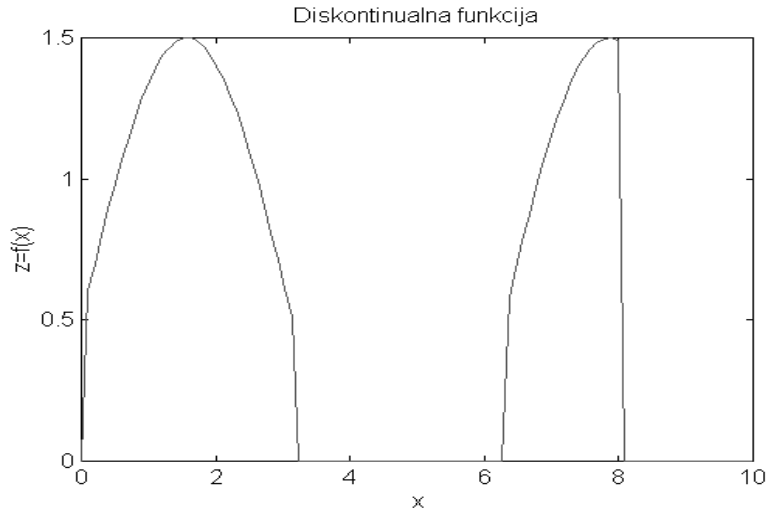
```
» A=1:9;B=9-A;
» tf=A>4
tf =
   0   0   0   0   1   1   1   1   1

» tf=~(A>4)
tf =
   1   1   1   1   0   0   0   0   0

» tf=(A>2)&(A<6)
tf =
   0   0   1   1   1   0   0   0   0
```

U posljednjem primjeru su kao rezultat vraćene jedinice za elemente koji su veći od dva i manji od šest. Na ovaj način je moguće generisati dio neke funkcije tako što se vrijednosti koje želimo da zadržimo množe sa jedinicom, a one koje ne želimo sa nulom (preporučujemo da ovo uradite u obliku m-fajla).

```
» x=linspace(0,10,100);
» y=sin(x);
» z=(y>=0).*y;
» z=z+0.5*(y>0);
» z=(x<=8).*z;
» plot(x,z)
» xlabel('x'),ylabel('z=f(x)'),
» title( 'Diskontinualna funkcija')
```



Kao dodatak gore navedenim relacionim i logičkim operatorima MATLAB posjeduje takođe niz relacionih i logičkih funkcija:

xor(x,y) - ex-ili operacija
any(x) - vraća jedinicu ako je bilo koji element vektora **x** nenulte vrijednosti ili vraća jedinicu za svaki element neke matrice **x** koji je nenulte vrijednosti
all(x) - vraća jedinicu ako su svi elementi vektora **x** različiti od nule. Ovo se odnosi i na elemente kolona matrica
isnan(x) - vraća jedinicu na mjestu elementa *NaNs*
isinf(x) - vraća jedinicu na mjestu elementa *Infs*
finite(x) - vraća jedinicu na mjestu elementa konačne vrijednosti.

Kontrolne petlje

Razni programski jezici nude mnoštvo struktura koje omogućuju kontrolu toka programa. MATLAB nudi tri vrste ovakvih petlji:

- for petlje
- while petlje
- if – else – end strukture

Zbog njihovog čestog korištenja one se smještaju u m–fajlove, jer je na taj način izbjegnuta potreba za stalnim ponovnim unošenjem ovih struktura.

For petlje

Ovaj tip petlji omogućuje da se neka grupa naredbi izvrši unapred definisan broj puta. Sintaksa ove petlje je sljedeća:

```
for x=(a:inkrement:b)
  blok naredbi
end
```

```
» for n=1:10
x(n)=sin(n*pi/10);
end

» x
x =
  Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

For petlja se ne može ograničiti, odnosno prekinuti redefinisanjem promjenjive unutar same petlje.

```
» for n=1:10
    x(n)=sin(n*pi/10);
    n=10;
   end
```

```
» x
x =
  Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

U okviru for petlje se može definisati bilo kakvo polje

```
» data=[3  9  45  6;  7  16  -1  5]
data =
     3     9    45     6
     7    16    -1     5

» for n=data
x=n(1)-n(2)
end
x =
    -4
x =
    -7
x =
    46
x =
     1
```

Također mogu da se koriste i ugniježdene for petlje

```
» for n=1:5
    for m=5:-1:1
     A(n,m)=n^2+m^2;
    end
  disp(n)
  end
     1
     2
     3
     4
     5

» A
A =
     2     5    10    17    26
     5     8    13    20    29
    10    13    18    25    34
    17    20    25    32    41
    26    29    34    41    50
```

For petlje treba izbjegavati kad god je do rješenja moguće doći korištenjem matričnog pristupa. Oba pristupa vode istom rješenju, ali je drugi brži i zahtjeva često manje kucanja.

While petlje

Sintaksa ove petlje je

```
while izraz
  blok naredbi
end
```

Blok naredbi između while i end se izvršava dok god je izraz istinit.

```
» num=0;EPS=1;
» while (1+EPS)>1
    EPS=EPS/2;
    num=num+1;
  end
» num
num =
   53
» EPS=2*EPS
EPS =
  2.2204e-016
```

U ovom primjeru je prikazan jedan od načina izračunavanja vrijednosti eps. Eps kreće od jedinice i sve dok je uslov (1+EPS)>1 zadovoljen naredbe unutar while petlje se izvršavaju. Eps se stalno djeli sa dva i kada postane tako mali da (eps+1) više nije veće od jedinice petlja se završava. Ovo se dešava zbog ograničenog broja decimalnih mjesta predviđenih za predstavljanje realnog broja.

If-else-end strukture

Sintaksa:

```
 if izraz
blok naredbi
end
```

Npr.

```
» a=10;
» c=a*25
c =
  250
» if a>5
    c=(1-20/100)*c;
  end
» c
c =
  200
```

U slučaju da postoji i druga alternativa ova struktura ima sljedeći oblik

```
if izraz
        blok naredbi koje se izvršavaju ako je izraz istinit
else
        blok naredbi koje se izvršavaju ako je izraz nije istinit
```

```
end
```

Rješeni primjeri

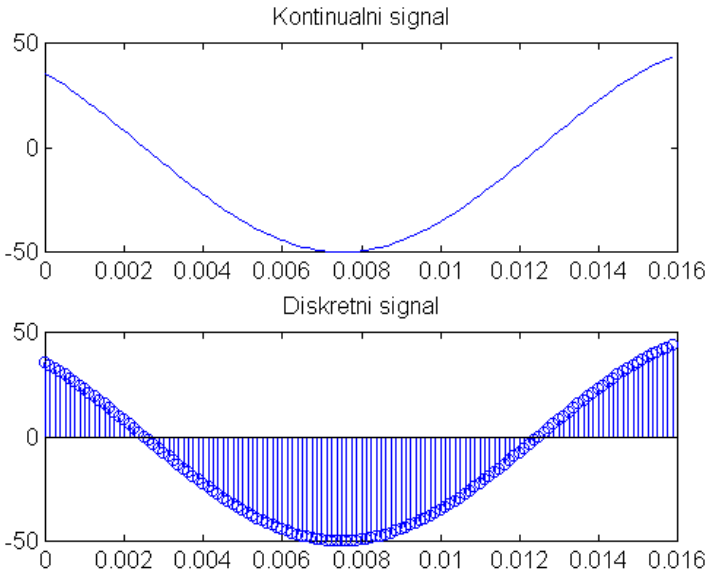
Primjer 1: ***Generisanje signala***

Generisati i nacrtati 128 odmjeraka diskretne kosinusoide sa amplitudom 50, frekvencije 50 Hz i faznim uglom $\varphi = \pi/4$. *Frekvencija odmjeravanja je 8 kHz . Uzeti da je početni vremenski trenutak t=0.*

Rješenje:

Rješenje u Matlab-u je:

```
A=50;f=50;ugao=pi/4;
fod=8000;Tod=1/fod;
tpoc=0;
t=tpoc:Tod:127*Tod;
y=A*cos(2*pi*f*t+ugao);
subplot(211),plot(t,y),title('Kontinualni signal'),pause
subplot(212),stem(t,y),title('Diskretni signal')
```



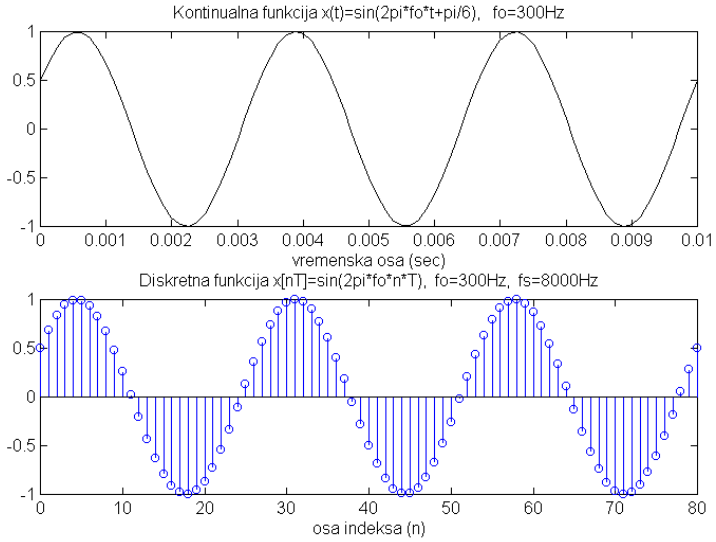
Primjer 2: ***Odabiranje signala (1)***

Dat je kontinualan signal $x(t) = \sin(2\pi f_0 t + \pi/6)$. *Ako je frekvencija sinusoide* $f_0 = 300\,\text{Hz}$ *a frekvencija odabiranja* $f_s = 8\,\text{kHz}$, *izvršiti odabiranje ovog signala na intervalu* $0-1\,\text{ms}$. *Koristeći naredbe* plot i stem, *nacrtati jedan ispod drugog "kontinualan" i diskretan signal.*

Rješenje:

Rjesenje u Matlab-u je:

```
fs=8000;Ts=1/fs;            % frekvencija i perioda obrtanja
fo=300;fi=pi/6;             % frekvencija sinusoide i fazni ugao
Tpoc=0;Tkraj=0.01;          % pocetni i krajnji vremenski trenutak
t=Tpoc:Ts:Tkraj;            % definisanje trenutaka odabiranja
x=sin(2*pi*fo*t+fi);        % diskretna sinusoida
%%%%  simulacija kontinualnog signala crtanjem pomocu naredbe plot
subplot(211),plot(t,x,'w'),xlabel('vremenska osa (sec)')
title('Kontinualna funkcija x(t)=sin(2pi*fo*t+pi/6),  fo=300Hz')
n=0:length(x)-1;            % osa indeksa za crtanje diskretnog signala
subplot(212),stem(n,x),xlabel('osa indeksa (n)')
title('Diskretna funkcija x[nT]=sin(2pi*fo*n*T),  fo=300Hz,  fs=8000Hz')
```



Primjer 3: ***Odabiranje signala (2)***

Data je kontinualna kosinusoida $x(t) = cos(2\pi f_0 t)$

a) *Ako je frekvencija odabiranja* $f_s = 8\ kHz$ *izvršiti odabiranje u intervalu 0-1 ms mijenjajući frekvenciju signala:*

$$f_{01} = 100Hz \quad , \quad f_{02} = 225Hz$$
$$f_{03} = 350Hz \quad i \quad f_{04} = 475Hz .$$

Nacrtati sva 4 slučaja na istom ekranu.

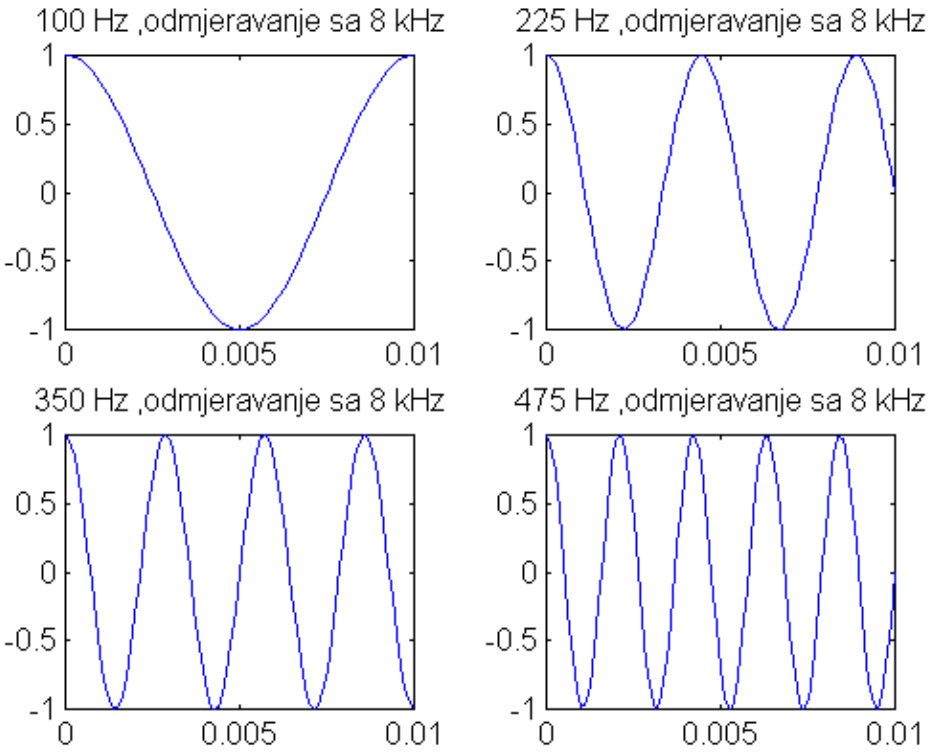
b) *Ponoviti postupak za*

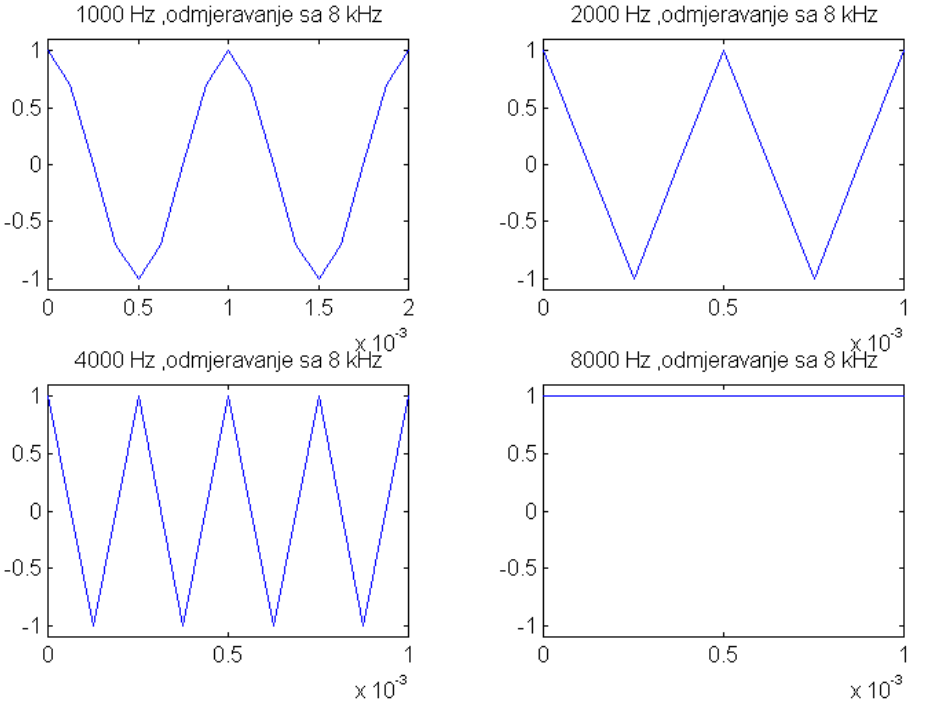
$$f_{05} = 1000Hz \quad , \quad f_{06} = 2000Hz$$
$$f_{07} = 4000Hz \quad , \quad f_{08} = 8000Hz .$$

Koristiti plot naredbu za crtanje.

Rješenje:

```
fs=8000;Ts=1/fs;
f1=100;f2=225;f3=350;f4=475;
f5=1000;f6=2000;f7=4000;f8=8000;
Tstart=0;Tend=0.01;
t=Tstart:Ts:Tend;
subplot(221),plot(t,cos(2*pi*f1*t)),title('100 Hz ,odmjeravanje sa 8 kHz');
subplot(222),plot(t,cos(2*pi*f2*t)),title('225 Hz ,odmjeravanje sa 8 kHz');
subplot(223),plot(t,cos(2*pi*f3*t)),title('350 Hz ,odmjeravanje sa 8 kHz');
subplot(224),plot(t,cos(2*pi*f4*t)),title('475 Hz ,odmjeravanje sa 8 kHz');
figure
subplot(221),plot(t,cos(2*pi*f5*t)),title('1000 Hz ,odmjeravanje sa 8 kHz');
axis([0 0.002 -1.1 1.1]);
subplot(222),plot(t,cos(2*pi*f6*t)),title('2000 Hz ,odmjeravanje sa 8 kHz');
axis([0 0.001 -1.1 1.1]);
subplot(223),plot(t,cos(2*pi*f7*t)),title('4000 Hz ,odmjeravanje sa 8 kHz');
axis([0 0.001 -1.1 1.1]);
subplot(224),plot(t,cos(2*pi*f8*t)),title('8000 Hz ,odmjeravanje sa 8 kHz');
axis([0 0.001 -1.1 1.1]);
```


100 Hz ,odmjeravanje sa 8 kHz
225 Hz ,odmjeravanje sa 8 kHz
350 Hz ,odmjeravanje sa 8 kHz
475 Hz ,odmjeravanje sa 8 kHz
1
0.5
0
-0.5
-1
0
0.005
0.01


1000 Hz ,odmjeravanje sa 8 kHz
2000 Hz ,odmjeravanje sa 8 kHz
4000 Hz ,odmjeravanje sa 8 kHz
8000 Hz ,odmjeravanje sa 8 kHz
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
x 10^-3

Primjer 4: ***Frekvencijski modulisan signal***

Neka je dat frekvencijski modulisan signal $x(t) = sin(2\pi f(t)t)$ *čija se frekvencija mijenja linearno po zakonu* $f(t) = at + b$, *gdje je* $a = 50Hz/s$ *i* $b = 10Hz$.

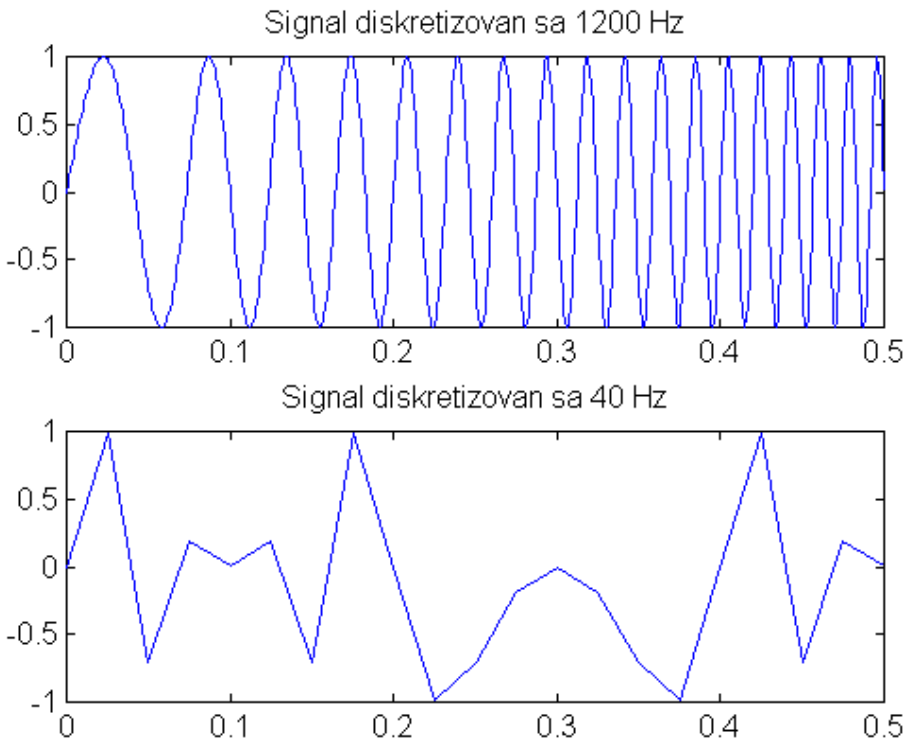
a) *Ako se signal x(t) posmatra u vremenskom intervalu od 0 do 0,5s odrediti frekvencijski opseg koji obuhvata ovaj signal. Odrediti minimalnu frekvenciju odabiranja* $f_{s\,min}$ *tako da se izvrši pravilna diskretizacija ovog signala.*

b) *Ako je frekvencija odabiranja 1200Hz nacrtati tako diskretizovan signal. Koristiti plot naredbu.*

c) *Ponoviti prethodni postupak sa frekvencijom odabiranja od 40Hz.*

Rješenje:

```
clear
fs=1200;Ts=1/fs;
t=0:Ts:0.5;
f=50*t+10;
x=sin(2*pi*f.*t);
subplot(211),plot(t,x)
title('Signal diskretizovan sa 1200 Hz');
fs=40;Ts=1/fs;
t=0:Ts:0.5;
f=50*t+10;
y=sin(2*pi*f.*t);
subplot(212),plot(t,y)
title('Signal diskretizovan sa 40 Hz');
```

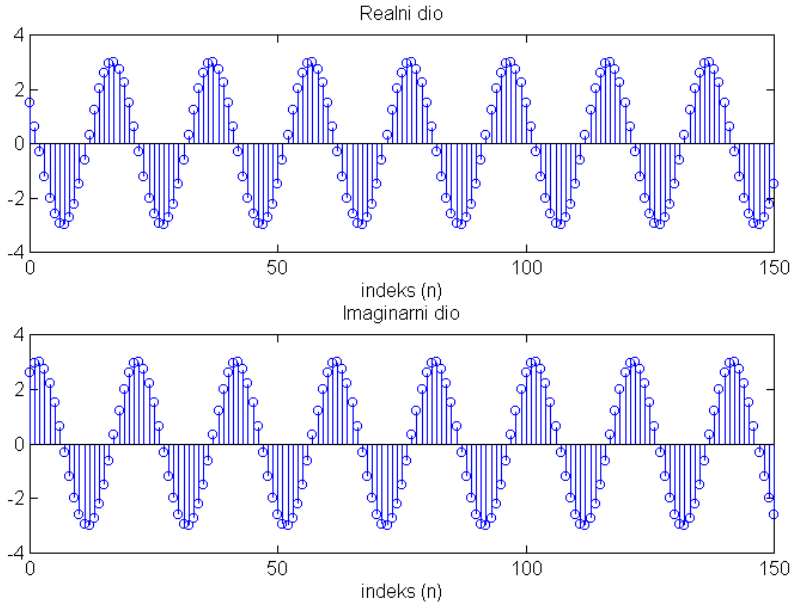


Primjer 5: ***Kompleksni signali***

Generisati kompleksnu eksponencijalnu sekvencu $e[n] = |A|\, e^{j(\omega n+\varphi)}$ *frekvencije* $f = 0.05$, *amplitude A=3 i početne faze* $\pi/3$.

Rješenje:

```
n=0:150;
f=0.05;
omega=2*pi*f;
A=3;
faza=pi/3;
e=A*exp(j*(omega*n+faza));
subplot(211),stem(n,real(e));
title('Realni dio'),xlabel('indeks (n)')
subplot(212),stem(n,imag(e));
title('Imaginarni dio'),xlabel('indeks (n)')
```



Primjer 6: ***Računanje konvolucije***

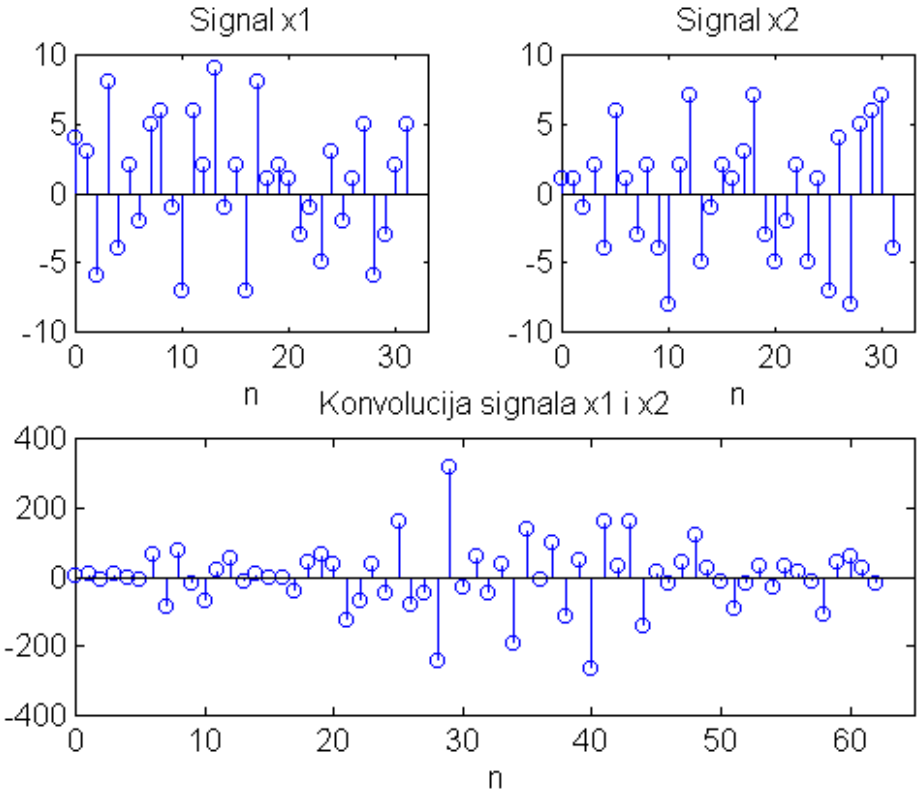
Izračunati konvoluciju signala:

$x_1 = [4\ 3\ -6\ 8\ -4\ 2\ -2\ 5\ 6\ -1\ -7\ 6\ 2\ 9\ -1\ 2\ -7\ 8\ 1\ 2\ 1\ -3\ -1\ -5\ 3\ -2\ 1\ 5\ -6\ -3\ 2\ 5]$

$x_2 = [1\ 1\ -1\ 2\ -4\ 6\ 1\ -3\ 2\ -4\ -8\ 2\ 7\ -5\ -1\ 2\ 1\ 3\ 7\ -3\ -5\ -2\ 2\ -5\ 1\ -7\ 4\ -8\ 5\ 6\ 7\ -4]$

Rješenje:

```
x1=[4 3 -6 8 -4 2 -2 5 6 -1 -7 6 2 9 -1 2 -7 8 1 2 1 -3 -1 -5 3 -2 1 5 -6 -3 2 5];
x2=[1 1 -1 2 -4 6 1 -3 2 -4 -8 2 7 -5 -1 2 1 3 7 -3 -5 -2 2 -5 1 -7 4 -8 5 6 7 -4];
y=conv(x1,x2);
Nx1=length(x1);
Nx2=length(x2);
Ny=Nx1+Nx2-1;
nx1=0:Nx1-1;
nx2=0:Nx2-1;
ny=0:Ny-1;
subplot(221),stem(nx1,x1),title('Signal x1'),xlabel('n')
axis([0 33 -10 10])
subplot(222),stem(nx2,x2),title('Signal x2'),xlabel('n')
axis([0 33 -10 10])
subplot(212),stem(ny,y),title('Konvolucija signala x1 i x2'),xlabel('n')
axis([0 65 -400 400])
```



Primjer 7: ***Teorema o konvoluciji***

Provjeriti teoremu o konvoluciji na primjeru nizova:

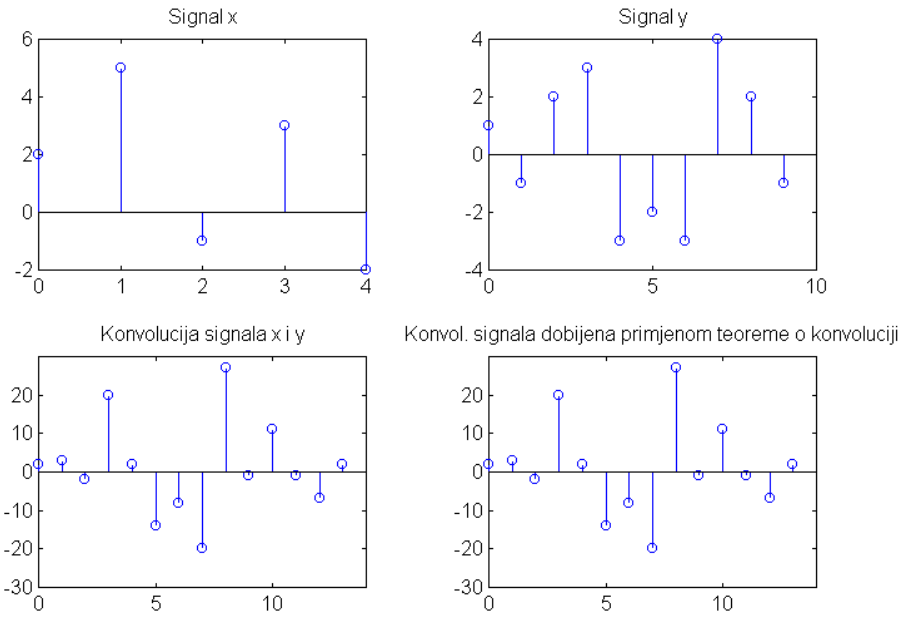
$$x = [2\ 5\ -1\ 3\ -2]\ i\ y = [1\ -1\ 2\ 3\ -3\ -2\ -3\ 4\ 2\ -1]$$

i dobijene rezultate predstaviti grafički.

Rješenje:

Jedno od rješenja je

```
x=[2 5 -1 3 -2];
y=[1 -1 2 3 -3 -2 -3 4 2 -1];
x1=length(x);
x0=0:x1-1;
subplot(221),stem(x0,x),title('Signal x');
y1=length(y);
y0=0:y1-1;
subplot(222),stem(y0,y),title('Signal y');
zconv=conv(x,y);
t=length(zconv);
t0=0:t-1;
subplot(223),stem(t0,zconv),title('Konvolucija signala x i y')
axis([0 14 -30 30])
x2=[x zeros(1,t-x1)];
y2=[y zeros(1,t-y1)];
X=fft(x2);
Y=fft(y2);
Z=X.*Y;
z=ifft(Z);
z1=real(z);
subplot(224),stem(t0,z1)
title('Konvol. signala dobijena primjenom teoreme o konvoluciji')
axis([0 14 -30 30])
```



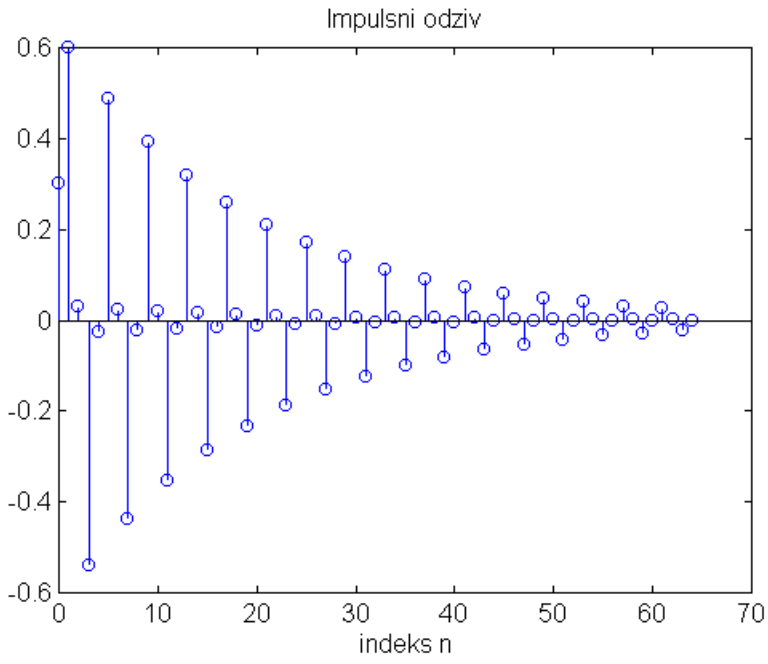
Primjer 8: ***Impulsni odziv sistema***

Izračunati i nacrtati 64 odmjerka impulsnog odziva sistema koji je opisan sa

$$y(n)+0{,}9y(n-2)=0{,}3x(n)+0{,}6x(n-1)+0{,}3x(n-2)$$

Rješenje:

```
a=[1 0 0.9];
b=[0.3 0.6 0.3];
impuls=[1 zeros(1,64)];
h=filter(b,a,impuls);
n=0:length(h)-1;
stem(n,h),title('Impulsni odziv'),xlabel('indeks n');
```



Primjer 9: ***Digitalni filter***

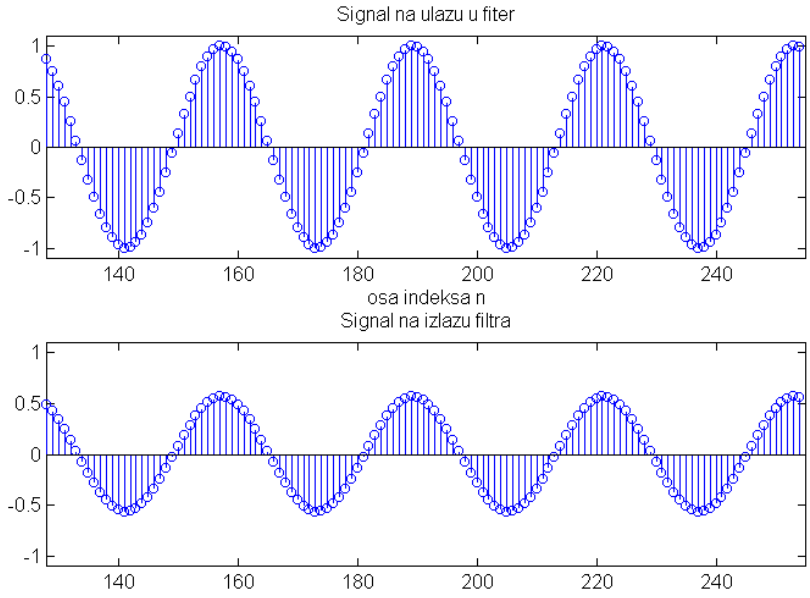
Na ulaz sistema opisanog jednačinom diferencija

$$y(n)+1{,}9y(n-1)+1{,}5y(n-2)+0{,}4y(n-3)=0{,}1x(n)+2x(n-1)+0{,}5x(n-2)+0{,}1x(n-3)$$

dovesti 256 odmjeraka sinusoide čija je kružna frekvencija $\omega_0=\pi/16$ *a faza* $\varphi=2\pi/3$. *Nacrtati, jedan ispod drugog, u istoj razmjeri, signale na ulazu i izlazu, od 128 do 255 odmjerka.*

Rješenje:

```
n=0:255;
omega=pi/16;
a=[1 1.9 1.5 0.4];
b=[0.1 2 0.5 0.1];
x=sin(omega*n+2*pi/3);
y=filter(b,a,x);
subplot(211),stem(n(128:255),x(128:255));
title('Signal na ulazu u fiter');
xlabel('osa indeksa n'),axis([128 255 -1.1 1.1]);
subplot(212),stem(n(128:255),y(128:255));
title('Signal na izlazu filtra');
axis([128 255 -1.1 1.1]);
```



Primjer 10: ***Izračunavanje impulsnog odziva digitalnog filtra***

Jedan digitalni filter opisan je sledećom funkcijom prenosa:

$$H(z)=0.067569\frac{1+2z^{-1}+2z^{-2}}{1-1.1421z^{-1}+0.41421z^{-2}}.$$

a)*Izračunati i nacrtati u Matlab-u impulsni odziv ovog filtra u 50 tačaka.*

b)*Na ulaz filtra se dovodi 128 odbiraka signala*

$$x(t)=\cos(2\pi F_1 t)+\cos(2\pi F_2 t)+\cos(2\pi F_3 t),$$

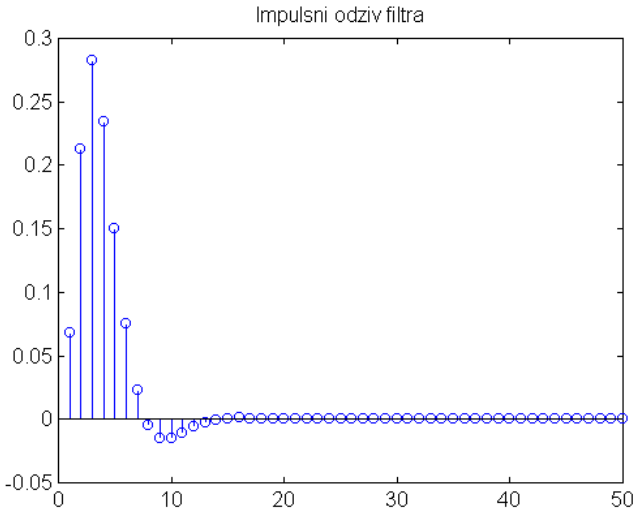
F_1=500 Hz, F_2=750 Hz, F_3=3000 Hz

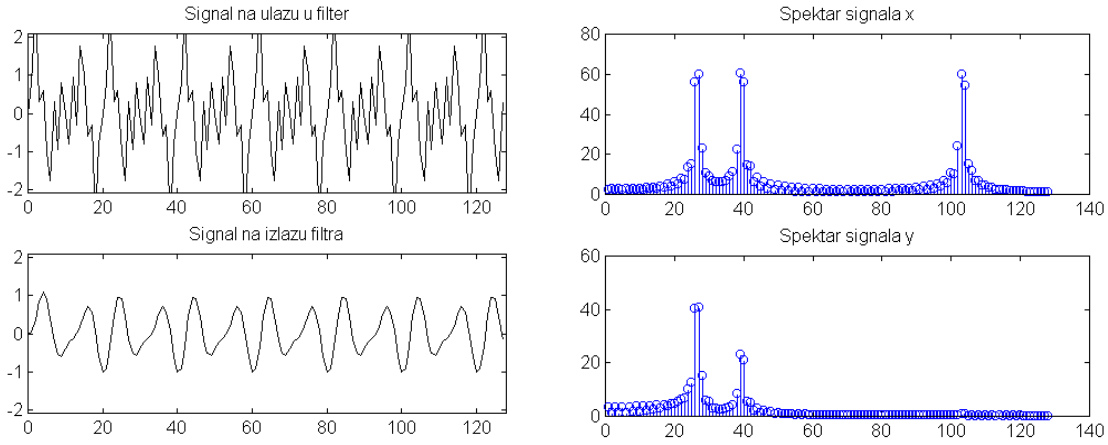
Signal je diskretizovan sa frekvencijom odabiranja F_S=10 kHz. *Izračunati i nacrtati signal na izlazu.*

c)*Nacrtati spektre signala na ulazu i izlazu filtra.*

Rješenje:

```
%%%% a)
b=0.067569*[1 2 1];a=[1 -1.1421 0.4142];
impuls=[1 zeros(1,49)];h=filter(b,a,impuls);
stem(h),title('Impulsni odziv filtra')
pause
%%%% b)
clg
N=128;
b=0.067569*[1 2 1];a=[1 -1.1421 0.4142];
F1=500;F2=750;F3=3000;Fs=5000;
n=0:N-1;
f1=F1/Fs;f2=F2/Fs;f3=F3/Fs;
x=sin(2*pi*f1*n)+sin(2*pi*f2*n)+sin(2*pi*f3*n);
subplot(211),plot(n,x,'w'),title('Signal na ulazu u filter'),axis([0 N -2.1 2.1]);
y=filter(b,a,x);
subplot(212),plot(n,y,'w'),title('Signal na izlazu filtra'),axis([0 N -2.1 2.1]);
pause
%%%% c)
clg
X=abs(fft(x,256));Y=abs(fft(y,256));
subplot(211),stem(X(1:length(X)/2)),title('Spektar signala x')
subplot(212),stem(Y(1:length(Y)/2)),title('Spektar signala y')
```





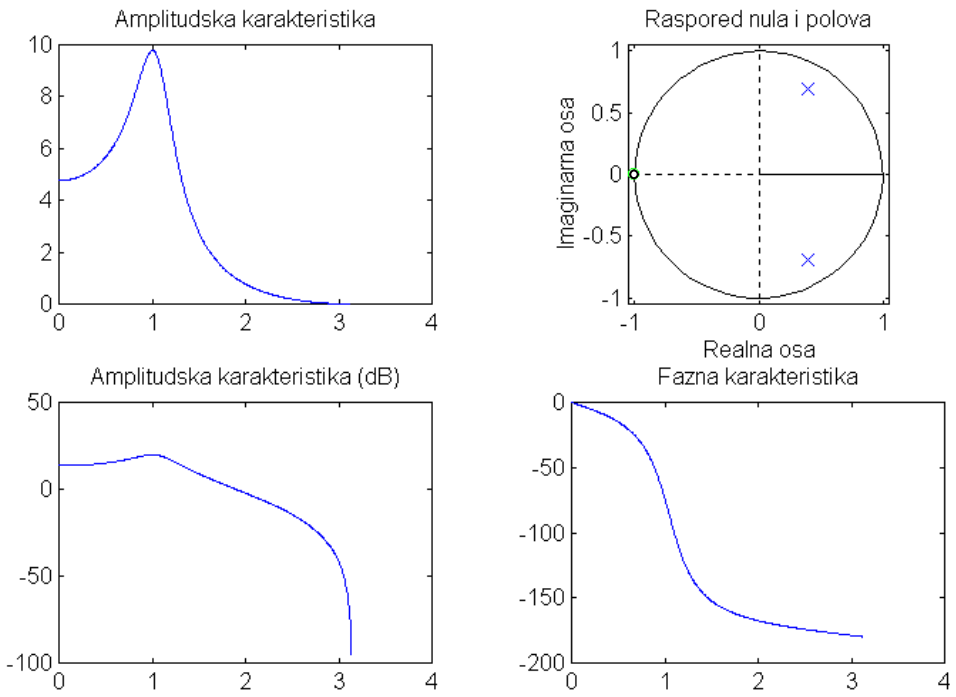
Primjer 11: ***NF filter***

Napisati Matlab program koji za zadate parametre r i θ, crta amplitudsku i faznu karakteristiku NF filtra, kao i raspored nula i polova.

Rješenje:

```
%%%  Funkcija prenosa filtra propusnika niskih frekvencija
clc,clg
G=1;
r=input('r=');
teta=input('teta=');
b=[1 2 1];a=[1 -2*r*cos(teta) r*r]';
nule=roots(b);polovi=roots(a);
[h,w]=freqz(b,a,512);ampl=abs(h);phase=180*angle(h)/pi;
%%  crtanje rasporeda nula i polova NF filtra
subplot(222),zgrid(1,0,'new'),pzmap(polovi,nule),xlabel('Realnaosa')
ylabel('Imaginarna osa')
opseg=1.05*max(max(abs(nule)),max(abs(polovi)));
if opseg==0, opseg1;end
axis([-opseg opseg -opseg opseg]),axis('square')
title('Raspored nula i polova')
%%  crtanje amplitudske karakteristike
subplot(221),plot(w,ampl)
title('Amplitudska karakteristika')
%%  crtanje amplitudske karakteristike u dB
subplot(223),plot(w,20*log10(ampl))
title('Amplitudna karakteristika (dB)')
%%  crta faznu karakteristiku
subplot(224),plot(w,phase),title('Fazna karakteristika')
```

Na slici su prikazane frekvencijske karakteristike i raspored nula i polova NF filtra za vrednosti $r = 0.8$ i $\theta = \pi/3$



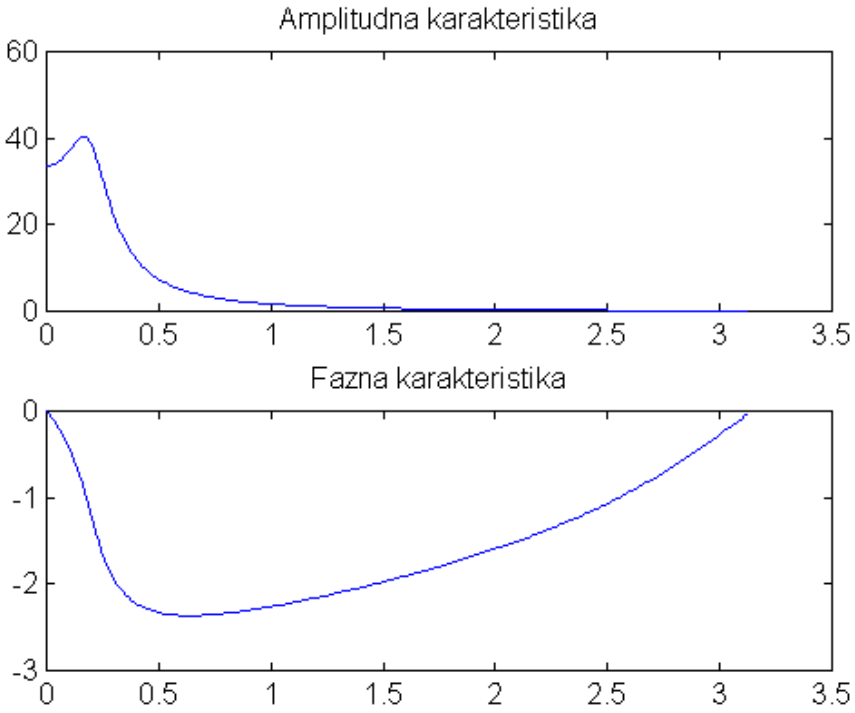
Primjer 12: ***Amplitudna i fazna karakteristika digitalnog filtra***

Nacrtati amplitudnu i faznu karakteristiku digitalnog filtra datog sa

$$y(n)-1{,}8\cos\left(\frac{\pi}{16}\right)y(n-1)+0{,}81y(n-2)=x(n)+0{,}5x(n-1)$$

Rješenje:

```
b=[1 0.5];
a=[1 -1.8*cos(pi/16) 0.81];
[H,omega]=freqz(b,a,256);
Hampl=abs(H);
Hfazna=angle(H);
subplot(211),plot(omega,Hampl),title('Amplitudna karakteristika');
subplot(212),plot(omega,Hfazna),title('Fazna karakteristika');
```



Primjer 13: ***Projektovanje analognog Butterworth-ovog filtra***

Projektovati analogni Batervortov filtar, čija je granična frekvencija $\Omega_p = 1\text{rad/sec}$.
Uzeti da je red filtra:

a) $N=2$ *;* ***b)*** $N=4$ *;* ***c)*** $N=8$

Na istom grafiku nacrtati amplitudske karakteristike sva tri filtra. Nacrtati raspored polova za sva tri slučaja.

Rješenje:

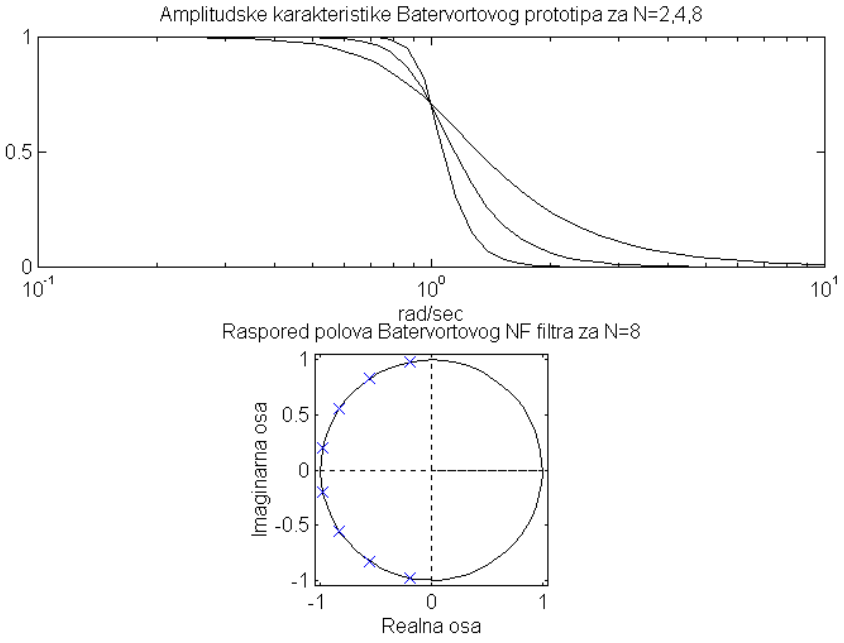
Prilikom projektovanja IIR filtara korišćenjem Batervortove aproksimacije uobičajeno je da se vrši diskretizacija funkcije prenosa čija je granična frekvencija $\Omega_p = 1\text{rad/sec}$. Takav filtar se još naziva analogni NF prototip ili normalizovani filtar. U Matlab-u se funkcija prenosa analognog Batervortovog prototipa dobija naredbom $[\text{z},\text{p},\text{k}]=\text{buttap}[\text{N}]$, gdje je N željeni red funkcije prenosa a z, p i k su nule, polovi i pojačanje dobijenog filtra, respektivno. Naredbom buttap se može projektovati samo filtar sa slabljenjem od 3dB na granici propusnog opsega. Ukoliko treba sintetizovati neki drugi filtar, potrebno je izvršiti skaliranje

korijena funkcije prenosa pomoću:

$$s_k = \frac{\Omega_p}{\sqrt[N]{\varepsilon}} e^{j\pi\frac{2k+N-1}{2N}} \quad , \quad k = 1,2,\ldots,2N$$

Rješenje zadatka u Matlab-u može se izvesti na sledeći način:

```
clc,clg
N=2;
[z,p,k]=buttap(N);b=k*poly(z);a=poly(p);
w=logspace(-1,1);h=freqs(b,a,w);H2=abs(h);          %ampl. karakteristika za N=2
N=4;
[z,p,k]=buttap(N);b=k*poly(z);a=poly(p);
w=logspace(-1,1);h=freqs(b,a,w);H4=abs(h);          %ampl. karakteristika za N=4
N=8;
[z,p,k]=buttap(N);b=k*poly(z);a=poly(p);
w=logspace(-1,1);h=freqs(b,a,w);H8=abs(h);          %ampl. karakteristika za N=8
subplot(211),semilogx(w,H2,w,H4,w,H8),xlabel('rad/sec')
title('Amplitudske karakteristike Batervortovog prototipa za N=2,4,8')
%%%%%%%%%  Raspored nula i polova za N=8   %%%%%%%%%%%%
subplot(212),zgrid(1,0,'new'),pzmap(b,a)
title('Raspored polova Batervortovog NF filtra za N=8')
axis([-1.05 1.05 -1.05 1.05]),axis('square')
```



Primjer 14: ***Nule i polovi funkcije prenosa***

Za svaku od sledećih funkcija prenosa, odrediti u Matlab-u nule, polove i pojačanje k i nacrtati njihov položaj u z ravni.

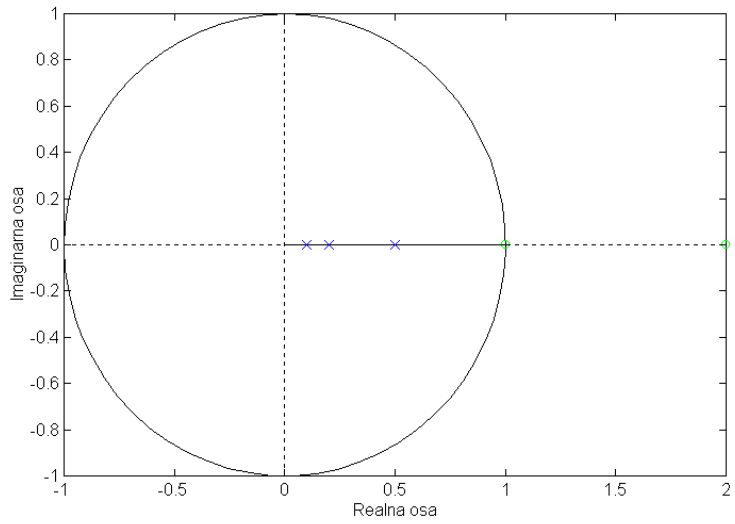
a) $H(z) = \dfrac{z^2 - 6z + 4}{z^4 - z^3 + 0{,}33z^2 - 0{,}044z + 0{,}002}$, ***b)*** $H(z) = \dfrac{5z - 4}{z^4 - 3{,}2z^3 + 4{,}01z^2 - 2{,}542z + 0{,}732}$

Rješenje:

Nule i polovi funkcije prenosa se u Matlab-u mogu nacrtati naredbom pzmap(num,den). U vektor num (*numerator*) se smještaju koeficijenti polinoma u brojniku funkcije prenosa, a u vektor den (*denominator*) se smještaju koeficijenti u nazivniku funkcije prenosa. Naredbom zgrid, zadaje se podjela za z domen. U suprotnom, crtanje se vrši sa podjelom za s domen, odnosno nule i polovi se crtaju kao nule i polovi kontinualne funkcije

prenosa $H(s)$.

```
%%%%%%      resenje
%%% a)
brojnik=[2 -6 4];
nazivnik=[1 -1 0.33 -0.044 0.002];
k=brojnik(1)/nazivnik(1);
zo=roots(brojnik);zp=roots(nazivnik);
clc
disp('Nule funkcije prenosa su: ');
disp(zo);
disp('Polovi funkcije prenosa su: ');
disp(zp);
disp('Pojacanje k je: ');
disp(k);
pause,clg,zgrid(1,0,'new'),pzmap(brojnik,nazivnik)
xlabel('Realna osa'),ylabel('Imaginarna osa')
```



Primjer 15: ***Stabilnost kauzalnog sistema***

Ispitati stabilnost kauzalnog diskretnog sistema opisanog sledećom funkcijom prenosa

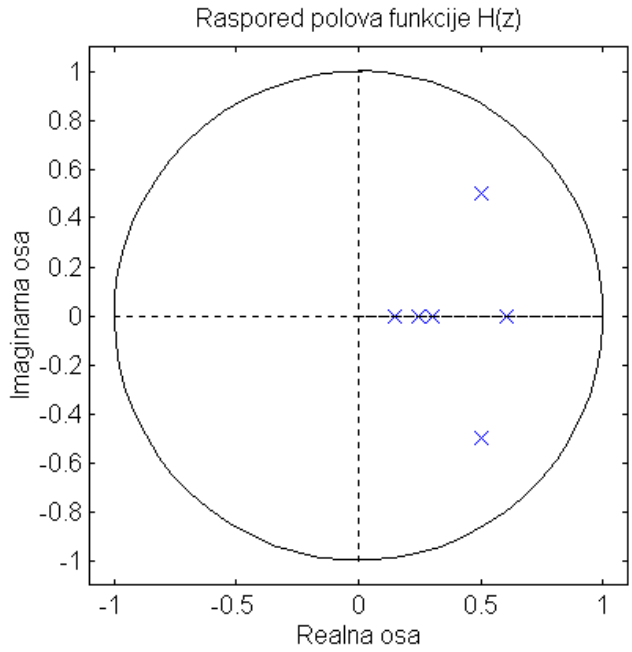
$$H(z)=\frac{1}{z^6-2{,}3z^5+2{,}3756z^4-1{,}33z^3+0{,}3989z^2-0{,}0589z+0{,}0033}$$

Rješenje:

Sledeći Matlab program ispituje stabilnost sistema proverom da li svi polovi leže unutar jediničnog kruga, odnosno da li je njihova apsolutna vrednost manja od jedan. Nakon toga program crta raspored polova u odnosu na jedinični krug.

```
clc,clg
brojnik=[1];nazivnik=[1 -2.3 2.3756 -1.33 0.3989 -0.0589 0.0033];
nule=roots(brojnik);polovi=roots(nazivnik);
disp('Polovi funkcije prenosa su: ');
disp(polovi);
%%%%% ispitivanje da li su svi polovi unutar jedinicnog kruga
if abs(polovi)<1,disp('Sistem je stabilan')
        else disp('Sistem nije stabilan');end
pause
%%%%% crtanje rasporeda polova u odnosu na jedinicni krug
zgrid(1,0,'new'),pzmap(1,nazivnik),xlabel('Realna osa'),ylabel('Imaginarna osa');
opseg=1.05*max(abs(polovi));if opseg<1,opseg=1.1;end
```

```
axis([-opseg opseg -opseg opseg]),axis('square')
title('Raspored polova funkcije H(z)')
```



Raspored polova sistema opisanog zadatom funkcijom prenosa prikazan je na slici. U prethodnom programu, treba obratiti pažnju na dio kojim se ispituje da li je apsolutna vrijednost svakog pola manja od jedan. Standardno rješenje u nekom programskom jeziku, bila bi upotreba for petlje kojom se ispituje da li je apsolutna vrijednost svakog elementa vektora polovi manja od jedan. U Matlab-u se ovaj problem može rješiti mnogo elegantnije, uz uštedu računarskog vremena, pošto logički operatori imaju mogućnost da rade i sa vektorima i sa matricama. Tako sledeća linija koda

```
[0.1 0.5 0.9 2 3 4]<1
```

vraća kao odgovor promjenljivu [1 1 1 0 0 0] u kojoj 0 označava FALSE (netačno) a 1 označava TRUE (tačno). Upotrebom ovakvog načina programiranja u Matlabu i čestim korištenjem vektorskih operacija, mogu se izbjeći višestruke for petlje čime se program značajno ubrzava.

Prilog – naredbe MATLAB-a

U ovom prilogu su ukratko objašnjene naredbe iz Signal Processing Toolbox-a, kao i neke osnovne narebe MATLAB-a koje se odnose na digitalnu obradu signala.

PROZORSKE FUNKCIJE	
NAREDBA	**ZNAČENJE**
bartlett	Bartletov prozor
blackman	Blekmenov prozor
boxcar	Pravougaoni prozor
chebwin	Čebiševljev prozor
hamming	Hamingov prozor
hanning	Hanov prozor
kaiser	Kajzerov prozor
triang	Trougaoni prozor
NAREDBE KOJE SE KORISTE PRI PROJEKTOVANJU FILTERA	
NAREDBA	**ZNAČENJE**
besselap	Projektovanje Beselovog analognog NF prototipa
besself	Projektovanje analognog Beselovog filtra
buttap	Batervortov analogni NF prototip
butter	Projektovanje Batervortovog digitalnog ili analognog filtra
buttord	Određivanje potrebnog reda Batervortovog filtra
cheb1ap	Projektovanje Čebiševljevog analognog NF prototipa prve vrste
cheb1ord	Određivanje potrebnog reda za Čebiševljev filter prve vrste
cheb2ap	Projektovanje Čebiševljevog analognog NF prototipa druge vrste
cheb2ord	Određivanje potrebnog reda za Čebiševljevog filtar druge vrste
cheby1	Projektovanje direktnog Čebiševljevog digitalnog ili analognog filtra
cheby2	Projektovanje inverznog Čebiševljevog digitalnog ili analognog filtra
ellip	Projektovanje eliptičkog digitalnog ili analognog filtra
ellipap	Projektovanje eliptičkog analognog NF prototipa
ellipord	Određivanje potrebnog reda za eliptički filter
fir1	Projektovanje fir filtra primjenom prozorskih funkcija
fir2	Projektovanje fir filtra primjenom prozorskih funkcija sa proizvoljnom frekvencijskom karakteristikom
firls	Projektovanje fir filtra linearne faze minimizacijom srednje kvadrane greške
lp2bp	Transformacija niskopropusnog filtra u propusnik opsega
lp2bs	Transformacija niskopropusnog filtra u nepropusnik opsega
lp2hp	Transformacija niskopropusnog filtra u visokopropusni
lp2lp	Transformacija niskopropusnog filtra u niskopropusni
prony	Pronijev metod za projektovanje IIR filtera u vremenskom domenu
remez	Projektovanje fir filtara primjenom Parks-MekKlelanovog algoritma
remezord	Procjena reda fir filtra
remlpord	Procjena reda fir filtra propusnika niskih učestanosti
yulewalk	Projektovanje IIR filtra primjenom metoda minimalne kvadratne greške
NAREDBE ZA DECIMACIJU I INTERPOLACIJU	
NAREDBA	**ZNAČENJE**
decimate	Decimacija nakon niskopropusnog filtriranja
interp	Interpolacija signala
intfilt	Projektovanje interpolacionog (decimalnog) filtra
resample	Promjena frekvencije odabiranja
NAREDBE KOJE SE KORISTE ZA FILTRIRANJE	
NAREDBA	**ZNAČENJE**
filter	Implementacija digitalnog filtra
fftfilt	Fir filtriranje primjenom FFT-a
filtfilt	Filtriranje sa nultom fazom
filtic	Formira početne uslove za naredbu *filter*

medfilt1	Jednodimenzioni median filter
NAREDBE ZA GENERISANJE POSEBNIH FUNKCIJA I MATRICA	
NAREDBA	**ZNAČENJE**
bessel	Beselove funkcije
diric	Dirihleova ili periodična sinc funkcija
sawtooth	Povorka trougaonih impulsa
sinc	Funkcija sin(pi*x)/(pi*x)
square	Povorka pravougaonih impulsa
NAREDBE KOJE SE KORISTE ZA MODULACIJU/DEMODULACIJU	
NAREDBA	**ZNAČENJE**
demod	Demodulacija signala
modulate	Modulacija signala
vco	Naponom kontrolisan oscilator
RAZLIČITE TRANSFORMACIJE SIGNALA I POMOĆNE NAREDBE	
NAREDBA	**ZNAČENJE**
bilinear	Bilinearna transformacija
czt	Chirp z-transformacija
dct	Diskretna kosinusna transformacija
dftmtx	DFT matrica
fft	Diskretna Furijeova transformacija
hilbert	Hilbertova transformacija
idct	Inverzna diskretna kosinusna transformacija
ifft	Inverzna Furijeova transformacija
impinvar	Impulsno invarijantna transformacija
NAREDBE KOJE SE KORISTE U SPEKTRALNOJ ANALIZI	
NAREDBA	**ZNAČENJE**
cceps	Kompleksni kepstrum
cohere	Estimacija koherencije dva signala
csd	Estimacija združene gustine snage
psd	Estimacija spektralne gustine snage
rceps	Realni kepstrum
specgram	Izračunava spektrogram signala
specplot	Crta izlaz naredbe *spectrum*
spectrum	Estimacija spektralne gustine snage za jedan ili dva ulazna signala
xcorr	Procjena kroskorelacione funkcije
xcorr2	Dvodimenzionalna kroskorelacija
NAREDBE ZA PREDSTAVLJANJE I ANALIZU DISKRETNIH I KONTINUALNIH FUNKCIJA PRENOSA	
NAREDBA	**ZNAČENJE**
freqs	Frekvencijski odziv analognog filtra
freqz	Frekvencijski odziv digitalnog filtra
grpdelay	Grupno kašnjenje digitalnog filtra
impz	Impulsni odziv digitalnog filtra
invfreqs	Analogni filter koji odgovara zadatom frekvencijskom odzivu
invfreqz	Digitalni filter koji odgovara zadatom frekvencijskom odzivu
lpc	Koeficijenti lineane predikcije
poly2rc	Izračunava refleksione koeficijente fir filtra
polystab	Stabilizacija polinoma
rc2poly	Izračunava koeficijente fir filtra iz refleksionih koeficijenata
sos2tf	Prebacivanje kanoničke realizacije u direktnu
sos2zp	Prebacivanje kanoničke realizacije u direktnu
tf2zp	Prebacivanje funkcije prenosa zadate preko nula i polova u direktnu realizaciju
tfe	Estimacija prenosne funkcije
zp2sos	Pronalazi kanoničku realizaciju za funkciju zadatu preko nula i polova
zp2tf	Pronalazi direktnu realizaciju za funkciju zadatu preko nula i polova