Kratke upute za korištenje MATLAB-a

O MATLAB-u

Program MATLAB služi za rješavanje različitih matematičkih problema, te čitav niz izračunavanja i simulacija vezanih uz obradu signala, upravljanje, regulaciju i identifikaciju sustava. Prva verzija MATLAB-a napisana je krajem 1970. godine na sveučilištima *University of New Mexico* i *Stanford University* s ciljem primjene u matričnoj teoriji, linearnoj algebri i numeričkoj analizi. Razvijene su poznate *fortranske* biblioteke funkcija LINPACK i EISPACK.

Danas svojstva MATLAB-a daleko prelaze originalni "matrični laboratorij". Radi se o interaktivnom sustavu i programskom jeziku za opća tehnička i znanstvena izračunavanja. Uz osnovni paket postoje i brojni programski paketi koji pokrivaju gotovo sva područja inženjerske djelatnosti: obradu signala, slike, 2D i 3D grafičke prikaze, automatsko upravljanje, identifikaciju sustava, statističke obrade, analizu u vremenskoj i frekvencijskoj domeni, simboličku matematiku i brojne druge. Paket SIMULINK je dodatak MATLAB-u koji omogućuje simulaciju kontinuiranih i diskretnih sustava pomoću funkcijskih blok dijagrama i dijagrama stanja. MATLAB je otvoren sustav u kojem korisnik može graditi svoje vlastite alate i biblioteke te modificirati postojeće, jer su dostupni u obliku izvornog koda.

Osnovne mogućnosti MATLAB-a

Svi podaci u MATLAB-u tretiraju se kao matrice čije dimenzije nije potrebno čuvati kao posebne varijable. Čak i skalarne veličine predstavljaju se kao matrice s dimenzijom 1×1. Svi su podaci interno zapisani u *double float* obliku (pomični zarez dvostruke preciznosti – 64 bita) što osigurava vrlo visok dinamički raspon i točnost za brojne primjene. Pored realnih brojeva i matrica, podržane su i kompleksne.

Po svojoj formi, MATLAB je interaktivni jezik – *interpreter*, namijenjen matričnim izračunavanjima. Po svojoj formi blizak je načinu na koji i inače zapisujemo matematičke formule, pa jedan redak u MATLAB-u može zamijeniti stotine redaka napisanih u nekom programskom jeziku opće namjene (C++, PASCAL, BASIC i sl.).

Nakon ulaska u program, kao i nakon svake izvedene naredbe pojavljuje se oznaka za unos oblika » iza koje se nalazi kursor. To označava da MATLAB očekuje unos nove naredbe. Svaka naredba mora završiti tipkom *Enter* – u nastavku teksta oznaka **<ENT>.**

Najjednostavniji primjeri su obična skalarna izračunavanja:

```
» 2+3 <ENT>

ans =
```

```
     5

» 7-4*5.1 <ENT>

ans =
     -13.4000

»
```

Rezultat izračunavanja pojavljuje se u sljedećim recima (ans = -13.4000), kao i oznaka za unos sljedeće naredbe. MATLAB poštuje matematički redoslijed operacija: potenciranje (13^2) prije množenja (13*2) i dijeljenja (13*2), množenje i dijeljenje prije zbrajanja (13+2) i oduzimanja (13-2), a moguće je i korištenje zagrada:

```
» (5 + 2 * (3 - 7.26)) / 1e2 + 2^3 <ENT>

ans =
    7.9648

» 5 + 2 * 3 - 7.26 / 1e2 + 2^3 <ENT>

ans =
   18.9274
```

Eksponencijalni zapis 1e2 znači 1*10^2, što iznosi 100. Za unošenje matrica koriste se uglate zagrade i točka-zarez:

```
» A=[1 2 3; 4 5 6] <ENT>

A =
     1     2     3
     4     5     6

» B = [1, 2; 3, 4; 5, 6] <ENT>

B =
     1     2
     3     4
     5     6
```

Točka-zarez (;) razdvaja retke matrice, a razmak ili zarez elemente istog retka. Alternativni način unosa istih matrica je:

```
» A=[1 2 3 <ENT>
4 5 6]; <ENT>

» B = [1, 2 <ENT>
3, 4 <ENT>
5, 6] ; <ENT>
```

Točka-zarez (;) na kraju naredbe označava da ne želimo ispis rezultata. Obje varijante naredbe definiraju nove *varijable* u radnom prostoru – realne matrice A i B s nekoliko redaka i stupaca, te željenih vrijednosti elemenata. Napomenimo da MATLAB *razlikuje* velika i mala slova, pa su varijable A i a, B i b različite. Imena varijabli moraju počinjati slovom, a smiju sadržavati najviše 19 alfanumeričkih znakova uključivši i donju crticu (_). Popis svih varijabli koje trenutno postoje u radnom prostoru dobiva se naredbom whos:

```
» whos <ENT>
  Name        Size              Bytes  Class

  A           2x3                  48  double array
  B           3x2                  48  double array
  ans         1x1                   8  double array

Grand total is 13 elements using 104 bytes
```

Ispis sadržaja postojeće varijable postiže se unosom njenog imena (bez ;):

```
» A <ENT>

A =
     1     2     3
     4     5     6
```

Brisanje varijabli iz radnog prostora izvršava se naredbom clear:

```
» clear B <ENT>                    % briše varijablu B
» clear <ENT>                      % briše sve varijable
```

Jednoliko rastući niz brojeva moguće je zadati odjednom pomoću operatora dvotočke:

```
» RastuciNiz = [1 : 0.1 : 1.6] <ENT>

RastuciNiz =

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000
```

Prvi broj u uglatoj zagradi (1) je početna vrijednost, drugi je korak (0.1), a treći završna vrijednost (1.6). Operator dvotočka zadaje raspon vrijednosti, od početne do završne. Negativna vrijednost koraka rezultirala bi padajućim nizom brojeva. Ako korak nije naveden, podrazumijeva se vrijednost 1:

```
» t = [1:10] <ENT>

t =

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

Pristup pojedinim elementima matrica moguć je korištenjem okruglih zagrada:

```
» A = [1 2 3; 4 5 6] <ENT>

A =

     1     2     3
     4     5     6

» A(2,3) <ENT>          % ispis elementa matrice A u drugom retku i trećem stupcu

ans =

     6
```

Kod pridruživanja se može pridružiti element matrice elementu matrice ili čak pojedini stupci ili redci pojedinim stupcima ili redcima. Jedini uvjet jest jednaka dimenzija elemenata s obje strane znaka pridruživanja - ne možemo pridružiti matricu dimenzija 3×3 matrici dimenzija 3×4. Na primjer:

```
» C = A(2,3) <ENT>      % spremanje elementa (2,3) matrice A u varijablu C

C =

     6

» C = A(:,3) <ENT>      % spremanje trećeg stupca matrice A u varijablu C

C =

     3
     6

» C = A(2,:) <ENT>      % spremanje drugog retka matrice A u varijablu C

C =

     4     5     6
```

Operator dvotočka bez zadanog raspona daje *sve* elemente neko stupca ili retka. No raspon može biti i zadan tako da na jednostavan način možemo odabrati neku podmatricu matrice. Možemo na primjer spremiti elemente matrice A koji su u drugom retku od drugog do trećeg stupca u matricu D na slijedeći način:

```
» D = A(2,2:3) <ENT>

D =

     5     6
```

Pri unosu kompleksnih brojeva koriste se posebne varijable `i` ili `j`, koje imaju vrijednost kompleksne jedinice, odnosno korijena iz -1. Tako bi kompleksni broj zadali na sljedeći način:

```
» z = 3 + 4 * i <ENT>                % realni dio je 3, a imaginarni je 4

z =

   3.0000 + 4.0000i

» z = 3 + 4 * j <ENT>

z =

   3.0000 + 4.0000i

» z = 3 + 4 * sqrt(-1) <ENT>

z =

   3.0000 + 4.0000i
```

Oprez: u MATLAB-u je moguće promijeniti vrijednost svih varijabli, pa tako i posebnih. Stoga je bolje koristiti definicijski izraz `sqrt(-1)`, pogotovo u vlastitim skriptama i funkcijama. Postoje i druge posebne varijable, kao što su `pi` (po definiciji `4*atan(1)`), zatim `ans` za međurezultate, `inf` za beskonačno veliku vrijednost itd.

Unos je moguć i u polarnom obliku, pomoću modula i faze:

```
» z=5*exp(i*0.927295218) <ENT>       % kut je zadan u radijanima

z =

   3.0000 + 4.0000i
```

Kompleksna se matrica može unijeti na različite načine:

```
» E=[1 2; 3 4] + i*[5 6; 7 8];
» E=[1+5*i 2+6*i; 3+7*i 4+8*i]

E =

   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 7.0000i   4.0000 + 8.0000i
```

`sqrt()` i `exp()` su MATLAB funkcije za računanje kvadratnog korijena i eksponenciranje. Argument funkcija se prosljeđuje unutar okruglih zagrada. Za pristup realnom, odnosno imaginarnom dijelu kompleksnih matrica koriste se

funkcije `real()` i `imag()`, dok funkcije `abs()` i `angle()` daju komponente polarnog modela kompleksnog broja: apsolutnu vrijednost ili modul, te fazu u radijanima.

Ugrađena pomoć i neke napredne funkcije ljuske

Za svaki operator ili funkciju, kao i za čitave programske pakete u MATLAB-u postoje detaljne upute *on line*. Unutar MATLAB ljuske do njih se dolazi korištenjem naredbe `help`.

```
» help                  % daje popis svih programskih paketa
» help ops              % daje popis svih operatora
» help mldivide         % daje upute za matrično lijevo dijeljenje
» help mrdivide         % daje upute za matrično desno dijeljenje
» help inv              % daje upute za funkciju inv (inverzija kvadratne matrice)
» help ime_naredbe      % daje upute za navedenu naredbu i sl.
```

Ako nas zanima funkcija `cos` pomoć možemo dobiti naredbom help:

```
» help cos <ENT>

 COS    Cosine.
    COS(X) is the cosine of the elements of X.

 Overloaded methods
    help sym/cos.m
```

Osim ove pomoći MATLAB ima i grafičko sučelje za pomoć, takozvani *HelpDesk* koji se pokreće zadavanjem naredbe `helpdesk` ili izborom `Help→MATLAB Help` unutar glavnog izbornika. MATLAB *HelpDesk* omogućuje napredna pretraživanja, a osim jednostavne pomoći sadrži i kompliciranije primjere korištenja pojedinih funkcija. Osim MATLAB *HelpDeska* svu MATLAB dokumentacija u elektroničkom obliku (HTML) možete pronaći na zavodskom serveru http://matlab.zesoi.fer.hr/.

Ukoliko ne znamo točno ime naredbe, vrlo je korisna naredba `lookfor`, koja daje popis naredbi "kandidata" za traženi pojam:

```
» lookfor image <ENT>
CONTRAST Gray scale color map to enhance image contrast.
FRAME2IM Convert movie frame to indexed image.
IM2FRAME Convert indexed image into movie format.
IM2JAVA Convert image to Java image.
IMAGE  Display image.
...
```

MATLAB ljuska također podržava automatsko nadopunjavanje ako se pritisne tipka **<TAB>**. Napišemo li par početnih slova naredbe te zatim pritisnemo **<TAB>** MATLAB će napisati ostatak naredbe samo ako postoji samo jedna naredba koja tako započinje. U slučaju da postoji više naredbi MATLAB oglašava zvonce. U

tom slučaju ponovnim pritiskom na **<TAB>** dobivamo popis mogućih naredbi. Ako znamo da neka naredba započinje s npr. `imag`, možemo napisati `imag` te upotrijebiti **<TAB>**. Tada MATLAB ispisuje sve valjane naredbe koje započinju s `imag`:

```
» imag <TAB><TAB>
imag        imagedemo   imagem      imageview
image       imageext    imagesc
» imag
```

Osim osnovnog nadopunjavanja MATLAB ljuska pamti i određeni broj prethodnih naredbi. Kroz prethodne naredbe se prolazi pritiskom na **<↑>**. No ako znamo da naredba koju smo već prije izveli započinje s npr. [Y,FS, NBITS] = možemo napisati početak naredbe te tek sada koristiti **<↑>**. U tom slučaju MATLAB prolazi samo kroz one naredbe koje započinju s već napisanim znakovima:

```
» [Y,FS,N <↑>
» [Y,FS,NBITS] = wavread('z:\spus\glasovi\u.wav') <↑>
» [Y,FS,NBITS] = wavread('z:\spus\glasovi\a.wav')
```

Osnovne operacije

Osnovne matematičke operacije

Na matrice je moguće primijeniti osnovne aritmetičke operacije +, -, * i /. MATLAB sve ove operacije primjenjuje na matrice. Kod množenja matrica broj stupaca lijeve matrice mora biti jednak broju redaka desne:

```
» A = [1 2 3; 4 5 6]; <ENT>          % dva retka i tri stupca
» B = [1 2; 3 4; 5 6]; <ENT>         % tri retka i dva stupca
» C = A * B <ENT>

C =                                  % rezultat ima dva retka i dva stupca
    22    28
    49    64
```

Kod zbrajanja i oduzimanja dimenzije matrice moraju biti istih dimenzija:

```
» D = [4 5 6; 1 2 3]; <ENT>          % dva retka i tri stupca
» S = A + D <ENT>

S =
     5     7     9
     5     7     9
```

MATLAB omogućuje i skalarne operacije na matricama, koje se izvršavaju na svakom članu matrice. Primjer množenja skalarom:

```
» 2 * A <ENT>

ans =
     2     4     6
     8    10    12
```

Primjer zbrajanja sa skalarnom veličinom (kod zbrajanja sa skalarom skalar se dodaje svakom elementu matrice):

```
» S = A + 2 <ENT>

S =
     3     4     5
     6     7     8
```

Kod dijeljenja matrica postoje dvije mogućnosti: lijevo i desno dijeljenje. Ako vrijedi A * X = B, tada je moguće pronaći X pomoću:

```
» X = A\B; <ENT>          % matričnog lijevog dijeljenja, koje odgovara izrazu
» X = inv(A) * B;
```

S druge strane, ako vrijedi X * A = B, tada se X nalazi desnim matričnim dijeljenjem:

```
» X = B/A; <ENT>          % što odgovara izrazu
» X = B * inv(A);
```

Naravno, u oba slučaja matrica A ne smije biti singularna, tj. det(A) ne smije biti nula.

Matrica se može potencirati cijelim brojem što odgovara uzastopnom množenju matrice same sa sobom. Tada matrica mora biti kvadratna:

```
» X = A^4; <ENT>          % je isto što i
» X = A*A*A*A;
```

Transpozicija matrice vrši se operatorom .':

```
» A = B.' <ENT>           % pridružuje A vrijednost transponirane matrice B
```

Operator ' je Hermitsko konjugiranje matrice. Hermitsko konjugiranje odgovara transpoziciji za realne matrice, a ako je matrica kompleksna tada se osim zamjene redaka i stupaca matrica konjugira, pa tako npr. element a+i*b u drugom retku i trećem stupcu matrice B ide u treći redak i drugi stupac matrice A kao a-i*b.

```
» A = B' <ENT>            % pridružuje A vrijednost hermitski konjugirane matrice B
```

Operacije po elementima

Matematičke operacije moguće je obaviti i između pojedinačnih elementa matrica. Tada prije matematičkog željenog operatora stavljamo točku . (.*, ./, .^, itd.):

```
» x = [1 2 3]; <ENT>
» y = [4 5 6]; <ENT>
» z = x .* y <ENT>

z =

     4    10    18          % pojedinačno množenje elemenata: 1*4 2*5 3*6
```

Ovakvo množenje je različito od matričnog množenja koje rezultira unutarnjim ili vanjskim produktom vektora:

```
» z = x * y' <ENT>          % skalarni ili unutarnji produkt vektora: 1*4+2*5+3*6

z =
    32

» z = x' * y <ENT>          % vanjski produkt vektora: svaki element sa svakim

z =
     4     5     6
     8    10    12
    12    15    18
```

Primjeri operacija po elementima:

```
» w = x ./ y <ENT>

w =
    0.2500 0.4000 0.5000   % daje kvocijente elemenata 1/4 2/5 3/6

» tt = x .^ y <ENT>

tt =
     1 32 729              % daje potencije elemenata 1^4 2^5 3^6.

» tt = x .^ 2 <ENT>

tt =
     1 4 9                 % daje potenciranje skalarom 1^2 2^2 3^2
```

Relacijski operatori

MATLAB podržava sljedeće relacijske operatore:

```
» a < b          % manje
» a <= b         % manje ili jednako
» a > b          % veće
» a >= b         % veće ili jednako
```

```
» a == b          % jednako
» a ~= b          % nije jednako
```

Rezultat relacijskog operatora je 0 ako uvjet nije zadovoljen ili 1 ako jest.

```
» b = 123 > 11 <ENT>

b=
    1             % jer 123 je veće od 11

» b ~= b <ENT>

ans=
    0             % jer je varijabla b uvijek jednaka sama sebi
```

Relacijski operatori se mogu povezivati pomoću logičkih operatora:

```
» a & b           % AND, operacija logičko "i"
» a | b           % OR, operacija logičko "ili"
» a ~ b           % NOT, operacija logičko "ne"

» (a >= 13) | (b < 5)        % daje vrijednost 1 ako je a veće ili jednako 13
                              % ili ako je b manje od 5
```

Ako je operand u relacijskom izrazu matrica rezultat je opet matrica, gdje se relacijski operatori primjenjuju se na svaki element matrice posebno:

```
» B = [1 -2 5; 3 7 4]; <ENT>
» C = (B > 0) | (B < -3) <ENT>

C =

     1     0     1          % jedino -2 nije veći od 0 i manji od -3
     1     1     1
```

Posebne matrice

Funkcija `ones(m,n)` vraća matricu sa `m` redaka i `n` stupaca popunjenu jedinicama, dok funkcija `zeros(m,n)` vraća matricu istih dimenzija popunjenu nulama. Iz navedenog primjera vidimo da ako MATLAB funkcija ima više argumenata, oni se međusobno odvajaju zarezom:

```
» A = ones(3) <ENT>

A =

     1     1     1
     1     1     1
     1     1     1

» A = ones(2,3) <ENT>
```

```
A =

     1     1     1
     1     1     1

» A = 1.2*ones(1, 3) <ENT>          % daje redak od 3 elementa vrijednosti 1.2

A =

    1.2000    1.2000    1.2000
```

Matrica `eye(n)` je kvadratna matrica dimenzija n×n s jedinicama na dijagonali (tzv. jedinična matrica).

```
» eye(3) <ENT>

ans =

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

Operacije nad matricama

Osim osnovnih matematičkih operacija na matrice je moguće primjenjivati i različite funkcije. Neke od njih operiraju nad pojedinim elementima matrice, neke nad stupcima, a neke nad cijelim matricama.

Elementarne funkcije primjenjuju se na svaki element matrice, npr:

```
» A = [1 2 3; 4 5 6]; <ENT>
» cos(A) <ENT>

ans =

    0.5403   -0.4161   -0.9900
   -0.6536    0.2837    0.9602
```

Naredba `help elfun` daje popis elementarnih funkcija dostupnih u MATLAB-u, kao što su trigonometrijske funkcije `sin`, `cos`, `tan`, `atan`, `atan2`… pa hiperbolne funkcije `sinh`, `cosh`, `tanh`, `atanh`… pa eksponencijalne i logaritamske funkcije `exp`, `log`, `log2`, `log10`… itd.

Osim elementarnih funkcija MATLAB poznaje i njihove matrične ekvivalente, npr. `expm`, `logm`, `sqrtm`… koje realiziraju matričnu eksponencijalu, matrični logaritam, matrični kvadratni korijen:

```
» A=[1 2; 3 4]; <ENT>
» exp(A) <ENT>

ans =                % obična eksponencijala se primjenjuje na svaki element matrice
```

```
    2.7183    7.3891
   20.0855   54.5982

» expm(A) <ENT>

ans =              % dok se matrična eksponencijala primjenjuje na matricu

   51.9690   74.7366
  112.1048  164.0738
```

Neke od preostalih funkcije operiraju nad stupcima matrice. Stupci matrice se tada tretiraju kao neovisni vektori na koje se primjenjuje zadana operacija. Npr. `max(x)` daje vrijednosti najvećih elemenata svakog stupca od `x`, `min(x)` daje vrijednosti najmanjih elemenata svakog stupca od `x`, `sum(x)` daje zbroj svih elemenata stupaca, `prod(x)` daje produkt svih elemenata stupaca itd.

```
» A = [1 2 3; 4 5 6]; <ENT>
» max(A) <ENT>

ans =

     4     5     6                    % najveći elementi po stupcima
```

No kako pronaći najveći element u matrici ako ona ima više stupaca? Jedna od mogućnosti je uzastopna primjena naredbe `max`, no moguće je odrediti maksimalni element i korištenjem naredbe `reshape`:

```
» max(max(A)) <ENT>

ans =

     6

» max(reshape(A,1,prod(size(A)))) <ENT>

ans =

     6
```

Naime, naredba `max` i sve slične naredbe operiraju na stupcima osim ako se radi o vektoru. Naredbe koje operiraju nad stupcima ne razlikuju vektor-retke ili vektor-stupce tako da uzastopnom primjenom naredbe `max` možemo odrediti maksimum.

MATLAB sve matrice pamti na jednak način kao niz elemenata. Sama dimenzija matrice je posebna informacija koju možemo promijeniti naredbom `reshape`. U gornjem primjeru kada smo tražili maksimalni element matrice naredbom `reshape` smo matricu `A` veličine 2×3 pretvorili u novu matricu veličine $1\times2\cdot3$.

Prikaz rezultata

Osnovna MATLAB ljuska nije pogodna za grafički prikaz rezultata. Da bi mogli grafički prikazati rezultate moramo otvoriti novi prozor za njihov prikaz što se postiže naredbom `figure`.

```
» figure <ENT>              % otvara novi grafički prozor
```

Možemo imati više različitih grafičkih prozora. MATLAB-ove naredbe za crtanje kao što je `plot` crtaju u trenutno aktivni prozor te je prije crtanja potrebno odabrati neki prozor. Svaki grafički prozor ima svoj broj koji se koristi kod odabira aktivnog prozora pomoću naredbe `figure`.

```
» figure(2) <ENT>           % čini prozor s brojem 2 aktivnim ili otvara novi prozor
                            % ako takav ne postoji
» x = [0:0.1:10]; <ENT>
» plot(x); <ENT>            % crtamo vektor x u prozor 2
```

Ako koristimo naredbu `plot` i ako ne postoji aktivni grafički prozor, MATLAB će otvoriti jedan.

Naredba `plot(x)` crta vektor `x` na taj način da se na *x*-osi nalaze indeksi vektora `x`, a na *y*-osi nalaze vrijednosti vektora. Na samom prikazu naredba `plot` točke (*indeks*, *vrijednost vektora za indeks*) spaja pravcima. Ako ne želimo spajati točke pravcima možemo koristiti sljedeće:

```
» plot(x,'o'); <ENT>        % crta pojedine točke pomoću kružića ali ih ne spaja
» stem(x); <ENT>            % za stupčasti prikaz diskretnih signala
» stairs(x); <ENT>          % za stepeničasti prikaz
```

Osim prikaza jednog vektora u zavisnosti o indeksu, možemo nacrtati i već prethodno zadane parove točaka (*x*, *y*). Pri tome se funkciji `plot` prosljeđuju dva vektora od kojih prvi sadrži *x*-koordinate dok drugi sadrži *y*-koordinate. Pri tome oba vektora moraju imati jednak broj elemenata. Ovakav prikaz se i najčešće koristi.

```
» t = [1:0.01:10]; <ENT>    % prvo zadajemo vektor vremena t
» x = sin(2*pi*t); <ENT>    % sada definiramo sinusni signal
» plot(t,x) <ENT>           % kojeg na kraju nacrtamo u t-x koordinatnom sustavu
```

MATLAB funkcija plot može se koristiti i za crtanje matrica. Pri tome isto kao i za naredbu `max` matrica se promatra po stupcima, tj. pretpostavlja se da svaki stupac predstavlja pojedinačni signal.

```
» y = [1 2 3 <ENT>          % stupci matrice y tretiraju se kao nezavisni signala
4 5 6 <ENT>
7 8 9 <ENT>
```

```
10 11 12]; <ENT>
» plot(y) <ENT>              % naredba plot(y) crta vektore [1 4 7 10].', [2 5 8 11].'
                            % i [3 6 9 12].' u tri različite boje, dok se na x-osi
                            % nalaze indeksi elemenata po retcima [1 2 3 4]
```

Možemo istodobno nacrtati i više različitih parova signala. Općenito naredba plot očekuje ulazne podatke koji su redom *x*-koordinate, *y*-koordinate, način crtanja pa opet *x*-koordinate, *y*-koordinate, način crtanja itd. Pri tome neki od elemenata ponekad mogu biti izostavljeni.

```
» x1 = [1.1 2.2 3.3]; <ENT>          % x koordinate prvog grafa
» y1 = [4.4 5.5 6.6]; <ENT>          % y koordinate prvog grafa
» x2 = [1.5 2.7 3.1 4.3]; <ENT>      % x koordinate drugog grafa
» y2 = [4.0 5.1 5.9 7.2]; <ENT>      % y koordinate drugog grafa
» plot(x1,y1,x2,y2) <ENT>            % crta graf zadan parovima (x1, y1) u jednoj
                                     % i graf zadan parovima (x2, y2) u drugoj boji
» plot(x1,y1,'r-',x2,y2) <ENT>       % crta graf zadan parovima (x1, y1) u crvenoj
                                     % boji punomlinijom te graf zadan parovima
                                     % (x2, y2) u drugoj boji
» plot(x1,y1,':',x2,y2,'g') <ENT>    % prvi graf je sada crtan točkastim linijama,
                                     % dok je drugi u zelenoj boji
```

Pri ovakvom crtanju se različite krivulje mogu razlikovati u broju točaka, dakle krivulja određena parovima (x_1, y_1) ne mora imati jednak broj točaka kao krivulja određena parovima (x_2, y_2).

Svaka naredba `plot` i općenito svako crtanje u grafički prozor briše prethodni sadržaj prozora. Ukoliko želimo crtati preko već nacrtanog moramo prije naredbe `plot` zadati naredbu `hold on`. Ova mogućnost se isključuje zadavanjem naredbe `hold off`.

```
» t = [1:0.01:10]; <ENT>
» x = sin(2*pi*t); <ENT>
» plot(t,x) <ENT>
» hold on <ENT>              % uključujemo crtanje preko postojećeg
» plot(t, t) <ENT>
» hold off <ENT>             % isključujemo crtanje preko postojećeg
```

Svaki prozor za crtanje se može zatvoriti naredbom `close`.

```
» close(1) <ENT>             % zatvara prozor s brojem 1
» close all <ENT>            % zatvara sve prozore
```

Za ostale detalje o naredbi `plot` pogledajte pomoć za naredbu `plot` (`help plot` ili detaljniju pomoć u HelpDesku). Osim naredbe `plot` ostale zanimljivije funkcije za crtanje su `semilogx`, `semilogy`, `loglog`, `grid`, `clf`, `clc`, `title`, `xlabel`, `ylabel`, `axis`, `axes`, `hold`, `subplot`, a za trodimenzionalne prikaze `graph3d`.

Grafika u MATLAB-u je objekta, te se sva svojstva prikaza i sam prikaz mogu mijenjati s naredbama `get` i `set`. Naredba `get` dohvaća sva svojstva nekog objekta, dok ih naredba `set` mijenja:

```
» h = figure; <ENT>                    % stvaramo novi objekt h, h je broj slike
» get(h) <ENT>                         % MATLAB ispisuje sva svojstva slike
      BackingStore = on
      CloseRequestFcn = closereq
      Color = [0.8 0.8 0.8]
      Colormap = [ (64 by 3) double array]
      CurrentAxes = []
      CurrentCharacter =
      ...

» set(h, 'Name', 'Slicica') <ENT>  % mijenjamo ime prozora u 'Slicica'
```

Spremanje i učitavanje podataka

MATLAB omogućava spremanje varijabli u datoteke. Prva i najjednostavnija mogućnost jest korištenje glavnog izbornika u kojem se odabere `File→SAVE Workspace As...` stavka. Ovime spremamo sve varijable u neku datoteku. Učitavanje se vrši na jednak način odabirom `File→Open`.

No MATLAB podržava naredbe za spremanje i učitavanje točno određenih varijabli iz datoteka:

```
» whos <ENT>                    % pogledajmo prvo koje varijable imamo
  Name      Size          Bytes  Class

  A         3x3              72  double array
  B         3x3              72  double array
  a         1x1               8  double array

Grand total is 19 elements using 152 bytes

» save <ENT>                    % spremamo sve varijable u datoteku matlab.mat

Saving to: matlab.mat

» load <ENT>                    % čitavamo sve varijable iz datoteke matlab.mat

Loading from: matlab.mat

» save pero A B <ENT>           % spremamo varijable A i B u datoteku pero.mat
» whos -file pero <ENT>         % koje varijable postoje u datoteci pero.mat
  Name      Size          Bytes  Class

  A         3x3              72  double array
  B         3x3              72  double array

Grand total is 18 elements using 144 bytes

» load pero A <ENT>             % učitavamo samo varijablu A iz datoteke pero.mat
» save pero a -append <ENT>     % dodajemo još varijablu a u datoteku pero.mat
```

```
» clear all <ENT>                % brišemo sve varijable u MATLAB-u
» close all <ENT>                % zatvaramo sve prozore u MATLAB-u
» pack <ENT>                     % oslobađamo privremeno zauzetu memoriju
» load pero <ENT>                % učitavamo sve iz datoteke pero.mat i
                                 % započinjemo rad iznova na mjestu gdje smo stali
```

Nekad osim samog spremanja varijabli želimo imati bilješke svega što smo napravili tijekom interaktivnog rada s MATLAB-om. U tu svrhu se koristi naredba `diary`.

```
» diary moj_rad <ENT>            % sve što MATLAB prikazuje od sada se sprema
                                 % u tekstualnu datoteku moj_rad koja predstavlja
                                 % dnevnik rada
» whos <ENT>
  Name        Size            Bytes  Class

  A           3x3                72  double array
  B           3x3                72  double array
  a           1x1                 8  double array

Grand total is 19 elements using 152 bytes

» diary off <ENT>                % isključuje spremanje u dnevnik rada
```

Simbolička matematika

Jedan od mnogih programskih paketa/modula unutar MATLAB-a podržava simboličku matematiku. Njegove mogućnosti možete pogledati naredbom `help symbolic`, dok naredba `symintro` daje kratki uvod s primjerima.

Osim numeričkih varijabli koje smo već upoznali MATLAB sa simboličkim paketom podržava simboličke varijable:

```
» lambda = sym('lambda') <ENT>   % definiramo novu simboličku varijablu

lambda =

lambda

» whos
  Name          Size            Bytes  Class

  A             3x3                72  double array
  a             1x1                 8  double array
  lambda        1x1               136  sym object

Grand total is 17 elements using 216 bytes
```

Svaki simbolički objekt može imati i dodatna svojstva koja se određuju kod deklaracije varijable:

```
» x = sym('x'); <ENT>              % simbolička varijabla x bez dodatnih svojstava
» x = sym('x','real'); <ENT>       % realna simbolička varijabla x
```

Ako nam dodatna svojstva nisu bitna naredbom `syms` možemo odmah definirati više simboličkih varijabli:

```
» syms x y z <ENT>                % definiramo tri simboličke varijable
```

MATLAB naredbe na simboličkim varijablama izvršavaju se simbolički:

```
» a = [-3+4*i 4*i -3-3*i <ENT>
-3-i -6-i 3+3*i <ENT>
4*i 4*i -6-3*i]; <ENT>
» pomocna = lambda * eye(3) - a <ENT>

pomocna =

[ lambda+3-4*i,          -4*i,         3+3*i]
[          3+i,   lambda+6+i,        -3-3*i]
[         -4*i,          -4*i, lambda+6+3*i]

» KarakPolinom = det(pomocna) <ENT>

KarakPolinom =

lambda^3+15*lambda^2+81*lambda+135

» KarakVrijednosti = solve(KarakPolinom) <ENT>

KarakVrijednosti =

[     -3]
[ -6+3*i]
[ -6-3*i]
```

Korištenjem simboličkih izraza mogu se definirati i simboličke funkcije koje se potom mogu derivirati, integrirati i sl. Primjer:

```
» syms a x <ENT>                  % simboličke varijable a i x
» f = sin(a * x) <ENT>            % definiramo simboličku funkciju f(x)

f =

sin(a*x)

» diff(f) <ENT>                   % derivacija funkcije po x

ans =

cos(a*x)*a

» int(f) <ENT>                    % neodređeni integral funkcije po x
```

```
ans =

-cos(a*x)/a

» syms Donja Gornja <ENT>
» int(f, Donja, Gornja) <ENT>     % određeni integral funkcije po x

ans =

-cos(Gornja*a)/a+cos(Donja*a)/a
```

Simboličke funkcije se mogu raspisati u Taylorov red:

```
» taylor(f,7) <ENT>               % prvih 7 članova Taylorovog reda funkcije f(x)

ans =

a*x-1/6*a^3*x^3+1/120*a^5*x^5
```

Može se izračunati Laplaceova transformacija:

```
» Laplace(f) <ENT>                % Laplaceova transformacija funkcije f(x)

ans =

a/(s^2+a^2)
```

Simboličke funkcije mogu se nacrtati korištenjem naredbe `ezplot`. Naredba `solve` služi za rješavanje sustava jednadži dok `dsolve` služi za simboličko rješavanje sustava diferencijalnih jednadžbi itd.

Programiranje u MATLAB-u

Jednostavni MATLAB program

MATLAB je potpun programski jezik u kojem je moguće napisati vlastite programske odsječke. Pojedine naredbe moguće je izvršiti uvjetno ili ponoviti više puta.

Pomoću ugrađenih funkcija i programskih paketa moguće je graditi nove programe. Svaki skup MATLAB naredbi napisan korištenjem bilo kojeg tekst editora koji je pohranjen u datoteci s nastavkom `.m` predstavlja jedan MATLAB program. Dakle, svi MATLAB programi se spremaju u obične tekstualne datoteke a najjednostavnije ih je pisati korištenjem ugrađenog editora koji se poziva naredbom `edit`.

```
» edit <ENT>                      % poziva ugrađeni MATLAB editor
```

Pokažimo najprije jedan jednostavni MATLAB program koji samo ispisuje jednu poruku na ekranu:

```
1.   function hello
2.   % HELLO - Jednostavan MATLAB program.
3.
4.   % Komentar započinje s postotkom.
5.   disp('Hello MATLAB!');
```

Neka je program spremljen u datoteku `hello.m`. Unutar MATLAB prozora ga pozivamo jednostavnim zadavanjem naredbe `hello`.

```
» hello <ENT>            % pozivamo naš program hello
Hello MATLAB!
» help hello <ENT>       % naredba help hello ispisuje prvi komentar ispod ključne
                         % riječi function sve do prvog praznog reda

  HELLO - Jednostavan MATLAB program.

» what <ENT>             % what ispisuje sve programe unutar trenutnog direktorija

M-files in the current directory e:\

hello
```

Prenošenje argumenata

Pokažimo sada kako izgleda prenošenje varijabli u funkciju te kako funkcija vraća rezultat na primjeru funkcije koja zbraja dva broja.

```
1.   function c = zbroji(a, b)
2.   % ZBROJI - Zbrajanje dva broja
3.
4.   % Provjera ulaznih argumenata.
5.   if nargin ~= 2
6.      error('Zbrajamo dva broja.');
7.   end
8.
9.   % Kod programa.
10.  c = a + b;
```

Funkciju opet pozivamo jednostavnim navođenjem imena `zbroji`, ali ovog puta moramo zadati i argumente. MATLAB strogo ne provjerava broj ulaznih argumenata, već pridružuje varijable redom kako su navedene. To nam omogućava veću fleksibilnost jer možemo napisati funkciju s više ulaznih argumenata koji poprimaju neke unaprijed zadane vrijednosti kada ih ne zadamo kod poziva funkcije. MATLAB će javiti poruku o pogrešci samo ako je zadano previše ulaznih argumenata, a ako je zadano manje ulaznih argumenata oni koji nisu zadani ostaju nedefinirani. Stoga je potrebno napraviti provjeru unutar tijela funkcije (redovi 4.-7.).

```
» zbroji(2,4) <ENT>              % zbrajamo dva broja
```

```
ans =

     6                                % MATLAB odmah ispisuje rezultat

» rezultat = zbroji(2, 4) <ENT>  % ovime rezultat spremamu u varijablu rezultat

rezultat =

     6

» zbroji <ENT>                   % pozovemo li funkciju bez argumenata MATLAB
??? Error using ==> zbroji       % ispisuje poruku o grešci
Zbrajamo dva broja.
```

Kada smo pozvali funkciju `zbroji` bez argumenata MATLAB je ispisao našu poruku o pogrešci (šesta linija koda). Od pete do sedme linije koda ispitujemo koliki je broj ulaznih argumenata pomoću funkcije `nargin`, te u slučaju da nemamo dva ulazna argumenta ispisujemo poruku o pogrešci. Češće umjesto poruke o pogrešci neprenesenim varijablama pridružimo neku predefiniranu vrijednost:

```
1.   % Provjera ulaznih argumenata.
2.   if nargin == 0
3.      a = 1;
4.      b = 1;
5.   elseif nargin == 1
6.      b = 1;
7.   elseif nargin ~= 2
8.      error('Zbrajamo dva broja.');
9.   end
```

Kontrola toka programa

MALTAB razumije osnovne programske petlje (`for`, `while`) te uvjetna i bezuvjetna grananja (if, switch, break) kojima se kontrolira tok programa.

`for` petlja se izvršava na taj način da brojač unutar petlje poprima sve vrijednosti stupaca unutar zadane matrice.

```
1.   for i = X                   % za svaki stupac matrice X
2.      disp(i);                 % izvrši tijelo petlje
3.   end
```

Naravno, ukoliko se radi o vektor-retku brojač poprima sve vrijednosti unutar tog retka:

```
1.   for i = [1 2 5 7]           % za svaki element vektora
2.      disp(i);                 % izvrši tijelo petlje
3.   end
```

Obično se koristi operator : koji omogućuje jednostavno postavljanje početne i konačne vrijednosti te koraka:

```
a = zeros(1,10);              % inicijaliziraj vektor a
for i = 1:0.5:10              % za svaki i od 1 do 10 s korakom 0.5
  a(i) = i*i -3;              % na i-to mjesto vektora a ubaci vrijednost i*i-3
end                           % kraj for petlje
```

`while` petlja se izvršava dok je ispunjen logički uvjet petlje:

```
i = 7;                        % inicijaliziraj varijablu i
while (i >= 0)                % dok je i veći ili jednak 0 ponavljaj
   if (a(i) ~= 5)             %       ako a(i) nije jednako 5
      a(i) = a(i) - 3;        %              tada a(i) smanji za tri
   else                       %       inače
      a(i) = 127;             %              u a(i) spremi 127
   end                        %       kraj if naredbe
   i = i - 1;                 %       smanji i za 1
end                           % kraj while petlje
```

Naredbu `break` koristimo za bezuvjetni izlazak samo iz `for` ili `while` petlje.

```
i = 7;
while (i >= 0)
   if (i > 10)                % ako je i veći od 10
      break                   %       bezuvjetno prekini petlju
   end                        % kraj if naredbe
end
```

`if` naredba omogućuje uvjetna izvršavanja koda, dok se za kompliciranija uvjetna izvršavanja obično koristi naredba `switch`.

```
if a == b
   c = a + b;
elseif abs(a) == b
   c = a - b;
else
   c = 0;
end
```

Kreiranje vlastitih MATLAB programa i skripti

Do sada je pokazano kako izgledaju MATLAB programi (odnosno funkcije). Osim programa MATLAB podržava i skripte. I programi i skripte spremaju se u obične tekstualne datoteke s nastavkom `.m`. Glavna razlika jest u tome da MATLAB programi odnosno funkcije započinju s ključnom riječi `function`, dok je sadržaj MATLAB skripte uistinu identičan nizu naredbi koje ručno unosimo u interaktivnom modu rada.

Važna razlika između programa i skripti je i *doseg varijabli*. Sve varijable unutar MATLAB funkcije postoje samo za vrijeme izvođenja funkcije (osim ako

se ne radi o globalnoj varijabli), dok sve varijable deklarirane unutar MATLAB skripte postoje i dostupne su unutar interaktivnog korisničkog sučelja.

Jednostavna MATLAB skripta:

```
% primjer MATLAB skripte koja crta Bessleove funkcije

n=[0 1 2 3 4 5 6 7 8 9];
m=[0:0.1:12];
figure, plot(m,besselj(n',m)), grid;
set(gca,'FontName','Times');
xlabel('\sl m'), ylabel('\sl J_n(m)');
title('Besselove funkcije prve vrste,'...
'reda \sl n \rm varijable \sl m');
print -deps besselm.eps;

n=[0:0.1:12];
m=[1 2 3 4 5 6 7];
figure, plot(n,besselj(n',m)), grid;
set(gca,'FontName','Times');
xlabel('\sl n'), ylabel('\sl J_n\rm(\sl m\rm)');
title('Besselove funkcije prve vrste,'...
'reda \sl n \rm varijable \sl m');
print -deps besseln.eps;
```

Kostur složene MATLAB funkcije:

```
function [izlaz1, izlaz2, ...] = ime(ulaz1, ulaz2, ...)
% IME Funkcija IME radi nešto korisno.
%     Ulazne varijable su ulaz1, ulaz2,...
%     Izlazne varijable su izlaz1, izlaz2,...

% autor, datum

% Deklaracija globalnih varijabli
global globalna_varijabla_1

% Deklaracija i inicijalizacija lokalnih varijabli
lokalna_varijabla_1 = 1;
lokalna_varijabla_2 = 2;

% Provjera ulaznih argumenata
if nargin ~= 3
   warning('Neispravan broj ulaznih argumenata.');
end

% Tijelo funkcije
izlaz1 = TeskaMatematika(ulaz1, ulaz2, ...);
izlaz2 = JosTezaMatematika(ulaz1, ulaz2, lokalna_varijabla);

% Provjera izlaznih varijabli
if nargout ~= 3
   warning('Neispravan broj izlaznih argumenata.');
end
```

Ime funkcije mora se podudarati s imenom datoteke (bez nastavka `.m`). Lokalne varijable u funkciji nevidljive su pozivatelju i obrnuto. Prenose se samo

ulazni i izlazni argumenti. Funkcije `nargin` (korištena u 16. liniji koda) i `nargout` (korištena u 25. liniji koda) daju konkretan broj ulaznih, odnosno izlaznih argumenata pri pozivu neke funkcije.

Većina MATLAB funkcija realizirana je korištenjem osnovnih naredbi MATLAB-a, dok manji dio čine osnovne funkcije. Realizacija postojećih MATLAB funkcija može se vidjeti naredbom `type`:

```
» type tic <ENT>                          % ispisuje kod MATLAB funkcije ili skripte

function tic
%TIC Start a stopwatch timer.
%   The sequence of commands
%       TIC, operation, TOC
%   prints the time required for the operation.
%
%   See also TOC, CLOCK, ETIME, CPUTIME.

%   Copyright (c) 1984-97 by The MathWorks, Inc.
%   $Revision: 5.3 $  $Date: 1997/04/08 06:53:31 $

% TIC simply stores CLOCK in a global variable.
global TICTOC
TICTOC = clock;

» type max <ENT>                          % neke funkcije su ugrađene (odnosno nisu
max is a built-in function.               % realizirane u MATLAB-u)
```

Reference

This section contains detailed descriptions of all MATLAB functions. It begins with a list of functions grouped by subject area and continues with the reference entries in alphabetical order. Information is also available through the on-line Help facility.

General

help	help facility
demo	run demonstrations
who	list variables in memory
what	list M-files on disk
size	row and column dimensions
length	vector length
clear	clear workspace
computer	type of computer
^C	local abort
quit	terminate program
exit	same as quit

Matrix Operators & Array Operators

+	addition	+	addition
-	subtraction	-	subtraction
*	multiplication	.*	multiplication
/	right division	./	right division
\	left division	.\	left division
^	power	.^	power
'	conjugate transpose	.'	transpose

Relational and Logical Operators

<	less than	&	AND
<=	less than or equal	|	OR
>	greater than	~	NOT
>=	greater than or equal		
==	equal		
~=	not equal		

Special Characters

=	assignment statement
[	used to form vectors and matrices
]	see [
(	arithmetic expression precedence
)	see (
.	decimal point
...	continue statement to the next line
,	separate subscripts and function arguments
;	end rows, suppress printing
%	comments
:	subscripting, vector generation
!	execute operating system command

Special Values

ans	answer when expression is not assigned
eps	floating point precision
pi	number pi
i,j	square root of -1
Inf	Infinity
NaN	Not-a-Number
clock	wall clock
date	date
flops	floating point operation count
nargin	number of function input arguments
nargout	number of function output arguments

Text and Strings

abs	convert string to ASCII values
eval	evaluate text macro
num2str	convert number to string
int2str	convert integer to string
setstr	set flag indicating matrix is a string
sprintf	convert number to string
isstr	detect string variables
strcmp	compare string variables
hex2num	convert hexadecimal string to number

Graph Paper

plot	linear X-Y plot
loglog	loglog X-Y plot
semilogx	semi-log X-Y plot
semilogy	semi-log X-Y plot
polar	polar plot
mesh	3-dimensional mesh surface
contour	contour plot
meshdom	domain for mesh plots
bar	bar charts
stairs	stairstep graphs
errorbar	add errorbars

Graph Annotation

title	plot title
xlabel	x-axis label
ylabel	y-axis label
grid	draw grid lines
text	arbitrarily positioned text
gtext	mouse-positioned text
ginput	graphics input

Graph Window Control

axis	manual axis scaling
hold	hold plot on screen
shg	show graph screen
clg	clear graph screen
subplot	split graph window

Control Flow

if	conditionally execute statements
elseif	used with if
else	used with if
end	terminate if, for, while
for	repeat statements a number of times
while	do while
break	break out of for and while loops
return	return from functions
pause	pause until key press

Disk Files

chdir	change current directory
delete	delete file
diary	diary of the session
dir	directory of files on disk
load	load variables from file
save	save variables on file
type	list function or file
what	show M-files on disk
fprintf	write to a file
pack	compact memory via save

Relational and Logical Functions

any	logical conditions
all	logical conditions
find	find array indices of logical values
exist	check if variables exist
isnan	detect NaNs
finite	detect infinities
isempty	detect empty matrices
isstr	detect string variables
strcmp	compare string variables

Trigonometric Functions

sin	sine
cos	cosine
tan	tangent
asin	arcsine
acos	arccosine
atan	arctangent
atan2	four quadrant arctangent
sinh	hyperbolic sine
cosh	hynerbolic cosine
tanh	hyperbolic tangent
asinh	hyperbolic arcsine
acosh	hyperbolic arccosine
atanh	hyperbolic arctangent

Elementary Math Functions

abs	absolute value or complex magnitude
angle	phase angle
sqrt	square root
real	real part
imag	imaginary part
conj	complex-conjugate
round	round to nearest integer
fix	round towards zero
floor	round towards -Infinity
ceil	round towards +Infinity
sign	signum function
rem	remainder or modulus
exp	exponential base e
log	natural logarithm
log10	log base 10

Polynomials

poly	characteristic polynomial
roots	polynomial roots - companion matrix method
roots1	polynomial roots - Laguerre's method
polyval	polynomial evaluation
polyvalm	matrix polynomial evaluation
conv	multiplication
deconv	division
residue	partial-fraction expansion
polyfit	polynomial curve fitting

Matrix manipulation

rot90	rotation
fliplr	flip matrix left-to-right
flipud	flip matrix up-and-down
diag	extract or create diagonal
tril	lower triangular part
triu	upper triangular part
reshape	reshape
.'	transposition
:	general rearrangement

Decompositions and Factorizations

cdf2rdf	convert complex-diagonal to real-diagonal
eig	eigenvalues and eigenvectors
hess	Hessenberg form
inv	inverse
lu	factors from Gaussian elimination
nnls	nonnegative least-squares
null	null space
orth	orthogonalization
pinv	pseudoinverse
qr	orthogonal-triangular decomposition
qz	QZ algorithm
svd	singular value decomposition

Elementary Matrix Functions

expm	matrix exponential
logm	matrix logarithm
sqrtm	matrix square root
poly	characteristic polynomial
det	determinant
trace	trace

Diferential Equation Solution

ode23	2nd/3rd order Runge-Kutta method
ode45	4th/5th order Runge-Kutta-Fehlberg method

Numerical Integration

quad	numerical function integration
quad8	numerical function integration

Nonlinear Equations and Optimization

fmin	minimum of a function of one variable
fmins	minimum of a multivariable function (unconstrained nonlinear optimization)
fsolve	solution to a system of nonlinear equations (zeros of a multivariable function)
fzero	zero of a function of one variable

Columnwise Data Analysis

max	maximum value
min	minimum value
mean	mean value
median	median value
std	standard deviation
sort	sorting
sum	sum of elements
prod	product of elements
cumsum	cumulative sum of elements
cumprod	cumulative product of elements
diff	approximate derivatives
hist	histogram
corrcoef	correlation coefficients
cov	covariance matrix
cplxpair	reorder into complex pairs

Signal Processing

abs	complex magnitude
angle	phase angle
conv	convolution
corrcoef	correlation coefficients
cov	covariance
deconv	deconvolution
fft	fast Fourier transform
fft2	two-dimensional FFT
ifft	inverse fast Fourier transform
ifft2	inverse 2-D FFT
fftshift	swap quadrants of matrices