Zavod za elektroničke sustave i obradbu informacija
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu

Kratke upute za korištenje MATLAB-a

Tomislav Petković

Zagreb, travanj 2005.

Sadržaj

1. O MATLAB-u 2

2. Osnovne mogućnosti MATLAB-a 2

2.1. Ugrađena pomoć i neke napredne funkcije ljuske 7
2.2. Osnovne operacije 8
2.2.1. Osnovne matematičke operacije 8
2.2.2. Operacije po elementima 10
2.2.3. Relacijski operatori 10
2.3. Posebne matrice 11
2.4. Operacije nad matricama 12

3. Prikaz rezultata 14

4. Spremanje i učitavanje podataka 17

5. Simbolička matematika 18

6. Programiranje u MATLAB-u 20

6.1. Jednostavni MATLAB program 20
6.2. Prenošenje argumenata 21
6.3. Kontrola toka programa 22
6.4. Kreiranje vlastitih MATLAB programa i skripti 24
6.5. Napredne mogućnosti ugrađenog editora 25
6.5.1. Prekidanje izvršavanja programa 25
6.5.2. Rad s kodom 26
6.5.3. M-Lint 27
6.6. Programiranje u C ili C++ jeziku 27

7. O Simulinku 30

7.1. Simulacija jednostavnih sustava 32
7.2. Povezivanje s MATLAB-om 34
7.3. Izrada funkcijskih blokova 35

8. Literatura 36

Kratke upute za korištenje MATLAB-a

1. 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, jednostavni matrični laboratorij (*Matrix Laboratory*), 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. Korišten je Fortran i dijelovi biblioteka LINPACK i EISPACK. Početkom 80-tih se prelazi na C programski jezik uz dodavanje novih mogućnosti, i to prvenstveno u područjima obradbe signala i automatskog upravljanja. Od 1984. MATLAB je dostupan kao komercijalni proizvod tvrtke MathWorks.

Danas svojstva MATLAB-a daleko prelaze originalni "matrični laboratorij". Radi se o interaktivnom sustavu i programskom jeziku za opća tehnička i znanstvena izračunavanja. Osim osnovnog sustava postoje i brojni programski paketi koji ga proširuju te pokrivaju gotovo sva područja inženjerske djelatnosti: obradu signala i slike, 2D i 3D grafičke prikaze, automatsko upravljanje, identifikaciju sustava, statističke obrade, analizu u vremenskoj i frekvencijskoj domeni, simboličku matematiku i brojne druge. Jedan od važnijih paketa je SIMULINK—vizualni alat koji omogućuje simulaciju kontinuiranih i diskretnih sustava pomoću funkcijskih blok dijagrama te time ne zahtijeva od korisnika detaljno poznavanje sintakse nekog programskog jezika.

MATLAB je također zamišljen kao sustav u kojem korisnik na jednostavan način može graditi svoje vlastite alate i biblioteke te modificirati postojeće. U tu svrhu se koristi jednostavni programski jezik. Također je moguće koristiti C, Fortran, Adu ili Javu.

2. Osnovne mogućnosti MATLAB-a

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

Po svojoj formi MATLAB je interaktivni jezik—*interpreter*, namijenjen prvenstveno 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, C++, PASCAL, BASIC i sl.).

Nakon pokretanja MATLAB-a, 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 primjer je obično računanje s brojevima:

```
» 2+3 <ENT>

ans =
     5

» 7-4*5.1 <ENT>

ans =
    -13.4000

»
```

Rezultat se pojavljuje odmah u sljedećem retku (ans = -13.4000), a nakon njega slijedi oznaka za unos nove 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). Ako niste sigurni u redoslijed izvođenja operacija najbolje je koristiti zagrade:

```
» (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.

Pogledajmo sada kako izgleda računanje s matricama. Definirajmo matricu A s dva retka i tri stupca te matricu B s tri retka i dva stupca. 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>
```

Stavimo li točku-zarez (;) na kraj naredbe time označavamo da ne želimo ispis rezultata. Bez obzira na točku-zarez obje varijante naredbe definiraju nove *varijable* u radnom prostoru—realne matrice `A` i `B`. 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
```

Ponekad želimo zadati niz brojeva. 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

» B(:,1) = A(:,1) <ENT>   % operandi su različitih dimenzija
??? Subscripted assignment dimension mismatch.
```

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 jer se `i` ili `j` obično iskoriste kao brojači u `for` petlji. 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
```

Pogledajmo sada koje elementarne matematičke funkcije postoje. Detaljan popis funkcija možete dobiti naredbom `help elfun`. Na primjer, `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.

Primijenimo li takve funkcije na matrice MATLAB računa odgovarajuću operaciju posebno za svaki element matrice:

```
» A <ENT>                              % ispis elementa matrice A

A =

     1     2     3
     4     5     6

» sqrt(A) <ENT>                        % računamo korijen svakog elementa iz A

ans =

    1.0000    1.4142    1.7321
    2.0000    2.2361    2.4495
```

2.1. 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
```

Ovakva pomoć je zamišljena kao podsjetnik u radu s MATLAB-om. Osim nje 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.

Svu MATLAB dokumentacija u elektroničkom obliku (HTML) možete pronaći na zavodskom serveru http://matlab.zesoi.fer.hr/ (pristup je dozvoljen samo s zavodskih računala).

Ukoliko ne znamo točno ime naredbe, vrlo je korisna naredba `lookfor` koja daje popis naredbi koje u opisu ili imenu sadrže 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 zvučno upozorenje[1]. 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')
```

2.2. Osnovne operacije

2.2.1. 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 (matrice moraju biti ulančane):

```
» 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 matrice moraju biti istih dimenzija:

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

[1] Od inačice 7 MATLAB odmah otvara interaktivnu listu mogućih naredbi ako postoji više mogućih završetaka.

```
S =
     5     7     9
     5     7     9
```

MATLAB omogućuje i operacije na matricama koje se izvršavaju na svakom članu matrice posebno. Primjer je množenje skalarom:

```
» 2 * A <ENT>

ans =
     2     4     6
     8    10    12
```

Na jednak način se interpretira i zbrajanje 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
```

2.2.2. 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
```

2.2.3. 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. Nadalje, operatori &, | i ~ se primjenjuju upravo na matrice te je rezultat opet matrica popunjena jedinicama ili nulama ovisno o rezultatu operacije. Kada se radi s skalarnim veličinama mogu se koristiti i operatori && i || koji su neznatno brži:

```
» 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

» B & C <ENT>

ans =

     1     0     1
     1     1     1

» B && C <ENT>              %  operator je definiran samo za skalarne vrijednosti
??? Operands to the || and && operators must be convertible to logical
scalar values.
```

2.3. 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
```

2.4. 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

» max(A, [], 2) <ENT>

ans =

     3
     6                              % najveći elementi po retcima (dimenzija 2)
```

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` ili korištenjem dvotočke:

```
» max(max(A)) <ENT>

ans =

     6

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

ans =

     6

» max(A(:)) <ENT>

ans =

     6
```

Naime, naredba `max` i sve slične naredbe operiraju na stupcima osim ako se radi o vektoru, odnosno takve naredbe ne razlikuju vektor-retke ili vektor-stupce. Nadalje, za višedimenzionalne matrice naredbe operiraju uzduž zadnje

dimenzije po kojoj struktura ima barem 2 elementa 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 1×2·3.

3. 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. Želimo li crtati u neki određeni prozor prije crtanja ga je potrebno odabrati, odnosno učiniti aktivnim. Stoga svaki grafički prozor ima svoj broj koji ga identificira te koji se koristi prilikom 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
» h = figure <ENT>            % otvaramo novi prozor koji ima oznaku h - dodijeljena
                              % oznaka je prva slobodna oznaka
h =

     3
```

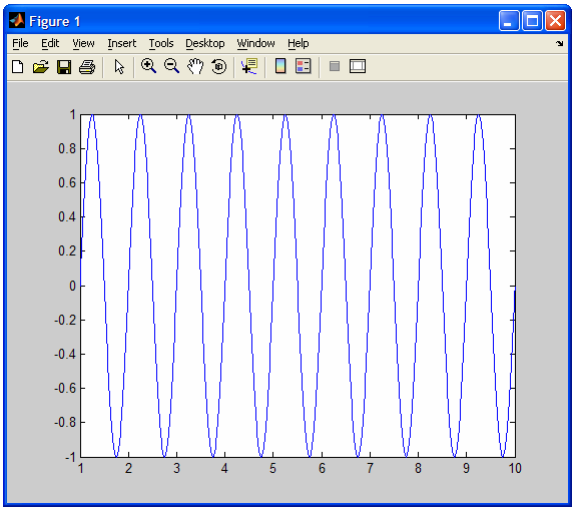
Ako koristimo naredbu `plot` ili bilo koju naredbu za crtanje te ako ne postoji aktivni grafički prozor MATLAB će otvoriti jedan.

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` zadane točke oblika (*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
```

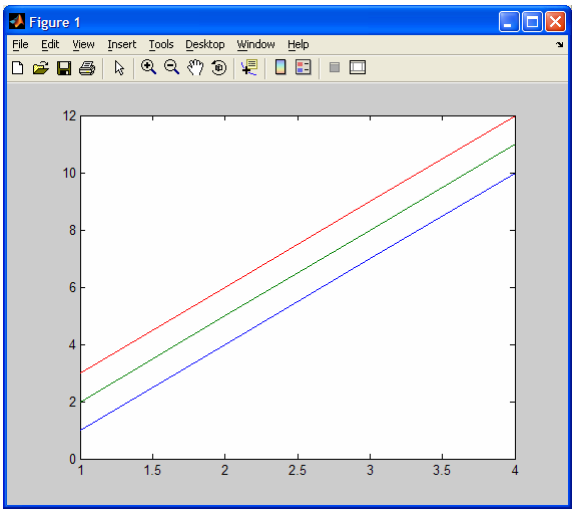
Osim prikaza jednog vektora u zavisnosti o indeksu možemo nacrtati i već prethodno zadane parove točaka (*x*, *y*). Pri tome se funkciji `plot` prosljeđuju dva vektora od kojih prvi sadrži *x*-koordinate dok drugi sadrži *y*-koordinate. Pri tome oba vektora moraju imati jednak broj elemenata. Ovakav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

4. 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
```

5. 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
```

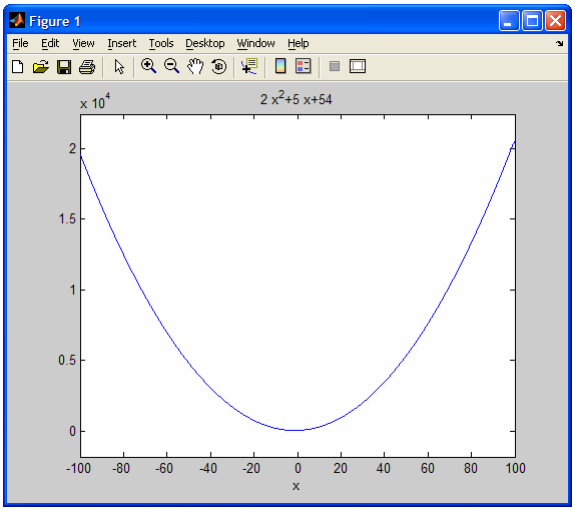
Može se izračunati Laplaceova transformacija:

```
» 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. Pogledajmo na primjeru kako bi nacrtali jednostavnu kvadratnu funkciju te odredili njenu vrijednost za neki broj:

```
» sym('2*x^2+5*x+54') <ENT>          % definiramo kvadratnu funkciju

ans =

2*x^2+5*x+54

» ezplot(ans,[-100,100]) <ENT>       % crtamo je na intervalu [-100, 100]
» subs(ans,'x',5) <ENT>              % računamo njenu vrijednost u točci 5

ans =

129
```



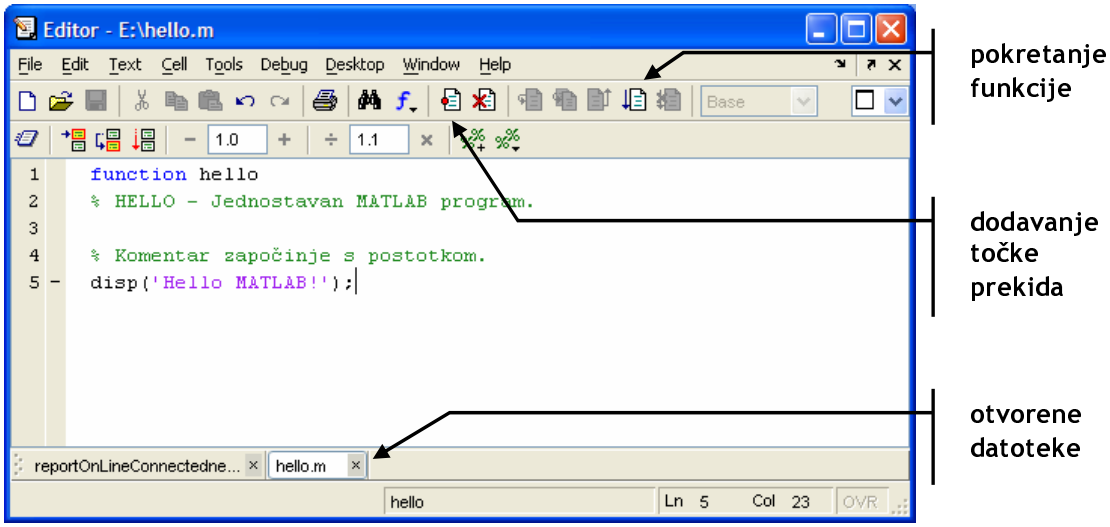
6. Programiranje u MATLAB-u

6.1. 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
```

6.2. 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 ako 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 nepreneseni varijablama pridružimo neku predefiniranu vrijednost. U naš program za zbrajanje bi mogli dodati slijedeću provjeru:

```
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
```

U tom slučaju bi se funkcija izvršila čak i ako ne definiramo ulazne vrijednosti.

6.3. 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 način da brojač unutar petlje poprima sve vrijednosti stupaca unutar zadane matrice.

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

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

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

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

Za kompliciranija uvjetna izvršavanja obično koristimo naredbu `switch`:

```
a = 10;
switch a
   case a < 9
      disp('Broj je manji od devet.');
   case a > 10
      disp('Broj je veći od deset');
   otherwise
      disp('Čestitamo! Osvojili ste Bingo!');
end
```

6.4. 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
```

```
21.   izlaz1 = TeskaMatematika(ulaz1, ulaz2, ...);
22.   izlaz2 = JosTezaMatematika(ulaz1, ulaz2, lokalna_varijabla);
23.
24.   % Provjera izlaznih varijabli
25.   if nargout ~= 3
26.      warning('Neispravan broj izlaznih argumenata.');
27.   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)
```

6.5. Napredne mogućnosti ugrađenog editora

Ugrađeni MATLAB Editor podržava i neke napredne mogućnosti. Od važnijih ćemo spomenuti mogućnosti prekida izvršavanja programa, rad s kodom po dijelovima te automatsku provjeru koda.

6.5.1. Prekidanje izvršavanja programa

Razmotrimo najjednostavniji program za zbrajanje dva broja koji sadrži namjerno unesenu pogrešku (MATLAB razlikuje varijable a i A):

```
1.   function c = zbroji(a, b)
2.   % ZBROJI - Zbrajanje dva broja.
3.
4.   c = A + b;
```

Pokrenemo li program MATLAB javlja poruku o grešci:

```
» zbroji(2, 3) <ENT>                % pozivamo naš program
```

```
??? Undefined function or variable 'A'.

Error in ==> zbroji at 4
c = A + b;
```

U ovom slučaju je naravno trivijalno ispraviti pogrešku, no pogledajmo kako bi mogli postupiti i u znatno kompliciranijim situacijama. Ugrađeni editor nam omogućuje da definiramo točke prekida izvršenja programa. Osim definiranja točaka prekida možemo odabrati i prekid programa svaki put kada se dogodi pogreška odabirom `Debug→Stop If Errors/Warnings...` iz izbornika. Pokrenimo sada naš program:

```
» zbroji(2, 3) <ENT>                % pozivamo naš program
??? Undefined function or variable 'A'.

Error in ==> zbroji at 4
c = A + b;
4   c = A + b;
K» <ENT>                            % primijetite slovo K (debugging mode)
K» whos <ENT>                       % svaka MATLAB naredba je dopuštena
  Name      Size                    Bytes  Class

  a         1x1                         8  double array
  b         1x1                         8  double array

Grand total is 2 elements using 16 bytes

K» A = a; <ENT>                     % vidmo da ne postoji A te je definiramo
K» c = A + b; <ENT>                 % računamo zbroj
K» dbcont <ENT>                     % nastavljamo s izvođenjem programa
```

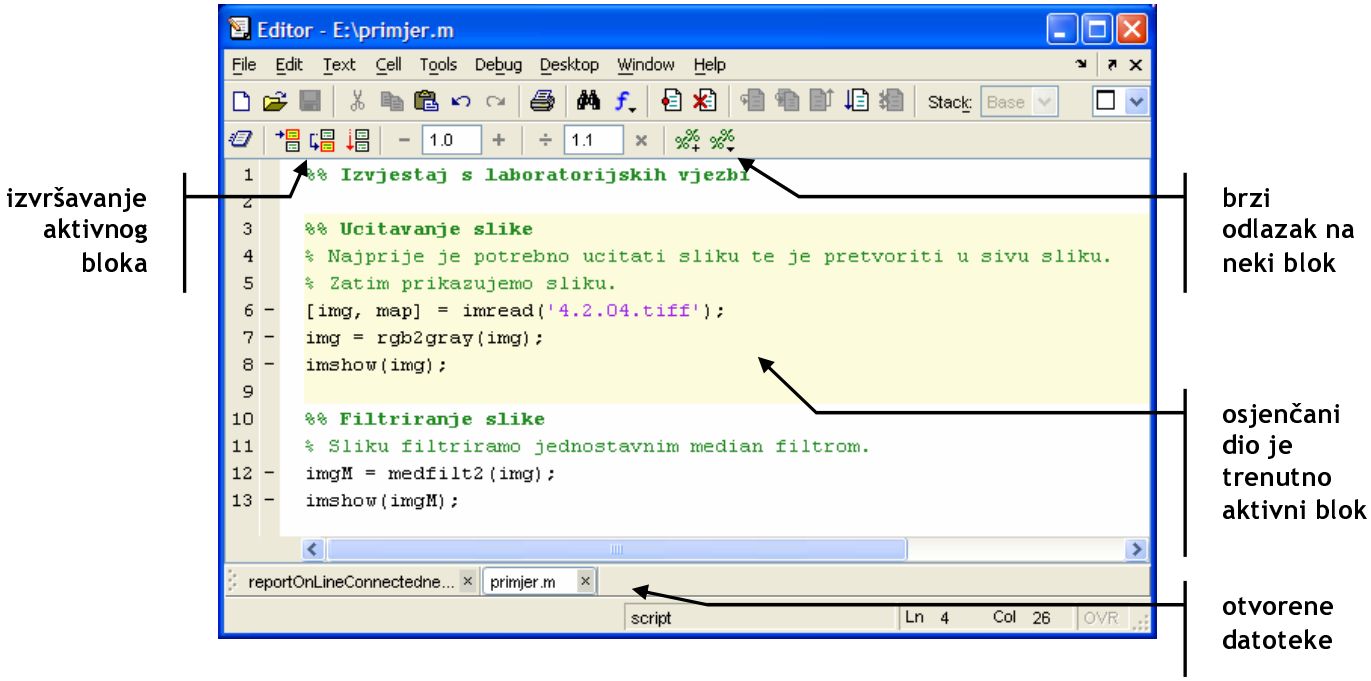
6.5.2. Rad s kodom

Od inačice 7 MATLAB podržava novi oblik komentara unutar koda te automatsko generiranje izvještaja na temelju koda. Obični komentar počinje znakom %, a osim njega uveden je i komentar koji počinje s %%. Takav komentar određuje početak bloka koji traje sve do slijedećeg retka s %% (*cell mode*). Ugrađeni editor podržava jednostavnu manipulaciju s takvim dijelovima koda što uključuje i izvođenje koda koji se nalazi između takva dva komentara.

Osim jednostavnog rada s blokovima moguće je odmah iz takvog dobro komentiranog i strukturiranog koda dobiti izvještaj u nekom od popularnih formata (Microsoft Word, PDF, HTML, LaTeX):

```
» primjer <ENT>                     % pozivamo skriptu
» publish('primjer', 'html') <ENT> % kada smo zadovoljni rezultatima možemo
                                    % iz koda odmah dobiti izvještaj u traženom
ans =                               % obliku

E:\html\primjer.html
```

Izvještaj se nalazi u datoteci E:\html\primjer.html i sadrži kod uz komentare koji su povezani preko linkova te sve rezultate koji su prikazani pomoću neke od MATLAB-ovih funkcija.



6.5.3. M-Lint

M-Lint je alat koji provjerava kod te identificira moguće probleme kod prenosivosti te način upotrebe varijabli i elemenata programskog jezika. Rezultat analize je detaljni izvještaj u kojem su nabrojani mogući problemi s kodom. Alat je dostupan iz izbornika ugrađenog editora ako odaberemo stavku `Tools→Save and Check Code with M-Lint`.

6.6. Programiranje u C ili C++ jeziku

MATLAB nam omogućuje pisanje programa i u nekom od običnih programskih jezika. Ovdje ćemo ukratko pokazati kako napisati novu funkciju u programskom jeziku C.

Razmotrimo najprije jednostavan primjer koji samo ispisuje pozdravnu poruku te broj argumenata:

```
1.    #include <mex.h>
2.
3.    /* Jednostavni MEX program */
4.    void mexFunction(int nlhs, mxArray *plhs[],
5.                     int nrhs, const mxArray *prhs[])
6.    {
7.      mexPrintf("Hello MATLAB!\n");
8.
9.      if (0 != nrhs)
10.       {
11.         mexPrintf("Imamo %d ulaznih parametara.\n", nrhs);
12.       }
13.     /* if */
14.
15.     if (0 != nlhs)
```

```
16.       {
17.         mexPrintf("Imamo %d izlaznih parametara.\n", nlhs);
18.       }
19.     /* if */
20.
21.     return;
22.
23.   }
24.   /* mexFunction */
```

Kada pišemo program za MATLAB ulazna funkcija nije `main` već je `mexFunction`. Toj funkciji MATLAB prosljeđuje broj ulaznih (`nrhs`) i izlaznih (`nlhs`) argumenata te same argumente kao polja pokazivača (`*plhs` i `*prhs`). Prisjetimo se da su sve varijable unutar MATLAB-a matrice. Iz C programskog jezika ih vidimo kao strukture tipa `mxArray` u kojoj su spremljene dimenzije i tip matrice te pokazivač na memorijsku lokaciju od koje započinju podaci. Ako se radi o običnim matricama one su spremljene po stupcima[2].

Kada smo napisali program potrebno ga je prevesti. MATLAB se isporučuje s slobodnom dostupnim običnim C prevodiocem LCC kojeg možemo iskoristiti za prevođenje programa. Neka se naš program nalazi u datoteci `hello.c`. Prevodimo ga korištenjem naredbe `mex`:

```
» mex hello.c <ENT>                    % prevodimo program hello.c
» what <ENT>                           % u trenutnom direktoriju sada postoji m i
                                       % MEX funkcija istog imena
M-files in the current directory E:\

hello      primjer    zbroji

MEX-files in the current directory E:\

Hello

» hello <ENT>                          % u tom slučaju se uvijek poziva MEX funkcija
Hello MATLAB!
» hello(a, b, c, d, e) <ENT>
Hello MATLAB!
Imamo 5 ulaznih parametara.
```

Pogledajmo sada nešto kompliciraniji primjer u kojem računamo razliku matrica po elementima. Zbog jednostavnosti su iz koda izostavljene sve provjere ulaznih varijabli:

```
1.   #include <mex.h>
2.
3.   /* Racunamo razliku dvije matrice. */
4.   void mexFunction(int nlhs, mxArray *plhs[],
5.                    int nrhs, const mxArray *prhs[])
6.   {
7.
8.     const mxArray *x1, *x2;
```

[2] Matrice se u MATLAB-u spremaju po stupcima što možemo vidjeti i iz samog MATLAB-a zadamo li naredbu `x(:)` za neku matricu `x`. Tom naredbom ispisujemo elemente matrice po redu kako su spremljeni u memoriji, a usporedite li rezultat s elementima matrice `x` primijetiti ćete kako redom ispisujemo stupce matrice.

```
9.      int nex1, nex2;
10.
11.     mxArray *y;
12.     int ndy, *dimy;
13.
14.     int i;
15.
16.     double *px1, *px2, *py;
17.
18.     /* Najprije dohvacamo pokazivace na podatke spremljene
19.        u ulaznim podacima. */
20.     x1 = prhs[0];
21.     nex1 = mxGetNumberOfElements(x1);
22.     px1 = (double *)mxGetData(x1);
23.
24.     x2 = prhs[1];
25.     nex2 = mxGetNumberOfElements(x2);
26.     px2 = (double *)mxGetData(x2);
27.
28.     if (nex1 != nex2)
29.       {
30.         mexErrMsgTxt("Matrice nemaju jednak broj elemenata.\n");
31.         return;
32.       }
33.     /* if */
34.
35.     /* Alociramo prostor za rezultat. */
36.     ndy = mxGetNumberOfDimensions(x1);
37.     dimy = (int *)mxGetDimensions(x1);
38.     y = mxCreateNumericArray(ndy, dimy, mxDOUBLE_CLASS, mxREAL);
39.     if (NULL == y)
40.       {
41.         mexErrMsgTxt("Nema dovoljno memorije.\n");
42.         return;
43.       }
44.     /* if */
45.
46.     plhs[0] = y;
47.     nlhs = 1;
48.
49.     py = (double *)mxGetData(y);
50.
51.     /* Sada jednostavno racunamo razlike. */
52.     for (i=0; i<nex1; i++)
53.       {
54.         py[i] = px1[i] - px2[i];
55.       }
56.     /* for */
57.
58.   }
59.   /* mexFunction */
```

Glavni dio funkcije u kojem stvarno računamo razliku elemenata se sastoji od jedne `for` petlje koja prolazi kroz sve elemente matrica redom kako su zapisani u memoriji te računa razliku (redovi 51.-56.), dok se većina koda odnosi na dohvat i interpretaciju varijabli (redovi 18.-33.) te kreiranje izlazne varijable (redovi 35.-47.). Pri kreiranju izlazne varijable kao i pri dohvatu

vrijednosti iz prenesenih struktura koristimo funkcije iz MATLAB-ove biblioteke namijenjene manipulaciji s matricama (sve funkcije koje započinju s `mx`). Već u prvom primjeru smo se susreli s funkcijama iz MATLAB-ove biblioteke koje započinju s `mex`. Takve funkcije uglavnom služe za interakciju s radnim prostorom MATLAB-a (ispis poruka korisniku te eventualni dohvat varijabli).

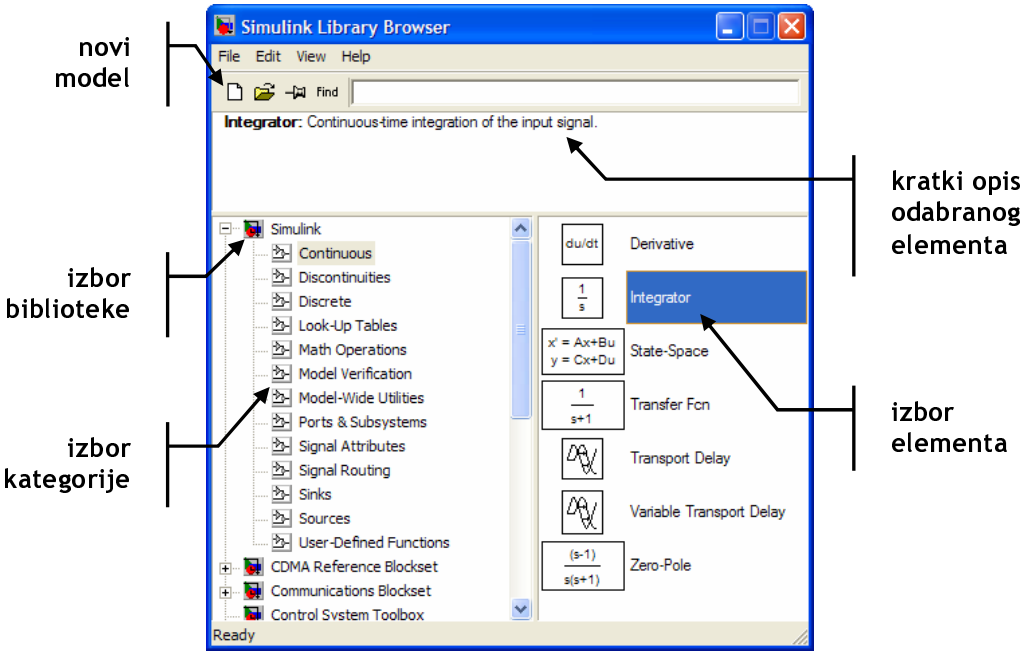
```
» mex razlika.c <ENT>                          % prevodimo program razlika.c
» A = [1 2 3; 4 5 6] <ENT>                     % definiramo matricu A

A =

     1     2     3
     4     5     6

» B = [1 1; 1 1; 1 1] <ENT>                    % definiramo matricu B

B =

     1     1
     1     1
     1     1

» razlika(A, B) <ENT>                          % računamo razliku po elementima
                                               % naša funkcija je definirana jer
ans =                                          % tražimo samo da matrice A i B imaju
                                               % jednaki broj elemenata
     0     1     2
     3     4     5

» reshape(A(:) - B(:), size(A)) <ENT>          % isti postupak se može izvesti u
                                               % jednom retku koristimo li MATLAB
ans =                                          % naredbe

     0     1     2
     3     4     5
```

Želimo li koristiti C++ jezik moramo imati neki C++ prevodioc jer se MATLAB isporučuje samo s LCC-om koji ne podržava C++. Nadalje je potrebno dodati `extern void _main();` liniju prije poziva ulazne procedure `mexFunction`.

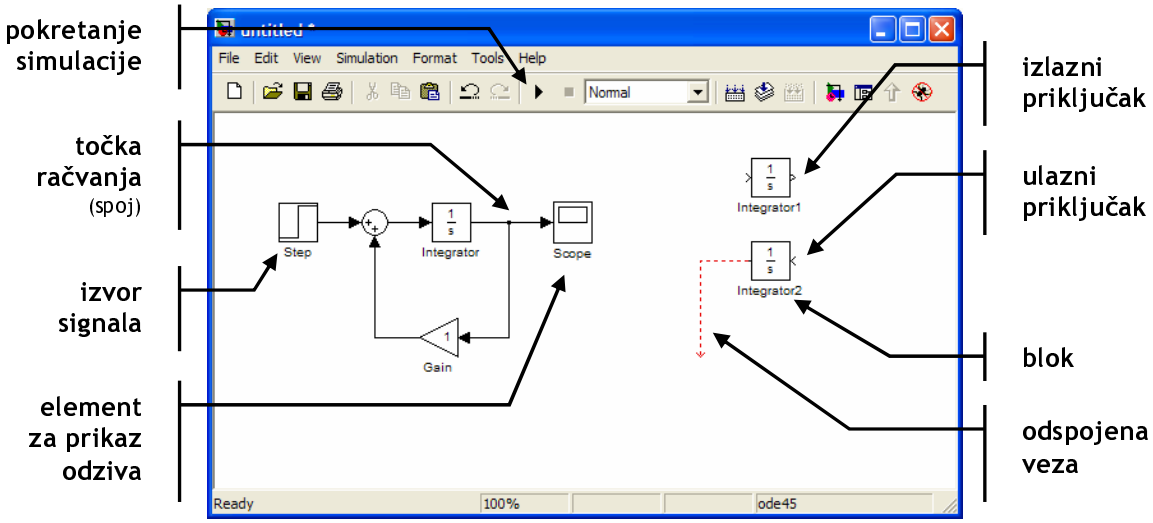
7. O Simulinku

Simulink je dio MATLAB-a namijenjen simuliranju dinamičkih sustava. Za sam unos i opis sustava koji se simulira koristi se jednostavno grafičko sučelje u kojem sastavljamo/crtamo model kombinirajući gotove komponente. Takvim pristupom je simulacija sustava značajno olakšana jer se od korisnika ne zahtijeva unos diferencijalnih ili diferencijskih jednadžbi koje opisuju sustav uz poznavanje MATLAB programskog jezika već je dovoljno znati blok-shemu sustava.

Simulink se pokreće unutar MATLAB-a zadavanjem naredbe `simulink` ili odabirom ikone iz alatne trake. Nakon pokretanja Simulinka otvara se prozor `Simulink Library Browser` prikazan na slici.



Novootvoreni prozor sadrži kolekciju svih blokova koje koristimo pri sastavljanju modela. U donjem dijelu prozora s desne strane nalaze se sve raspoložive kategorije blokova. Odabirom neke kategorije na lijevoj strani Simulink prikazuje sve blokove dostupne unutar odabrane kategorije. No da bi mogli slagati i povezivati blokove te tako definirati sustav kojeg želimo simulirati najprije moramo otvoriti novi model odabirom `File→New Model` ili klikom na ikonu alatne trake. Sada se otvara novi prozor u kojem sastavljamo model.



Na model element dodajemo tako da ga odaberemo, odvučemo i ispustimo unutar prozora novog modela. Svaki element kojeg smo dodali ima ulazne priključke označene s > i izlazne priključke označene s >. Blokove spajamo

crtanjem veza između blokova (pokazivač se pretvara u alat za crtanje kada ga postavimo iznad ulaznih ili izlaznih priključaka). Dvostrukim klikom na pojedini blok dobivamo izbornik u kojem postavljamo svojstva danog bloka.

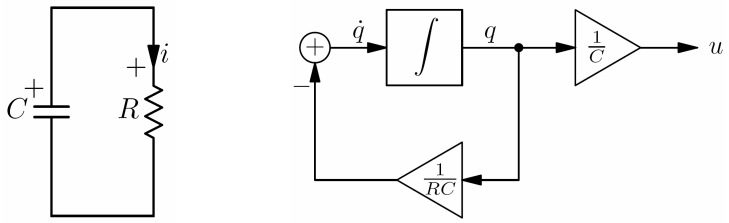
7.1. Simulacija jednostavnih sustava

Sastavimo i simulirajmo pomoću Simulinka jednostavni sustav prvog reda koji se sastoji od idealnog kapaciteta C i otpornika R kako je prikazano na slici. Sustav možemo opisati diferencijalnom jednadžbom

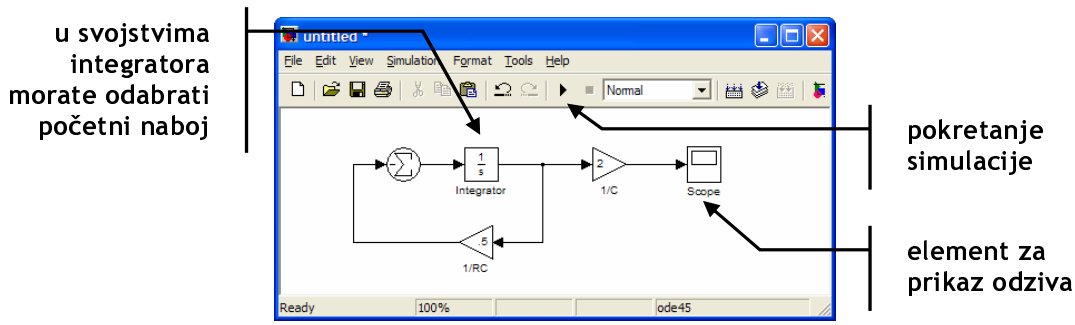
$$\frac{q}{RC}+\frac{dq}{dt}=0$$

Odaberemo li za izlaz sustava napon na otporniku R kao konačno rješenje dobivamo

$$u(t)=\frac{q_0}{C}e^{-\frac{1}{RC}(t-t_0)}.$$



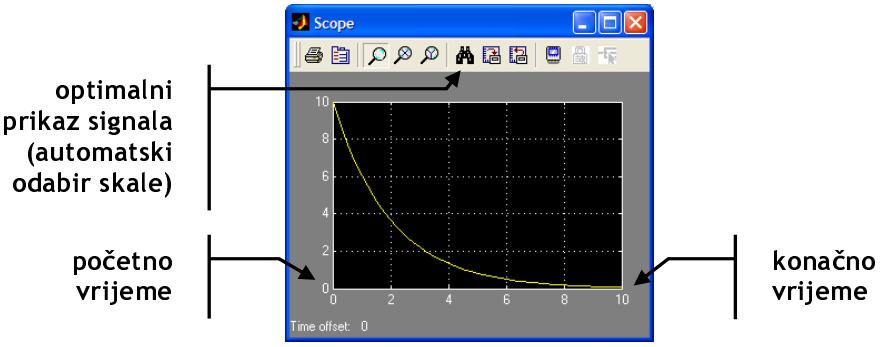
Mi naravno želimo dani sustav simulirati uz pomoć Simulinka. Pretpostavimo da raspolažemo s elementom za integriranje. Promotrimo li diferencijalnu jednadžbu koja opisuje sustav vidimo da je derivacija naboja proporcionalna naboju. Ako bi izlaz iz integratora bio naboj na kapacitetu C, ulaz u integrator tada mora biti derivacija naboja koja je proporcionalna s tim istim nabojem. Vidimo da je potrebno vratiti izlaz iz integratora na njegov ulaz uz odgovarajuće pojačanje. Time smo dobili blok-shemu sustava koju možemo odmah nacrtati u Simulinku.



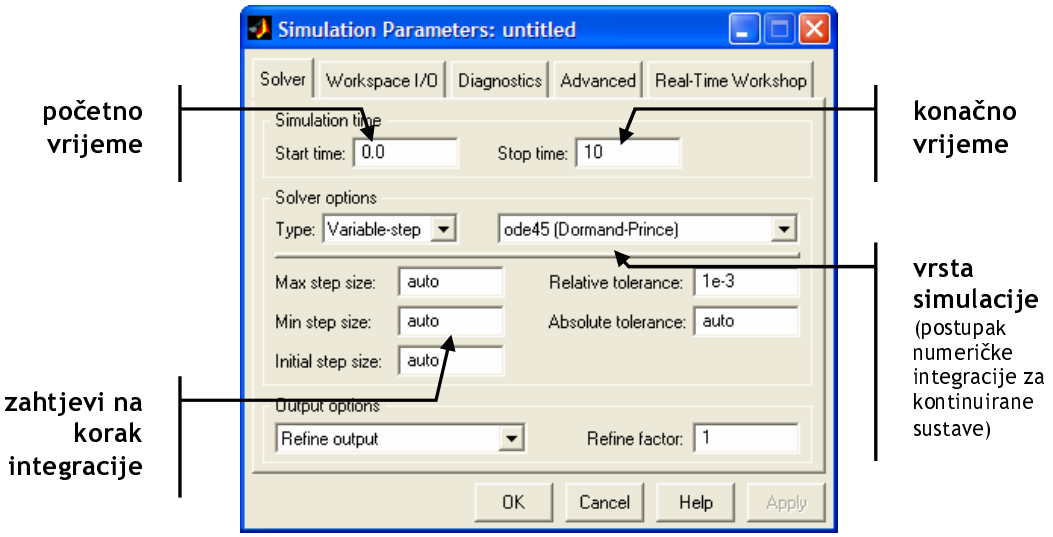
Odaberimo $R=4$ i $C=0{,}5$ te početni naboj $q_0=5$. Sada možemo nacrtati shemu u Simulinku. Pri tome pripazite na to da je blok za integriranje označen s

$1/s$ a ne s integralom zbog veze s Laplaceovom transformacijom. Početno stanje odabirete u svojstvima bloka za integriranje (dvostruki klik na blok).

Da bi mogli vidjeti rezultate simulacije nakon pokretanja morate aktivirati `Scope` blok dvostrukim klikom. Sada se otvara prozor prikazan na slici.



U prozoru vidimo da je odziv sustava eksponencijalna funkcija koja trne kako vrijeme teži k beskonačnosti. Početna vrijednost je određena nabojem na kapacitetu. Simulacijom smo dobili vrijednosti odziva od trenutka $t = 0$ do trenutka $t = 10$. Te trenutke podešavamo u dijalogu parametara simulacije (odaberite stavku `Simulation→Simulation Parameters` iz izbornika). Važni parametri simulacije uz početno i konačno vrijeme su vrijednosti vremenskog koraka i tip numeričke integracije koji se koristi.



MATLAB i Simulink raspolažu s nekoliko različitih metoda numeričke integracije (*ODE Solvers*) koji rješavaju probleme oblika

$$y' = F(t, y).$$

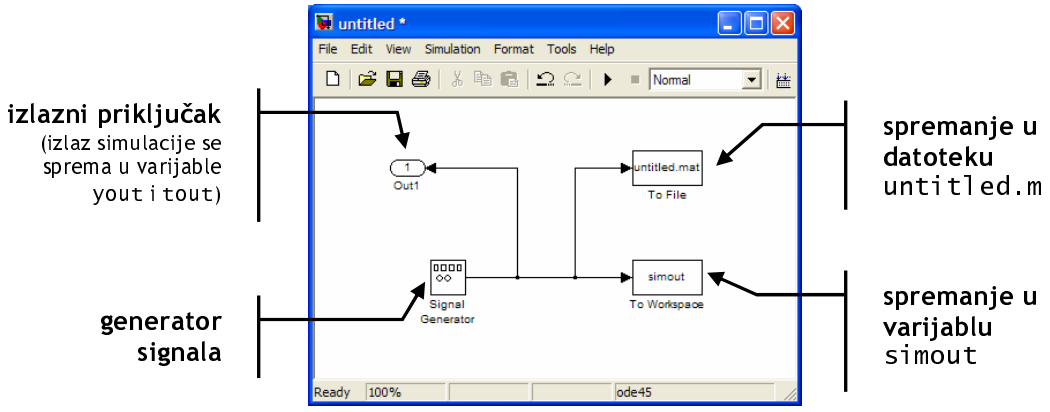
Pretpostavljena metoda numeričke integracije je ode45 i prikladna je za većinu problema. Od ostalih raspoloživih metoda simulacije sustava unutar Simulinka

bitno je spomenuti `discrete` postupak koji je primjenjiv za simulaciju diskretnih sustava ili općenito bilo kojih sustava koji nemaju kontinuirane varijable stanja. Pravilnim izborom postupka i parametara simulacije možete značajno skratiti vrijeme potrebno za simulaciju a istodobno povećati točnost.

7.2. Povezivanje s MATLAB-om

Simulink je sastavni dio MATLAB-a te se pri izgradnji modela mogu koristiti sve MATLAB funkcije. Također se modelu mogu proslijediti neki ulazi, a i rezultati simulacije se mogu dohvatiti iz ljuske.

Pogledajmo najprije kako rezultate simulacije spremiti bilo u datoteku bilo u neku varijablu koja je dostupna iz ljuske. Najjednostavniji primjer prikazan je na slici. U prikazanom slučaju izlaz iz generatora signala spremamo na različita mjesta.



Nakon pokretanja simulacije u MATLAB-u možemo pristupiti varijablama `tout` i `yout` (`tout` sadrži vremenske trenutke, a `yout` vrijednosti signala) strukturi `simout` (struktura sadrži i dodatne informacije o signalu) te datoteci `untitled.mat`:

```
» whos <ENT>                              % u ljusci možemo pristupiti rezultatima

  Name          Size                     Bytes  Class

  simout         1x1                      1202  struct array
  tout          51x1                       408  double array
  yout          51x1                       408  double array

Grand total is 181 elements using 2018 bytes

» whos -file untitled <ENT>               % isti rezultati su spremljeni u datoteku

  Name       Size                       Bytes  Class

  ans        2x51                         816  double array

Grand total is 102 elements using 816 bytes
```

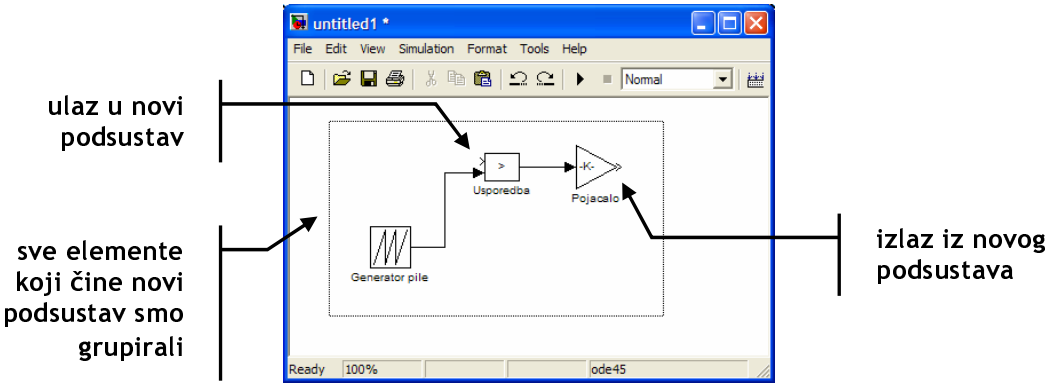
Sada na tako dobivene rezultate simulacije možemo primijeniti bilo koju od raspoloživih funkcija.

Ponekad je potrebno izvršiti veliki broj simulacija s različitim parametrima. U tom slučaju uobičajen način koji se sastoji od pokretanja simulacije, spremanja rezultata te naposljetku mijenjanja parametara nije pogodan. MATLAB podržava naredbu `sim` koja se može koristiti za pokretanje simulacije. Osim naredbe `sim` mogu se koristiti naredbe `set_param` i `get_param` za mijenjanje parametara modela. Kombiniranjem navedenih naredbi možemo napisati skriptu koja postavlja parametre simulacije i sprema rezultate. Osim navedenih postoje i mnoge druge naredbe za automatizirani rad sa Simulinkom.

7.3. Izrada funkcijskih blokova

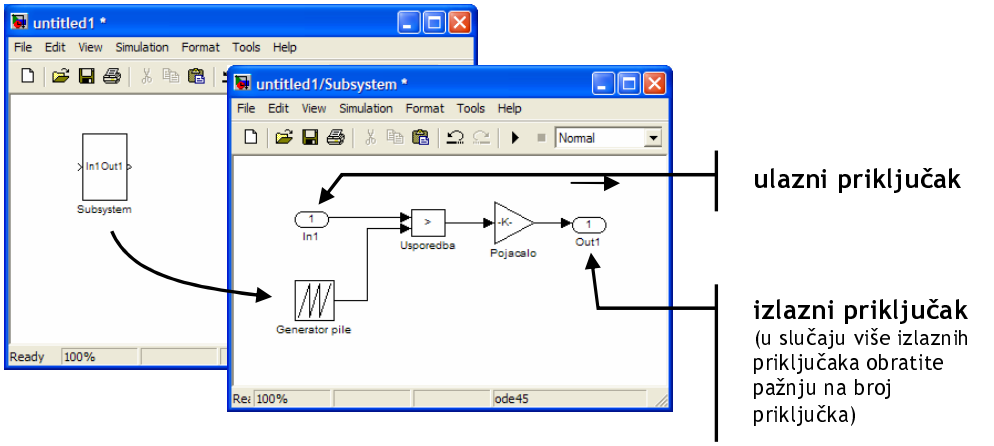
Pravi funkcijski blok za Simulink definiramo pisanjem S-funkcije. Iako za pisanje S-funkcija možemo koristiti C, C++ i neke druge programske jezike, najjednostavnije ih je napisati koristeći MATLAB-ov M programski jezik. Kako je struktura takvih funkcija nešto kompliciranija od uobičajenih MATLAB funkcija i skripti preporučamo vam da pokretanjem `sfundemos` demonstracije pogledate par ilustrativnih primjera.

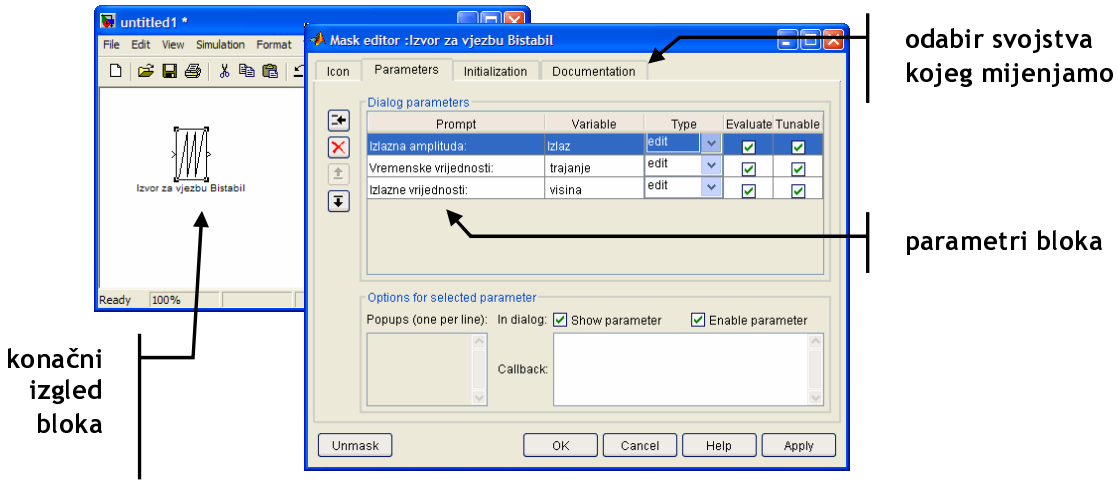
Umjesto pisanja S-funkcija ponekad je dovoljno iskoristiti gotove funkcijske blokove za slaganje novog bloka. Postupak se sastoji od sastavljanja gotovih blokova te grupiranja istih.



Najprije crtamo željeni sustav koji će nam predstavljati jedan blok. Pri tome određene ulaze i izlaze elemenata ne spajamo jer će oni postati novi ulazi i izlazi iz zamišljenog funkcijskog bloka. Sada grupiramo elemente koji čine naš podsustav te odabiremo `Edit→Create Subsystem`. Dobivamo novi blok koji ima onoliko ulaza i izlaza koliko smo ih ostavili odspojenima. Označimo li blok te odaberemo `Edit→Look Under Mask` možemo pogledati kako je složen podsustav.

Sva svojstva takvog sustava, izgled maske i kratki opis za korisnika možemo mijenjati odaberemo li `Edit→Edit Mask` opciju. U novom dijalogu sada postavljamo sve parametre vezane za podsustav.





8. Literatura

1. *Getting Started with* MATLAB *version 7.*, The MathWorks, ožujak 2002.
2. *Getting Started with* MATLAB *version 6.*, The MathWorks, srpanj 2002.
3. *Getting Started with* MATLAB *version 5*, 2. izdanje, The MathWorks, svibanj 1997.
4. Adrian Biran, Moshe Breiner, MATLAB 5 *for Engineers*, 2. izdanje, Addison-Wesley, 1999.

Popis naredbi

Ovaj dio sadrži popis češće korištenih MATLAB funkcija. Popis sadrži funkcije grupirane u skupine. Detalji o svakoj funkciji mogu se pronaći koristeći ugrađenu pomoć.

Opće funkcije

help	ugrađena pomoć
demo	demonstracija mogućnosti MATLAB-a
who	popis definiranih varijabli
what	popis M-datoteka u radnom direktoriju
size	veličina neke varijable (broj stupaca i redaka matrice)
length	duljina vektora
clear	brisanje varijabli
^C	prekidanje izvršenja programa
pack	čišćenje radne memorije unutar MATLAB-a
quit	izlaz iz MATLAB-a
exit	izlaz iz MATLAB-a
why	odgovor na gotovo svako pitanje

Matematičke operacije

+	zbrajanje	+	zbrajanje
-	oduzimanje	-	oduzimanje
*	množenje matrica	.*	množenje član-po-član
/	desno dijeljenje	./	desno dijeljenje član-po-član
\	lijevo dijeljenje	.\	lijevo dijeljenje član-po-član
^	potencija matrice	.^	potenciranje svakog člana
'	Hermitsko konjugiranje	.'	transponiranje

Relacijske i logičke operacije

<	manje od	&	I (po elementima matrice)
<=	manje ili jednako	|	ILI (po elementima matrice)
>	veće od	~	NE
>=	veće ili jednako	&&	I
==	jednako	||	ILI
~=	različito		

Posebni znakovi

=	pridruživanje vrijednosti varijabli
[	definiranje vektora i matrica
]	vidi [
{	definiranje skupova
}	vidi }
(	redoslijed izvršavanja aritmetičkih operacija
)	vidi (
.	decimalna točka (MATLAB ne koristi decimalni zarez)
...	nastavak naredbe je u slijedećem retku
,	razdvajanje funkcijskih argumenata i indeksa
;	kraj retka, zabrana ispisa rezultata
%	komentar
:	indeksiranje, generiranje nizova
!	izvršavanje naredbe operacijskog sustava

Posebne varijable i konstante

ans	rezultat naredbe ako ga ne pridružimo nekoj varijabli
pi	broj pi
i,j	korijen iz -1
Inf	beskonačno
NaN	nedefinirana vrijednost broja (nije broj)
eps	preciznost aritmetike s pomoćnim zarezom
realmax	najveći pozitivni broj
realmin	najmanji pozitivni broj
clock	vrijeme
date	datum
flops	broj operacija od trenutka pokretanja MATLAB-a
nargin	broj ulaznih argumenata funkcije
nargout	broj izlaznih argumenata funkcije

Tekst i tekstualni nizovi

abs	vraća ASCII kod za svaki znak
eval	izvrši naredbu
num2str	pretvori broj u tekst
int2str	pretvori tekst u broj
sprintf	ispiši formatirani izlaz u tekstualni niz
isstr	ispitaj je li varijabla tekstualni niz
strcmp	usporedi dva tekstualna niza
hex2num	pretvori heksadecimalni niz u broj

Crtanje

plot	linearni X-Y graf sa spajanjem točaka
plot3	linearni X-Y-Z graf sa spajanjem točaka
stem	linearni X-Y graf diskretnog niza (bez spajanja točaka)
loglog	logaritamski X-Y graf
semilogx	polu-logaritamski X-Y graf
semilogy	polu-logaritamski X-Y graf
polar	polarni graf
mesh	crtanje trodimenzionalne površine
surf	crtanje trodimenzionalne površine sa sjenčanjem
shading	kontrola načina sjenčanja
contour	crtanje kontura
meshgrid	generiranje parova koordinata u ravnini

Označavanje crteža

title	dodavanje naslova
xlabel	oznaka X-osi
ylabel	oznaka Y-osi
zlabel	oznaka Z-osi
grid	crtanje koordinatne mreže
text	dodavanje teksta u crtež
gtext	dodavanje teksta u crtež pomoću miša
line	dodavanje linije u crtež
ginput	unos koordinata pomoću miša

Kontrola prozora za crtanje

view	postavljanje gledišta
axis	određivanje intervala koordinatnih osi
axes	stvaranje koordinatnih osi
hold	zadržavanje nacrtanog prilikom crtanja (docrtavanje)
shg	čini trenutni aktivni prozor postaje vidljiv
clf	briše sadržaj aktivnog prozora
subplot	dijeli grafički prozor u dijelove
figure	otvara novi prozor za crtanje
close	zatvara aktivni prozor za crtanje
gca	vraća broj pridružen aktivnom paru koordinatnih osi
gcf	vraća broj pridružen aktivnom prozoru
get	dohvaća svojstva nekog grafičkog objekta
set	mijenja svojstva nekog grafičkog objekta
reset	vraća početna svojstva nekog grafičkog objekta

Kontrola toka programa

if	uvjetno izvršenje naredbi
elseif	koristi se s naredbom if
else	koristi se s naredbom if
end	završava if, for, while naredbe
for	ponavljanje bloka naredbi određeni broj puta
while	while petlja
break	bezuvjetni prekid izvršavanja for i while petlji
return	vraćanje iz funkcije
pause	čekanje na korisnika

Rad s datotekama

pwd	prikaz trenutnog radnog direktorija
cd	promjena radnog direktorija
dir	prikaz sadržaja direktorija
ls	prikaz sadržaja direktorija
delete	brisanje datoteka
diary	spremanje svih naredbi u tekstualnu datoteku
load	učitavanje varijabli iz datoteke
save	spremanje varijabli u datoteku
type	prikaz M-datoteke
what	popis M-datoteka u radnom direktoriju
which	prikaz pune staze do M-datoteke ili funkcije
edit	uređivanje M-datoteke
fprintf	formatirani ispis u datoteku
imread	učitavanje podataka iz slikovne datoteke
imwrite	spremanje podataka u slikovnu datoteku
wavread	učitavanje podataka iz WAV datoteke
wavwrite	spremanje podataka u WAV datoteku

Relacijske i logičke funkcije

any	logički uvjet (barem jedan element nije nula)
all	logički uvjet (svi elementi su različiti od nule)
find	traženje indeksa elemenata koji zadovoljavaju uvjet
exist	provjera postojanja varijable
isnan	ispitivanje sadrži li matrica NaN vrijednosti
finite	ispitivanje sadrži li matrica beskonačne vrijednosti
isempty	ispitivanje da li je matrica prazna
isstr	detekcija tekstualnih nizova
strcmp	usporedba tekstualnih nizova

Trigonometrijske funkcije

sin	sinus
cos	kosinus
tan	tangens
asin	arkus sinus
acos	arkus kosinus
atan	arkus tangens
atan2	kut kompleksnog broja
sinh	sinus hiperbolni
cosh	kosinus hiperbolni
tanh	tangens hiperbolni
asinh	arkus sinus hiperbolni
acosh	arkus kosinus hiperbolni
atanh	arkus tangens hiperbolni

Elementarne matematičke funkcije

abs	apsolutna vrijednost
angle	kut kompleksnog broja
sqrt	kvadratni korijen
real	realni dio kompleksnog broja
imag	imaginarni dio kompleksnog broja
conj	konjugiranje
round	zaokruživanje prema najbližem cijelom broju
fix	zaokruživanje prema nuli
floor	zaokruživanje prema minus beskonačnosti
ceil	zaokruživanje prema plus beskonačnosti
sign	predznak
rem	ostatak pri dijeljenju
mod	ostatak pri dijeljenju (s predznakom)
exp	eksponencijalna funkcija s bazom e
log	prirodni logaritam
log10	dekadski logaritam

Polinomi

poly	karakteristični polinom
roots	nule polinoma
polyval	računanje vrijednosti polinoma u točki
polyvalm	računanje vrijednosti matričnog polinoma
conv	množenje polinoma
deconv	dijeljenje polinoma
residue	rastavljanje u parcijalne razlomke
polyfit	određivanje interpolacijskog polinoma

Rukovanje matricama

rot90	rotacija matrice
fliplr	zrcaljenje matrice lijevo-desno
flipud	zrcaljenje matrice gore-dolje
flipdim	zrcaljenje u zadanoj dimenziji
diag	glavna dijagonala matrice
tril	donji trokutasti dio
triu	gornji trokutasti dio
reshape	promjena dimenzija matrice
.'	transpozicija
:	pretvaranje matrice u vektor

Elementarne matrične funkcije

expm	matrična eksponencijala
logm	matrični logaritam
sqrtm	matrični kvadratni korijen
poly	karakteristični polinom
rank	rang matrice
det	determinanta
trace	trag
cdf2rdf	pretvaranje kompleksne-dijagonale u realnu-dijagonalu
eig	vlastite vrijednosti i vlastiti vektori
null	nul-potprostor
orth	ortogonalizacija
inv	inverzna matrica (za kvadratne matrice)
pinv	pseudoinverzna matrica
lu	LU dekompozicija

Analiza podataka po stupcima

max	maksimalna vrijednost
min	minimalna vrijednost
mean	srednja vrijednost
median	medijan
std	standardna devijacija
sort	sortiranje
sum	zbroj elemenata po stupcima
prod	umnožak elemenata po stupcima
cumsum	kumulativni zbroj po stupcima
cumprod	kumulativni umnožak po stupcima
diff	diferencija po stupcima
hist	histogram
corrcoef	koeficijent korelacije
cov	kovarijanca

Obrada signala

sinc	sin(pi*x)/(pi*x) funkcija
diric	periodična sinc funkcija
abs	apsolutna vrijednost
angle	kut kompleksnog broja
phase	faza kompleksnog niza (bez premotavanja)
conv	konvolucija
xcorr	korelacija
xcov	kovarijanca
deconv	dekonvolucija
freqs	frekvencijska karakteristika kontinuiranog sustava
freqz	frekvencijska karakteristika diskretnog sustava
fft	brza Fourierova transformacija (FFT)
fft2	2D FFT
fftn	N-dimenzionalna FFT
ifft	inverzna FFT
ifft2	inverzna 2D FFT
ifftn	inverzna N-dimenzionalna FFT
fftshift	zamjena kvadranata unutar matrice
dct	diskretna kosinusna transformacija
idct	inverzna diskretna kosinusna transformacija
psd	gustoća spektra snage
specgrm	spektrogram
decimate	decimacija diskretnog niza
interp	interpolacija diskretnog niza