3. Programiranje u Matlab-u

3.1 M-datoteke

M-datoteka nije ništa drugo do obična tekstualna datoteka koja sadrži MATLAB komande i sačuvana je sa ekstenzijom *.m*. Postoje dva tipa M-datoteka, **skriptovi** i **funkcije.** Preporučljivo je da se M-datoteke pišu u MATLAB-ovom ugrađenom editoru, sa obzirom da je odlično integrisan u okruženje i da pruža mnoge mogućnosti za otkrivanje i ispravljenje grešaka u kodu (debugging) i njegovu optimizaciju. Ovaj editor se može otvoriti iz komandnog prozora korišćenjem naredbe *edit.* Druge opcije su da se otvori iz padajućeg menija *File/New/Script(Function)* ili pritiskom kombinacije tastera *Ctrl+N*. Naravno moguće je koristiti i bilo koji drugi tekst editor. Da bi M-datoteka mogla da se izvršava pozivom iz komandnog prozora mora biti sačuvana u direktorijumu koji se nalazi u MATLAB-ovoj stazi za pretragu (path). Koristite naredbu *editpath* ili raspoložive menije kako biste izmenili stazu za pretragu.

Da bismo sačuvali M-datoteku u padajućem meniju ćemo izabrati:

File > Save As.

Izabraćemo ime za datoteku, na primer *proba* i dodati ekstenziju *.m*, a zatim kliknuti na *Save*. Datoteku treba sačuvati u direktorijumu koji se nalazi na Matlab putanji!!

Da bismo otvorili već postojeću m-datoteku iz Komandnog prozora, napisaćemo sledeću naredbu:

```
>> edit proba
```

Program se može pokrenuti iz samog Editora, a može i iz Komandnog prozora ukucavanjem imena M-datoteke:

```
>>proba
```

Ukoliko u programu postoje neke greške, u Komandnom prozoru će se prikazati upozorenje ili poruka o grešci. Matlab Debugger nam omogućava da pregledamo delove programa u toku rada programa. Možemo prekinuti izvršenje programa u bilo kojoj liniji programa, a zatim nastaviti izvršavanje od te tačke, prolazeći kroz kod, liniju po liniju, pregledajući rezultate svake izvršene operacije. Tačke prekida u izvršavanju programa se postave na željena mesta u programu, a zatim se program pokrene, nakon čega Matlab otvara prozor Editor/Debugger, u kome zelena strelica pokazuje na sledeću liniju koja se izvršava.

Da bismo kontrolisali tok izvršavanja naredbi, koristimo četiri mogućnosti za pisanje programa koje nam Matlab obezbeđuje, a to su sledeće petlje:

- for
- while
- if-else-end
- switch-case

VRSTE M-DATOTEKA

Postoje dve vrste m-datoteka:

1. *Skripte*

2. *Funkcije*

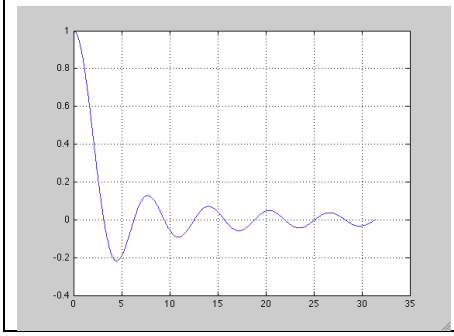
Glavna razlika između skripti i funkcija je u tome što funkcije počinju sa ključnom reči *function*, dok je sadržaj Matlab skripte identičan nizu naredbi koje ručno unosimo u interaktivnom modu rada. Takođe, važna razlika između skripte i funkcije je domen promenljivih. Sve promenljive u Matlab funkciji postoje samo za vreme izvođenja funkcije (osim ako promenljiva nije deklarisana kao globalna), dok sve promenljive u skripti postoje i dostupne su tokom ineraktivnog korisničkog rada.

Skripte

Skripta je najjednostavnija vrsta M-datoteke, jer nema ulazne argumente i ne vraćaju izlazne argumente. Matlab skripte su korisne za automatizaciju niza koraka koje želimo da izvršimo više puta. Skripte čuvaju promenljive u Workspace-u koji dele sa drugim skriptama i sa interfejsom Matlab komandne linije. U skriptama se može raditi sa postojećim promenljivima iz Workspace-a, a formiraju se i novi podaci koji se čuvaju u Workspace-u.

Primer skripte *proba.m*:

```
% Skripta proba
x=pi/100:pi/100:10*pi;
y=sin(x)./x;
plot(x,y)
grid
```



Prvi red u skripti *proba.m* počinje sa '%' što označava komentar. Komentari (linije koda koje se ne izvršavaju) se u skriptovima i funkcijama navode iza znaka za procenat '%'. Bilo koji unos nakon ovog znaka u istoj linije biva ignorisan od strane interpretera, izuzev ako se '%' pojavljuje u okviru nekog stringa (znakovnog niza). Komentari se dodaju da bi se olakšalo tumačenje programskog koda. U

naredna dva reda formiraju se nizovi *x* i *y*. Niz *x* sadrži 1000 brojeva iz opsega *[π/100,10π]* sa korakom *π/100*, dok y sadrži vrednosti funkcije *sinc=sin(x)/x*. Naredba plot formira grafik funkcije *sinc*, a grid dodaje mrežu grafiku.

Funkcije

Funkcija kao vrsta m-datoteke, za razliku od skripte, prihvata ulazne argumente i vraća izlazne argumente i čuva svoje promenljive u internom Workspace-u.

Primer funkcije *descsort.m*:

```
function [b,j]=descsort(a)

% Funkcija descsort uredjuje realan niz a u opadajucem redosledu
% Drugi izlazni parametar j sadrzi permutaciju niza b od niza a

[b,j]=sort(-a);
b=-b;
```

Ova funkcija uzima jedan ulazni parametar – niz realnih brojava, a vraća uređeni niz zajedno sa informacijom o broju izvršenih permutacija. Korišćena je ugrađena funkcija *sort.*

Definišimo niz *a* i pozovimo funkciju iz Komandnog prozora navodeći ime funkcije i navodeći vrednosti ulaznih argumenata u zagradi:

```
>> a=[3 2 7 5 0]
a =
3    2    7    5    0
>> [b,j]=descsort(a)
b =
7    5    3    2    0
j =

3    4    1    2    5
```

Funkciju *descsort.m* možemo pozvati i bez navođenja izlaznih parametara, ali tada će informacija o permutaciji biti izgubljena:

```
>> descsort(a)

ans =

7    5    3    2    0
```

DELOVI FUNKCIJE

U funkciji razlikujemo sledeće delove:

1. Linija za definisanje funkcije

U ovom delu funkcije definiše se ime funkcije, broj i redosled ulaznih i izlaznih argumenata. Zaglavlje počinje propisanom rečju *''function''*, iza koje sledi ime izlazne promenljive ili niza promenljivih koje funkcija vraća kao rezultat. Ukoliko funkcija ima veći broj izlaznih argumenata, njih treba staviti u uglaste zagrade:

```
function [a,b,c]= primer(d,e,f)
```

Ukoliko funkcija nema izlaznih argumenata, ne pišemo ništa ili stavljamo prazne uglaste zagrade []:

```
function = printresults(x)
function [] = printresults(x)
```

Iza imena izlazne promenljive sledi znak jednakosti i ime funkcije (isto kao ime datoteke u kojoj se čuva funkcija), a zatim u okruglim zagradama sledi popis ulaznih promenljivih (jedne ili više) koje će služiti kao argumenti funkcije kod njenog poziva. Promenljive koje učitavamo kao ulazne argumente funkcije ne moraju imati isto ime kao promenljive u liniji za definisanje funkcije.

Ime funkcije mora počinjati sa slovom. Da bismo proverili da li je ime validno, koristimo sledeću naredbu:

```
>> isvarname proba
ans =
1

>> isvarname 8th_column

ans =

0
```

Funkcija *isvarname* vraća 1 ukoliko je ime validno, kao u prvom slučaju, dok je u drugom slučaju rezultat 0, jer ime funkcije počinje sa brojem.

2. Linija H1

Deo funkcije u kojoj se opisuje program, koji se prikazuje kada potražite *help* o celom direktorijumu ili kada koristite naredbu *lookfor*.

3. HELP - tekst

Deo u kome se detaljnije opisuje program, koji se prikazuje zajedno sa linijom H1 kada potražite *help* o određenoj funkciji.

4. Telo funkcije

Programski kod koji izvršava određena izračunavanja i koji dodeljuje vrednosti izlaznim argumentima.

5. Komentar

Komentar počinje znakom %. Tekst u telu programa u kojem se objašnjavaju koraci u programu.

Ime datoteke koja sadrži funkciju mora imati isto ime kao funkcija i ekstenziju .m.

TIPOVI FUNKCIJA

1. Glavne funkcije – prva funkcija u m-datoteci koja uglavnom sadrži glavni program.

2. Podfunkcije – ponašaju se kao potprogrami glavne funkcije. Mogu se koristiti za definisanje višestrukih funkcija u jednoj m-datoteci.

3. Ugnježdene funkcije – funkcije koje su definisane u okviru neke druge funkcije. Mogu poboljšati čitljivost programa i omogućiti fleksibilniji pristup promenljivima.

4. ''Privatne funkcije'' – funkcije sa ograničenim pristupom.

3.2. Funkcije u Matlabu

Funkcije se prema svom poreklu mogu podeliti u tri grupe:

1. UGRAĐENE FUNKCIJE
2. FUNKCIJE U TOOLBOX-OVIMA
3. FUNKCIJE KOJE DEFINIŠE KORISNIK

Mnoge funkcije koje Matlab obezbeđuje su implementirane u vidu m-datoteke. U tu grupu spadaju funkcije koje se nalaze u Toolbox-ovima, kao i funkcije koje definiše korisnik. U prethodnom poglavlju smo videli kako se definišu te funkcije.

Toolbox-ovi predstavljaju kolekciju izabranih funkcija (m-datoteka) namenjenih rešavanju problema iz određene oblasti. Postoje Toolbox-ovi iz sledećih oblasti:

- analiza signala,
- automatsko upravljanje,
- simulacija dinamičkog ponašanja sistema,
- identifikacija sistema,
- veštačke neuronske mreže,
- optimizacije,
- analiza robusnosti sistema...

Ako želimo da utvrdimo da li je funkcija implementirana u vidu m-datoteke ili je u pitanju ugrađena funkcija, možemo koristiti funkciju *exist*, koja vraća broj 2 ako funkcija postoji u vidu m-datoteke, odnosno broj 5 ukoliko se radi o ugrađenoj funkciji.

```
>> exist bessel

ans =
2

>> exist abs

ans =

5
```

Prednost funkcija koje su implementirane u vidu m-datoteke je u tome što možemo pristupiti izvornom kodu, što nam olakšava razumevanje funkcije. Za razliku od sadržaja implementiranih funkcija koji možemo videti, sadržaju ugrađenih funkcija se ne može pristupiti.

Poreklo funkcije se može odrediti i korišćenjem naredbe *which*:

```
>> which abs
built-in  (/Applications/MATLAB_R2009a/MATLAB_R2009b.app/toolbox/matlab/elfun/@double/abs)
% double method

>> which bessel

/Applications/MATLAB_R2009a/MATLAB_R2009b.app/toolbox/matlab/specfun/bessel.m
```

Ako je funkcija ugrađena, Matlab to i napiše, a pored toga napiše i mesto na disku gde je funkcija smeštena.

Matlab ima veliki broj standardnih elementarnih matematičkih funkcija, uključujući *abs, sqrt, exp i sin*, kao i veliki broj složenih matematičkih funkcija kao što su *Bessel-ove* funkcije. Broj i vrsta raspoloživih funkcija u Matlab-u zavisi od broja instaliranih Toolbox-ova. Neke od funkcija, kao što su *sqrt* i *sin*, su već ugrađene u Matlab. One su veoma efikasne, ali detalji implementacije izračunavanja

nisu lako dostupni. Druge funkcije, kao što je *sinh*, implementirane su kao m-datoteke. Njihov kod je vidljiv i čak može da se menja po potrebi.

3.2.1. Elementarne matematičke funkcije

Elementarne matematičke funkcije definisane su u oblasti kompleksnih brojeva i rezultate daju takođe u skupu kompleksnih brojeva. Argumenti ovih funkcija mogu biti tipa skalara, vektora i matrica, a rezultat je promenljiva istog tipa i dimenzije kao i ulazni argument. Prema tome, primena funkcije na matricu daće kao rezultat matricu istih dimenzija kao što je matrica argumenta čiji su elementi rezultat primene funkcije na pojedini element matrice argumenta.

Ako želite da vidite listu elementarnih matematičkih funkcija, navedite u komandnoj liniji sledeće:

```
>> help elfun
Elementary math functions.
```

Dobićete sledeći spisak ELEMENTARNIH MATEMATIČKIH FUNKCIJA:

```
Trigonometric.
sin       - Sine.
sind      - Sine of argument in degrees.
sinh      - Hyperbolic sine.
asin      - Inverse sine.
asind     - Inverse sine, result in degrees.
asinh     - Inverse hyperbolic sine.
cos       - Cosine.
cosd      - Cosine of argument in degrees.
cosh      - Hyperbolic cosine.
acos      - Inverse cosine.
acosd     - Inverse cosine, result in degrees.
acosh     - Inverse hyperbolic cosine.
tan       - Tangent.
tand      - Tangent of argument in degrees.
tanh      - Hyperbolic tangent.
atan      - Inverse tangent.
atand     - Inverse tangent, result in degrees.
atan2     - Four quadrant inverse tangent.
```

atanh - Inverse hyperbolic tangent.

sec - Secant.

secd - Secant of argument in degrees.

sech - Hyperbolic secant.

asec - Inverse secant.

asecd - Inverse secant, result in degrees.

asech - Inverse hyperbolic secant.

csc - Cosecant.

cscd - Cosecant of argument in degrees.

csch - Hyperbolic cosecant.

acsc - Inverse cosecant.

acscd - Inverse cosecant, result in degrees.

acsch - Inverse hyperbolic cosecant.

cot - Cotangent.

cotd - Cotangent of argument in degrees.

coth - Hyperbolic cotangent.

acot - Inverse cotangent.

acotd - Inverse cotangent, result in degrees.

acoth - Inverse hyperbolic cotangent.

hypot - Square root of sum of squares.

Exponential.

exp - Exponential.

expm1 - Compute exp(x)-1 accurately.

log - Natural logarithm.

log1p - Compute log(1+x) accurately.

log10 - Common (base 10) logarithm.

log2 - Base 2 logarithm and dissect floating point number.

pow2 - Base 2 power and scale floating point number.

realpow - Power that will error out on complex result.

reallog - Natural logarithm of real number.

realsqrt - Square root of number greater than or equal to zero.

sqrt - Square root.

nthroot - Real n-th root of real numbers.

nextpow2 - Next higher power of 2.

Complex.

abs - Absolute value.

angle - Phase angle.

complex - Construct complex data from real and imaginary parts.

conj - Complex conjugate.

imag - Complex imaginary part.

real - Complex real part.

unwrap - Unwrap phase angle.

isreal - True for real array.

cplxpair - Sort numbers into complex conjugate pairs.

Rounding and remainder.

fix - Round towards zero.

floor - Round towards minus infinity.

ceil - Round towards plus infinity.

round - Round towards nearest integer.

mod - Modulus (signed remainder after division).

rem - Remainder after division.

sign - Signum.

3.2.2. Specijalne matematičke funkcije

Da biste dobili listu specijalnih matematičkih funkcija, napišite sledeću naredbu:

```
>> help specfun
Specialized math functions.
```

Kao rezultat dobićete sledeću listu funkcija:

Specialized math functions.

airy - Airy functions.
besselj - Bessel function of the first kind.
bessely - Bessel function of the second kind.
besselh - Bessel functions of the third kind (Hankel function).
besseli - Modified Bessel function of the first kind.
besselk - Modified Bessel function of the second kind.

beta - Beta function.
betainc - Incomplete beta function.
betaln - Logarithm of beta function.
ellipj - Jacobi elliptic functions.
ellipke - Complete elliptic integral.
erf - Error function.
erfc - Complementary error function.
erfcx - Scaled complementary error function.
erfinv - Inverse error function.
expint - Exponential integral function.
gamma - Gamma function.
gammainc - Incomplete gamma function.
gammaln - Logarithm of gamma function.
psi - Psi (polygamma) function.
legendre - Associated Legendre function.
cross - Vector cross product.
dot - Vector dot product.

Number theoretic functions.

factor - Prime factors.
isprime - True for prime numbers.
primes - Generate list of prime numbers.
gcd - Greatest common divisor.
lcm - Least common multiple.
rat - Rational approximation.
rats - Rational output.
perms - All possible permutations.
nchoosek - All combinations of N elements taken K at a time.
factorial - Factorial function.

Coordinate transforms.

cart2sph - Transform Cartesian to spherical coordinates.
cart2pol - Transform Cartesian to polar coordinates.
pol2cart - Transform polar to Cartesian coordinates.
sph2cart - Transform spherical to Cartesian coordinates.
hsv2rgb - Convert hue-saturation-value colors to red-green-blue.
rgb2hsv - Convert red-green-blue colors to hue-saturation-value.

3.2.3. Funkcije za obradu vektora i matrica

Grupu funkcija za obradu vektora i matrica čine funkcije čiji su argumenti vektori ili matrice i koje kao rezultat daju logičku promenljivu ili vektor ili matricu određenih dimenzija. To su funkcije za formiranje matrica, relacijske i logičke funkcije nad matricama, funkcije za određivanje različitih veličina karakterističnih za matrice i vektore.

Da biste dobili listu funkcija za rad sa matricama, ukucajte sledeću naredbu:

>> help elmat

Elementary matrices and matrix manipulation.

Dobićete sledeću listu funkcija:

Elementary matrices.

zeros - Zeros array.

ones - Ones array.

eye - Identity matrix.

repmat - Replicate and tile array.

linspace - Linearly spaced vector.

logspace - Logarithmically spaced vector.

freqspace - Frequency spacing for frequency response.

meshgrid - X and Y arrays for 3-D plots.

accumarray - Construct an array with accumulation.

: - Regularly spaced vector and index into matrix.

Basic array information.

size - Size of array.

length - Length of vector.

ndims - Number of dimensions.

numel - Number of elements.

disp - Display matrix or text.

isempty - True for empty array.

isequal - True if arrays are numerically equal.

isequalwithequalnans - True if arrays are numerically equal.

Matrix manipulation.

cat - Concatenate arrays.

reshape - Reshape array.

diag - Diagonal matrices and diagonals of matrix.

blkdiag - Block diagonal concatenation.

tril - Extract lower triangular part.

triu - Extract upper triangular part.

fliplr - Flip matrix in left/right direction.

flipud - Flip matrix in up/down direction.

flipdim - Flip matrix along specified dimension.

rot90 - Rotate matrix 90 degrees.

: - Regularly spaced vector and index into matrix.

find - Find indices of nonzero elements.

end - Last index.

sub2ind - Linear index from multiple subscripts.

ind2sub - Multiple subscripts from linear index.

bsxfun - Binary singleton expansion function.

Multi-dimensional array functions.

ndgrid - Generate arrays for N-D functions and interpolation.

permute - Permute array dimensions.

ipermute - Inverse permute array dimensions.

shiftdim - Shift dimensions.

circshift - Shift array circularly.

squeeze - Remove singleton dimensions.

Array utility functions.

isscalar - True for scalar.

isvector - True for vector.

Special variables and constants.

ans - Most recent answer.

eps - Floating point relative accuracy.

realmax - Largest positive floating point number.

realmin - Smallest positive floating point number.

pi - 3.1415926535897....

i - Imaginary unit.

inf - Infinity.

nan - Not-a-Number.

isnan - True for Not-a-Number.

isinf - True for infinite elements.

isfinite - True for finite elements.

j - Imaginary unit.

why - Succinct answer.

Specialized matrices.

compan - Companion matrix.

gallery - Higham test matrices.

hadamard - Hadamard matrix.

hankel - Hankel matrix.

hilb - Hilbert matrix.

invhilb - Inverse Hilbert matrix.

magic - Magic square.

pascal - Pascal matrix.

rosser - Classic symmetric eigenvalue test problem.

toeplitz - Toeplitz matrix.

vander - Vandermonde matrix.

wilkinson - Wilkinson's eigenvalue test matrix.

Funkcije za definisanje matrica

Ove funkcije nam omogućavaju da definišemo matricu dimenzije mxn sa svojstvima određenim funkcijom. Za sve funkcije ovog skupa važi da mogu imati jedan ili dva skalarna argumenta tipa integer. Ukoliko se radi o funkciji sa jednim argumentom, on određuje dimenzije izlazne kvadratne matrice. Ako se funkcija poziva sa dva argumenta, ti argumenti određuju broj vrsta i kolona izlazne matrice. Neke od funkcije za definisanje matrica su: *zeros, ones, eye, rand*, itd.

Relacijske i logičke funkcije za rad sa matricama

Relacijske i logičke funkcije za rad sa matricama ispituju sadržaj matrica koje su date kao ulazni argumenti funkcije i vraćaju logičku promenljivu ili matricu logičkih promenljivih koji imaju vrednost 1 ako je uslov ispunjen, a vrednost 0 ako uslov nije ispunjen. Neke od funkcija koje spadaju u ovu grupu su: *any, all, isnan, isempty*, itd.

Funkcije za obradu vektora

Funkcije za obradu vektora mogu imati za argument vektor ili matricu. Ako je argument vektor, rezultat je skalarna veličina. Ako je argument matrica, rezultat je vektor-vrsta čiji svaki element predstavlja rezultat operacije nad odgovarajućom kolonom matrice. Neke od ovih funkcija su: *min, max, mean, median*, itd.

Funkcije za obradu matrica

Funkcije za obradu matrica kao argument imaju matricu, a kao rezultat daju matricu ili vektor. Neke od funkcija za obradu matrica su: *trace, det, rot90, inv, diag*, itd.

3.2.4. Funkcije za obradu stringova

String je niz ASCII karaktera pridružen nekoj promenljivoj. U Matlab-u string se definiše jednostrukim navodnicima na početku i kraju željenog niza znakova:

```
>> s='Zdravo!'
s =
Zdravo!
```

Matlab sadrži veliki broj fukcija za rad sa stringovima, a listu tih funkcija je moguće videti korišćenjem sledeće naredbe:

```
>> lookfor string
```

Ovo su samo neke od funkcija koje postoje u Matlabu, kao ugrađene ili implementirane u vidu m-datoteke.

3.3. Promenljive u Matlab-u

U Matlab-u mogu se definisati 2 vrste promenljivih:

1. Lokalne

2. Globalne

3.3.1. Lokalne promenljive

Svaka Matlab funkcija ima svoje lokalne promenljive, koje su odvojene od onih promenljivih iz drugih funkcija (osim ugnježdenih) i od promenljivih iz osnovnog Workspace-a. Promenljive koje se definišu u funkciji ne ostaju u memoriji između poziva funkcije, osim ako se ne definišu kao globalne promenljive.

Skripte, za razliku od funkcija, nemaju zaseban Workspace. Promenljive koje se definišu u skripti čuvaju se u osnovnom Workspace-u i dostupne su drugim skriptama.

3.3.2. Globalne promenljive

Ako nekoliko funkcija i osnovni Workspace proglase neku promenljivu globalnom, onda svi dele jednu kopiju te promenljive. Bilo koja dodela toj promenljivoj, u bilo kojoj funkciji, dostupna je svim drugim funkcijama koje su je deklarisale kao globalnu.

Svaka funkcija u kojoj se koristi globalna promenljiva mora prvo da deklariše tu promenljivu kao globalnu. Najčešće se deklaracija vrši na početku.

```
global MAXLEN
```

Ako m-datoteka sadrži podfunkciju, tada svaka funkcija koja želi da pristupi globalnoj promenljivoj , mora da je deklariše kao globalnu. Ukoliko želimo da pristupimo promenljivoj iz komandne linije, moramo deklarisati promenljivu kao globalnu u komandnoj liniji.

Imena Matlab globalnih promenljivih su duža i deskriptivnija od lokalnih promenljivih i često se pišu velikim slovima.

Ako želimo da vidimo samo globalne promenljive, koristimo naredbe *who* i *whos*:

```
>> global MAXDUZINA MAXSIRINA
>> MAXDUZINA=36;
>> MAXSIRINA=12;
>> duzina=7;
>> sirina=3;
>> who global

Your variables are:

MAXDUZINA  MAXSIRINA

>> whos global
Name         Size          Bytes  Class     Attributes

MAXDUZINA     1x1             8  double    global
MAXSIRINA     1x1             8  double    global
```

Pri izboru imena za promenljivu treba voditi računa o tome da to ime nije već korišćeno kao ime funkcije, bilo da je to funkcija u vidu m-datoteke ili ugrađena funkcija. Ukoliko definišemo promenljivu sa imenom neke funkcije, nećemo moći da pozovemo tu funkciju sve dok ne izbrišemo promenljivu iz memorije sa naredbom *clear*. Da bismo proverili da li je neko ime već u upotrebi kao ime funkcije, koristimo naredbu *which*.

```
>> abs=5
abs =
5
>> a=abs(8)
??? Index exceeds matrix dimensions.
```

```
>> clear
>> a=abs(8)
a =
8
>> which abs
built-in  (/Applications/MATLAB_R2009a/MATLAB_R2009b.app/toolbox/matlab/elfun/@double/abs)
% double method
```

Zarad vežbe i daljeg utvrđivanja gradiva proučite M-datoteke dobijene u prilogu: Povrsina1.m, Povsina2.m, kvadform.m, mojminvek.m.

3.4. Naredbe uslovnog grananja: `if` i `switch`

Pri programiranju je često jpotrebno omogućiti grananje programa na osnovu nekih uslova. MATLAB poseduje strukture, slične onima u drugim programskim jezicima, koje ovo omogućavaju.

U sledećem primeru prikazane su osnovne karakteristike korišćenja *if-elseif-else* strukture:

```
if isinf(x)||~isreal(x) % logicki uslov za
                        % izvrsavanje bloka komandi
                        % koji sledi
    disp('Pogresan unos')
    y=NaN

elseif(x==round(x))&&(x>0) % drugo grananje
    y=prod(1:x-1)

else                       % ukoliko nijedan od                          %
prethodnih uslova nije                      % zadovoljen
    y=gamma(x)

end
```

Uslovi izvršavanje *if* bloka mogu uključivati operatore poređenja *==, <, <=, ~=, >, >=* ili funkcije kao što je *isinf*, koje vraćaju logičke vrednosti. Mogu se koristiti i numeričke vrednosti, pri čemu nenulta vrednost ima značenje *true*(istinito). Međutim bolje je, u smislu čitljivosti i pouzdanosti koda, koristiti formu *if x~=0* nego *if x*, iako oba izraza vraćaju istu logičku vrednost, kada je promenljiva x broj. Takođe treba obratiti pažnju u slučaju da se niz (matrica) koriste u konstrukciji uslova za *if* strukturu. Kada uslov nije zadat u formi skalara vraćaće vrednost *true* (istinito) samo u slučaju kada svi elementi datog niza (matrice) zadovoljavaju uslov. Da bi se izbegla zabuna, kada je to moguće, treba koristiti logičke operatore *any* ili *all* koji vrše redukciju logičkih nizova na skalarne vrednosti.

U nekim slučajevima *isequal* komanda pokazuje robusnije ponašanje nego logički operator poređenja *==*. Na primer izraz *isequal(s,'zdravo')* vraća *false* (0) ukoliko *s* nije identično stringu *'zdravo'*, dok će izraz *s=='zdravo'* vraćati grešku ukoliko *s* nije dimenzija *1x5*, tj. ukoliko sadrži kraći ili duži string. U većini slučajeva poželjnije je ponašanje po prvom modelu.

Strukturu *else/if* treba koristiti u slučajevima kada broj alternativa nije velik. U slučaju većeg broja opcija uobičajeno je da se koristi *switch/case* struktura:

```
switch jedinica
    case 'duzina'
        disp('metar')
    case 'zapremina'
        disp('litar')
    case 'vreme'
        disp('sekunda')
    otherwise
        disp('odustajem')
end
```

Switch uslov može biti dat kao string ili kao brojna vrednost. Kada interpreter naiđe na prvi *case* slučaj koji zadovoljava ovaj uslov izvršava se njemu pripadajući blok komandi. Ukoliko je opcija *otherwise* prisutna, ona daje podrazumevan blok komandi koji se izvršava kada nijedan od *case* slučajeva ne zadovoljava postavljeni *switch* uslov.

3.5. Kontrolne petlje: `for` i `while`

Mnogi algoritmi podrazumevaju iteracije, tj. višestruko ponavljanje nekog bloka komandi. Slično drugim programskim jeyicima i u MATLAB-u postoje programske strukture koje ovo omogućavaju: petlje. Na početku ćemo samo dati jednu napomenu: sa obzirom da se slova *i* i *j* najčešće koriste za označavanje brojača u petljama, preporučuje se da se za označavanje imaginarne jedinice $\sqrt{-1}$ koriste *1i* ili *1j*.

Kao primer korišćenja *for* petlje daćemo kod koji računa prvih deset članova čuvenog Fibonačijevog niza:

```
f=[1 1];
for n=3:10
    f(n)=f(n-1)+f(n-2);
end
```

U telu petlje se može nalaziti proizvoljan broj linija koda. U našem primeru vrednost indeksa *n* se menja od 3 do 10, sa korakom 1, pri čemu se sa svakom promenom izvršava i kod u telu petlje. Treba imati na umu da izraz *3:10* nije ništa drugo do vektor vrsta sa svojih osam elemenata. Za kontrolu broja ponavljanja *for* petlje može se zapravo korisiti bilo koji vektor vrsta, a ne samo onaj formiran upotrebom operatora dve tačke. Na primer:

```
x = 1:100; s = 0;
for j = find(isprime(x))
s = s + x(j);
end
```

Ovaj kod računa sumu svih jednostavnih brojeva manjih od 100.

U nekim slučajevima treba ponavljati neki programski blok do zadovoljenja određenog uslova. Ovo se postiže korišćenjem petlje *while*:

```
while abs(x)>1
    x=x/2;
end
```

Sa obzirom da se zadovoljenje uslova proverava pre svakog izvršavanja petlje moguće je imati i slučaj bez ijedne iteracije. U nekim slučajevima pametno je ograničiti broj ponavljanja kako bi se izbeglo formiranje bezkonačne petlje (što bi moglo da se desi u gornjem primeru za *x=inf*). Ovo se može učiniti na nekoliko načina, ali uobičajeno je da se koristi naredba *break*:

```
n=0;
while abs(x)>1
    x=x/2;
    n=n+1;
    if n>50,break,end
end
disp('Prevelik broj iteracija !')
```

Naredba *break* prekida izvršavanje petlje i prebacuje interpreter na prvu naredbu nakon petlje. Dobra progamerska praksa je da se obezbedi neko dijagnostičko obaveštenje u slučaju nasilnog (abnormalnog) izlaska iz petlje.