RAČUNARSKI ALATI - MATLAB

PROGRAMIRANJE

Programiranje u MATLAB-u

- Newton-ov algoritam
  - Nula funkcije f se pod određenim uslovima može naći sledećim iterativnim postupkom:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

  - $\sqrt{a}$ se može naći gornjim postupkom kao nula funkcije $f(x)=x^2-a$, odgovarajuća procedura je:

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{x_n^2 + a}{2x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n} \right)$$

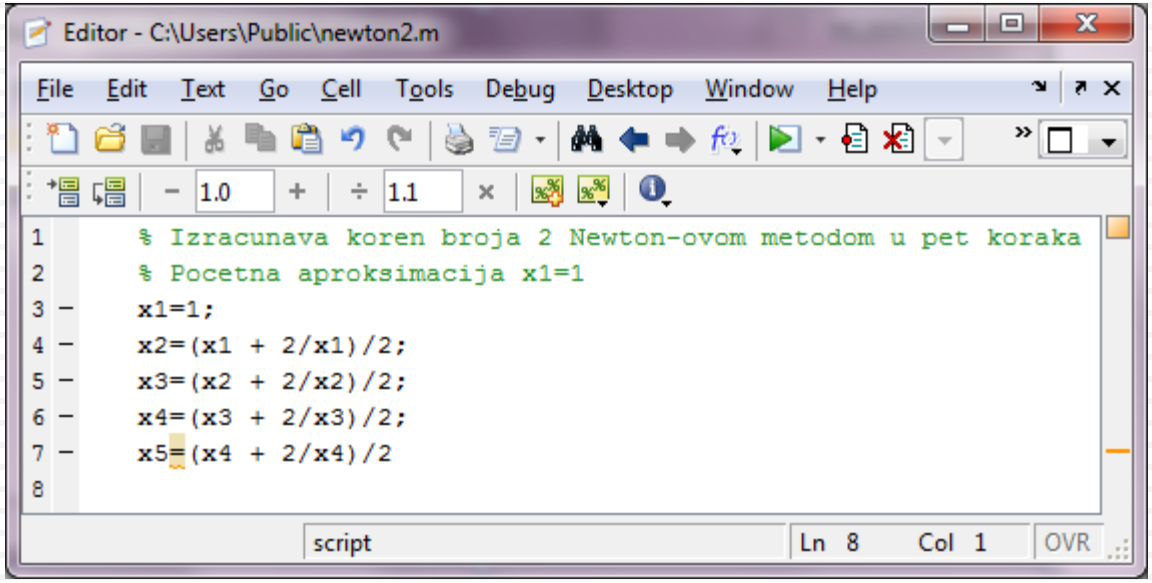
  - Izaberimo $x_1=1$ i nađimo $\sqrt{2}$

Matlab skript datoteka

```
>> x1=1;

>> x2=(x1 + 2/x1)/2
x2 =
   1.5000

>> x3=(x2 + 2/x2)/2
x3 =
   1.4167

>> x4=(x3 + 2/x3)/2
x4 =
   1.4142

>> x5=(x4 + 2/x4)/2
x5 =
   1.4142
```

Ove naredbe možemo kao celinu sačuvati u datoteku (M-datoteka) sa ekstenzijom .m

- Skript datoteka je tekstualna datoteka koju čine naredbe Matlab-a



Matlab skript datoteka

- Skript datoteka se pokreće navođenjem njenog imena u komandnoj liniji (pod uslovom da se skript datoteka nalazi u tekućem direktorijumu)

```
>> newton2
x5 =
    1.4142
>>
```

- Naredbe se ne moraju unositi pri ponovnom izvršavanju

Matlab skript datoteka

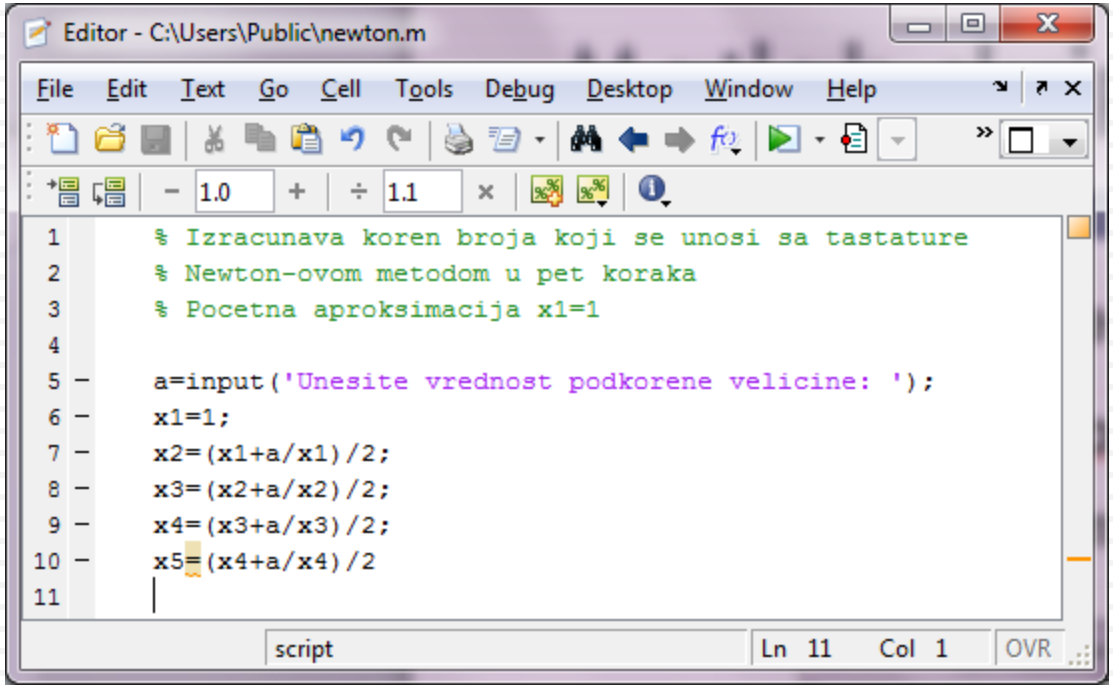
- Izvršava se u Matlab (osnovnom) radnom prostoru
  - Nema svoj radni prostor

```
>> whos
  Name        Size              Bytes  Class     Attributes

  x1          1x1                   8  double
  x2          1x1                   8  double
  x3          1x1                   8  double
  x4          1x1                   8  double
  x5          1x1                   8  double
```

- Nema ulaznih/izlaznih parametara
  - Kako odrediti koren nekog drugog nenegativnog broja?
  - Šta je sve rezultat?

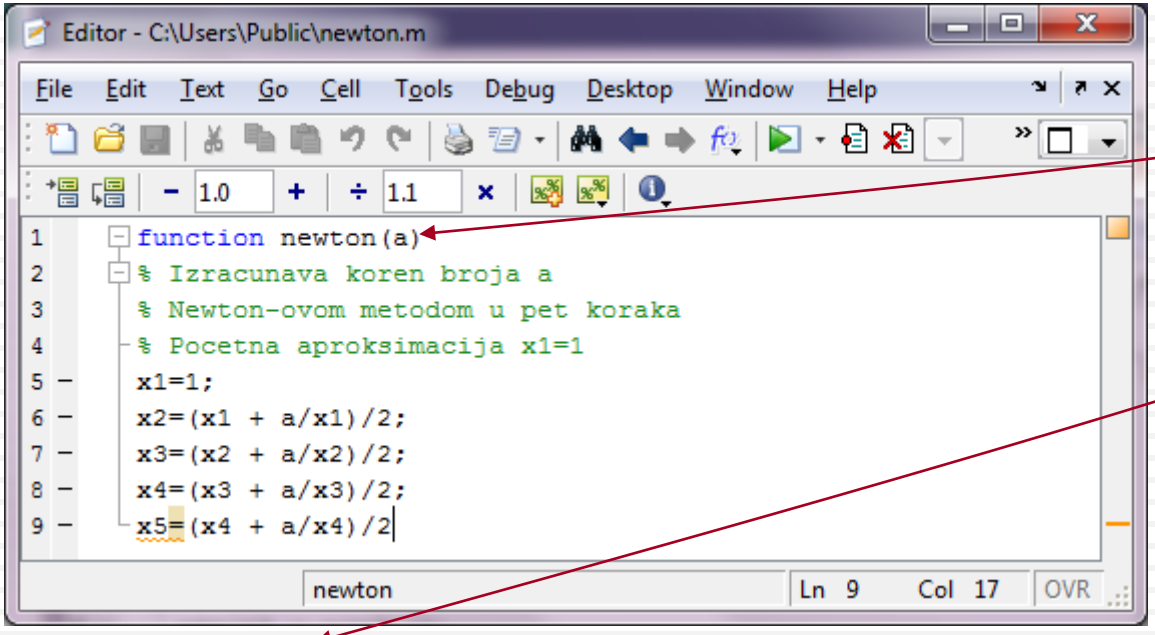
Matlab skript datoteka

- Bilo kakva izmena zahteva promenu skript datoteke
- Parametrizacija skript datoteke se može izvesti input naredbom



Skript datoteka→Funkcija

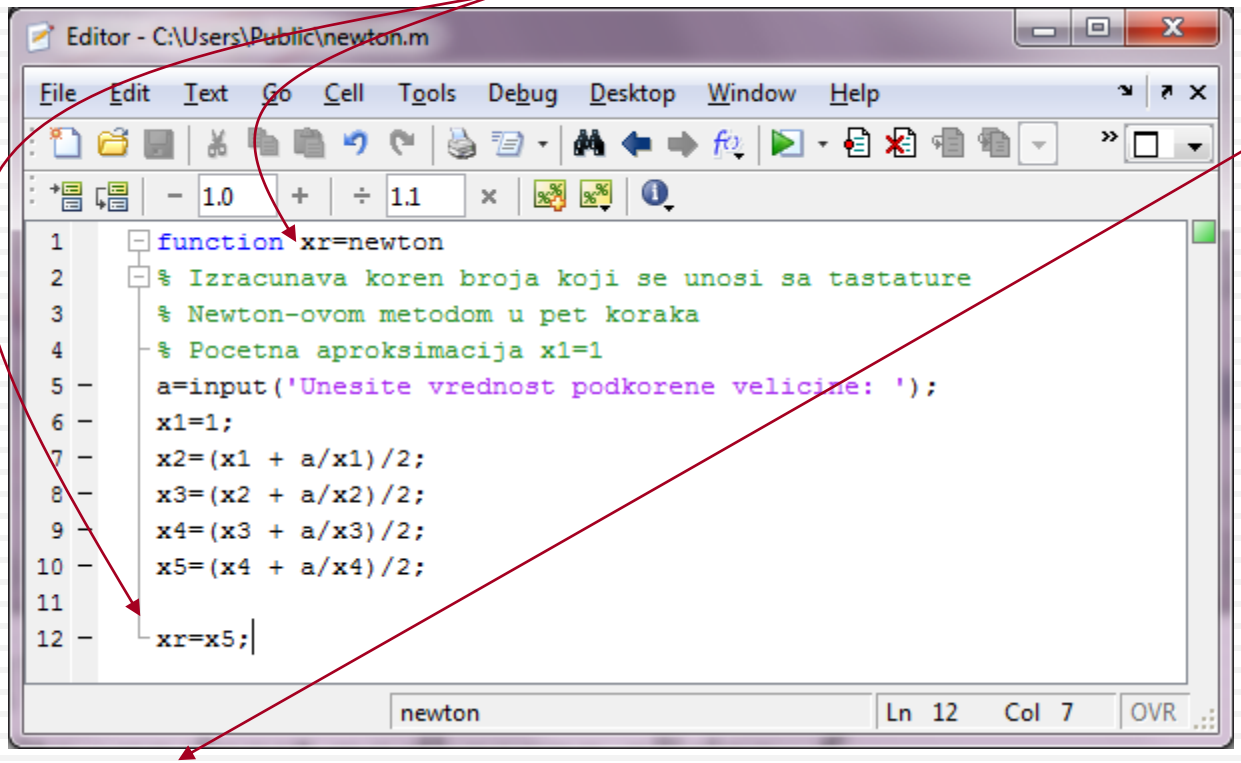
Efikasnije je vrednosti ulaznih parametara zadavati pri startovanju skript datoteke, odnosno ugraditi u poziv funkcije




```
>> newton(5)
x5 =
   2.2361
>>
```

Skript datoteka→Funkcija

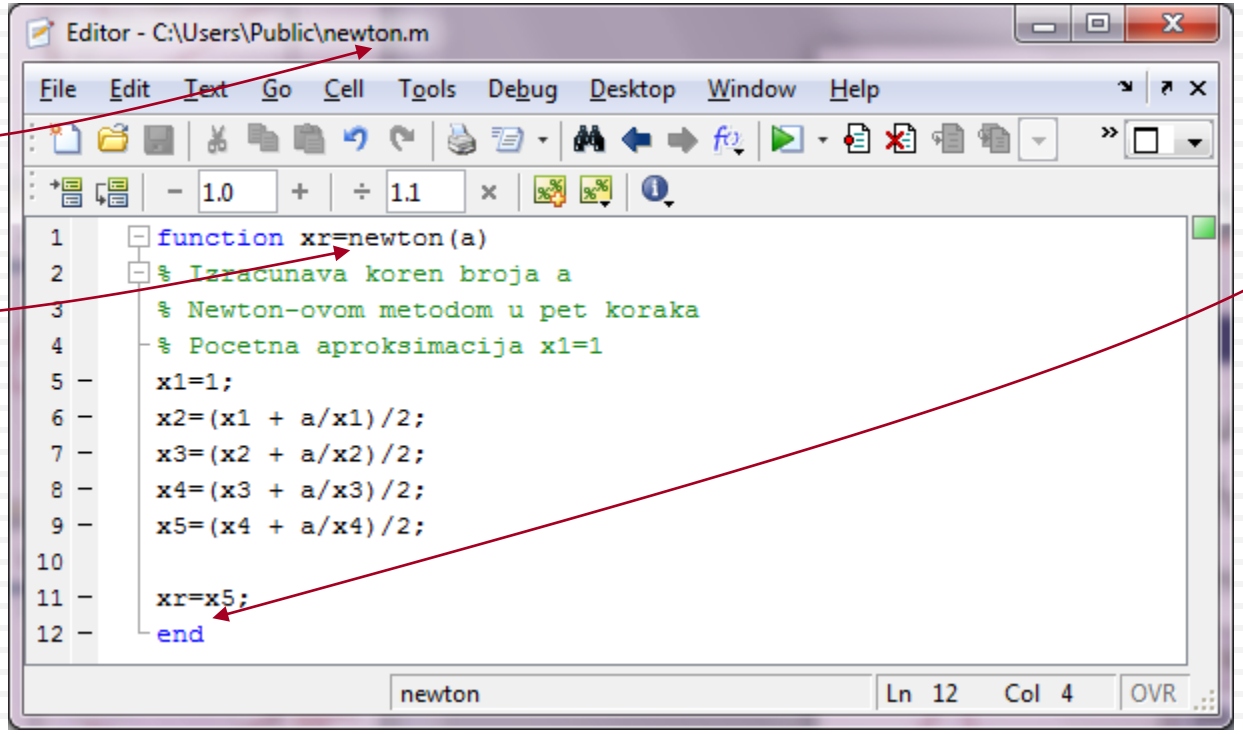
Slično, funkcija koja ima izlazni parametar može biti deo izraza




```
>> (1+newton)/2
Unesite vrednost podkorene velicine: 5
ans =
    1.6180
```

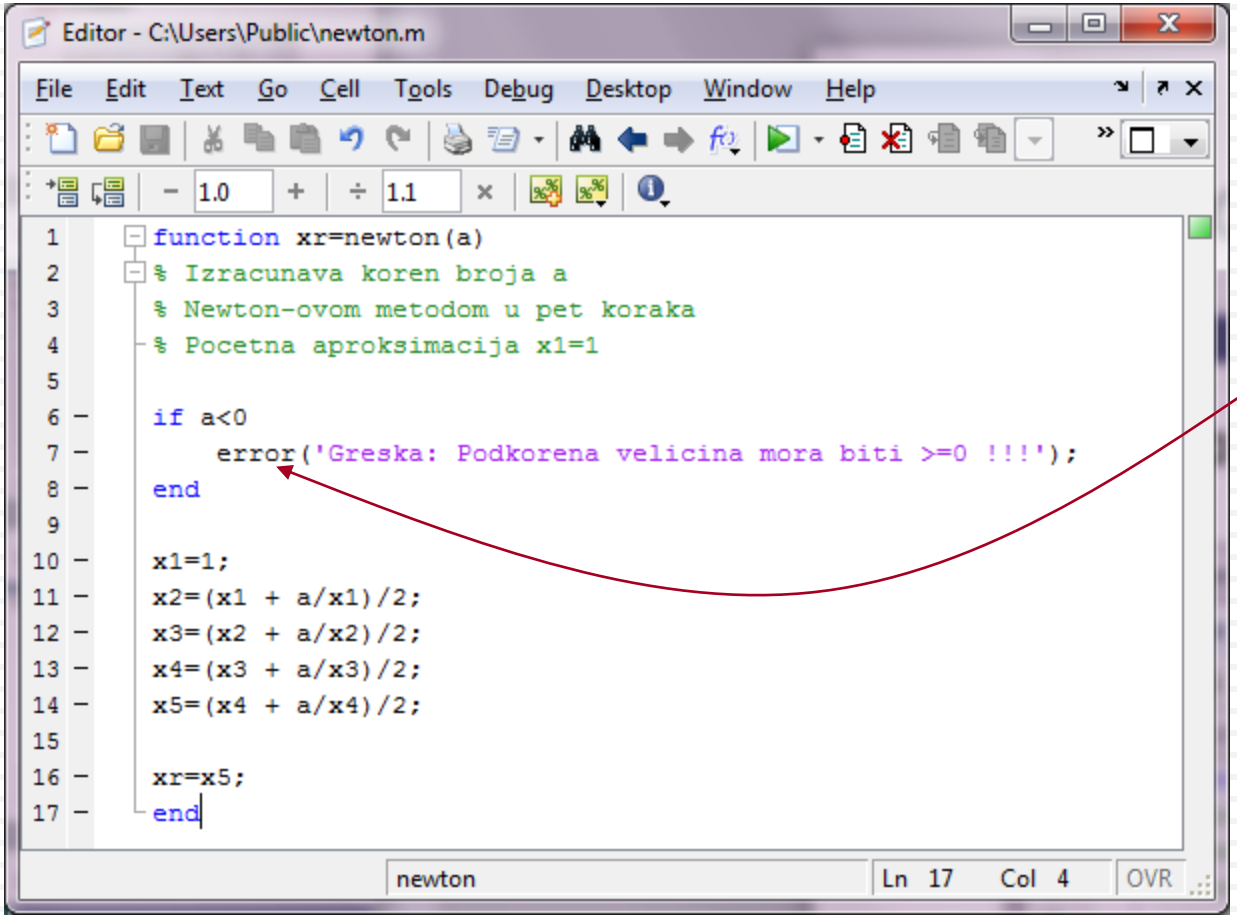
Matlab funkcija

- Preporuka je da
  - Ime funkcije bude isto kao i ime M-datoteke
  - Poslednja naredba funkcije bude end naredba



Naredbe za kontrolu toka

- Testirajmo ulazni parametar *a*



error funkcija prekida izvršavanje i ispisuje poruku o grešci

Naredbe za kontrolu toka

- U iterativnom postupku se ponavlja isto izračunavanje. Formirajmo vektor aproksimacija x

```
function xr=newton(a)
% Izracunava koren broja a
% Newton-ovom metodom u pet koraka
% Pocetna aproksimacija x1=1

if a<0
    error('Greska: Podkorena velicina mora biti >=0 !!!');
end

x(1)=1;
for i=1:4
    x(i+1)=(x(i)+a./x(i))./2;
end

xr=x(5);
end
```

Naredbe za kontrolu toka

- Kako je za izračunavanje sledeće aproksimacije potrebna samo prethodna, dovoljne su dve promenljive:
  - xold : prethodna aproksimacija
  - xnew : naredna aproksimacija (ima ulogu prethodne aproksimacije u sledećoj iteraciji)

```
function xr=newton(a)
% Izracunava koren broja a
% Newton-ovom metodom u pet koraka
% Pocetna aproksimacija x1=1

if a<0
    error('Greska: Podkorena velicina mora biti >=0 !!!');
end

xold=1;
for i=1:4
    xnew=(xold+a./xold)./2;
    xold=xnew; % Nova aproksimacija je stara aproksimacija u sledecoj iteraciji
end

xr=xold; % Poslednja izracunata aproksimacija je smestena u xold
end
```

Naredbe za kontrolu toka

- Nije očigledno koliko iteracija treba izvršiti. Kvalitetnije rešenje je ponavljati postupak dok aproksimacije ne budu dovoljno bliske (abs(xnew-xold)<=1e-6)

```
function xr=newton(a)
% Izracunava koren broja a
% metodom tangente u tri koraka

if a<0 % Nema resenja
    error('Greska: Podkorena velicina a mora biti >=0!');
end

xold=1;
xnew=(xold + a/xold)/2; % Dobijena je nova aproksimacija, pa se mogu uporediti
while abs(xnew-xold) > 1e-6
    xold=xnew; % Nova aproksimacija je stara aproksimacija u sledecoj iteraciji
    xnew=(xold + a/xold)/2;
end

xr=xnew; % Poslednja aproksimacija je smestena u xnew
end
```

Ulazni parametri funkcije

- Ocigledno imamo dve fiksne vrednosti početnu aproksimaciju 1 i tačnost 1e-6, prirodno je ugraditi ih kao ulazne parametre funkcije x_0 i *eps*

```
function xr=newton(a, x0, eps)
% Izracunava koren broja a
% metodom tangente u tri koraka

if a<0 % Nema resenja
    error('Greska: Podkorena velicina a mora biti >=0!');
elseif x0==0 % Prva aproksimacija nije definisana za x0=0 ( x1=(x0+a/x0)/2 -> Inf )
    error('Greska: Pocetna aproksimacija x0 mora biti ~=0!');
end

xold=x0;
xnew=(xold + a/xold)/2; % Dobijena je nova aproksimacija, pa se mogu uporediti
while abs(xnew-xold) > eps
    xold=xnew; % Nova aproksimacija je stara aproksimacija u sledecoj iteraciji
    xnew=(xold + a/xold)/2;
end

xr=xnew; % Poslednja aproksimacija je smestena u xnew
end
```

```
>> phi=(1+newton(5, 1, 1e-6))/2
```

← Poziv

Ulazni parametri funkcije

- Ako želimo da ograničimo broj iteracija uvedimo ulazni parametar *maxiter*

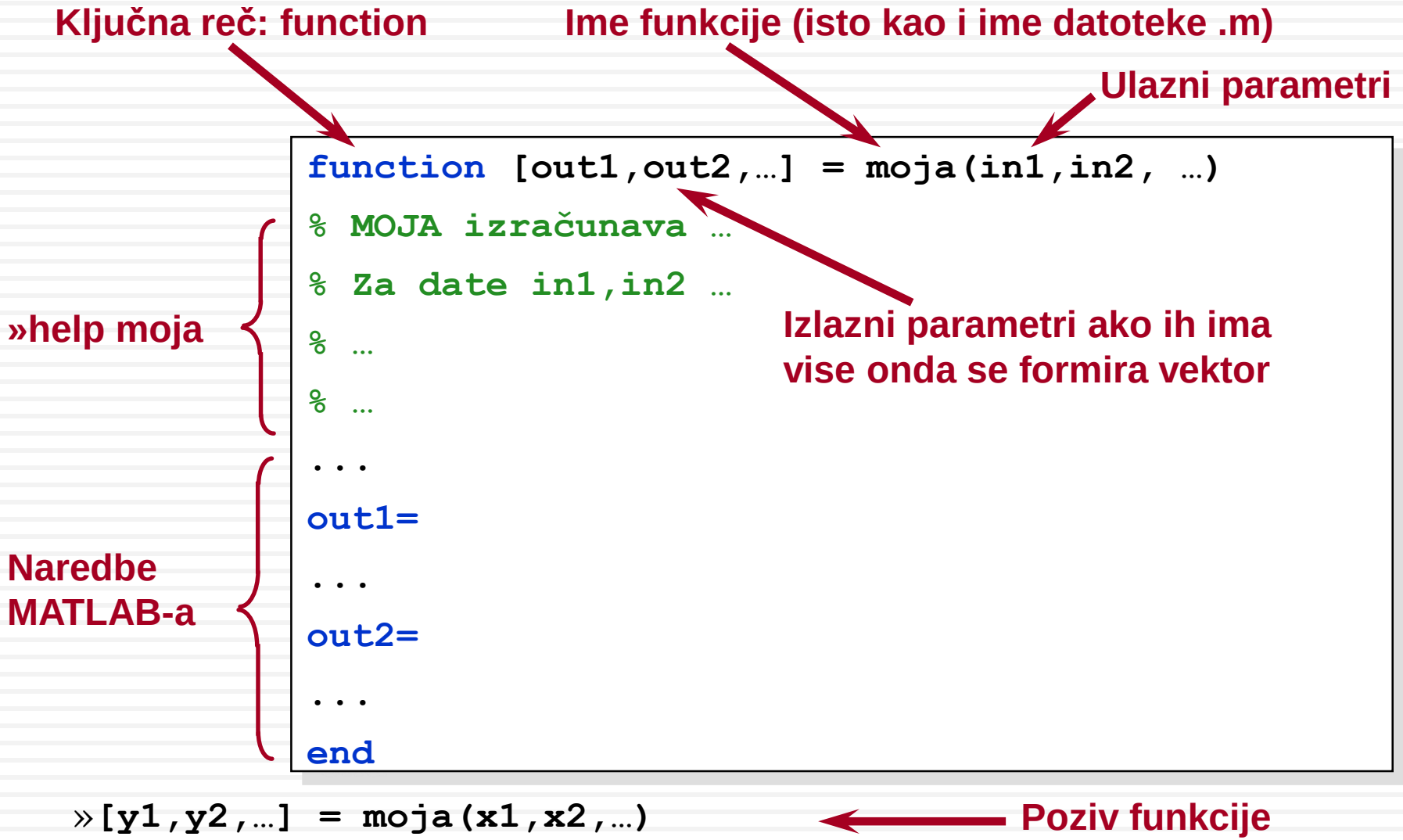
```
function xr=newton(a, x0, eps, maxiter)
% Izracunava koren broja a
% metodom tangente u tri koraka

if a<0 % Nema resenja
    error('Greska: Podkorena velicina a mora biti >=0!');
elseif x0==0 % Prva aproksimacija nije definisana za x0=0 ( x1=(x0+a/x0)/2 -> Inf )
    error('Greska: Pocetna aproksimacija x0 mora biti ~=0!');
end

xold=x0;
xnew=(xold + a/xold)/2; % Dobijena je nova aproksimacija, pa se mogu uporediti
noiter=1; % Izvrsena je prva iteracija
while abs(xnew-xold) > eps

    if noiter >= maxiter % Dostignut je maksimalni broj iteracija, izlaz iz petlje
        break;
    end

    xold=xnew; % Nova aproksimacija je stara aproksimacija u sledecoj iteraciji
    xnew=(xold + a/xold)/2;
end

xr=xnew; % Poslednja aproksimacija je smestena u xnew
end
```

Izlazni parametri funkcije

- Da li je postignuta željena tačnost ili je dostignut maksimalni broj iteracija? Uvedimo izlazni parametar izvršeni broj iteracija *noiter*

```
function [xr, noiter]=newton(a, x0, eps, maxiter)
% Izracunava koren broja a
% metodom tangente u tri koraka

if a<0 % Nema resenja
    error('Greska: Podkorena velicina a mora biti >=0!');
elseif x0==0 % Prva aproksimacija nije definisana za x0=0 ( x1=(x0+a/x0)/2 -> Inf )
    error('Greska: Pocetna aproksimacija x0 mora biti ~=0!');
end

xold=x0;
xnew=(xold + a/xold)/2; % Dobijena je nova aproksimacija, pa se mogu uporediti
noiter=1; % Izvrsena je prva iteracija
while abs(xnew-xold) > eps

    if noiter >= maxiter % Dostignut je maksimalni broj iteracija, izlaz iz petlje
        break;
    end

    xold=xnew; % Nova aproksimacija je stara aproksimacija u sledecoj iteraciji
    xnew=(xold + a/xold)/2;
    noiter=noiter+1;
end

xr=xnew; % Poslednja aproksimacija je smestena u xnew
end
```

Struktura funkcijske M-datoteke



Poziv funkcije sa promenljivim brojem parametara

- Matlab funkcije se mogu pozivati sa različitim brojem ulaznih odnosno izlaznih parametara.

```
>> zeros
ans =
     0
>>
```

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

```
>> zeros(1, 2)
ans =
     0     0
>>
```

nargin, nargout, narginchk, nargoutchk

- U samoj funkciji se može testirati struktura poziva funkcije
- nargin,nargout: Broj ulaznih odnosno izlaznih parametara sa kojim je funkcija pozvana

```
if nargin~=2
    error('Greska: Broj ulaznih parametara je 2 !!!');
end
```

- narginchk(min,max), nargoutchk(min,max): Testira da li je broj ulaznih odnosno izlaznih parametara u datim granicama

Primer

```
function c = addme(a,b)
if nargin==2
  % dati su i a i b
  c = a + b;
elseif nargin==1
  % dato je samo a
  c = a + a;
else
  % vrati 0
  c = 0;
end
end % kraj tela funkcije
```

```
function [dif,absdif] = subtract(y,x)
dif = y - x;
if nargout > 1
  % izračunava se i apsolutna vrednost razlike
  absdif = abs(dif);
end
end % kraj tela funkcije
```

inline i anonimne funkcije

- Funkcije se mogu kreirati i u radnom prostoru (bez .m datoteke), jednom naredbom.

inline funkcija

```
>> f=inline('sin(2.*a+pi/3)')
f =
     Inline function:
     f(a) = sin(2.*a+pi/3)
>> f([1:3])
ans =
    0.0943   -0.9445    0.6918
>>
```

anonimna funkcija

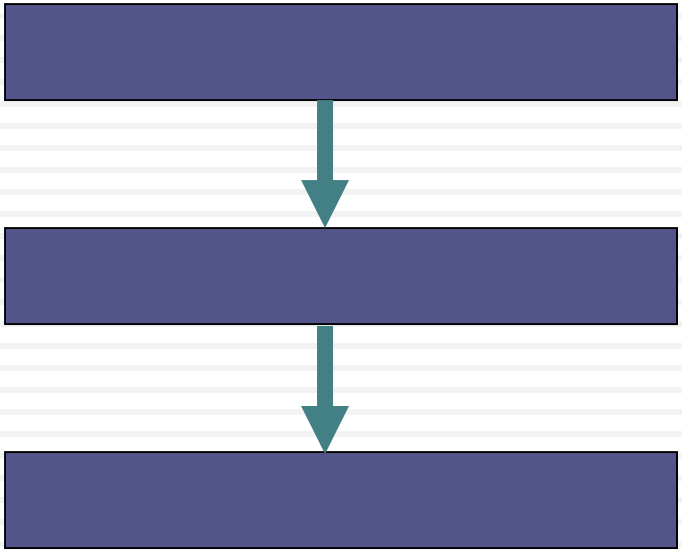
```
>> f=@(u,v) exp(-u).*sin(v)
f =
    @(u,v)exp(-u).*sin(v)
>> f([1:3], 0:pi/4:pi/2)
ans =
         0    0.0957    0.0498
>>
```

Radni prostor

- Matlab (osnovni) radni prostor
  - Promenljive komandne linije i skript datoteka
- Radni prostor funkcije
  - Svaka funkcija ima svoj privatni radni prostor za svoje lokalne promenljive
- Globalni radni prostor
  - Može mu se pristupati iz različitih lokalnih radnih prostora

Tok izvršavanja

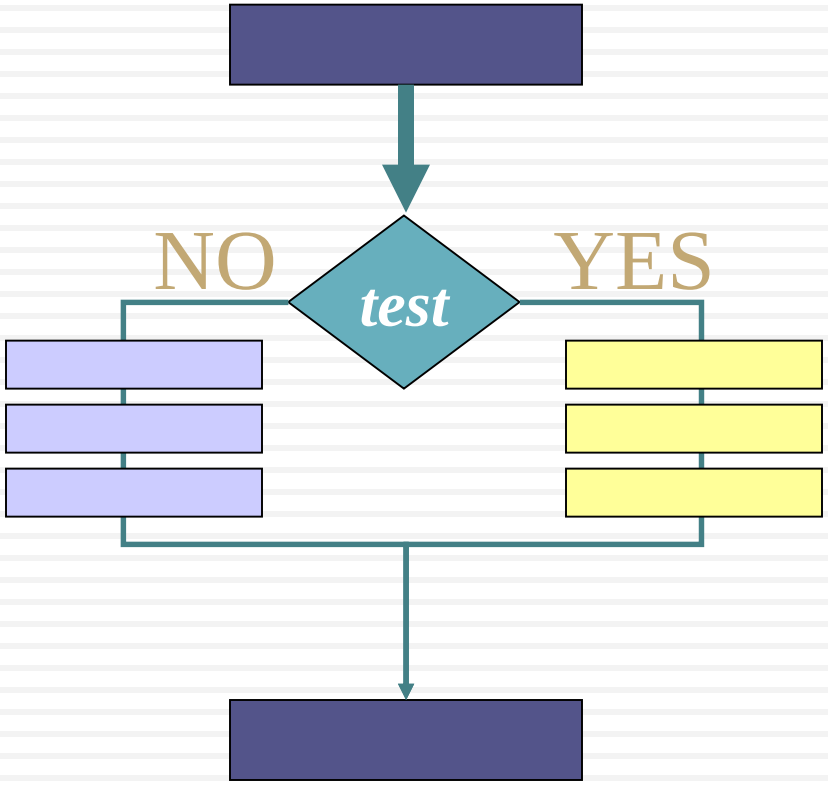
1. **Sekvencijalno izvršavanje**



Naredbe se izvršavaju jedna za drugom prema redosledu navodjenja u programskom tekstu

Tok izvršavanja

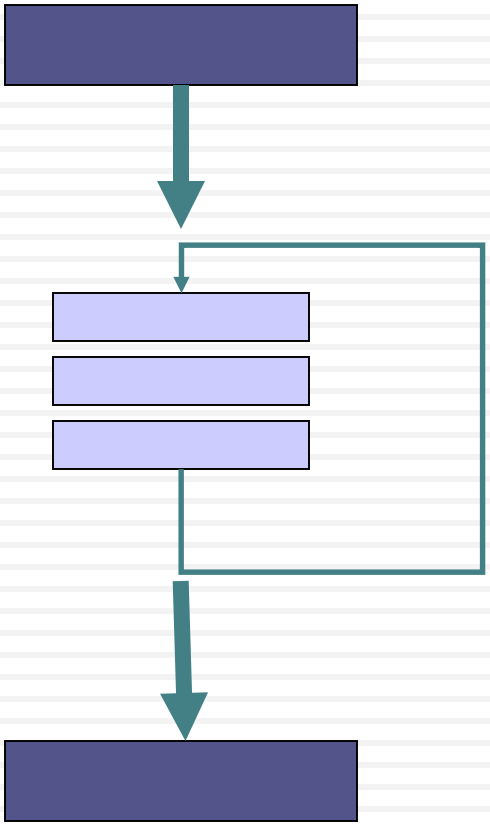
2. Uslovno izvršavanje



U zavisnosti od rezultata testa izvršava se jedan programski blok

Tok izvršavanja

3. Iterativno izršavanje



ponovi…

Nakon što se izvrši programski blok ide se na njegovo ponovno izvršavanje

if naredba

```
if izraz_1
      blok_1
elseif izraz_2
      blok_2
......
elseif izraz_k
      blok_k
else
      blok
end
```

- Sekcije elseif i else nisu obavezne (mogu se izostaviti)
- Redom se izračunavaju logički izrazi: *izraz_1*,*izraz_2*,…
- Izvršava se blok koji odgovara prvom izrazu koji je tačan (ima vrednost ~=0)
- Ako nijedan izraz nije tačan izvršava se blok u else sekciji ako je navedena

Primer: sign(x)

```
x = input('Unesite x: ');
if x<0
  y=-1;
elseif x==0
  y=0;
else
  y=1;
end
disp(['sign(' num2str(x) ')=' num2str(y)]);
```

switch naredba

```
switch izraz
   case vrednost_1
      blok_1
   case vrednost_2
      blok_2
   ......
   case vrednost_k
      blok_k
   otherwise
      blok
end
```

- Vrednost *izraza* se upoređuje redom sa vrednostima ponuđenim u case sekcijama *vrednost_1*, *vrednost_2*,…
- Izvršava se *blok* koji odgovara prvoj case sekciji u kojoj je došlo do poklapanja vrednosti
- Ako nema poklapanja vrednosti izvršava se *blok* u otherwise sekciji
- Više vrednosti se može pridružiti case sekciji ako se navedu u {}

Primer: broj dana u mesecu (februar: 28 dana)

```
mesec = input('Unesite naziv meseca: ', 's');
switch lower(mesec)
  case {'januar', 'mart', 'maj', 'jul', 'avgust', 'oktobar', 'decembar'}
      dana = 31;
  case 'februar'
      dana = 28;
  case {'april', 'jun', 'septembar', 'novembar'}
      dana = 30;
otherwise
      dana = NaN; disp(['Nepostojeci mesec:' mesec]);
end
disp([mesec ': ' num2str(dana) ' dana']);
```

Petlje – for naredba

for promenljiva=niz_vrednosti

blok

end

- *promenljiva* uzima prvu vrednost iz *niza vrednosti* zatim se izvršava *blok* naredbi. Postupak se nastavlja dok se ne iscrpe sve vrednosti iz *niza vrednosti*.

```
f=1;
for i=2:5
  f=f*i;
end
f
```

Korak u for petlji ne mora biti ravnomeran

```
a=rand(1,20);
suma=0;
for i=[1,4,9,11,18]
  suma=suma+a(i);
end
suma
```

Petlje – while naredba

while izraz

blok

end

- Sve dok je logički *izraz* tačan (ima vrednost ~=0) izvršava se *blok* naredbi.

```
suma=0;
x=input('Unesite vrednost (0 kraj unosa): ');
while x~=0
  suma=suma+x;
  x=input('Unesite vrednost (0 kraj unosa): ');
end
suma
```

break, continue

- Upravljaju izršavanjem petlji
- break
  - Prekida izvršavanje tekuće petlje
- continue
  - Prekida izvršavanje tekuće iteracije i prelazi se na novu iteraciju

Primer: suma pozitivnih vrednosti

```
sumPoz=0;
while 1 % uvek ispunjen uslov – beskonacna petlja
  x=input('Unesite vrednost (0 kraj unosa): ');
  if x==0
    break; % prekida se izvršavanje petlje
  end
  if x<0 % uneta je negativna vrednost ponovni unos
    continue; % predji na novu iteraciju
  end
  sumPoz=sumPoz+x;
end
sumPoz
```

Vektorizacija

- Sve Matlab funkcije su optimizovane za vektorske operacije. Samim tim vektorizovani kod se značajno brže izvršava
- Vektorizacija se odnosi na način izvršavanja takav da se operacija simultano izvršava na nizu vrednosti a ne pojedinačno na svakom elementu niza
- Na primer y=sin(1:100) je vektorizovana naredba za razliku od naredbe y1=sin(1),y2=sin(2), …

Petlje u Matlab-u

Brisanje radnog prostora

```
% U for petlji formirajmo niz korena brojeva od 1 do 1000
clear;
tic;
for i=1:1000
   b(i)=sqrt(i);
end
t=toc;
disp(['Vreme utroseno for petljom je:',num2str(t)]);
```

Startovanje tajmera

Zaustavljanje tajmera i očitavanje proteklog vremena u sekundama

Vreme utroseno for petljom je: 0.00053356

Petlje u Matlab-u

Korišćenje vektorskih operacija umesto petlji značajno ubrzava izvršavanje

Brisanje radnog prostora

```
% Formirajmo niz korena brojeva od 1 do 1000 koristeći
% vektorizovanu korenu funkciju sqrt
clear;
tic;
b=sqrt(1:1000);
t=toc;
disp(['Vreme utroseno vektorizovanom funkcijom je: ',num2str(t)]);
```

Startovanje tajmera

Zaustavljanje tajmera i očitavanje proteklog vremena u sekundama

Vreme utroseno vektorizovanom funkcijom je: 7.5784e-05