MATLAB

Matlab je interaktiven sistem za matematično modeliranje. Kratica MATLAB pomeni **MATrixLABoratory**. Vsi objekti v Matlabu so matrike. Če ima matrika eno samo vrstico in en sam stolpec, je to skalar. Vektorji so matrike z eno samo vrstico oz. enim samim stolpcem.

Matlab vedno računa numerično. Vse izraze, ki jih natipkamo, Matlab izračuna in interpretira. Matlab loči male in velike črke. Brez dodatnega paketa za simbolično računanje Matlab ne zna računati s simboli. Če na primer vtipkamo spremenljivko, ki ji še nismo določili vrednosti, potem Matlab izpiše, da spremenljivke ne pozna.

```
>> a
??? Undefined function or variable 'a'.
>>
```

Spremenljivki lahko priredimo vrednost z ukazom:

```
>> a=5
a =
    5
>>
```

Če nočemo, da se ovrednoten izraz izpiše dodamo ;

```
>> a=5;
>>
```

Če imamo predolg ukaz, ga lahko nadaljujemo v naslednji vrstici, če zapišemo ...

```
>> 1+2+3+4+5+6+7+8+9+10+...
11+12+13+14+15
ans =
   120
>>
```

Vidimo, da v primeru, ko rezultata ne priredimo nobeni spremenljivki, le tega Matlab shrani v ans.

Matrike vpisujemo v oklate oklepaje, pri čemer ločimo elemente s presledki ali vejicami. S podpičjem nakažemo skok v novo vrstico.

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
    1    2    3
    4    5    6
    7    8    9
>> B=[-1, 4, -3; 6, 5, 3; 1, 0, 5]
```

```
B =
   -1    4   -3
    6    5    3
    1    0    5
>>
```

Vpisovanje vektorjev:

```
>> x=[1 2 3 6]
x =
    1    2    3    6
>> y=[6; 3; 2; -1]
y =
    6
    3
    2
   -1
>>
```

Osnovne matrične operacije:

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> B=[-1, 4, -3; 6, 5, 3; 1, 0, 5];
>> A+B
ans =
    0    6    0
   10   10    9
    8    8   14
>> A-B
ans =
    2   -2    6
   -2    0    3
    6    8    4
>> A*B
ans =
   14   14   18
   32   41   33
   50   68   48
>> A^3
ans =
      468       576       684
     1062      1305      1548
     1656      2034      2412
>>
```

Osnovne računske operacije lahko izvajamo na matrikah tudi po komponentah, če pred operacijo zapišemo piko.

```
>> A.*B
ans =
   -1    8   -9
   24   25   18
    7    0   45
>> A.^2
ans =
    1    4    9
   16   25   36
   49   64   81
>>
```

Transponiranje matrik:

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

Levo deljenje \ ustreza reševanju linearnega sistema enačb A*x=b. In sicer je rešitev x=A\b.

```
>> A=[1 -2 0; 4 -5 1; 7 8 9];
>> b=[-1,4,-5]';
>> x=A\b
x =
   27.0000
   14.0000
  -34.0000
>> A*x-b
ans =
  1.0e-013 *
   0.0355
   0.1421
        0
>>
```

Desno deljenje / ustreza reševanju linearnega sistema enačb x*A=b. In sicer je rešitev x=b/A.

```
>> A=[1 -2 0; 4 -5 1; 7 8 9];
>> b=[-1,4,-5];
>> x=b/A
x =
  -79.6000   25.6000   -3.4000
>> x*A-b
ans =
```

```
 1.0e-014 *
      0   -0.7105         0
>>
```

Računanje skalarnega produkta:

```
>> x=[1;2;3;4];
>> y=[5;6;-1;0];
>> x'*y
ans =
   14
>>
```

Dostopanje do matričnih elementov. Do elementa v i-ti vrstici in j-tem stolpcu pridemo z ukazom A(i,j). Z ukazomA(i,:) dobimo i-to vrstico, z ukazom A(:,j) pa j-ti stolpec.

```
>> A=[2 -4 5 -6 7; 2 -1 0 3 3; 1 -2 0 5 -5; 0 1 7 -1 2]
A =
    2   -4    5   -6    7
    2   -1    0    3    3
    1   -2    0    5   -5
    0    1    7   -1    2
>> A(2,4)
ans =
    3
>> A(2,:)
ans =
    2   -1    0    3    3
>> A(:,2)
ans =
   -4
   -1
   -2
    1
>>
```

Do podmatrik pridemo tako, da v oglatih oklepajih naštejemo indekse vrstic in indekse stolpcev elementov, ki jih želimo.

```
>> A([1,3],[2,3,5])
ans =
   -4    5    7
   -2    0   -5
>> A(:,[2 3])
ans =
   -4    5
   -1    0
   -2    0
```

```
    1     7
>> A([2 3], 2)
ans =
   -1
   -2
>> A([2 3], :)
ans =
    2   -1    0    3    3
    1   -2    0    5   -5
>>
```

Vrednosti tabeliramo z ukazom *zacetek:korak:konec.* Če koraka ne podamo, je privzeta vrednost za korak enaka 1.

```
>> 1:8
ans =
    1    2    3    4    5    6    7    8
>> 1:2:8
ans =
    1    3    5    7
>> 0:0.1:1
ans =
       0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000
>>
```

Kompleksna števila podajamo kot a+ib ali a+jb:

```
>> 3+2i
ans =
  3.0000 + 2.0000i
>> 3+2j
ans =
  3.0000 + 2.0000i
>>
```

Osnovne funkcije za delo s kompleksnimi števili:

real(z) → realni del
imag(z) → imaginarni del
abs(z) → absolutna vrednost
conj(z) → konjugirana vrednost

```
>> z=3+2i;
>> real(z)
ans =
    3
>> imag(z)
ans =
```

```
    2
>> abs(z)
ans =
   3.6056
>> conj(z)
ans =
   3.0000 - 2.0000i
>>
```

VGRAJENE FUNKCIJE ZA GENERIRANJE MATRIK

Ukazi za generiranje posebnih matrik:

rand(n) → zgenerira naključno matriko velikosti n×n z vrednostmi med 0 in 1
rand(n,m) → zgenerira naključno matriko velikosti n×n z vrednostmi med 0 in 1 velikosti n×m
randn(n) → zgenerira naključno n×n matriko s standardno normalno porazdeljenimi vrednostmi
randn(n,m) → zgenerira naključno n×m matriko s standardno normalno porazdeljenimi vrednostmi
zeros(n) → zgenerira matriko samih ničel velikosti n×n
zeros(n,m) → zgenerira matriko samih ničel velikosti n×m
ones(n) → zgenerira matriko samih enic velikosti n×n
ones(n,m) → zgenerira matriko samih enic velikosti n×m
eye(n) → zgenerira identiteto velikosti n×n
diag([d1,d2,...,dn]) → zgenerira diagonalno matriko z elementi d1,d2,...,dn na diagonali

```
>> rand(3)
ans =
   0.8147   0.9134   0.2785
   0.9058   0.6324   0.5469
   0.1270   0.0975   0.9575
>> rand(3,4)
ans =
   0.9649   0.9572   0.1419   0.7922
   0.1576   0.4854   0.4218   0.9595
   0.9706   0.8003   0.9157   0.6557
>> randn(2,3)
ans =
  -0.4326   0.1253  -1.1465
  -1.6656   0.2877   1.1909
>> zeros(2)
ans =
    0    0
    0    0
>> ones(2,4)
ans =
```

```
    1    1    1    1
    1    1    1    1
>> eye(3)
ans =
    1    0    0
    0    1    0
    0    0    1
>> eye(3,4)
ans =
    1    0    0    0
    0    1    0    0
    0    0    1    0
>> diag([1,2,3,4])
ans =
    1    0    0    0
    0    2    0    0
    0    0    3    0
    0    0    0    4
```

diag(A) → vrne diagonalo matrike A
tril(A) → vrne spodnje trikotno matriko matrike A
tril(A, i) → vrne spodnje trikotno matriko od i-te naddiagonale navzdol
tril(A, -i) → vrne spodnje trikotno matriko od i-te poddiagonale navzdol
triu(A) → vrne zgornje trikotno matriko matrike A
triu(A,i) → vrne zgornje trikotno matriko od i-te naddiagonale navzgor
triu(A,-i) → vrne zgornje trikotno matriko od i-te poddiagonale navzgor

```
>> A=[1,2,3,4; 5 6 7 8; 9 10 11 12; -5 -4 -3 0]
A =
    1    2    3    4
    5    6    7    8
    9   10   11   12
   -5   -4   -3    0
>> diag(A)
ans =
    1
    6
   11
    0
>> tril(A)
ans =
    1    0    0    0
    5    6    0    0
    9   10   11    0
   -5   -4   -3    0
```

```
>> tril(A,1)
ans =
    1    2    0    0
    5    6    7    0
    9   10   11   12
   -5   -4   -3    0
>> tril(A,-1)
ans =
    0    0    0    0
    5    0    0    0
    9   10    0    0
   -5   -4   -3    0
>> triu(A)
ans =
    1    2    3    4
    0    6    7    8
    0    0   11   12
    0    0    0    0
>> triu(A,1)
ans =
    0    2    3    4
    0    0    7    8
    0    0    0   12
    0    0    0    0
>> triu(A,-1)
ans =
    1    2    3    4
    5    6    7    8
    0   10   11   12
    0    0   -3    0
>>
```

Matrike lahko gradimo tudi bločno:

```
>> A=[1 2; 3 4]
A =
    1    2
    3    4
>> B=[1 1; 0 0]
B =
    1    1
    0    0
>> C=[A, B; zeros(2), A]
C =
    1    2    1    1
    3    4    0    0
```

```
   0    0    1    2
   0    0    3    4
>>
```

SKALARNE, VEKTORSKE IN MATRIČNE FUNKCIJE

Skalarne funkcije v Matlabu:
sin, cos, asin, acos, tan, atan, exp, log, sqrt, rem, round, floor, ceil, sign, ...

```
>> [sin(pi), cos(pi), asin(0.5), acos(0.5), tan(pi), atan(0.5)]
ans =
   0.0000   -1.0000    0.5236    1.0472   -0.0000    0.4636
>> [exp(1), log(2), sqrt(4)]
ans =
   2.7183    0.6931    2.0000
>> rem(13,4)
ans =
    1
>> [round(12.45), floor(12.45), ceil(12.45)]
ans =
   12    12    13
>> [round(12.55), floor(12.55), ceil(12.55)]
ans =
   13    12    13
>> [sign(3), sign(-3),sign(0)]
ans =
    1   -1    0
>>
```

Če skalarne funkcije uporabimo na vektorju ali matriki, se izvedejo na vsakem elementu posebej.

```
>> A=[1,2,3; 4 5 6];
>> exp(A)
ans =
   2.7183    7.3891   20.0855
  54.5982  148.4132  403.4288
```

Vektorske funkcije v Matlabu:
max, min, sort, sum, prod, mean, std, any, all

```
>> x=[2,6,11,-8,3,4,0,1];
>> max(x)
ans =
   11
>> min(x)
```

```
ans =
   -8
>> sort(x)
ans =
   -8    0    1    2    3    4    6   11
>> sum(x)
ans =
   19
>> prod(x)
ans =
    0
>> mean(x)
ans =
   2.3750
>> std(x)
ans =
   5.4232
```

any → vrne 1, če je vsaj en element v vektorju različen od 0, sicer vrne 0
all → vrne 1, če so vsi elementi v vektorju različni od 0, sicer vrne 0

```
>> any(x)
ans =
    1
>> all(x)
ans =
    0
>> any([0 0 0 0])
ans =
    0
>> all([1 2 3 4])
ans =
    1
>>
```

Če vektorske funkcije uporabimo na matriki, se izvedejo po stolpcih.

```
>> A=round(10*rand(4))
A =
    0    8    2    0
    8    7    7    1
    9    4    0    8
    7    7    3    7
>> sort(A)
ans =
    0    4    0    0
    7    7    2    1
```

```
    8     7     3     7
    9     8     7     8
>> sum(A)
ans =
   24    26    12    16
>> max(A)
ans =
    9     8     7     8
>>
```

Matrične funkcije: size, det, rank, norm, inv, ...

```
>> A=[0 8 2 0; 8 7 7 1; 9 4 0 8; 7 7 3 7];
>> size(A)
ans =
    4     4
>> det(A)
ans =
       1236
>> rank(A)
ans =
    4
>> norm(A)
ans =
   21.0490
>> inv(A)
ans =
    0.0324    0.1100    0.2298   -0.2783
    0.1731   -0.0113    0.0793   -0.0890
   -0.1926    0.0453   -0.3172    0.3560
   -0.1230   -0.1181   -0.1731    0.3576
>>
```

Funkcije imajo lahko več izhodnih argumentov. Npr. funkcija eig izračuna lastne vrednosti in lastne vektorje matrike A:

```
>> A=[1 2 3 6; 2 5 -1 4; 3 -1 2 1; 6 2 1 4]
A =
    1     2     3     6
    2     5    -1     4
    3    -1     2     1
    6     2     1     4
>> [X, D]=eig(A)
X =
    0.5293    0.7848   -0.1886   -0.0129
    0.5472    0.0370    0.8051   -0.5738
    0.1817   -0.2855   -0.5519   -0.6390
```

```
   0.6224   -0.5489   -0.1081    0.5121
D =
  11.1527         0         0         0
       0   -4.1935         0         0
       0         0    4.6799         0
       0         0         0    0.3609
>> X*D*inv(X)
ans =
   1.0000    2.0000    3.0000    6.0000
   2.0000    5.0000   -1.0000    4.0000
   3.0000   -1.0000    2.0000    1.0000
   6.0000    2.0000    1.0000    4.0000
>>
```

RELACIJE V MATLABU

Za relacije Matlab uporablja <, >, <=, >=, ==, ~=
Če je relacija izpolnjena vrne 1, sicer vrne 0. Na vektorjih in matrikah se relacije vrednotijo po komponentah.

```
>> 3< 5
ans =
    1
>> 3 >= 5
ans =
    0
>> [1 2; 3 4] <= [0 4; 1 1]
ans =
    0    1
    0    0
>> [1 2; 3 4] ~= [5 6; 3 4]
ans =
    1    1
    0    0
>> [1 2; 3 4] == [5 6; 3 4]
ans =
    0    0
    1    1
>>
```

IF STAVKI IN ZANKE V MATLABU

Struktura if stavka:

if pogoj1
    stavki 1;
elseif pogoj2
    stavki 2;
else
    stavki 3;
end

```
>> x=-3;
>> if x < 0, znak=-1, elseif y==0, znak=1, else znak=0, end
znak =
   -1
>>
```

Struktura for zanke:

for iterator
    stavki;
end

```
>> x=[];
>> for i=1:5, x=[x, 2^i]; end
>> x
x =
    2    4    8   16   32
>>
```

```
>> x=[];
>> for i=[-1,0,2,5], x=[x, 2^i]; end
>> x
x =
   0.5000    1.0000    4.0000   32.0000
>>
```

Struktura while zanke:

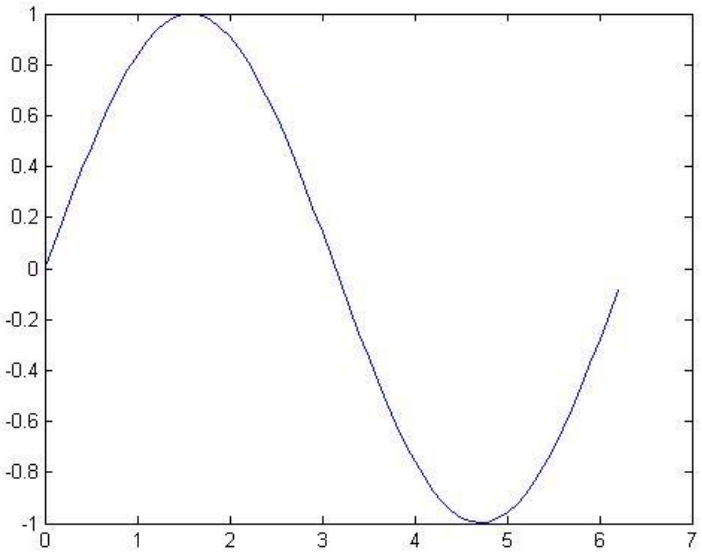
while pogoj
    stavki;
end

```
>> i=0;
>> while i<=5, disp(i); i=i+1; end
    0
    1
    2
    3
    4
    5
>>
```

RISANJE GRAFOV

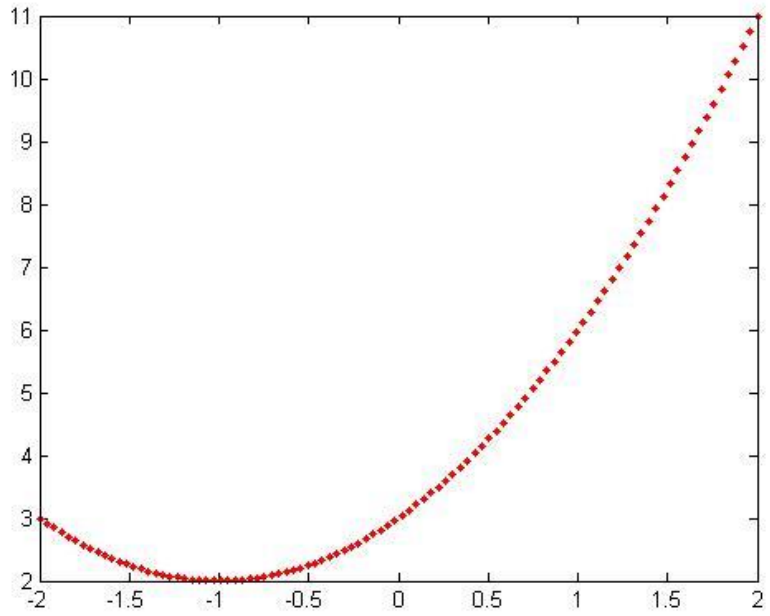
Risanje grafov v Matlabu. Za risanje grafov funkcij ene spremenljivke se uporablja ukaz plot. Risanje poteka diskretno. V izbranem intervalu izberemo dovolj gosto množico točk (X) in izračunamo vrednosti funkcije v teh izbranih točkah (Y). Kličemo plot(X,Y), ki nariše točke s koordinatami (xi,yi) in jih poveže z odsekoma linearno funkcijo.

```
>> x=0:0.1:2*pi;
>> y=sin(x);
>> plot(x,y)
```

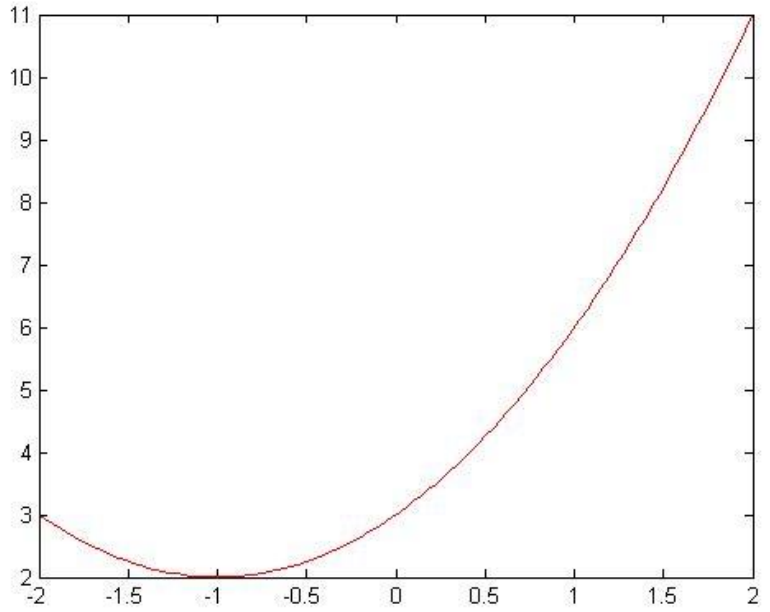


Za razdelitev intervala [a,b] na N ekvidistantnih delov lahko uporabimo ukaz linspace(a,b,N).

```
>> x=linspace(-2,2,100);
>> y=x.^2+2*x+3;
>> plot(x,y,'r.')
```

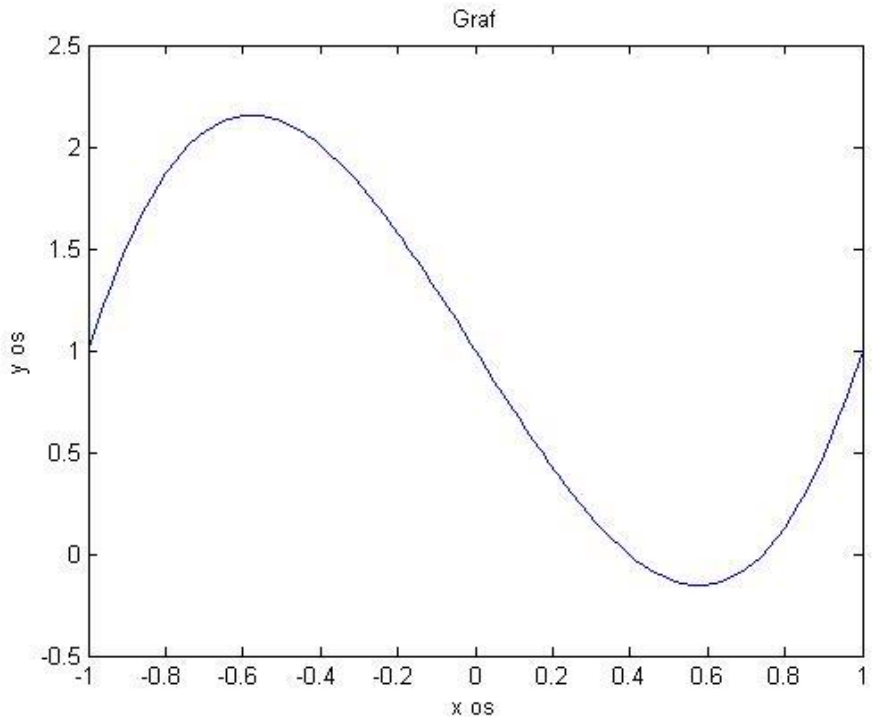



```
>> x=linspace(-2,2,100);
>> y=x.^2+2*x+3;
>> plot(x,y,'r-')
```



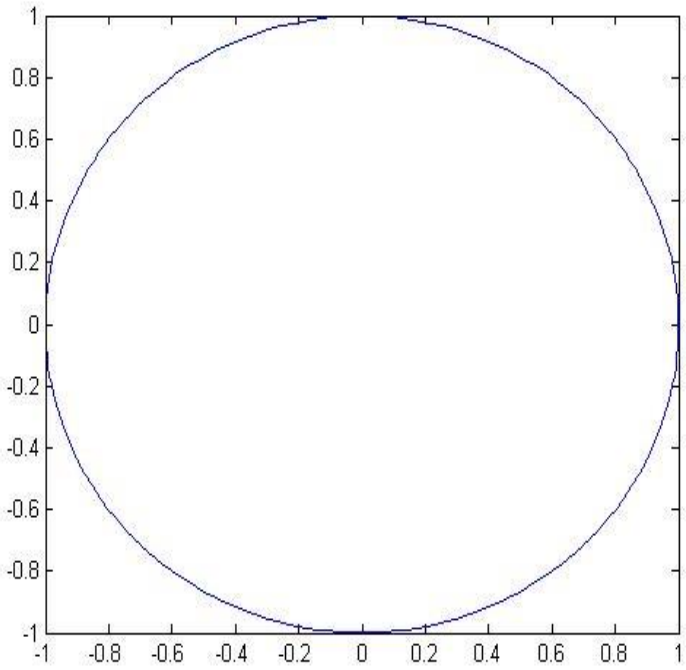

```
>> x=linspace(-1,1,100);
```

```
>> y=x.^3+2*x.^3-3*x+1;
>> plot(x,y)
>> title('Graf')
>> xlabel('x os')
>> ylabel('y os')
```



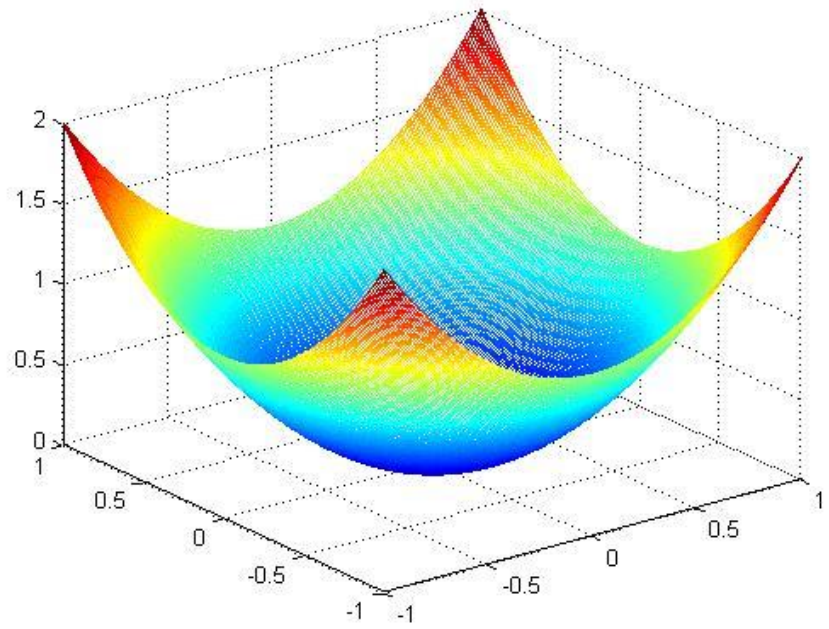
Risanje parametričnih krivulj:

```
>> t=linspace(0,2*pi,200);
>> x=sin(t);
>> y=cos(t);
>> plot(x,y)
>>
```

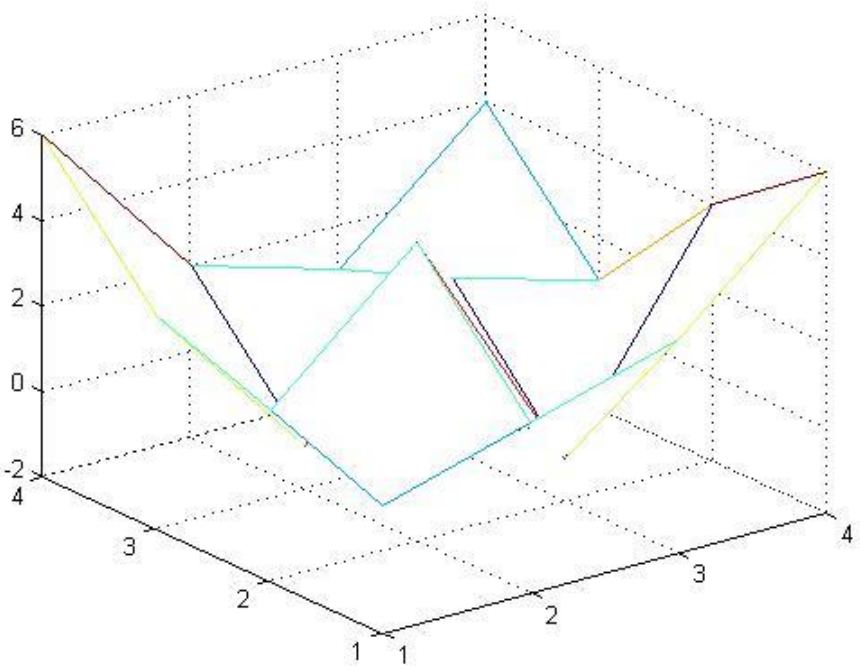


Risanje ploskev:

```
>> [X,Y]=meshgrid(-1:0.01:1,-1:0.01:1);
>> Z=X.^2+Y.^2;
>> mesh(X,Y,Z)
```



Risanje matričnih elementov:
>> A=[1 2 3 6; 2 5 -1 4; 3 -1 2 1; 6 2 1 4];
>> mesh(A)



SKRIPTNE IN FUNKCIJSKE DATOTEKE V MATLABU

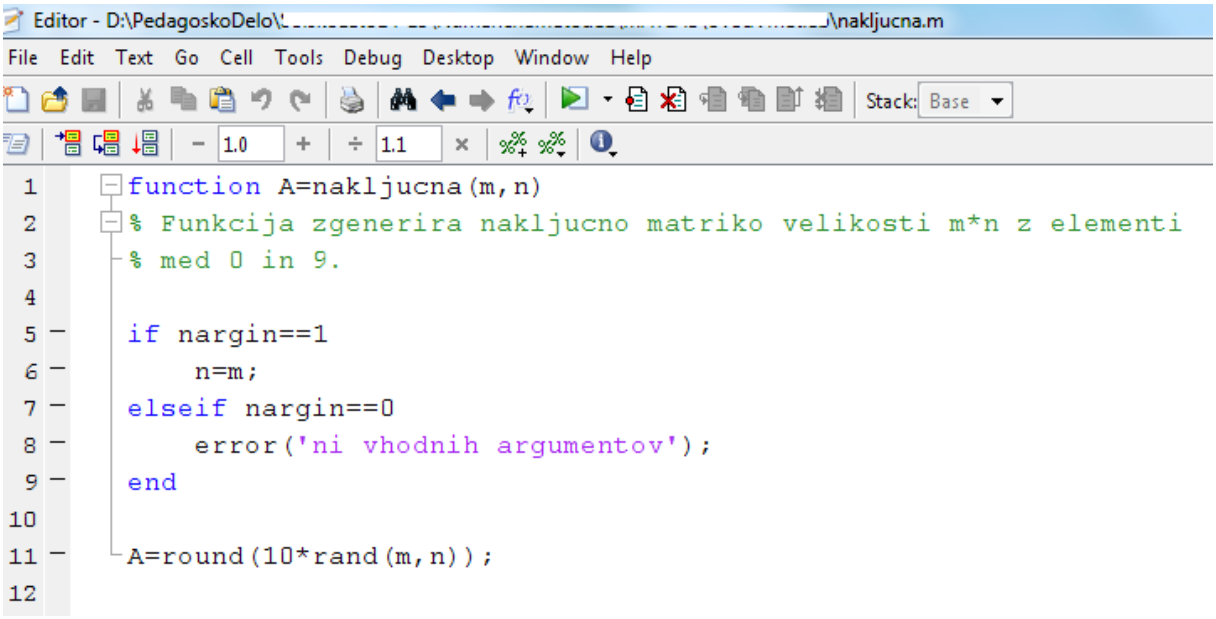
Kadar hočemo v Matlabu izvesti več zaporednih ukazov, lahko le te zapišemo v skriptno datoteko. Vanjo pišemo v *Matlabovem Editorju*. Ko kliknemo na *save*, se nam odpre okno, v katerega vpišemo ime datoteke. Paziti je treba, da pri izbiri imena ne uporabljamo šumnikov ali kakšnih drugih posebnih znakov. Preden datoteko shranimo, je dobro, da si v okencu *Current Directoy* nastavimo ustrezno mapo, kamor naj se datoteke shranjujejo. Ko hočemo, da se zaporedje ukazov iz skriptne datoteke izvrši, vpišemo njeno ime v ukazno okence ter kliknemo *enter*. Pomembno je vedeti, da so vse spremenljivke v skriptni datoteki globalne.

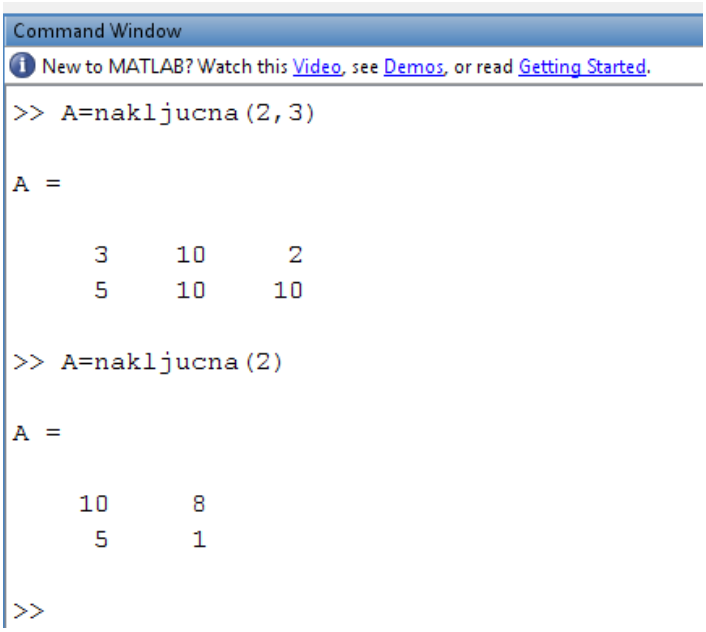
V Matlabu pa lahko pišemo tudi svoje funkcije. Postopamo zelo podobno kot pri skriptnih datotekah, razlika je le v prvi vrstici, ki mora biti oblike:

function [izhodni arumenti]=imeFunkcije(vhodni argumenti)

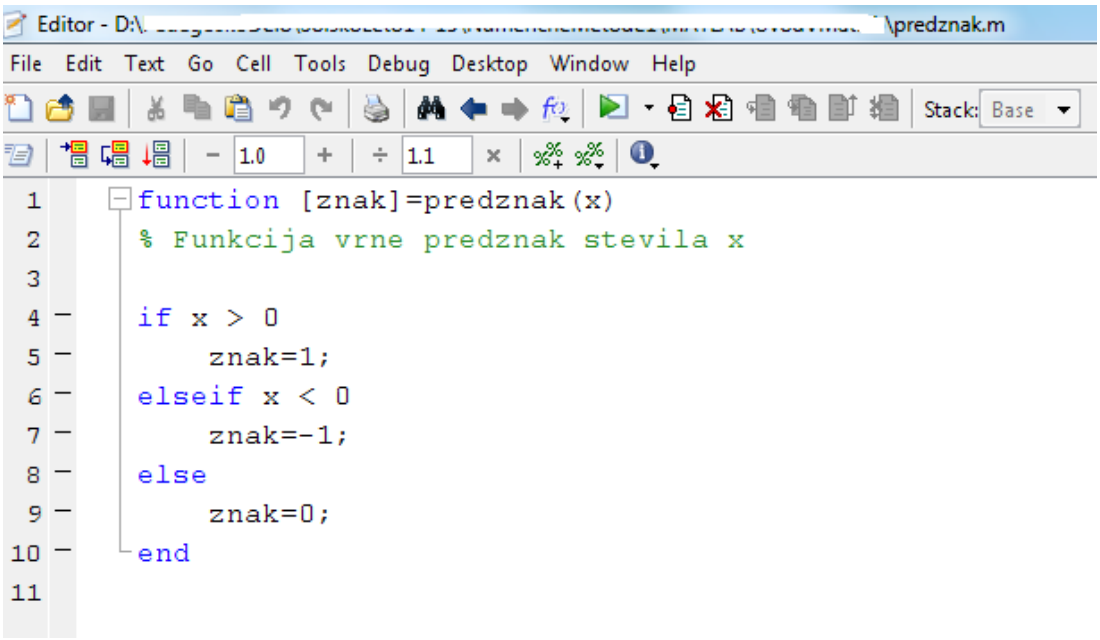
Vse spremenljivke, ki jih definiramo znotraj funkcije, so lokalne. Funkcije lahko imajo en ali več vhodnih/izhodnih argumentov. Število vhodnih argumentov je lahko tudi spremenljivo. Pri shranjevanju moramo biti pazljivi, da datoteki dodelimo isto ime, kot je ime funkcije.

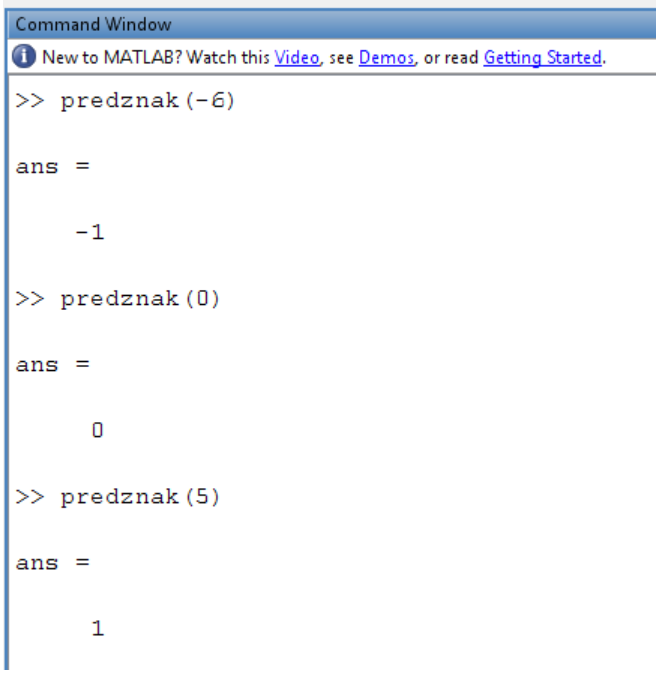
Poglejmo si primer funkcije, ki zgenerira naključno matriko velikosti m×n s celimi števili med 0 in 9. Izhodni argument je matrika, vhodna argumenta pa sta m in n. V primeru, da uporabnik vpiše en sam vhodni argument, naj bo m=n. Za določitev števila vhodnih argumentov uporabimo funkcijo *nargin.*





Naslednja funkcija vrne predznak števila x.





POLINOMI V MATLABU

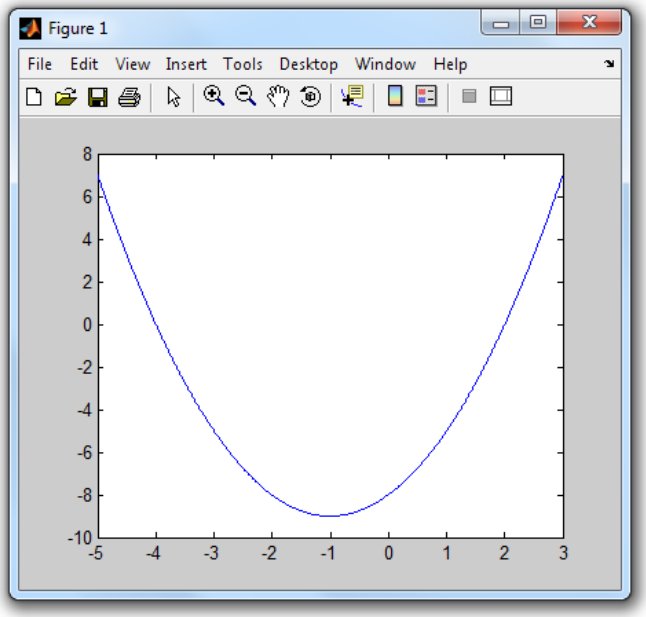
Polinomi so v Matlabu predstavljeni s seznamom koeficientov. In sicer nam seznam P=[p1,p2,...,pn] predstavlja polinom stopnje n-1 z vodilnim koeficientom p1 in prostim koeficientom pn. Funkcije za delo s polinomi so sledeče.

polyval(P,x) → vrne vrednost polinoma s koeficienti P v točki x
roots(P) → vrne ničle polinoma s koeficienti P
poly(N) → vrne koeficiente polinoma z ničlami N

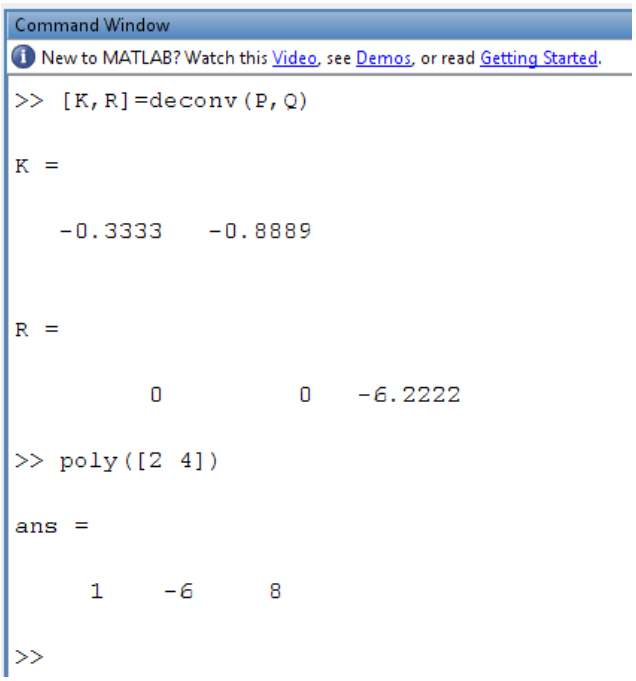
conv(P,Q) → vrne koeficiente produkta dveh polinomov s koeficienti P in Q

deconv(P,Q) → vrne koeficiente kvocienta ter ostanka pri deljenju P s Q

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> P=[1 2 -8]

P =

     1     2    -8

>> polyval(P,0)

ans =

    -8

>> roots(P)

ans =

    -4
     2

>> plot(-5:0.01:3,polyval(P,-5:0.01:3))
>>
```




```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> P=[1 2 -8]

P =

     1     2    -8

>> Q=[-3,2]

Q =

    -3     2

>> conv(P,Q)

ans =

    -3    -4    28   -16
```


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> [K,R]=deconv(P,Q)
K =
-0.3333 -0.8889
R =
0 0 -6.2222
>> poly([2 4])
ans =
1 -6 8
>>