

1 Integrované výpočtové prostredie **MATLAB**TM

The Math Works, Inc.



Stručná charakteristika a základy použitia

Charakteristika MATLAB-u

- **Integrované výpočtové prostredie pre realizáciu vedecko-technických výpočtov**
- **Výkonné výpočtové jadro, ktoré je možné používať v príkazovom („ručnom“) režime alebo v programovom („automatickom“) režime**
- **Stovky hotových funkcií a nástrojov použiteľných pri tvorbe rôznych aplikácií a výpočtov**
- **Univerzálny a jednoduchý programovací jazyk**
- **Modelovací a simulačný nástroj (Simulink)**
- **Nástroje na tvorbu finálnych, priamo spustiteľných aplikácií**

- **Výkonné grafické nástroje pre zobrazovanie výsledkov v 2D a 3D priestore v statickom aj dynamickom (animovanom) režime**
- **Nástroje na tvorbu grafického rozhrania s používateľom**
- **Matlab dnes predstavuje celosvetový štandard pre tvorbu vedecko-technických výpočtových, modelovacích a simulačných aplikácií**
- **Použitie Matlabu môže významne zjednodušiť a urýchliť vývojové a programovacie práce**

Aplikačné domény Matlabu

- **Matematika**
- **Vývoj algoritmov a programovanie**
- **Kybernetika, riadenie**
- **Spracovanie signálov**
- **Spracovanie obrazu**
- **Elektrotechnika**
- **Umelá inteligencia**
- **Mechanika, robotika**
- **Letectvo, kozmonautika**
- **Ekonómia a financie**
- **Virtuálna realita**
- **Biológia ...**
- **Používateľom vytvorené nástroje pre ľubovoľné aplikačné domény**

Stručný kurz Matlabu

- **Prostredie - hlavné okno**
- **Údaje (dáta)**
- **Operácie s dátami**
- **Funkcie**
- **Programovanie**
- **Grafika**
- **Simulink**
- **Príklady**

1.1 Prostredie

- **Command Window** – zadávanie príkazov, spúšťanie programov
- **Workspace** – zoznam premenných v pracovnom priestore (v pamäti)
- **Current Directory** – aktuálny adresár
- **Command History** – minulé príkazy
- **Menu**
- **Help**
- **Tlačidlo Start**
- **Ďalšie typy okien** – obrázky, animácie ...

Prostredie

Nový skript

Command window

Variable editor (dvojklik)

simulink

Current folder

Workspace (zoznam premenných)

Command history

The screenshot shows the MATLAB 7.11.0 (R2010b) environment. The 'Current Folder' pane on the left shows the path 'C:\Users\User\Documents\MATLAB'. The 'Command Window' displays the execution of 'a = magic(5)' and 'b = magic(500);'. The 'Workspace' pane on the right lists variables 'a' (5x5 double) and 'b' (500x500 double). The 'Command History' pane at the bottom right shows a list of executed commands, including 'A = magic(5);', 'length(A)', 'A = magic(5,1)', 'A = magic([5 6]);', 'length(A)', 'size(A)', 'max(A)', 'size(A)', 'find(A>10)', '[value, position] = find(A>10)', '[value, rows, cols] = find(A>10)', 'find(A>10)', 'clc', 'a = A(:);', 'AA = reshape(a,[5 6]);', 'AA = reshape(a,[5 5]);', 'A', 'A - AA', 'clc', 'a = magic(5);', and 'b = magic(500);'. A small number '7' is visible at the bottom right of the Command History pane.

Name	Value	Min	Max
a	<5x5 double>	1	25
b	<500x500 double>	1	25000

```
>> a = magic(5)

a =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

>> b = magic(500);
fx >> |
```

```
A = magic(5);
length(A)
A = magic(5,1)
A = magic([5 6])
length(A)
size(A)
max(A)
size(A)
find(A>10)
[value, position] = find(A>10)
[value, rows, cols] = find(A>10)
find(A>10)
clc
a = A(:);
AA = reshape(a,[5 6])
AA = reshape(a,[5 5])
A
A - AA
clc
a = magic(5);
b = magic(500);
```

Príkazový režim práce

>> příkaz1 + Enter

>> příkaz2 + Enter

>> ...

Programový režim práce

>> program + Enter

Command Window:

```
>> r=2.7
```

```
r =
```

```
2.7000
```

```
>> v=8
```

```
v =
```

```
8
```

```
>> objem_valca=pi*r^2*v
```

```
objem_valca =
```

```
183.2177
```

```
8
```


Niektoré dôležité príkazy

clc – vymazanie okna

clear x – vymazanie premennej x

clear all – vymazanie všetkých premenných, používať iba v prípade potreby, inak stačí samotný *clear* – bez príznaku vymaže všetky vaše premenné vo workspace

close h – zatvorí obrázok s handlom h

close all – zatvorí všetky obrázky

clc, clear, close all – **veľmi vhodné na začiatok každého skriptu**

who – zoznam premenných

whos – zoznam a veľkosti prem.

lookfor *heslo* – vyhladá príkazy, v ktorých sa vyskytuje *heslo*

help / doc *heslo* – *stručný help / obsahla dokumentácia*

cd – zmena adresára

pwd – zobrazenie aktuálneho adresára

dir – obsah aktuálneho adresára

which *príkaz* – vyhladá adresár, v ktorom sa nachádza *príkaz*

↑ - návrat k predchádzajúcim príkazom

Niektoré dôležité príkazy – matice / viacrozmerne polia

Väčšina maticových príkazov pracuje defaultne po stĺpcoch

Príklad: `max(A)` zavolá `max(A(:,i))` pre každý stĺpec zvlášť a výsledok zoradí do riadkového vektora

length / size – rozmery polí / matic

find(a > 0) – nájde hodnoty prvkov ktoré vyhovujú podmienke

min / max – min/max. prvok

[m,indx]=max(a) – možný aj takýto zápis, kde okrem maxima *m*, vráti aj poradie v poli *indx*

reshape(vector, [rows, cols]) – zreformuje vector do matice, po stĺpcoch

A(:) – z matice vytvorí stĺpcový vektor, prvky budú po stĺpcoch

A(end) – posledný prvok

Realizácia príkazu (programu)

>> **abc**

1. ak *abc* je premenná, zobrazí sa jej obsah
 2. ak *abc* je názov príkazu/programu (m-file) v aktuálnom adresári, spustí ho
 3. inak hľadá *abc* v inom adresári, kde má nastavené cesty a spustí ho
- Nepomenovávať skripty rovnako ako natívne / knižničné skripty (matlab to dovoľí). Prioritu má to čo je v lokálnom priečinku.
 - Nepomenovávať premenné ako akékoľvek skripty (častá chyba pomenovania vlastnej premennej ako max, min,..)
- Oboje vedie ku častým a ťažko dohl'adateľným chybám

1.2 Dátové štruktúry

- **matice** (polia) - základný objekt v ML
n-rozmerné polia, 2-rozmerné polia
- **vektory** - 1-rozmerné polia
- **skaláry** - 1-prvkové vektory
- **súbory** - dáta prenesené z prac. priestoru na
iné médium

Zadanie skalárnej premennej →

```
>> skalar=1.78
```

```
skalar =
```

```
1.7800
```

Ak sa za príkazom napíše ; výsledok sa nevypisuje →

```
>> skalar=1.78;
```

```
>>
```

```
>> vektor=[1 2 3]
```

Zadanie riadkového vektora →

```
vektor =
```

```
1      2      3
```

```
>> vektor=[1,2,3]
```

```
vektor =
```

```
1      2      3
```

```
>>
```

Zadanie stĺpcového vektora



```
>> vektor=[1;2;3]
```

```
vektor =
```

```
1
```

```
2
```

```
3
```

alebo pomocou transpozície (x')



```
>> vektor=[1,2,3]'
```

```
vektor =
```

```
1
```

```
2
```

```
3
```

```
>> vektor'
```

```
ans =
```

```
1
```

```
2
```

```
3
```

```
>>
```

Zadanie matice
(2D-poľ'a)



```
>> matica=[1 2 3;4 5 6;7 8 9]
```

```
matica =
```

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

alebo pomocou Enteru
na konci riadku



```
>> matica2=[1 2 3 4
```

```
5 6 7 8
```

```
9 10 11 12]
```

```
matica2 =
```

```
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

```
>>
```

Inicializácia polí

Pre urýchlenie programového napĺňania pol'a napr. cez *for* je dobré si polia najskôr inicializovať s určenou veľkosťou napr.

rand, zeros, ones

```
>> a=zeros(3)
```

```
a =
```

```
    0    0    0
    0    0    0
    0    0    0
```

```
>> b=zeros(2,3)
```

```
b =
```

```
    0    0    0
    0    0    0
```

```
>> c=ones(2)*7
```

```
c =
```

```
    7    7
    7    7
```

```
>> d=rand(2,4)
```

```
d =
```

```
    0.6822    0.5417    0.6979    0.8600
    0.3028    0.1509    0.3784    0.8537
```

```
>> e=[]
```

```
e =
```

```
    []
```

```
>> f=2:1:7
```

```
f =
```

```
    2    3    4    5    6    716
```


Indexovanie do poľa Výber / zápis

```
>> a=[1 2 3 4;5 6 7 8;9 10 11 12]
```

```
a =
```

1	2	3	4
5	6	7	8
9	10	11	12

```
>> a(2,3)
```

```
ans =
```

7

```
>> a(1,:) 
```

```
ans =
```

1	2	3	4
---	---	---	---

```
>> a(:,1)
```

```
ans =
```

1
5
9

- Indexovanie vždy (riadky , stĺpce) !!

- Pri priradovaní vektora do matice musí sedieť rozmer

- Index „:“ znamená „všetky“

Presnosť → help format

Veľkosť polí

size(pole); [rows, cols] = size(pole);

size(pole,1) veľkosť prvej dimenzie poľa

size(pole,2) veľkosť druhej dimenzie poľa

size(pole,3) veľkosť tretej dimenzie poľa ...

length(vektor);

Indexovanie do poľa Výber / zápis

```
a =
```

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

```
>> a(3,2)=8
```

```
a =
```

1	1	1	1
1	1	1	1
1	8	1	1
1	1	1	1

```
a =
```

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

```
>> a(3,2:4)=8
```

```
a =
```

1	1	1	1
1	1	1	1
1	8	8	8
1	1	1	1

Viacrozmerne polia

[rows, cols, depth] = size(pole);

```
>> P(3,2,2)=7
```

```
P(:, :, 1) =
```

```
    0    0
    0    0
    0    0
```

```
P(:, :, 2) =
```

```
    0    0
    0    0
    0    7
```

```
>> size(P)
```

```
ans =
```

```
    3    2    2
```

Spájanie polí

```
a =
```

```
    1    1  
    1    1
```

```
>> b
```

```
b =
```

```
    2    2    2  
    3    3    3
```

```
>> a=[a b]
```

```
a =
```

```
    1    1    2    2    2  
    1    1    3    3    3
```

```
p =
```

```
    1    2    3
```

```
>> q
```

```
q =
```

```
    7    7    7
```

```
>> r=[p q]
```

```
r =
```

```
    1    2    3    7    7    7
```

```
>> s=[p;q]
```

```
s =
```

```
    1    2    3  
    7    7    7
```

Špeciálne konštanty

pi = 3.1416

i, j - imaginárna časť komplexného čísla

Inf - nekonečno (napr. po delení nulou)

NaN - Not a Number (0/0 ...)

clock - čas

date - dátum

eps - najmenšie možné číslo

ans - výsledok operácie

Uklanie dát do súborov

save *meno* x y A - uloží premenné x,y,A do súboru *meno.mat*

save *meno.qqq* x -ascii - uloží premennú x do súboru *meno.qqq*
vo formáte ascii

save *meno* - uloží všetky premenné pracovného priestoru do
súboru *meno.mat*

load *meno* - načíta všetky premenné zo súboru *meno*

1.3 Operácie s dátami

- **základné matematické operácie**
- **maticové operácie**
- **knižnice funkcií**
- **používateľom vytvorené funkcie**

Základné matematické operácie

premenná=výraz;

Výrazy obsahujú konštanty, premenné a operácie:

$a+b$, $a-b$, $a*b$, a/b , a^b , $\text{sqrt}(\cdot)$, ...

Príklad:

$$\mathbf{x=2^{(5/(8+2))} \quad \rightarrow \quad x=2^{(5/10)}}$$

Maticové operácie

- súčet matic: $\mathbf{A+B}$ ← Matlab prioritne vykonáva
- súčin matic: $\mathbf{A*B}$ ← operácie maticami !!!
- násobenie matice konštantou: $\mathbf{A*const}$
- pričítanie konštanty k matici: $\mathbf{A+const}$
- inverzia matice: $\mathbf{inv(A)}$
- operácia "po prvkoch": $\mathbf{A.*B}$ (aj vektor je matica [n x 1]), pre ľubovoľné operácie $+ - * / ^$

Funkcie

y=funkcia(x);

Funkcie môžu vracať viac premenných:

[y1,y2,...,yn]=funkcia(x);

- **základné matematické funkcie:**

$\sin(x)$, $\cos(x)$, ... , $\exp(x)$, $\text{abs}(x)$, ...

- **iné funkcie:**

knižnice (toolbox-y) so stovkami funkcií

→ pozri Help / Doc

- **používateľsky vytvorené funkcie**

Knižnice funkcií

- **Datafun** - analýza dát (help Datafun)
- **Polyfun** - interpolácia a polynómy
- **Graph2D, Graph3D** - grafy
- ...
- >> help
- >> demo

```
>> help
```

```
HELP topics:
```

toolbox\genetic	- (No table of contents file)
matlab\general	- General purpose commands.
matlab\ops	- Operators and special characters.
matlab\lang	- Programming language constructs.
matlab\elmat	- Elementary matrices and matrix manipulation.
matlab\elfun	- Elementary math functions.
matlab\specfun	- Specialized math functions.
matlab\matfun	- Matrix functions - numerical linear algebra.
matlab\datafun	- Data analysis and Fourier transforms.
matlab\audio	- Audio support.
matlab\polyfun	- Interpolation and polynomials.
matlab\funfun	- Function functions and ODE solvers.
matlab\sparfun	- Sparse matrices.
matlab\graph2d	- Two dimensional graphs.
matlab\graph3d	- Three dimensional graphs.
matlab\specgraph	- Specialized graphs.
matlab\graphics	- Handle Graphics.
matlab\uitools	- Graphical user interface tools.
matlab\strfun	- Character strings.
matlab\iofun	- File input/output.
matlab\timefun	- Time and dates.

Príklad použitia funkcie max(x)

```
a =  
  
    4.1865    8.3812    5.0281    1.9343    6.9790    4.9655    6.6023  
    8.4622    0.1964    7.0947    6.8222    3.7837    8.9977    3.4197  
    5.2515    6.8128    4.2889    3.0276    8.6001    8.2163    2.8973  
    2.0265    3.7948    3.0462    5.4167    8.5366    6.4491    3.4119  
    6.7214    8.3180    1.8965    1.5087    5.9356    8.1797    5.3408  
  
>> max(a)  
  
ans =  
  
    8.4622    8.3812    7.0947    6.8222    8.6001    8.9977    6.6023  
  
>> max(max(a))  
  
ans =  
  
    8.9977
```

Polynomiálna regresia - preloženie polynómu cez 7 bodov v rovine pomocou polynómu 4.stupňa v tvare

$$P(x)=a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$$

Výsledkom sú koeficienty polynómu.

```
x =  
    1    2    3    4    5    6    7  
  
>> y  
  
y =  
    2    1    5    7    0    5    9  
  
>> polyfit(x,y,4)  
  
ans =  
    0.1439   -2.0808    9.9167  -17.0014   10.7143
```

Výpočet koreňov polynómu

$$y=10x^5+27x^4+12x^3+8x^2+2x+39$$

```
>> korene=roots([10 27 12 8 2 39])  
  
korene =  
  
    -2.4389  
    -0.7832 + 1.0044i  
    -0.7832 - 1.0044i  
     0.6527 + 0.7481i  
     0.6527 - 0.7481i
```

Riešenie sústavy lineárnych rovníc

A – matica koeficientov lin. rovníc

b – vektor pravých strán

```
A =  
  
     2     4     6     1  
     1     0     3     7  
     2    -1    -2     8  
     2     2     4     1
```

```
>> b
```

```
b =
```

```
    10  
    15  
     9  
     2
```

```
>> x=inv(A)*b
```

```
x =
```

```
   -4.4478  
    3.9254  
    0.0746  
    2.7463
```

Používateľom definované funkcie

Príklad:

```
function [M,m,s,r]=statistika(x1,x2,...,xn)
```

```
M=max(x);
```

```
m=min(x);
```

```
s=mean(x);
```

```
r=std(x);
```

vracané parametre

meno funkcie

vstupné parametre

```
x = [2.3 0.01 7 -9.21 10 25.4 1.0 2.77];  
[max, min, priem, rozpt] = statistika(x);
```


1.4 Grafy

- **2D grafy (→help graph2D)**
- **3D grafy (→ help graph3D)**

2D grafy (obrázky)

plot(x,y) – kreslenie grafu

axis([x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}]) - zmena mierky osí

grid – nakreslenie rastru v grafe

hold – zafixovanie (odfixovanie) grafu pre viacnásobné
kreslenie grafov (hold on, hold off)

xlabel('text') – označenie osi x

ylabel('text') – označenie osi y

title('text') – nadpis obrázku

gtext('text') – umiestnenie textu do obrázku

figure – nový obrázok

figure(n) – otvorenie/adresovanie obrázku číslo n

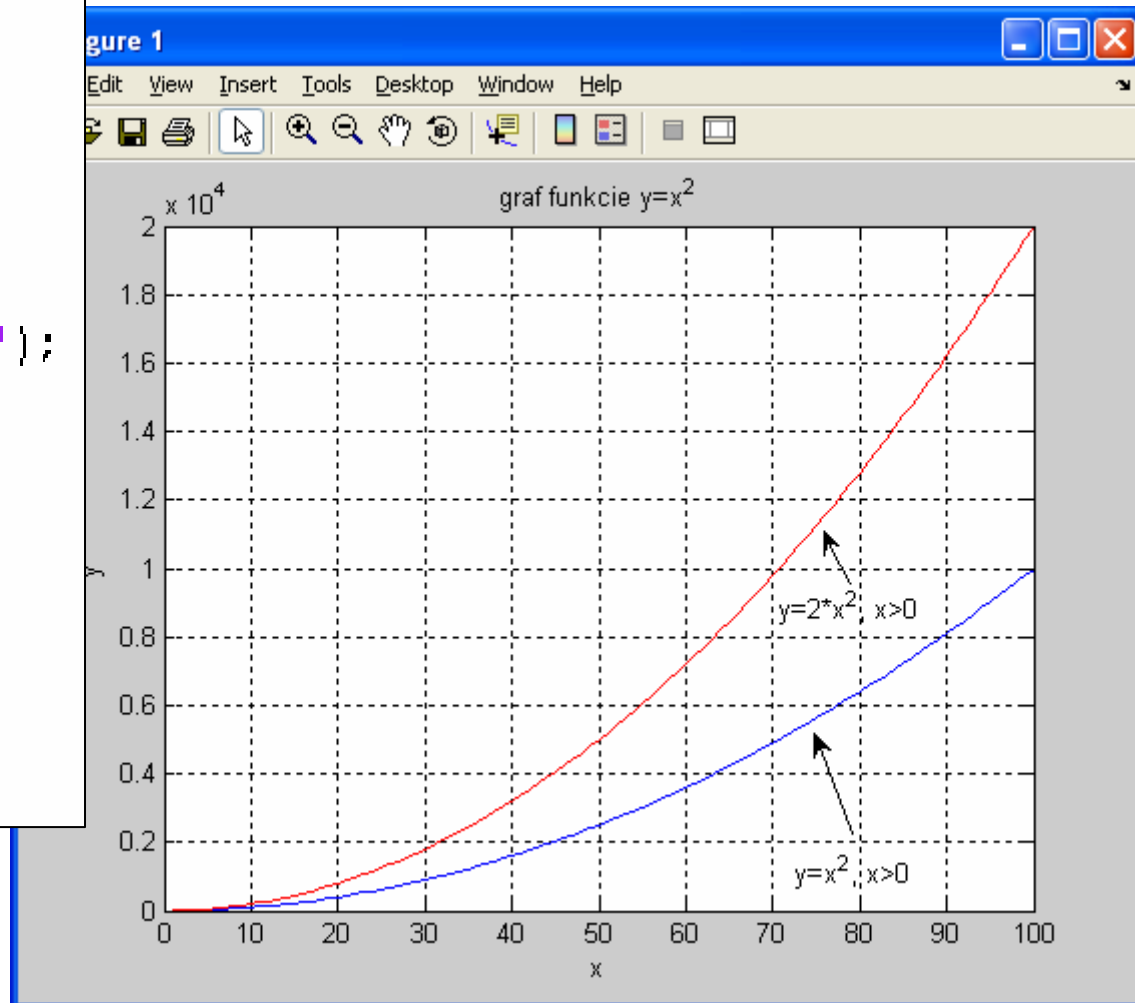
clf – vymazanie obrázku

subplot(abc) – rozdelí okno na maticu a x b podokien,
adresuje podokno s poradovým číslom c

close(n)/close all – zavretie obrázku č. n / všetkých obr.

Príklad 2D grafu

```
>> x=1:1:100;  
>> y=x.^2;  
>> plot(x,y)  
>> xlabel('x');  
>> ylabel('y');  
>> title('graf funkcie y=x^2');  
>> grid  
>> gtext('y=x^2, x>0')  
>> hold  
Current plot held  
>> y2=2*x.^2;  
>> plot(x,y2,'r')
```



plot (→ help plot)

plot(y) – vykreslenie vzoriek vektora y do grafu

plot(x,y) – vykreslenie závislosti $y=f(x)$

plot(x,y,'color') – definovanie farby grafu

color: b-modrá (blue)

r-červená (red)

g-zelená (green)

y-žltá (yellow)

m-fialová (magenta)

c-bledomodrá (cyan)

k-čierna (black)

plot(x,y,'co') – definovanie farby grafu a symbolu

o-symbol (x,o,*,+, . ,d, ...), c-farba, x, y - vektory

plot(x,y,'co') – vykreslenie bodu v rovine

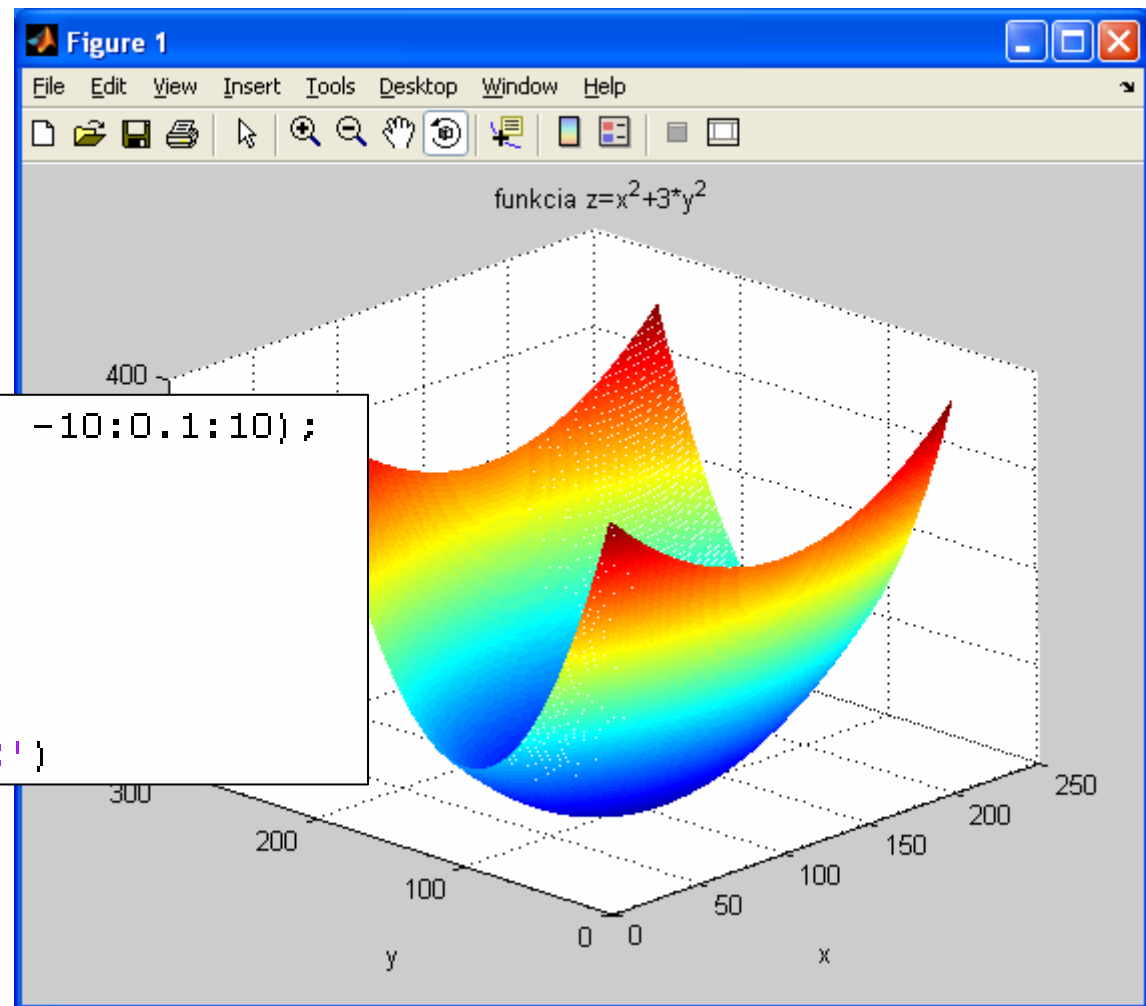
o-symbol (x,o,*,+, .), c-farba, x,y – súradnice bodu

3D grafy

→ help graph3D

```
>> [x,y]=meshgrid(-10:0.1:10, -10:0.1:10);  
>> z=x.^2+3*y.^2;  
>> mesh(z)  
>> xlabel('x')  
>> ylabel('y')  
>> zlabel('z')  
>> title('funkcia z=x^2+3*y^2')
```

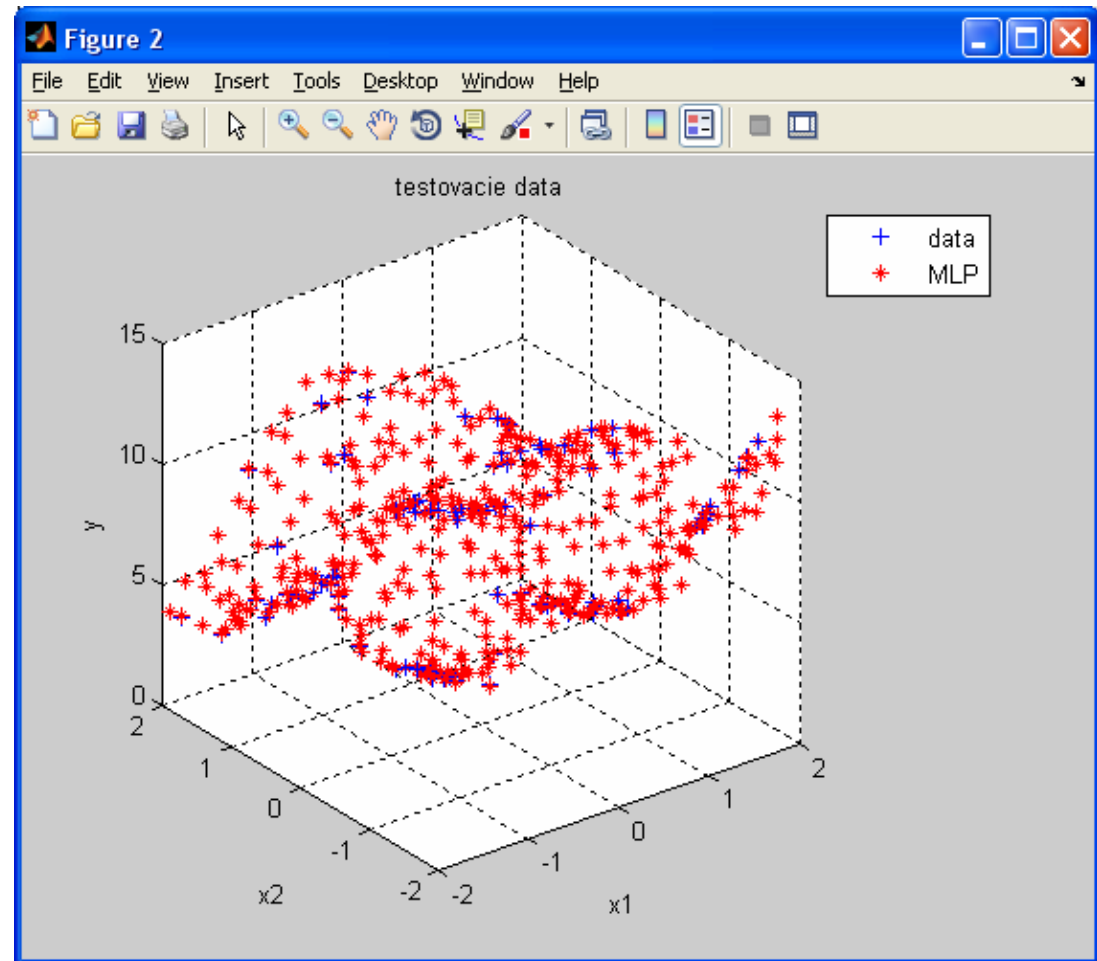
- Vid' doc...
- Scatter
- plot3d
- Surf
- ...



3D graf

→ Porovnanie mer. bodov
funkcie $y=f(x_1, x_2)$ v 3D

```
>> figure  
>> plot3(x1t,x2t,yt,'b+',x1t,x2t,z,'r*')  
>> title('testovacie data')  
>> xlabel('x1')  
>> ylabel('x2')  
>> zlabel('y')  
>> legend('data','MLP')  
>> grid on  
>> axis([-2 2 -2 2 0 15])
```

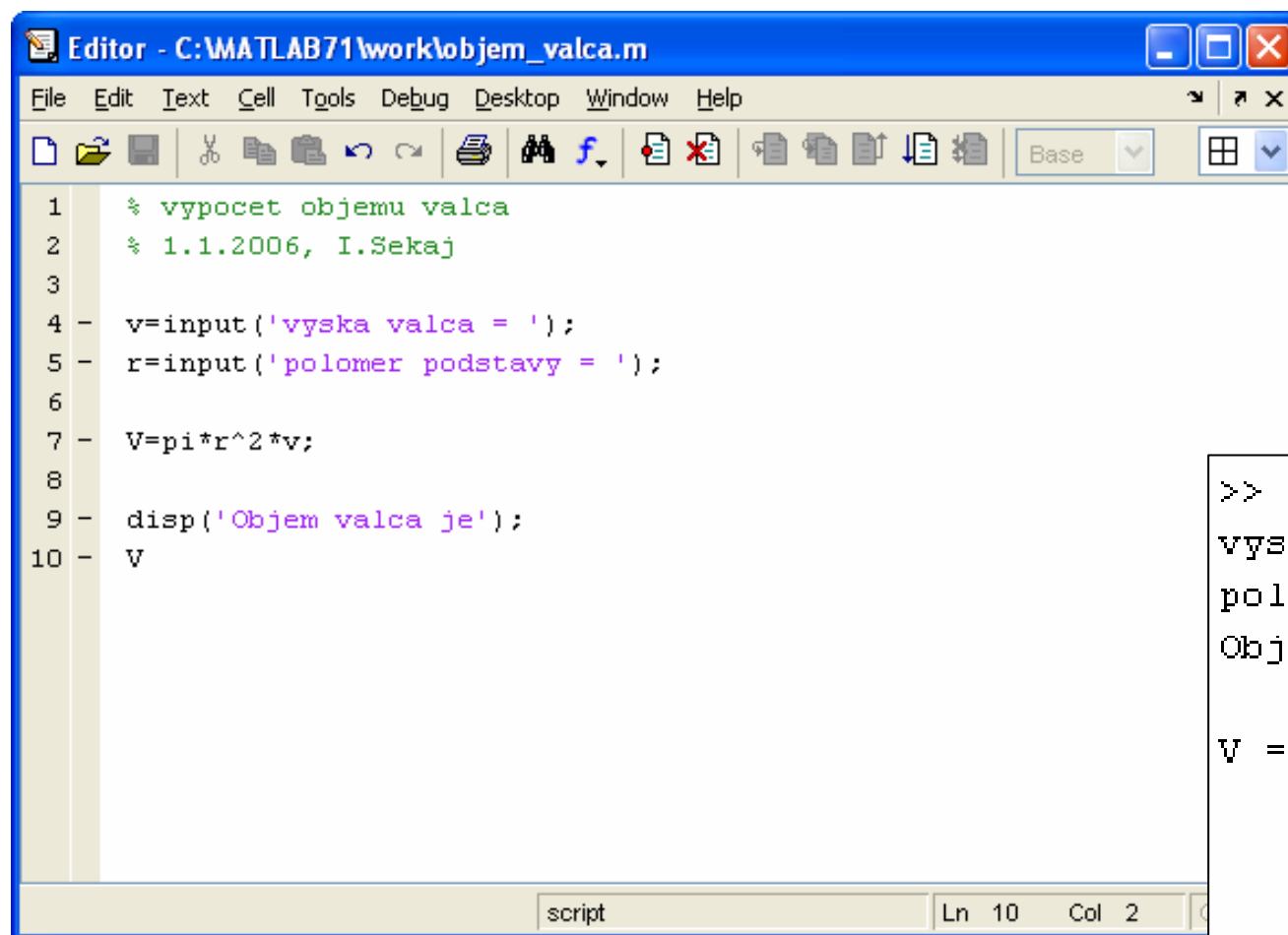


1.5 Programovanie

tvorba m-súborov (m-files)

- Skript zdieľa pamäť s hlavným workspacom
- Funkcia má svoj vlastný pamäťový priestor
- Tvorba používateľských programov,
- postupnosť príkazov ako v riadkovom režime,
- môžu byť použité príkazy, funkcie aj iné m-súbory,
- tvorba v integrovanom textovom editore (file/new/M-file)
- alebo v ľubovoľnom inom textovom editore, uložiť ako textový súbor – *meno.m*,
- spustiť z riadkového režimu: `>>meno`
alebo z Matlab editora → *run*

Príklad programu



The image shows a MATLAB Editor window titled "Editor - C:\MATLAB71\work\objem_valca.m". The window has a menu bar (File, Edit, Text, Cell, Tools, Debug, Desktop, Window, Help) and a toolbar with various icons. The script content is as follows:

```
1 % vypocet objemu valca
2 % 1.1.2006, I.Sekaj
3
4 - v=input('vyska valca = ');
5 - r=input('polomer podstavy = ');
6
7 - V=pi*r^2*v;
8
9 - disp('Objem valca je');
10 - V
```

The status bar at the bottom indicates "script", "Ln 10", and "Col 2".

```
>> objem_valca
vyska valca = 2.5
polomer podstavy = 0.7
Objem valca je

V =

    3.8485

>>
```


Konštrukcie programovacieho jazyka Matlabu

- for, end
- if, else, elseif, end
- while, end
- break
- continue
- switch, case
- a iné ...

>> help lang

```
>> help lang
Programming language constructs.

Control flow.
  if                - Conditionally execute statements.
  else              - Execute statement if previous IF condition failed.
  elseif           - Execute if previous IF failed and condition is true.
  end               - Terminate scope of control statements.
  for               - Repeat statements a specific number of times.
  while             - Repeat statements an indefinite number of times.
  break             - Terminate execution of WHILE or FOR loop.
  continue          - Pass control to the next iteration of a loop.
  switch            - Switch among several cases based on expression.
  case              - SWITCH statement case.
  otherwise         - Default SWITCH statement case.
  try               - Begin TRY block.
  catch             - Begin CATCH block.
  return            - Return to invoking function.
  error             - Display message and abort function.
  rethrow           - Reissue error.

Evaluation and execution.
  eval              - Execute string with MATLAB expression.
  evalc             - Evaluate MATLAB expression with capture.
  feval             - Execute the specified function.
  evalin            - Evaluate expression in workspace.
  builtin           - Execute built-in function from overloaded method.
  assignin          - Assign variable in workspace.
  run               - Run script.
```

Niektoré typy podmienok pri vetvení programu

(priradenie: $x=2$)

podmienka rovnosti: $\text{if } x==2 \dots$

podmienka rôznosti / nerovnosti: $\text{if } x\neq 2 \dots$

podmienka väčší / menší: $\text{if } x>=2 \dots$

logické AND: $\text{if } x \& y$

logické OR: $\text{if } x | y$

negácia: $\text{if } \sim x$

Cyklus typu: for, if, elseif, else

```
for i=1:100
    x(i)=(i-1)/10;
end;
plot(x);
```

```
x(1)=0;
for i=2:100
    if i<=25
        x(i)=x(i-1)+0.1;
    elseif i<=50
        x(i)=x(i-1)-0.1;
    elseif i<=75
        x(i)=x(i-1)+0.1;
    else
        x(i)=x(i-1)-0.1;
    end;
end;
plot(x);
```

Cyklus typu: while

```
x=0;
k=1;

figure(1);
clf;          % wymazanie predchadzajuceho obrazku
hold on;      % zafixovanie predchadzajucich plotov

while x<10    % pokial nie je splnena podmienka, bezi cyklus
    x=x+rand;    % pripocitanie nahodneho cisla < 1
    plot(k,x,'*');
    pause(0.2); % pauza 0.2 s
    k=k+1;
end;
```

Break, continue

```
x=0;

figure(1); % inicializacia obr.1
clf;      % vymazanie predchadzajuceho obrazku
hold on;  % zafixovanie predchadzajucich plotov

for k=1:1000
    r=rand;
    if r<0.1 continue; end; % preskoci danu iteraciu
    x=x+r;    % pripocitanie nahodneho cisla < 1
    plot(k,x,'*'); % vykreslenie jedneho znaku *
    pause(0.2); % pauza 0.2 s
    if x>=10 break; end;    % ak je x>10 vyskoc z cyklu for
end;
```

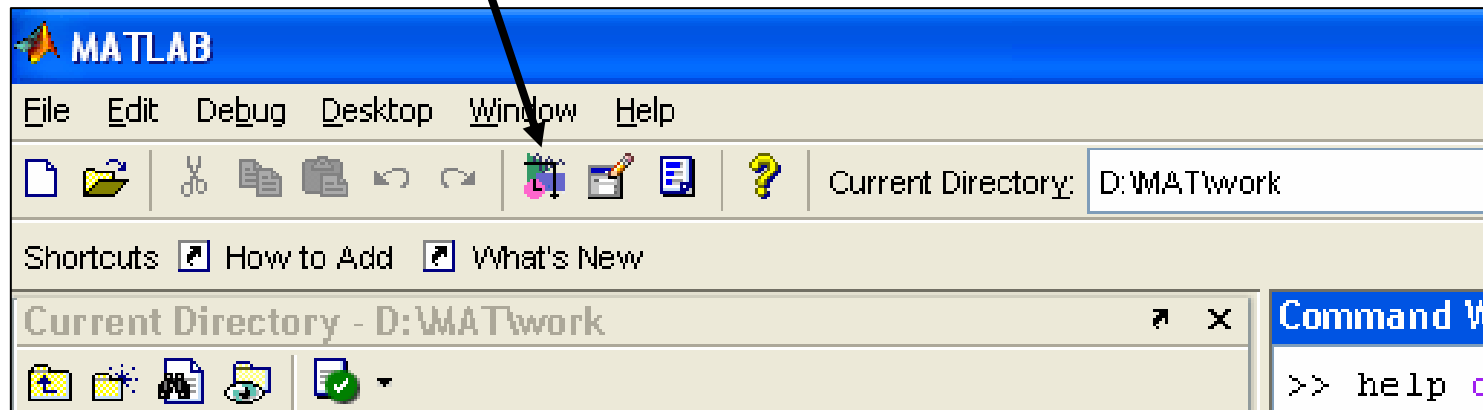
Switch, case, otherwise

```
d=zeros(1,10);
for n=1:100
    r=rand*10;
    z=ceil(r);
    switch(z);
        case 1
            d(1)=d(1)+1;
        case 2
            d(2)=d(2)+1;
        case 3
            d(3)=d(3)+1;
        case 4
            d(4)=d(4)+1;
        case 5
            d(5)=d(5)+1;
        case 6
            d(6)=d(6)+1;
        case 7
            d(7)=d(7)+1;
        case 8
            d(8)=d(8)+1;
        case 9
            d(9)=d(9)+1;
        otherwise
            d(10)=d(10)+1;
    end;
end;
bar([1 2 3 4 5 6 7 8 9 10],d);
d
```

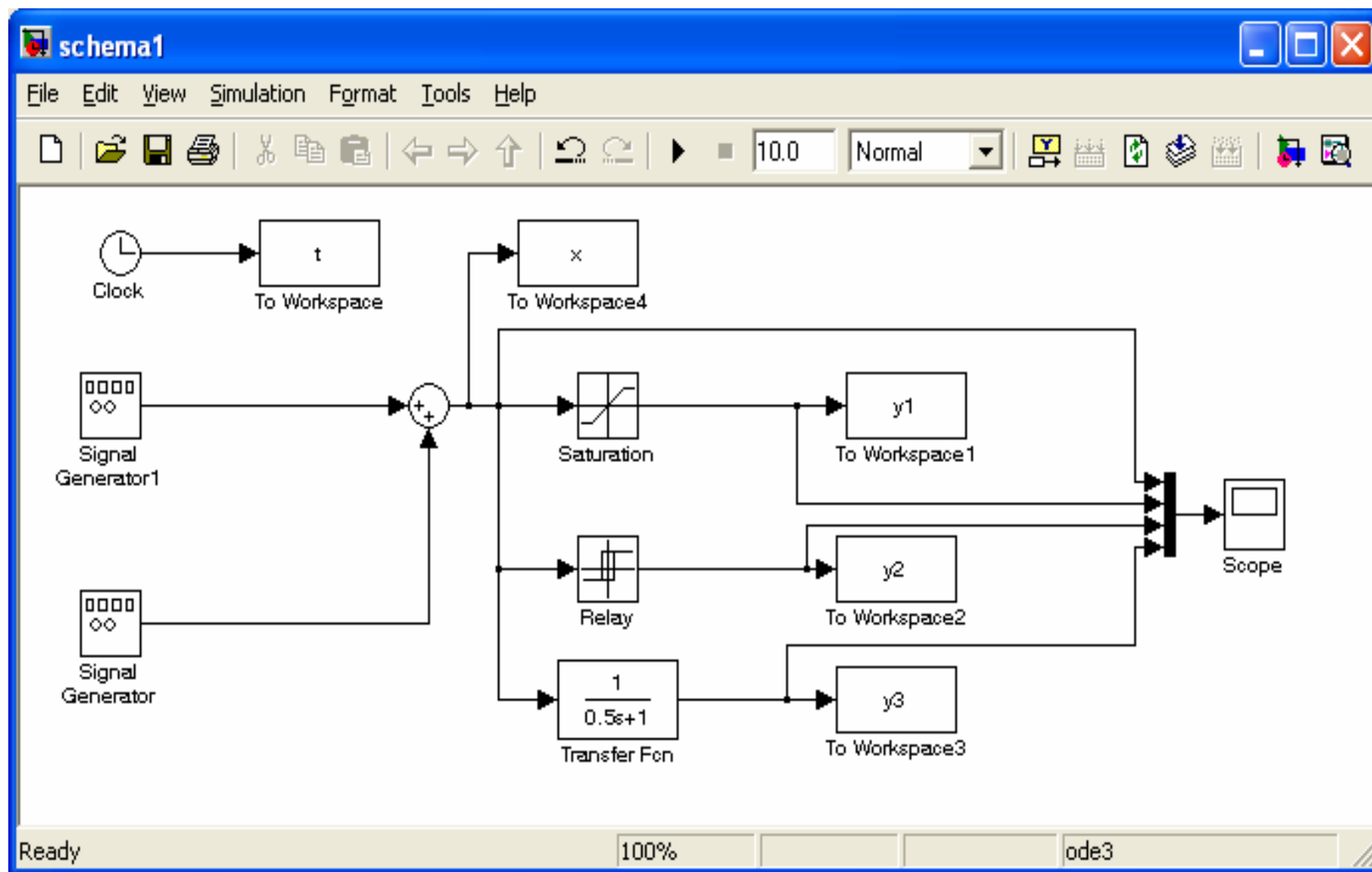
1.6 Simulink

- Samostatný nástroj Matlabu na tvorbu ODE modelov a realizáciu časových simulácií
- Rozsiahla knižnica rôznych typov objektov
- Vizualizácia
- spustenie: a) `>> Simulink`

b)



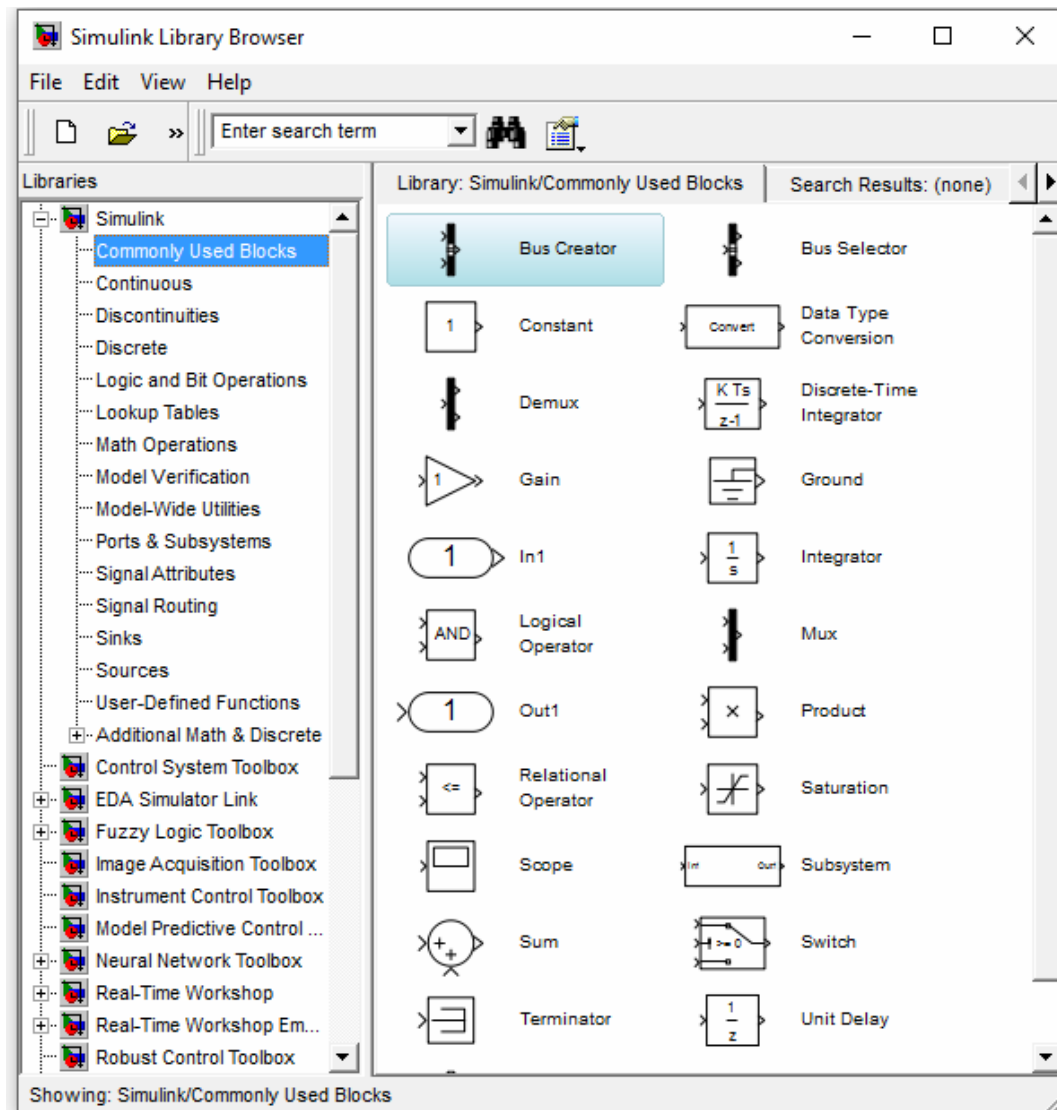
Príklad simulačného modelu v Simulinku



Simulink – knižnice

Bloky do modelu
pridávame cez
drag & drop

- Commonly used blocks
- Continuous
- Sources (Signal builder)
- Sinks (From / To workspace, on the run grafy - Scope)
- Math operations (+-* /)
- Signal routing – mux / demux



1.7 Iné nástroje Matlabu

- **GUI – grafické používateľské rozhranie**
- **VR-Toolbox – virtuálna realita**
- **RT-Toolbox - pripojenie na reálny svet (real-time)**
- **Matlab Compiler – prekladač z MATLAB-u do C++**
- **Matlab Web Server – sieťové aplikácie**
- **Paralell computing – rozdelenie výpočtu na viac PC**
- **a iné ...**

Príklad používateľom vytvoreného GUI

Manipulácia s kontajnermi

Vykladací plán

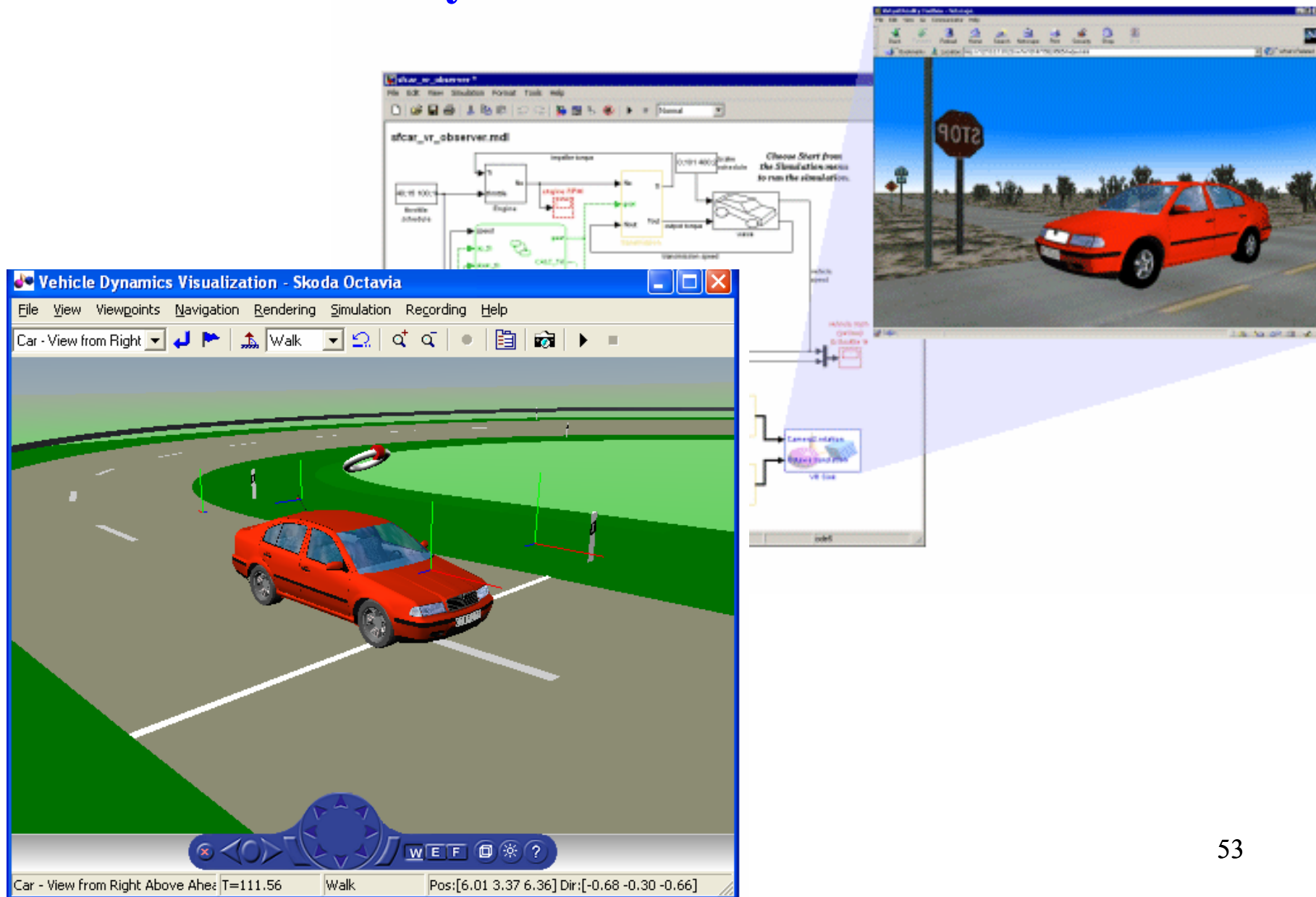
Pozícia vo vlaku:	1	2	3	4	5	6	7	8	9	10
Pozícia v sklade:	8	6	3	1	11	2	10	4	7	9

Počet krokov: 1. žeriav 2. žeriav Spolu

Pohyb žeriavov

	1	2	3	4	5	6	7	8	9	10
žeriav1:	3	6	1	2	0	0	0	0	0	0
žeriav2:	0	0	0	0	8	7	1	5	4	2
Zastavenie žeriav1	0	0	0	2	2	0	13	0	0	0
Zastavenie žeriav2	0	0	9	0	0	7	0	0	14	0

Virtual Reality Toolbox



1.8 Neural network toolbox

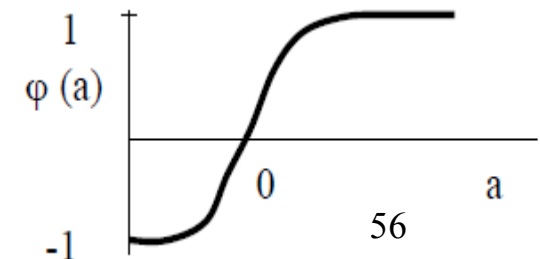
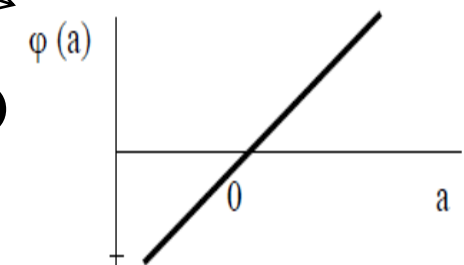
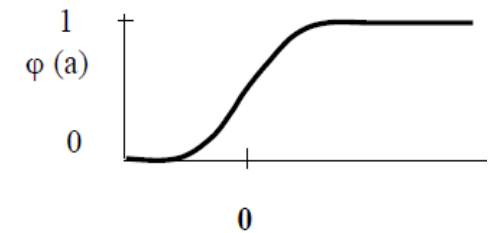
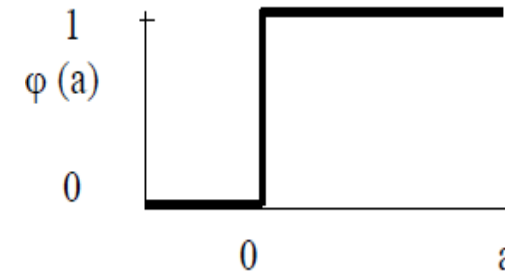
- **Knižnica pre prácu s neurónovými sieťami (ver. 7.0 - Matlab R2010b)**
- **Základné funkcie pre NS (vytváranie štruktúry pre rôzne typy NS, učenie, simulácie výstupu, zobrazenie chyby učenia, procesu trénovania, ...)**
- **Grafické prostredie pre NS– spustenie príkazom
>> nntool**
- **Pomoc zavoláme príkazom >> help nnet**
- **Dopredné siete (feedforward) >> feedforwardnet
(Viacvrstvová Perceptrónová sieť – MLP)**
- **RBF sieť >> newrb**
- **Klasifikácia, rozpoznávanie >> patternnet (nprtool)**
- **Modelovanie, aproximácia >> fitnet (nftool)**

Funkcie na vytvorenie štruktúr NS NNET toolboxu

- **network** - Vytvorenie vlastnej neurónovej siete
- **feedforwardnet** - Vytvorenie doprednej neurónovej siete.
- **linearlayer** - Vytvorenie jednovrstvovej lineárnej neurónovej siete
- **perceptron** - Vytvorenie jednovrstvovej perceptrónovej siete.
- **newrb** - Návrh RBF siete
- **newhop** - Vytvorenie Hopfieldovej rekurentnej siete
- **selforgmap** - Vytvorenie samoorganizujúcej sa mapy (Kohonenova sieť)
- **cascadeforwardnet** - Vytvorenie kaskádnej doprednej siete
- **elmannet** - Vytvorenie Elmanovej rekurentnej siete.
- **timedelaynet** - Vytvorenie doprednej siete s oneskorením na vstupe.
- **fitnet** - Vytvorenie neurónovej siete na aproximáciu, modelovanie.
- **patternnet** - Vytvorenie siete na klasifikáciu, rozpoznávanie.
- **newlvq** - Create a learning vector quantization network.
- **narnet, narxnet** - Vytvorenie siete na modelovanie dyn. systémov.

Transformačné (aktivačné) funkcie NNET toolboxu

- **compet** - konkurenčná funkcia.
- **hardlim** - skoková funkcia (0,1).
- **hardlims** - symetrická skoková funkcia (-1,1)
- **logsig** - sigmoida logistická funkcia (0,1)
- **poslin** - kladná lineárna funkcia. $(0, \infty)$
- **purelin** - lineárna funkcia. $(-\infty, \infty)$
- **radbas** - Radial basis funkcia (0,1)
- **satlin** - saturevaná kladná lineárna funkcia. (0,1)
- **satlins** - symetricky saturevaná lineárna funkcia. (-1,1)
- **softmax** - Soft max funkcia.
- **tansig** - hyperbolický tangens (-1,1)
- **tribas** - jednoduchá trojuholníková funkcia. (0,1)

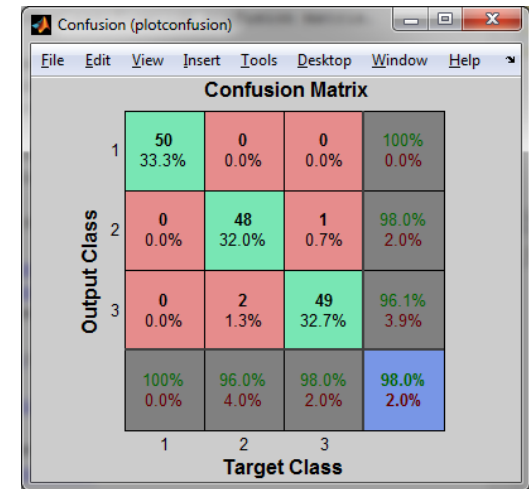
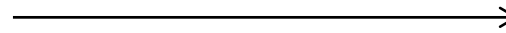


Trénovacie funkcie NNET toolboxu

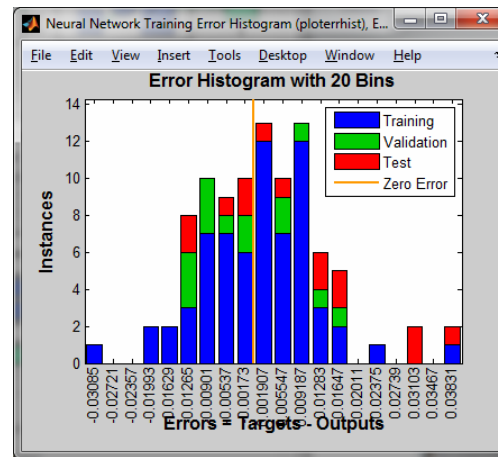
- **traingd** - Gradient descent backpropagation.
- **traingdm** - Gradient descent with momentum backpropagation.
- **traingda** - Gradient descent with adaptive lr backpropagation.
- **traingdx** - Gradient descent w/momentum & adaptive lr backpropagation.
- **trainlm** - Levenberg-Marquardt backpropagation.
- **trainoss** - One step secant backpropagation.
- **trainrp** - Resilient backpropagation (Rprop).
- **trainbfg** - BFGS quasi-Newton backpropagation.
- **trainscg** - Scaled conjugate gradient backpropagation.
- **traincgb** - Powell-Beale conjugate gradient backpropagation.
- **traincgf** - Fletcher-Powell conjugate gradient backpropagation.
- **traincgp** - Polak-Ribiere conjugate gradient backpropagation.

Funkcie pre zobrazenie v NNET toolboxe

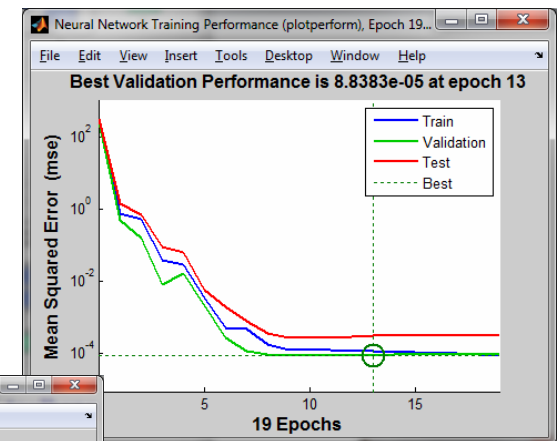
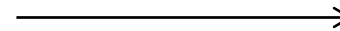
- **plotconfusion** - Zobrazenie klasifikačnej matice.



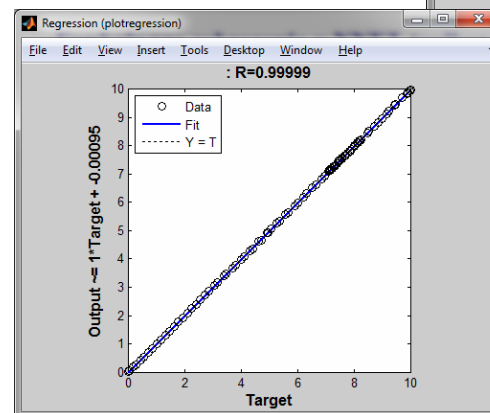
- **ploterrhist** - Zobrazenie histogramu pre chybu siete



- **plotfit** - Zobrazenie chyby pri aproximácii.
- **plotperform** - Zobrazenie chyby siete v priebehu tréovania



- **plotregression** - Zobrezenie lineárnej regresie medzi výstupnými datami a výstupom NS



Vybrané funkcie NNET toolboxu

- **patternnet** – vytvorenie doprednej NS na klasifikáciu

net = patternnet ([S1 S2...SN], BTF)

Výstupná
štruktúra NS

Počty neurónov vo vrstvách NS

- 1. prvok vektora je počet neurónov v 1. skrytej vrstve
- posledný prvok vektora je počet neurónov v poslednej skrytej vrstve

Trénovacia
funkcia

- default **trainscg**

- **fitnet** – vytvorenie doprednej NS na aproximáciu

net = fitnet ([S1 S2...SN], BTF)

Vytvorenie NS na klasifikáciu s jednou skrytou vrstvou s 15 neurónmi

>> net = patternnet(15);

Aktivačné funkcie sú „tansig“, trénovací algoritmus je „trainscg“

Vytvorenie NS na aproximáciu s dvoma skrytými vrstvami s 10 a 8 neurónmi

>> net = fitnet([10 8]);

**Aktivačné funkcie sú „tansig“ skryté vrstvy a „purelin“ výstupná vrstva ,
trénovací algoritmus je „trainlm“**

Vybrané funkcie NNET toolboxu

- **newrb** - vytvorenie RBF siete

[net,tr] = newrb(P,T,GOAL,SPREAD,MN,DF)

Výstupná
štruktúra NS

Výstupné
data NS

Matica vstupných dát siete
riadky matice = data pre vstupy NS

Koef. naťahovania
RB funkcií
- default 1

Ukončovacia
chyba učenia
- default 0

Max. počet neurónov
- default je počet dát

Počet neurónov
pridaných medzi
zobrazením
- default 25

- **perceptron** - vytvorenie jednovrstvovej perceptrónovej siete

net = perceptron(S)

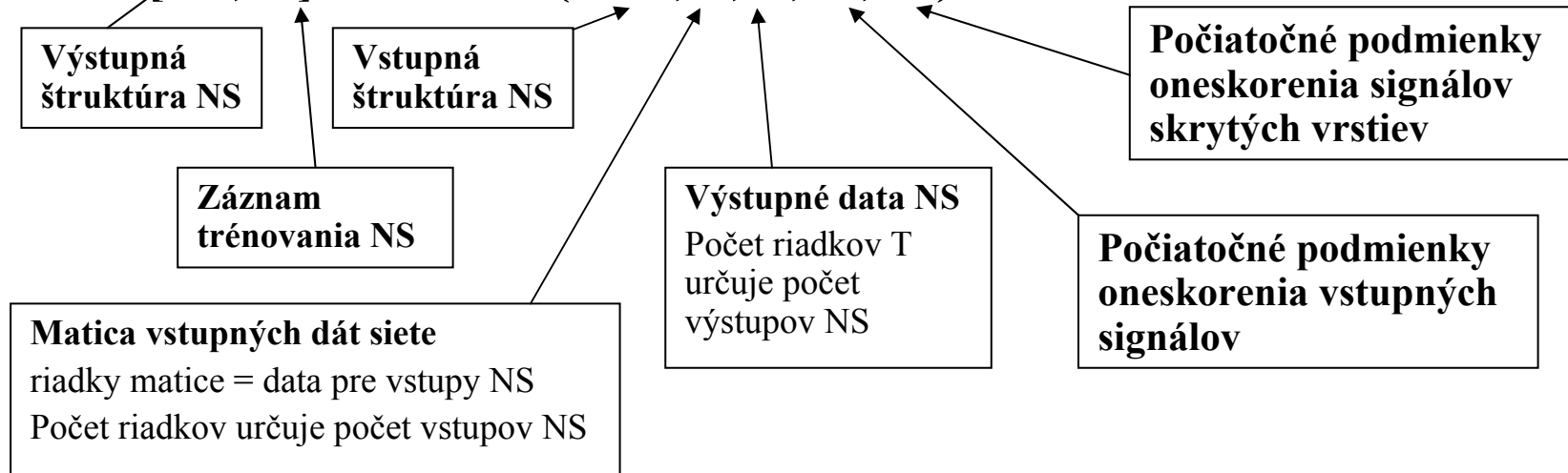
Výstupná
štruktúra NS

Počet
neurónov

Vybrané funkcie NNET toolboxu

train – tréovanie (učenie) NS

[net, tr] = train(NET, P, T, Pi, Ai)



net.trainParam.goal - Ukončovacia podmienka na chybu.

net.trainParam.show - Frekvencia zobrazovania priebehu chyby tréovania

net.trainParam.epochs - Max. počet tréovacích cyklov.

net.trainParam.min_grad - Ukončovacia podmienka na gradient chyby.

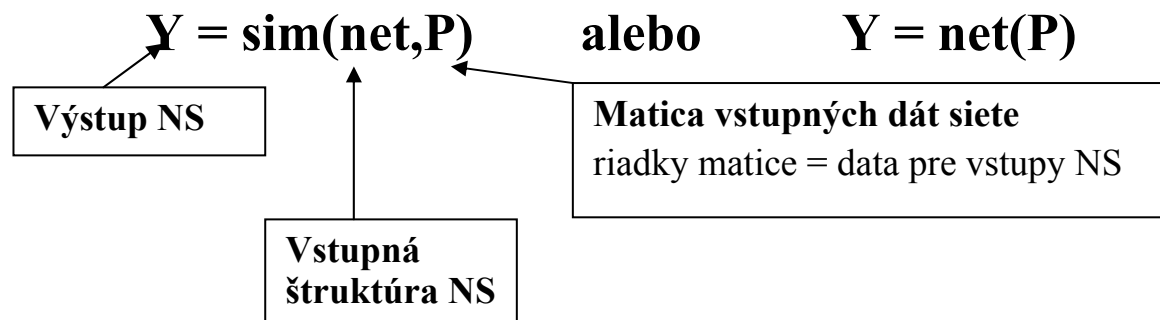
net.trainParam.mc - Momentum konštanta

net.performFcn - Kriteiálna (chybová) funkcia 'mse' alebo 'sse'

net.divideFcn - Funkcia na rozdelenie dát 'dividettrain' alebo 'dividerand'

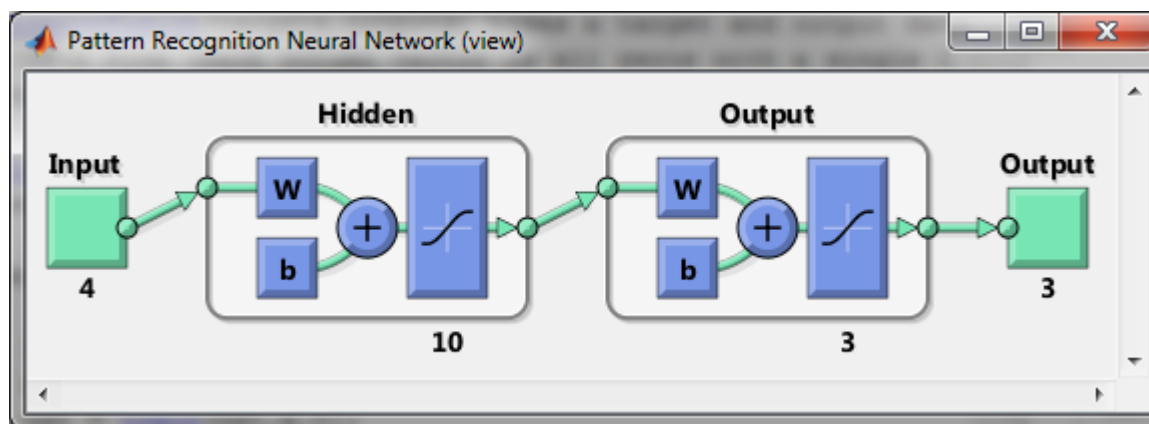
Vybrané funkcie NNET toolboxu

sim – simulácia výstupu NS



view – zobrazenie štruktúry NS

view(net)



Spôsoby generovania (rozdeľovania) tréningových, validačných a testovacích dát

Typ funkcie na generovanie dát

net.divideFcn

```
net.divideFcn='dividerand';    % náhodné rozdelenie  
net.divideFcn='divideblock';   % rozdelenie po blokoch dát za sebou  
net.divideFcn='divideint';     % je použitá každá n-tá vzorka  
net.divideFcn='dividetrain';   % všetky dáta sú iba tréningové
```

% parametre rozdelenia dát

```
net.divideParam.trainRatio=0.8;  
net.divideParam.valRatio=0.1;  
net.divideParam.testRatio=0.1;
```

Spôsoby generovania (rozdeľovania) trénovacích, validačných a testovacích dát

Typ funkcie na generovanie dát

net.divideFcn

```
net.divideFcn='divideind';    % indexové rozdelenie
```

```
% parametre rozdelenia dát
```

```
net.divideParam.trainInd=1:2:n;
```

```
net.divideParam.valInd=2:2:n2;
```

```
net.divideParam.testInd=n2+1:2:n;
```


Príklad klasifikácie iris kvetov do 3 skupín

```
[X,P]=iris_dataset; % načítanie dát

net = patternnet(10); % vytvorenie štruktúry MLP siete, 1 skrytá vrstva, 10 neurónov

% nastavenie parametrov tréovania
net.performFcn = 'sse'; % Suma štvorcov odchýliek
net.trainParam.goal = 0.000001; % Ukončovacia podmienka na chybu SSE.
net.trainParam.epochs = 1000; % Max. počet tréovacích cyklov.
net.trainParam.min_grad=1e-12; % Ukončovacia podmienka na min. gradient

% náhodné rozdeľovanie dát na (tréovanie 80 %, validácia 10%, test 10%)
net.divideFcn='dividerand'; net.divideParam.trainRatio=0.8;
net.divideParam.valRatio=0.1; net.divideParam.testRatio=0.1;

net = train(net,X,P); % tréovanie siete

Y=sim(net,X) % simulácia výstupu NS
classes = vec2ind(Y) % zaradenie do tried
plotconfusion(P,Y); % porovnanie klasifikácie pomocou NS s reálom
```

Príklad aproximácie nelineárnej funkcie $y=f(x)$

```
[x,y]=simplefit_dataset;      % načítanie dát

net = fitnet(12); % vytvorenie štruktúry MLP siete, 1 skrytá vrstva, 12 neurónov

% nastavenie parametrov tréovania
net.trainParam.goal = 1e-7;      % Ukončovacia podmienka na chybu MSE.
net.trainParam.epochs = 100;     % Max. počet tréovacích cyklov.

% rozdeľovanie dát – každá n-tá vzorka (tréovanie 80 %, validácia 10%, test 10%)
net.divideFcn='divideint'; net.divideParam.trainRatio=0.8;
net.divideParam.valRatio=0.1; net.divideParam.testRatio=0.1;

net = train(net,x,y);    % tréovanie siete

yn=sim(net,x);          % simulácia výstupu NS

figure
plot(x,y, 'ro',x,yn, 'b*')      % porovnanie aproximácie pomocou NS s reálom

plotfit(net,x,y);    % porovnanie aproximácie pomocou NS s reálom cez GUI
```