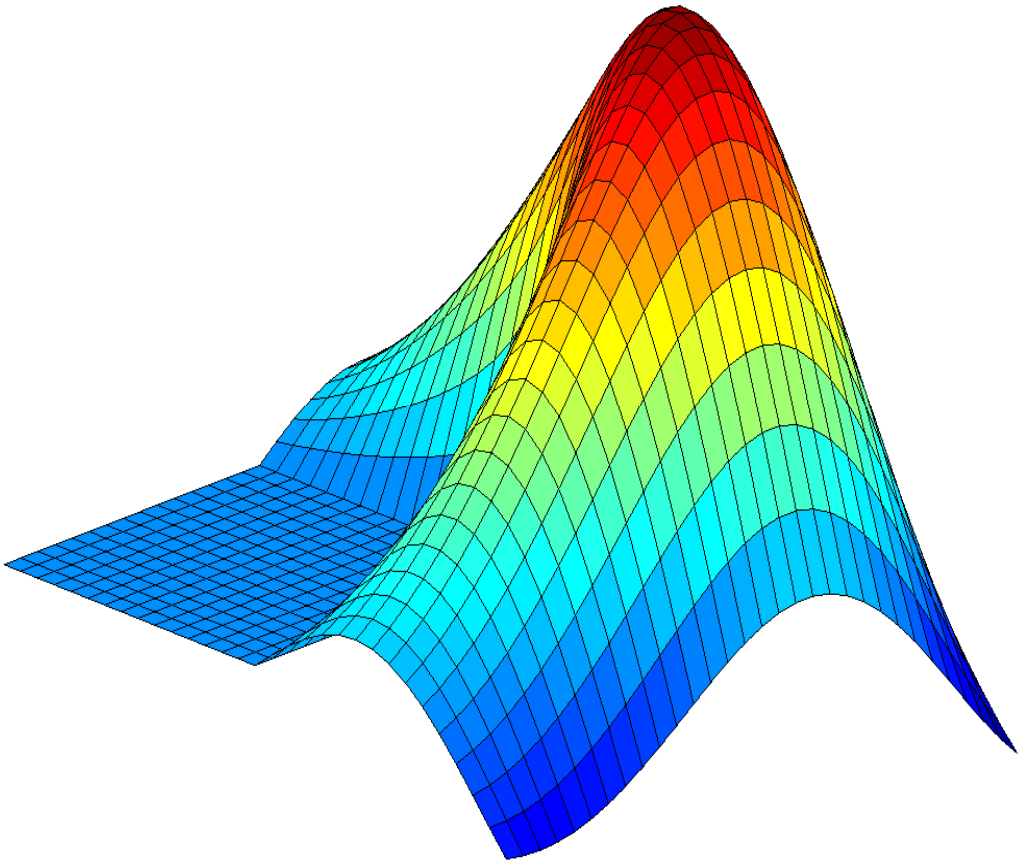
MATLAB
ver. 7.x

Úvod do práce s programem



Určeno pouze pro vnitřní potřebu semináře

Část připravované příručky základní práce s Matlabem.

1.0 Úvod

Následující text je určen pouze pro interní potřebu semináře a jeho úkolem je seznámit uživatele se základními funkcemi a ovládáním MATLABu. Uživatel – začátečník má možnost seznámit se s programem a s jeho strukturou. Je popisována verze 7 (R14). Texty jsou soustředěny do čtyř samostatných celků, které uživatele postupně seznamují s možnostmi programu. Vzhledem k rozsahu MATLABu jsou podrobně probírány pouze některé jeho části, a to ty, které nejsou příliš složité, a které považujeme za důležité uživateli při prvním seznámení s MATLABem sdělit. Texty nemají při práci sloužit jako referenční manuál nebo podrobný návod, ale měly by být při práci určitým vodítkem. Účelem je vytvořit pro uživatele stručný přehled základních možností MATLABu a jeho použití. Tímto také chceme uživateli usnadnit orientaci v tak rozsáhlém programu a ulehčit další samostudium oblasti zvolené podle jeho profesních potřeb.

Každá ze čtyř částí obsahuje vybrané funkce podle tématu s uvedením jednoduchých příkladů. Na závěr části je vždy probrané téma stručně shrnuto.

První část seznamuje uživatele s desktopem MATLABu, jeho možnostmi a nastavením, s orientací na disku a nebo v nápovědě. Dále popisuje práci v příkazovém řádku a zápis jednoduchého výrazu nebo proměnné. Vedle zobrazování v příkazovém okně (formát zobrazení) je uveden zápis komplexního čísla a související funkce. Následují zvláštní typy proměnných, speciální znaky. Hlavní část je věnována maticím, na jejichž principu je činnost MATLABu založena. K nejdůležitějším patří zápis a operace s maticemi, jejich indexování.

Druhá část v úvodu popisuje práci s řetězci a vzájemný převod mezi textem a čísly. Hlavní část této sekce je věnována 2D, 3D grafice a práci grafy. Uživatel se seznámí s tím, jak zajistit popis grafu, jak zakreslit více funkcí do jednoho grafu nebo jak vytvořit na jeden obrázek více grafů. Uvedeny jsou také speciální typy grafů. Je vysvětleno, jak s grafy zacházet z příkazové řádky i z grafického rozhraní (vysvětlení objektu *figure*). Dále následuje stručný úvod do zobrazování 3D těles, jejich zobrazení v prostoru, jak pracovat s rastrovými soubory, využití barevné škály a typy grafických formátů.V závěru této části jsou uvedeny možnosti tisku, ovládání z příkazové řádky a zápis do různých grafických formátů s nastavením jejich rozlišení.

Ve *třetí části* se uživatel seznamuje se soubory a importem a exportem textových i binárních dat do MATLABu. V úvodu do programování je uživatel seznámen s používanými relačními a logickými operátory. Použití operátorů je předvedeno při práci s polem a s maticí, je uvedeno použití funkcí s výstupem logických dat při testování a vyhledávání jednotlivých prvků matice,

vyhledávání minimálních a maximálních hodnot prvků. Dále se uživatel seznámí s prostředky pro řízení chodu programu.

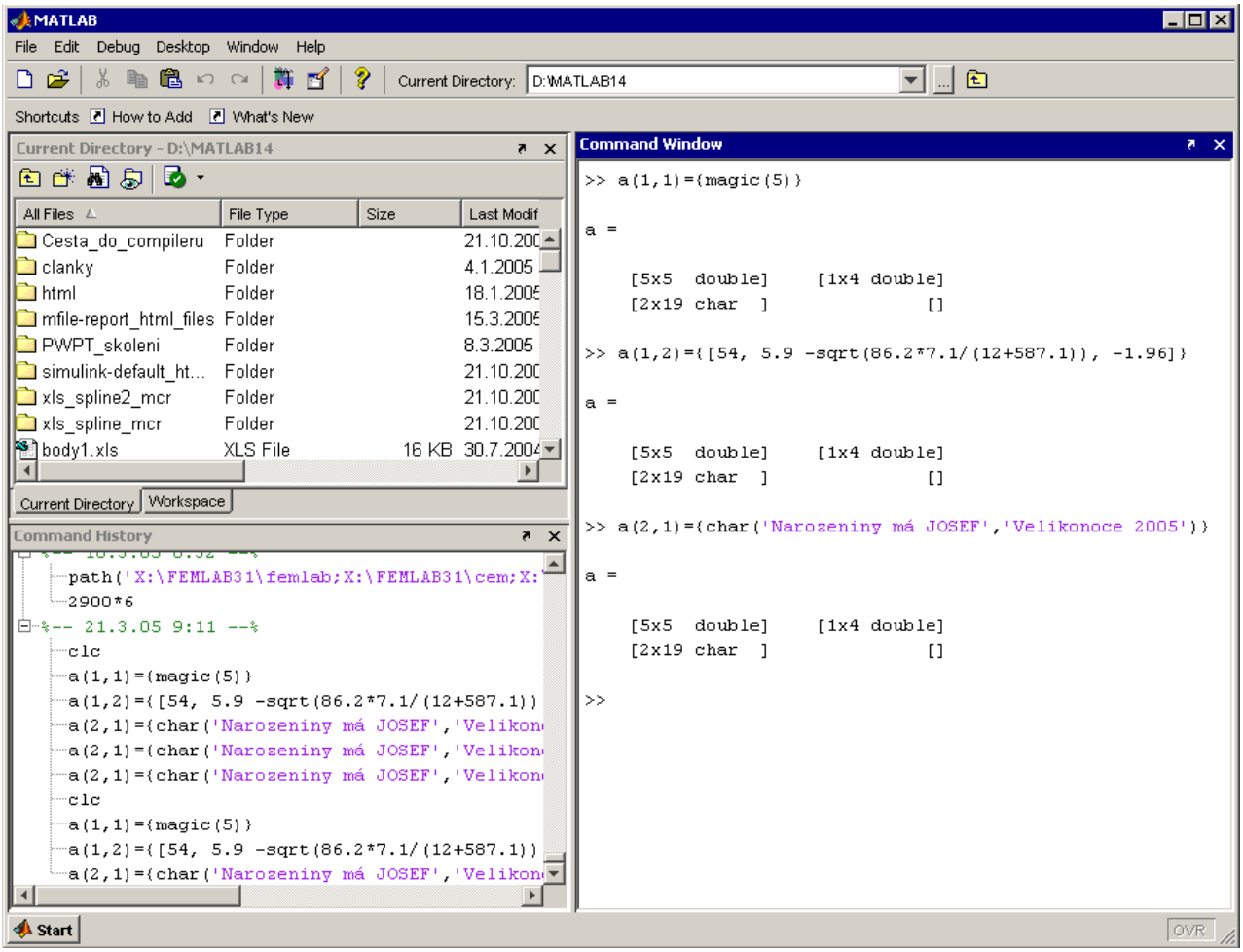
Čtvrtá část textu poskytne uživateli základy při vytváření jednoduchých skriptů (M-souborů), stavbě funkcí a jejich volání. V další části se seznámíme se speciálními datovými typy jako jsou struktury, buňky a pole buněk, řídké matice a celočíselné datové typy. Naznačena jsou také vícerozměrná pole. V závěru se uživatel seznamuje s vyjímečnými stavy při výpočtu.
Jako speciální tématický celek jsou zařazeny funkce pro práci s polynomy, prokládání křivek a výpočet kořenů polynomu.

2.0 Prostředí MATLAB desktop

Desktop MATLABu se skládá ze 3 hlavních oken :

- Command Window
- Command History
- Workspace, Current Directory

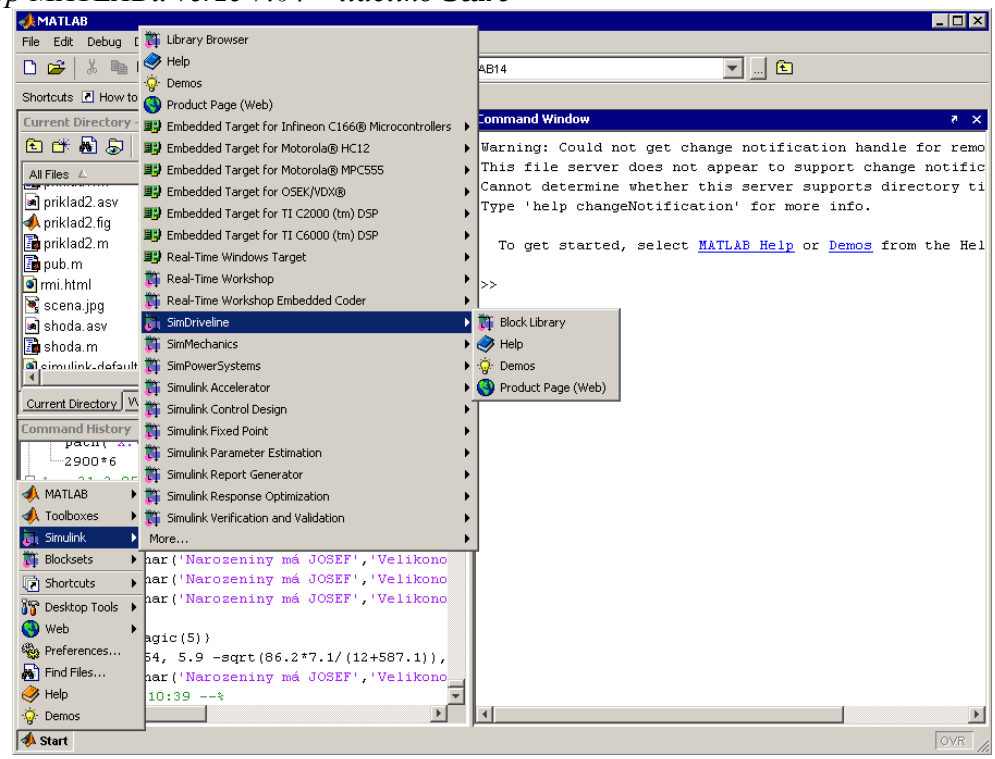
Desktop MATLABu verze 7.04 (R14)



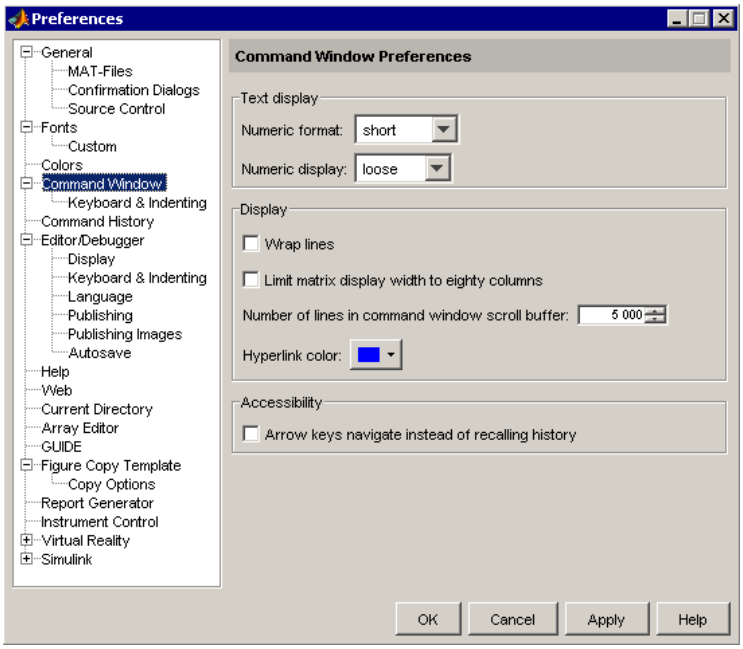
Kombinace jednotlivých oken v desktopu je libovolná, s každým oknem lze pracovat samostatně a do desktopu lze zakomponovat také další okna. Uživatelské nastavení může být provedeno z roletového menu **View** v horní liště základního okna v položce **Desktop Layout** nebo kliknutím myši na šipku v pravém horním rohu každého okna. Nastavení oken do defaultního uspořádání lze opět z rolety **View → Desktop Layout → Default**.

Způsob práce v desktopu názorně popisuje několk animovaných demo programů v sekci *Desktop Tools and Development Environment.* Spuštění přes *Command Window* příkazem *demo* (nebo z roletového menu **Help**, popř. tlačítkem START v položce Demos)

*Desktop MATLABu verze 7.04 – tlačítko **Start***

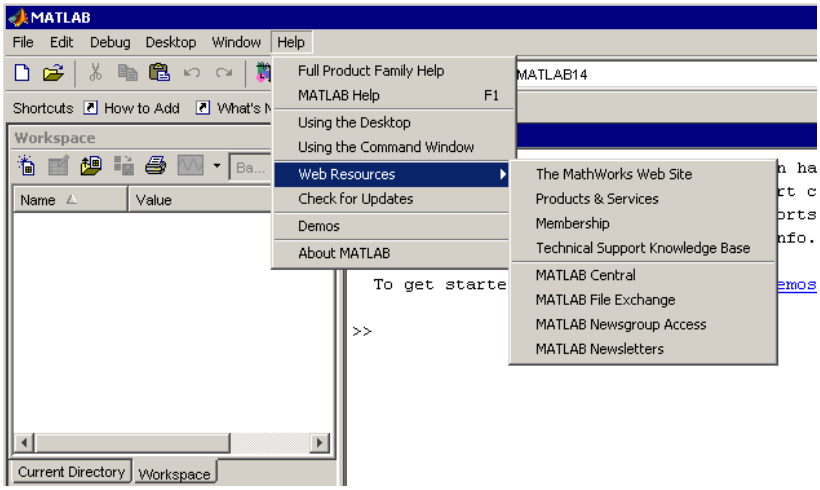


Preference:

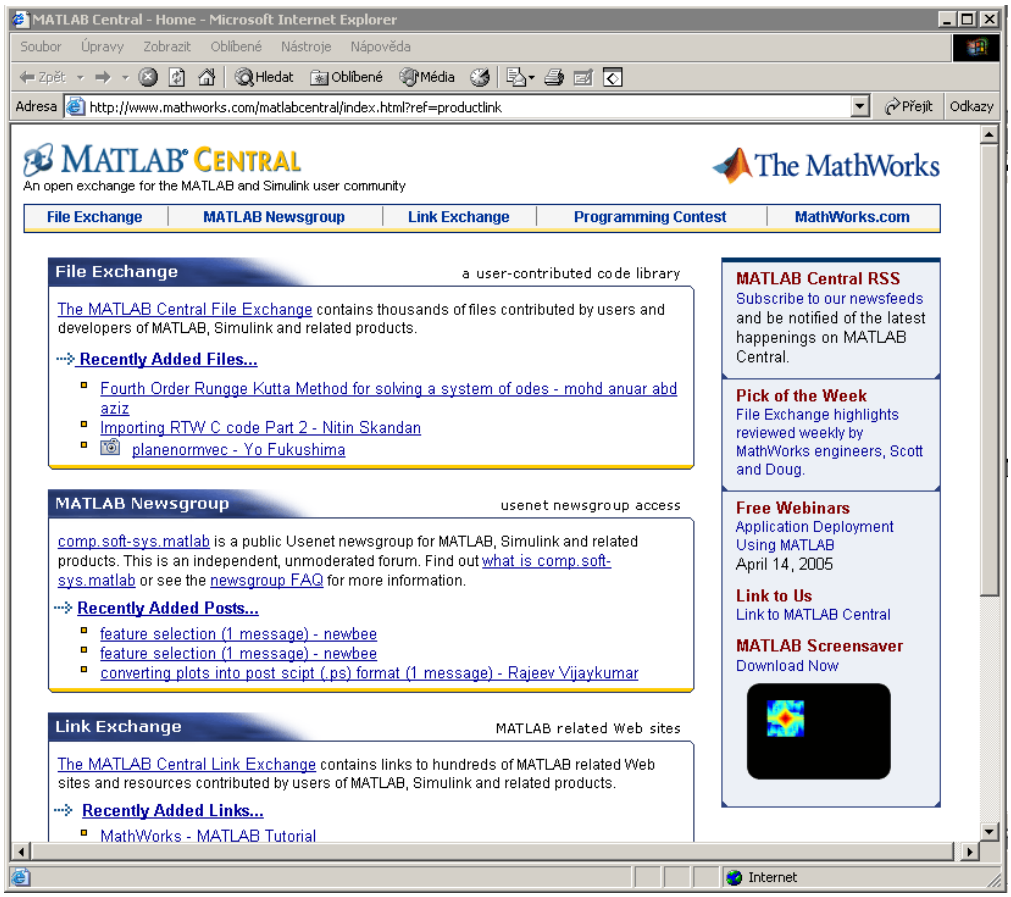


V dialogu Preferences je možné nastavit prostředí MATLABu a jeho součástí jako je Command Window, Command History, Current Directory, textový editor, fonty a další.

Internet a MATLAB:



MATLAB Central:



Práce v příkazovém řádku - Command window:

Ruční zadávání příkazů, spouštění M-souborů a jednotlivých funkcí MATLABu , výpisy proměnných, volání helpu.

Užitečné příkazy:

whos – výpis proměnných (textová kopie Workspace),
who – zjednodušený výpis proměnných, pouze jejich název
clear <*proměnná*>, clear all – mazání proměnných z prostření workspace
helpwin, help <*funkce*>**, doc** <*funkce*> - informace o zadaných funkcích.
pwd – aktuální adresář
dir, ls – výpis obsahu adresáře
format – zobrazování výstupů v příkazovém okně
lookfor - vyhledává zadané klíčové slovo ve všech nápovědách
workspace - zobrazí okno pracovního prostoru
cd - mění pracovní adresář
copyfile - kopíruje soubor nebo adresáře
movefile - přesouvá soubory nebo adresáře
mkdir - vytváří nový adresář
rmdir - odstraňuje adresář
what - vrací seznam specifikovaných souborů v aktuálním adresáři
type - vypíše obsah M-souboru

web - zobrazí HTML soubor nebo zadanou HTTP adresu

Celkový výpis příkazů do příkazového řádku je:
```
>> help matlab/general
```

3.0 Základní vlastnosti prostředí v Command Window

Prompt v příkazovém okně, který představuje připravenost MATLABu k činnosti je:
```
>>
```
Zápis jednoduchého výrazu:
```
>>15*4 + 50/2 - 10
ans =
       75
```

Výsledek se ukládá do implicitní proměnné **ans**, se kterou lze dále pracovat. Využití této proměnné však může později vést k nežádoucím změnám v dalších výpočtech, protože do **ans** může být uložena jakákoliv hodnota.
Přiřazování hodnot do proměnných:
```
>>pocet=25
pocet =
       25
>>strana=32
strana =
       32

>>prumer=pocet+strana/2
prumer =
       28.5
```

Základní matematické operace v MATLABu

Operace	*Symbol*	*Příklad*
Sčítání	+	10 + 15.25
Odčítání	-	18.8 - 10
Násobení	*	25.4*0.303
Dělení	/ nebo \	24.8/3 nebo 3\24.8
Umocňování	^	2^5

Operace ve výrazech jsou prováděny zleva doprava, a umocňování má nejvyšší prioritu. Následuje se stejnou prioritou násobení a dělení, součet a odečítání. Změnu priorit lze provést použitím závorek.
Protože se jednotlivé příkazy ukládají do historie, je možné v řádcích listovat použitím šipek na klávesnici ↑ a ↓. Historii jednotlivých příkazů lze také sledovat v okně *Command History*, které je v implicitním uspořádání umístěno na levé straně celkového okna desktopu MATLABu. V této historii můžeme listovat a myší vybraný příkaz spustit kliknutím na pravé tlačítko. Výběrem myší a přidržením jejího pravého tlačítka lze přenést vybraný příkaz do příkazového okna.
Příkaz *diary* umožňuje zapnout nebo vypnout historii, popř. zapsat historii do definovaného souboru. Implicitně se historie zapisuje do souboru '`diary`'.

3.1 Proměnné v prostředí MATLABu

Proměnná se může skládat až z 31 znaků, v názvu se rozlišují malá a velká písmena (je case-sensitive), jména proměnných musí začínat písmenem a nesmí obsahovat tečku „ **.**“
Seznam některých speciálních proměnných:

ans	default proměnná
beep	zvuk
pi	Ludolfovo číslo
Inf (inf)	nekonečno 1/0
NaN (nan)	Not – a – Number 0/0
i nebo *j*	složka komplexního čísla odmocnina z (-1)
realmin	nejmenší použitelné kladné reálné číslo
realmax	největší použitelné kladné reálné číslo
bitmax	největší použitelné kladné celé číslo
varargin	proměnné vstupující do funkce (pole buněk)
varargout	proměnné vystupující z funkce (pole buněk)
nargin	počet proměnných vstupujících do funkce
nargout	počet proměnných vystupujících z funkce

Další nejpoužívanější znaky v prostředí MATLABu jsou:

. (tečka)	desetinná tečka, oddělovač proměnných ve struktuře
; (středník)	potlační následného výpisu v příkazovém okně po volání funkce, M-souboru nebo přiřazení hodnoty do proměnné
: (dvojtečka)	např. rozsah hodnot nebo indexů v poli, v matici
% (procento)	označení komentáře na řádce
… (tři tečky)	rozdělení dlouhého řádku
, (čárka)	možnost oddělení příkazů na řádku (výpis není potlačen), parametrů
!	spouštění systémových příkazů
[ ]	ohraničuje obsah definovaného pole nebo matice
{ }	ohraničuje obsah definované buňky
( )	použití při indexování polí, obsahy
' (apostrof)	transpozice matice, ohraničení textové proměnné

Příkazy lze řadit za sebou a oddělovat je čárkou (,) nebo středníkem (;). Čárka nepotlačuje výpis.

```
>> x1 = 36, x2 = 51; y = 1.5
x1 =
  36
y =
1.5000
```

Pokud je příkaz dlouhý, lze rozdělit na více řádků použitím tří teček (**…**)

```
>>y  = sin(x1) + ...      % Rozdělení řádku
cos(x2)
y =
-0.2496
```

3.2 Zobrazování čísel v příkazovém řádku

MATLAB pracuje ve *dvojnásobné přesnosti (double precision)*. Celá čísla (integer) MATLAB zobrazuje jako celá, reálná čísla jsou zobrazována s desetinou tečkou, implicitně na 4 desetinná místa (formát short). Zobrazovaná čísla s přesností nesouvisí. Výpis formátu čísel na obrazovku a regulace mezer mezi řádky umožňuje příkaz format:

```
>>format short      (zobrazovaný formát čísla na 5 číslic)
>>format compact    (v příkazovém okně potlačuje při odezvě volný řádek)
>>format loose      (zpět na dlouhý výpis)
```

Formáty zobrazení

Příkaz	*Příklad – číslo pi*	**Popis**
format short	3.1416	5 číslic
format short eng	3.1416e+000	Min. 5 číslic + exponent (násobek tří)
format long	3.14159265358979	16 číslic
format long eng	3.14159265358979e+000	16 číslic + exponent (násobek tří)
format short e	3.1416e+000	5 číslic + exponent
format long e	3.14159265358979e+000	16 číslic + exponent
format short g	3.1416	nejlepší z *short* nebo *short e*
format long g	3.14159265358979	nejlepší z *long* nebo *long e*
format hex	400921fb54442d18	nexadecimální
format bank	3.14	2 číslice
format +	+	Kladné (+), záporné (-) nebo 0
format rat	355/113	Racionální přiblížení
format debug	Structure address = 2937f38 m = 1 n = 1 pr = 434d4908 pi = 0 3.1416	Informace o vnitřním uložení čísla s konečným zobrazením *short g*

Zobrazení všech operátorů v MATLABu:
>> help /

3.3 Práce s komplexními čísly

Zadání komplexního čísla do příkazového řádku:

```
>>k2 = 9.8 + 10i

k2 =
8.0000 +10.0000i
```

nebo

```
>>k2 = 9.8 + 10j
k2 =
8.0000 +10.0000j
```

Absolutní hodnota komplexního čísla je délka průvodiče v komplexní rovině:

```
>>abs(k2)
ans =
14.0014
```

Úhel mezi průvodičem a reálnou osu v komplexní rovině:

```
>> angle(k2)
ans =
0.7955        (úhel v radiánech)
```

Zobrazení komplexně sdruženého čísla (conjugate):

```
>>conj(k2)
ans =
9.8000 -10.0000i
```

Zobrazení reálné a imaginární části komplexního čísla – **`real(k2), imag(k2)`**

3.4 Funkce používané v MATLABu

MATLAB obsahuje řadu funkcí, které jsou rozděleny do následujících skupin:
(Část výpisu po zadání příkazu `help`)

```
>> help
HELP topics

matlab\general    - Příkazy všeobecného použití.
matlab\ops        - Operátory a speciální znaky.
matlab\lang       - Příkazy používané při programování v MATLABu.

matlab\elmat      - Elementární matice a práce s maticemi.
matlab\elfun      - Elementární matematické funkce.
matlab\specfun    - Specializované matematické funkce.
matlab\matfun     - Maticové funkce - lineární algebra.
matlab\datafun    - Analýza dat a Fourierovy transformace.
matlab\polyfun    - Interpolace a polynomy.
matlab\funfun     - Funkce funkce a řešiče ODE.
matlab\sparfun    - Řídké matice.
matlab\scribe     - Anotace a  editace vykreslování
matlab\graph2d    - Grafy ve 2-D.
matlab\graph3d    - Grafy ve 3-D.
matlab\specgraph  - Specializované grafy.
matlab\graphics   - Handle Graphics.
matlab\uitools    - Prostředky pro tvorbu uživatelského grafického rozhraní.
matlab\strfun     - Řetězce a práce s nimi.
matlab\imagesci   - Obrázky a vstupy/výstupy vědeckých dat
matlab\iofun      - Soubory - vstupy/výstupy.
matlab\audiovideo - Podpora vudio a video
matlab\timefun    - Čas a datum.
matlab\datatypes  - Typy dat a struktury.
matlab\verctrl    - Kontrola verzí.
matlab\codetools  - Příkazy pro psaní a ladění kódu
```

`matlab\helptools` - Příkazy pro nápovědu
`matlab\winfun` - Soubory pro rozhraní s Windows (DDE/ActiveX)
`matlab\demos` - Příklady a dema.
`matlab\timeseries` - Vizualizace časových řad, analýza (není)
`matlab\hds` - (bez nápovědy)

Pozn.: goniometrické funkce pracují s hodnotami úhlů v radiánech, ale argumenty se mohou zadávat i ve stupních. Funkce potom mají příponu „-d“:

x v rad	**x** ve °	**x** v rad	**x** ve °
sin(x)	sind(x)	asin(x)	asind(x)
cos(x)	cosd(x)	acos(x)	acosd(x)
tan(x)	tand(x)	atan(x)	atand(x)
sec(x)	secd(x)	asec(x)	asecd(x)

3.5 Systémové informace o MATLABu

hostid – pouze výpis License Number MATLABu
license – ekvivalent příkazu **hostid**
ver – výpis údajů o MATLABu (verze, platformy, License Number MATLABu se všemi instalovanými moduly)

```
>>hostid
    '45597'
>>license
ans =
45597

>> ver
-------------------------------------------------------------------------------
MATLAB Version 7.0.4.365 (R14) Service Pack 2
MATLAB License Number: 45597
Operating System: Microsoft Windows 2000 Version 5.0 (Build 2195: Service Pack 3)
Java VM Version: Java 1.5.0 with Sun Microsystems Inc. Java HotSpot(TM) Client VM
-------------------------------------------------------------------------------
MATLAB                                                Version 7.0.4      (R14SP2)
Simulink                                              Version 6.2        (R14SP2)
Aerospace Blockset                                    Version 1.6.2      (R14SP2)
Bioinformatics Toolbox                                Version 2.0.1      (R14SP2)
CDMA Reference Blockset                               Version 1.1        (R14SP2)
Communications Blockset                               Version 3.1        (R14SP2)
Communications Toolbox                                Version 3.1        (R14SP2)
Control System Toolbox                                Version 6.2        (R14SP2)
...
```

3.6 Vektory a matice

Vektory i matice jsou při svém vytváření obecně ohraničeny hranatou závorkou [ ].
a) vektory
Zápis prvků do jednorozměrného pole **x**:

```
>> x = [1,23,3.5,10]
```

```
x =
1.0000   23.0000    3.5000   10.0000
```

nebo

```
>> x = [1 23 3.5 10]
x =
1.0000   23.0000    3.5000   10.0000
```

Jiný způsob vytvoření vektoru:

```
>> v=1:1.2:5
v =
1.0000    2.2000    3.4000    4.6000
```

nebo

```
>> v=-10:2:0
v =
-10    -8    -6    -4    -2     0
```

Lineárně rozložené prvky vektoru **v** (na intervalu 1 – 5 je 5 prvků)

```
>> v = linspace(1,5,5)
v =
1     2     3     4     5
```

počet bodů na intervalu 1 - 5

Logaritmicky rozložené prvky vektoru **v** (na intervalu 1 – 5 je 5 prvků)

```
>> v = logspace(1,5,5)
v =
    10         100        1000       10000      100000
```

Pozn.:

Zapisovat lze pole s různými prvky, např:

```
>>x = [10.5 6 sqrt(9.5) 2-3i]
```

b) *matice*

Zápis matice **A**:

$$A = \begin{pmatrix} 2 & 8 & 5 \\ 7 & 6 & 2 \\ 6 & 1 & 8 \end{pmatrix}$$

Její zápis v MATLABu je: `>>A = [2 8 5; 7 6 2; 6 1 8];`

Výpis obsahu matice **A** :

```
>> A
A =
     2     8     5
     7     6     2
     6     1     8
```

Ekvivalentní zápis matice **A**: `>>A = [2,8,5;7,6,2;6,1,8]`

nebo

```
>>A = [2 8 5
       7 6 2
       6 1 8]
```

Speciální typy matic – *ones, zeros, eye, diag, rand, randn*

```
>>a=ones(3)
a =
     1     1     1
     1     1     1
     1     1     1
>>zeros(2,5)
```

```
b =
     0     0     0     0     0
     0     0     0     0     0
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

Obsazení diagonály matice definovaným vektorem **v**:

```
>>v = 1:2:7
v =
     1     3     5     7
>>diag(v)-
```

nebo také **diag(**1:2:7**)**

```
ans =
     1     0     0     0
     0     3     0     0
     0     0     5     0
     0     0     0     7
```

Posunutí diagonály nahoru:

```
>> diag(v,1)
ans =
     0     1     0     0     0
     0     0     3     0     0
     0     0     0     5     0
     0     0     0     0     7
     0     0     0     0     0
```

Součet prvků na diagonále:

```
>> a=magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
>> h=trace(a)
h =
    15
```

Matice náhodných čísel v intervalu 0 až 1 (krajní meze se nikdy neobjeví), stejnoměrné rozdělění:

```
>>rand(4)
ans =
    0.9501    0.7621    0.6154    0.4057
    0.2311    0.4565    0.7919    0.9355
    0.6068    0.0185    0.9218    0.9169
    0.4860    0.8214    0.7382    0.4103
```

Normální rozdělení (nemá meze):

```
>>randn(2,3)
ans =
    0.8580   -1.5937    0.5711
    1.2540   -1.4410   -0.3999
```

3.6.1 Maticové operace v MATLABu

- Součet a rozdíl dvou matic: **A + B, A - B**
- Násobení dvou matic: **C = A*B**, odpovídá algebraickému násobení.

- Dělení matic: **D = A/B**, odpovídá algebraickému **A.** B^{-1}.
- Dělení je buď zprava: **A/B**, algebraicky **A.** B^{-1}
  nebo zleva: **A\B**, algebraicky A^{-1}**.B**

Při použití **.*** se násobí prvky matice první s prvním, druhý s druhým, atd. Stejně jako při násobení lze dělit jednotlivé **./** nebo **.** prvky matice první s prvním, druhý s druhým, atd.

- Umocnění matice **A^2** (jen pro čtvercové matice) nebo **.^** umocnění každého prvku matice
- Transpozice matice **A'**
- Inverze matice **inv(A)**
- Determinant matice **det(A)**
- Soustavu lineárních rovnic lze řešit zápisem

Ax = b (řešení je **x =** A^{-1} **b**)

Např.:

$$\begin{aligned} -x_1 + x_2 + 2x_3 &= 2 \\ 3x_1 - x_2 + x_3 &= 6 \\ -x_1 + 3x_2 + 4x_3 &= 4 \end{aligned}$$

```
>> A = [-1 1 2; 3 -1 1;-1 3 4];
>> b = [2;6;4];
```

Řešení může být zapsáno jako

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

nebo

```
>>x = A\b
```

a výsledek:

```
x =
 1.0000
-1.0000
 2.0000
```

Řešení systému lineárních rovnic lze provádět také pomocí příkazu LINSOLVE. Předchozí příklad:

```
>> x=linsolve(A,b)
x =
    1.0000
   -1.0000
    2.0000
```

- Přeurčenou soustavu rovnic (metoda nejmenších čtverců) a lze řešit pomocí \ (zpětné lomítko):

a = X\y kde **a** je vektor hledaných koeficientů
X je obdélníková matice s naměřenými prvky pro jednotlivé koeficienty v **a**
y je vektor závisle proměnných

Většina funkcí MATLABu není definovaná na maticích, ale na jejich prvcích. Pro celou matici je určena například funkce **expm(A)** pomocí Padého aproximace.

3.6.2 Operace mezi skaláry a polem

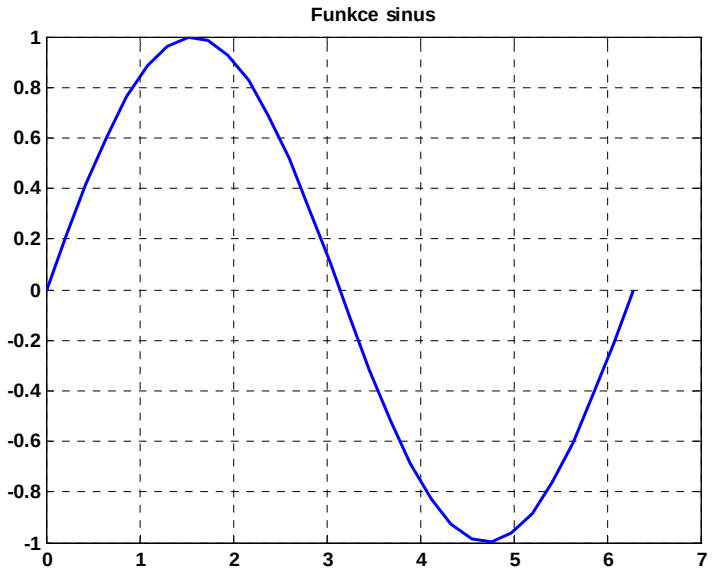
Všechny operace mezi skalárními hodnotami a maticemi se provádějí po jejich jednotlivýh prvcích. Např.

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

>> b=2*A +1
b =
    17     3    13
     7    11    15
     9    19     5

>> d=5*A/3
d =
   13.3333    1.6667   10.0000
    5.0000    8.3333   11.6667
    6.6667   15.0000    3.3333
```

9.0 Grafika v MATLABu ve 2D

Základní funkce pro vykreslování grafů nebo obrázků - `plot`. Vykreslení jednoduché funkce může být následující:

```
>>x=linspace(0,2*pi,30);    % vektor x v ose x s rozsahem 0 - 2*pi rozdělen na 30 bodů
>>y=sin(x);                 % vektor y v ose y obsahující hodnoty fce sinus
>>plot(x,y), title('Funkce sinus');  % vykreslení grafu y = sin(x) a nadpis grafu
>>grid;
```



Obecná syntaxe příkazu `plot` má tvar :

`plot``(x,y,`<*barva*><*značky*><*typ čáry*>`)`

nebo

`plot``(x1,y1,`<*barva*><*značky*><*typ čáry*>`,x2,y2,`<*barva*><*značky*><*typ čáry*>`,..),`

kde vektory $x_1, y_1, x_2, y_2, \ldots, x_i, y_i$ popisují různé grafy $y_i = f(x_i)$ s vlastní definicí čáry. Údaje ve špičatých závorkách popisují čáru příslušného grafu a při její definici jsou definice spojeny do jednoho řetězce. Chceme-li, aby vykreslovaná čára grafu `x,y` byla zelená, body zobrazeny jako hvězdičky a čára byla čárkovaná, napíšeme `plot(x,sin(x),'g*--')`

Pro zakreslení více grafů do jednoho obrázku můžeme použít příkaz `hold`.

```
>> hold on        % umožní  vykreslení nové funkce do stejného osového systému
>> hold off
```

Pokud jsou argumenty **x** a **y** vektor a matice, **`plot`** vykreslí postupně každý sloupec matice versus vektor. Např.

```
>> x=0:0.1:2*pi;
>> y=sin(x);
>> z=cos(x);
>> w=atan(x);


>> W=[y;z;w];
>> plot(x,W)
```

Záměnou proměnných x a W - `plot(W,x)` se graf otočí o 90°.

Pozn.: Pokud není uveden vektor **x**, bere *plot* hodnoty `1:m`, kde m je počet řádků v matici.

Přehled barev a typů čar, které lze v příkazu plot použít:

Barva čáry		Bod		Typ čáry	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

9.1 Popisy grafu z příkazové řádky, funkce `title, xlabel, ylabel`

```
title('Funkce sinus')      hlavička grafu
xlabel('pi')               popis osy x
ylabel('sin(x)')           popis osy y
grid                       rastr
```

Pozn.:
Popis grafu ve více řádcích je třeba zapsat do složených závorek, např. `title({'první řádek','druhý řádek'})`. Platí i pro funkce `xlabel, ylabel`.

9.2 Regulace os z příkazové řádky

```
>>axis ([xmin xmax ymin ymax])      % definice každé osy odděleně
>>axis auto              % nastavení os do automatického default módu
>>axis xy                %
>>axis manual            %
>>axis equal             %
>>axis square            % zajistí čtvercové zobrazení os
>>axis image             % ekvivalent equal
>>axis on                % vypnutí osového systému
>>axis off               % zapnutí osového systému
>>axis normal            % ruší nastavená měřítka a efekt square a equal
```

default nastavení os je: `axis auto on xy`

Mezní hodnoty os *x* a *y* můžeme zjistit funkcemi `xlim, ylim`.

9.3 Legenda grafu z příkazové řádky

Popis jednotlivých čar v grafu umožňuje funkce `legend`. Např.

```
>> legend('sinus','cosinus','atan')
```

Některé další možnosti pro práci s legendou:

```
>>legend off          odstranění legendy z grafu
>>legend hide         skrytí legendy
>>legend show         zviditelnění legendy
>>legend boxoff       odstranění rámečku legendy a zprůhlednění její plochy
>>legend boxon        zviditelnění plochy legendy
```

Polohu legendy lze jednoduše změnit levým tlačítkem myši. Umístění legendy při jejím zápisu umožňuje poslední celočíselný parametr v závorce. Např. `legend('sinus',-1)` umístí legendu mimo osy grafu. Dále viz. `help legend`.

9.4 Umístění textu v grafu

Funkce `text` a `gtext` umožňuje umístit text z příkazové řádky na plochu grafu. Syntaxe příkazu text je:

```
>>text(x,y,'řetězec')
```

Např.:

```
>>text(pi,sin(pi),'Sinusovka')
```

umístí řetězec 'Sinusovka' do souřadnic grafu `x = pi` a `y = sin(pi)`

funkce `gtext('řetězec')` umožní vložit text na zvolené místo grafu pomocí myši a záměrného kříže, který se při volání této funkce objeví.

Shora uvedené možnosti pro práci se souřadnými osami, texty, popisy nebo typy čar je možné využít z menu příslušného obrázku – figure. Jsou to ikony ve Figure Toolbar, volby v roletovém menu **Insert**, položky **X Label**, **Y Label**, **Title**, **Legend**, **Line**, **Text** nebo **Axes**. V roletovém menu **Edit** jsou položky **Figure Properties**, **Axes properties** a **Current Object Properties**, které umožňují pracovat s objekty ještě podrobnějším způsobem.

Pozn.:

Dosud uvedené možnosti při práci s grafem jsou určeny uživatelům, kteří nepotřebují ke své práci hlubší znalost grafiky MATLABu a jeho grafiku využívají k rychlému vykreslení výsledků svých výpočtů. MATLAB však interpretuje všechny tyto entity jako grafické objekty, které mají určité vlastnosti a které mají svůj specifický identifikátor – handle (viz. Handle Graphics). Mezi jednotlivými objekty je stanovena hierarchie ve smyslu rodiče a potomků. Pro orientaci v tomto hierarchickém stromu a pro práci s vlastnostmi vybraných objektů slouží potom příkazy `get` a `set`. Podrobnější seznámení s Handle Graphics je uvedeno v textech pro školení MATLAB II – tvorba vlastních aplikací.

9.5 Uzavření a mazání obsahu okna s grafem

Chceme-li okno s vykresleným grafem zavřít z příkazové řádky, můžeme použít příkaz `close` nebo `close(h)`, kde `h` je číslo obrázku, které je uvedeno jeho horní liště (např. Figure No.1) a identifikuje tak každé okno. Pokud máme otevřeno více oken a chceme zavřít všechna najednou, použijeme příkaz `close all`. K vymazání obsahu okna s grafem lze použít příkazy `clf` nebo `clf reset`. Příkaz `clf reset` resetuje všechny vlastnosti okna (figure) kromě zvolených jednotek a jeho polohy. Vysunutí aktuálního okna do popředí na obrazovce zajišťuje příkaz `shg`.

9.6 Rozdělění obrázku – figure pro více grafů

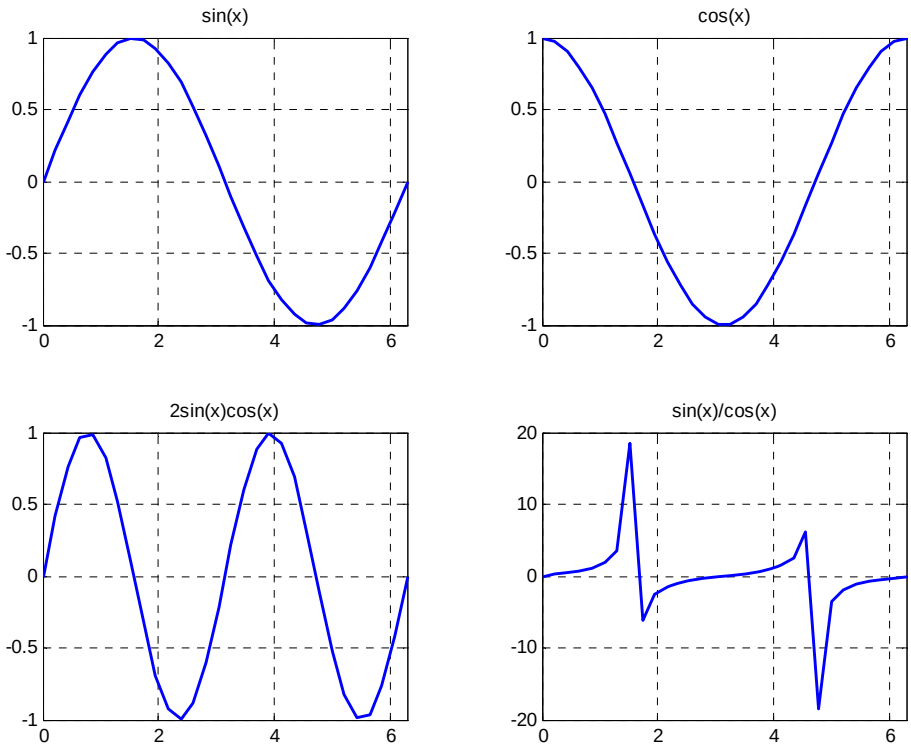
`subplot` `(m,n,p)` nebo **`subplot`** `(mnp)`

Kreslící plocha obsahuje ***m*** `x` ***n*** grafů a je rozdělena na ***m*** řádků a ***n*** sloupců. Číslo ***p*** určuje, do kterého grafu se právě kreslí. Grafu jsou kresleny po řádcích od shora dolů.

Postup při zobrazení 4 grafů do jednoho obrázku:

```
x=linspace(0,2*pi,30); y=sin(x); z=cos(x);
a=2*sin(x).*cos(x);
b=sin(x)./(cos(x)+eps);
subplot(2,2,1)
   plot(x,y), axis([0 2*pi -1 1]),title('sin(x)')
```

```
subplot(2,2,2)
   plot(x,z), axis([0 2*pi -1 1]),title('cos(x)')
subplot(2,2,3)
   plot(x,a), axis([0 2*pi -1 1]),title('2sin(x)cos(x)')
subplot(2,2,4)
   plot(x,b), axis([0 2*pi -20 20]),title('sin(x)/cos(x)')
```



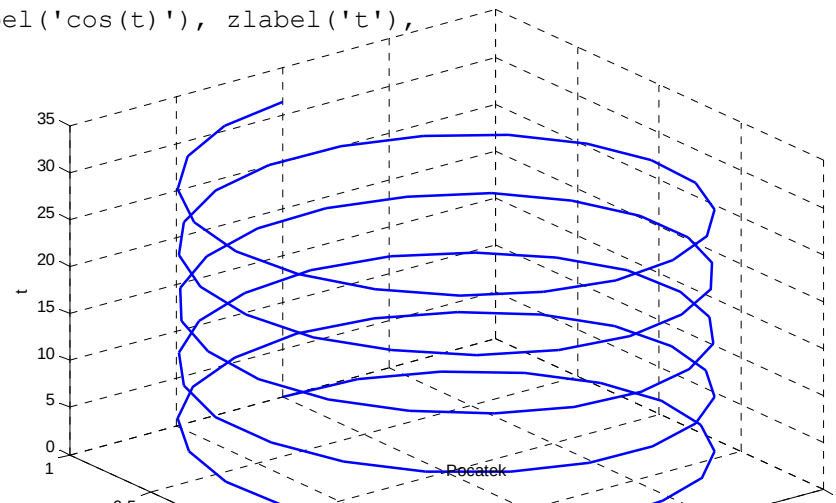
Kreslení dvou grafů s různými měřítky na dvou y-osách:

plotyy(x1,y1,x2,y2)

10.0 Grafy ve 3D

Kreslení grafů ve 3D je obdobné jako ve 2D. Ve funkci `plot3` přibyl pouze třetí rozměr – souřadnice **z**. Řetězec specifikující vykreslované body a čáry je stejný jako ve 2D.
Syntaxe **plot3**(x,y,z) nebo **plot3**(x,y,z,<*barva*><*značky*><*typ čáry*>)

```
>>t=linspace(0,10*pi);
>>plot3(sin(t),cos(t),t)
>>xlabel('sin(t)'), ylabel('cos(t)'), zlabel('t'),
>>text(0,0,0,'Pocatek')
>>grid on
```



Práce s grafy ve 3D je stejná jako ve 2D.

10.1 Speciální typy grafů ve 2D a 3D

Speciální typy grafů používají pro vykreslování běžná pravidla pro kreslení, v řadě případů lze jednotlivé grafy slučovat a kombinovat.

a) sloupcové a plošné grafy

```
A = [3 6 1
     9 8 6
     3 5 3]
```

`bar(X,Y)` – zobrazí sloupce m x n matice Y jako m skupin s n svislými sloupci.

`barh(X,Y)` - zobrazí sloupce m x n matice Y jako m skupin s n vodorovnými sloupci.

`bar3(Y,Z)` – zobrazí sloupce matice m x n matice Z jako m skupin s n svislými 3D sloupci.

`bar3h(Y,Z)` - zobrazí sloupce matice m x n matice Z jako m skupin s n vodorvnými sloupci.

`area(X,Y)` – zobrazí vektorová data po řádcích.

`pie(X)` - koláčový graf

`n=hist(Y,m)` – histogram v kartézských souřadnicích

`rose(THETA,N)` - histogram v kartézských souřadnicích

b) zobrazování diskrétních dat

`stem(X,Y)` – zobrazí sekvenci diskrétních dat (y-onové svislé čáry od osy x)

`stem3(X,Y,Z)` – zobrazí sekvenci diskrétních dat (y-onové svislé čáry od roviny xy)

`stairs(X,Y)` – zobrazí sekvenci diskrétních dat (y-onové svislé čáry jako "schody" v osy x)

Další specifické grafy ve 2D:

`errorbar(x,y,e), polar(t,r)`

c) grafy pro znázorňování směru a vektorů rychlosti

`compass(Z)` – zobrazuje vektory v polárních souřadnicích s počátkem ve středu kružnice, stupnice úhlů proti směru hod. ručiček (Z= (u,v))

`feather(Z)` - zobrazuje vektory v kartézských souřadnicích s počátkem na vodorovné čáře

`quiver(X,Y,U,V)` – zobrazuje 2D vektory rychlosti ve složkách u,v.

`quiver3(X,Y,Z,U,V,W)` – zobrazuje 3D vektory rychlosti ve složkách u,v,w.

d) vykreslování obrysů ve smyslu izočar

`[CS,H]=contour(X,Y,Z,N)` – zobrazuje 2D izočáry generované v matici Z
`clabel(CS,H)` – generuje popisy při vykreslování izočar
`contour3` – zobrazuje 3D izočáry generované v matici Z, jinak stejné jako `contour`
`contourf` – zobrazuje 2D izočáry a vyplňuje plochy mezi jednotlivými izočarami
`contourc` – funkce počítá matici kontur s využitím jiné matice kontur

10.2 Kreslení sítí a ploch ve 3D

MATLAB vykresluje síť jako drátově zobrazenou plochu, kdy barevná čára propojuje definované body. Při vykreslení povrchu jsou zobrazeny nejenom propojovací čáry, ale také plochy v příslušné barvě.
Pokud je argument příkazu `plot3` matice stejného rozměru, MATLAB vykreslí čáry získané ze sloupců x, y a z. Např.:

```
>> [x,y] = meshgrid(-2:0.1:2);
>> z = x.*exp(-x.^2-y.^2);
>> plot3(x,y,z);
>> grid on
```

Funkce pro vykreslování ploch:

`mesh, surf` – vykreslení sítě a plochy ve 3D
`meshc, surfc` – nakreslí plochu s vykreslením kontury pod touto plochou
`meshz` – nakreslí plochu s vykreslením svislých čar na obvodu k referenční rovině
`surf1` – nakreslí vystínovanou plochu v kombinaci ambientního, difúzního a zrcadlového světelného modelu
`surface` – funkce nízké úrovně pro vytváření grafických plošných objektů

Funkce, které vykreslují plochy mohou obsahovat přídavný vektor nebo matici.

```
>> mesh(X,Y,Z,C)
```
nebo
```
>> surf(X,Y,Z,C)
```

Obvykle bývá tímto parametrem údaj o barvě. Vertexu v z-ové souřadnici `Z(i,j)` pak odpovídá příslušná barva matice `C(i,j)`

Příklady

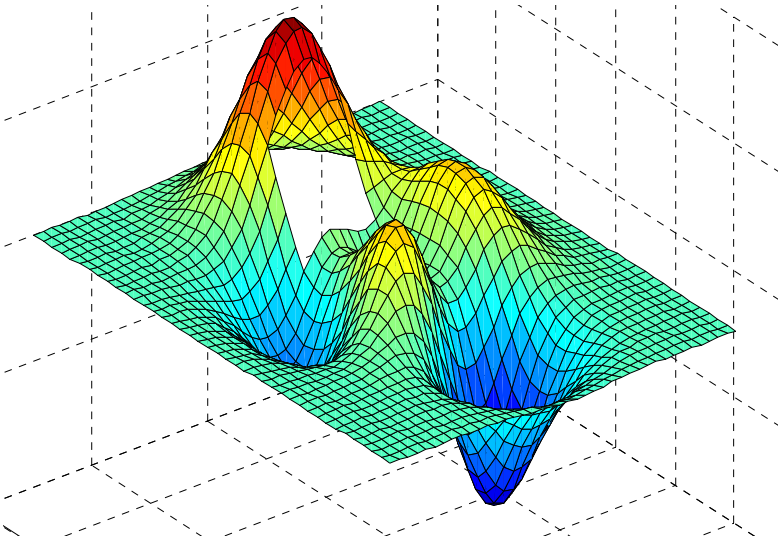
Příklad1:

```
[X,Y,Z]=peaks(40);
surf(X,Y,Z)
title('Vykresleni plochy')
xlabel('X-axis'), ylabel('Y-axis'), zlabel('Z-axis')
```

Otvory v povrchu lze vytvářet pomocí proměnné NaN. Protože NaN nemá hodnotu , funkce pro vykreslení ignoruje všechny datové body NaN a tím vytváří v těchto oblastech otvor:

Příklad2:

```
[X,Y,Z]=peaks(40);
x=X(1,:);
y=Y(:,1);
i=find(y>0.8 & y<1.2);
j=find(x>-0.6 & x<0.5);
Z(i,j)=nan;
surf(X,Y,Z)
title('Vykresleni plochy')
xlabel('X-axis')
ylabel('Y-axis')
zlabel('Z-axis')
```



Příklad3:

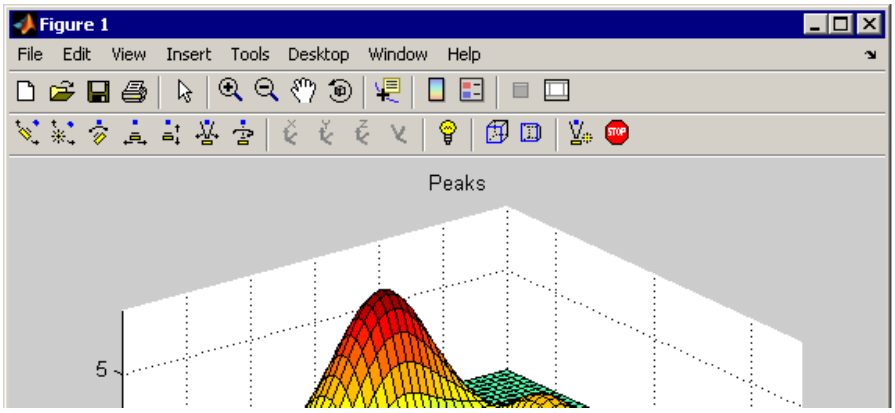
```
subplot(2,3,1)
       x = -3:0.3:3; y = x;
       [X,Y]=meshgrid(x,y);
       [theat,R] = cart2pol(X,Y);
       Z = sinc(R);
       contourf(peaks(30), 10)
       colorbar
       grid on

       title('Funkce peaks - (CONTOURF & COLORBAR)')
subplot(2,3,2)
       plot3(X,Y,Z)
       grid on
       axis([-3 3 -3 3 -1 1])
       title('Funkce sinc - (PLOT3)')
subplot(2,3,3)
       waterfall(membrane(1));
       title('Membrana ve tavru L - (WATERFALL)')
subplot(2,3,4)
       contour3(peaks(30), 25);
       title('Funkce peaks - (CONTOUR3)')
subplot(2,3,5)
       mesh(X,Y,Z)
       axis([-3 3 -3 3 -1 1])
       title('Funkce sinc - (MESH)')
subplot(2,3,6)
       surf(membrane(1))
       title('Membrana ve tavru L - (SURF)')

       echo off
       shg
```

Pozn.:

- Regulace průhlednosti 3D objektu – `hidden on, off`
- Regulace průhlednosti 3D objektu – `alpha(x)`, kde `x` je v rozsahu 0 až 1

10.3 Úhel pohledu na 3D objekt – azimut a elevace, kamera, světlo

Na 3D objekty lze pohlížet z různých stran a pod různými úhly. K definici pohledu slouží *azimut* a *elevace*. Azimut je úhel v rovině **x-y** měřený v od osy **x**. Elevace je úhel mezi rovinou x-y a spojnicí oka pozorovatel a počátku souřadného systému. Implicitně jsou hodnoty Az = -37.5°, El = 30°.
Z příkazové řádky `view(Az, El)`. Ve 2D jsou hodnoty `Az = 90°, El = 0°`.



Ovládání 3D scény je možné z dalšího řádku ikon v objektu figure

Nastavení scény v 3D prostoru

Ke stínování 3D objektů složí příkaz `shading`. Příkaz slouží k barevnému stínování objektů `surfce` a `patch`, které vytvářejí funkce `surf`, `mesh`, `pcolor`, `fill`, a `fill3`
Existují tři způsoby stínování:

`shading flat` - nastavuje shading na režim `flat`, stínování je po částech konstantní.
`shading interp` - nastavuje na režim `interpolated` (Gouraudovo), po částech lineární.
`shading faceted` - nastavuje na režim `faceted` (default), stínování po ploškách.

10.4 Barevná mapa – colormap

MATLAB může vykreslovaná data (nejčastěji ve 3D) mapovat podle specifikovaných barev.
Rozsah barev lze předdefinovat příkazem `colormap` nebo jsou využívány barvy (barevné mapy), které jsou nastaveny explicitně.

Mohou být vyžívány dvě techniky využití barev:

- indexováním, kdy každý bod vykreslených dat představuje index, který využívá příslušnou barvu v barevné škále. MATLAB tímto způsobem přiřazuje barvy v závislosti na použitém stínováním (*faceted*, *flat* nebo *interp*)
- truecolor (24-bit), kdy vykreslený povrch využívá explicitně definované barvy (v mapě RGB). MATLAB opět přiřazuje objektu barvy podle typu použitého stínování.

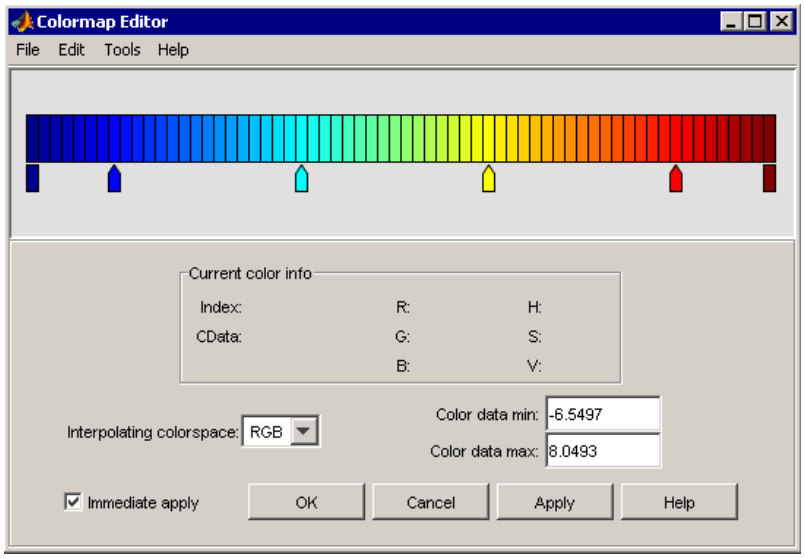
Barevná mapa

Každý figure v sobě obsahuje barevnou mapu v RGB. Tato mapa je matice o rozměru m x 3, kdy každý řádek této mapy definuje barvu třemi složkami - červená, zelená a modrá. Hodnoty každé barvy jsou v jednom sloupci v rozmezí 0 – 1. Defaultní barevná mapa v MATLABu obsahuje 64 barev.
Ke zobrazení barevné mapy slouží pruhu vedle osového systému (axis) - funkce `colorbar` - který aktuální barevnou mapu zobrazí. Implicitně MATLAB používá barevnou mapuu `jet`. Některé další barevné mapy používané v MATLABu: hot, cool, summer, autumn winter, ….

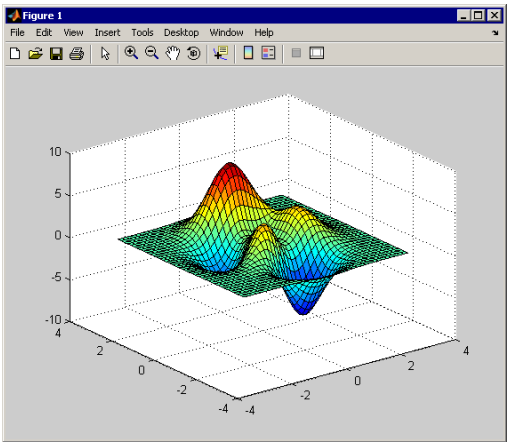
Barvy v podání RGB.

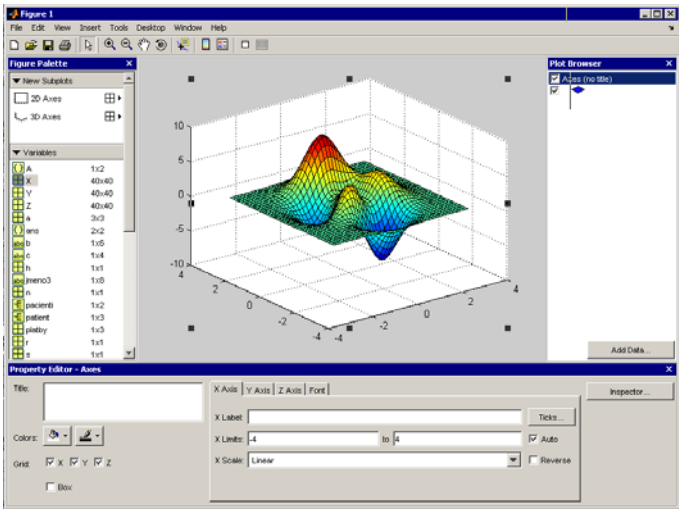
Červená Red	Zelená Green	Modrá Blue	Barva
1	0	0	Red
0	1	0	Green
0	0	1	Blue
1	1	0	Yellow
1	0	1	Magenta
0	1	1	Cyan
0	0	0	Black
1	1	1	White
0.5	0.5	0.5	Medium gray
0.67	0	1	Violet
1	0.4	0	Orange
0.5	0	0	Dark red
0	0.5	0	Dark green

Editor pro úpravu barevné mapy - Colormap editor:



Objekt figure





11.0 Rastrové obrazy

MATLAB zpracovává tři typy rastrových obrazů:

- *indexované* (vztah mezi údaji v matici obrazu – indexy a barevnou škálou)
  Pokud je matice obrazu třídy `double`, odpovídá první bod této matice prvnímu řádku barevné škály. Jsou-li data třídy `uint8`, nebo `uint16`, odpovídá první bod matice obrazu druhému řádku škály, druhý bod třetímu, atd. Je zde posunutí o 1.
- *ve škále šedi* (intensity images), každý prvek datové matice odpovídá jednomu pixelu v obraze. Data mohou být ve třídě `double`, `uint8` a `uint16`.

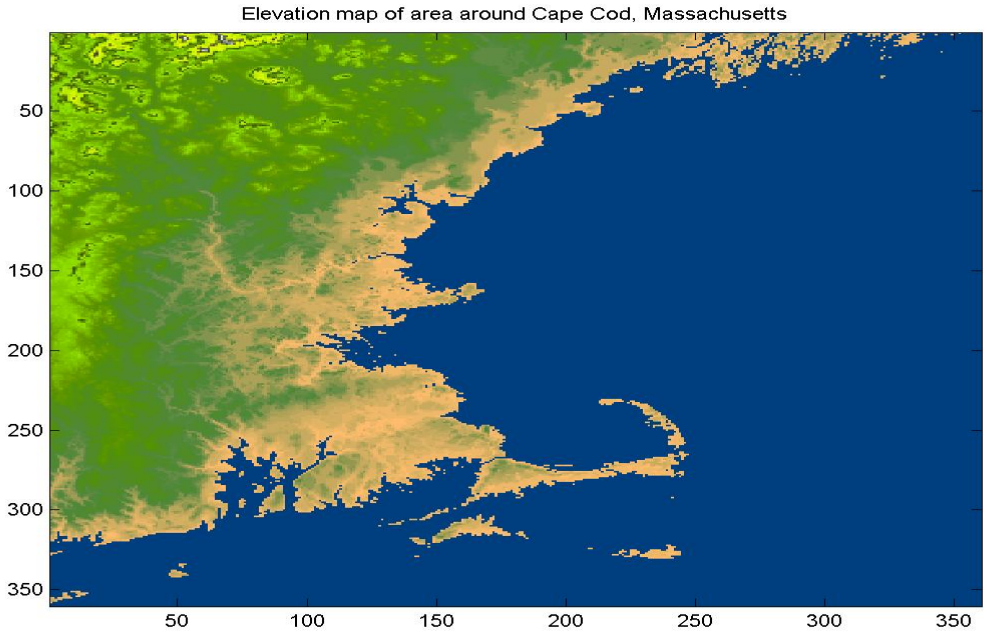
- *RGB* (truecolor, také 24-bit), barva 1 pixelu se skládá ze 3 barev červené, zelené a modré. Viz. tabulka s barvami v podání RGB.

Pozn.:
Vysvětlení datové třídy uvedeno v kapitole 6.4 až 6.8

11.1 Čtení a zápis rastrových souborů

Používané funkce `load, imread, image, imagesc`
Načtení předdefinovaného souboru příkazem `load` (MAT. soubor) vytvoří v pracovním prostoru datovou matici X a barevnou mapu `map`.

```
>>load cape
>>image(X)
>>colormap(map)
>>title ('Elevation map of area around Cape Cod, Massachusetts')
```




```
>>rgb=imread('ngc6543a.jpg');
>>image(rgb)
```

Funkce `imread` čte soubory v následujících formátech:

JPEG , ***TIFF*** (jakýkoliv soubor na bázi formátu TIFF, včetně 1-bit, 8-bit a 24-bit; 1-bit obrázky s CCITT kompresí; 16-bit černobílé, 16-bit indexované, a 48-bit RGB obrayky)
GIF libovolný GIF soubor od 1-bit do 8-bit
BMP 1-bit, 4-bit, 8-bit, 16-bit, 24-bit, a 32-bit nekomprimované; 4-bit and 8-bit RLE soubory
PNG libovolný PNG obrázek, včetně 1-bit, 2-bit, 4-bit, 8-bit, a 16-bit černobílých; 8-bit a

16-bit indexované; 24-bit a 48-bit RGB
HDF 8-bit se škálou i bez škály; 24-bit
PCX 1-bit, 8-bit a 24-bit obrázky
XWD 1-bit a 8-bit ZPixmaps; XYBitmaps; 1-bit XYPixmaps
ICO 1-bit, 4-bit a 8-bit nekomprimované obrázky
CUR 1-bit, 4-bit a 8-bit nekomprimované obrázky

Zápis rastrového souboru do souboru ve specifikovaném formátu BMP:

```
>>load clown
>>imwrite(X, map, 'clown.bmp')
```

nebo

```
>>imwrite(rgb,'kupa.png','BitDepth',16)
```

V příkladě jsme vytvořili soubor ve formátu ve formátu PNG o hloubce 16-bitů (datová třída uint16).
Pozn.:

- Pokud bychom zapsali uvedený soubor bez uvedení `BitDepth`, vytvořený soubor bude 8-bitový. MATLAB automaticky pracuje s hloubkou 8-bitů a všechny rastrové soubory do této třídy převádí.
- Převod indexovaného souboru do tvaru RGB zajišťuje funkce `ind2rgb`.

Možné výstupní formáty:

BMP 1-bit, 8-bit a 24-bit nekomprimované

TIFF základní řadu TIFF souborů, včetně 1-bit, 8-bit, 16-bit a 24-bit; 1-bit soubory CCITT 1D, Group 3 a Group 4 compression

JPEG základní řada

PNG 1-bit, 2-bit, 4-bit, 8-bit, a 16-bit černobílé; 8-bit a 16-bit černobílé obrázky s alfa kanály; 1-bit, 2-bit, 4-bit, a 8-bit indexované; 24-bit a 48-bit truecolorové soubory; 24-bit a 48-bit truecolorové soubory s alfa kanály

HDF 8-bit rastrová data, s nebo bez přiložené škály; 24-bit rastrová data; komprimovaná nebo s kompresí RLE nebo JPEG.

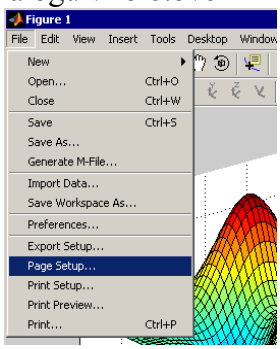
PCX 8-bit soubory

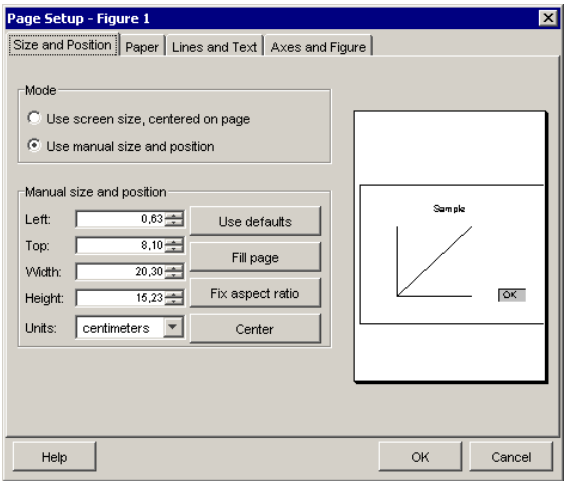
XWD 8-bit ZPixmap

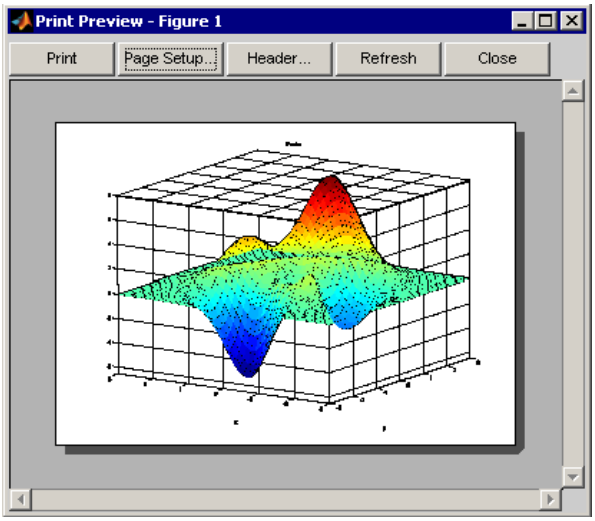
12.0 Tisk grafických objektů

Grafické objekty (grafy a modely) lze tisknout několika způsoby:

A) Z dialogu v roletovém menu **File** příslušného figure







B) Z příkazové řádky `>>print -device -options filename`
C) Tisk ve tavru funkce - `print('-device','-options','filename')`

Option `-device` představuje jméno ovladače, který chceme k tisku použít. Charakteristické je, že jeho jméno začíná vždy písmenem "d". Seznam ovladačů viz. `help print`.
Pokud použijeme option `-dsetup`, vyvolá se dialog pro nastavení tiskárny (`>>print -setup`)

Možnosti tisku z příkazového řádku:

a) Přímý tisk na tiskárnu: - místo `filename` se uvede *jméno tiskárny*. Samotný příkaz
```
>>print
```
vytiskne aktuální figure na implicitní tiskárnu.

Pozn.:
Tisk v síti Novell je nutné provádět podle pravidel print serveru sítě příkazem nprint. Např.:
```
>>nprint test1.ps /j=ps
```

b) Tisk obrázku do souboru pro pozdější tisk: - uvede se `filename` a *jméno tiskárny*
c) Export obrázku do souboru určitého formátu pro pozdější využití v nějaké aplikaci
d) Export obrázku do clipboardu

12.1 Řízení rozlišení tisku – defaultní hodnoty pro různé typy grafických formátů

-Vestavěné exportní formáty v MATLABu - rozlišení 150dpi (kromě EMF a ILL)
-Vestavěné drivery MATLAB – Windows a PostScript na tiskárny 150dpi – (pro `painters` 864 dpi), GUI nebo příkaz `print`
-**EMF** export format – 150dpi, příkaz `print`
-**EPS** – 150dpi, příkaz `print`
-**Ghostscript** formát– 72dpi, *nelze měnit*
-**Ghostscript** tiskárna – 150dpi, příkaz `print`
-tiskárny s **HPGL** driverem - 1116 dpi

-**ILL** export format (AdobeIllustrator) – 72dpi, *nelze měnit*

Rozlišení je libovolné a určuje je pouze výstupní zřízení.
Nastavení `print -r`*number* , `r`*0* je rozlišení obrazovky

1) tisk do *PostScriptoveho* souboru (append - přidává do existujiciho souboru)
```
>>print -dps -append myfile
```

2) tisk do souboru specifikovaného formátu – `figure(2)` s rozlišením 300dpi do 24-bit **JPEG**:

```
>>print -djpeg -f2 -r300 myfigure
```
do 24-bit **TIFF**:
```
>>print -dtiff myfigure
```

3) tisk do barevného **EPS** s preview v TIFF
```
>>print -depsc -tiff myfigure
```

MATLAB při tisku podporuje následující grafické formáty:

Figure je uložen do souboru určitého grafického formátu. Jsou podporovány formáty rastrové i vektorové:
AI, BMP*, EMF*, EPS, HDF, HPGL, JPEG, PBM, PCX, PGM, PNG, PPM, TIFF.
*Formáty označené hvězdičkou mohou být exportovány do clipboardu.
Specifikace zařízení -device určuje výstup na tiskárnu nebo do souboru.
Ovladače mohou být např.:

-dwin pošle obrázek na aktuální tiskárnu černobíle
-dwinc pošle obrázek na aktuální tiskárnu barevně
-dsetup otevře dialog pro nastavení tiskárny

Další ovladače v MATLABu např:

-dps PostScript pro černobílé tiskárny
-dpsc PostScript pro barevné tiskárny
-depsc2 Encapsulated Level 2 Color PostScript
-dtiff TIFF s packbits kompresí
-dpng obrázky ve formátu Portable Network Graphic 24-bit truecolor (pouze figures)

Pozn.:
Nastavení barev:

```
set(gcf, 'color', 'blue');
set(gca, 'color','yellow');
set(gcf, 'InvertHardCopy', 'off');
```

Vyjmutí uicontrols z obrázku (-noui):

```
>>print -depsc -noui myfile.eps
```

Seznam fontů, které podporují drivery MATLABu:

```
AvantGarde, Helvetica-Narrow, Times-Roman, Bookman, NewCenturySchlbk,
ZapfChancery, Courier, Palatino, ZapfDingbats, Helvetica, Symbol
```

14.0 Čtení a zápis dat do souboru, základy programování v M-souborech

MATLAB může pracovat s různými typy souborů. Při práci si uživatel může uložit rozdělanou práci se všemi proměnnými do souboru s příponou .MAT (binární soubor) nebo do ASCII (textového souboru). K tomu slouží příkaz `save`, který ukládá proměnné z pracovního prostoru MATLABu do definovaného souboru:

>>**`save matice A B c`** - ukládá proměnné A,B a c do souboru 'matice.mat'
>>**`save matice.txt A B c –ascii`** - ukládá stejné proměnné do textového souboru
nebo
>>**`save('matice.txt','A','B','c')`** - ve tvaru funkce

K načtení zapsaných souborů zpět do pracovního prostoru slouží funkce `load`:
>> **`load matice`** - načte .MAT soubor do pracovního prostoru MATLABu

`>> load matice.txt` nebo `>>load('matice.txt')` – načte textový soubor `matice.txt`

Pozn.:

- Samostatné použití příkazů `save` a `load` uloží a načte soubor `'matlab.mat'` do aktuálního adresáře
- Při čtení ASCII souborů s více proměnnými (matice) je MATLAB načte, pokud mají shodný rozměr. Při čtení MAT souborů struktura proměnných nehraje roli.
- Při ukládání MAT souborů ve verzi MATLAB 7 a jejich následnému čtení v nižší verzi (6.5 a nižší) je třeba uložit s přepínačem -v6, protože zde má MATLAB jinou strukturu.
- Při ukládání dat příkazem save lze použít přepínač –append pro připojení dalších proměnných do existujícího MAT souboru.

Další příkazy pro čtení textových souborů:

`dlmwrite` – zapisuje textový soubor s definovaným oddělovačem.

Např.:

```
>> dlmwrite('dlmwrite_file.txt',matice,',')
```

Zapíše proměnnou '`matice`' do souboru 'dlmwrite_file', ve kterém jsou jednotlivé prvky odděleny čárkou.

```
>> dlmwrite('dlmwrite_file.txt',matice,'\t')
```

Zapíše proměnnou '`matice`' do souboru 'dlmwrite_file', ve kterém jsou jednotlivé prvky odděleny tabelátorem.

```
>> dlmwrite('dlmwrite_file.txt',matice,'\n')
```

Zapíše proměnnou '`matice`' do souboru 'dlmwrite_file', ve kterém jsou jednotlivé prvky seřazeny pod sebe.

```
>> dlmwrite('dlmwrite_file.txt',matice,'\t',12,10)
```

Zapíše proměnnou '`matice`' do souboru 'dlmwrite_file', ve kterém je proměnná matice posunutá v ofsetu 12 řádků, 10 sloupců. Základna pro počet řádků a sloupců je 0,0

Čtení a zápis ASCII dat oddělených čárkou funkcemi: `csvread`, `csvwrite`

Syntaxe pro čtení:

```
m = csvread('filename')
m = csvread('filename',r,c,rng)
```

Syntaxe pro zápis:

```
csvwrite(filename,m)
csvwrite(filename,m,r,c)
```

Čtení souboru s oddělovačem:

Např.:

```
>> A=dlmread('dlmwrite_file.txt','\t',range)
```

Načte obsah souboru `'dlmwrite_file.txt'` s oddělovačem \t (tabelátor) do matice A v zadaném rozsahu, kde v `range` zadáváme levý horní a pravý dolní roh v datech, např. – `range=[1 0 12 10]`

`xlsread` – čte XLS soubor z MS Excel.

Obecná syntaxe:

```
 NUMERIC = xlsread(SOUBOR,LIST,ROZSAH);
[NUMERIC,TXT]=xlsread(SOUBOR,LIST,ROZSAH);
[NUMERIC,TXT,RAW]=xlsread(SOUBOR,LIST,ROZSAH);
```

Např.

```
>>A = xlsread('pokusy1.xls')  - načte první list souboru ve formátu Excel
>>A = xlsread('pokusy1.xls','list2', 'a2:e5')
```

Načte data v rozsahu buněk **`a2:e5`** z listu **`listu2`** a ze souboru **`pokusy1.xls`**

`xlswrite` – zapisuje data do XLS souboru MS Excel.

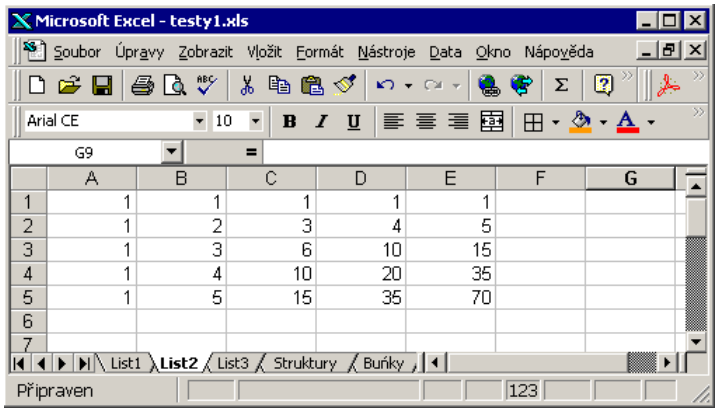
Obecná syntaxe: `[SUCCESS,MESSAGE]=XLSWRITE(SOUBOR,ARRAY,SHEET,RANGE)`

kde `ARRAY` je číselné pole nebo buňka
`SHEET` je název listu v XLS souboru
`RANGE` je rozsah buněk pro zápis

Např.:

```
>> B=pascal(5)
B =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70

>> [aa,bb]=xlswrite('testy1.xls',B,'List2')
aa =
     1
bb =
       message: ''
    identifier: ''
```

bb je struktura, obsah polí `message` a `identifier` zjistíme zápisem `bb.message` a `bb.identifier`



Čtení souboru s oddělovačem:

Např.:

```
>> A=dlmread('dlmwrite_file.txt','\t',range)
```

Načte obsah souboru `'dlmwrite_file.txt'` s oddělovačem \t (tabelátor) do matice A v zadaném rozsahu, kde v `range` zadáváme levý horní a pravý dolní roh v datech, např. – `range=[1 1 12 10]`
Pozn.:

- Použití češtiny v textech nebo popisech listů může způsobit nenačtení souboru a ohlášení chyby.
- čtení grafických objektů a zápis do formátu vrml 2.0 , `vrml(h,jm_souboru)` (viz. `help vrml`)

Čtení dat z textového souboru lze provádět také příkazem `textread`. (viz. `help textread`)
Např. soubor `seznam.txt` obsahuje matici:

```
12  36  45  2
3   13  63  3
```

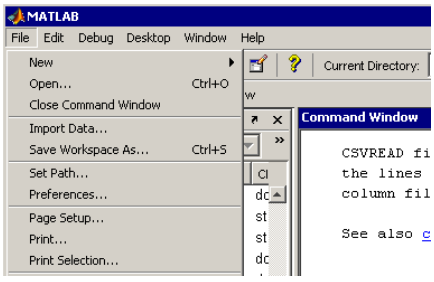
Její čtení potom:

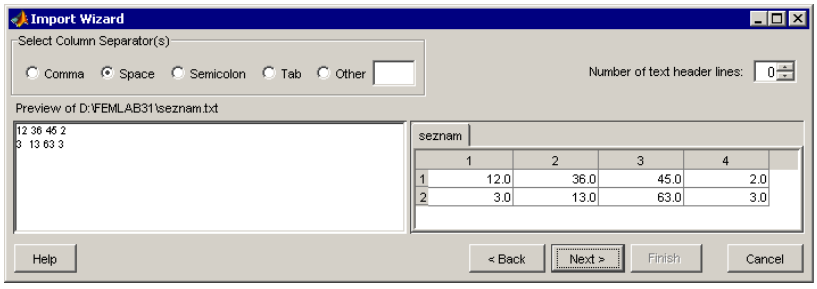
```
>>[a,b,c,d]=textread('seznam.txt','%f%f%f%f')
a =
    12
     3
b =
    36
    13
c =
    45
    63
d =
     2
     3
```

do proměnných `a,b,c,d` jsou načteny sloupce matice uvedené v souboru *seznam.txt*. Zápis do jediné proměnné: `>> A=textread('seznam.txt')`

```
A =
   12    36    45     2
    3    13    63     3
```

Interaktivní načítání dat:

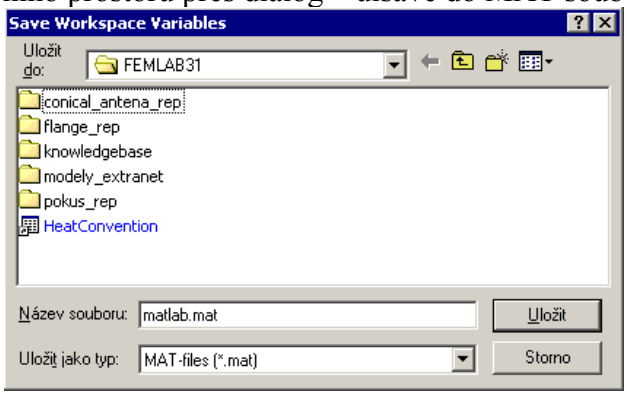




nebo také z příkazového řádku funkce `uiimport`.
Lze načíta data textová i obrazová ve formátech:
Grafické soubory: *.GIF, *.CUR, *.HDF(HDF-EOS), *.ICO, *.JPG, *.PNG, *.JPEG, *.TIF, *.BMP, *.PCX
Textová data: *.TXT, *.DAT,*.DLM,*.TAB
Spreadsheety: *.CSV, *.XLS, WK1

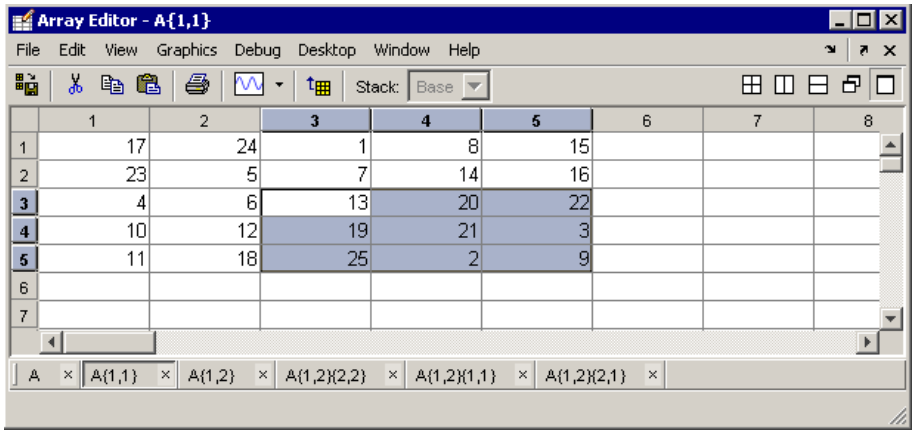
Zvuk: *.AU, *.SND, *.WAV
Movie: *.AVI
Binární MATLAB: *.MAT

Ukládání dat z pracovního prostoru přes dialog – uisave do MAT souboru:

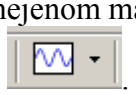


Array Editor:

```
>> openvar('A')
```



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

Array Editor slouží k interaktivní manipulaci s proměnnými uloženými v pracovním prostoru MATLABu. Argument funkce `openvar` je vždy řetězec.
Lze načítat a editovat nejenom matice, ale také strruktury a buňky. Označená data lze jednoduše vykreslovat přes ikonu  .

Vytvoření textového souboru se jménem `matice.txt` pomocí funkcí `fopen, fprintf` a `fclose`:

```
>>fid = fopen('matice.txt', 'wt');
>>fprintf(fid, '%s\n', 'Toto je matice 3 x 3');
>>fprintf(fid, '%i\t%i\t%i\n', [1 2 3; 4 5 6; 7 8 9]');
>>fclose(fid);
```

Přečtení vytvořeného souboru:

```
>>fid=fopen('matice.txt', 'rt');
>>title = fgetl(fid);
>>[data,count]=fscanf(fid, '%i');
```

```
>>data = reshape(data, 3, 3);
>>data_transpose = data';
>>fclose(fid);
```

kde `%i` značí integer, `\t` tabelátor a `\n`, čtení celého řádku `fgetl`

Zápis *binárního* souboru:

```
>> fid = fopen('matice.bin','wb')
>> fwrite(fid,'Toto je čtvercová matice', 'char');
>> fwrite(fid, [1 2 3; 4 5 6; 7 8 9]',int32');
>> fclose(fid);
```

Popis:

1. Otevření souboru (`fopen`) pro zápis v binárním módu `('wb')` a vytvoření identifikátoru `fid`
2. Zápis hlavičky do souboru `(fwrite)`
3. Zápis numerických dat do souboru `(fwrite)`, kde `'char'` – značí písmena, `'int32'` - značí 32 bitová čísla integer
4. Uzavření souboru

Shrnutí:

`fread` - čte binární data ze souboru - `[a, countT] = fread(fid,size,precision)`
`fwrite` – zapisuje binární data do souboru - `count = fwrite(fid,A,precision)`
`fprintf` – zapisuje formátovaná data do souboru - `count = fprintf(fid,format,A,...)`

čtení dat po řádcích:
`fgetl, fgets.`

15.0 Relační operátory

MATLAB pracuje s následujícími relačními operátory:

<	je menší než
>	je větší než
>=	je větší nebo rovno
<=	je menší než
==	je rovno
~=	není rovno

Když podmínka platí, je výsledek TRUE (1), když neplatí je FALSE (0)

Např. když `a = 15/8` je

```
>> a > 1;
ans =
```

```
     1
```

Relační operátory pracují také s maticemi a jejich jednotlivými prvky. Výsledkem testu jsou pak u jednotlivých prvků 0 a 1.
Chceme například všechny prvky matice A, které jsou <= 3, položit nule a ostatní zachovat. Výsledek zapíšeme do matice C:

Matice A:

```
>>A=magic(3)
   A =
      8     1     6
      3     5     7
      4     9     2

C=(A>3).*A                  nebo pro lineární indexy
   C =                            >>i = A>3
      8     0     6               >>C = A(i)' ->
      0     5     7               C =
      4     9     0                     8    4    5    9    6    7
```

Porovnání prvků dvou matic A, B a vytvoření matice kde D, která obsahuje shodné prvky co do velikosti a pozice:

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

D=A.*(A==B)
D =
   0     0     0
   0     5     0
   0     0     0
```

13.0 Logické operátory

& - logické AND (když jsou splněny všechny podmínky)
| - logické OR (když je splněna jedna z podmínek)
~ - NOT

V MATLABU mohou Booleovské operátory pracovat opět s celými maticemi najednou a testovat každý jejich prvek nezávisle. Výsledek Booleovské operace je uložen jako logický vektor (nuly a jedničky)
Např.: Mějme dva vektory **u** a **v**:

```
>>u = [1 2 0 12 31 5];
>>v = [1 5 0 11 1 0];
```

Operátorem AND (&) určíme, které z prvků obou vektorů jsou nenulové (musí být oba dva ve stejné pozici):

```
>>u&v
ans =
     1     1     0     1     1     0
```

Naopak určení nulových prvků vektoru **u**

```
>>~u
ans =
     0     0     1     0     0     0
```

17.0 Funkce pro testování prvků matice

Pro vektory vrací 1, pokud je alespoň jeden prvek nenulový. V opačném případě vrací 0. V maticích funkce pracují implicitně po řádcích s vyhodnocením každého sloupce. Jinak můžeme testovanou dimenzi zvolit jako druhý parametr. 1 značí řádky (také funkce bez parametru) , 2 jsou sloupce.

any(x), any(A,dim)– je pravda (1), jestliže je alespoň jeden prvek pole **x** (matice **A**) nenulový.
all(x), all(A,dim)– je pravda (1), jestliže jsou všechny prvky pole **x** (matice **A**) nenulové.
i = find(x) – vrací indexy nenulových prvků pole **x**
[i,j] = find(A) – vrací indexy (i -řádky, j -sloupce) nenulových prvků matice **A**
[i,j,v] = find(A) – parametr **v** obsahuje nenulové hodnoty odpovídající **i, j**
min(x) – hledá minimální prvky pole (matice)
max(x) – pro vektor vybere minimální, maximální prvek
sum(x), – součet prvků matice

Zapišme matici:

```
>>b = [1 -5 0;13 9 -1;0 33 0]
```

Je pravda, když alespoň jeden prvek je nenulový – **any**

```
>> any(b)
ans =
      1     1     1
>> any(b>10,1)
ans =
      1     1     0
>> any(b>10,2)
ans =
       0
       1
       1
```

Je pravda, když všechny prvky jsou nenulové – **all**

```
>> all(b)
ans =
       0     1     0
```

```
>> a=b
 a =
      1    10     0
     13     9    10
      0    33     0
 >> a(:,end)=10
 a =
      1    10    10
     13     9    10
      0    33    10
 >> all(a==10,1)
 ans =
      0     0     1
 >> all(a==10,2)
 ans =
      0
      0
      0
```

Hledání indexů prvků a hodnot samotných prvků s nenulovou hodnotou:

```
>> [i,j,v]=find(b);
>> i'
ans =
     1     2     1     2     3     2
>> j'
ans =
     1     1     2     2     2     3
>> v'
ans =
     1    13    -5     9    33    -1
```

Funkce `find` vrací lineární idexy nenulových prvků.

Zobrazení indexů prvků < 0

```
>>find(b<0) % zobrazí se formou lineárních indexů
ans =
     4
     8

>>[i,j]=find(b<0) % zobrazí se formou indexů řádků (i) a sloupců (j)
>> i'
ans =
     0     2
>> j'
ans =
     2     3

>> b(find(b<0))=10
b =
     1    10     0
    13     9    10
     0    33     0
```

Pozn.:

Mějme na paměti, že náhrada vyhledaných prvků je možná přes lineární indexy – funkce `find` bez výstupních parametrů.

17.1 Hledání maximálního minimálního prvku

```
>>A=magic(4)
   A =
       16     2     3    13
        5    11    10     8
        9     7     6    12
        4    14    15     1

>> max(A)        % maximální hodnota v každém sloupci
ans =
    16    14    15    13
>> max(A,[],1)   % max v 1. dimenzi je ekvivalentní příkazu max(A)
ans =
    16    14    15    13

>> max(A,[],2)   % max. hodnoty v jednotlivých řádcích
ans =
    16
    11
    12
    15
>> [hodnota,index]=max(A,[],2) % max. hodnoty v jednotlivých
hodnota =
    16
    11
    12
    15
index =
     1    %1. prvek v řádku
     2    %2. prvek v řádku
     4    %4. prvek v řádku
     3    %3. prvek v řádku

Maximální hodnota prvku v celé matici.

>> max(A(:))
ans =
    16
```

(stejné pro funkci `min`)

17.2 Součty prvků matice, práce s relačními operátory

```
   >> b
   b =
        1    -5     0
       13     9    -1
        0    33     0
   >> sum(b)      % součet prvků v jednotlivých sloupcích
   ans =
       14    37    -1
   >> sum(b,1)    % součet prvků v jednotlivých sloupcích. Ekvivalent sum(b)
   ans =
       14    37    -1
```

```
>> sum(b,2)    % součet prvků ve jednotlivých řádcích.
ans =
    -4
    21
    33
```

V následujícím příkladě je několik testů platnosti dat obsažených v matici **M**. Každý prvek se testuje, jestli:

- je kladný (větší než a nebo roven nule) – vrací pole stejného rozměru jako je **M**
- všechny kladné (všechny větší a nebo rovny nule) – vrací skalár
- každý je kladný a konečný – vrací pole stejného rozměru jako M

```
>> M = [-2 10 NaN 30 -11 Inf 31];
>> pozice = M >= 0      % výběr platných pozic (1) pro prvky >= 0
pozice =
    0     1     0     1     0     1     1

>> vse = all(M >= 0)
vse =
    0                   % vrací 0, protože všechny prvky pole nejsou >= 0

>> vsechny_pozice = (M >=0)&(isfinite(M))
vsechny_pozice =
    0     1     0     1     0     0     1
% výběr platných pozic (1) pro prvky >= 0 a současně těch, které jsou konečné.


>> jen_dobre = M(vsechny_pozice)    %Zápis hodnot prvků  (M >=0)&(isfinite(M))
jen_dobre =
   10    30    31
```

18.0 Predikáty – testování existence specifických hodnot nebo podmínek

V MATLABu lze testovat nejenom konkrétní hodnoty, ale také různé typy proměnných nebo stavy. Volané funkce nazýváme predikáty, které vrací logickou hodnotu 0 nebo 1. Některé z nich jsou uvedeny v následující tabulce:

`ispc`	– vrací 1 u MATLABu ve verzi pro Windows
`isunix`	-vrací 1 u MATLABu pro UNIX
`isreal(X)`	– vrací 1 pokud je hodnota X nemá imaginární část
`isnan(X)`	– vrací 1 pokud hodnota proměnné X není definovaná
`isinf(X)`	– vrací 1 pokud je hodnota proměnné X nekonečno
`isfinite(X)`	– vrací 1 pokud je hodnota proměnné X konečný prvek (není Inf nebo Nan)
`ischar(S)`	– vrací 1 pokud je hodnota proměnné S písmeno

`isempty(A)` – vrací 1 pokud je matice A prázdná
`isnumeric(A)` – vrací 1 pokud je A pole čísel (buňky a struktury nejsou číselná pole)

`islogical(X)` – vrací 1 pokud je X logické pole
`isobject(a)` – vrací 1 pokud je a objekt MATLABu
`iscell(C)` – vrací 1 pokud je proměnná C buňka
`isstruct(S)` – vrací 1 pokud je proměnná S struktura
`isfield(S,'field')` – vrací 1 pokud je řetězec `'field'` jméno pole ve struktuře S
`[FLAG,FIG] = figflag(STR,SILENT)` – vrací 1 jestliže je jakýkoliv figure označený STR aktuálně zobrazen na obrazovce. Pokud je `SILENT=0`, jsou okna přenesena do popředí.

19.0 Příkazy pro běh programu

Při chodu programu je třeba usměrňovat jeho běh, využívat za specifikovaných podmínek jenom jeho části nebo větve. K tomu slouží funkce *if, switch, try*.
Testování prvků polí nebo naplňování proměnných v cyklech zajišťují funkce *for, while.*

19.1 Rozhodování v blocích

Příkaz `if-elseif-else-end`:

```
if výraz
            příkazy
      elseif výraz  ( elseif může být libovolné množství)
            příkazy
        ...
      elseif výraz
            příkazy
       else
            příkazy
end
```

Příklad:

```
I=1;
J=2;
if I == J
      A(I,J) = 2;
elseif abs(I-J) == 1
      A(I,J) = -1;
else

      A(I,J) = 0;
end
```

Příkaz `switch-case-otherwise-end`:

```
switch výraz
          case výraz,
            příkaz, ..., příkaz
```

```
        case {case_výraz1, case_výraz2, case_výraz3,...}
          příkaz, ..., příkaz
         ...
        otherwise,
          příkaz, ..., příkaz
    end
```

19.2 Příkazy cyklu

Příkaz `while-end`:

```
while výraz
    Sled příkazů
    end
```

Příklad:

```
I=1; N=10;
    while I<=N

    J=1;
      while J<=N
         A(I,J)=1/(I+J-1);
         J=J+1;
      end
      I=I+1;
    end
```

Příkaz `for-end`:

```
for proměnná = výraz, příkaz, ..., příkaz end
```

Příklad:

```
N=10;
for I = 1:N,
   for J = 1:N,
      A(I,J) = 1/(I+J-1);
   end
end
```

19.3 Vektorizace

Je známo, že používání smyček `for - end` ve zdrojovém kódu v řadě případech program zpomaluje. Východiskem může být tzv. vektorizace, kdy uživatel využívá vlastností MATLABu. Kromě urychlení programu se struktura kódu zjednodušuje. Proveďme výpočet vztahu $y = x^2$ pro rozsah `x` od 1 do 5:

```
x = [1 2 3 4 5];
```

Vektor funkčních hodnot vypočteme v cyklu:

```
y = zeros(size(x));
for i = 1:length(x)
      y(i) = x(i)^2;
end
```

Stejný výpočet s využitím vektorizace:

```
x = [1 2 3 4 5];
y = x.^2;
```

Příklad výpočtu a vykreslení funkce **y = sin(2x)+cos(x)**:

```
x = -500:.1:500;
y = zeros(size(x));

for i = 1:length(x)
      y(i) = sin(2*x(i))+cos(x(i));
end
plot(x,y)
```

Použitím vektorizace:

```
x = -500:.1:500;
y = sin(2.*x)+cos(x);
plot(x,y)
```

Příklad na výpočet hustoty:

```
M = rand(5,1000); L = rand(5,1000);
W = rand(5,1000); H = rand(5,1000);
[rows, cols] = size(M);
for I = 1:rows
      for J = 1:cols
   Density(I,J) = M(I,J)/(L(I,J)*W(I,J)*H(I,J));
      end
end
```

Použití vektorizace:

```
Density = M./(L.*W.*H);
```

Cvičení: Výpočet teplotního gradientu.

Teplotní gradient při prostupu tepla zdí se stanoví podle vztahu:

$T_x = (T_{in} - T_{out}) * x/L + T_{out}$

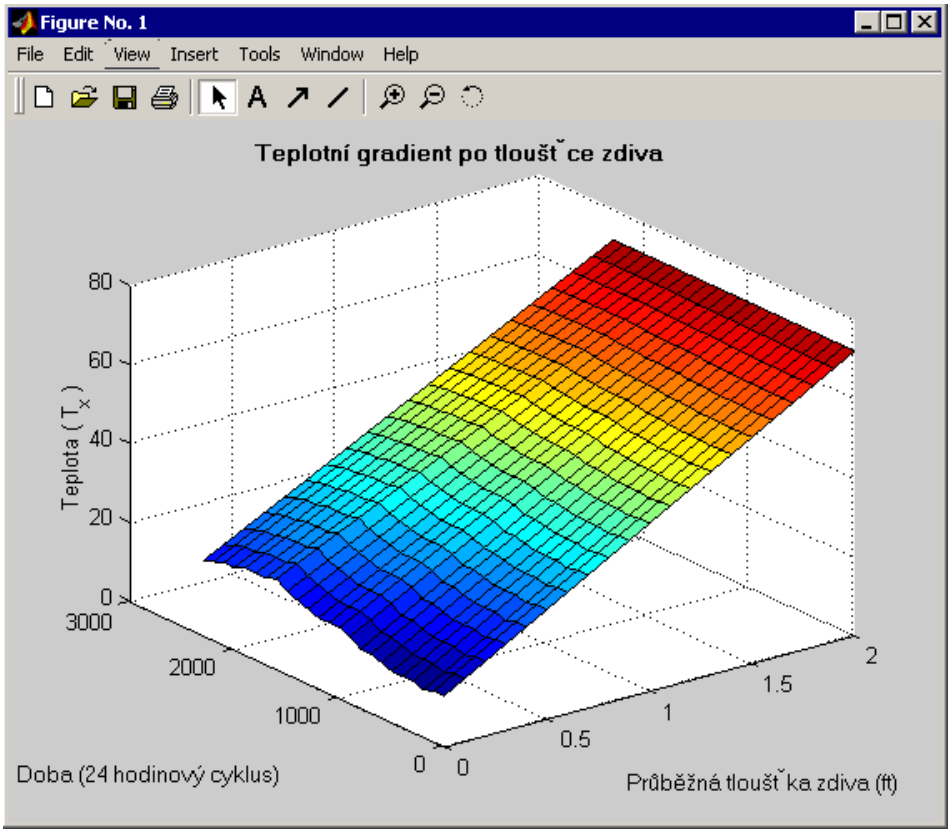
kde T_x je teplota v příslušné tloušťce zdi
T_{in} je vnitřní teplota
L je tloušťka zdiva
x je okamžitá tloušťka v rozmezí 0 - L

Úkolem je vykreslit plochu s teplotním gradientem, který znázorňuje teplotu T_x v příslušné tloušťce x.
Tabulka měřených teplot ve 24 hodinovém cyklu:

```
Teplota:
temp =[3  13  12  13  13  13  14  15  15  17  18  18  18 19  20  21  23  22  22  22  21  21  20 19]

Čas měření:
```

`time` = [0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000 2100 2200 2300]

Vnitřní teplota: `Tin = 72 F`
Tloušťka zdiva: `L = 2 ft`



Řešení:

```
Tin = 72;
L = 2;
x = 0:.1:L; % vytvoření x nebo x = linspace(0,L,21);

% příprava sítě kdy každá teplota 'temp' odpovídá tloušťce 'x'
ttemp = repmat(temp,1,length(x));
xx = repmat(x,24,1);

% výpočet teploty Tx (gradientu)
Tx = (Tin - ttemp) .* xx./L + ttemp;
surf(xx,time,Tx) % vykreslení plochy
xlabel('Průběžná tloušťka zdiva(Ft)')
ylabel('Doba (24 hodinový cyklus)')
zlabel('Teplota ( T_x )')
title('\bf{Teplotní gradient po tloušťce zdiva}')
```

22.0 Regrese a prokládání křivek

přeurčená soustava lineárních rovnic a hledání koeficientů, použití \ (zpětného lomítka)

```
t = [0 .3 .8 1.1 1.6 2.3]';
y = [0.5 0.82 1.14 1.25 1.35 1.40]';
plot(t,y,'o'), grid on
```

Polynomiální regrese

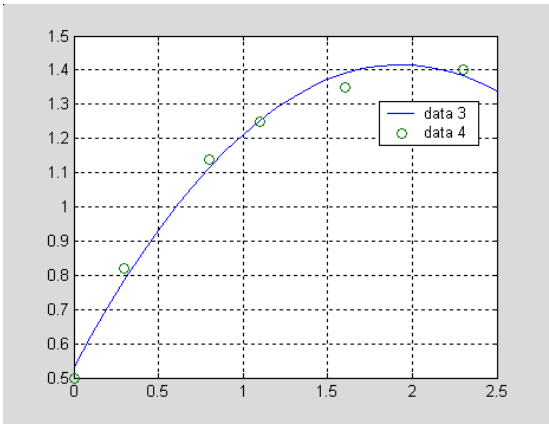
model:

$$y = a_0 + a_1 t + a_2 t^2$$

```
X = [ones(size(t))  t  t.^2]
a = X\y
a =   0.5318
       0.9191
      -0.2387

T = (0:0.1:2.5)';
Y = [ones(size(T))  T  T.^2]*a;

plot(T,Y,'-',t,y,'o'), grid on
```

Regrese lineární v parametrech

model:



$$y = a_0 + a_1 e^{-t} + a_2 t e^{-t}$$

```
X = [ones(size(t))  exp(-t)  t.*exp(-t)];
a = X\y
a =
    1.3974
  - 0.8988
    0.4097

T = (0:0.1:2.5)';
Y = [ones(size(T))  exp(-T)  T.*exp(-T)]*a;
plot(T,Y,'-',t,y,'o'), grid on
```

Násobná regrese
model:

$$y = a_0 + a_1 x_1 + a_2 x_2$$

```
   x1 = [.2 .5 .6 .8 1.0 1.1]';
   x2 = [.1 .3 .4 .9 1.1 1.4]';
y  = [.17 .26 .28 .23 .27 .24]';

   X = [ones(size(x1))  x1  x2];
   a = X\y
   a =
       0.1018
       0.4844
      -0.2847
```

K ověření modelu hledáme maximální absolutní hodnotu odchylky dat od modelu

```
Y = X*a;
MaxErr = max(abs(Y - y))
MaxErr =
  0.0038
```

23.0 Kořeny polynomu

Funkce pro práci s polynomy:

```
poly, polyval, polyfit, polyder, polyint, roots

A=[1 2 3;4 5 6;7 8 0]
A =
     1     2     3
     4     5     6
     7     8     0
```

Převod na charakteristický polynom a jeho koeficienty **,** kde A je rozměru n x n.

$$\det(\lambda I - A) = p_1\lambda^n + p_2\lambda^{n-1} + ... + p_n\lambda + p_{n+1}$$

```
p=poly(A)
p = 1.0000   -6.0000  -72.0000  -27.0000
```

Kořeny polynomu:

```
p(1)*x^3 + p(2)*x^2  + p(3)*x + p(4)=0

r=roots(p)
ans = 12.1229
      -5.7345

      -0.3884
```

Nebo také přímo z matice A:

```
ra= eig(A)
ra =  12.1229
      -0.3884
      -5.7345
```

Derivace polynomu:

```
p_der = polyder(p)
```

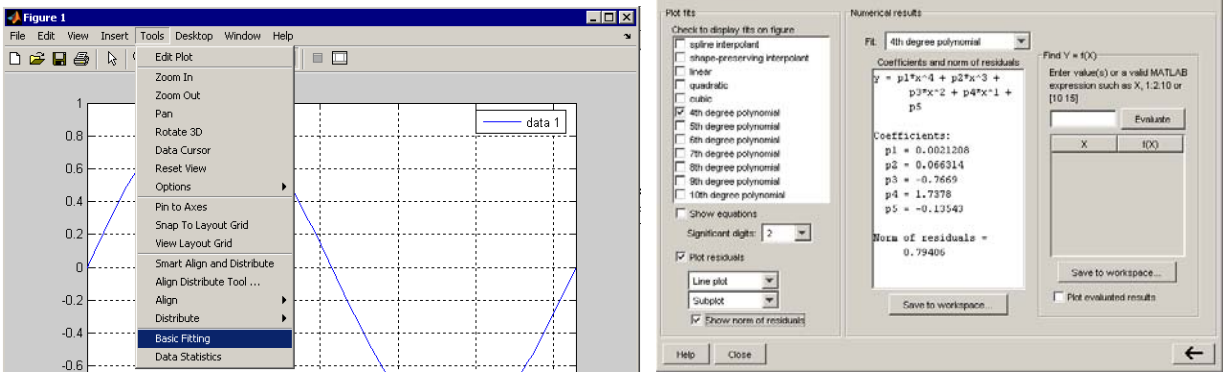
Integrace polynomu:

```
p_int = polyint(p)
```

Vykreslení průběhu polynomů pro vektor nezávisle proměnných x:

```
x=linspace(1,10,10)
x = 1     2     3     4     5     6     7     8     9    10

y=polyval(p,x);
y_der=polyval(p_der,x);

y_int=polyval(p_int,x);

subplot(311)
plot(x,y),grid
subplot(312)
plot(x,y_der),grid
subplot(313)
plot(x,y_int),grid

load census
sdate = (cdate - mean(cdate))./std(cdate)
[p2,S2] = polyfit(sdate,pop,2);
[pop2,del2] = polyval(p2,sdate,S2);
plot(cdate,pop,'+',cdate,pop2,'g-',cdate,pop2+2*del2,'r:',...
cdate,pop2-2*del2,'r:'), grid on
```

Využití nástrojů pro prokládání dat v objektu figure



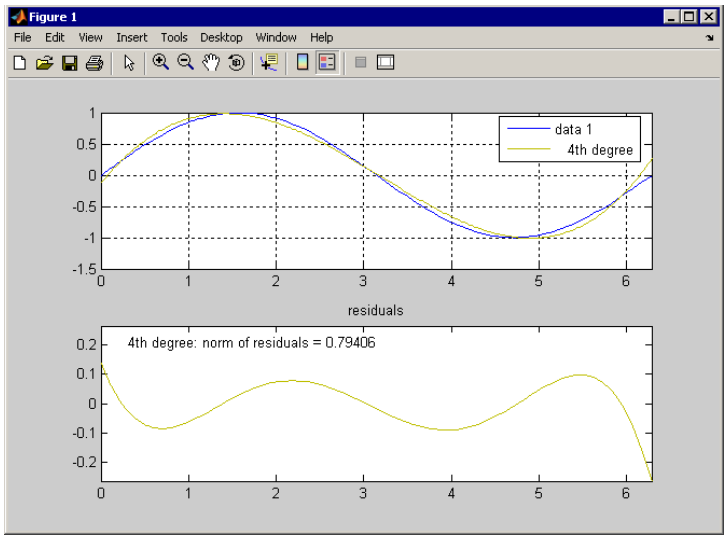


Figure 1
File Edit View Insert Tools Desktop Window Help
data 1
4th degree
residuals
4th degree: norm of residuals = 0.79406