Práce s vektory

Vytváření vektorů

Vektory se vytvářejí:

- pomocí hranatých závorek (viz 1. týden),
- pomocí operátoru dvojtečka (aritmetické posloupnosti) - viz generování vektorů,
- pomocí knihovních funkcí - viz generování vektorů,
- importováním dat do Workspace - pomocí funkcí `load` (z MAT nebo TXT souboru) nebo `xlsread` (z XLS souboru), viz 1. týden.

Přístup k prvkům vektoru

Pro přístup k jednotlivým prvkům vektoru musíme použít **kulaté závorky** s uvedením indexu, např.
```
>> v=[3 -2 3 5 1 -4 0 3]; v(5)
```
Vektory jsou v MATLABu **indexovány vždy od jedničky** (index smí být kladné přirozené číslo, jinak je oznámena chyba `Subscript indices must either be real positive integers or logicals.`). Při překročení délky vektoru je ohlášena chyba `Index exceeds matrix dimensions.`

Pro poslední prvek lze využít "index" `end`, např.:
```
>> v=[3; -2; 35; 1; -43]; v(end)
```

Zpracování prvků vektoru

Pro zpracování jednotlivých prvků vektoru můžeme využít cykly (nejlépe `for`), ale pokud je to možné, tak je **vždy rychlejší cykly vektorizovat**, tj. využít vhodný operátor (viz níže) nebo funkci (viz níže) a zpracovat vektor(y) "najednou".

Knihovní funkce pro práci s vektory

Generování vektorů

- pomocí operátoru *dvojtečka*:
  `v = a:b` ... od `a` do `b` s krokem 1, tj. řádkový vektor délky `b-a+1`, kde jsou hodnoty `a`, `a+1`, `a+2`, ..., `b`
  vyzkoušejte
  ```
  >> v1=3:20;
  >> v2=10:5 % prázdný vektor
  ```
  `v = a:k:b` ... od `a` do `b` s krokem `k`, tj. řádkový vektor délky `N=floor((b-a)/i)+1`, kde jsou hodnoty `a`, `a+k`,`a+2k`, ..., `a+(N-1)k` (resp. `b`, pokud je přesně dosažitelné z čísla `a` s krokem `k`)
  vyzkoušejte
  ```
  >> v1=3:4:20
  >> v2=0:0.1:0.5
  >> v3=10:-3:1
  >> v4=10:0:5 % prázdný vektor
  ```
- **`v = linspace(a,b)`** ... řádkový vektor délky 100, kde jsou hodnoty lineárně rozloženy od `a` (první prvek) do `b` (poslední prvek)

`v = linspace(a,b,n)` ... řádkový vektor délky `n`, kde jsou hodnoty lineárně rozloženy od `a` (první prvek) do `b` (poslední prvek)

- `v = logspace(a,b)` ... řádkový vektor délky 50, kde jsou hodnoty logaritmicky rozloženy od `10^a` do `10^b`
  `v = logspace(a,b,n)` ... řádkový vektor délky `n`, kde jsou hodnoty logaritmicky rozloženy od `10^a` do `10^b`
  `v = logspace(a,pi)` ... řádkový vektor, kde jsou hodnoty logaritmicky rozloženy od `10^a` do `pi` (π) - užitečné při zpracování obrazů
  vyzkoušejte
  ```
  >> v1=logspace(0,1);
  >> v2=logspace(0,1,4)
  >> v3=logspace(1,4,4)
  ```
- `v = ones(1,n)` nebo `ones(n,1)` ... řádkový/sloupcový vektor délky `n` obsahující samé jedničky
- `v = zeros(1,n)` nebo `zeros(n,1)` ... řádkový/sloupcový vektor délky `n` obsahující samé nuly
- `v = rand(1,n)` nebo `rand(n,1)` ... řádkový/sloupcový vektor délky `n` obsahující pseudonáhodná čísla

Zjištění rozměru vektoru

- `d = length(vektor)` ... počet prvků vektoru
- `[r,s] = size(vektor)` ... počet řádků a sloupců vektoru

Zjištění, zda proměnná je vektor

- `isscalar(proměnná)` ... je proměnná skalár (tj. číslo)?
- `isvector(proměnná)` ... je proměnná vektor?

Výpis vektoru

- `disp(vektor)` ... přímý výpis do Command Window
- `s = sprintf(vektor)` ... formátovaný převod vektoru na řetězec, např. `>> disp(sprintf('%d, ',v))`

Matematické funkce

A. skalární (aplikují se na každý prvek vektoru):
   - goniometrické: `sin(vektor)`, `cos(vektor)`, `tan(vektor)`, `cot(vektor)`,...
   - exponenciální: `sqrt(vektor)`, `exp(vektor)`, `log(vektor)`, `log10(vektor)`,...
   - pro práci s komplexními čísly: `abs(vektor)`, `real(vektor)`, `imag(vektor)`, `conj(vektor)`, `isreal(vektor)`,...
   - zaokrouhlovací: `fix(vektor)`, `round(vektor)`, `ceil(vektor)`, `floor(vektor)`,...

B. vektorové (aplikují se na celý vektor):
   - norma vektoru (eukleidovská): `norm(vektor)`
   - pro analýzu dat: `max(vektor)`, `min(vektor)`, `mean(vektor)`, `sum(vektor)`,...
   - další statistické: `median(vektor)`, `std(vektor)`, `var(vektor)`, `hist(vektor)`,...
   - řazení: `sort(vektor)`, `issorted(vektor)`

Operátory pro práci s vektory

Nejprve si zopakujeme operátory pro práci se skaláry:

1) aritmetické operátory			**2) relační operátory**			**3) logické operátory**		
a) unární			<	je menší	3<4	**a) unární**		
+	unární plus	+11	<=	je menší nebo rovno	3<=4	~	negace	~11
-	unární minus	-11	>	je větší	3>4	**b) binární se zkráceným vyhodnocováním**		
b) binární			>=	je větší nebo rovno	3>=4	&&	logický součin	7&&11
+	součet	7+11	==	je rovno (nepleťte si s přiřazením!)	3==4	||	logický součet	23||0
-	rozdíl	23-8	~=	je nerovno	3~=4	**c) binární bez zkráceného vyhodnocování**		
*	součin	3*4	Výsledkem je vždy logická 1 (true) nebo 0 (false).			&	logický součin	7&11
/	podíl	25/7				|	logický součet	23|0
^	mocnina	3^2				Výsledkem je vždy logická 1 (true) nebo 0 (false). Jakákoli nenulová hodnota je považována za logickou 1 (true), číslo nula je logická 0 (false).		

Nyní si uvedeme operátory pro práci s vektory:

1) aritmetické operátory

unární operátory			
operace	**operátor**	**význam**	**příklad**
plus	+	ponechá znaménko všech prvků vektoru	>> +[2,-4,7]
minus	-	změní znaménko všech prvků vektoru na opačné	>> -[2,-4,7]
transpozice v $\mathbf{C}^n$	'	transponuje a adjunguje (změní čísla na komplexně sdružená)	>> [2-3i,-4i,3,7+2i]'
transpozice v $\mathbf{R}^n$	.'	jen transponuje vektor	>> [2-2i,-4i,3,7+i].'
binární operátory			
operace	**operátor**	**význam**	**příklad**
součet	+	součet vektorů (sečte odpovídající si prvky) vektory musí mít stejný rozměr!	>> [1,1,6,-2]+[3,-2,5,0]
rozdíl	-	rozdíl vektorů (odečte odpovídající si prvky) vektory musí mít stejný rozměr!	>> [1,1,6,-2]-[3,-2,5,0]
součin	*	součin vektorů podle pravidel násobení matic z lineární algebry (nutné správné "vnitřní" rozměry!)	využití pro výpočet skalárního součinu: >> [1,-3,5]*[2;1;0] využití pro generování matic, např.: >> [1;2;3;4]*[1:10]

součin po prvcích	`.*`	pronásobí se odpovídající si prvky vektory musí mít stejný rozměr!	`>> [2,4,5,-1].*[1,-1,1,6]`
dělení	`/`	provádí se jako "dělení matic" (více příště)	budeme používat pouze v kombinaci vektor-skalár: `>> [6,12,4,5]/3`
dělení po prvcích	`./`	vydělí odpovídající si prvky vektorů vektory musí mít stejný rozměr!	`>> [3,5,-7,4]./[2,5,14,2]`
umocnění	`^`	pro vektory NELZE VYUŽÍT (`u^3` funguje totiž jako maticové `u*u*u`)	

umocnění po prvcích	`.^`	umocní každý prvek vektoru daným skalárem nebo odpovídajícím prvkem 2. vektoru (nutné stejné rozměry!)	`>> [3,5,-2,10].^2`
			`% vektor .^ skalár`
			`>> [3,5,-2,10].^[2,3,2,3]`
			`% vektor .^ vektor`

operátor dvojtečka (angl. colon operator)			
generování vektoru	`a:b` `a:k:b`	generuje aritmetickou posloupnost od `a` do `b` s krokem 1 nebo s krokem `k` !všechny vstupy jsou skaláry!	viz generování vektorů

U všech binárních operací pro vektory lze místo jednoho z vektorů použít skalár - pak se operace provede s každým prvkem vektoru a daným skalárem:

```
>> [1 -2 3 14] + 5
>> 10 - [1 -2 3 14]
>> [1 -2 3 14] * 2 % [1 -2 3 14] .* 2
>> [1 -2 3 14] ./ 10 % [1 -2 3 14] / 10
>> [1 -2 3 14] .^ 5
```

2) relační operátory

Všechny relační operátory se ke vstupním vektorům STEJNÉ DÉLKY chovají skalárně (tj. aplikují se na odpovídající si prvky), např.:

```
>> [3,2,-5,10]>[2,3,2,3]
>> [3,2,-5,10]~=[2,0,2,3]
>> [2 1 5 4 0]<=[0:5] % chyba
```

Relační operátory lze použít také pro vektor-skalár a vracejí vektor stejné délky jako vstupní vektor(y). Výsledek obsahuje logické 1 a 0 (true/false):

```
>> 2<=[0:5]
```

3) logické operátory

Logické operátory ~(negace), `&` (logický součin) a `|` (logický součet) se ke vstupním vektorům chovají skalárně (tj. aplikují se na odpovídající si prvky). Vracejí vektor logických 1 a 0 stejné délky, jakou měly vstupní vektory. Jedním ze vstupů může být i skalár.
Např.:

```
>> ~[3,2,0,-5,10,0,1]
>> [3,2,-5,10]&[2,0,2,3]
>> [1,2,0,4,-3,-10]|0
```

Pozor: operátory se zkráceným vyhodnocováním (`&&` a `||`) nelze pro vektory použít!

Priorita operátorů určených pro skaláry a vektory

	symbol	poznámka
1.	`()`	závorky
2.	`' .' ^ .^`	transpozice, umocnění
3.	`+ - ~`	unární plus, unární minus, negace
4.	`* .* / ./`	násobení, dělení
5.	`+ -`	sčítání, odčítání
6.	`:`	dvojtečka
7.	`< <= > >= == ~=`	relační operátory
8.	`&`	logický součin, AND
9.	`|`	logický součet, OR
10.	`&&`	logický součin se zkráceným vyhodnocením
11.	`||`	logický součet se zkráceným vyhodnocením

Poznámka: operátorů je v MATLABu víc. Kompletní přehled si uděláme příště v rámci *Práce s maticemi*. (Pro nedočkavé: `>> help precedence`.)

Poznámky k operátorům pro práci s vektory

- Operátory s tečkou se musí psát dohromady (bez mezery)!!!
- **Skalární součin** 2 vektorů v $\mathbf{R}^n$ lze vypočítat několika způsoby:
  - pomocí operátoru `*`, kdy první vektor musí být řádkový a druhý sloupcový (oba stejné délky):
    ```
    u = [2 1 4 -3]; v = [5 -7 2 1]'; u*v  nebo
    u = [2 1 4 -3]; v = [5; -7; 2; 1]; u*v  nebo
    u = [2 1 4 -3]; v = [5 -7 2 1]; u*v' % 8
    ```
  - pomocí funkce **`dot(v,w)`** (vektory musí být stejně dlouhé, lze kombinovat řádkový se sloupcovým):
    ```
    u = [2 1 4 -3]; v = [5 -7 2 1]; dot(u,v)
    ```
  - pomocí cyklu `for` - toto je ale v MATLABu nejpomalejší možnost!
    ```
    u = [2 1 4 -3]; v = [5 -7 2 1];
    s = 0;
    for i=1:length(u)
        s = u(i)*v(i);
    ```
- **Vektorový součin** 2 vektorů v $\mathbf{R}^3$ vrací funkce **`cross(v,w)`**. Výsledkem je tříprvkový vektor.

Práce s řetězci

Řetězce neboli pole znaků, (anglicky character arrays nebo strings) jsou v podstatě vektory, které obsahují znaky (většinou znaky anglické abecedy, protože v případě použití "českých" znaků mívají starší verze MATLABu problémy se správným zobrazením).

Vytváření řetězců

Pro vytvoření řetězce slouží **apostrofy** (anglické!), do kterých se vepíše obsah řetězce (například `'ahoj'` nebo `'Toto je nejaky dlouhy text.'`), přičemž takto vytvořený řetězec můžeme uložit do proměnné:

```
>> s1 = 'nejaky text';
>> nazev = 'Programovani v MATLABu'
```

Ve Workspace se řetězec objeví jako proměnná typu `char`, resp. `char array` (starší verze).

Poznámky:

- ukládání řetězců do matic:
  - je nutné, aby všechny řádky (= řetězce) měly stejnou délku! Doplňuje se většinou mezerami, tedy:
  ```
  >> jmena = ['Ilona'; 'Jana '; 'Ludva'; 'Eva  '; 'Pepa '];
  >> str2 = ['Jana'; 'Jarmila'] % chyba!
  ```
  Abychom nemuseli počítat mezery, můžeme využít funkci **`char`** s proměnným počtem vstupů, tedy
  ```
  >> jmena2 = char('Ilona','Jana ','Ludva','Eva','Pepa');
  ```
  Přístup k písmenu "J": `>> jmena2(2,1)`
- ukládání řetězců do pole buněk:
  - tento způsob je elegantnější, neboť není třeba doplňovat řetězce mezerami na jednotnou délku:
  ```
  >> jmena3 = {'Ilona'; 'Jana '; 'Ludva'; 'Eva'; 'Pepa'};
  ```
  Přístup k písmenu "J": `>> jmena3{2}(1)`

Výpis řetězců

Nyní si v podstatě shrneme to, co jsme v minulých týdnech už používali. Řetězce lze vypisovat:

- pomocí názvu proměnné

  Pokud máme řetězec uložený v nějaké proměnné, můžeme jej vypsat úplně stejným způsobem jako obsah kterékoli jiné proměnné, tedy *zadáním jména proměnné* (například `>> p1`).

- pomocí funkce `disp`

  Tato funkce se specializuje na výpis řetězců. V rámci volání **`disp(p1)`** můžeme jako parametr *`p1`* uvést řetězec nebo název proměnné (která nemusí obsahovat řetězec, ale obecně vektor).

- pomocí funkce `error` (výpis chybového hlášení)

  Pro výpis chybových hlášení (a zároveň i pro UKONČENÍ skriptu/funkce) použijeme funkci **`error(p1)`**. Jejím parametrem je chybové hlášení (řetězec), které se do Command Window vypisuje červeně.

Spojování řetězců

Řetězce lze spojovat do větších celků několika způsoby:

- horizontálně (viz také vytváření řetězců):
  - pomocí hranatých závorek a čárky/mezery, např.:
  ```
  >> s1 = 'Ahoj'; s2 = 'lidi';
  >> pozdrav = [s1 ' ' s2 '!'] % Ahoj lidi!
  ```
  - pomocí funkce **`strcat`**:
  ```
  >> s1 = 'Ahoj'; s2 = 'lidi';
  >> pozdrav = strcat(s1, ' ', s2, '!') % Ahoj lidi!
  ```

  - pomocí funkce **sprintf**:
```
>> s1 = 'Ahoj'; s2 = 'lidi';
>> pozdrav = sprintf('%s %s!', s1, s2) % Ahoj lidi!
```

- vertikálně:
  - pomocí hranatých závorek a středníku (nutno dodržet STEJNOU délku všech řetězců!):
```
>> str2=['ahoj    '; 'nazdárek'] % matice se 2 řádky
```
  - pomocí funkce **char**:
```
>> s1 = 'Ahoj'; s2 = 'lidi';
>> pozdravy = char(s1, s2, 'nazdárek') % matice se 3 řádky
```
  - pomocí funkce **strvcat**:
```
>> s1 = 'Ahoj'; s2 = 'lidi';
>> pozdravy = strvcat(s1, s2, 'nazdárek') % matice se 3 řádky
```

Konverze čísel na řetězec a obráceně

- konverze libovolného skaláru/vektoru na řetězec se zadáním formátu - funkce **sprintf**
- konverze celočíselného skaláru na řetězec - funkce **int2str**
- konverze skaláru/vektoru/matice na řetězec - funkce **num2str** (jako druhý parametr lze zadat přesnost):
```
>> c=num2str(15.712)
>> c1=num2str([15.712; 3.5; -1.14])
% srovnejte:
>> c2=int2str([15.712; 3.5; -1.14])
```
- konverze řetězce na číslo - funkce **str2num**:
```
>> c=str2num('-123.456e-2')
```
- konverze řetězce na čísla (podle ASCII kódu znaků) - funkce **str2double** nebo **double**
```
>> c=double('ahoj lidičky')
```
- konverze přirozeného čísla/vektoru na řetězec (dle ASCII kódu) - funkce **char**
```
>> ascii = char(reshape(32:127, 32, 3)') % ASCII tabulka ve 3 řádcích
```

Převod řetězce na velká/malá písmena

- převod na velká písmena - funkce **upper**
- převod na malá písmena - funkce **lower**

Funguje pro znakovou sadu ISO Latin-1.

Porovnávání řetězců

- porovnání 2 řetězců - funkce **strcmp** nebo case-insensitivní **strcmpi**
```
>> str = 'Jana'; str2 = 'Jarmila'; str3 = char([74 97 110 97]); % Jana
>> strcmp(str,str3)
>> strcmp(str2,str3)
% srovnejte:
>> str==str3
```
- porovnání prvních *n* znaků 2 řetězců - funkce **strncmp** nebo case-insensitivní **strncmpi**
```
>> strncmp(str,str2,2)
>> strncmp(str,str2,3)
```

Vyhledávání a nahrazování v řetězci

- vyhledávání - funkce **`findstr`**
  `k = findstr(kde, co)` ... case-sensitive hledání řetězce `co` v řetězci `kde`. Funkce vrací vektor pozic výskytu začátku hledaného řetězce:
```
>> textik='Mila maminko, mam se dobre!'
>> findstr(textik,'mam') % 6 15
>> findstr(textik,'b') % prázdný výstup
>> findstr(textik,'M') % 1
```
- vyhledávání lze provést také funkcí **`strfind`**, která může dostat i pole buněk. Více v nápovědě.
- nahrazování - funkce **`strrep`**
  `s = strrep(kde,co,čím)` ... case-sensitive nahrazování:
```
>> nakup='3 jogurty, 3 rohliky, 1 mleko'
>> strrep(nakup,'3','4')
```

Regulární výrazy

MATLAB nabízí také funkce pro pokročilejší vyhledávání, resp. nahrazování pomocí regulárních výrazů. Více naleznete v nápovědě k funkcím `regexp`, `regexpi` a `regexprep`.

Další funkce pro práci s řetězci

Kompletní přehled názvů funkcí pro práci s řetězci vypíše příkaz `>> help strfun`.

2D grafika

MATLAB umožňuje snadnou vizualizaci dat. Jeho možnosti pro tvorbu dvourozměrných grafů jsou rozsáhlé - dnes si ukážeme část funkcí, které budeme používat při kreslení 2D grafů.

Při práci s 2D grafikou využíváme grafický režim MATLABu, který má svá vlastní grafická okna (Figure 4, Figure 12 apod.)

Postup:

1. příprava dat (nezávisle proměnná, závisle proměnná/é)
2. výběr grafického okna, případně podokna
3. vykreslení grafu
4. popis grafu
5. nastavení os, mřížky a případné další úpravy grafu
6. uložení grafu do souboru (obrázek)

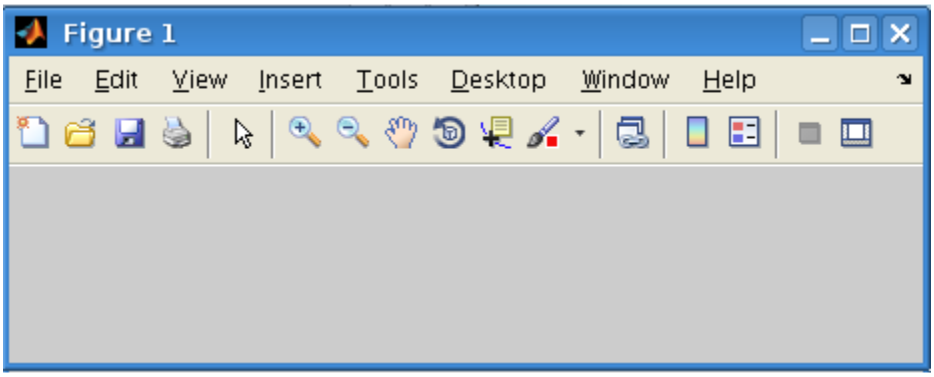
Následuje popis funkcí a příkazů, které lze pro uvedené účely použít.

Výběr grafického okna - funkce *figure*

Grafické funkce automaticky otevřou nové grafické okno, pokud zatím žádné neexistuje (`Figure 1`). V případě, že je již nějaké grafické okno otevřeno, je použito pro vykreslování to, které je *aktivní* (jedná se

o poslední okno, které bylo otevřeno nebo na které bylo kliknuto). Chceme-li vykreslovat do úplně nového okna (přestože jiná již existují) nebo vykreslovat do okna se zvoleným číslem, použijeme funkci `figure` s jedním nebo žádným parametrem:

`figure` - otevře úplně nové grafické okno a učiní jej aktivním:



`figure(n)` - učiní aktivním (event. jej i otevře) grafické okno s číslem *n*

Zavření grafického okna - funkce *close*

Potřebujeme-li grafické okno zavřít, můžeme to provést pomocí myši (křížek v pravém horním rohu okna) nebo pomocí příkazu/funkce `close`:

`close` - zavře aktivní grafické okno

`close(X)` - zavře grafické okno s názvem `Figure` *X* (okno nemusí být aktivní, ale musí existovat - jinak MATLAB ohlásí chybu)

`close all` - zavře všechna grafická okna (netýká se tedy ani okna M-editoru, ani okna MATLABu)

Kreslení X,Y-bodového grafu - funkce *plot*

Funkce `plot` kreslí graf zadaný pomocí hodnot ve vektoru/vektorech. K tomu si otevře nové grafické okno (pokud žádné neexistovalo) nebo začne kreslit do aktivního okna, ve kterém smaže všechny předchozí grafy (pokud nebylo nastaveno `hold on` - viz níže).

Syntaxe:

1. `plot(y)` ... vykreslí hodnoty vektoru `y` v závislosti na jejich indexu (pořadí)
2. `plot(x,y)` ... vykreslí hodnoty vektoru `y` v závislosti na hodnotách vektoru `x`
3. `plot(x,y,str)` ... vykreslí hodnoty vektoru `y` v závislosti na hodnotách vektoru `x` a pomocí řetězce `str` ovlivní výsledný vzhled grafu: barvu značky a čáry, typ značky a typ čáry (viz níže)
4. `plot(x,y1,str1, x,y2,str2, ...)` ... vykreslí více grafů, tj. hodnoty vektorů `y1` a `y2` v závislosti na hodnotách vektoru `x` a pomocí řetězců `str1` nebo `str2` můžeme (ale nemusíme) ovlivní výsledný vzhled grafu. Povinná je vždy dvojice `x,y`, řetězec za nimi následovat nemusí
5. `plot(x,y, vlast1,hod1, vlast2,hod2,...)` ... vykreslí jeden graf, přičemž můžeme ovlivnit jeho vzhled pomocí dvojic `vlastnost-hodnota`, kde `vlastnost` je řetězec (viz objekty a 2D grafika) a `hodnota` je nastavovaná nová hodnota (skalár, vektor, řetězec,... podle dané vlastnosti)

Proměnná `str` je řetězec obsahující jednu až tři hodnoty vlastností, v pořadí barva, typ značky a typ čáry. Implicitně je nastavena barva 'b' (modrá), značka žádná a plná čára. Hodnoty jednotlivých vlastností jsou v tabulce 1.

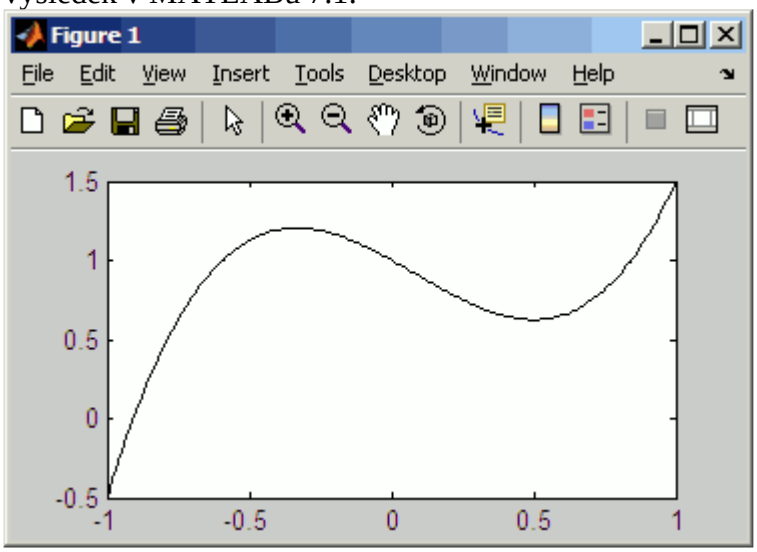
Tabulka 1: hodnoty `str`

Barva		Značka		Typ čáry	
b	modrá	.	tečka	-	plná
g	zelená	o	kolečko	:	tečkovaná
r	červená	x	křížek	-.	čerchovaná
c	sv.modrá	+	křížek	--	čárkovanaá
m	fialová	*	hvězdička		
y	žlutá	s	čtverec		
k	černá	d	diamant		
		v	trojúhelník nahoru		
		^	trojúhelník dolů		
		<	trojúhelník doleva		
		>	trojúhelník doprava		
		p	pentagram		
		h	hexagram		

Příklad - graf polynomu černou plnou čárou:

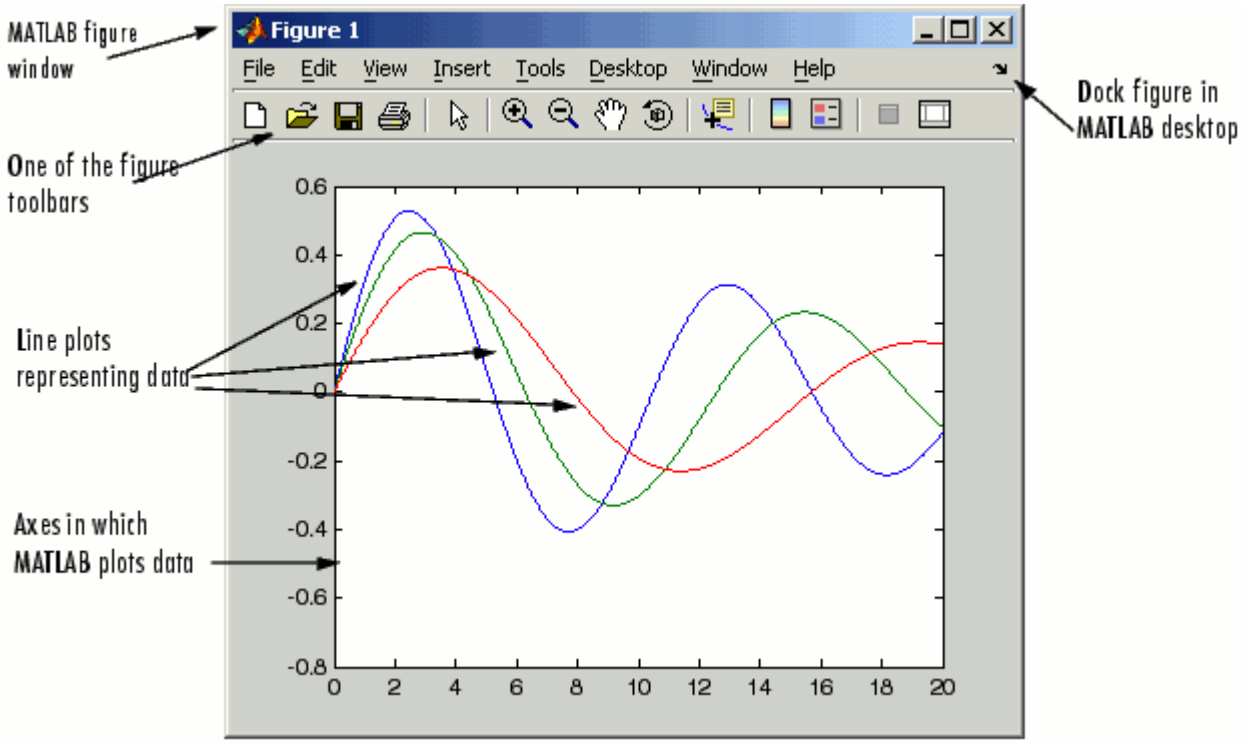
```
>> x=-1:0.01:1;
>> plot(x,2*x.^3-.5*x.^2-x+1,'k')
```

výsledek v MATLABu 7.1:



"Anatomie" grafu

- okno (figure), souřadnice (axes), čáry (lines,line plots). Viz také objekty a 2D grafika



Další typy X,Y-bodových (čárových) grafů

- graf s vedlejší osou *y*
  `plotyy(x,y1,x,y2)` ... graf `x,y1` vynášený na osu *y* vlevo a graf `x,y2` vynášený na osu *y* vpravo.
  Vyzkoušejte:
```
>> x = 0:0.1:6;
>> y1 = sin(x);
>> y2 = exp(x);
>> plotyy(x,y1,x,y2)
```
- graf s logaritmickou stupnicí na ose *y*:
  `semilogy(x,y)` ... funguje jako `plot`, ale na ose *y* je zvolena logaritmická stupnice o základu 10, na ose *x* zůstává stupnice lineární.
  Vyzkoušejte:
```
>> x = 0:0.1:6; % nezapomínejme na středník - dat je hodně!
>> y = 10.^x; % umocňování člen po členu operátorem s tečkou ( .^ )
>> semilogy(x,y)
```
- graf s logaritmickou stupnicí na ose *x*:
  `semilogx(x,y)` ... funguje jako `plot`, ale na ose *x* je zvolena logaritmická stupnice o základu 10.
- graf s logaritmickou stupnicí na obou osách:
  `loglog(x,y)` ... funguje jako `plot`, ale na obou osách je logaritmická stupnice o základu 10.
- graf v polárních souřadnicích:
  `polar(theta,rho)`
  Vyzkoušejte:
```
>> t = 0:.01:2*pi;
>> polar(t, sin(2*t).*cos(2*t), '--r') % čárkovaná červená čára
```

Popis grafu - funkce *title, xlabel, ylabel, text* a *legend*

`title(`*`popisek`*`)` - přidání názvu grafu

`xlabel(`*`popisek`*`)` - přidání popisu osy *x*
`ylabel(`*`popisek`*`)` - přidání popisu osy *y* (otočen o 90°)

`text(`*`x,y,popisek`*`)` - přidání textu do grafu: *x* a *y* jsou souřadnice bodu, kam je řetězec `popisek` umístěn (levým dolním rohem).
`gtext(`*`popisek`*`)` - umístění textu pomocí myši (neznáme-li souřadnice).

Ve všech případech je *`popisek`* libovolný řetězec, který může obsahovat TeXovské znaky (viz tabulka 2). V případě, že potřebujete vypsat řetězec podle pravidel LaTeXu, musíte nastavit správný interpret (vlastnost `Interpreter` s hodnotou `Latex`).

Chceme-li přidat ke grafu legendu (popsat více grafů v jednom okně), použijeme funkci
`legend(text1,text2,...)`, resp.
`legend(text1,text2,...,'Location',kam)`,
jejímiž parametry jsou texty popisující všechny vykreslené grafy (v pořadí daném vykreslováním těchto grafů) a volitelně lze zadat i umístění (např. `kam='SouthWest'` - více naleznete v nápovědě. Implicitní umístění je vpravo nahoře). Místo textů lze také použít pole buněk - více v nápovědě.
Příklad:
```
legend('graf1','graf2','graf3','graf4')
legend('graf1','graf2','graf3','graf4','Location','NorthWest') % vlevo nahoře
```

Tabulka 2: některé speciální znaky (podle TeXovské konvence)

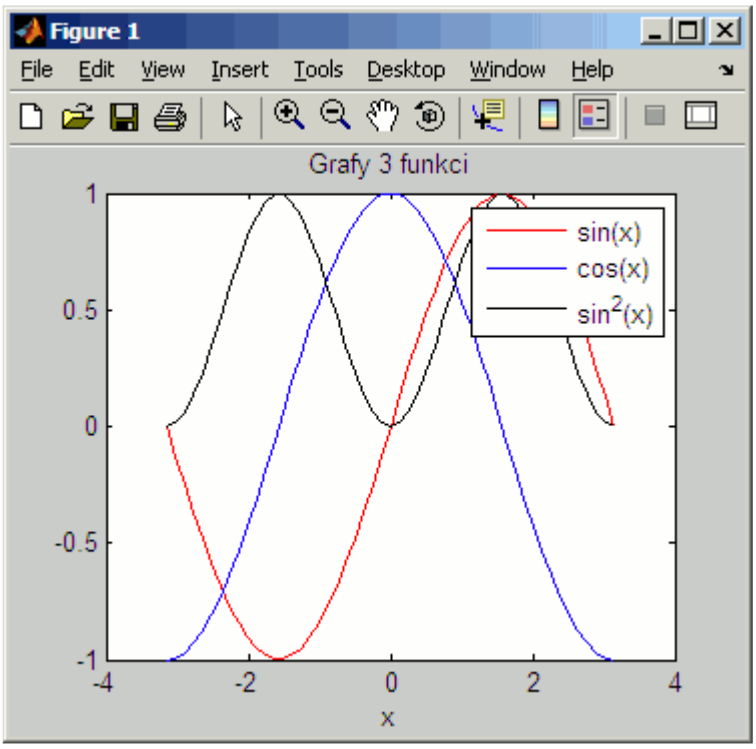
dolní index	_ např. `f_1` nebo `x_{12}`				
horní index	^ např. `e^3` nebo `e^{-x}`				
Znak	**Zápis**	**Znak**	**Zápis**	**Znak**	**Zápis**
α	`\alpha`	β	`\beta`	γ	`\gamma`
δ	`\delta`	ε	`\epsilon`	ω	`\omega`
λ	`\lambda`	ξ	`\xi`	π	`\pi`
ρ	`\rho`	σ	`\sigma`	τ	`\tau`
σ	`\sigma`	Δ	`\Delta`	Σ	`\Sigma`
∇	`\nabla`	∂	`\partial`	∞	`\infty`
$\surd$	`\surd`	$\int$	`\int`	$\neq$	`\neq`
$\in$	`\in`	$\subset$	`\subset`	$\subseteq$	`\subseteq`
$\leq$	`\leq`	$\geq$	`\geq`	$\uparrow$	`\uparrow`
$\wedge$	`\wedge`	$\vee$	`\vee`	$\downarrow$	`\downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\leftarrow$	`\leftarrow`	$\rightarrow$	`\rightarrow`
$\neg$	`\neg`	$\forall$	`\forall`	$\exists$	`\exists`

Více grafů v jednom okně (funkce *plot* nebo příkaz *hold*)

Pokud chceme vykreslovat více grafů do téhož (pod)okna, můžeme použít dva způsoby:

1. funkci `plot`, které předáme jako parametry všechny vykreslované grafy (u každého může být také uvedena barva, značka a typ čáry):
```
>> x=-pi:pi/25:pi; % nezavisle promenna
>> y1=sin(x); y2=cos(x); y3=sin(x).^2;% 3 funkce
>> plot(x,y1,'r',x,y2,'b',x,y3,'k') % tri v jednom (cerveny, modry a cerny graf)
>> plot(x,y1,'r',x,y2,x,y3,'k') % totez - modra barva se nemusi zadavat
>> xlabel('x') % popis osy x
>> title('Grafy 3 funkci') % nazev grafu
>> legend('sin(x)', 'cos(x)', 'sin^2(x)') % pridani legendy ke grafu
```
2. příkaz **`hold on`**, který "zapne" možnost přikreslovat do grafického okna jiné grafy, aniž by "zmizel" graf již nakreslený. Standardní stav (kdy se graf překresluje) pak docílíme příkazem **`hold off`**:
```
>> x=-pi:pi/25:pi; % nezavisle promenna
>> y1=sin(x); y2=cos(x); y3=sin(x).^2;% 3 funkce
>> plot(x,y1,'r') % prvni graf (cerveny)
>> hold on % prikreslime neco dalsiho
>> plot(x,y2) % prvni graf (modry - netreba udavat barvu)
>> plot(x,y3,'k') % treti graf (cerny)
>> xlabel('x') % popis osy x
>> title('Grafy 3 funkci') % nazev grafu
>> legend('sin(x)', 'cos(x)', 'sin^2(x)') % pridani legendy ke grafu
>> hold off % uz nic prikreslovat nebudeme
```

Výsledek (v obou případech):



Zobrazení mřížky - příkaz *grid*

`grid on` - zobrazí mřížku
`grid off` - vypne zobrazení mřížky (implicitně nastaveno)

Volba zobrazení os - funkce *axis, xlim* a *ylim*

Pro změnu rozsahu osy *x* (zkrácení nebo prodloužení) použijeme příkaz `xlim(v)`, kde v je vektor obsahující minimální a maximální hodnotu na ose *x*.
Příklady:

```
>> xlim([-5 2.3]) % 1. zpusob
>> osa_x=[-5 2.3]; xlim(osa_x) % 2. zpusob
```

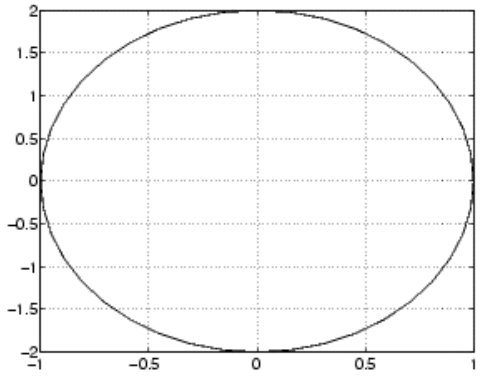
Pro změnu rozsahu osy *y* (zkrácení nebo prodloužení) použijeme příkaz `ylim(v)`, kde v je vektor obsahující minimální a maximální hodnotu na ose *y*.
Příklady:

```
>> ylim([0.5 12.5])% 1. zpusob
>> osa_y=[0.5 12.5]; ylim(osa_y) % 2. zpusob
```

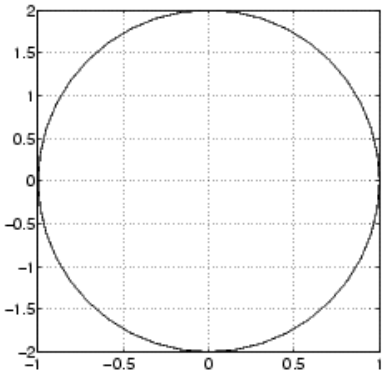
Pro změnu rozsahu obou os (*x* i *y*) lze použít funkci `axis(v)`, kde v je vektor obsahující minimální hodnotu na ose *x*, maximální hodnotu na ose *x*, minimální hodnotu na ose *y* a maximální hodnotu na ose *y*.
Příklady:

```
>> axis([-5 2.3 0.5 12.5]) % 1. zpusob
>> rozsah=[-5 2.3 0.5 12.5]; axis(rozsah) % 2. zpusob
```

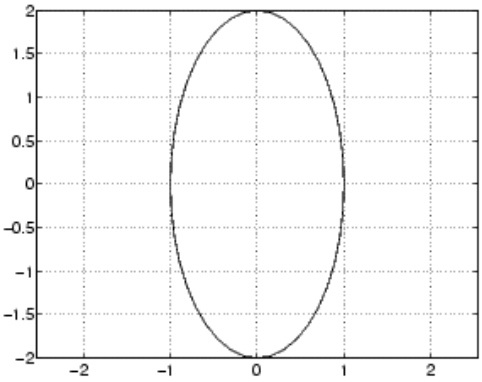
Pro vypnutí obou os lze použít funkci `axis` jako příkaz `axis off` a zpětně zapnout `axis on`. Pro nastavení ekvidistantních os (stejné jednotky na obou osách) použijte příkaz `axis equal`. Pro zformátování os do čtverce (jednotky se upraví automaticky) použijte `axis square`:



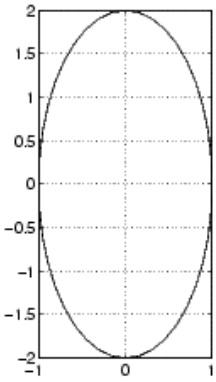
axis normal



axis square



axis equal



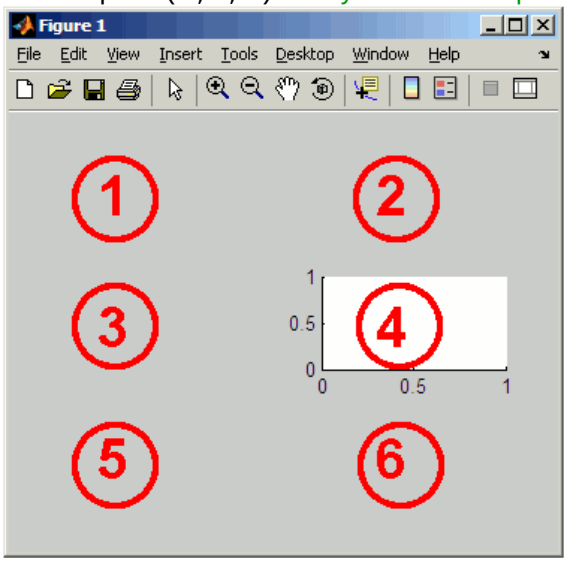
axis equal tight

Rozdělení grafického okna na části - funkce *subplot*

Pokud potřebujeme do jednoho grafického okna umístit více navzájem nezávislých grafů, použijeme funkci `subplot`, která vymezí konkrétní část grafického okna a aktivuje ji pro vykreslování:

`subplot(m,n,c)` - funkce si fiktivně rozdělí graf na *m* řádků a *n* sloupců, očísluje si jednotlivé "buňky" (zleva doprava, odshora dolů) a aktivuje právě JEDNO podokno, a to *c*-té podokno:

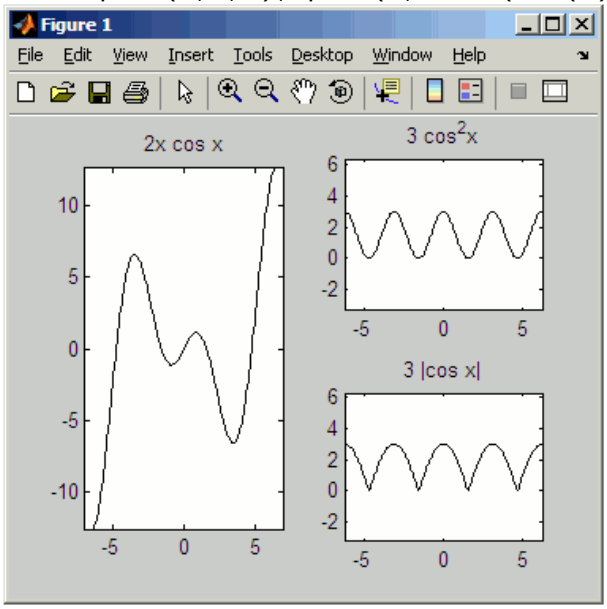
```
>> subplot(3,2,4) % vybíráme 4. podokno při rozdělení 3x2
```



Funkce `subplot` vytváří pouze JEDNO podokno, takže ji většinou musíme volat vícekrát. V případě, že by došlo ke kolizi, vítězí pozdější podokno (tj. je-li na daném místě již jiné podokno, resp. část jiného podokna, bude (celé) staré podokno odstraněno).

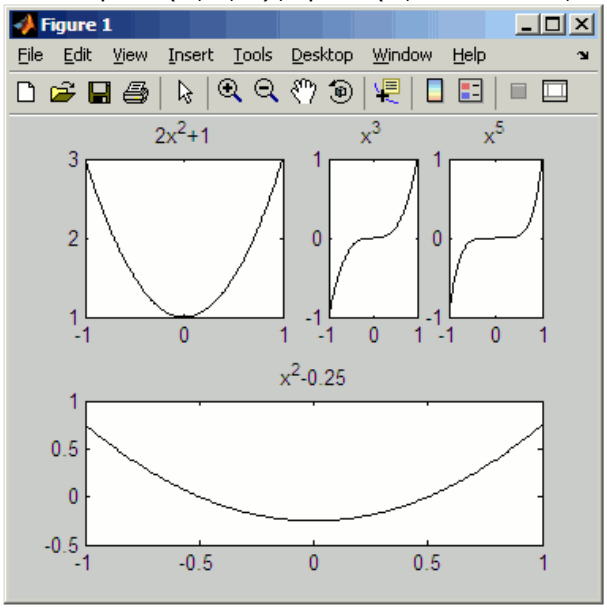
Příklady (výsledky jsou v MATLABu 7.1):

```
>> x=-2*pi:pi/50:2*pi;
>> subplot(1,2,1); plot(x,2*x.*cos(x),'k'), title('2x cos x'), axis equal
>> subplot(2,2,2); plot(x,3*cos(x).^2,'k'), title('3 cos^2x'), axis equal
>> subplot(2,2,4); plot(x,3*abs(cos(x)),'k'), title('3 |cos x|'), axis equal
```



Lze kombinovat podokna různé délky či šířky, pouze se nesmějí vzájemně překrývat:

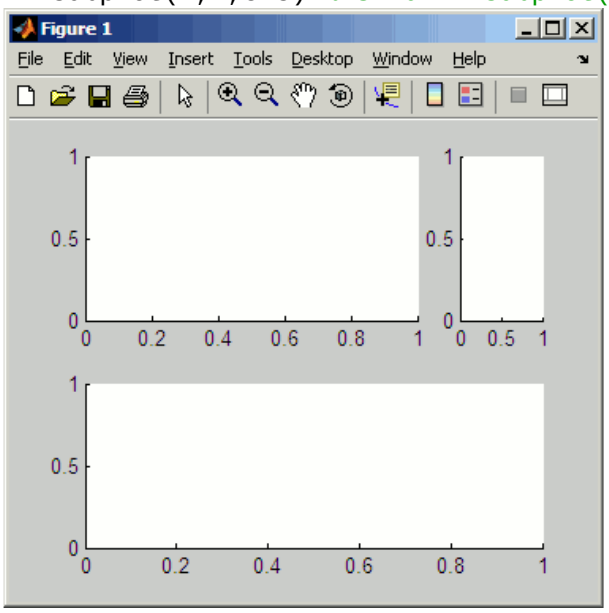
```
>> x=-1:0.01:1;
>> subplot(2,2,1); plot(x,2*x.^2+1,'k'), title('2x^2+1')
>> subplot(2,4,3); plot(x,x.^3,'k'), title('x^3')
>> subplot(2,4,4); plot(x,x.^5,'k'), title('x^5')
>> subplot(2,1,2); plot(x,x.^2-0.25,'k'), title('x^2-0.25')
```



Vytvářené podokno lze "natáhnout přes několik řádků/sloupců", např.:

```
>> subplot(2,4,1:3) % přes 3 sloupce
>> subplot(2,4,4)
```

```
>> subplot(2,4,5:8) % STEJNÉ: subplot(2,1,2)
```



Nástroje grafického okna

Každé grafické okno můžeme upravovat pomocí panelu nástrojů (není-li zobrazen, zaškrtneme Figure Toolbar v menu View):

MATLAB 6.5: 

MATLAB 7.0: 

MATLAB 7.6: 

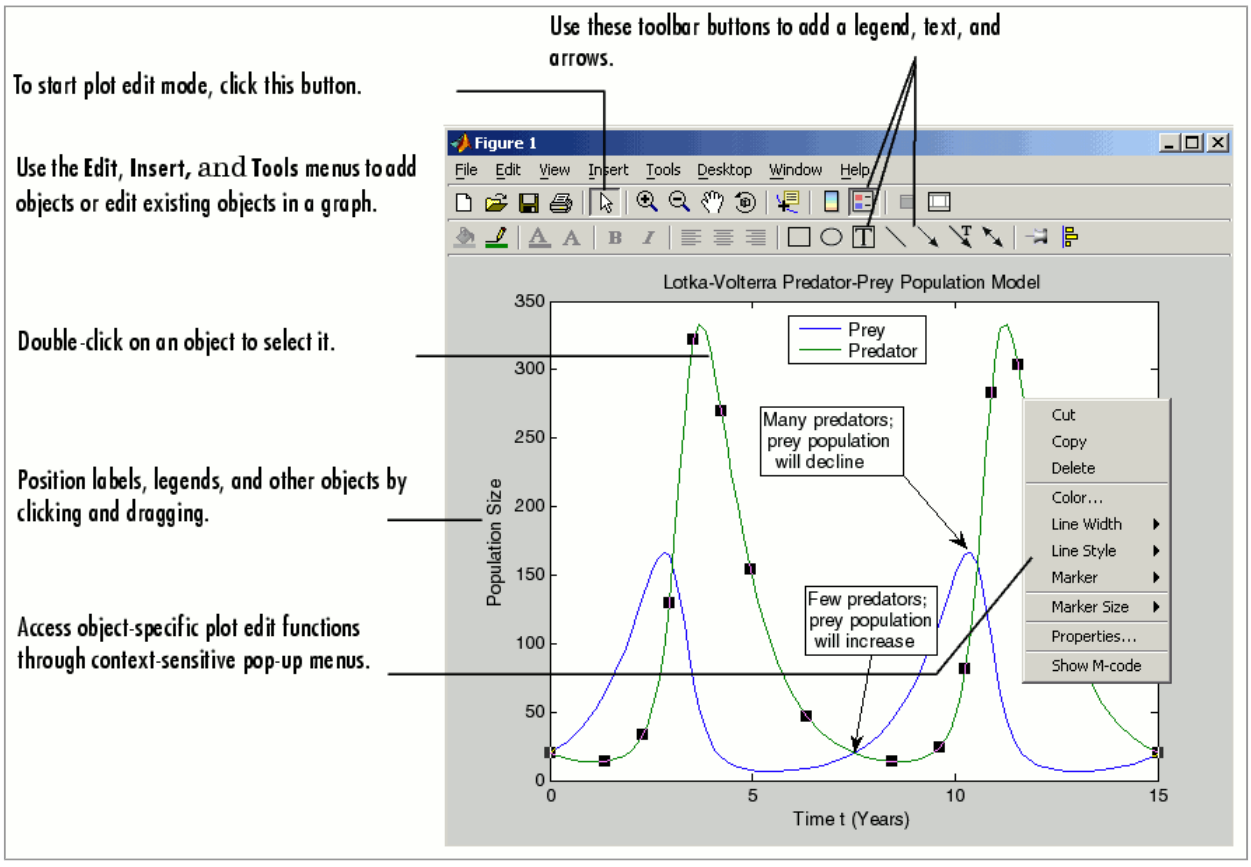
Význam jednotlivých nástrojů Figure Toolbaru:

MATLAB 6.5	MATLAB 7.0	MATLAB 7.6	význam
			otevře nové grafické okno
			otevře uložený graf (`*.fig`)
			uloží graf do souboru (`*.fig`)
			odešle graf na tiskárnu
			umožňuje měnit vlastnosti objektů v grafickém okně
	menu Insert - TextBox		umožňuje přidat do grafu text
	menu Insert - Arrow		umožňuje přidat do grafu šipky
	menu Insert - Line		umožňuje přidat do grafu čáry
			lupa - zvětšení v místě kliknutí
			lupa - zmenšení v místě kliknutí
nemá			posun grafu
			rotace (efektní jen u 3D grafů)
nemá			umožňuje odečíst souřadnice bodu čáry grafu

nemá			angl.: data brush
nemá			angl.: data link
nemá			zapne/vypne Colorbar
nemá			zapne/vypne legendu
nemá			vypne nástroje (Plot Tools) - nyní neaktivní
nemá			zobrazí nástroje (Plot Tools) - nyní aktivní

Editace grafu pomocí nástroje *Edit plot*

1. vybereme nástroj *Edit plot* (klikneme na)
2. kliknutím levou myší vybíráme objekty - např. plochu os (axes) nebo čáru grafu (line)
3. pravým tlačítkem myši zobrazíme kontextové menu (viz obr. 1, kde je navíc i postup v angličtině)
4. dvojklikem zobrazíme *Property editor* pro vybraný objekt: lze nastavovat vlastnosti, které má vybraný objekt (viz též objekty a 2D grafika)



Obr. 1 - editace grafu

Uložení grafu do souboru

A. **uložení grafu jako obrázek**, který je možno vkládat do dokumentů (texty v MS Wordu, www stránky,...):
obsah grafického okna uložíme do souboru následujícím způsobem:

1. menu File→Save, resp. Save as... (v MATLABu 6.5 položka Export...)
2. vyplnění údajů v otevřeném dialogovém okně:
   - výběr cílového adresáře
   - výběr typu souboru (např. pro vložení do MS Wordu lze použít formát BMP (bezeztrátový), JPG nebo PNG, pro LaTeX formát EPS...)
   - zadání jména souboru
3. aktivace tlačítka Save (tím se vytvoří obrázkový soubor; někdy uložení trvá déle, a proto *počkejte* se zavíráním grafických oken, dokud nezmizí přesýpací hodiny na kurzoru myši)

Poznámka: z příkazové řádky umí tisknout funkce `print` nebo `saveas` (ta jen do souboru)
např. `>> print -depsc -tiff -r200 graf_soubor % uloží EPS obrázek`

B. **uložení grafu jako soubor `*.fig`**, který umí otevřít/zobrazit MATLAB:

1. menu File→Save (nebo Save as...)
2. výběr cílového adresáře
3. kontrola typu souboru (`*.fig`)
4. zadání jména souboru
5. aktivace tlačítka Save

Poznámka: soubor typu FIG se dá kdykoli otevřít v MATLABu (např. přes menu File→Open) - otevře se uložené grafické okno.

Objekty a 2D grafika

V grafickém režimu MATLABu lze každou část okna chápat jako objekt, který má svůj *handle* (klika, rukověť, "ovladač"). Pomocí handlu lze ovlivňovat vlastnosti každého objektu (viz níže: funkce `set`). Např. každé grafické okno otevřené pomocí `figure` je objektem typu Figure, jeho potomkem je plocha grafu (objekt Axes), jejími potomky jsou pak jednotlivé čáry grafů (objekty Line) a texty (objekty Text)...

Získání handlu objektu

Handle objektu získáme:

1. odebráním výstupu funkce, která vytváří daný objekt, např.:
   ```
   >> handle1 = figure(5); % figure
   >> handle2 = plot(x,f1); % line
   ```
2. využitím funkcí
   `gcf` ... vrátí handle aktuálního okna (Get handle to Current Figure)
   `gca` ... vrátí handle pro aktivní osy ("plocha grafu"; Get handle to Current Axis)
   `gco` ... handle pro aktivní objekt (Get handle to Current Object)
   `allchild` ... všechny děti zadaného objektu, např. `d = allchild(gca); % d=get(gca,'Children')`
   `findall` ... všechny grafické objekty (i skryté)

Zjištění aktuálních hodnot vlastností objektu pomocí jeho handlu

`get(handle)` ... vypíše VŠECHNY vlastnosti objektu s jejich aktuálními hodnotami
`get(handle,'vlastnost')` ... vypíše aktuální hodnotu zadané vlastnosti

Nastavení vlastnosti(í) objektu pomocí jeho handlu

`set(objekt, 'vlastnost', 'hodnota')` ... nastaví novou hodnotu dané vlastnosti
Lze upravovat více vlastností naráz (někdy je to dokonce nutné)
Barvy: každá z barev je definována kombinací 3 základních barev (červená - zelená - modrá) v rozsahu [0;1]

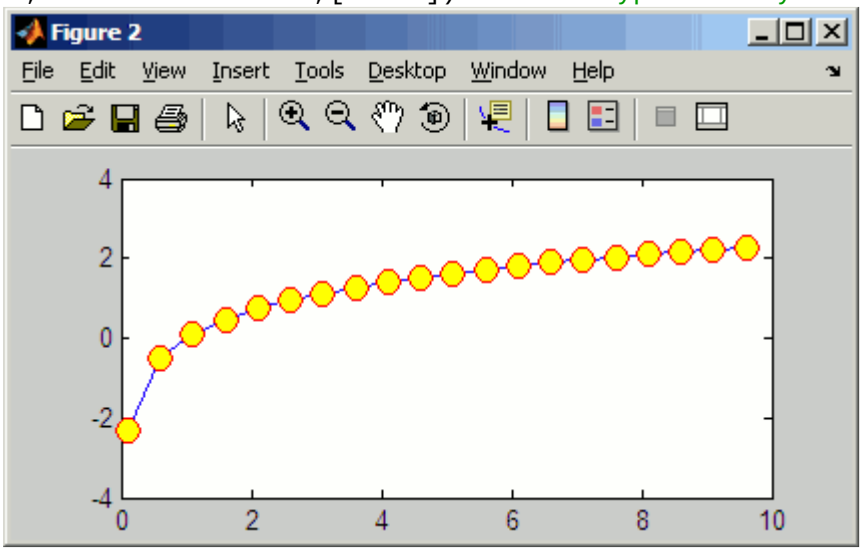
Příklad - nastavení pozice okna Figure:

a. v režimu Pixels:
```
set(gcf,'Position',[1 50 400 350]) % okno 400x350 pixelů, vlevo dole = [1;50]
```
Pozn.: `>> get(0,'ScreenSize')` vrátí vektor popisující obrazovku (objekt root): [levý dolní šířka výška]

b. v režimu Normal(ized):
```
set(gcf,'Units','Normalized','Position',[x1, y1, x2, y2])
```
kde `x1`, `y1` je pozice levého dolního rohu grafického okna vzhledem k obrazovce ([0;0] je levý dolní roh obrazovky) a `x2`, `y2` šířka a výška grafického okna. Vše z intervalu [0;1]
```
>> set(gcf,'Units','Normalized','Position',[0 .05 .3 .3]
% přibližně totéž co v bodu a. (při rozlišení 1280x1024)
```
Pozn.: `>> set(0,'Units','normalized') ,get(0,'ScreenSize')` vrátí vektor [0 0 1 1]

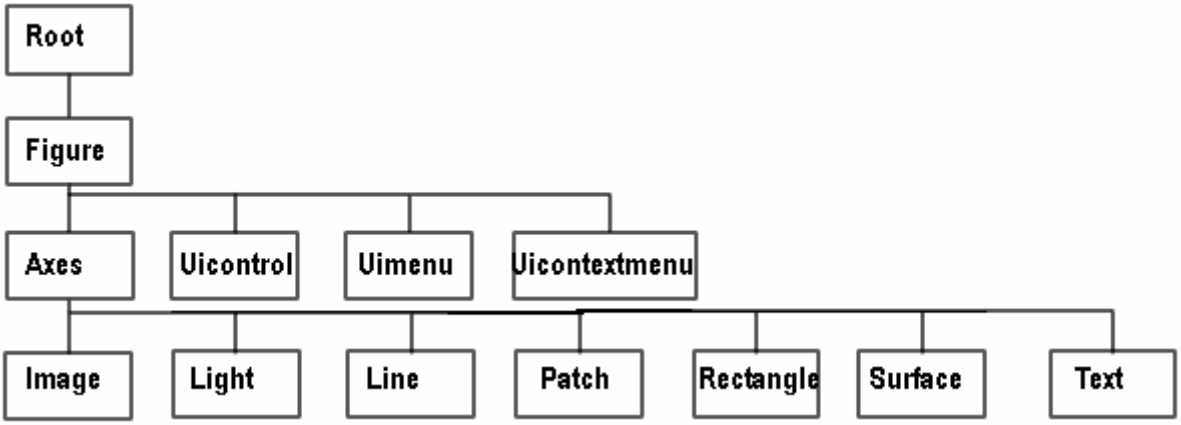
c. v jiných režimech - viz nápověda

Příklad - nastavení značek grafu:
```
>> figure % dalsi nove okno
>> cara = plot([0.1:0.5:10],log([0.1:0.5:10])); % vykresleni grafu, s odebranim handle cary grafu
>> get(cara) % vypis vlastnosti cary grafu
>> set(cara,'Marker','o','MarkerSize',10) % znacka = kolecko, velikost znacky 10
>> set(cara,'MarkerEdgeColor',[1 0 0]) % cervena hrana znacky
>> set(cara,'MarkerFaceColor',[1 1 0]) % zluta vypln znacky
```
Výsledek:



Přehled grafických objektů

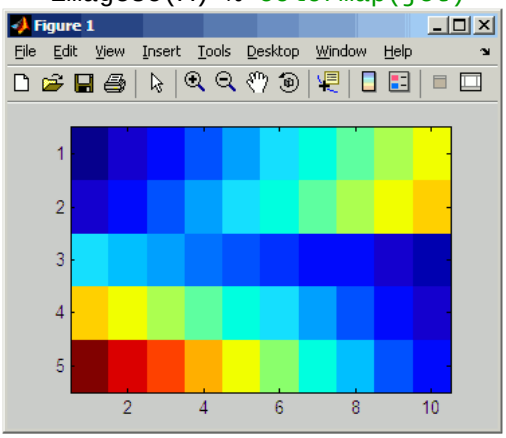
- **Root** - nejvyšší objekt v hierarchii. Odpovídá obrazovce počítače. Je jediný a všechny ostatní graf. objekty jsou mu podřízeny. Existuje po dobu spuštění MATLABu, většinou se neprovádí změna jeho vlastností.
- **Figure** - každé grafické okno. Jejich počet není MATLABem omezen (dokud stačí kapacita počítače). Vytváří se funkcí `figure` nebo (když žádné Figure není otevřeno) funkcemi, které kreslí grafy, (např. `plot` a `surf`). Je-li otevřeno více oken, potom právě jedno je označeno jako aktuální ("current figure") a je cílem grafických výstupů.
- **Axes** - "osy" = plocha, kam se budou vykreslovat všechny podřízené objekty - např. line, text, image. Axes jsou podřízeny objektu figure. Všechny funkce, které kreslí graf (např. `plot`, `surf`, `mesh`, `bar`) vytvářejí objekt axes, pokud zatím neexistuje. Je-li v jednom figure více objektů typu axes (např. při použití `subplot`, pak jeden objekt axes je vždy označen jako aktuálni ("current axes") a je cílem zobrazování (uicontrol a uimenu nejsou objekty podřízené objektu axes).



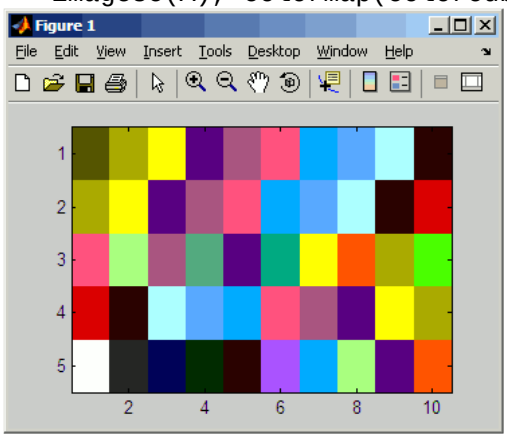
Obr. 2 - hierarchie grafických objektů

- **Line** - základní grafické objekty používané pro kreslení 2D a některých 3D grafů (funkce `line`, `plot`, `plot3`, `loglog` a další). Systém souřadnic použitý u nadřazeného objektu axes rozhoduje o poloze a orientaci kreslených čar.
- **Text** - textové řetězce vytvořené pomocí funkcí `title`, `xlabel`, `ylabel`, `zlabel`, `text` a `gtext` (high-level functions). Jsou to potomci objektu axes.
- **Image** - obrázek. Objekt image se v MATLABu skládá z dat obrázku (= nejčastěji 2D matice) a barevné palety (colormap; je volitelná). Jsou tři základní typy objektu image (podle interpretace dat v matici coby pixelů): indexed, intensity, truecolor.
  Příklad:

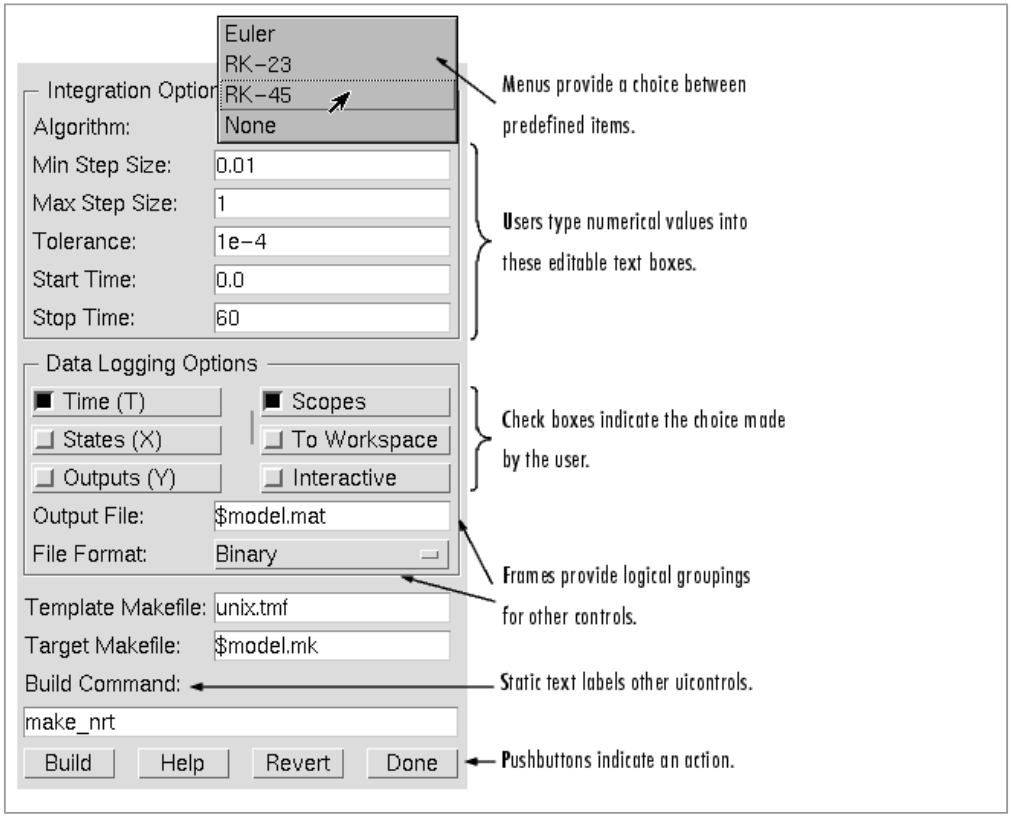
```
>> A = [0:2:18; 2:2:20; 10:-1:1; 20:-2:2; 30:-3:3];
>> imagesc(A) % colormap(jet)
```



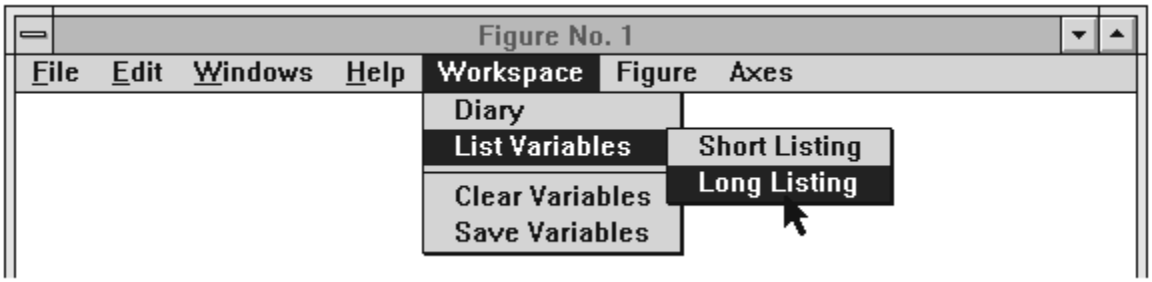

```
>> imagesc(A), colormap(colorcube)
```



- **Surface** - základní grafické objekty používané pro kreslení grafů funkcí 2 proměnných (plochy, 3D grafy) nad rovinou *xy*. Skládají se ze "sítě" čtyřúhelníkových plošek, jejichž vrcholy jsou dány vykreslovanou maticí. Plošky mohou být vyplněny barvou (funkce `surf`) nebo prázdné (funkce `mesh`). Poloha a orientace 3D grafu závisí na systému souřadnic rodičovského objektu axes.
- **Light** - objekt light definuje světelné zdroje pro plochy uvnitř axes. Objekty light nejsou vidět, ale lze měnit jejich vlastnosti (styl, barvu pozici atd.)
- **Patch** - vyplněné polygony s hranami; vytvářené pomocí funkcí `fill`, `fill3` a `contour3`. Rodičovský objekt axes rozhoduje svým systémem souřadnic o pozici objektů patch.
  Vyzkoušejte osmiúhelník:
  `>> t = (1/16:1/8:1)'*2*pi; x = sin(t); y = cos(t);` **`fill(x,y,'r')`**`, axis square`
- **Rectangle** - 2D objekty typu obdélník (i se zaoblenými hranami). Jsou využívány např. pro kreslení sloupcových grafů (obecně pro flow-chart type drawings). Rodičem je objekt axes.
  Vyzkoušejte elipsu:
  `>> rectangle('Position',[1,2,5,10],'FaceColor','g'), axis equal`
  `>> rectangle('Position',[1,2,5,10],'Curvature',[1,1],'FaceColor','g'), axis equal`
- **Uicontrol** - objekty uživatelského rozhraní, které spustí danou činnost v momentě jejich aktivace. Je mnoho typů ovládacích prvků (pushbuttons, listboxes, sliders,...). Každý typ prvku akceptuje určité informace od uživatele (např. listboxes jsou používány pro zobrazení souborů apod.). Uicontrols jsou podřízeny objektu figure, a tedy jsou nezávislé na axes. Uicontrols lze vzájemně kombinovat, a vytvářet tak např. dialogové boxy (viz obr. 3).
- **Uimenu** - jsou to objekty typu vysunovací menu (pull-down menus), které aktivují příslušnou akci podle uživatelova výběru. MATLAB umisťuje tyto objekty do hlavního menu okna (figure window menu bar), a to napravo (za systémová menu). Ani tyto objekty nezávisí na axes (jsou podřízeny objektu figure). Obr. 4 ukazuje tři hlavní menu (top-level menu) s názvem Workspace, Figure a Axes.



Obr. 3 - ukázka použití uicontrol (MATLAB 6.5)
(pop-up menus, editable text boxes, check boxes, pushbuttons, static text, frames)



Obr. 4 - vysunovací menu s aktivovaným Workspace (MATLAB 6.5)

Nízkoúrovňové (low-level) funkce pro vytváření grafických objektů

Každý grafický objekt (kromě objektu roots) má příslušnou funkci, která jej vytváří. Tyto funkce jsou volány grafickými funkcemi vyšší úrovně (jako např. `plot`) automaticky:

Funkce	**Objekt**
`axes`	obdélníkový systém souřadnic; rodič pro objekty line, text, image, light, patch a surface
`figure`	grafické okno Figure
`image`	2D obrázek definovaný maticí hodnot
`light`	zdroj směrovaného osvětlení uvnitř axes - ovlivňuje objekty patch a surface
`line`	čára vytvořená spojením zadaných souřadnicových bodů
`patch`	polygon vytvořený z každého sloupce zadané matice souřadnic
`rectangle`	vyplněný obdélník (až elipsa - když je zaoblení rohů nastaveno na 1)
`surface`	3D graf (síť) nad rovinou *xy*
`text`	řetězec umístěný do souřadnicového systému
`uicontextmenu`	kontextové menu pro nějaký grafický objekt
`uicontrol`	user-interface
`uimenu`	menu (v horní části okna Figure)

Každá funkce vytvářející objekt má tento formát:

```
handle = function('propertyname',propertyvalue,...)
```

Další typy grafů

Kromě X,Y-bodového grafu je někdy potřeba (např. ve statistice) zobrazit jiné typy grafů:

- **`bar`** ... sloupcový graf (vertikální sloupce) - viz příklad na konci textu
- **`bar3`** ... 3D sloupcový graf (vertikální sloupce)
- **`barh`** ... pruhový graf (horizontální sloupce)
- **`bar3h`** ... 3D pruhový graf (horizontální sloupce)

- `area` ... skládaný plošný graf
- `pie` ... koláčový graf (s možností rozložení)
- `pie3` ... 3D koláčový graf (s možností rozložení)

Diskrétní grafy:

- `stem` ... "kolečka se stopkou"
- `stem3` ... "kolečka se stopkou" ve 3D
- `stairs` ... "schody"

Histogramy:

- `hist` ... kartézské souřadnice
- `rose` ... polární souřadnice

Další naleznete v nápovědě MATLABu pod heslem Graphics --- Creating Specialized Plots.

Příklad: ukázka vlivu parametrů na funkci `bar`

```
Y = round(rand(5,3)*10)
subplot(2,2,1), bar(Y,'group'), title('Group')
subplot(2,2,2), bar(Y,'stack'), title('Stack')
subplot(2,2,3), barh(Y,'stack'), title('Stack')
subplot(2,2,4), bar(Y,1.5), title('Width = 1.5')
```