Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta – Katedra fyziky
Jeronýmova 10, 371 15 České Budějovice

Příprava výukového materiálu pro program MATLAB

Bakalářská práce

Autor: Jakub Šimek

Vedoucí práce: RNDr. Petr Bartoš Ph.D.

České Budějovice 2009

Anotace:

Cílem práce bylo vytvořit výukový materiál pro studenty předmětu MATLAB. Práce je koncipována jako učební text, snažící se na konkrétních příkladech vysvětlit základy práce v programu a uvést čtenáře do problematiky zpracování a vizualizace dat v MATLABu. Jednotlivé kapitoly vždy obsahují několik příkladů, aby čtenář co nejlépe pronikl do dané problematiky a naučil se co nejefektivněji využívat systém MATLAB, jeden z nejkomplexnějších a nejrozšířenějších programů pro matematické výpočty.

Abstract:

The aim of this work is set up educational material for students of subject MATLAB. This work is conceive as teaching text, trying to explain bases of work with this programme on certain examples and introduce the reader into questions of data processing and visualization in MATLAB. Individual chapters contains several examples. This is the good way to get the reader in the problems and teach him effectively how utilize MATLAB, one of the most complicated and most spread programmes for methematical calculations.

Prohlášení:

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 25.4.2009 Jakub Šimek

Poděkování:

Tímto bych chtěl poděkovat všem, kteří mě podporovali při tvorbě této bakalářské práce, zejména pak mému školiteli RNDr. Petru Bartošovi Ph.D. za odborné vedení, cenné připomínky a pomoc.

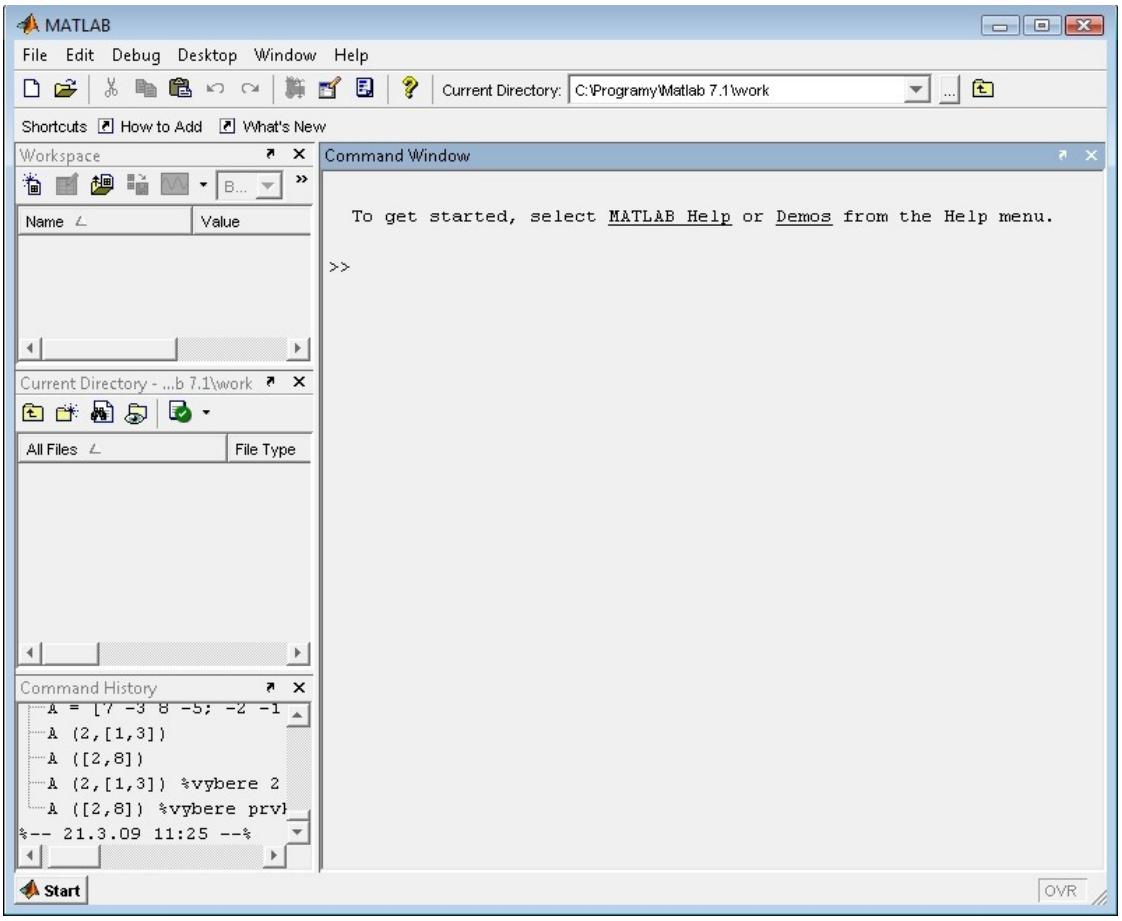
OBSAH

1 Úvod k programu MATLAB

MATLAB je komplexní program určený především pro technické výpočty. Je využíván v mnoha odvětvích a vědních oborech. Kromě základního prostředí, kterým se budeme zabývat v této kapitole, disponuje MATLAB ještě tzv. toolboxy, což jsou rozšiřující balíky funkcí, které umožňují jeho využití ve specializovaných oblastech a zajišťují tak jeho všestranné použití.

1.1 Seznámení s grafickým rozhraním programu

Samotné prostředí MATLABu (MATLAB desktop) lze rozdělit na několik části, z nichž každé má svoji specifickou funkci. Stručně si je popišme:



Obr. 1.1 Prostředí programu MATLAB

Command Window – Příkazové okno pro přímou komunikaci s programem. Příkazy napsané v tomto okně jsou po stisknutí klávesy ENTER ihned provedeny.

Current Directory – Zobrazí obsah aktuálního adresáře, dovoluje základní operace se soubory, otevření (*Open*), spuštění (*Run*), prohlížení (*Open As Text*), načtení dat (*Import Data*).Přístup je k němu přes kliknutí pravým tlačítkem myši na vybraném souboru.

Workspace – Seznam všech použitých proměnných načtených v paměti počítače. Je možné je přímo v tomto okně editovat. Přístup opět přes pravé tlačítko myši.

Command History – Historie použitých příkazů. Je možné se v nich pohybovat klávesami ↑↓. Pokud tedy některé příkazy používáme opětovně, není nutné je znovu zapisovat, pouze stiskneme příslušnou šipku.

1.2 Nastavení pracovního adresáře

Pokud budeme ke své práci využívat soubory, skripty a funkce, je vhodné si nastavit pracovní adresář, do kterého se budou námi vytvořená data ukládat.

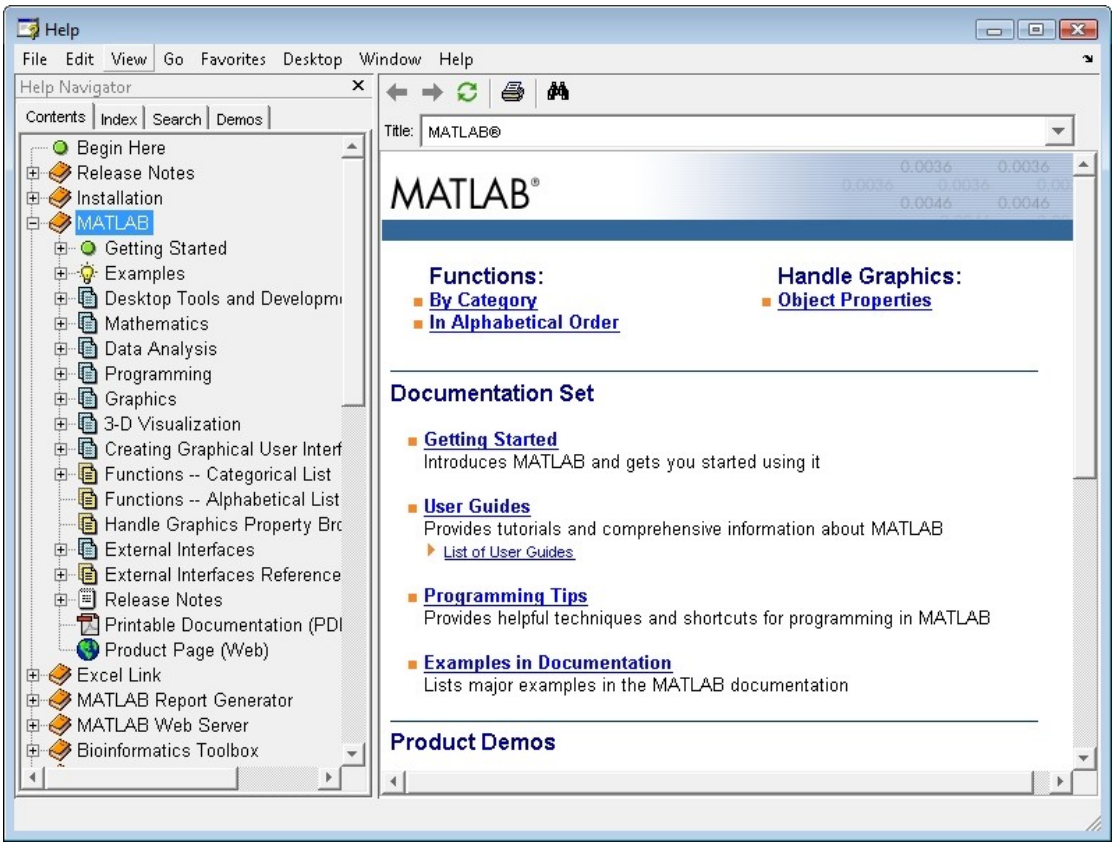
Obr. 1.2 Pracovní adresář

Pokud adresář nenastavíme, budou data ukládána do adresáře work , který je ve složce, v níž máme program MATLAB nainstalovaný.

1.3 Nápověda [Matlab help]

Vyvoláme ji stiskem klávesy F1 nebo voláním položek Help - MATLAB Help. Samotná nápověda je velmi obsáhla a obsahuje několik částí:

- začínáme [Getting Started]
- příklady [Examples]
- popis prostředí [Desktop Tools and Development Environment]
- matematika [Mathematics]
- analýza dat [Data Analysis]
- programování [Programming]
- grafika [Graphics]
- 3D – vizualizace [3-D Visualization]
- programování grafického prostředí [Creating Graphical User Interfaces]



Obr. 1.3 Nápověda programu MATLAB

K nápovědě je možné přistupovat i přes okno **Command Window**. Tento způsob zápisu je velmi užitečný zejména pokud chceme znát syntaxi zápisu konkrétní funkce. Stačí pouze v okně **Command window** zadat příkaz `help` a za něj jméno funkce.

```
>> help abs
 ABS    Absolute value.
    ABS(X) is the absolute value of the elements of X. When
    X is complex, ABS(X) is the complex modulus (magnitude) of
    the elements of X.

    See also sign, angle, unwrap, hypot.

    Overloaded functions or methods (ones with the same name in other directories)
       help frd/abs.m
       help iddata/abs.m
       help sym/abs.m

    Reference page in Help browser
       doc abs
```

Obr. 1.4 Nápověda programu MATLAB volaná z Command window

2 Základy práce s programem MATLAB

Program MATLAB je založen na maticovém počtu. Platí tedy jednoduchá filozofie, že vše je matice. Napíšeme li například do **Command Window** výraz `a=5,` bude toto číslo uložené jako matice o rozměru 1x1.

2.1 Základní operace s čísly

Stejně jako kalkulačka umožňuje MATLAB provádět základní matematické operace (viz. tabulka 2.1). Příkazy zapisujeme do okna **Command Window** jsou prováděny ihned po stisku klávesy ENTER. Pokud uživatel neurčí jinak je výsledek provedené operace ukládán do proměnné `ans`, která je implicitní proměnnou.

operátor	operace	operátor	operace
+	součet	/	podíl
-	rozdíl	^	umocnění
*	součin	()	použití závorek

Tab. 2.1 Základní matematické operace

MATLAB je zejména nástroj pro provádění matematických výpočtů. Proto jsou zde definovány základní elementární funkce:

funkce	**zápis**	**popis**	**funkce**	**zápis**	**popis**
sin	sin(pi)	sinus	**fix**	fix(2.6)	zaokrouhlení dolů
sinh	sinh(0.34)	hyperbolický sinus	**rem**	rem(10,4)	zbytek po dělení
asin	asin(1)	inversní sinus	**sign**	sign(-100)	znaménková funkce
asinh	asinh(pi)	inversní hyperbol. sin	**round**	round(2.5)	zaokr.k nejbližší. číslu
cos	cos(2*pi)	cosinus	**exp**	exp(2)	exponenciální funkce
cosh	cosh(0.1)	hyperbolický cosinus	**log**	log(10)	přirozený logaritmus
acos	acos(0.11)	inversní cosinus	**log10**	log10(100)	dekadický logaritmus
acosh	acosh(1)	inversní hyperbol cos	**sqrt**	sqrt(25)	druhá odmocnina
tan	tan(pi/2)	tangens	**pow2**	pow2(8)	mocnina se základem 2

Tab. 2.2 Přehled základních elementárních funkcí

Zde je několik příkladů zápisu příkazu:

```
>> 9+7*5-32/4 %zakladni matematické operace
ans =
    36
>> (3^2+1)/(27^(1/3)-1) %mocniny, odmocniny, zavork
ans =
     5
>> log10(5)% dekadicky logaritmus
ans =
    0.6990
```

Obr. 2.1 Příklad zápisu výrazů

```
>> (exp(0.1)/sqrt(25))*log2(3.5)
ans =
    0.3995
>> (sin(pi/3)+pow2(4))^2-tan(pi/8)*180
ans =
  209.9044
```

Obr. 2.2 Příklad zápisu výrazů

2.2 Využití proměnných

Někdy je nutné zadávané výrazy uchovat např. pro další výpočty. K tomu slouží tzv. proměnné. Proměnná je identifikátor námi zadaného výrazu. Název proměnné může obsahovat až 31 znaků, přičemž jsou povoleny pouze tyto znaky: písmena anglické abecedy (a-z, A-Z), číslice (0-9) a podtržítko (_).

```
>> U=12; %V napeti
>> I=3; %A proud
>> R=U/I %Výpočt odporu
R =
     4
```

Obr. 2.3 Využití proměnných

Vidíme, že čísla již nejsou ukládána do proměnné *ans*, nýbrž do proměnné, kterou si uživatel sám zvolil.

2.3 Příkaz who a whos

Není nutné si pamatovat všechny použité proměnné. Ke zjištění všech použitých proměnných slouží příkaz *who*, nebo *whos*. Použité proměnné je též možné sledovat v okně **Workspace**.

```
>> who

Your variables are:

A      B      Jmeno  ans    f

>> whos
  Name         Size                    Bytes  Class

  A            3x3                        72  double array
  B            2x2                        32  double array
  Jmeno        1x5                        10  char array
  ans          1x1                         8  double array
  f            1x1                         8  double array

Grand total is 20 elements using 130 bytes
```

Obr. 2.4 Příkazy who a whos

Příkaz *whos* kromě výpisu použitých proměnných uvede ještě jejich maticový rozměr, počet bytů na kterých je proměnná uložená a způsob uložení.

2.4 Příkaz clear

Pokud již nechceme některou proměnnou používat, je vhodné ji z důvodu přehlednosti a úspory paměti vymazat. K tomu slouží příkaz *clear*. Jeho syntaxe je *clear_promenna*. Např. pokud bychom chtěli vymazat proměnnou Jméno z předchozí části, zápis by zněl: *clear Jmeno*. V případě, že budeme chtít vymazat všechny proměnné, můžeme použít příkaz *clear all*.

2.5 Využití středníku

U velké části programů není nutné, aby se všechny definované proměnné vypisovaly. Tuto funkci zastává středník, který potlačí výpis.

```
>> f = sin(pi/2);
>>
```

Obr. 2.5 Potlačení výpisu středníkem

2.6 Předdefinované konstanty

Můžeme je chápat jako předdefinované proměnné. Pro výpočty můžeme využit následující konstanty:

```
>> i %imaginarni jednotka komplexniho cisla
ans =
        0 + 1.0000i
>> j %imaginarni jednotka komplexniho cisla
ans =
        0 + 1.0000i
>> pi %Ludolfovo cislo
ans =
    3.1416
>> eps % nejmensi rozlisitelna hodnota pri double presnosti
ans =
  2.2204e-016
>> realmax %nejvetsi desetinne cislo
ans =
  1.7977e+308
>> realmin %nejmensi desetinne cislo
ans =
  2.2251e-308
```

Obr. 2.6 Konstanty

2.7 Formát zobrazených čísel

MATLAB implicitně zobrazuje čísla nebo výrazy na čtyři desetinná místa. Jde o zobrazení v pevné řadové čárce (fix point). Samotné výpočty jsou prováděny s plnou přesností (double). Seznam dostupných možností formátování získáme *help format*. Implicitně je nastaven formát short.

format short	5 platných míst, 4 desetinná místa, pevná desetinná čárka
format long	15 platných míst, 14 desetinná místa, pevná desetinná čárka
format short e	5 platných míst, 4 desetinná místa, plovoucí desetinná čárka
format long e	15 platných míst, 14 desetinná místa, plovoucí desetinná čárka

```
>> format short
>> pi
ans =
    3.1416
>> format short e
>> pi
ans =
  3.1416e+000
>> format long
>> pi
ans =
   3.14159265358979
>> format long e
>> pi
ans =
    3.141592653589793e+000
```

Obr. 2.7 Formát zobrazených čísel

2.8 Operace s komplexními čísly

Proměnné v MATLABu nemusí být pouze reálné, ale mohou být také komplexní. Komplexní čísla je možné zadávat ve složkovém i exponenciálním tvaru, přičemž imaginární jednotku zapisujeme jako **i** nebo **j**.

```
>> c1=2+3i %zadani ve slozkovém tvaru - imaginarni jednotka i
c1 =
   2.0000 + 3.0000i
>> c2=5-3j %zadani ve slozkovem tvaru - imaginarni jednotka j
c2 =
   5.0000 - 3.0000i
>> c3=10*exp(pi/2*j) % zadani v exponencialnim tvaru
c3 =
   0.0000 +10.0000i
>> c4=6+i*sin(pi/2) % zadani ve slozkovem tvaru
c4 =
   6.0000 + 1.0000i
```

Obr. 2.8 Zápis komplexních čísel

Aritmetické operace pro komplexní čísla lze používat stejně jako pro čísla reálná.

```
>> sqrt(-2) %odmocnina za zaporneho cisla
ans =
        0 + 1.4142i
>> c1*c2 %soucin komplexnich cisel
ans =
  19.0000 + 9.0000i
>> c1+c2 %soucet komplexnich cisel
ans =
     7
```

Obr. 2.9 Operace s komplexními čísly

MATLAB obsahuje ještě několik dalších funkcí pro práci s komplexními čísly. Jejich použití si ukážeme na následujícím příkladě.

```
>> c=3+4i;
>> real (c) %realna cast komplexniho cisla
ans =
     3
>> imag (c) %imaginarni cast komplexniho cisla
ans =
     4
>> abs (c) %velikost komplexniho cisla
ans =
     5
>> angle (c) %uhel v radianech v Gausove rovine
ans =
    0.9273
>> conj (c) %komplexni doplnek
ans =
   3.0000 - 4.0000i
```

Obr. 2.10 Operace s komplexními čísly

2.9 Řetězcové proměnné

Řetězce se zapisují jako řada znaků ASCII uzavřených do apostrofů. Je vhodné používat pouze znaky anglické abecedy, protože při použití háčků a čárek bývá problém s jejich zobrazením.

```
>> text='retezec napsany v MATLABU'
text =
retezec napsany v MATLABU
```

Obr. 2.11 Zápis řetězce

Každý znak je v ASCII tabulce reprezentován číselnou hodnotou. Pro zjištění této hodnoty se používá funkce *double*. Pro znak pak funkce *char*.

```
>> double ('X') %Hodnota odpovidajici ASCI znaku
ans =
    88
>> char (89) %Znak opovidajici ASCI hodnote
ans =
Y
```

Obr. 2.12 Převod v ASCII tabulce

Pokud chceme pouze určit, jestli je daný objekt řetězec, nebo numerická hodnota, využijeme funkci *class*.

```
>> class (text) % zjisteni tridy objektu
ans =
char
```

Obr. 2.13 Určení třídy objektu

Pro práci s řetězci existuje celá řada funkcí. Jejich přehled bychom získali voláním nápovědy *help_strfun*. Zde uveďme pouze několik nejčastějších příkladů:

```
>> length (text) %zjisteni delky retezce
ans =
    25
>> findstr(text,'a') %pozice znaku v retezci
ans =
    10    13
>> text (end:-1:1) %vypsani retezce v opacnem poradi
ans =
UBALTAM v ýnaspan cezěteř
>> text=strrep(text,'MATLABU','matlabu') %nahrazeni casti rete
text =
řetězec napsaný v matlabu
```

Obr. 2.14 Manipulace s řetězci

Mezi číselnou reprezentací a řetězcem existuje vzájemná relace. Číslo, které je zapsáno jako řetězec, lze převést na číslo a obráceně:

```
>> text = '1.602e-19';
>> cislo = str2num (text) % prevod cisla na retezec
cislo =
  1.6020e-019
>> cislo = '238.45e-2';
>> cislo = str2num (text) % prevod retezce na cislo
cislo =
  1.6020e-019
```

Obr. 2.15 Převody mezi datovými typy

Řetězce lze zapisovat také do matic, ale je nutné, aby každý prvek matice měl stejnou délku. Prvky se doplňují mezerami. Pokud nechceme hlídat délku jednotlivých řetězců, je možné použít funkci *char* nebo *strvcat*:

```
>> % Pole retezcu - jednotlive prvky musi mit stejnou delku
>> nakup = ['mouka  ';'maslo  ';'rohliky']
nakup =
mouka
maslo
rohliky
>> % Pole retezcu - jednotlive prvky nemusi mit stejnou delku
>> nakup = char ('mouka  ','maslo  ','rohliky')
nakup =
mouka
maslo
rohliky
```

Obr. 2.16 Zápis řetězců do matic

3 Vektory a matice

Matice jsou základním stavebním kamenem celého programu, proto je jejich zvládnutí pro práci s programem nutností.

3.1 Zadávání matic a vektorů

Matice můžeme zadávat několika způsoby. Jejich velikost je omezena pouze operační pamětí konkrétního počítače. Způsoby zadávání jsou následující:

- Zadání matice po prvcích
- vygenerování příkazem nebo funkcí
- načtení z externího souboru nebo aplikace

Matice se zapisují do hranatých závorek, kdy jednotlivé prvky na řádku jsou odděleny čárkou nebo mezerou a jednotlivé řádky středníkem.

```
>> A =[1 -2 4 ; 2 1 -5] %matice rozmeru 2x3
A =
     1    -2     4
     2     1    -5
```

Obr. 3.1 Zápis matice

Je li matice jen jeden řádek nebo sloupec, nazýváme ji vektorem a jedná-li se pouze o jedno číslo skalárem (viz. Základy práce s programem MATLAB).

```
>> v = [sin(0) sin(pi/2) sin(pi) sin(3*pi/2) sin(2*pi)]
v =
         0    1.0000    0.0000   -1.0000   -0.0000
```

Obr. 3.2 Zápis vektoru

3.2 Funkce pro vytváření matic

Některé speciální druhy matic je možné vytvořit, aniž bychom tyto matice museli pracně zapisovat. Pro jejich tvorbu můžeme použít následující funkce:

```
>> eye (2) %jednotkova matice o rozmeru 2x2
ans =
     1     0
     0     1
>> zeros (2,3) %matice nul o rozmeru 2x3
ans =
     0     0     0
     0     0     0
>> ones (3,2) %matice jednicek o rozmeru 3x2
ans =
     1     1
     1     1
     1     1
>> rand (2) %matice nahodnych cisel z intervalu <0,1>
ans =
    0.4447    0.7919
    0.6154    0.9218
```

Obr. 3.3 Funkce pro vytváření matic

```
>> magic (3) %ctvercova matice - stejne soucty radku a sloupce
ans =
     8     1     6
     3     5     7
     4     9     2
```

Obr. 3.4 Funkce pro vytváření matic

3.3 Operátor dvojtečka

Jeden z nejdůležitějších operátorů. Slouží například k vytváření posloupností. Jeho syntaxe je například: *Jmeno_promenne = pocet:krok:konec*. Tento zápis vytvoří vektor, který bude začínat hodnotou *pocet*, každá další souřadnice vektoru bude o hodnotu *krok* větší než ta předchozí a poslední souřadnice vektoru bude mít hodnotu *konec*.

```
>> v1 = -3:3 %vektor od -3 do 3 s krokem 1
v1 =
    -3    -2    -1     0     1     2     3
>> v1 = 1:2:11 %vektor od 1 do 11 s krokem 2
v1 =
     1     3     5     7     9    11
```

Obr. 3.5 Vytvoření vektoru

3.4 Funkce linspace

Používá se, pokud chceme vytvořit řadu s neznámým krokem, ale známým počtem prvků: Syntaxe je obdobná jako u „:“. *Jmeno_promenne = linspace(pocatek, konec, pocet)*. Vytvoří se vektor od *pocatek* do *konec* s počtem prvků *krok*.

```
>> fi = linspace (0, 2*pi, 5)*180/pi
fi =
     0    90   180   270   360
```

Obr. 3.6 Vytvoření vektoru pomocí linspace

Pokud chceme vytvořit logaritmickou řadu je nám k dispozici funkce *logspace*, která má stejnou funkci jako *linspace*, pouze prvky uspořádá logaritmicky.

3.5 Maticové operace

Matematické operace, které můžeme v MATLABU využít se dělí na dvě skupiny:

- operace podle pravidel lineární algebry
- operace nad jednotlivými prvky (skalární počet)

aritmetické operace	
+,-	sčítání a odčítání – matice musí mít stejný rozměr
*	násobení – Počet řádků první matice musí být stejný jako počet sloupců druhé matice
\	dělení zleva
^	umocnění
’	transpozice – prohození řádků a sloupců matice

Tab. 3.1 Aritmetické maticové operace

Pří využití maticového počtu je nutné dodržovat pravidla lineární algebry, jinak program nahlásí chybu.

prvkové operace	
.*	násobení se prvky na odpovídajících pozicích
.^	prvkové umocnění
./	prvkové dělení zprava
.\	prvkové dělení zleva

Tab. 3.2 Prvkové maticové operace

Příklady použití maticových a prvkových operací:

```
>> A = [-2 3 1; 2 2 5]
A =
    -2     3     1
     2     2     5
>> B = [-1 4; -1 6; -3 1]
B =
    -1     4
    -1     6
    -3     1
>> A*B %Nasobeni - matice A stejny pocet radku jako B sloupcu
ans =
    -4    11
   -19    25
>> A-B'%rozdil - matice musi mit stejny rozmer
ans =
    -1     4     4
    -2    -4     4
>> (A+B')\A %Podil - deleni zleva
ans =
    0.5556   -0.5556    0.0556
   -0.1667    0.6667    0.5833
         0         0         0
>> A.^2 % umocneni matice po prvcich
ans =
     4     9     1
     4     4    25
>> A.*B' %Nasobeni matice po prvcich
ans =
     2    -3    -3
     8    12     5
```

Obr. 3.7 Operace s maticemi

3.6 Výběr prvků a tvorba submatic

Jednotlivé prvky matice jsou v MATLABU indexovány a proto je pak možné manipulovat s jednotlivými prvky, řádky či sloupci. Možností, jak přistupovat k určité části matice, je opět několik:

- pomocí řádkových a sloupcových indexů
- pomocí řádkového indexů
- pomocí logického výrazu

- pomocí seznamu prvků v hranatých závorkách uvnitř kulatých závorek

Pro ukázku přístupu si vytvoříme matici **A** o rozměru 4x3

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

Obr. 3.8 Vytvoření matice

Následující část kódu ukazuje, jak je možno přistupovat k určitému prvku matice. Zde jsme vybrali prvek na druhém řádku a třetím sloupci. Poté jsme provedli jeho nahrazení hodnotou -4.

```
>> A (2,3) %vypise prvek na 2 radku a 3 sloupci
ans =
     6
>> A (2,3)=-4 %nahradi prvek na pozici 2,3 cislem -4
A =
     7    -3     8    -5
    -2    -1    -4     3
     4    -4     2    -7
```

Obr. 3.9 Výběr a nahrazení prvku

Přistupovat k jednotlivým prvkům můžeme i jinak než přes řádky a sloupce. Jedná se o přístup pomocí pořadového indexu. Prvky jsou indexovány tak, jako bychom jednotlivé sloupce matice napsali na jeden řádek. Zde jsme například vybrali prvek na třetím řádku a třetím sloupci, čemuž odpovídá pořadový index s číslem 9.

```
>> A(3,3)
ans =
     2
>> A(9) % stejny prvek jako na pozivi 3,3
ans =
     2
```

Obr. 3.10 Výběr prvku pomocí pořadového indexu

Pokud se chystáme přistupovat k celým řádkům či sloupcům matice, používáme operátor „:“.

```
>> A (3,:) %vybere treti radek a vsechny sloupce
ans =
     4    -4     2    -7
>> A (:,2) %vybere vsechny radky a druhy sloupec
ans =
    -3
    -1
    -4
```

Obr. 3.11 Výběr řádků a sloupců

Chceme-li vybrat více řádků současně, můžeme použít podobný zápis jako pří tvorbě posloupnosti. Následující část zdrojového kódu vybere řádky začínající indexem 1 s krokem 2 do řádku 3 a sloupce začínající indexem 2 s krokem 2 do sloupce 4.

```
>> A(1:2:3,2:2:4) %vybere 1 a 3 radek, 2 a 4 sloupec
ans =
    -3    -5
    -4    -7
```

Obr. 3.12 Výběr různých řádků a sloupců

Pokud chceme např. vybrat všechny sloupce mimo prvního, můžeme využít slovo *end*, které označuje poslední prvek.

```
>> A(1:2,2:end) %vybere 1 a 2 radek, 2 az posledni sloupec
ans =
    -3     8    -5
    -1     6     3
```

Obr. 3.13 Výběr řádků a sloupců pomocí end

Prvky lze také vybírat pomocí jejich řádkového a sloupcového indexu nebo pomocí pořadového indexu.

```
>> A (2,[1,3]) %vybere 2 radek 1 a 3 sloupec
ans =
    -2     6
>> A ([2,8]) %vybere prvky s indexy 2 a 8
ans =
    -2     6
```

Obr. 3.14 Výběr více prvků pořadovými indexy

3.7 Manipulace s maticemi

V předchozí části jsme se zabývali možnostmi výběru jednotlivých prvků, řádků či sloupců nebo několika řádků a sloupců. Nyní si ukážeme, jak jednotlivé řádky mazat, prohazovat a vytvářet matice z již existujících matic. Mazání se provádí přiřazením prázdného vektoru. Následující příkazy vymažou nejprve první řádek matice a poté všechny liché sloupce.

```
>> A = [1 5 -2 4 9; 2 -4 -6 1 4; 10 -7 -2 5 6]
A =
     1     5    -2     4     9
     2    -4    -6     1     4
    10    -7    -2     5     6
>> A(1,:)= [] %prirazeni prazdneho vektoru - smazani 1 radku
A =
     2    -4    -6     1     4
    10    -7    -2     5     6
>> A(:,1:2:end)= [] %smazani lichych sloupcu
A =
    -4     1
    -7     5
```

Obr. 3.15 Mazání řádků a sloupců matice

Kromě výběru můžeme také prohazovat jednotlivé řádky a sloupce:

```
>> A = [1 3 -2; -4 1 -6; 2 -1 3]
A =
     1     3    -2
    -4     1    -6
     2    -1     3
>> A ([2,3],:)= A ([3,2],:) %prohozeni 2 a 3 radku
A =
     1     3    -2
     2    -1     3
    -4     1    -6
```

Obr. 3.16 Prohazování řádků a sloupců matice

Matice je možné skládat. Je však nutné zachovat jejich rozměr, jinak MATLAB vytiskne chybové hlášení.

```
A =
     1     2
     3     1
>> v = -1:2
v =
    -1     0     1     2
>> B = [A A';v] %slozena matice - nutno zachovat romer
B =
     1     2     1     3
     3     1     2     1
    -1     0     1     2
```

Obr. 3.17 Skládání matic a vektorů

3.8 Čtvercové matice

Jedná se o matice, které mají stejný počet řádků jako sloupců a díky tomu mají určité speciální vlastnosti. V MATLABUu existuje početná skupina příkazů určená pro tento druh matic.
Nejprve si ukážeme výpočet determinantu a inverzní matice:

```
>> A = [1 3; 2 4];
>> det(A) %dereminant matice A
ans =
    -2
>> inv(A) %inverzni matice k A
ans =
   -2.0000    1.5000
    1.0000   -0.5000
```

Obr. 3.18 Determinant a inverzní matice

Další možností je výběr horní a dolní trojúhelníková matice:

```
>> A = [3 2 -1; 6 -2 -5; -1 3 5];
>> triu(A) %horni trojuhelnikova matice
ans =
     3     2    -1
     0    -2    -5
     0     0     5
>> tril(A) %dolni trojuhelnikova matice
ans =
     3     0     0
     6    -2     0
    -1     3     5
```

Obr. 3.19 Horní a dolní trojúhelníková matice

Pro určení hodnosti matice můžeme využít příkaz *rank*:

```
>> A = [2 -1 5; -4 7 -1; 3 2 3];
>> rank (A) %hodnost matice
ans =
     3
```

Obr. 3.20. Hodnost matice

Je možné zde nalézt ještě celou řadu dalších funkcí a příkazů pro práci s těmito maticemi. Jejich úplný výčet a způsoby použití nám poskytne nápověda zadáním příkazu *help matfun*.

3.9 Další funkce pro práci s maticemi

```
>> A = [2 6 -1; 4 -2 5; 1 3 1];
>> sum(A) %vektor jehoz vysledky jsou soucty sloupcu
ans =
     7     7     5
>> diag(A) %vektor prvku na hlavni diagonale
ans =
     2
    -2
     1
>> max(A) %vektor nejvetsich prvku jednotlivych sloupcu
ans =
     4     6     5
```

Obr. 3.21 Funkce pro práci s maticemi

4 Práce se soubory

MATLAB umí zpracovávat velké množství formátu od binárních až po textové.

4.1 Ukládání dat

Pro ukládání slouží příkaz *save* Pokud jej použijeme bez dalších parametrů, budou data uloženy do binárního souboru *matlab.mat* v aktuálním adresáři.

```
>> M = [1 -3 2; -1 6 -1];
>> save

Saving to: matlab.mat
```

Obr. 4.1 Uložení dat do souboru matlab.mat

Možnost ukládat data pouze do jednoho souboru by byla značně omezená. Proto si uživatel může přímo určit adresář, do kterého mají byt data uložena.

```
>> M = [1 -3 2; -1 6 -1];
>> save 'matice.mat'
```

Obr. 4.2 Uložení dat do námi zvoleného souboru

Další možností je uložení pouze proměnných:

```
>> v = (1:10).^2;
>> save 'vystup.mat', v
v =
     1     4     9    16    25    36    49    64    81   100
```

Obr. 4.3 Uložení proměnných do souboru

Pokud nechceme data ukládat v binární podobě, ale textové, můžeme využít příkaz *save 'jmeno_souboru'-ascii*:

```
>> v = (1:10).^2;
>> save 'vystup2.mat'-ascii
```

Obr. 4.4 Uložení dat v textové podobě

Kromě tohoto základního textového formátu můžeme například zajistit, aby byla data oddělena tabulátory, či na kolika bytech mají být ukládána.

```
>> %vystup oddeleny tabulatory
>> save 'vystup.mat'-ascii -tabs
>> %16 bitovy vystup
>> save 'vystup.mat'-ascii -double
>> %16 bitovy vystup oddeleny tabulatory
>> save 'vystup2.mat'-ascii -double -tabs
```

Obr. 4.5 Další možnosti ukládání dat do souboru

4.2 Načítání dat

Načítání dat je téměř analogické s jejich ukládám, pro vyzvednutí souboru ovšem používá příkaz *load*:

load	načte všechna data ze souboru *matlab.mat*

load 'jmeno_souboru.mat'	načte všechna data ze souboru *jmeno_souboru.mat*
load 'jmeno_souboru.mat', promenna	načte *promenou* ze souboru *jmeno_souboru.mat*
load 'jmeno_souboru.mat' –ascii	načte data z ASCII souboru *jmeno_souboru.mat*

Chceme-li pouze zjistit druh dat uložený v některém souboru, můžeme využít příkazu `whos`:

```
>> whos -file 'matice.mat'
  Name      Size                     Bytes  Class

  M         2x3                         48  double array

Grand total is 6 elements using 48 bytes
```

Obr. 4.6 Zjištění proměnných v souboru

4.3 Otevření souboru pro zápis a čtení

Pro otevření souboru, se kterým se chystáme pracovat, používáme funkci *fopen*. Celá syntaxe zápisu má pak následující tvar: *fid = fopen ('nazev_souboru','pristup')*.
Slovo fid je tzv. identifikátor a slouží k rozpoznání souboru u dalších funkcí. Parametr přístup pak označuje způsob, jakým budou data ze souboru čtena nebo do něj zapisována.

'r'	otevření souboru pro čtení
'w'	otevření souboru pro zápis, pokud existuje, přepíše se, pokud neexistuje, vytvoří se
'a'	otevření souboru pro přidání, pokud existuje, data se připojí na konec, pokud neexistuje, založí se
'r+'	otevření binárního souboru pro čtení
'w+'	otevření binárního souboru pro zápis, pokud existuje, přepíše se, pokud neexistuje, vytvoří se
'a+'	otevření binárního souboru pro přidání, pokud existuje, data se připojí na konec, pokud neexistuje, založí se

Jako příklad si uvedeme otevření binárního souboru pro zápis:

```
>> fid = fopen ('matice.bin','w+')
fid =
     3
```

Obr. 4.7 Otevření binárního souboru pro zápis

Po ukončení práce se souborem je třeba jej zavřít použitím funkce `fclose`.

```
>> fclose (fid)
ans =
     0
```

Obr. 4.8 Uzavřeni souboru

Pokud pracujeme s několika soubory zároveň, je možné k jejich hromadnému uzavření využít parametr *all*.

4.4 Zápis a čtení u binárních souborů

Pokud se nám úspěšně podaří otevřít a určit způsob operace, která bude se souborem prováděna, můžeme přistoupit k vlastnímu vložení či načtení dat. U binárních souborů k tomu používáme dvě základní funkce: *fwrite* a *fread*
Způsob zápisu a čtení dat si ukážeme na následujícím příkladu:

```
>> v = (2:2:10); %vektor v
>> fid = fopen ('vektor.bin','w+'); %otevreni souboru
>> fwrite(fid,v,'uint8') %ulozeni vektoru do souboru
ans =
     5
>> fclose(fid); %uzavreni souboru
```

Obr. 4.9 Uložení dat do binárního souboru

Vidíme, že funkce *fwrite* má celkem tři parametry: identifikátor souboru, ukládaná dat a formát ukládaných dat. Poslední parametr je nepovinný a může nabývat následujících hodnot:

Znaménkové:

int8	data uložena na 8 bitech	-2^7 až 2^7-1
int16	data uložena na 16 bitech	-2^{15} až $2^{15}-1$
int32	data uložena na 32 bitech	-2^{31} až $2^{31}-1$
int64	data uložena na 64 bitech	-2^{63} až $2^{63}-1$

Neznaménkové:

uint8	data uložena na 8 bitech	0 až 2^8-1
uint16	data uložena na 16 bitech	0 až $2^{16}-1$
uint32	data uložena na 32 bitech	0 až $2^{32}-1$
uint64	data uložena na 64 bitech	0 až $2^{64}-1$

Poslední parametr tedy určuje, na kolika bitech budou data uložena.

Nyní si provedeme načtení hodnot vektoru v:

```
>> fid = fopen ('vektor.bin','r+'); %otevreni soubor
>> a = fread (fid,[1,5],'uint8') %nacteni vektoru
a =
     2     4     6     8    10
>> fclose(fid); %uzavreni souboru
```

Obr. 4.10 Načtení dat z binárního souboru

Vidíme, že funkce *fread* má také tři parametry: identifikátor, počet čtených hodnot, a formát čtených dat.

4.5 Zápis a čtení u textových souborů

Využíváme univerzální funkce *fprintf*, která data převádí na řetězce. Syntaxe funkce fprintf je následující: *fprintf(identifikátor,'format',promene)*.
Identifikátor musí být shodný s parametrem *fid* funkce *fopen*. V parametru formát provádíme vlastní převod pomocí % a způsob zobrazení převáděných dat.

%d	celé číslo v desítkové soustavě
%f	desetinné číslo s plovoucí řadovou čárkou

%e	číslo v exponenciálním tvaru
%g	automatická volba formátu
%s	řetězec znaků
%c	jednotlivé znaky

Spolu s převodními znaky lze do parametru *format* zapsat libovolně dlouhý řetězec. Můžeme také použít znaky sloužící pro formát zobrazení a tisku:

\n	odřádkování
\t	tabulátor
\f	nová stránka

```
>> h = (0:45:360); %vektror uhlu
>> x = cos(h*pi/180); %hodnoty cosinu jednotlivych uhlu
>> tab = [h;x]; % matice uhlu a cosinu
>> fid = fopen('cosinus.txt','w'); %otevreni souboru
>> fprintf (fid,'%6.2f %12.8f\n',tab);
>> fclose(fid); %uzavreni souboru
```

Obr. 4.11 Uložení formátovaného textu

Samotnou funkci je možné používat nejen při ukládání do souborů a i při formátování běžného výstupu. Velké využití má zejména u skriptů a funkcí.

5 Polynomy

MATLAB poskytuje pro práci s polynomy celou řadu funkcí. Jejich kompletní výčet bychom získaly zadáním `help_polyfun`. Polynomy se zadávají jako vektory o n+1 prvcích.

$p(x) = 7x^5 + 2x^4 - 3x^3 + 5x^3 - 12x + 2$

Chceme-li tedy např. zadat polynom pátého stupně, provedeme to pomocí šesti prvků.

```
>> p = [7, 2, -3, 5, 12, 2]
p =
     7     2    -3     5    12     2
```

Obr. 5.1 Vytvoření polynomu

Pokud chceme zjistit hodnotu polynomu pro danou hodnotu x, např. x=1, můžeme využít funkci `polyval`.

```
>> polyval (p,1)
ans =
    25
```

Obr. 5.2 Hodnota polynomu

Budeme-li chtít znát jeho derivaci, lze použít funkci `polyder`.

```
>> polyder(p)
ans =
    35     8    -9    10    12
```

Obr. 5.3 Derivace polynomu

Derivovaný polynom p(x) má tedy tvar: $p(x) = 35x^4 + 8x^3 - 9x^2 + 10x + 12$

Každý polynom má svoje kořeny – ať už reálné nebo komplexní. V MATLABu pro jejich zjištění můžeme využít funkci `roots`.

```
>> roots(p)
ans =
   0.8379 + 0.8704i
   0.8379 - 0.8704i
  -0.8898 + 0.5324i
  -0.8898 - 0.5324i
  -0.1821
```

Obr. 5.4 Kořeny polynomu

Pokud budeme znát kořeny, je vhodné využít funkci `poly`, která je funkcí inverzní k funkci `roots`.

```
>> a = [1, 3, 2] %polynom druheho stupne
a =
     1     3     2
>> p = roots(a) % koreny polynomu
p =
    -2
    -1
>> poly(p) % polynom k danym korenum
ans =
     1     3     2
```

Obr. 5.5 Vytvoření polynomu z kořenů

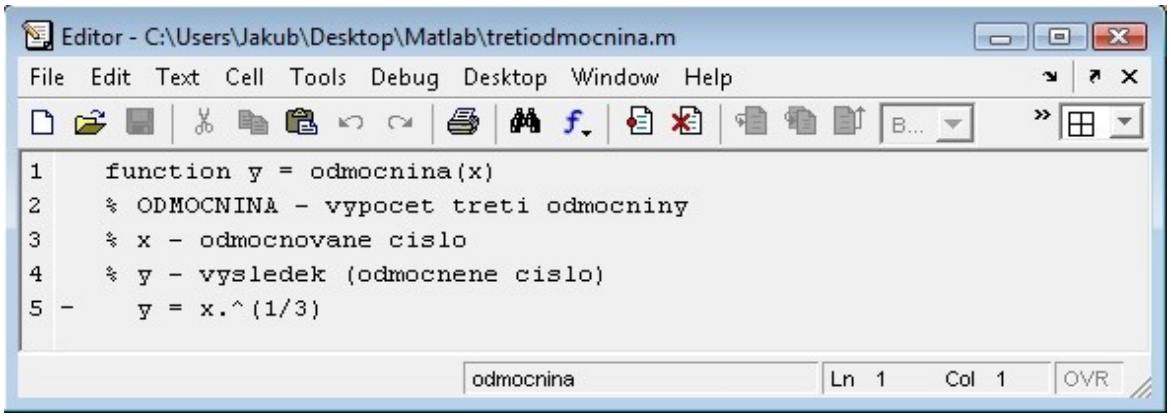
Pro násobení a dělení polynomů můžeme využít funkcí *conv* a *deconv*.

```
>> p1 = [1, -5, 1]; %zadane polynomy
>> p2 = [1, 2, -6];
>> conv (p1,p2) %nasobeni polynomu
ans =
     1    -3   -15    32    -6
>> [a, b] = deconv (p1,p2) %deleni polynomu
a =
     1
b =
     0    -7     7
```

Obr. 5.6 Násobení a dělení polynomů

6 Skripty a funkce

V případě, že řešíme jednoduché úlohy, zcela si vystačíme s oknem **Command Windows**. Může však nastat situace, kdy chceme například počítat s různými daty podle stejného předpisu. Pro tuto možnost MATLAB disponuje textovým editorem, kde je možné příkazy zapsat a poté najednou spustit. Editor otevřeme příkazy *File → New→M-file.*



Obr. 6.1 Funkce pro vypočet třetí odmocniny

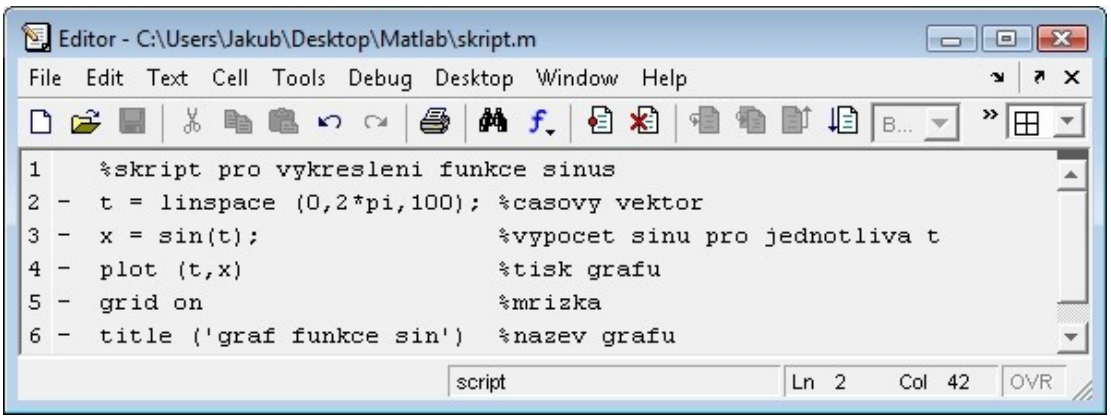
Otevře se prázdné okno editoru, do kterého můžeme zapisovat libovolné příkazy. Zde je to například jednoduchá funkce pro výpočet třetí odmocniny z daného čísla x. Vytvořenou funkci uložíme kliknutím na *File* a položku *Save As..*, zadáme název souboru a uložíme. Při zápisu názvu souboru je nutné dodržet také několik pravidel:

- smí obsahovat pouze znaky anglické abecedy - písmena(a-z, A-Z), číslice (0-9) a podtržítko (_).
- Název nesmí začínat písmenem
- nesmí obsahovat rezervovaná slova

Pokud máme správně nastavenou cestu, můžeme všechny vytvořené skripty a funkce vidět v okně **Current Directory**.

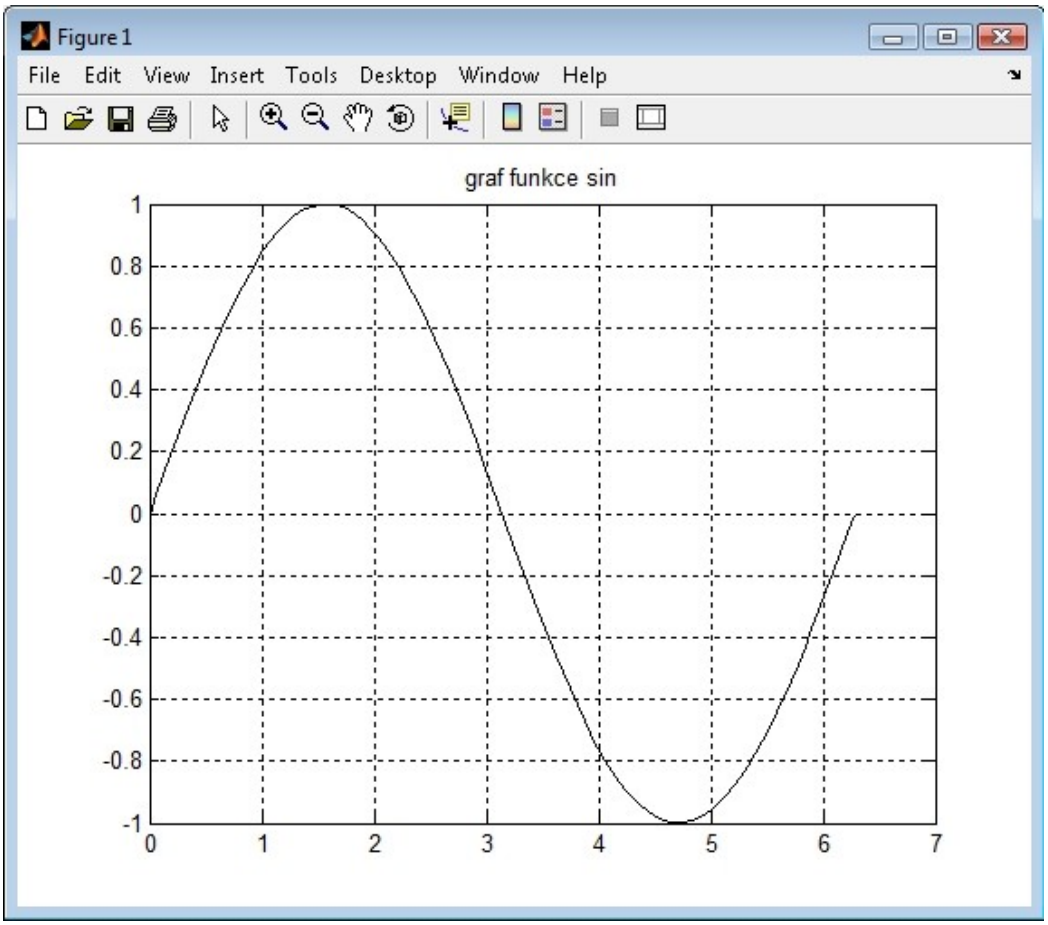
6.1 Skripty

Každý skript pracuje s proměnnými pracovního prostředí, takže může vytvářet nové nebo mazat či měnit vybrané. Výsledky skriptu tedy zůstávají v pracovním prostředí i po jeho skončení. Skripty samozřejmě mohou volat jiné skripty nebo funkce, vytvářet grafická okna či vypisovat do **Command Window**.



Obr. 6.2 Skript pro vytvoření sinusového průběhu

Skript můžeme spustit z okna **Command Window** zadáním jeho názvu bez přípony. Například soubor *graf.m* spustíme voláním *graf*
Skript můžeme též spustit přímo v *m-File* položkami *Debug→Run* nebo stiskem klávesy *(F5)*.
Po proběhnutí našeho skriptu se vykreslí graf funkce sinus.



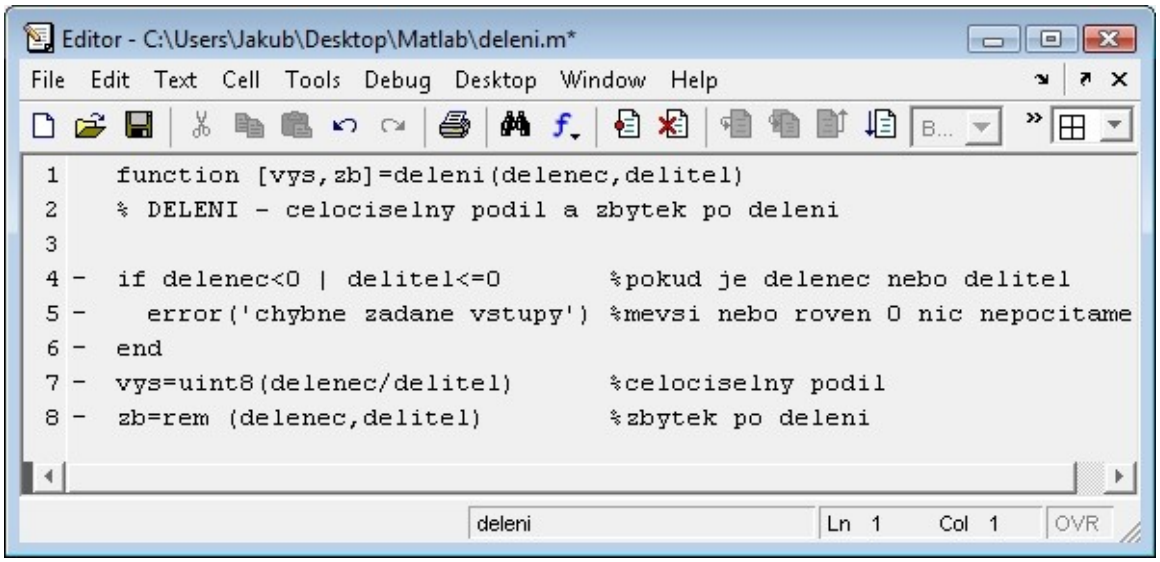
Obr. 6.3 Graf funkce sinus

6.2 Funkce

Pokud chceme využít stejný postup pro více situací, jsou skripty nevyhovující. Řešení nám nabízí funkce. Funkce je stejně jako skript *m-File.*Vytvoříme si tedy nový M-soubor příkazy *File → New→m-file.*
Protože funkce využívá vlastní prostředí, jsou všechny proměnné lokální – existují jen po dobu spuštění funkce. Proto je pro tvorbu funkcí nutné dodržovat určitá pravidla. Obecný předpis pro vytvoření funkce by měl vypadat takto:

```
function [vystupni_parametry] = jmeno_funkce
(vstupni_parametry)
```

První řádek tvoří tzv. definici funkce, která se skládá z klíčového slova function, vstupních parametrů (budou zobrazený v **Command Windows**), jména funkce sloužící pro její volání a vstupních parametrů (zadávány při spuštění funkce z **Command Windows**). Za definicí funkce pak následuje tělo funkce, které zajistí vlastní provedení postupu. Aby algoritmus tvořící funkci proběhl, musí být načteny všechny vstupy a vypočítány všechny výstupy nadefinované v prvním řádku funkce.



Obr. 6.4 Funkce pro celočíselný podíl dvou čísel

Pravidla pro zápis parametrů jsou následující:

výstupní parametry:
- je-li jich víc, oddělují se čárkou
- je-li jen jeden,hranaté závorky nejsou nutné
- funkce nemusí mít žádný výstup

jméno funkce:
- název funkce by měl vystihovat její činnost
- musí splnit pravidla pro název proměnné, jinak se funkci nepodaří spustit
- název funkce se nesmí shodovat s žádným názvem její proměnné

vstupní parametry:
- je-li jich víc, oddělují se čárkou
- funkce nemusí mít žádný vstup (pak se podobá skriptu, ale má své lokální **workspace)**

Vytvořenou funkci uložíme a můžeme ji spustit v okně **Command Windows**. Příkaz pro spuštění vypadá v případě, kdy chceme výsledky uložit do proměnných takto: *`[vystupni_parametry] = jmeno_funkce (vstupni_parametry)`* pokud není nutné výsledky ukládat do proměnných takto: *`jmeno_funkce (vstupni_parametry)`* Příklad volání zde vytvořené funkce:

```
>> [podil,zbytek]=deleni(17,4)
podil =
     4
zbytek =
     1
```

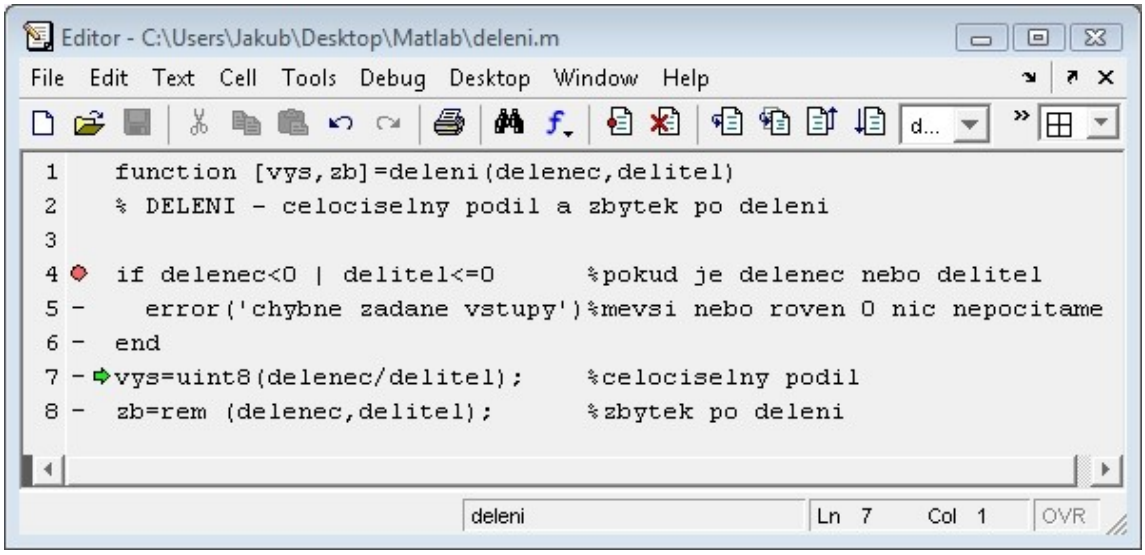
Obr. 6.5 Volání funkce z Command Windows

6.3 Ladění funkcí a skriptů

Pokud funkce nepracuje správně, přestože je syntakticky bez chyb, můžeme použít Debugger, který nám umožní její krokování a nabídne nám i řadu dalších užitečných funkcí:

1. v *m-editoru* nastavíme breakpoint (F12)
2. spustíme funkci z **Command window**)

3. pomocí F10 a F11 v M-editoru krokujeme (spouštíme jednotlivé příkazy)
4. výsledky kontrolujeme pomocí myši (kurzor přesuneme nad proměnnou) nebo pomocí okna **Workspace**.



Obr. 6.6 Odlaďování funkce

7 Řídící struktury

Lze je rozdělit do dvou skupin a to na podmínky a cykly. Podmínky využijeme, pokud například chceme, aby se určitý příkaz či skupina příkazů provedla při jejich splnění. Naproti tomu cykly slouží k opakovanému provádění příkazu.

7.1 Řídící struktura if

Slouží k rozvětvení programu, přičemž instrukce v těle funkce, tedy mezi *if* a *end* se provedou pouze tehdy, pokud bude splněna podmínka. Syntaxe příkazu je následující:

```
if podmínka
     příkaz
end
```

Překladač nejprve otestuje podmínku a v případě že je splněna, provede příkazy v těle struktury if. Není-li splněna, program pokračuje příkazem následujícím za tímto blokem.

```
1 -  a = input ('zadej a: ');
2 -  b = 2;
3 -  if a ~= 0
4 -      b=b^a
5 -  end
```

Obr. 7.1 Program pro výpočet druhé mocniny

V případě že chceme, aby se některý příkaz provedl za neplatnosti podmínky, můžeme využít *else*:

```
if podmínka
     příkaz 1
else
     příkaz 2
end
```

Program nejprve otestuje podmínku a v případě že je splněna, provede příkazy v těle struktury if. Není-li splněna, provede příkazy následující za *else*.

```
1    function [in]= cislo          %rozhodovaci funkce
2 -  a = input ('Zadejte a\n a= ') %vstupni parametr
3 -  if a >=0                      %rozhodovaci podminka
4 -    disp ('cislo je kladne')
5 -  else
6 -    disp ('cislo je zaporne')
7 -  end
```

Obr. 7.2 Program určující záporná a kladná čísla

Každá struktura větvení může obsahovat pouze právě jedno *if* a *else*. Vnoření další podmínky do již větvené struktury je možné pomocí konstrukce *elseif*:

```
if podmínka 1
     příkaz 1
elseif podmínka 2
     příkaz 2
else
     příkaz 3
end
```

Překladač nejdříve otestuje podmínku a v případě že je splněna, provede příkazy v těle struktury if. Není-li splněna, otestuje následující podmínku za *elseif*. V případě, že je podmínka splněna, provede příkazy, jestliže ne, provede příkazy následující za *else*.

```
 1  function [x1,x2]= KRCE (a,b,c)  %kvadraticka funkce
 2  a = input ('Zadejte a\n a= ')   %vstupni parametry
 3  b = input ('Zadejte b\n b= ')
 4  c = input ('Zadejte c\n c= ')
 5  if a==0                         %overeni reseni
 6    if b==0
 7        x='Nema reseni'
 8    else
 9        x=-c/b
10    end
11    %elseif error ('chyba vstupu')
12  else
13    x1=(-b+sqrt(b^2-4*a*c))/(2*a) %koreny rovnice
14    x2=(-b-sqrt(b^2-4*a*c))/(2*a)
15  end
16    x_kontrola=roots([a b c])     %kontrola vysledku
```

Obr. 7.3 Program pro výpočet kvadratické rovnice

7.2 Řídící struktura switch-case

Konstrukce *switch-case* je velmi podobná rozvětvenému příkazu *if*. Slouží k provádění stejného příkazu s mnoha hodnotami. Syntaxe je následující:

```
switch výraz
     case podmínka 1
          příkaz 1
     case podmínka 2
          příkaz 2
     otherwise
     příkaz
end
```

Překladač přečte výraz a poté vyhodnotí, zda je pro tento výraz splněna první podmínka. Pokud ano, provedou se příkazy následující za touto podmínkou, pokud ne, pokračuje se na další podmínku. V případě, že nevyhovuje ani jedna z podmínek, provedou se příkazy následující za *otherwise*.

```
 1 -  znamka = input ('zadej znamku: ');
 2 -  switch znamka
 3 -      case 1
 4 -          disp ('vybore, lepsi to byt nemuze')
 5 -      case 2
 6 -          disp ('slusny vykon')
 7 -      case 3
 8 -          disp ('docela to ujde')
 9 -      case 4
10 -          disp ('mel by jsi se polepsit')
11 -      case 5
12 -          disp ('takhle by to neslo')
13 -      otherwise
14 -          disp ('chyne zadany vstup')
15 -  end
```

Obr. 7.4 Program hodnotící známky

7.3 Řídící struktura for

Řídící struktura for provede opakování příkazu nebo skupiny příkazů, při předem stanoveném počtu opakování. Zápis je následující:

```
for n = vektor
     příkaz
end
```

Příkaz následující po *for n = vektor* se provádí pro každý prvek vektoru. Hodnota n se tedy s každým během smyčky zvětšuje o 1, dokud nedojde k horní hranici. Program pak pokračuje dalším příkazem následujícím po bloku *for*

```
 1    function [vys]=Faktorial(n)
 2 -  n= input ('Zadej n:\n n= ')
 3 -  vys=1;                      %inicializece vysledku
 4 -  if n>0                      %pouze kladna cisla
 5 -     for i=2:n
 6 -        vys=vys*i;            %vypocet faktorialu
 7 -     end
 8 -  else error ('chyba vstupu') %chybove hlaseni
 9 -  end
```

Obr. 7.5 Program pro výpočet faktoriálu

Jednotlivé cykly je možné i vnořovat. Konstrukce pak vypadá následovně:

```
for n = vektor
     for n = vektor
          příkaz
     end
end
```

```
 1 -  M1 = [-2,3,9;5,-1,-3;9,7,1];
 2 -  [a,b]=size (M1);              %zjisteni rozmeru matice
 3 -  M2=zeros(a,b);                %nulova matice o
 4                                  %stejnem rozmeru
 5 -  for i=1:a
 6 -      for j=1:b
 7 -          M2(i,j)=M1(i,b-j+1); %prehozeni sloupcu
 8 -      end
 9 -  end
10 -  M1                           %vypis puvodni matice
11 -  M2                           %vypis nove matice
```

Obr. 7.6 Přehození řádků a sloupců matice pomocí for cyklu

V případě většího počtu iterací by mělo být pole předem alokováno, pokud tomu tak není, musí MATLAB při každém kroku vytvářet nové větší pole a pole z předchozího kroku do něj kopírovat, čímž se značně zpomalí doba provádění skriptu či funkce.

7.4 Řídící struktura while

Pokud předem neznáme počet opakování, můžeme využít příkazu *while*. Tento cyklus se provádí tak dlouho, dokud není splněna daná logická podmínka.

```
while podmínka
     příkaz
end
```

```
1 -  n = input ('do kolika budu pocitat: \n');
2 -  k=1;    % inicializace promennych
3 -  s=0;
4 -  while k<=n  % mame jeste pricitat
5 -     s=s+k;   % pricteme k cislu
6 -     k=k+1;   % zvetsime clen o jedna
7 -  end
8 -  s           %vypis vysledku
```

Obr. 7.7 Program výpisu číselné řady

Cyklus bude provádět příkazy tak dlouho, dokud bude platná podmínka. Její platnost se testuje na začátku každého běhu cyklu. Pokud podmínka platí, provedou se příkazy nalézající se v těle cyklu. Pokud ne, program pokračuje za slovem end.

7.5 Využití input, break a continue, return

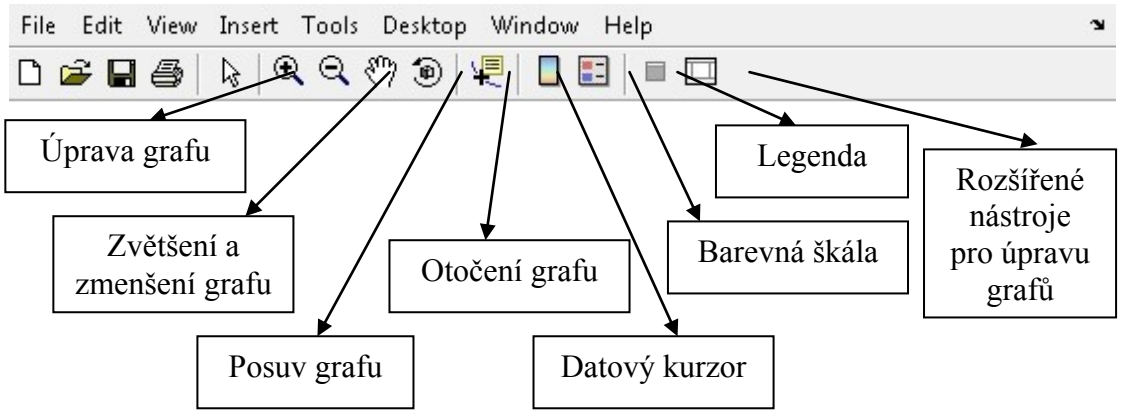
Input slouží k zadávání vstupních dat uživatelem. Jeho syntaxe vypadá následovně: *promenna= input('text')*. Vstupní data jsou načtena do proměnné *promenna*. V uvozovkách je textový řetězec, který je typu *string*.
Break slouží k přerušení právě probíhajícího cyklu. Program pokračuje za *end* právě přerušené smyčky.
Continue přejde na další iteraci právě probíhajícího cyklu. Příkazy následující po *continue* se přeskočí.
Return slouží k předčasnému ukončení funkce.

8 Grafický systém

Jednou z největších zbraní MATLABU je práce s grafikou. Můžeme si vybírat z velkého množství grafů, ať dvojrozměrných či trojrozměrných. Grafy je možné opatřovat názvy, měnit popisky a rozsah os, barvu a způsob vykreslování jednotlivých bodů či čar, přidávat další objekty, měnit celé pohledy. Zde se budeme zaobírat jen základními problémy grafického zobrazování dat, protože možnosti jsou opravdu široké a vydaly by na několik velmi silných knih.

8.1 Možnosti tvorby grafu

Grafy se zobrazují v novém okně **Figure Window** Přímo v tomto okně je možné za pomoci nástrojové lišty grafy upravovat.



Obr. 8.1 Popis nástrojové lišty grafického prostředí

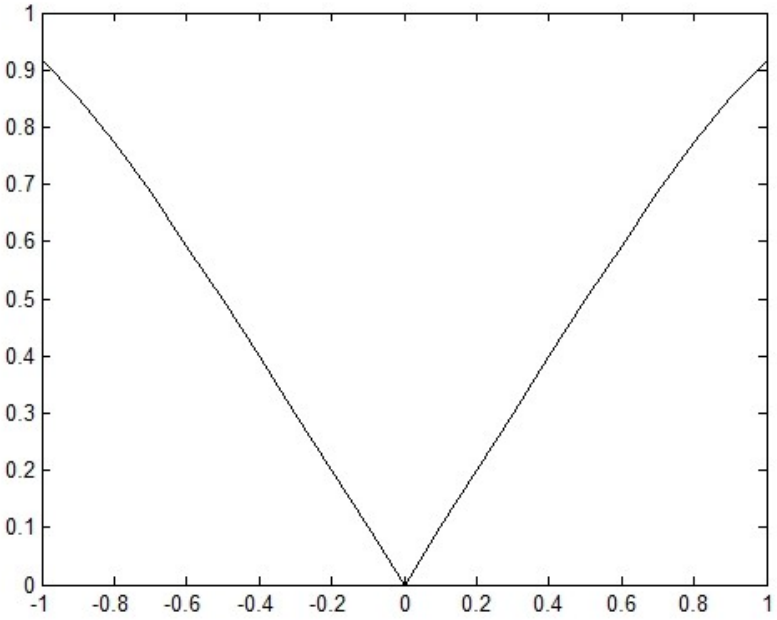
8.2 Grafika ve 2D

Základní funkcí pro vytvoření grafu je funkce plot. Pro ukázku vykresleme graf funkce $y = \sqrt{\sin x^2}$.

```
>> x=-1:0.1:1;
>> y= sqrt(sin(x.^2));
>> plot (x,y)
```

Obr. 8.2 Vytvoření 2D grafu

Nejprve si vytvoříme vektor v rozsahu od 0 do 2π s krokem 0,1. Těmto hodnotám přiřadíme jednotlivá y a pomocí funkce `plot` zobrazíme. Funkce `plot` sama otevře grafické okno **Figure window**, určí rozsah os a zobrazí jednotlivé body.



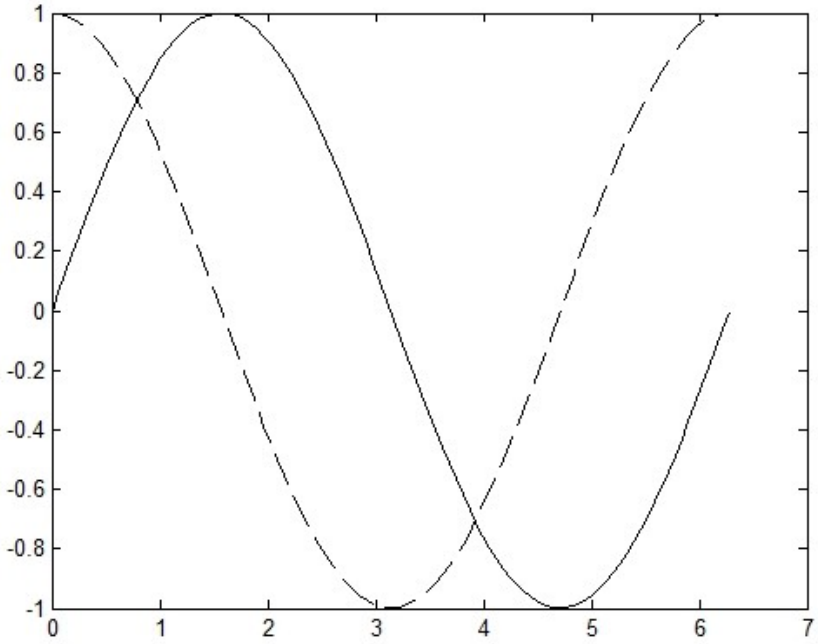
Obr. 8.3 Vytvoření 2D grafu

Do jednoho obrázku je samozřejmě možné vykreslit několik grafů. To provedeme přidáním dalšího páru vektorů.

```
>> x = linspace (0, 2*pi, 100);
>> y1 = sin(x);
>> y2 = cos(x);
>> plot (x, y1, x, y2)
```

Obr. 8.4. Vytvoření funkce sinus a cos

Zde jsme pro generování vektoru použili funkci `linspace`, která je pro tyto případy vhodnější. Obě křivky se zobrazily v jednom grafu. MATLAB implicitně odlišuje křivky barvami, což např. při černobílém tisku není příliš vhodné. Proto jsme dvakrát poklepali na křivku a v řádce *line* jsme změnili typ čáry na čerchovanou a barvu na černou.



Obr. 8.5 Graf funkce sinus a cosinus

Pokud je jedním z parametru vektor a druhým matice, vykreslí se každý sloupec matice na zadaném vektoru stejně. Předchozí graf je tedy možné vykreslit více způsoby.

```
>> Y = [y1; y2];
>> plot (x, Y)
```

Obr. 8.6. Jiná možnost vytvoření grafu funkce z předchozího příkladu

8.3 Styl čar, značky a barva

V předchozím případě jsme si ukázali, jak měnit barvu a styl čáry pomocí palety nástrojů v okně **Figure Window**. Samotné změny můžeme však provádět již při psaní zdrojového kódu. Změny provádíme pomocí textového řetězce, jenž tvoří nepovinný parametr funkce `plot`. Symboly sloužící k těmto změnám jsou uvedeny v následujících tabulkách:

Symbol	Barva
b	modrá (blue)
g	zelená (green)
r	červená (red)
c	modrozelená (cyan)
m	purpurová (magenta)
y	žlutá (yellow)
k	černá (black)
w	bílá (white)

Tab. 8.1 Barvy značek a čar

Symbol	Značka
.	bod
o	kruh
x	křížek
+	plus
*	hvězda
s	čtverec
d	diamant
v	trojúhelník dolů
^	trojúhelník nahoru
>	trojúhelník vpravo
<	trojúhelník vlevo
p	pentagram
h	hexagram

Tab. 8.2 Typy značek

Symbol	styl čáry
-	plná
:	tečkovaná
-.	čerchovaná
--	čárkovaná

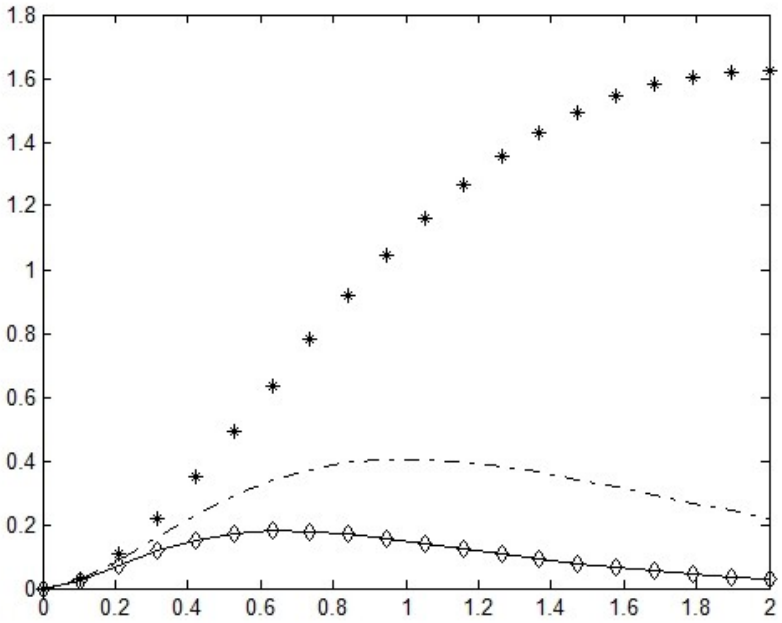
Tab. 8.3 Styly čar

Pokud nepoužijeme tyto symboly, MATLAB automaticky kreslí křivky plnou čarou bez značek v modré barvě.

```
>> x=linspace (0,2,20);
>> y1 = 3*x.^2.*exp(-3*x);
>> y2 = 3*x.^2.*exp(-2*x);
>> y3 = 3*x.^2.*exp(-1*x);
>> plot (x, y1, 'kd-', x, y2, 'k-.', x, y3, 'k*')
```

Obr. 8.7. Změna stylu čar, značek a barvy

Výsledek bude opět zobrazen v jednom grafickém okně. Tentokrát již s odpovídajícími parametry.



Obr. 8.8 Výsledek změny stylu čas, značek a barvy

8.4 Styl grafu, mřížka, popisky os

Pro zadání názvu grafu disponuje MATLAB příkazem *title*. Dále je možné zobrazit popisky os příkazy *xlabel* pro osu x a *ylabel* pro osu y. Kromě nadpisu a popisku os můžeme graf vybavit samozřejmě také legendou. Tu spustíme příkazem *legend*. Mřížku je možné zapnout příkazem *grid on* a vypnout příkazem *grid off*. Příkaz grid bez parametru přepne aktuální zobrazení. Většina grafů se standardně vypisuje s vypnutou mřížkou.
Každý 2D graf je uzavřen do rámečku který lze vypnout příkazem *box*.

```
>> x = linspace (0,2*pi,50);
>> y =exp(sin(2*x));
>> plot (x,y1,'k');
>> title ('Graf funkce exp(sin(2*x))');
>> xlabel ('x');
>> ylabel ('y');
>> grid on;
>> legend ('exp(sin(2*x))')
```

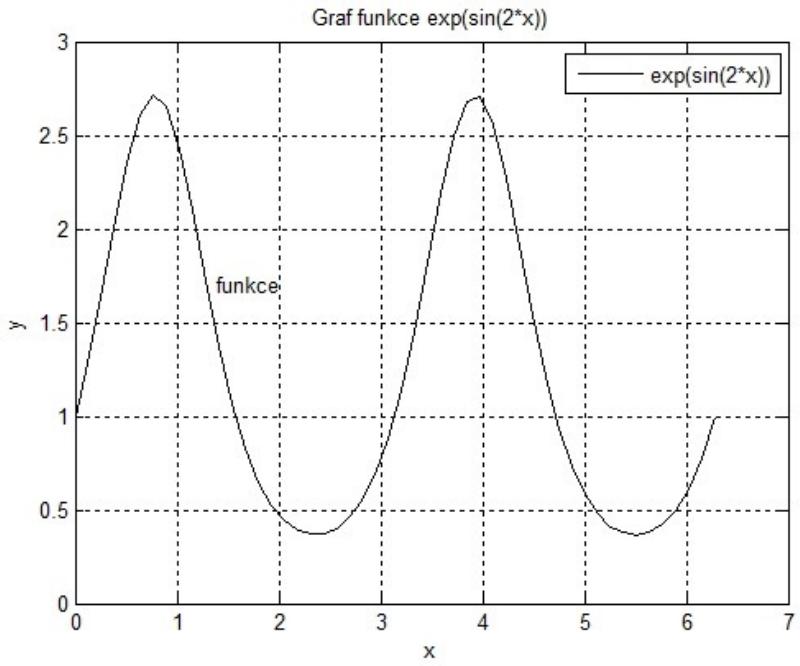
Obr. 8.9 Změna stylu grafu, popisky os a mřížka

Velmi užitečný je také příkaz *text(x, y, 'textovy retezec')*, který umožňuje zapsat do grafu libovolný textový řetězec na námi určené souřadnice. Jako měřítka pro určení souřadnic se berou měřítka os grafu.

```
>> text (1.4,1.7,'funkce')
```

Obr. 8.10 Popis funkce

Výsledek vidíme v následujícím grafu:



Obr. 8.11 Výsledek změny slylu grafu, popisky os a mřížka

Kromě popisků os je také možné měnit jejich měřítka pomocí příkazu `xlim` a `ylim`, jejichž parametrem je vektor.
Příkaz axis slouží k nastavení měřítka obou os. Některé z možných syntaxí jsou uvedeny v následující tabulce.

`axis ([xmin, xmax, ymin, ymax])`	Nastavení meze, parametr funkce je vektor s mezními hodnotami
`axis equel`	Na obou osách stejné měřítko
`axis square`	Čtvercový tvar grafu
`axis normal`	Standardní vzhled
`axis off`	Vypne osy, popisky a značky na osách
`axis on`	Zapne osy, popisky a značky na osách

8.5 Tloušťka čar, velikost a barva značek

Kromě změn stylu křivek, můžeme ovlivňovat také jejich grafický vzhled. Kromě tedy již známých parametrů jako je typ čáry, její barva, druh značky, může příkaz `plot` obsahovat ještě další parametry.

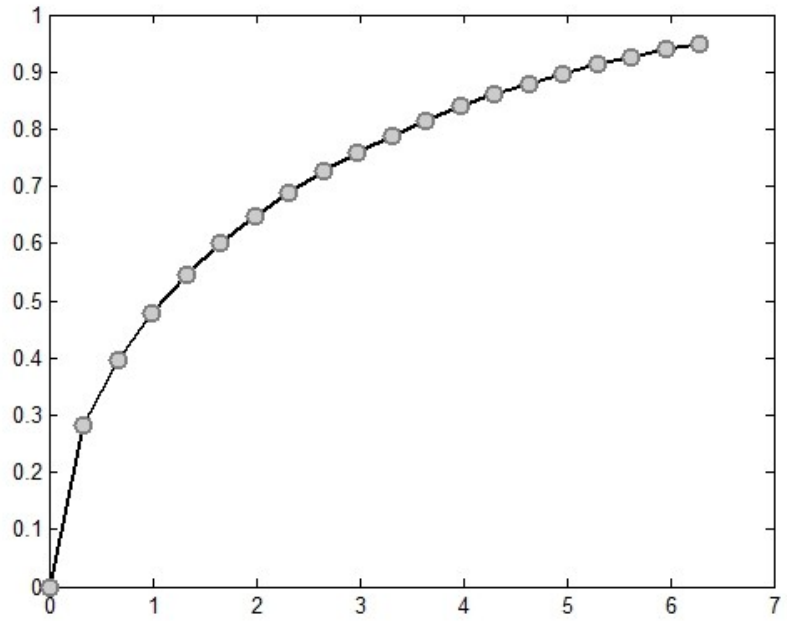
`LineWidth`	Tloušťka čáry v bodech
`MarkerSize`	Velikost značky v bodech
`MarkerEdgeColor`	Barva okraje značky
`MarkerFaceColor`	Barva výplně značky

Pokud neurčíme jinak, bude se graf vykreslovat s tloušťkou čáry jedna. To samé platí i pro tloušťku značek. Dvojnásobné tloušťce odpovídá tedy hodnota dva body poloviční půl apod.

```
>> x = linspace (0, 2*pi, 20);
>> y = sin (sqrt(x)/2);
>> plot (x,y,'k-o','LineWidth', 2,'MarkerSize',8,...
'MarkerEdgeColor','g','MarkerFaceColor','y');
>> grid on
```

Obr. 8.12 Změna tloušťky čar, velikosti a barvy značek

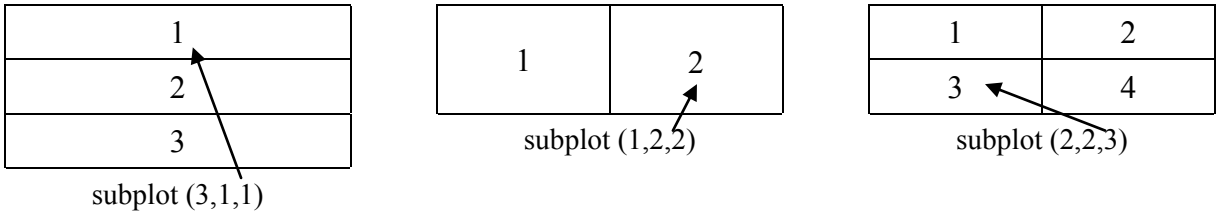
Příkaz plot vykreslí graf funkce. První dva parametry jsou vektory x a y pro vytvoření vlastní funkce, třetí parametr nastaví styl čáry, její barvu a druh značky. Pak následují naše nové parametry *'LineWidth',2,* nastaví tloušťku čáry na velikost dvou bodů *'MarkerSize',8,* pak velikost značky na osm bodů. *'MarkerEdgeColor','g',*Zabarví vnější okraj značky do zelena a *'MarkerFaceColor','y',* vyplní plochu značky žlutou barvou. Výsledek pak vypadá takto:

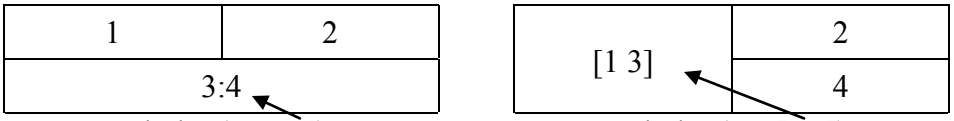


Obr. 8.13 Výsledek změny tloušťky čar, velikosti a barvy značek

8.6 Funkce subplot

Okno, ve kterém se graficky zobrazují výsledky naší práce, nemusí vždy obsahovat pouze jeden graf. Použijeme-li funkci *subplot (m, n, p)*, můžeme v jednom okně vykreslit hned několik grafů. Přesněji m x n grafů. Funkce totiž rozdělí grafické okno mřížkou o m řádcích a n sloupcích. Parametr p pak slouží pro přístup k jednotlivým pozicím. Samotné pozice jsou určeny indexy. To znamená, že příkaz subplot (1,2,2) rozdělí grafické okno mřížkou o jednom řádku a dvou sloupcích, z nichž aktivní bude ten druhý.





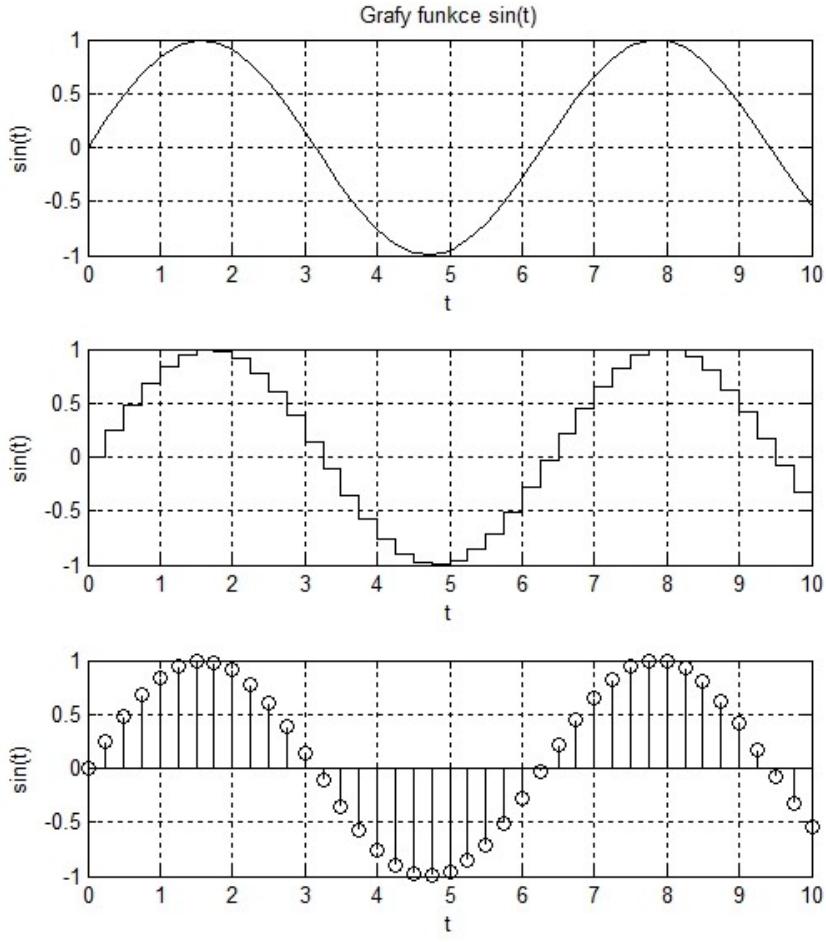
subplut (2,2,3:4) subplot (2,2,[1 3])

Jednotlivé řádky či sloupce můžeme spojovat a vytvářet tak větší prostor pro grafy. Princip je podobný jako při vybírání řádků či sloupců u matic.

```
>> x = (0:0.25:10);
>> y = sin (x);
>> subplot (3,1,1)
>> plot (x,y,'k-'),xlabel('t'),ylabel('sin(t)'),grid on
>> title ('Grafy funkce sin(t)')
>> subplot (3,1,2)
>> stairs(x,y,'k'),xlabel('t'),ylabel('sin(t)'),grid on
>> subplot (3,1,3)
>> stem(x,y,'k'),xlabel('t'),ylabel('sin(t)'),grid on
```

Obr. 8.14 Využití subplot

Praktické využití funkce subplot vidíme na následujícím obrázku:



Obr. 8.15 Graf s využitím subplot

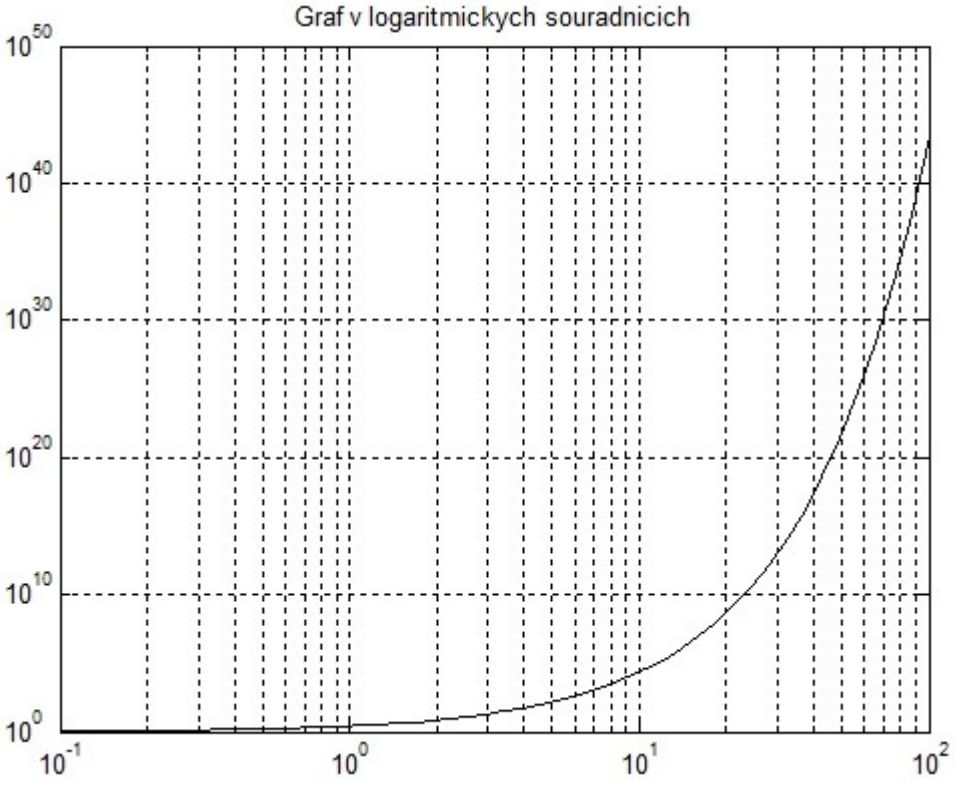
8.7 Kreslení v logaritmických a semilogaritmických souřadnicích

Pokud chceme nahradit některou z os grafu nebo obě logaritmickými souřadnicemi, disponuje MATLAB třemi funkcemi *semilogx*, *semilogy*, *loglog*. Funkce *semilogx* a *semilogy* slouží pro kreslení v semilogaritmických souřadnicích, kdy je jedna osa lineární a druhá logaritmická. Funkce *loglog* pak pro grafy kde jsou třeba obě osy zobrazit logaritmicky.

```
>> x=logspace (-1,2); %logaritmicky vektor
>> loglog (x, exp(x),'k');
>> grid on;
>> title ('Graf v logaritmickych souradnicich')
```

Obr. 8.16 Kreslení v logaritmických souřadnicích

Následuje příklad grafu s oběma osami v logaritmických souřadnicích. Při jeho tvorbě jsme využili funkci *logspace(a,b)*, která slouží ke generování náhodného vektoru od a do b logaritmicky.



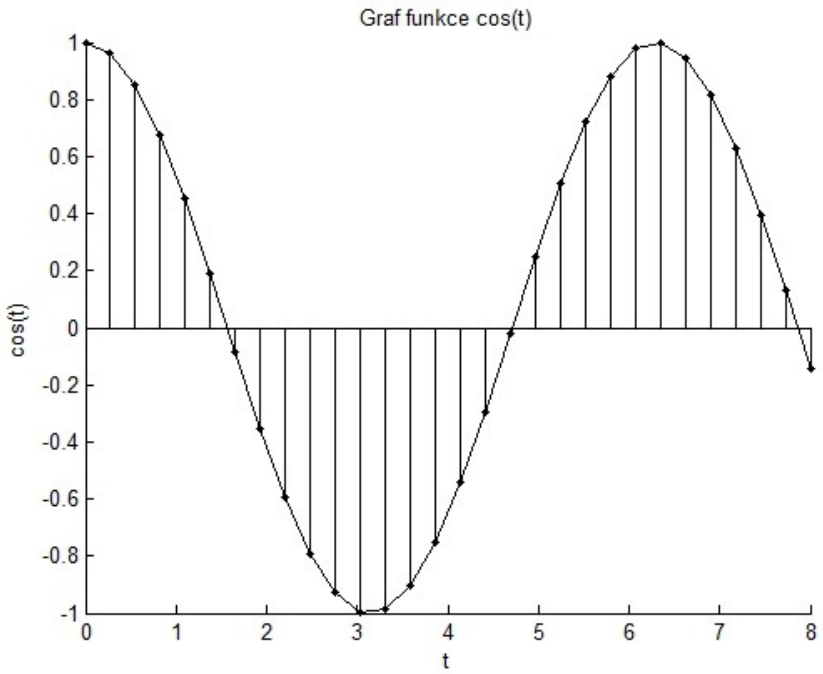
Obr. 8.17 Graf v logaritmických souřadnicích

8.8 Více typů grafů v jednom okně

Více grafů v jednom obrázku můžeme vytvořit pomocí nám již známého příkazu *hold on*, který způsobí, že se grafické okno neotevře ihned po zapsání příkazu plot, ale dovolí nám vložit do téhož obrázku např. stopkový (stem) graf. Vše pak ukončíme příkazem *hold off*.

```
>> k = linspace (0,8,30);
>> hold on
>> plot (k,cos(k),'k-')
>> stem (k,cos(k),'k.-')
>> ylabel('cos(t)')
>> xlabel('t')
>> title ('Graf funkce cos(t)')
>> hold off
```

Obr. 8.18 Vytvoření více grafů v jednom okně



Obr. 8.19 Výsledek zobrazení více grafů v jednom okně

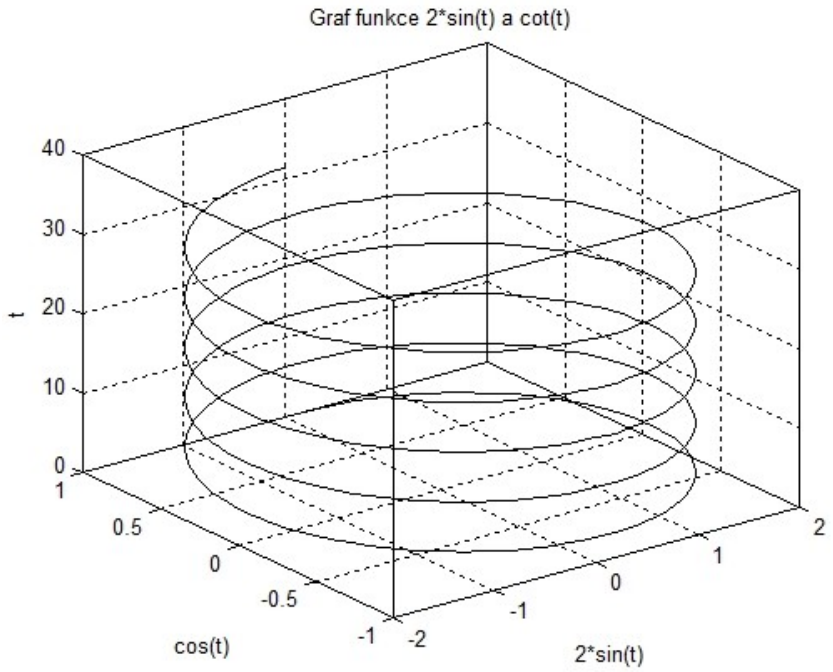
8.9 Grafika ve 3D

Příkazy používané při tvorbě 3D grafů se příliš neliší od dvourozměrných. Grafy se zobrazují v trojrozměrných souřadnicích. Příkaz `plot(x,y)` na který již jsme zvyklí z 2D musíme tedy ještě rozšířit o jeden vektor. Výsledkem je příkaz `plot3(x,y,z)`, kde dvě proměnné jsou nezávislé a jedna závislá.

```
>> t=0:pi/50:10*pi;
>> plot3(2*sin(t),cos(t),t,'-k')
>> grid on
>> box on
>> xlabel ('2*sin(t)')
>> ylabel ('cos(t)')
>> zlabel ('t')
>> title ('Graf funkce 2*sin(t) a cot(t)')
```

Obr. 8.20 Vykreslení 3D grafu

Příkazy sloužící k editaci grafů, tedy změnám stylu čar, značek, barev, zapnutí mřížky, popiskům os, tloušťky čar apod. Jsou stejné jako ve 2D zobrazení.



Obr. 8.21 Výsledek vykreslení 3D grafu

8.10 Skalární funkce dvou proměnných

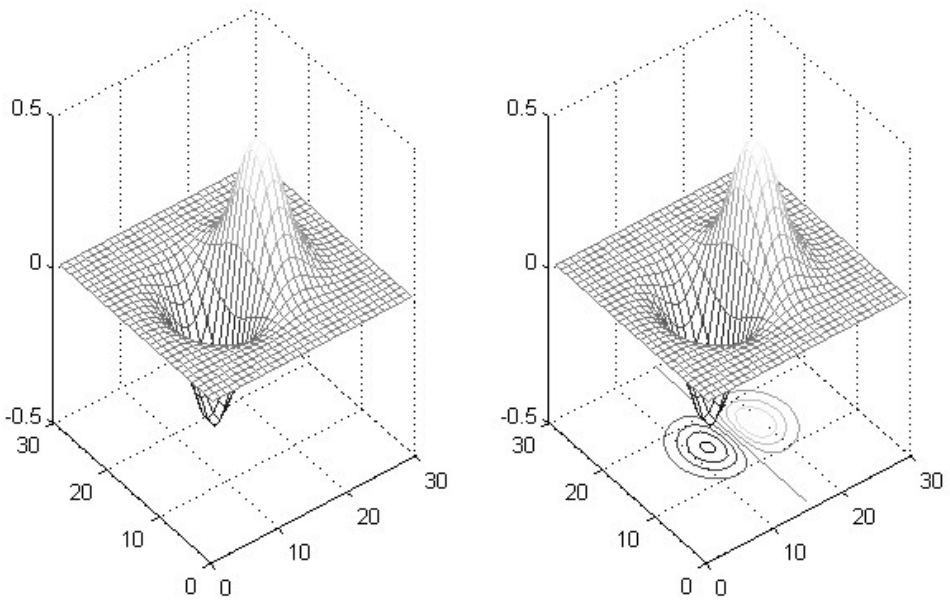
Kromě spojitých grafů lze kreslit i grafy plošné. MATLAB je orientován maticově, proto pro vykreslení takového grafu je třeba dvou vektorů a matice, jež bude funkcí těchto vektorů. Dojde k vytvoření matic z vektorů v příslušných směrech. V MATLABu je k tomuto účelu funkce *meshgrid*. Ta vytvoří výstupy obsahující rozkopírované matice.

Samotný graf je pak vykreslen příkazem *mesh*, který vytvoří síťovaný graf. Hustota sítě je závislá na počtu bodů definovaných vektorů.

```
>> x=linspace (-3,3,30);
>> y=x;
>> [xm,ym]=meshgrid(x,y);
>> z=xm.*exp(-xm.^2-ym.^2);
>> hold on
>> subplot (1,2,1)
>> mesh(z)
>> subplot (1,2,2)
>> meshc(z)
>> hold off
```

Obr. 8.22 Vykreslení 3D grafu pomocí funkce mesh

Příklad síťovaného grafu:

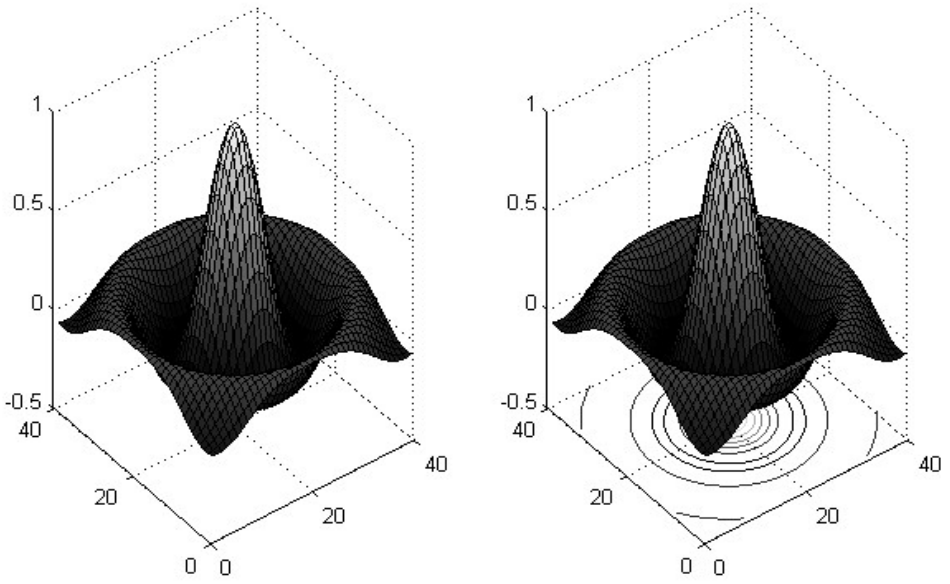


Obr. 8.23 Výsledek vykreslení 3D grafu pomocí funkce mesh

Místo instrukce *mesh* můžeme využít příkaz *surf*. Výsledný graf bude mít stejný průběh, ale jednotlivé plošky budou barevně vyplněné.

```
>> x= linspace (-8,8,40);
>> y=x;
>> [xm, ym]=meshgrid (x,y);
>> z=sin(sqrt(xm.^2+ym.^2))./sqrt(xm.^2+ym.^2);
>> subplot (1,2,1)
>> surf (z)
>> subplot (1,2,2)
>> surfc (z)
>> hold off
```

Obr. 8.24 Vykreslení 3D grafu pomocí funkce surf



Obr. 8.25 Výsledek vykreslení 3D grafu pomocí funkce surf

9 Další typy 2D a 3D grafů

Systém MATLAB disponuje ještě celou řadou dosud nezmiňovaných grafů. Na některé z nich se podrobněji podíváme v této části.

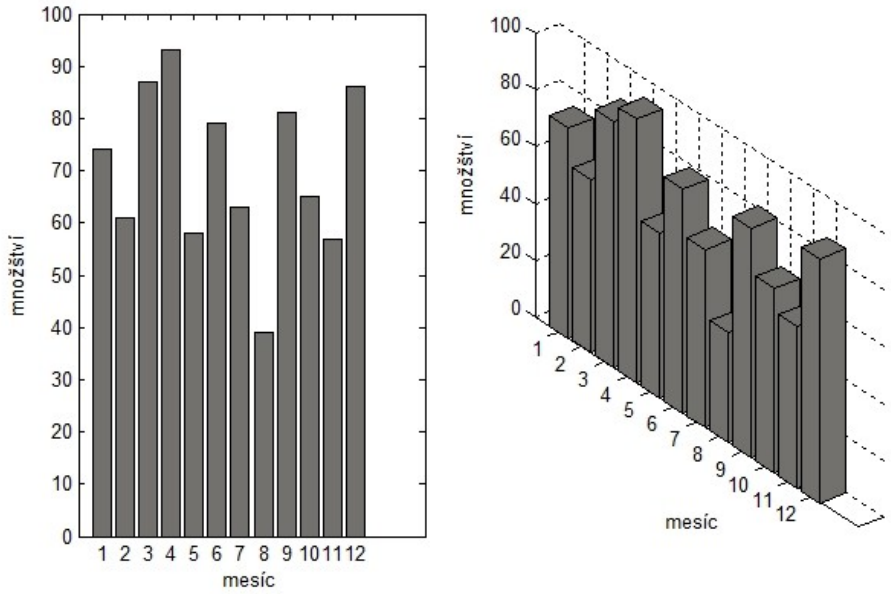
9.1 Sloupcový graf

Sloupcové grafy můžeme tvořit příkazem *bar*, nebo *bar3*, pokud data chceme zobrazovat ve 3D rovině.

```
>> x=1:12;
>> y=[74,61,87,93,58,79,63,39,81,65,57,86];
>> hold on
>> subplot (1,2,1)
>> bar(x,y),xlabel('mesíc'), ylabel('množství')
>> subplot (1,2,2)
>> bar3(x,y),ylabel('mesíc'), zlabel('množství')
>> hold off
```

Obr. 9.1 Vytvoření sloupcového grafu

Parametry grafu můžeme měnit stejně jako u příkazu *plot*. Chceme li graf zobrazit horizontálně, je možné využít funkci *barh (x,y)* Další podrobnosti o tvorbě sloupcových grafů najdete zadáním *help bar* v příkazovém řádku.



Obr. 9.2 Sloupcový 2D a 3D graf

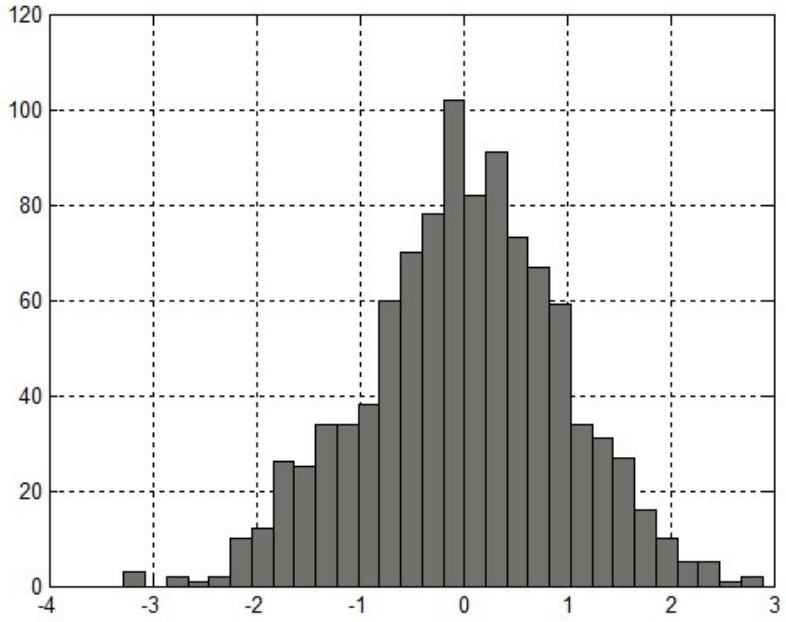
9.2 Histogram

Histogramy jsou grafy sloužící k zobrazení četnosti hodnot. Využívají se např. při statistickém vyhodnocování dat.

```
>> x=randn(1000,1);
>> hist (x,30);
>> grid on
```

Obr. 9.3 Vytvoření histogramu

Zde jsme pro tvorbu histogramu použili generátor náhodných čísel, který generuje 1000 náhodných čísel s normálním rozdělením. Příkaz `hist` pak vygenerovaná data rozčlenil do 30 intervalů a zobrazil jejich četnost.



Obr. 9.4 Histogram

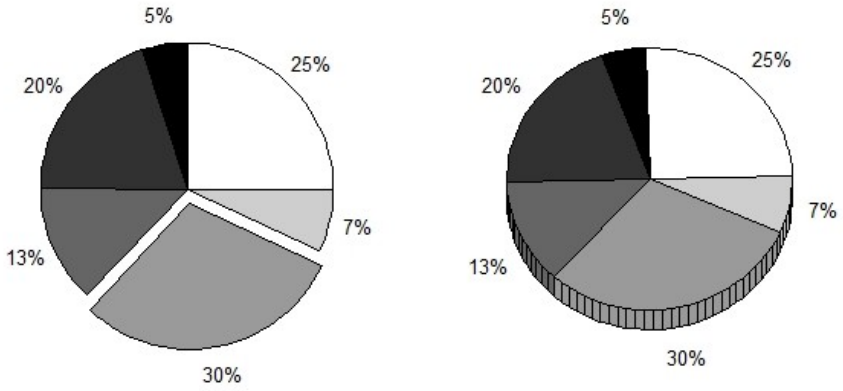
9.3 Výsečový graf

Výsečový graf je také často využívaným typem. Zobrazíme je příkazem `pie`, nebo `pie3`.

```
>> p=[0.5,2,1.3,3,0.7,2.5];
>> vysec = [0 0 0 1 0 0];
>> hold on
>> subplot (1,2,1)
>> pie(p,vysec)
>> subplot (1,2,2)
>> pie3(p)
>> hold off
```

Obr. 9.5 Vytvoření koláčového grafu

Zajímavá je možnost zvýraznění určité časti grafu, jež se provádí zapsáním jedničky v nulovém vektoru na pozici, která má být zvýrazněna.



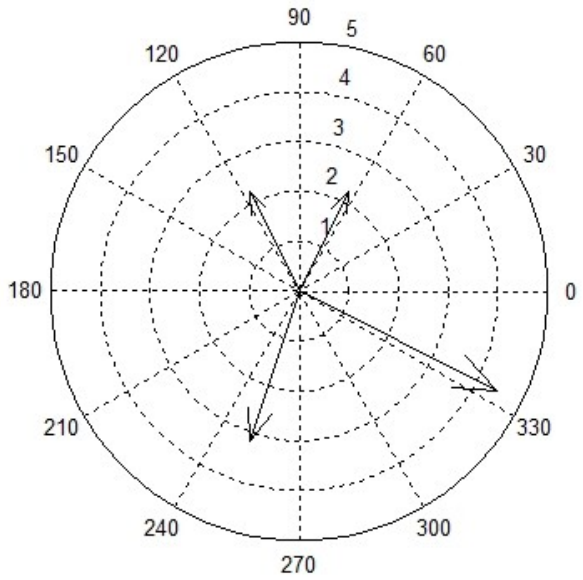
Obr. 9.6 Koláčový 2D a 3D graf

9.4 Kompasový graf

Vytvoříme jej příkazem `compass`. Slouží k zobrazování čísel v komplexní rovině. Výsledkem jsou šipky směřující od počátku souřadného systému.

```
>> x = [1+2j, 4-2j, -1-3j, -1+2j];
>> compass(x,'k')
```

Obr. 9.7 Vytvoření kompasového grafu



Obr. 9.8 Kompasový graf

9.5 Symboly a formát popisků

Některé veličiny se popisují symboly, které se běžně v anglické abecedě, jež MATLAB standardně používá, nevyskytují. Tyto symboly pak mají svůj charakteristický zápis. Seznam symbolů a jejich vyjádření v MATLABu je uveden v nápovědě k textu *(`doc text`)* odkaz `string`.
Možné je měnit také styl písma, jeho velikost, barvu, apod.

`\it {text}`	kurzíva
`\bf{text}`	tučné písmo
`\rm{text}`	normální písmo
`\fontname{nazev_pisma}`	změna písma
`\fontsize{velikost}`	změna velikosti písma
`\color{barva}`	změna barvy písma
_	dolní index
^	horní index

Pokud je potřeba více znaků pro indexování veličiny, je třeba je uzavřít do složených závorek. Takovéto způsoby formátování lze použít v příkazech `text`, `title`, `xlabel`, `ylabel`, `zlabel`. Použití ukazují následující příklady:

```
>> text(0.1,0.8,'\ite^{i\omega\tau} = cos(\omega\tau) + i sin(\omega\tau)')
```

$$e^{i\omega\tau} = cos(\omega\tau) + i\ sin(\omega\tau)$$

Příklad stylů a využití indexů:

```
>> text(0.1,0.4,'\fontsize{15}\it{a_nx^n+a_{n-1}x^{n-1}+...+a_1x+a_0=0}')
```

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0$$

Obr. 9.9. Symboly a formát popisků

10 Řešené příklady

V této části se zaměříme na praktické využití toho, co jsme doposud probrali a ukážeme si možnosti použití MATLABu na několika komplexních příkladech.

Příklad 1: Rovnoměrně zrychlený pohyb

Řidič spatří policejní vůz a začne brzdit. Na dráze 88m zpomalí z rychlosti 75 km/h na 45 km/h.

a) Určete zpomalení automobilu za předpokladu, že bylo během brzdění konstantní.
b) Jak dlouho řidič v této fázi pohybu brzdil.
c) Řidič dále brzdí se zrychlením určeným v čase (a) Za jak dlouho od začátku brzdění se automobil zcela zastaví?
d) Jakou dráhu urazí vůz od počátku brzdění do úplného zastaveni?

Matematické řešení:

$$a_x = \frac{v_x^2 - v_{0x}^2}{2\cdot(x - x_0)} = \frac{(45\text{km/h})^2 - (75\text{km/h})^2}{2\cdot(0{,}088)} = -2{,}05\cdot 10^4\ \text{km/h}^2 = -1{,}6\ \text{m}\cdot\text{s}^{-2}$$

$$t = \frac{2\cdot(x - x_0)}{v_{0x} + v_x} = \frac{2\cdot(0{,}088)}{(75+45)\text{km/h}} = 1{,}5\cdot 10^{-3}\,\text{h} = 5{,}4\ \text{s}$$

$$t_{celk} = \frac{v_x - v_{0x}}{a_x} = \frac{0 - (75\text{km/h})}{(-2{,}05\cdot 10^4\,\text{km/h}^2)} = 3{,}7\cdot 10^{=3}\,\text{h} = 13\ \text{s}$$

$$x - x_0 = v_{0x}t_{celk} + \frac{1}{2}a_x t_{celk}^2 = (75\text{km/h})\cdot(3{,}7\cdot 10^{=3}\,\text{h}) + \frac{1}{2}(-2{,}05\cdot 10^4\,\text{km}\cdot\text{h}^{=2})\cdot(3{,}7\cdot 10^{=3}\,\text{h})^2 =$$

$$= 0{,}137\text{km} = 140\text{m}$$

Řešení pomocí MATLABu:

```
%zadane veliciny
v = 45/3.6; % m/s
v0 = 75/3.6; % m/s
s = 88; % m

%vypocty
a= (v^2 - v0^2)/(2*s);
t=(2*s)/(v0+v);
t2=(0-v0)/a;
scelk=(v0*t2)+(0.5*a*t2^2);

%vypis vysledku
fprintf('\n')
fprintf ('a) a = %4.2f ms-2\n', a)
fprintf ('b) t = %4.2f s\n', t)
fprintf ('c) t celkova = %4.2f s\n', t2)
fprintf ('d) s celkova = %4.2f m\n', scelk)
```

Příklad 2: Práce

Určete práci, kterou vykoná sila F (5;3;1)[N] působící z bodu X (0;0;1)[m] po přímé dráze do bodu Y (1;1;3)[m] a určete úhel, který svírá sila F a dráha s

Matematické řešení:

$$\vec{s} = \overline{X} \cdot \overline{Y} = Y - X = \left((y_1 - x_1);(y_2 - x_2);(y_3 - x_3)\right) = \left((1-0);(1-0);(3-1)\right) = (1;1;2) \text{ m}$$

$$W = \vec{F} \cdot \vec{s} = (F_1 \cdot s_1) + (F_2 \cdot s_2) + (F_3 \cdot s_3) = 5 \cdot 1 + 3 \cdot 1 + 1 \cdot 2 = 10 \text{ J}$$

$$W = \left|\vec{F}\right| \cdot |\vec{s}| \cdot \cos\varphi \Rightarrow \cos\varphi = \frac{W}{\left|\vec{F}\right| \cdot |\vec{s}|} = \frac{10}{\sqrt{5^2+3^2+1^2} \cdot \sqrt{1^2+1^2+2^2}} = \frac{10}{\sqrt{35} \cdot \sqrt{6}} = \frac{10}{14{,}69694} = 0{,}68041$$

$$\varphi = 44°54'$$

Řešení pomocí MATLABu:

```
%zadane veliciny
X = [0 0 1];
Y = [1 1 3];
F = [5 3 1];

%vypocet prace a uhlu
s = Y - X;
W = sum(F.*s);
pom = W/sqrt(sum(F.^2)*sum(s.^2));
fi = cos (pom)*180/pi;

%vypis vysledku
fprintf ('\n')
fprintf ('W = %4.3f J\n', W)
fprintf ('fi = %4.3f°\n', fi)
```

Příklad 3: Soustava rovnic

Vyřešte soustavu lineárních rovnic:

$$2x_1 + 3x_2 + 4x_3 + 5x_4 = 26$$
$$3x_1 + 4x_2 + 5x_3 + 2x_4 = 25$$
$$4x_1 + 5x_2 + 2x_3 + 3x_4 = 24$$
$$5x_1 + 2x_2 + 3x_3 + 4x_4 = 23$$

Matematické řešení:

$$\left(\begin{array}{cccc|c} 2 & 3 & 4 & 5 & 26 \\ 3 & 4 & 5 & 2 & 25 \\ 4 & 5 & 2 & 3 & 24 \\ 5 & 2 & 3 & 4 & 23 \end{array}\right) \approx \left(\begin{array}{cccc|c} 2 & 3 & 4 & 5 & 26 \\ 0 & -0{,}5 & -1 & -5{,}5 & -14 \\ 0 & -1 & -6 & -7 & -28 \\ 0 & -5{,}5 & -7 & -8{,}5 & -42 \end{array}\right) \approx \left(\begin{array}{cccc|c} 2 & 3 & 4 & 5 & 26 \\ 0 & -1 & -6 & -7 & -28 \\ 0 & -0{,}5 & -1 & -5{,}5 & -14 \\ 0 & -5{,}5 & -7 & -8{,}5 & -42 \end{array}\right) \approx$$

$$\approx \left(\begin{array}{cccc|c} 2 & 3 & 4 & 5 & 26 \\ 0 & -1 & -6 & -7 & -28 \\ 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 26 & 30 & 112 \end{array}\right) \approx \left(\begin{array}{cccc|c} 2 & 3 & 4 & 5 & 26 \\ 0 & -1 & -6 & -7 & -28 \\ 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 13 & 15 & 56 \end{array}\right) \approx \left(\begin{array}{cccc|c} 2 & 3 & 4 & 5 & 26 \\ 0 & -1 & -6 & -7 & -28 \\ 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 0 & 28 & 56 \end{array}\right)$$

$h_s = h_r = n = 4 \Rightarrow$ jedno řešení

$28x_4 = 56 \Rightarrow x_4 = 2$

$2x_3 - 2 \cdot 2 = 0 \Rightarrow x_3 = 2$

$-x_2 - 6 \cdot 2 - 7 \cdot 2 = -28 \Rightarrow x_2 = 2$

$2x_1 + 3 \cdot 2 + 4 \cdot 2 + 5 \cdot 2 = 26 \Rightarrow x_1 = 1$

Řešení pomocí MATLABu:

```
%prava strana matice
A = [2 3 4 5; 3 4 5 2;4 5 2 3; 5 2 3 4];

%leva strana matice
b = [26; 25; 24; 23];

%determinant matice A
D = det(A);

%dosazeni b do matice A
A1 = [b,A(:,2:4)];
A2 = [A(:,1),b,A(:,3:4)];
A3 = [A(:,1:2),b,A(:,4)];
A4 = [A(:,1:3),b];

%vysledne x
x1=det(A1)/D;
x2=det(A2)/D;
x3=det(A3)/D;
x4=det(A4)/D;

%vypis vysledku
fprintf ('\n')
fprintf ('x1 = %2.0f \n', x1)
fprintf ('x2 = %2.0f \n', x2)
fprintf ('x3 = %2.0f \n', x3)
fprintf ('x4 = %2.0f \n', x4)
```

Příklad 4: Tlumené kmity

Tlumený oscilátor na obr. 10.1 je popsán parametry m = 250 g, k = 80 $N \cdot m^{-1}$ a b = 70 $g \cdot s^{-1}$. Určete periodu kmitáni a čas, za který se amplituda kmitání zmenši na polovinu své počáteční velikosti? Graficky znázorněte pro amplitudu x = 10.

Matematické řešení:

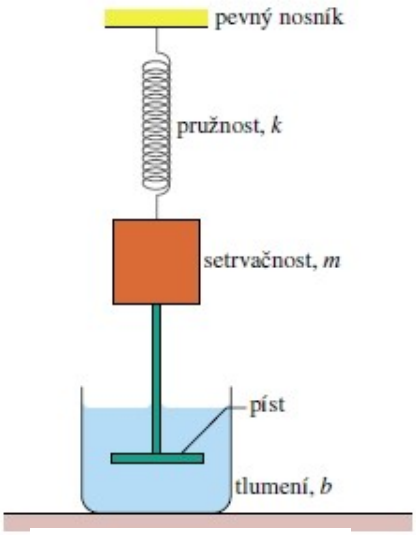
$$T = 2\pi \cdot \sqrt{\frac{m}{k}} = 2\pi \cdot \sqrt{\frac{0{,}25}{80}} = 0{,}34 \text{ s}$$

$x_m e^{-bt/(2m)} \Rightarrow$ rovnice pro $\frac{1}{2}$ amplitudy:

$$x_m e^{-bt/(2m)} = \frac{1}{2} x_m$$

$$\ln \frac{1}{2} = \ln e^{-bt/(2m)} = \frac{-bt}{2m}$$

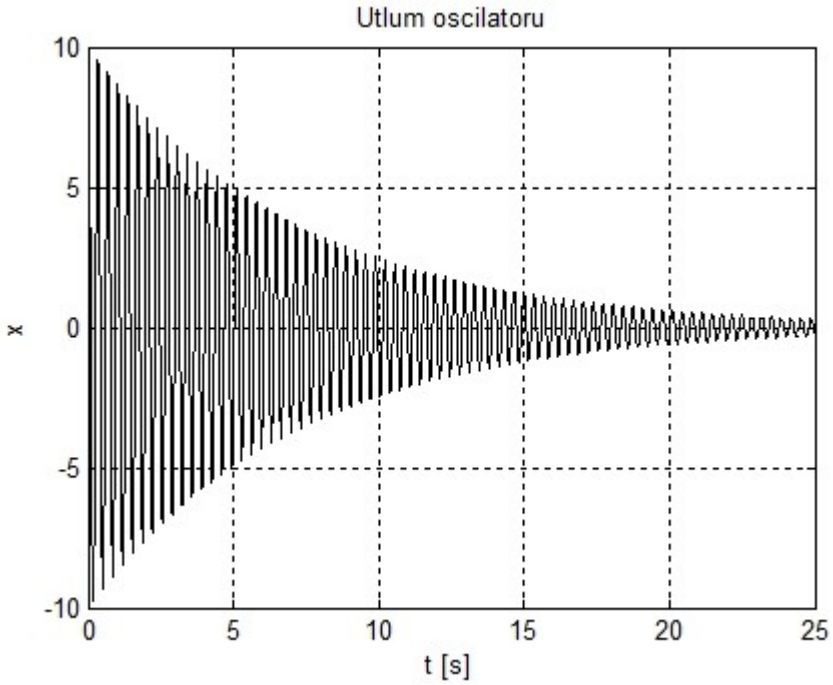
$$t = \frac{-2m \cdot \ln\left(\frac{1}{2}\right)}{b} = \frac{-2 \cdot 0{,}25 \cdot \ln\left(\frac{1}{2}\right)}{0{,}070} = 5{,}0 \text{ s}$$



Obr. 10.1 Oscilátor

Řešení pomocí MATLABu:

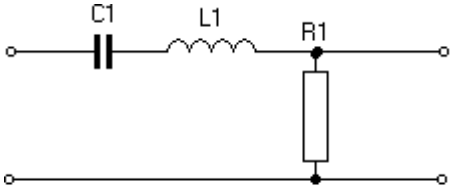
```
%vstupni veliciny
m = 250/1000 %kg;
k = 85 %N*m-1;
b = 70/1000 %kg*s-1;

%vypocet periody
T = 2*pi*sqrt(m/k);

%vypocet 1/2 amplitudy
t = (-2*m*log(0.5))/b;

%vypis vysledku
fprintf('\n')
fprintf ('T = %4.2f s\n', T)
fprintf ('t = %4.2f s\n', t)

%graficke znazorneni
tv = (0:0.01:25);
omega = sqrt(k/m-b^2/(4*m^2));
x = 10*exp((-b.*tv)/(2*m)).*cos(omega.*tv);
plot(tv,x,'k')
grid on
xlabel ('t [s]')
ylabel ('x')
title ('Utlum oscilatoru')
```



Obr. 10.2 Utlum oscilátoru

Příklad 5: Pásmová propusť

Určete impedanci Z a rezonanční kmitočet f_r pásmové propusti s parametry U=12V, L_1 = 1H, C_1=3μF, a R_1 = 100Ω Graficky znázorněte amplitudovou a frekvenční charakteristiku.



Obr. 10.3 Schéma zapojení pásmové propusti

Matematické řešení:

$$f_r = \frac{1}{2\pi \cdot \sqrt{L_1 \cdot C_2}} = \frac{1}{2\pi \cdot \sqrt{1 \cdot 3 \cdot 10^{-6}}} = 91{,}888 \text{ Hz}$$

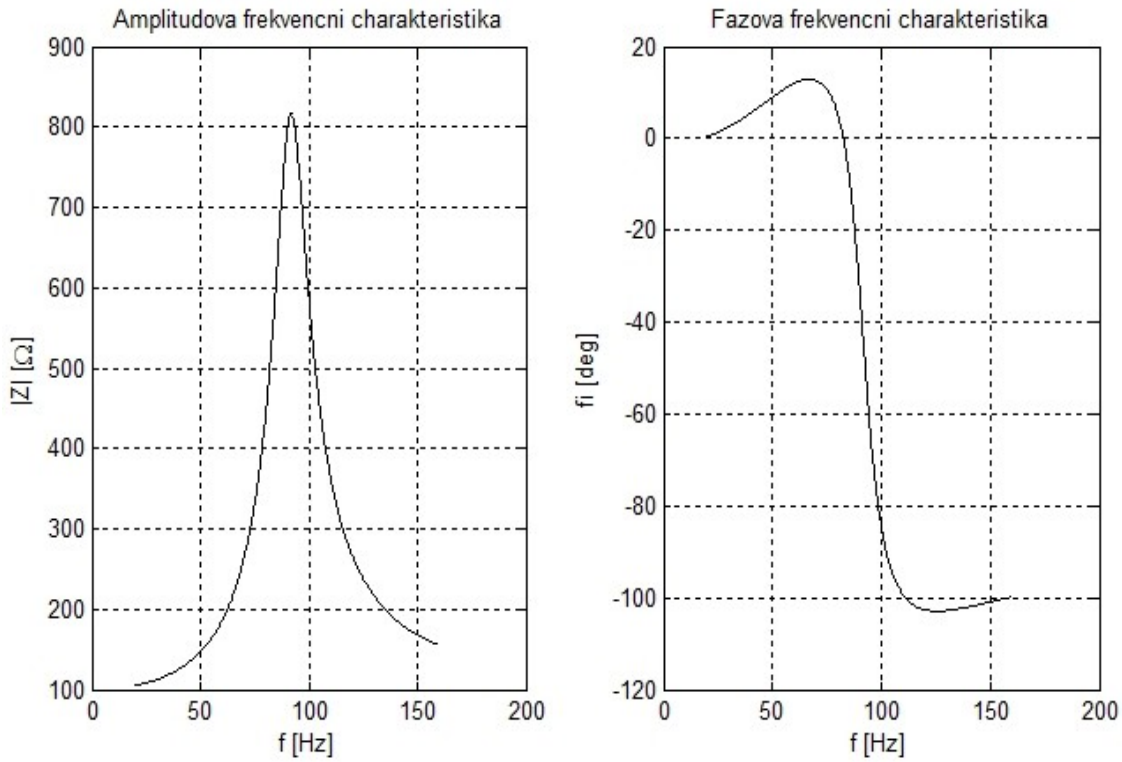
$$XL = \omega \cdot L_1 = 2 \cdot \pi \cdot f_r \cdot L_1 = 577{,}35 \ \Omega$$

$$XC = \frac{1}{\omega \cdot C_1} = \frac{1}{2 \cdot \pi \cdot f_r \cdot C_1} = 577{,}35 \ \Omega$$

$$Z = \frac{(j \cdot XL - j \cdot XC) \cdot R_1}{j \cdot XL - j \cdot XC + R_1} = 816{,}497 \angle -45^o \ \Omega$$

Řešení pomocí MATLABu:

```
%REZONANCNI OBVOD
%zadane hodnoty
R=100; %ohm
L=1; %H
C=3e-6; %F
U= 12; %V
f= linspace (20,160,1000); %Hz

%Vypocet Z a rezonancni f
fr=1/(2*pi*sqrt(L*C));
omega=2*pi*f;
Z=((omega*L-j*1./(omega*C)).*R)./(j*omega*L-j*1./(omega*C)+R)

%vypis vysldku
fprintf ('\n')
fprintf ('f_r = %4.3f Hz\n', fr)

%GFAR ampolitudove frekvencni charakteristiky
subplot (1, 2, 1)
plot (f,abs(Z),'k')
title ('Amplitudova frekvencni charakteristika')
xlabel('f [Hz]')
ylabel('|Z| [\Omega]')
grid on

%GFAR fazove frekvencni charakteristiky
subplot (1, 2, 2)
plot (f,angle(Z)*(180/pi),'k')
title ('Fazova frekvencni charakteristika')
xlabel('f [Hz]')
ylabel('fi [deg]')
grid on
```

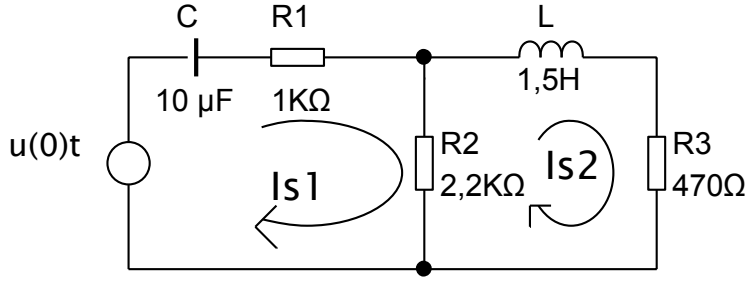


Obr. 10.4 Graf amplitudové a frekvenční charakteristiky

Příklad 6: Metoda smyčkových proudů:

Na obrázku je elektrický obvod, který je připojený na napětí $u_0(t)=100 \sin(\omega t + 30)$ o frekvenci 50 Hz. V obvodu jsou zapojeny rezistory s hodnotami odporu R_1=1 kΩ, R_2=2,2 kΩ a R_3=470 Ω, cívka o indukčnost L=1,5 H a kondenzátor o kapacitě C =10 μF. Vypočtěte a graficky zobrazte průběhy napětí na jednotlivých prvcích

a) pomocí fázorů.
b) v závislosti na čase.



Obr. 10.5 Schéma zapojení obvodu

Matematické řešení:

$$\omega = 2 \cdot \pi \cdot f \qquad X_L = \omega \cdot L = 471{,}23\ \Omega \qquad X_C = \frac{1}{\omega \cdot C} = 318{,}31\ \Omega$$

$$U_0 = \frac{100}{\sqrt{2}} \cdot e^{j30} = 70{,}71 \angle 30^\circ\ \text{V}$$

$$\begin{pmatrix} (-j \cdot X_C + R_1 + R_2) & -R_2 \\ -R_2 & (j \cdot X_L + R_2) \end{pmatrix} \cdot \begin{pmatrix} I_{s_1} \\ I_{S_2} \end{pmatrix} = \begin{pmatrix} U_0 \\ 0 \end{pmatrix}$$

Vypočteme třikrát determinant Δ, Δ_1, Δ_2

$$\Delta = \begin{pmatrix} (-j \cdot X_C + R_1 + R_2) & -R_2 \\ -R_2 & (j \cdot X_L + R_2) \end{pmatrix} = 5.212 \cdot 10^6 - 9.999 \cdot 10^5 \text{ j}$$

$$\Delta_1 = \begin{pmatrix} (-j \cdot X_C + R_1 + R_2) & U_0 \\ -R_2 & 0 \end{pmatrix} = 1.924 \cdot 10^5 + 1.111 \cdot 10^5 j$$

$$\Delta_2 = \begin{pmatrix} U_0 & -R_2 \\ 0 & (j \cdot X_L + R_2) \end{pmatrix} = 1.347 \cdot 10^5 + 7.778 \cdot 10^4 \text{ j}$$

$$I_{S_1} = \frac{\Delta_1}{\Delta} = 0{,}0317 + 0{,}0274\text{j A} \qquad I_{S_2} = \frac{\Delta_2}{\Delta} = 0{,}0222 + 0{,}0192\text{j A}$$

$$I_1 = I_{S_1} = 0{,}0317 + 0{,}0274\text{j A} \qquad I_2 = I_{S_2} = 0{,}0222 + 0{,}0192\text{j A}$$

$$I_3 = I_{S_1} - I_{S_2} = 0{,}0095 + 0{,}0082\text{j A}$$

$$U_C = I_1 \cdot (-j \cdot X_C) = 18{,}841 \angle -49{,}14^\circ \text{ V}$$

$$U_{R_1} = I_1 \cdot R_1 = 59{,}190 \angle 40{,}86^\circ \text{ V}$$

$$U_{R_2} = I_3 \cdot R_2 = 39{,}019 \angle 40{,}86^\circ \text{ V}$$

$$U_L = I_2 \cdot j \cdot X_L = 19{,}535 \angle 130{,}86^\circ \text{ V}$$

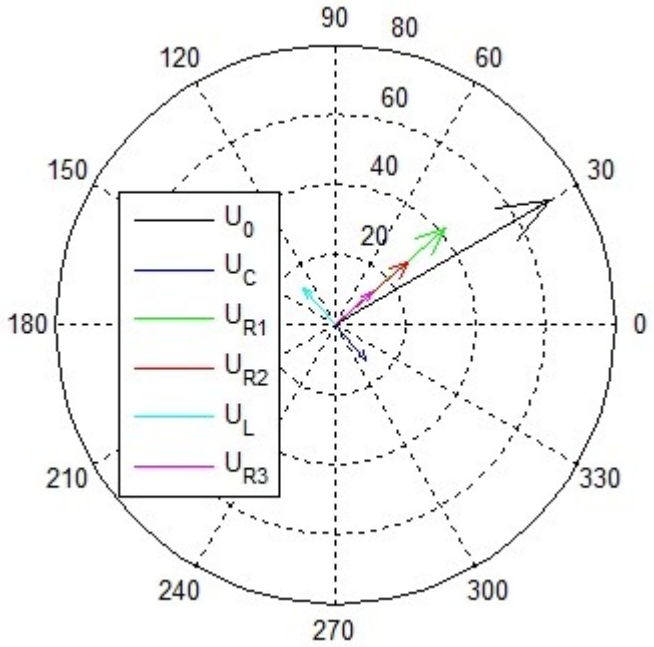
$$U_{R_3} = I_2 \cdot R_3 = 19{,}484 \angle 40{,}86^\circ \text{ V}$$

Řešení pomocí MATLABu:

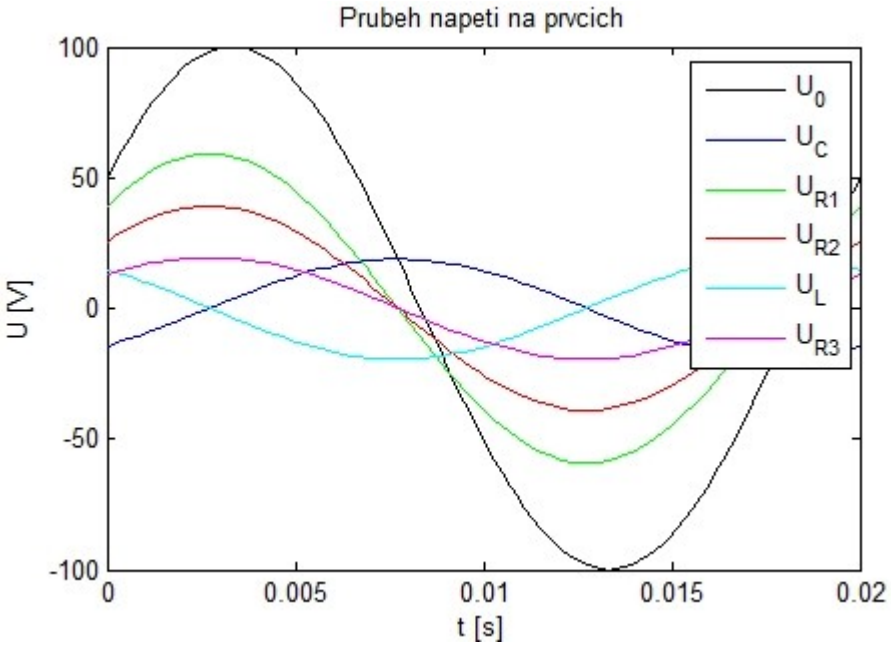
```
%METODA SMYCKOVYCH PROUDU
%zadane veliciny
Um = 100; %V
f = 50; %Hz
fi = 30; %stup
L = 1.5; %H
C = 10e-6; %F
R1 = 1e3; %ohm
R2 = 2.2e3; %ohm
R3 = 470; %ohm

%Prevod napeti a vypocet impedanci
Uef = Um/sqrt(2);
omega = 2*pi*f;
XL = omega*L;
XC = 1/(omega*C);
U0 = Uef*exp(j*fi/180*pi);
```

```
%matice pasivnich prvku
M = [(-j*XC+R1+R2)   -R2;...
        -R2          (R2+R3+XL)];

%vektor napeti
Us = [UO; 0];

%vypocet smyckovych proudu
Is =M\Us

%vypocet napeti na prvcich
UC = -j*XC*Is(1)
UR1 = R1*Is(1)
UR2 = R2*(Is(1)-Is(2))
UL = j*XL*Is(2)
UR3 = R3*Is(2)

U = [UO; UC; UR1; UR2; UL; UR3];

%vypocet amplitudy a uhlu
Um = (abs(U)*sqrt(2));
Uuhel = angle(U);
Ustup = Uuhel/pi*180;

%vypis vysledu
fprintf ('\n')
fprintf ('UO = %4.3f*sin(wt+ %4.3f)\n',Um(1),Ustup(1))
fprintf ('UC = %4.3f*sin(wt %4.3f)\n',Um(2),Ustup(2))
fprintf ('UR1 = %4.3f*sin(wt+ %4.3f)\n',Um(3),Ustup(3))
fprintf ('UR2 = %4.3f*sin(wt+ %4.3f)\n',Um(4),Ustup(4))
fprintf ('UL = %4.3f*sin(wt+ %4.3f)\n',Um(5),Ustup(5))
fprintf ('UR3 = %4.3f*sin(wt+ %4.3f)\n',Um(6),Ustup(6))

%Zobrazeni prubehu pomoci fazoru
figure;
compass (U(1),'k');
hold on;
compass (U(2),'b');
compass (U(3),'g');
compass (U(4),'r');
compass (U(5),'c');
compass (U(6),'m');
hold off
axis auto
legend ('U_O','U_C','U_{R1}','U_{R2}','U_L','U_{R3}')
```

```
%Zobrazeni napeti v zavislosti na cese
t=linspace (0, 1/f, 100);

figure;
plot (t,Um(1)*sin(omega*t+Uuhel(1)),'k')
hold on;
plot (t,Um(2)*sin(omega*t+Uuhel(2)),'b')
plot (t,Um(3)*sin(omega*t+Uuhel(3)),'g')
plot (t,Um(4)*sin(omega*t+Uuhel(4)),'r')
plot (t,Um(5)*sin(omega*t+Uuhel(5)),'c')
plot (t,Um(6)*sin(omega*t+Uuhel(6)),'m')
legend ('U_0','U_C','U_{R1}','U_{R2}','U_L','U_{R3}')
hold off;
title ('Prubeh napeti na prvcich')
xlabel ('U [V]')
ylabel ('t [s]')
```



Obr. 10.6 Průběhy napětí v polárních souřadnicích



Obr. 10.6 Průběhy napětí v závislosti na čase

Příklad 7: Pohyb bodu v krabici

Poslední příklad je ukázkou využití MATLABu nejen jako programu určeného pro výpočty ale i možnost jeho využití jako programovacího jazyka srovnatelného např. s C++. Zde vytvořený skript ukazuje pohyb bodu v krabici.

```
%hodnoty
x=0.3;
y=0.6;
z=0.9;

%rychlost ve smeru os
v_x=0.1; %m/s
v_y=0.1; %m/s
v_z=0.1;

%Vytvoreni krabice
h=plot3(x,y,z,'o','MarkerSize',7,'MarkerFaceColor','g')
axis([0 1 0 1 0 1])
grid on
box on

%pohyb bodu
for i=1:1:1500
    t=1;

    x=x+v_x*t;
    y=y+v_y*t;
    z=z+v_z*t;
```

```
%podminka zmeny smeru pro y - horni mez
    if y>=1
        y=0.9;
        v_y=-v_y;
    end
%podminka zmeny smeru pro y - dolni mez
    if y<=0
       y=0.1;
        v_y=-v_y;
    end;
%podminka zmeny smeru pro x - horni mez
    if x>=1
        x=0.9;
        v_x=-v_x;
    end
%podminka zmeny smeru pro x - dolni mez
    if x<=0
        x=0.1;
        v_x=-v_x;
    end;
%podminka zmeny smeru pro z - horni mez
    if z>=1
        z=0.9;
        v_z=-v_z;
    end
 %podminka zmeny smeru pro z - dolni mez
    if z<=0
       z=0.1;
        v_z=-v_z;
    end

    pause(0.2)
    set(h,'xdata',x,'ydata',y,'zdata',z)
    drawnow
end
```

11 Závěr

Předložená práce je odrazovým můstkem pro všechny zájemce o práci s programem MATLAB. Jelikož se jedná o mohutný, stále se rozvíjející komerční produkt, roste i počet příkazů, které jsou uživateli k dispozici. Každý uživatel tak po krátké práci s programem zjistí, že mu postačí znalost syntaxe pouze několika nejčastěji používaných příkazů. U těch méně používaných se pak ukazuje jako výhodné využití kvalitně zpracované programové nápovědy.

Doufám, že tato práce bude přínosem pro všechny zájemce o práci s programem MATLAB a pomůže jin co nejlépe řešit problémy a úkoly ve kterých budou tento systém využívat.

Použitá literatura:

[1] Karel Zaplatílek, Bohuslav Doňar : MATLAB pro začátečníky, 2.vydání, Praha, Ben 2005, počet str. 151, ISBN 80-7300-175-6

[2] Karel Zaplatílek, Bohuslav Doňar : MATLAB tvorba uživatelských aplikací, Praha, Ben 2004, počet str. 209, ISBN 80-7300-133-0

[3] Pavel Kaban, Výpočty a simulace v programech MATLAB a Simulink, Brno Computer Press, počet str. 220, ISBN 80-251-1301-9

[4] Halliday D.,Resnick R., Walker J.: Fyzika, Brno, Vutium, počet str. 1193, ISBN 80-214-1869-9

[5] Jiří Tesař: Sbírka úloh z matematiky pro fyziky, České Budějovice, Jihočeská univerzita 1995, počet str. 101 ISBN 80-7040-133-8

WWW zdroje:

[6] http://uprt.vscht.cz/majerova/matlab/

[7] http://www.cdm.cas.cz/czech/hora/vyuka/mvs/tutorial.pdf

[8] http://vyukaap.vscht.cz/HTML/kap1.html

[9] http://upload.wikimedia.org/wikibooks/cs/7/74/Matlab_(tvorba_grafů).pdf

[10] http://web.fit.vutbr.cz/study/courses/ISS/public/labs/01_matlab/matlab_hrave.pdf

[11] http://www.roznovskastredni.cz/dwnl/pel2007/13/Donar.pdf

[12] http://labe.felk.cvut.cz/~posik/x33scp/matlab-primer/uvoddomatlabu.html#12

[13] http://artax.karlin.mff.cuni.cz/~beda/cz/matlab/primercz/matlab-primer.html#toc

[14] http://www.mendelu.org/upload//matlab-zaklady.pdf

[15] http://ladislav.prskavec.net/pdf/2003_02_Tvorba_ucebniho_textu_MATLAB.pdf

[16] http://www.fs.cvut.cz/cz/U201/map/matlab/index.htm

[17] http://www.comtel.cz/files/download.php?id=3114

[18] http://upload.wikimedia.org/wikibooks/cs/d/d5/Matlab_(základní_informace).pdf

[19] http://www.tufts.edu/~rwhite07/Matlab.pdf

[20] http://lukas.am.vsb.cz/Teaching/FMMI/matlab.pdf

[21] https://www.powerwiki.cz/attach/ProStudenty/APP_cv1_Matlab.ppt

[22] www.kky.zcu.cz/uploads/courses/aks/MATLAB.ppt

[23] http://www.umt.fme.vutbr.cz/~ruja/vyuka/matlab0304.pdf

Jiné zdroje:

[24] Nápověda programu MATLAB – MATLAB Help