Informatika

MATLAB – úvod do použití

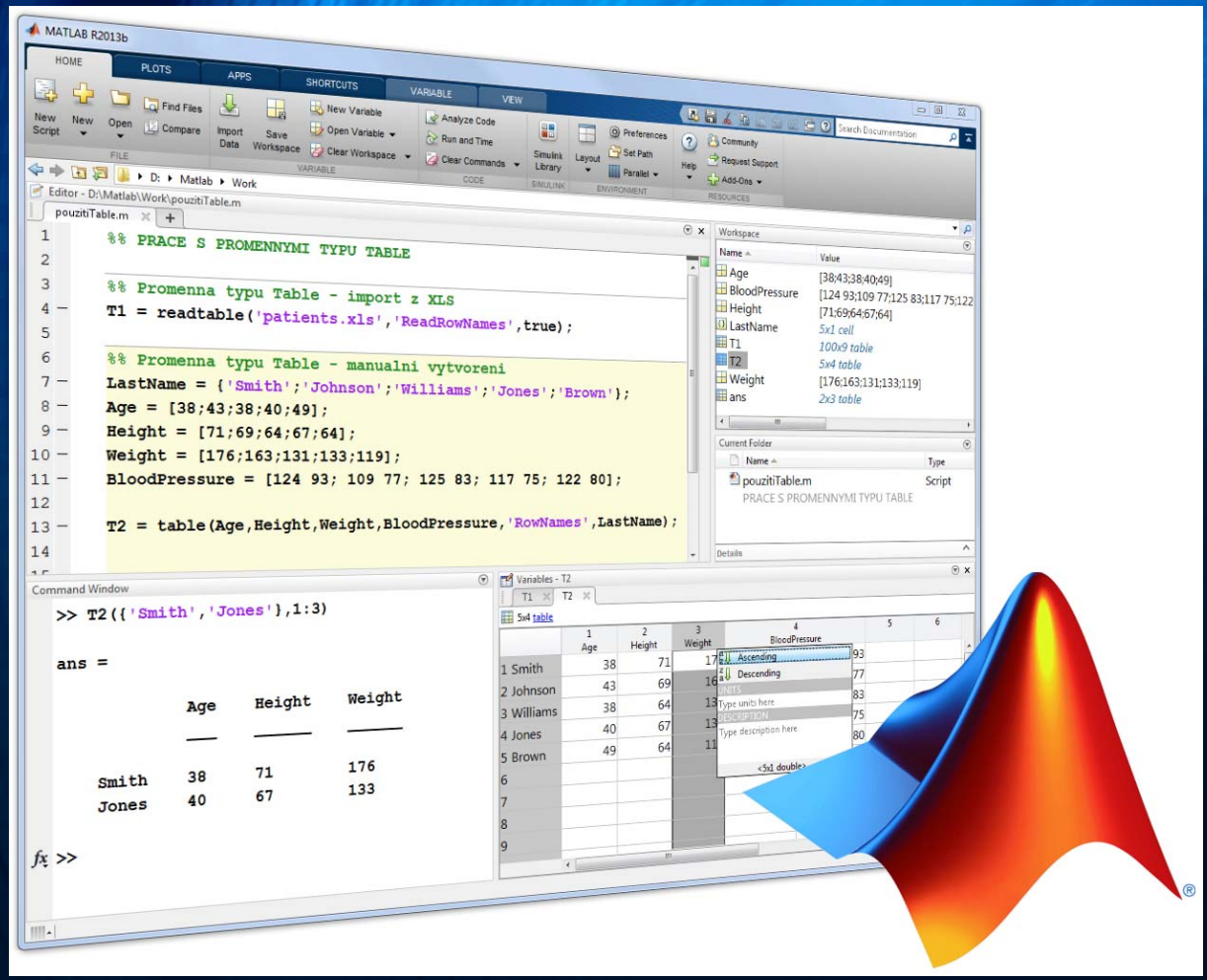
Část 1. - práce v příkazovém okně

Technická univerzita v Liberci, FM MTI , Janeček J.

MATLAB

(Matrix – Laboratory)

MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů.



PRODUKT

The MathWorks, Inc.
3 Apple Hill Drive, Natick, Massachusetts 01760 USA
http://www.mathworks.com/

Výhradní zastoupení pro ČR a Slovensko

HUMUSOFT s.r.o.
Pobřežní 20, 186 00 Praha 8
http://www.humusoft.cz/



- Systém MATLAB patří mezi základní výpočetní nástroje na mnoha vzdělávacích a výzkumných institucích po celém světě.
- Více než 5 000 univerzit používá MATLAB k výzkumu a zkvalitnění výuky v oblasti technických výpočtů, analýzy dat a simulace.
- MATLAB je běžným nástrojem v řadě průmyslových i ekonomických odvětví.

Doporučená literatura :

- Zaplatílek, K., Doňar, B.: *MATLAB pro začátečníky.* BEN, Praha 2005.
- Dušek, F: *Matlab a Simulink.* Univerzita Pardubice, 2000.
- Karban, P.: *Výpočty a simulace v programech Matlab a simulink.* Comuter Press, Brno 2006.
- Kozák, Š., Kajan, S.: *Matlab-Simulink 1* . STU, Bratislava, 1999.
- Kupka, L.: *Matlab a Simulink, úvod do použití*. SOŠ a SOU, Lanškroun, 2007.
- Janeček, J., Kupka, L.: *Matlab a Simulink, řešené příklady*. SOŠ a SOU, Lanškroun, 2007.

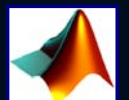
Freeware **Octave** (ne zcela plnohodnotný klon Matlabu) :

(Ale mluví česky !)

- http://www.orift.com/cs/instalace-gnu-octave-782/
- http://www.octave.cz/

Úvod do práce se systémem :

- MATLAB je **vysoce výkonný** jazyk určený pro technické výpočty.
- Umožňuje **rychlou práci** s maticemi, komplexními čísly a rozsáhlými datovými soubory.
- **Otevřená architektura** MATLABu obsahuje množství specializovaných knihoven funkcí a bloků (tzv. toolboxů) určené pro různé vědní a technické obory.
- **Podporuje** : matematické výpočty, vývoj algoritmů, měření a zpracování údajů, jejich analýzu a vizualizaci, modelování a simulaci, programování a vývoj aplikací.



Dobře míněné rady :

- **Nechtějte VŠECHNO pochopit**

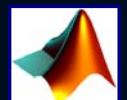
  (důležité pro zachování duševní rovnováhy)

  Matlab je tak rozsáhlý a složitý nástroj, že ho obsáhne v plném rozsahu (včetně všech Toolboxů) jen málokdo (nikoho takového neznám).

- **Trénink dělá mistra**

  Naučte se jen to, co potřebujete ke své práci, to ale používejte (ostatní stejně zapomenete) .

  Od toho jsou přece nápovědy, příručky a učebnice. Používejte je, budete se tak stále zdokonalovat v tom, co hlavně potřebujete.



Dobře míněné rady :

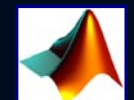
Přečtěte si 10x jak se má plavat a pak skočte do rybníka.
Utopíte se !

Ponaučení : chcete-li opravdu, abyste se naučili plavat, zkoušejte to ve vodě.

- **Začněte Matlab reálně používat**

- **Nenechte se odradit (počátečním) neúspěchem**

Dobře osvědčenou a často používanou metodou je ‘pokus-omyl’ (pozor - nepoužívat raději u zkoušky !).
Počítač je neobyčejně trpělivý, tolerantní a nic špatného si o vás nemyslí ani po desetkrát opakované elementární chybě (autorem mnohokrát experimentálně ověřeno).

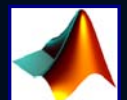


Důležitá poznámka :

Tento text NENÍ podrobným manuálem

Ambicí tohoto textu je být pouze pomocníkem pro první nesmělé krůčky Matlabem, být průvodcem po jeho nejběžnějších a nejpoužívanějších příkazech a možnostech.

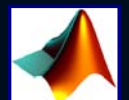
Podrobnosti hledejte v učebnicích, manuálech a nápovědách. Používejte je !
Užitečné je i pracovat se vzorovými aplikacemi (dema), která Matlab uživateli nabízí. Lze z nich i kopírovat celé sekvence příkazů.



Základní postulát :

VŠECHNO JE MATICE

- Má-li pouze jeden sloupec – je to **VEKTOR**.
- Má-li právě jeden sloupec a jeden řádek – je to **SKALÁR**.



Spuštění a vypnutí Matlabu :

SPUŠTĚNÍ :

- Dvojklik na ikonu 

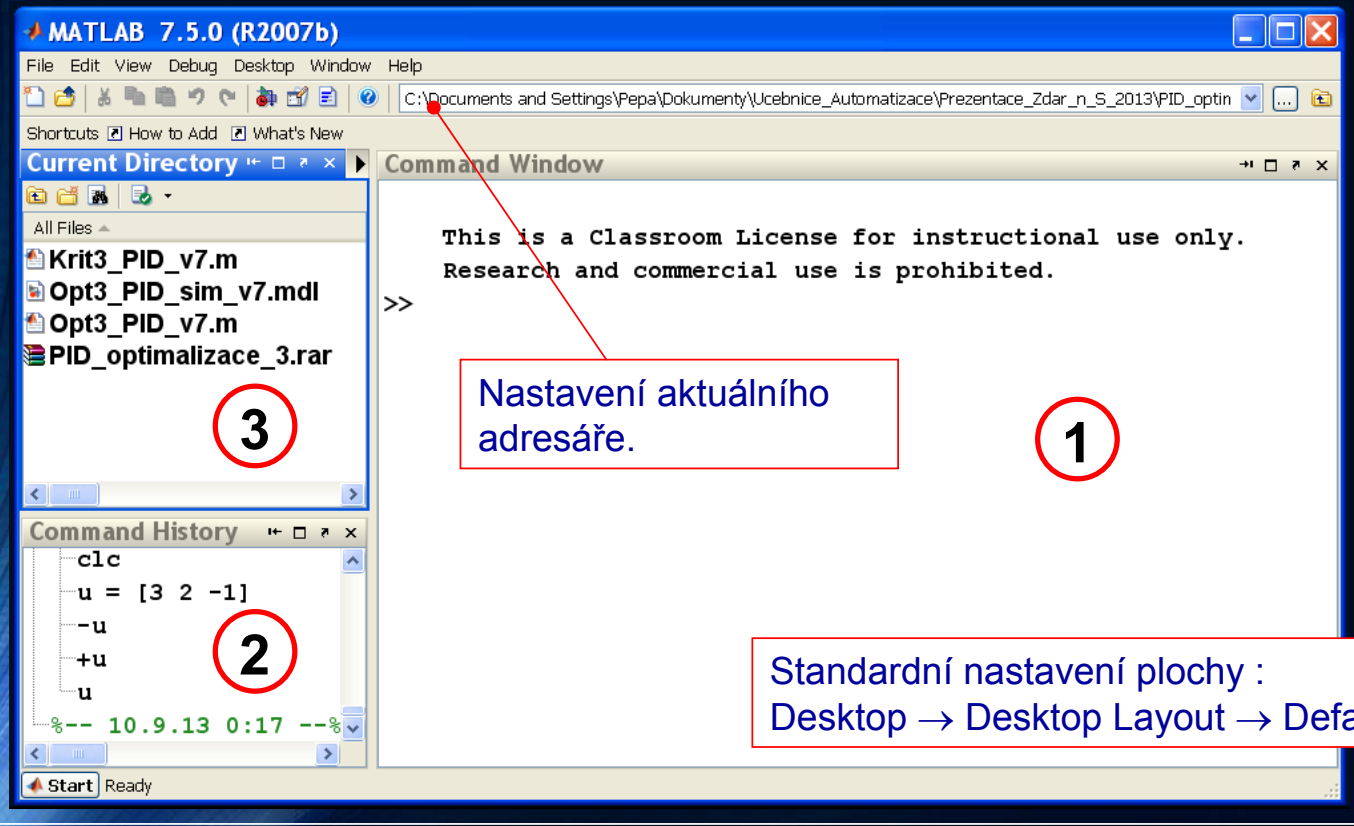
(ktrá se zpravidla nachází na pracovní ploše PC).
- Z operačního systému : Start → Programy →Matlab →Matlab.

VYPNUTÍ :

- Klik na ikonu  v záhlaví pracovní plochy Matlabu.
- Z nabídky File→Exit MATLAB.
- Stiskem kláves Ctrl+Q.

Pracovní plocha prostředí Matlab :

(Plocha se může poněkud lišit podle uživatelského nastavení a verze Matlabu.)



1. **Command Window :**
hlavní a nejdůležitější plocha, zapisují se zde příkazy, zobrazuje se odezva a systémová hlášení.

2. **Command History :**
zobrazují se zde všechny v minulosti zapsané příkazy, lze je znovu aktivovat.

3. **Current Directory :**
zobrazuje všechny soubory aktuálního adresáře.
Lze přepnout na okno Workspace, ve kterém se zobrazují hodnoty všech proměnných.

Práce v příkazovém okně (Command Window) :

Matlab jako kalkulačka

- Příkazy se ihned vykonávají (po stisknutí Enter).
- Základní operace : + , - , * , / , ^ (umocnění).
- Pokud není nazván jinak, výsledek se ukládá do proměnné ans (ANSwer).
- Je možno používat závorky (,) - kulaté, ostatní mají jiný syntaktický význam .

Zkus :

```
>> 3+2
ans =
     5
```

```
>> 3-2
ans =
     1
```

```
>> 3*2
ans =
     6
```

```
>> 3^2
ans =
     9
```

```
>> 3/2
ans =
    1.5000
```

```
>> (((3+2)/4)-5)^2)*2
ans =
   28.1250
```

Resp. :

```
>> ((3+2)/4-5)^2*2
ans =
   28.1250
```

platí obvyklá priorita operací ^ , * , / , + , -

ALE !

```
>> 2*1,5
ans =
     2
ans =
     5
```

V čem je problém ?

Práce v příkazovém okně (Command Window) :

Matlab jako kalkulačka - proměnné

- Desetinná čísla v Matlabu **POVINNĚ S TEČKOU !!!**

```
>> 2*1,5
ans =
     2
ans =
     5
```

ALE :

```
>> 2*1.5
ans =
     3
```

```
>> (((3+2)/4)-5)^2)*2
ans =
   28.1250
```

```
>> a=3+2
a =
     5
```

```
>> b=a/4
b =
    1.2500
```

```
>> c=(b-5)^2
c =
   14.0625
```

```
>> a_vysl=c*2
a_vysl =
   28.1250
```

- Výsledky matematických operací se mohou ukládat do proměnných.
- Proměnná musí : - začínat písmenem
  - mít max. 31 znaků anglické abecedy, číslice a podtržítko
  - nesmí obsahovat tečku, znaménko a jiné řídicí znaky
  - rozlišují se malá a velká písmena

  Typ proměnné se nemusí předem deklarovat.
- Proměnné se ukládají do *Workspace*, mohou se znovu použít nebo smazat

Práce v příkazovém okně (Command Window) :

Nejdůležitější triky a fígle

Možnosti, které nám usnadňují práci

- Napíšeme-li za příkaz středník, příkaz se provede, ale jeho výsledek se v příkazovém okně nezobrazí.
- Mezera v příkazu je nevýznamná.
- Na jednom příkazovém řádku může být více příkazů oddělených čárkou nebo středníkem.
- Pomocí kurzorových šipek ↑ a ↓ lze vybírat již provedené příkazy nebo je sledovat a vybírat v okně *Command History*.
- Příkazy jsou v textové formě a můžeme je přenášet pomocí Ctrl+C a Ctrl+V (třeba i z PC Wordu).
- Pokud je příkaz příliš dlouhý, lze ho rozdělit třemi tečkami i na další řádek.
- Znak % (procento) uvozuje poznámku, text až do konce řádky je ignorován.
- Obsah okna lze smazat pomocí příkazu clc, hodnoty proměnných se zachovávají.
- Příkaz who vypíše všechny proměnné, příkaz whos provede totéž s podrobnější informací.
- Mazání proměnné příkazem clear<proměnná>, příkaz clear all maže všechny proměnné.

Další možnosti a podrobnější informace viz učební text , help nebo manuál.

Práce v příkazovém okně (Command Window) :

Speciální a předdefinované proměnné

- **pi** ... Ludolfovo číslo π
- **realmax** ... největší možné kladné reálné číslo
- **realmin** ... nejmenší možné kladné reálné číslo
- **eps** ... přesnost výpočtu
- **i** nebo **j** ... imaginární složka komplexního čísla
- **ans** ... pomocná implicitní proměnná
- **Inf** ... nekonečno (infinity), např. výsledek 1/0
- **NaN** ... nedefinovaná hodnota (Not a Number), např. výsledek 0/0

```
>> pi
ans =
    3.1416
```

```
>> realmax
ans =
  1.7977e+308
```

```
>> eps
ans =
  2.2204e-016
```

```
>> realmin
ans =
  2.2251e-308
```

```
>> i
ans =
          0 + 1.0000i
```

```
>> j
ans =
          0 + 1.0000i
```

```
>> 1/0
ans =
   Inf
```

```
>> 0/0
ans =
   NaN
```

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Formát zobrazení čísel

MATLAB pracuje vždy ve dvojnásobné přesnosti, formát určuje jen tvar zobrazení čísel.

Mezery jsou zde významné !

- **format short** ... implicitní formát (4 desetinná místa)
- **format long** ... zobrazení na 15 desetinných míst
- **format short e** ... exponenciální tvar s krátkou mantisou
- **format long e** ... exponenciální tvar s dlouhou mantisou
- **format bank** ... zobrazení na dvě desetinná místa
- **format compact** ... potlačuje volný řádek v příkazovém okně

```
>> format short,pi
ans =
    3.1416
```

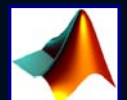
```
>> format long,pi
ans =
   3.141592653589793
```

```
>> format short e,pi
ans =
  3.1416e+000
```

```
>> format long e,pi
ans =
    3.141592653589793e+000
```

```
>> format bank,pi
ans =
          3.14
```

Další možnosti a podrobnější informace viz učební text, help nebo manuál.



Práce v příkazovém okně (Command Window) :

Funkce a operátory

MATLAB obsahuje velké množství funkcí a operátorů.

Přehled získáme :

- **help** ... úplný seznam a rozdělení do základních skupin
- **help elfun** ... přehled elementárních matematických funkcí
- **help specfun** ... přehled speciálních matematických funkcí
- **help general** ... přehled všeobecných příkazů
- **help ops** ... přehled všech typů operátorů
- **help arith** ... přehled aritmetických operátorů
- **help relop** ... přehled relačních operátorů

Používejte nápovědu !

Nesnažte se zvládnout najednou úplně všechny funkce MATLABU, jeho aplikační záběr je obrovský ! Začněte s nejjednoduššími a postupně si osvojujte i funkce, které budete potřebovat. Jinak vás široká nabídka zahltí !

Mezery jsou zde významné !

Argumenty funkcí jsou v kulatých závorkách za názvem funkce
např.: sin(x), sqrt(y1), …

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Matematické funkce (nejčastěji používané)

Goniometrické funkce :

- **sin** ... sinus, argument v radiánech
- **sind** ... sinus, argument ve stupních
- **asin** ... arcus sinus (inverzní sinus), výsledek v radiánech
- **asind** ... arcus sinus (inverzní sinus), výsledek ve stupních

atd.

Exponenciální funkce :

- **exp** ... exponenciální funkce e^x
- **log** ... přirozený logaritmus
- **log10** ... dekadický logaritmus
- **sqrt** ... druhá odmocnina

atd.

```
>> sin(pi/6)
ans =
    0.5000
```

```
>> sind(30)
ans =
    0.5000
```

```
>> asin(0.5)
ans =
    0.5236
```

```
>> asind(0.5)
ans =
   30.0000
```

```
>> y=exp(-0.5)
y =
    0.6065
```

```
>> x=log(y)
x =
   -0.5000
```

```
>> log10(1000)
ans =
     3
```

```
>> Alfa=pi/4; Obsah=3.4*(5.3*cos(Alfa))/2
Obsah =
    6.3710
```

Ostatní funkce : komplexní, pro zaokrouhlování, maticové, speciální, ...

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Chybová hlášení

Pokud je výraz zapsán chybně nebo není úplný dojde k chybovému hlášení.

Např. :

```
>> x 1=2/-0.5
??? Undefined function or method 'x' for input arguments of type 'char'.
```

Špatně zapsaná proměnná
(v názvu nesmí být mezera).

Správný zápis :

```
>> x1=2/-0.5
x1 =
     -4
```

nebo :

```
>> x1=2/-.5
x1 =
     -4
```

(pokud je před desetinnou tečkou nula,
může se vynechat).

Jiné chybové hlášení :

```
>> x1=2/.-5
??? x1=2/.-5
         |
Error: Unexpected MATLAB operator.
```

Špatně syntakticky zapsaný dělitel
(tečka předchází znaménko).

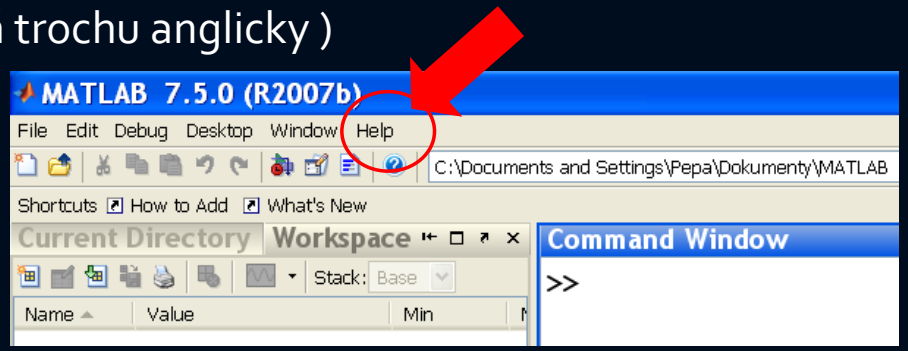
Ukázka označení místa chyby v chybovém hlášení.

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

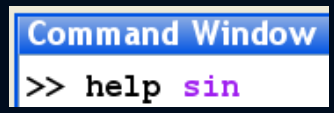
Nápověda

(autoři rozsáhlé nápovědy předpokládali, že umíte alespoň trochu anglicky)



Přístup k nápovědě :

- Hlavní okno Matlabu, základní menu v horní liště
  (rozkliknutím se dostaneme k menu na Internetu)



- Okno *Command Window* : - help <název funkce>
  - doc <název funkce> (podrobnější nápověda k funkci v okně *Help*)
  - helpwin (vyvolání okna *Help* pro podrobnou strukturovanou nápovědu)
  - help (vypsání dostupných položek nápovědy)
  - help help (podrobný popis použití nápovědy)

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Nápověda - ukázka konkrétní nápovědy

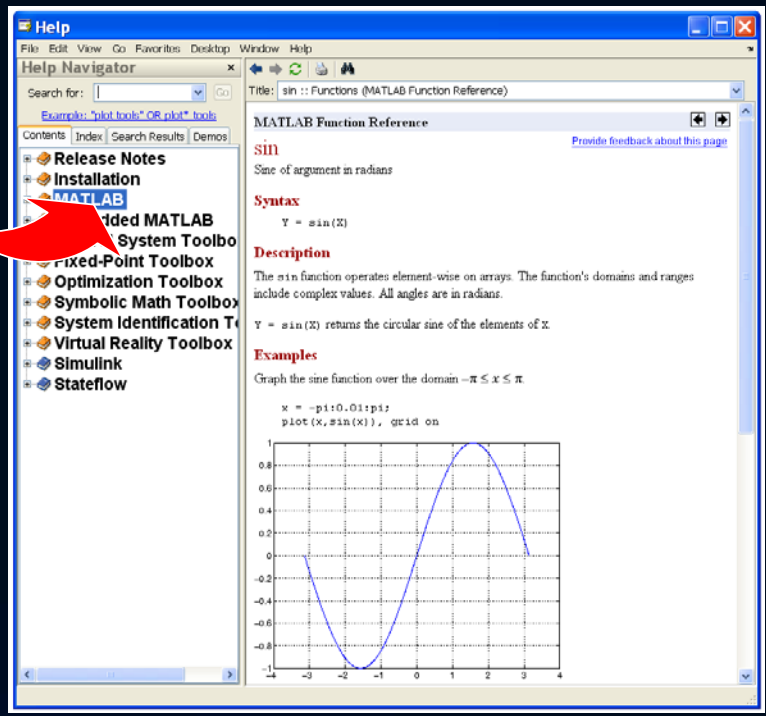
```
Command Window
>> help sin
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements of X.

    See also asin, sind.

    Overloaded methods:
       sym/sin

    Reference page in Help browser
       doc sin
```

Orientační nápověda
(užitečné je využití odkazů *See also*)

Nejčastější použití nápovědy

```
Command Window
>> doc sin
```

Podrobnější nápověda



Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Příklad 1

Vypočtěte hodnotu uvedeného výrazu :

$$(\sin\frac{\pi}{3}+2^4)^2-180\ \mathrm{tg}\frac{\pi}{8}$$

Řešení :

```
>> (sin(pi/3)+pow2(4))^2-180*tan(pi/8)
ans =
  209.9044
```

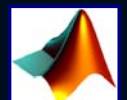
Použité příkazy :

sin, pow2, tan, format long

Výpočet s větší přesností :

```
>> format long, (sin(pi/3)+pow2(4))^2-180*tan(pi/8)
ans =
   2.099043716939449e+002
```

Přesněji : výpočet proběhl v obou případech se stejnou (dvojnásobnou) přesností.
Formát upravuje jen tvar výstupu, přesnost výpočtu neovlivňuje.



Práce v příkazovém okně (Command Window) :

Příklad 2

Potřebujeme koupit 12 rohlíků, 20 vajec a 5 jogurtů.
Rohlík stojí 1,50 Kč, vejce 2,70 Kč a jogurt 12,30 Kč.
Kolik zaplatíme za celý nákup ?

Řešení :

```
>> 12*1.50+20*2.70+5*12.30
ans =
  133.5000
```

nebo :

```
>> format bank, 12*1.50+20*2.70+5*12.30
ans =
        133.50
```

Za nákup tedy zaplatíme 133,50 Kč.

Stejný výpočet s použitím proměnných :

```
>> cena_rohlik=1.5; cena_vejce=2.7; cena_jogurt=12.3;
>> pocet_rohliku=12; pocet_vajec=20; pocet_jogurtu=5;
>> cena_nakupu=pocet_rohliku*cena_rohlik + ...
pocet_vajec*cena_vejce + pocet_jogurtu*cena_jogurt
cena_nakupu =
        133.50
```

Použité příkazy :

násobení, sčítání, format bank, potlačení výstupu středníkem, rozdělení příkazového řádku třemi tečkami

Výhodou tohoto řešení je, že se proměnné ukládají do *Workspace* a mohou se měnit nebo využít v jiném výpočtu.

Práce v příkazovém okně (Command Window) :

Příklad 3

1. Rozhodněme zda přímka y = 0,64 x–11,27 prochází bodem Q ≡ [21,13 ; 2,25] ?
2. Určete průsečík s osou x.

Řešení :

1.

```
>> k=0.64; q=-11.27; x=21.13; y=k*x+q
y =
    2.2532
```

Přímka bodem Q neprochází.

2.

```
>> y=0; x=(y-q)/k
x =
   17.6094
```

nebo přesněji :

```
>> format long, y=0; x=(y-q)/k
x =
  17.609375000000000
```

Přímka protíná osu x v bodě P ≡ [17,6094 ; 0].

Za povšimnutí stojí : použití proměnných, potlačení zobrazení středníkem, vliv formátu, zobrazení proměnných ve *Workspace*.

Práce v příkazovém okně (Command Window) :

Příklad 4

Rovníkový průměr zeměkoule je 12756,270 km.
Představme si, že zeměkoule je přesnou koulí a těsně kolem rovníku natáhneme ocelový drát. Prodloužíme ho pak o 1 metr a rovnoměrně po celém obvodu oddálíme.
Jak velká mezera vznikne ? Protáhne se jí myš ?

Řešení :

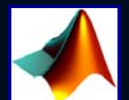
```
>> d=12756.27; Delta_d=1.0;                          %d [km], Delta_d [m]
>> Mezera=((pi*d*10^5+Delta_d*10^2)/pi-d*10^5)/2 %Pocitame v centimetrech
Mezera =
   15.9155
```

(Jedno z možných řešení, nikoliv nejjelegantnější, snad logicky srozumitelné .)

Poměrně známý paradox. Vznikne neuvěřitelně velká mezera ca 16 cm.
Myška samozřejmě pohodlně proleze.

Za povšimnutí stojí :

použití pí, mocniny, priority algebraických operací, závorek a poznámek.



Práce v příkazovém okně (Command Window) :

Práce s komplexními čísly

Zadání komplexního čísla – pomocí symbolů **i** nebo **j** (oba jsou ekvivalentní).

POZOR na používání obou symbolů ! Často se používají jako index v cyklech. Může dojít k jejich přepsání a chybné funkci programu.

Nejčastěji používané funkce :

- **real** ... reálná část čísla
- **imag** ... imaginární část
- **abs** ... absolutní hodnota (tzv.modul)
- **angle** ... fázový úhel (tzv.argument) v rad
- **conj** ... komplexně sdružené číslo

aj.

```
>> k1=2+0.5i
k1 =
   2.0000 + 0.5000i
>> real(k1)
ans =
     2
>> imag(k1)
ans =
    0.5000
```

```
>> k2=2+0.5j
k2 =
   2.0000 + 0.5000i
>> abs(k2)
ans =
    2.0616
>> angle(k2)
ans =
    0.2450
```

```
>> k3=5-4i
k3 =
   5.0000 - 4.0000i
>> conj(k3)
ans =
   5.0000 + 4.0000i
```

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

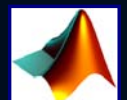
Práce s vektory a maticemi

Vektorem zpravidla chápeme matici pouze s jedním sloupcem.
Matici pouze s jedním řádkem zpravidla označujeme jako transponovaný vektor, příp. řádkový vektor.
Pokud má matice stejný počet řádků a sloupců, jedná se o matici čtvercovou.

Doporučení :

- **matice** ... značit velkými písmeny, např.: **A, B, M, Q,** ...
- **vektory** ... značit malými písmeny, např.: **a, b, v, r,** ...

Není to nutné (syntaxe Matlabu to nevyžaduje), je to ale obvyklé v matematice, zlepšuje to orientaci v zápisu.



Práce v příkazovém okně (Command Window) :

Práce s vektory a maticemi

Zadávání matic :

- Prvky matice se zadávají do hranatých závorek.
- Matice se zapisují po řádcích, které jsou odděleny středníkem (nebo novým řádkem).
- Prvky v řádku jsou odděleny mezerou (nebo čárkou).
- Prvky matice mohou být čísla, proměnné nebo libovolné výrazy.
- Počet řádků a sloupců nemusí být stejný.

```
>> M1=[1 2 3;4 5 6]
M1 =
     1     2     3
     4     5     6
```

```
>> M2=[log10(100) exp(1);-5*2 sin(pi/2)]
M2 =
    2.0000    2.7183
  -10.0000    1.0000
```

```
>> v1=[7;8;9]
v1 =
     7
     8
     9
```

```
>> v2=[10 11 12 13]
v2 =
    10    11    12    13
```

Za povšimnutí stojí : zadání matice M2.

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Prvky matic nemusejí být skalární

Příklad :

```
>> M1=[1 2 3;4 5 6]
M1 =
     1     2     3
     4     5     6
```

```
>> s=[1;4]
s =
     1
     4
```

```
>> r1=[2 3]
r1 =
     2     3
```

```
>> r2=[5 6]
r2 =
     5     6
```

```
>> M1=[s [r1;r2]]
M1 =
     1     2     3
     4     5     6
```

ale :

```
>> M1=[s;r1;r2]
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

Nesouhlasí dimenze (počty řádků / sloupců) jednotlivých submatic .

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Práce s vektory a maticemi : tvorba řádkových vektorů (tzv. polí)

Buď jako normální matici do hranatých závorek, ale pouze s jedním řádkem nebo s výhodou (zejména u rozsáhlých polí) :

Často používané !

- Dvojtečková konverze x = a : b
  vytvoří pole x od a (včetně) do b s krokem 1 (poslední prvek ≤ b) .
- Dvojtečková konverze x = a : k : b
  vytvoří pole x od a (včetně) do b s krokem k (poslední prvek ≤ b) .
- Příkazem linspace x = linspace(a, b, n)
  vytvoří pole x od a do b (včetně) s n prvky (lineárně rozloženými).
- Příkazem logspace x = logspace(a, b, n)
  vytvoří pole x od 10^a do 10^b (včetně) s n prvky (logaritmicky rozloženými).

```
>> x=0:pi
x =
     0     1     2     3
```

```
>> x=1:0.7:pi
x =
    1.0000    1.7000    2.4000    3.1000
```

```
>> x=linspace(-1,pi,4)
x =
   -1.0000    0.3805    1.7611    3.1416
```

```
>> x=logspace(0,2,4)
x =
    1.0000    4.6416   21.5443  100.0000
```

Povšimněte si posledních prvků polí.

Další možnosti a podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Řádkové vektory - příklady

1. a. Generujte pole (řádkový vektor) prvních pěti přirozených čísel.
b. Generujte podobné pole, ale s pěti tisíci čísly.

Řešení : a.

```
>> p1=1:5
p1 =
     1     2     3     4     5
```

b.

```
>> p2=1:5000;
```

Středník potlačuje zobrazení dlouhého pole. Všimněte si protrokolu ve *Workspace*.

Nezapomínejte na středník. Pokud bychom ho zde neuvedli, vypsalo by se do okna všech 5000 hodnot !

2. Generujte pole prvních pěti sudých čísel (počínaje nulou).

Řešení :

```
>> s=0:2:8
s =
     0     2     4     6     8
```

resp.:

```
>> s=linspace(0,8,5)
s =
     0     2     4     6     8
```

3. Generujte a uložte do pole tisíc hodnot funkce y = sin (x) pro $x \in < 0 , 2\pi >$.

Řešení :

```
>> x=linspace(0,2*pi,1000);
>> y=sin(x);
```

resp.:

```
>> y=sin(linspace(0,2*pi,1000));
```

Pro jinou funkci.

```
>> y1=sin(x)+cos(2*x-pi/2);
```

Argumentem funkce může být i jiná funkce, viz oba příklady.

Práce v příkazovém okně (Command Window) :

Speciální matice

(Praktické, šetří práci hlavně při velkých rozměrech.)

Nejčastěji prakticky používané :

- Nulová matice N = zeros(r, s)
  vytvoří čtvercovou matici s r řádky a s sloupci samých nul.
- Nulová čtvercová matice N = zeros(r)
  vytvoří čtvercovou matici s r řádky a r sloupci samých nul.
- Jednotková matice E = eye(r)
  vytvoří jednotkovou matici rozměru r (čtvercovou matici s r řádky a r sloupci, s jedničkami na hlavní diagonále, ostatní prvky nulové).
- Matice samých jedniček J = ones(r, s) resp. J = ones(r)
  vytvoří matici jedniček rozměru (r, s) resp. (r, r).
- Matice náhodných prvků R = rand(r, s) resp. R = rand(r)
  vytvoří matici náhodných čísel s rovnoměrným rozložením z intervalu (0, 1) rozměru (r, s) resp. (r, r).

```
>> N1=zeros(2,3)
N1 =
     0     0     0
     0     0     0
```

```
>> N2=zeros(2)
N2 =
     0     0
     0     0
```

```
>> E=eye(3)
E =
     1     0     0
     0     1     0
     0     0     1
```

```
>> J1=ones(2,3)
J1 =
     1     1     1
     1     1     1
```

```
>> J2=ones(2)
J2 =
     1     1
     1     1
```

```
>> R1=rand(2,3)
R1 =
    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975
```

```
>> R2=rand(2)
R2 =
    0.2785    0.9575
    0.5469    0.9649
```

Matlab nabízí další možnosti a varianty. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Maticové operace

Základní operace :

- **C = A + s** ... ke každému prvku matice **A** je přičten skalár **s**
- **D = A * s** ... každý prvek matice **A** je vynásoben skalárem **s**
- **E = A / s** ... každý prvek matice **A** je vydělen skalárem **s**
- **F = A + B** ... součet stejnolehlých prvků matic **A** a **B**
- **G = A - B** ... rozdíl stejnolehlých prvků matic **A** a **B**
- **H = A * B** ... klasické násobení matic **A** a **B**
- **I = A . * B** ... násobení stejnolehlých prvků matic **A** a **B**
- **J = A . / B** ... dělení stejnolehlých prvků matic **A** a **B**
- **K = A . \ B** ... dělení stejnolehlých prvků matic **B** a **A**
- **L = A / B** ... pravé dělení matic (**L = A B^{-1}**)
- **M = A \ B** ... levé dělení matic (**M = A^{-1} B**)
- **N = A ^ n** ... n-tá mocnina matice **A** (pouze čtvercové)
- **P = A . ^ n** ... n-tá mocnina každého prvku matice **A**

```
>> A=[1 2;3 4];
>> B=[1 1;-2 2]; A,B
A =
     1     2
     3     4
B =
     1     1
    -2     2
```

(Zkus i jiné operace.)

```
>> Q1=A*2.5
Q1 =
    2.5000    5.0000
    7.5000   10.0000
```

```
>> Q2=A-B
Q2 =
     0     1
     5     2
```

```
>> Q3=A*B
Q3 =
    -3     5
    -5    11
```

```
>> Q4=A.*B
Q4 =
     1     2
    -6     8
```

```
>> Q5=A/B
Q5 =
    1.5000    0.2500
    3.5000    0.2500
```

```
>> Q6=A^2
Q6 =
     7    10
    15    22
```

n nemusí být celé

např. n=0,5 … odmocnina

Matlab nabízí další možnosti a varianty. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Maticové funkce

Základní a často používané funkce :

- **B = A'** … (apostrof) transpozice matice **A** – záměna řádků a sloupců, **B** = $\mathbf{A}^T$
- **C = inv(A)** … inverze matice **A** (pouze pro čtvercovou regulární matici), **C** = A^{-1}
- **d = det(A)** … determinant matice **A** (pouze pro čtvercovou matici)
- **v = eig(A)** … vrací vektor vlastních čísel matice **A** (pouze pro čtvercovou matici)
- **v1 = size(A)** … vrací řádkový vektor, první prvek = počet řádek
  druhý prvek = počet sloupců matice **A**

```
>> A=[1 2;3 4]
A =
     1     2
     3     4
```

```
>> B=A'
B =
     1     3
     2     4
```

```
>> C=inv(A)
C =
   -2.0000    1.0000
    1.5000   -0.5000
```

```
>> E=A*inv(A)
E =
    1.0000         0
    0.0000    1.0000
```

```
>> E=inv(A)*A
E =
    1.0000         0
    0.0000    1.0000
```

```
>> d=det(A)
d =
    -2
```

```
>> v=eig(A)
v =
   -0.3723
    5.3723
```

```
>> v1=size([1 2 3;4 5 6])
v1 =
     2     3
```

Všimněte si : $A*A^{-1} = A^{-1}*A = E$

E … jednotková matice

Násobení matic však obecně komutativní NENÍ !!!

Matlab nabízí řadu dalších speciálních funkcí. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Výběr prvků matice

Nejčastěji používané :

- **M(r, s)** … **r** – index řádku, **s** – index sloupce
  (**r** může být i sloupcový vektor a **s** vektor řádkový zvolených indexů)
- **M(r, :)** … dvojtečková konverze značí výběr všech sloupců
- **M(:, s)** … dvojtečková konverze značí výběr všech řádků

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

```
>> M(2,3)
ans =
     6
```

```
>> M(2,:)
ans =
     4     5     6
```

```
>> M(:,3)
ans =
     3
     6
     9
```

```
>> M([1;3],2)
ans =
     2
     8
```

```
>> M([1;3],[1 2])
ans =
     1     2
     7     8
```

Všimněte si užití dvojtečkové konverze – často se používá.

Matlab nabízí řadu dalších možností. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Matice - užitečné, často používané funkce

Často používané :

- **M1 = fliplr (M)** ... přerovnání sloupců matice **M** (v obráceném pořadí)
- **M2 = flipud (M)** ... přerovnání řádků matice **M** (v obráceném pořadí)
- **n = lenght (v)** ... vrací počet prvků vektoru **v**
- **n = lenght (M)** ... vrací maximum z počtu řádků a počtu sloupců matice **M**

```
>> M=[1 2 3;4 5 6]
M =
     1     2     3
     4     5     6
```

```
>> M1=fliplr(M)
M1 =
     3     2     1
     6     5     4
```

```
>> M2=flipud(M)
M2 =
     4     5     6
     1     2     3
```

```
>> v1=M(1,:),v2=M(:,1)
v1 =
     1     2     3
v2 =
     1
     4
```

```
>> n=length(M)
n =
     3
```

```
>> n=length(v1)
n =
     3
>> n=length(v2)
n =
     2
```

Matlab nabízí řadu dalších možností. Podrobnější informace viz učební text, help nebo manuál.

Maticový počet - příklady použití

Příklad 1

Nalezněme součet kvadrátů prvních pěti přirozených čísel.

Řešení : Způsobů řešení zadané úlohy bývá zpravidla více. Například jedno z nich :

```
>> x=1^2+2^2+3^2+4^2+5^2
x =
    55
```

Řešení je sice správné, ale jeho "nešikovnost" by nás jistě napadla, kdybychom měli pracovat nikoliv pouze s pěti, ale např. se stem nebo tisícem prvků.

Princip řešení :

$$x = p^T p = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$$

Skalární součin = (Euklidova norma)^2.

Jiné možné řešení :

```
>> p=[1;2;3;4;5]
p =
     1
     2
     3
     4
     5
```

Resp.:

```
>> p=[1:5]'
```

Resp.:

```
>> p=rot90([1:5])
```

```
>> x=p'*p
x =
    55
```

Nebo rovnou „z jedné vody na čisto" :

```
>> p=rot90([1:5]); x=p'*p
x =
    55
```

Za povšimnutí stojí použití funkce **rot90** (ještě použita nebyla - zkus help).

Maticový počet - příklady použití

Příklad 2

Na louce byly slepice a krávy. Měly dohromady 60 hlav a 140 nohou. Kolik bylo kterých?

Řešení :

s … počet slepic
k … počet krav

$$\left.\begin{array}{rcl} s+k & = & 60 \\ 2s+4k & = & 140 \end{array}\right\} \rightarrow \underbrace{\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} s \\ k \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 60 \\ 140 \end{bmatrix}}_{\mathbf{b}} \rightarrow \mathbf{A}\cdot\mathbf{x} = \mathbf{b} \rightarrow \boxed{\mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}}$$

Maticový tvar vyjádření soustavy lineárních algebraických rovnic.

```
>> A=[1 1;2 4]; b=[60;140]; x=inv(A)*b
x =
    50
    10
```

Resp.:

```
>> x=A\b
x =
    50
    10
```

Srovnej také :

```
>> inv(A)
ans =
    2.0000   -0.5000
   -1.0000    0.5000
```

```
>> A^-1
ans =
    2.0000   -0.5000
   -1.0000    0.5000
```

Na louce bylo **50** slepic a **10** krav.

Maticový počet - příklady použití

Příklad 3

Babička poslala svým třem vnukům koš ořechů. Měli se rozdělit v poměru věku všech tří. Když je rozdělili, zjistili, že Karel má tolik jako Tomáš a ještě třetinu toho, co má Michal. Michal má tolik, co Karel a ještě třetinu Tomášova podílu. Tomáš má 20 ořechů a třetinu Karlova podílu. Spočtěte:

a) kolik bylo v koši ořechů, b) kolik má každý z chlapců,
c) kolik jim je let.

Řešení :

k … počet ořechů Karel
t … počet ořechů Tomáš
m … počet ořechů Michal

$$\left.\begin{array}{rcl} k &=& t + m/3 \\ m &=& k + t/3 \\ t &=& 20 + k/3 \end{array}\right\} \rightarrow \left\{\begin{array}{rcl} 3k - 3t - m &=& 0 \\ -3k - t + 3m &=& 0 \\ -k + 3t &=& 60 \end{array}\right.$$

$$\underbrace{\begin{bmatrix} 3 & -3 & -1 \\ -3 & -1 & 3 \\ -1 & 3 & 0 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} k \\ t \\ m \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 60 \end{bmatrix}}_{\mathbf{b}} \rightarrow \mathbf{A}\cdot\mathbf{x} = \mathbf{b} \rightarrow \mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}$$

```
>> A=[3 -3 -1;-3 -1 3;-1 3 0];
>> b=[0;0;60];
>> x=inv(A)*b
x =
   75.0000
   45.0000
   90.0000
```

```
>> k=x(1),t=x(2),m=x(3)
k =
   75.0000
t =
   45.0000
m =
   90.0000
```

```
>> Pocet_orechu=sum(x)
Pocet_orechu =
  210.0000
```

a) V koši bylo celkem 210 ořechů.
b) Karel má 75, Tomáš 45 a Michal 90 ořechů.

Všimněte si použití funkce **sum** (ještě použita nebyla - zkus help).

Maticový počet - příklady použití

Příklad 3 - pokračování

Všimněte si použití funkce **gcd**
(ještě použita nebyla - zkus help).

Bratři se rozdělili o ořechy od babičky v poměru jejich věku, Karel má 75, Tomáš 45 a Michal 90 ořechů . Úkolem je určit jejich věk.

```
>> k=xi(1),t=xi(2),m=xi(3)
k =
     75
t =
     45
m =
     90
```

k … počet ořechů Karel
t … počet ořechů Tomáš
m … počet ořechů Michal

Největší společný dělitel všech tří položek :

```
>> msd=gcd(75,gcd(45,90))
msd =
     15
```

```
>> r=xi/msd;rk=r(1),rt=r(2),rm=r(3)
rk =
      5
rt =
      3
rm =
      6
```

Věk jednotlivých bratrů.

```
>> xi=round(x);r=xi/gcd(xi(1),gcd(xi(2),xi(3)));
>> rk=r(1),rt=r(2),rm=r(3)
rk =
      5
rt =
      3
rm =
      6
```

Výpočet rovnou „z jedné vody na čisto" .

Karel má 5, Tomáš 3 a Michal 6 let.

Výsledek není jednoznačný – každý celočíselný násobek této trojice vyhovuje zadání , např.: (10, 6, 12), (15, 9, 18), atd.

Práce v příkazovém okně (Command Window) :

Soustava lineárních rovnic

Jednou z nejčastějších úloh lineární algebry je řešení soustav lineárních algebraických rovnic. Úlohu lze řešit různými metodami, Matlab poskytuje univerzální nástroj s možností výběru optimální numerické metody vzhledem k zadaným parametrům (rozsáhlé, řídké, špatně podmíněné a speciální tvary matic).

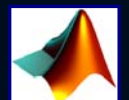
x = linsolve (A, b) ... **x** je vektor řešení, **A** matice soustavy , **b** vektor pravých stran.
Příkaz může mít i třetí parametr, je však nepovinný
(pro jednoduchost zde není uveden).
Má význam jen ve speciálních případech, zpřesňuje a zrychluje výpočet.
Podrobnosti viz učební text, help linsolve nebo manuál.

Příklad :

```
>> A=[1 1;2 4];b=[60;140];x=linsolve(A,b)
x =
    50
    10
```

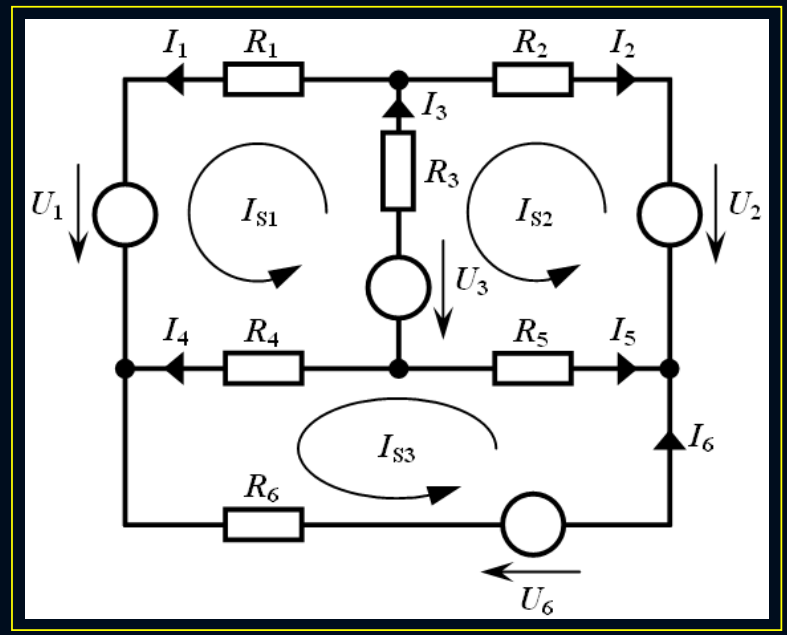
Srovnej s příkladem 2 předcházejícího textu.

(Funkce s potlačeným třetím parametrem používá defaultně metody LU rozkladu s problémy u špatně podmíněných matic.)



Maticový počet - příklady použití

Příklad 4



Analýza elektrického obvodu :

$R_1 = 4\ \Omega$, $R_2 = 20\ \Omega$, $R_3 = 8\ \Omega$, $R_4 = 15\ \Omega$, $R_5 = 5\ \Omega$, $R_6 = 10\ \Omega$,
$U_1 = 6$ V, $U_2 = 3$ V, $U_6 = 5$ V

Úkolem je určit :

- proudy tekoucí jednotlivými rezistory
- elektrická napětí na jednotlivých rezistorech

Řešení : Metodou smyčkových proudů sestavíme rovnice :

Příklad aplikace 2. Kirchhoffova zákona.

$$R_1 I_{S1} + U_1 + R_4 (I_{S1} - I_{S3}) - U_3 + R_3 (I_{S1} - I_{S2}) = 0$$
$$R_2 I_{S2} + R_3 (I_{S2} - I_{S1}) + U_3 + R_5 (I_{S2} - I_{S3}) - U_2 = 0$$
$$R_4 (I_{S3} - I_{S1}) + R_6 I_{S3} - U_6 + R_5 (I_{S3} - I_{S2}) = 0$$

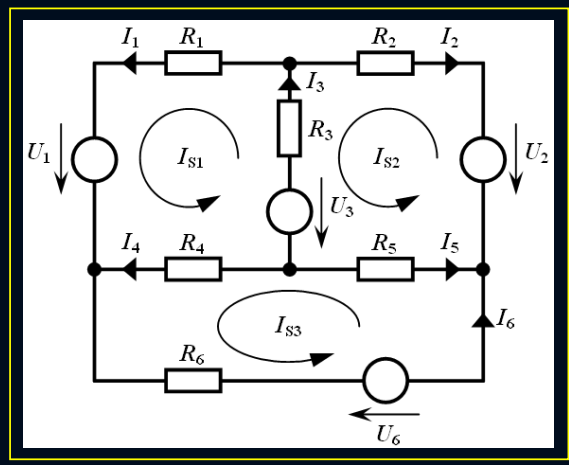
Maticový počet - příklady použití

Příklad 4 - pokračování

$$\begin{bmatrix} R_1+R_3+R_4 & -R_3 & -R_4 \\ -R_3 & R_2+R_3+R_5 & -R_5 \\ -R_4 & -R_5 & R_4+R_5+R_6 \end{bmatrix} \cdot \begin{bmatrix} I_{S1} \\ I_{S2} \\ I_{S3} \end{bmatrix} = \begin{bmatrix} -U_1+U_3 \\ U_2-U_3 \\ U_6 \end{bmatrix}$$

$$\mathbf{A} \qquad \mathbf{I_S} \qquad \mathbf{b}$$

$$\mathbf{A} \cdot \mathbf{I_S} = \mathbf{b} \quad \rightarrow \quad \boxed{\mathbf{I_S} = \mathbf{A}^{-1}\mathbf{b}}$$



$R_1 = 4\ \Omega$,
$R_2 = 20\ \Omega$,
$R_3 = 8\ \Omega$,
$R_4 = 15\ \Omega$,
$R_5 = 5\ \Omega$,
$R_6 = 10\ \Omega$,
$U_1 = 6$ V,
$U_2 = 3$ V,
$U_6 = 5$ V

Řešení :

```
>> R1=4; R2=20; R3=8; R4=15; R5=5; R6=10;
>> U1=6; U2=3; U3=12; U6=5;
>> A=[R1+R3+R4 -R3 -R4;-R3 R2+R3+R5 -R5;...
-R4 -R5 R4+R5+R6];
>> b=[-U1+U3;U2-U3;U6];
```

```
>> Is=inv(A)*b; Is'
ans =    IS1       IS2       IS3
    0.3630   -0.1354    0.3256
```

Smysl I_{S2} byl odhadnutý obráceně (znaménko).

Resp.:

```
>> Is=linsolve(A,b)
```

```
>> I=[Is(1),-Is(2),Is(1)-Is(2),...
Is(3)-Is(1),Is(2)-Is(3),Is(3)]
I =    I1        I2        I3        I4        I5        I6
    0.3630    0.1354    0.4984   -0.0374   -0.4610    0.3256
```

Fyzikální rozměr všech proudů [A].

```
>> U=[I(1)*R1,I(2)*R2,I(3)*R3,...
I(4)*R4,I(5)*R5,I(6)*R6]
U =    U1        U2        U3        U4        U5        U6
    1.4520    2.7079    3.9871   -0.5609   -2.3050    3.2560
```

Fyzikální rozměr všech napětí [V].

Práce v příkazovém okně (Command Window) :

Práce s polynomy

Polynomy jsou v Matlabu stadardně definovány pomocí polí (řádkových vektorů), která obsahují jejich koeficienty v sestupném pořadí od nejvyšší mocniny proměnné k nejnižší.

Příklad :

$p_1(x) = 3x^3 - 6x^2 + 5x - 1$

```
>> p1=[3 -6 5 -1]
p1 =
     3    -6     5    -1
```

$p_2(x) = x^2 - 2x$

```
>> p2=[1 -2 0]
p2 =
     1    -2     0
```

- **roots (p)** ... výpočet kořenů polynomu

```
>> roots(p1)
ans =
   0.8590 + 0.6666i
   0.8590 - 0.6666i
   0.2819
```

```
>> roots(p2)
ans =
     0
     2
```

Koeficienty polynomů mohou být i desetinná čísla (dokonce i komplexní).

Práce v příkazovém okně (Command Window) :

Práce s polynomy

- **polyval (p, x)** ... výpočet funkční hodnoty polynomu **p** pro zadanou hodnotu **x**

Příklad :

$p_1(x) = 3x^3 - 6x^2 + 5x - 1$

```
>> p1=[3 -6 5 -1]
p1 =
     3     -6      5     -1
```

```
>> y1=polyval(p1,-1)
y1 =
   -15
```

$y_1 = p_1(-1) = 3(-1)^3 - 6(-1)^2 + 5(-1) - 1 = -15$

$p_2(x) = x^2 - 2x$

```
>> p2=[1 -2 0]
p2 =
     1     -2      0
```

```
>> y2=polyval(p2,0.1)
y2 =
   -0.1900
```

$y_2 = p_2(0,1) = (0,1)^2 - 2(0,1) = -0,19$

Práce v příkazovém okně (Command Window) :

Práce s polynomy

- **p3 = conv (p1, p2)** ... násobení polynomu **p1** polynomem **p2**
- **[q, r] = deconv (p1, p2)** ... dělení polynomu **p1** polynomem **p2**
  (**q** … kvocient, **r** … zbytek)

Příklad :

$p_1(x) = 3x^3 - 6x^2 + 5x - 1$

$p_2(x) = x^2 - 2x$

```
>> p3=conv([3 -6 5 -1],[1 -2 0])
p3 =
     3    -12     17    -11      2      0
>> [q,r]=deconv([3 -6 5 -1],[1 -2 0])
q =
     3     0
r =
     0     0     5    -1
```

$$p_3 = (3x^3 - 6x^2 + 5x - 1)\cdot(x^2 - 2x) = 3x^5 - 12x^4 + 17x^3 - 11x^2 + 2x$$

$$(3x^3 - 6x^2 + 5x - 1) : (x^2 - 2x) = 3x$$
$$-\underline{3x^3 + 6x^2}$$
$$0 + 5x - 1$$

$$\frac{3x^3 - 6x^2 + 5x - 1}{x^2 - 2x} = \underbrace{3x}_{q} + \underbrace{\frac{5x - 1}{x^2 - 2x}}_{r}$$

Matlab nabízí řadu dalších možností. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Práce s textovými řetězci

Textový řetězec se zapisuje v Matlabu jako řada ASCII znaků uzavřený do apostrofů.

Nedoporučuje se používat diakritiku (nefunguje ve starších verzích).

Přístup k jednotlivým znakům je možný **pomocí indexů** jako u matic a vektorů.

Příklad :

```
>> s='Toto je textovy retezec'
s =
Toto je textovy retezec
>> length(s)
ans =
    23
>> s(17) %vyber 17.prvku retezce
ans =
r
>> s(17:end) %vyber od 17.prvku do konce retezce
ans =
retezec
```

- zadání textového řetězce
- délka řetězce (počet znaků včetně mezer)
- výběr prvku podle indexu
- výběr části řetězce

Všimněte si použití kovnerze **end** (zatím použita nebyla - viz help).

Matlab nabízí řadu dalších možností. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Práce s textovými řetězci

Textové řetězce lze ukládat **do polí** (podobně jako matice) .

- **S = char (s_1, s_2, …, s_n)** … vytvoří matici,
  její řádky jsou jednotlivé řetězce s_1, s_2, …, s_n

Příklad :

```
>> S=char('pondeli','utery','streda')
S =
pondeli
utery
streda
```

```
>> S(2,:)
ans =
utery
```

Všimněte si uložení textového řetězce také ve *Workspace*.

```
>> size(S)
ans =
     3     7
```

Textové pole má 3 řádky (pondeli, utery, streda),
a 7 sloupců (podle počtu znaků nejdelšího – pondeli),
neobsazené pozice zbylých řádků jsou vyplněny prázdným znakem (ASCII 32) .

Matlab nabízí řadu dalších možností. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Práce s textovými řetězci

Převod mezi textovým řetězcem a číselnou proměnnou :

- **x = str2num (s)** ... převod textu **s** na číslo **x**
- **s = num2str (x)** ... převod čísla **x** na text **s**

Příklady :

```
>> text='1.2345'
text =
1.2345
>> cislo=str2num(text)
cislo =
    1.2345
```

desetinná tečka je povinná
(zkuste a pochopte jak se to chová s čárkou)

```
>> pi
ans =
    3.1416
>> pi_retezec=num2str(pi)
pi_retezec =
3.1416
```

```
>> A=[1 2;3 4.5]; size(A) %A je matice 2x2
ans =
     2     2
>> B=num2str(A); size(B) %B je retezec 2x13
ans =
     2    13
```

```
>> r=3.5; p=pi*r^2;
>> s=['Kruh o polomeru r =',num2str(r),' ma obsah p =',num2str(p)]
s =
Kruh o polomeru r =3.5 ma obsah p =38.4845
```

Matlab nabízí řadu dalších možností. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

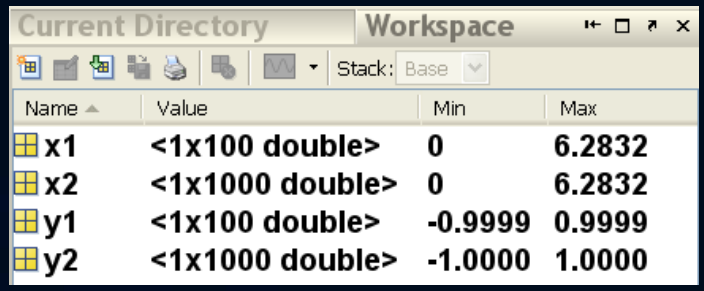
Práce s daty - odstranění proměnných z *Workspace*

- **clear** ... smaže <u>všechny</u> proměnné
- **clear p1 p2** ... smaže <u>jen uvedené</u> proměnné, zde **p1** a **p2**
  (výčet může být samozřejmě i delší)

POZOR !
Smazané proměnné nelze obnovit, pokud nebyly předem uloženy na HD.

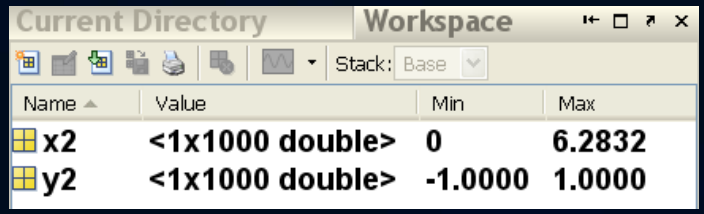
Příklad :

```
>> x1=linspace(0,2*pi,100);
>> x2=linspace(0,2*pi,1000);
>> y1=sin(x1);
>> y2=sin(x2);
```



Výpočet pořadnic funkce y = sin (x) na intervalu x $\in$< 0, 2*π > ve 100 a 1000 bodech.

```
>> clear x1 y1
```



Vymazání x1 a y1 pro 100 bodů.

- **Totéž přímo ve *Workspace* :** označit proměnnou + Del .

Práce v příkazovém okně (Command Window) :

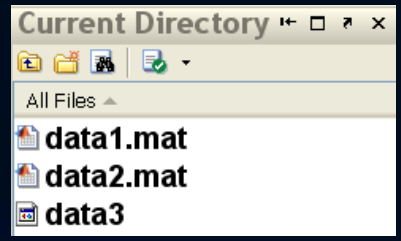
Práce s daty - ukládání proměnných do souboru

- **Do aktuálního adresáře** (*Current Directory*) :
  - **save** ... uloží všechny proměnné do souboru „matlab.mat"
  - **save nazev** ... do souboru „nazev.mat"
  - **save nazev p1 p2** ... uloží vybrané proměnné (zde **p1** a **p2**), příp. všechny uvedené do souboru „nazev.mat"
  - **save nazev p1 p2 -ascii** ... uloží vybrané proměnné (zde **p1** a **p2**), příp. všechny uvedené do souboru „nazev" (bez přípony v textové podobě)

(soubory *.mat jsou ve spec.formátu Matlabu)

Příklad :

```
>> A=[1 2 3;4 5 6;7 8 1]; b=[6; 6; 6];
>> save data1;              % ulozi vse do data1
>> save data2 A;            % ulozi A do data2
>> save data3 b -ascii;     % ulozi do textoveho data3 hodnoty b
```



- **Do jiného adresáře :** název souboru uvést vč. cesty, např.:

```
>> save 'G:\Data\data4' A b
```

Cestu vč.názvu uvést mezi apostrofy !

Práce v příkazovém okně (Command Window) :

Práce s daty - načtení proměnných ze souboru

Opačný postup než při ukládání, tentokrát pomocí příkazu **load** .

Soubor typu :

- ***.mat** ... uloží se do *Workspace* všechny proměnné i se svými původními názvy
- ***.txt** ... uloží se jen pod názvem souboru bez rozlišení proměnných

(Soubor *.txt nebo soubor bez přípony textového typu.)

Příklady :

```
>> clear all; load data1; who  % nacteni vsech dat

Your variables are:

A  b
```

```
>> clear all; load data2; who  % nacteni matice A

Your variables are:

A
```

```
>> clear all; load data3; who  % nacteni dat ASCII

Your variables are:

data3
```

```
>> data3
data3 =
     6
     6
     6
```

```
>> clear all; load 'G:\Data\data4'; who  % nacteni z jineho adresare

Your variables are:

A  b
```

Všimněte si a pochopte použití příkazů „who“ a „clear all“.

Matlab nabízí i další možnosti. Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

Grafika - vizualizace numerických dat, grafická okna

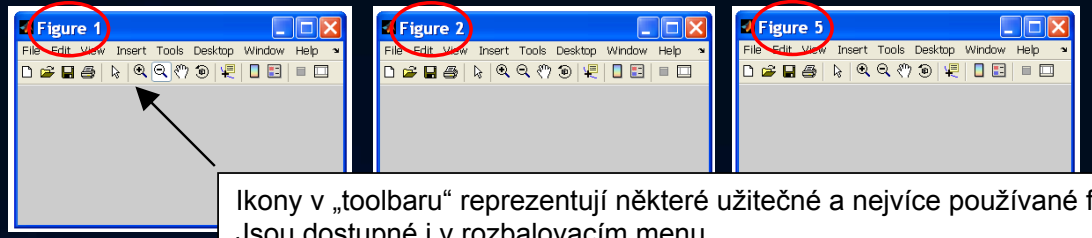
- Matlab nabízí širokou škálu možností 2D a 3D grafiky
- Grafický výstup je realizován v grafickém okně - Figure

Práce s grafickými okny :

– Grafická funkce Matlabu automaticky otevře grafické okno (Figure 1 pokud dosud neexistovalo) a graf do něj vykreslí
– Grafických oken může být i víc (liší se pořadovým číslem), můžeme mezi nimi přepínat
– Graf se vykreslí vždy do aktuálního okna (poslední otevřené, aktualizované příkazem nebo klikem)
– Příkazem figure (bez parametru) otevřeme grafické okno s dalším pořadovým číslem, resp. figure (n) s pořadovým číslem určeným parametrem n (n … přirozené)

Příklad :

```
>> figure; figure; figure(5);
```



Ikony v „toolbaru“ reprezentují některé užitečné a nejvíce používané funkce. Jsou dostupné i v rozbalovacím menu.

Práce v příkazovém okně (Command Window) :

Grafika - vizualizace numerických dat, grafická okna

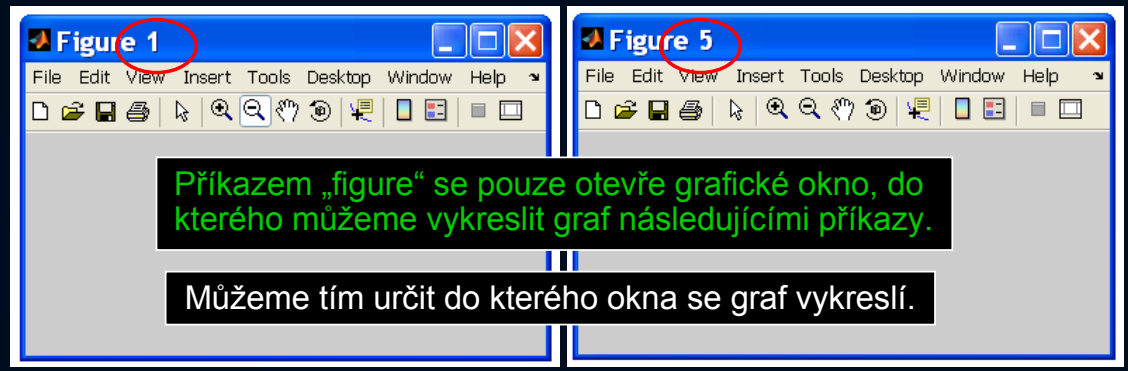
Zavření grafického okna :

- **close** … zavře aktivní grafické okno
- **close (n)** … zavře okno s pořadovým číslem určené parametrem n
- **close all** … zavře všechna grafická okna

Příklad :

```
>> figure, figure, figure(5)
>> close(2)
```

Byla otevřena tři okna:
Figure 1, Figure 2, Figure 5.
Druhým příkazem bylo Figure 2 zavřeno.



Matlab umožňuje řadu možností pro práci a nastavení grafického objektu a oken
(velikost, umístění, barvu, překrytí, …). Podrobnější informace viz učební text, help nebo manuál.

Práce v příkazovém okně (Command Window) :

2D grafika - použití příkazu plot

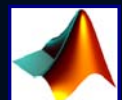
Základním příkazem pro vykreslení 2D grafu v prostředí Matlab je příkaz plot .

Syntaxe příkazu :

- **plot (y)** ... vykreslí a spojí lomenou čarou body, jejichž svislé pořadnice jsou dány hodnotami vektoru **y** (sloupcového nebo řádkového) v závislosti na pořadí
- **plot (x, y)** ... totéž, ale pořadnice **y** jsou vykresleny v závislosti na hodnotách vektoru **x**
- **plot (x, y, s)** ... totéž, ale pomocí textového řetězce **s** (uzavřeného mezi apostrofy) můžeme ovlivnit výsledný vzhled grafu (barvu, značky, čáry)

Proměnná **s** obsahuje až tři hodnoty vlastností v pořadí : barva, typ značky a typ čáry.
Pokud se některá z vlastností nedefinuje, použije se implicitní nastavení.

Implicitní nastavení : čára modrá, šířky 0,5 a bez značek jednotlivých bodů.

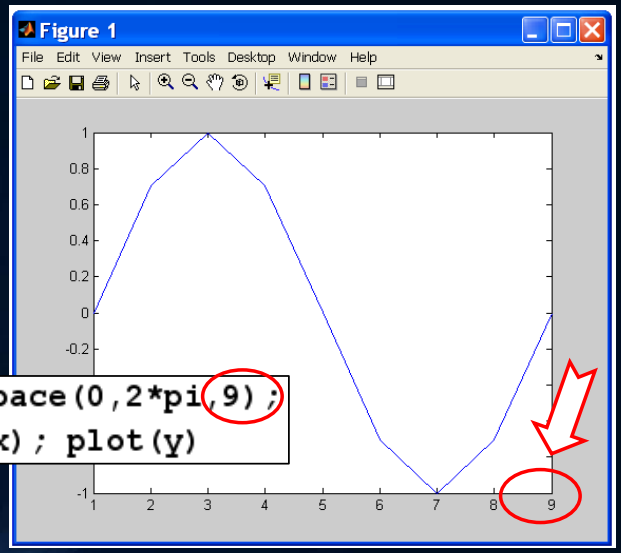


Práce v příkazovém okně (Command Window) :

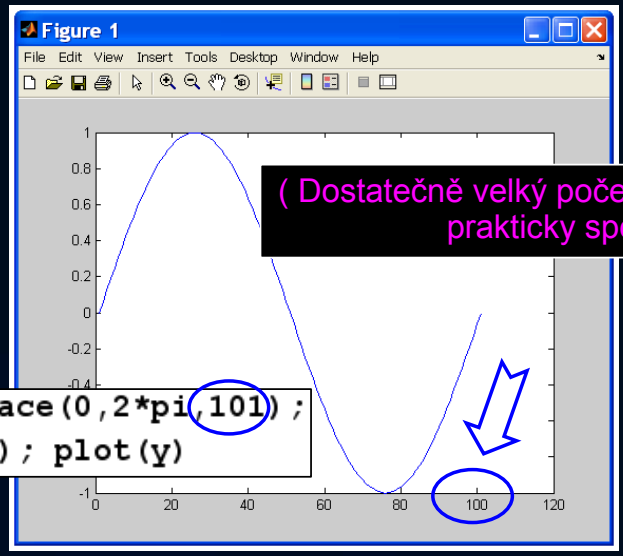
2D grafika - použití příkazu plot

- **plot (y)** ... vykreslí a spojí lomenou čarou body, jejichž svislé pořadnice jsou dány hodnotami vektoru **y** (sloupcového nebo řádkového) v závislosti na pořadí

Příklady :




```
>> x=linspace(0,2*pi,9);
>> y=sin(x); plot(y)
```




```
>> x=linspace(0,2*pi,101);
>> y=sin(x); plot(y)
```

(Dostatečně velký počet bodů vykreslí prakticky spojitou čáru.)

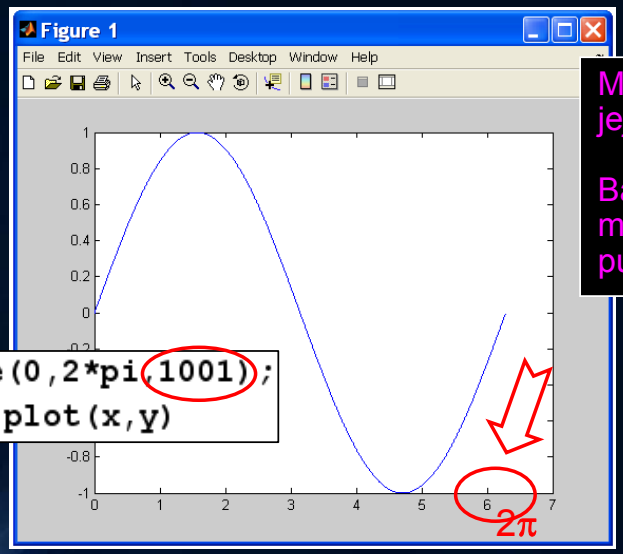
Pokud se nezadá jinak, Matlab měřítka grafu optimalizuje a nastaví automaticky sám.

Práce v příkazovém okně (Command Window) :

2D grafika - použití příkazu plot

- **plot (x, y)** ... vykreslí a spojí lomenou čarou body, jejichž svislé pořadnice jsou dány hodnotami vektoru **y** (sloupcového nebo řádkového) v závislosti na hodnotách vektoru **x**
- **plot (x1, y1, x2, y2, ...)** ... dtto pro více křivek (pro všechny uvedené dvojice souřadnic x_i, y_i)

Příklady :

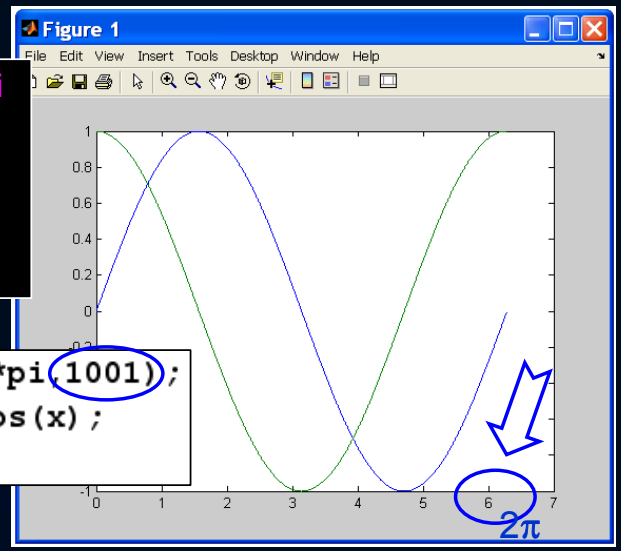



```
>> x=linspace(0,2*pi,1001);
>> y=sin(x); plot(x,y)
```

Měřítka grafu, styl, šířka čáry (0,5) i její barva se generují automaticky.

Barvy pro více grafů v pořadí : modrá, zelená, červená, tyrkysová, purpurová, žlutá, černá, bílá, …

```
>> x=linspace(0,2*pi,1001);
>> y1=sin(x);y2=cos(x);
>> plot(x,y1,x,y2)
```



Všechny úpravy vzhledu grafů (šířka, barva, styl, měřítko, mřížka, popis os, legenda, …) se dají dělat manuálně přímo v grafickém okně.

Práce v příkazovém okně (Command Window) :

2D grafika - použití příkazu plot

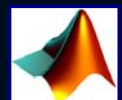
- **plot (x1, y1, s1, x2, y2, s2, …)** … totéž jako v minulém případě, ale můžeme pomocí textového řetězce s_i (uzavřeného mezi apostrofy) ovlivnit výsledný vzhled grafu (barvu, značky, čáry)

  Proměnná s_i obsahuje až tři hodnoty vlastností v pořadí : barva, typ značky a typ čáry.
  Pokud se některá z vlastností nedefinuje, použije se implicitní nastavení.

Přehled možných barev, typů čar a značek

Barva čáry		Typ čáry		Značka bodu			
b	**modrá** (blue)	**–**	**plná** (solid)	**.**	**tečka** (point)	**v**	**trojúhelník** (triangle – down)
g	**zelená** (green)	**:**	**tečkovaná** (dotted)	**o**	**kroužek** (circle)	**^**	**trojúhelník** (triangle – up)
r	**červená** (red)	**–.**	**čerchovaná** (dash-dot)	**x**	**křížek** (x-mark)	**<**	**trojúhelník** (triangle – right)
c	**tyrkysová** (cyan)	**– –**	**čárkovaná** (dashed)	**+**	**křížek** (plus)	**>**	**trojúhelník** (triangle – left)
m	**purpurová** (magenta)	(nic)	**bez čáry** (je-li zadán bod)	*****	**hvězdička** (star)	**p**	**pětiúhelník** (pentagram)
y	**žlutá** (yellow)			**s**	**čtverec** (square)	**h**	**šestiúhelník** (hexagram)
k	**černá** (black)			**d**	**kosočtverec** (diamond)		
w	**bílá** (white)						

Např. pro vykreslení červené tečkované čáry s jednotlivými body vyznačenými křížky bude **s = ' r x : '** .



Práce v příkazovém okně (Command Window) :

2D grafika - použití příkazu plot

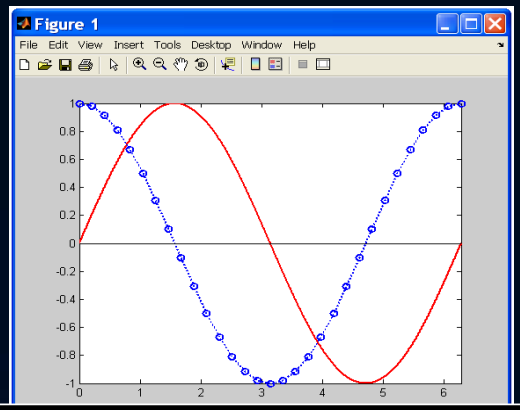
- **plot (x1, y1, s1, x2, y2, s2, …)** … vykreslí všechny deklarované křivky určené trojicemi x_i , y_i , s_i

Další užitečné možnosti :

- Šířku čáry lze nastavit pomocí parametru *LineWidth*, velikost značky pomocí parametru *MarkerSize* (viz příklad)
- Vykreslení dalšího průběhu do stejného okna zadáním příkazu **hold on** , nastavení trvá až do zadání příkazu **hold off** (implicitní nastavení je hold off)

Příklad :

```
>> x1=0:pi/1000:2*pi; y1=sin(x1);
>>        %vypocet sin s jemnym krokem
>> x2=linspace(0,2*pi,31); y2=cos(x2);
>>        %vypocet cos s hrubym krokem (kvuli znackam)
>> plot(x1,y1,'r',x2,y2,'bo:','MarkerSize',6,'LineWidth',2)
>>        %vykresleni sin cervene
>>        %vykresleni cos modre teckovane s kruhovymi znackami
>>        %velikost znacek 6, sirka cary 2
>> hold on, plot([0 2*pi],[0 0],'k')
>>        %vykresleni cerne slabe osy x
>> axis tight
>>        %nastaveni mezi os podle rozsahu dat
```



Všiměte si a pochopte použití příkazu **„axis tight“**.

Pokud nezadáme „hold on“, následující graf přemaže graf předcházející.

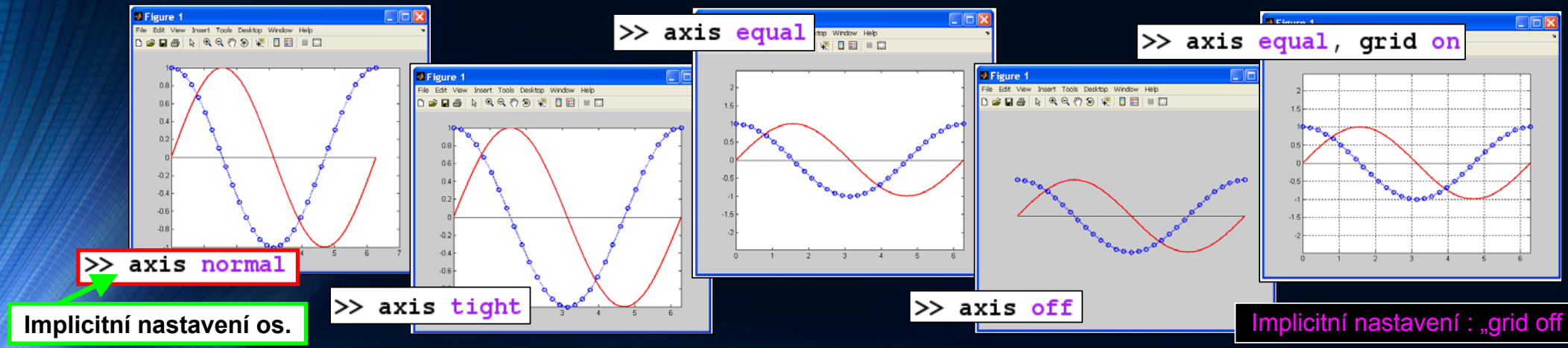
Práce v příkazovém okně (Command Window) :

2D grafika - ovládání souřadných os

- **axis tight** … nastaví meze souřadných os podle rozsahu dat
- **axis equal** … nastaví stejné měřítko obou souřadných os
- **axis ([x_{min} x_{max} y_{min} y_{max}])** … nastaví meze os podle zadaných parametrů
- **axis on** resp. **off** … zobrazí resp. potlačí souřadné osy
- **grid on** resp. **off** … zobrazí resp. potlačí mřížku

Matlab nabízí i další možnosti. Podrobnější informace viz učební text, help nebo manuál.

Příklady :

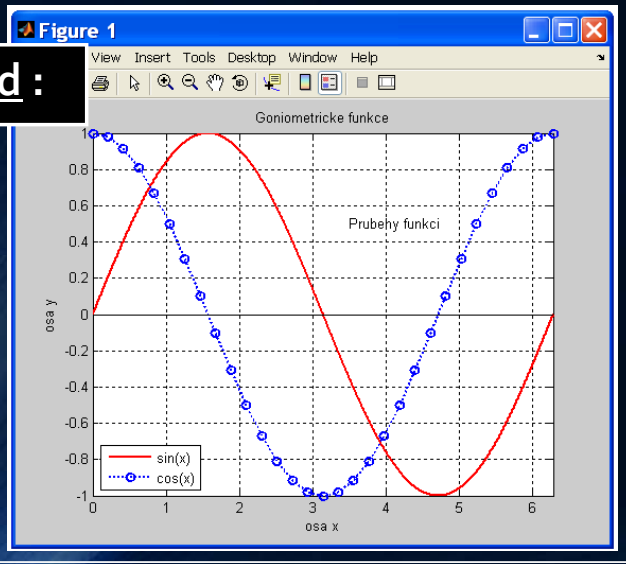


Práce v příkazovém okně (Command Window) :

2D grafika - textový popis grafu

- **title (s)** … název grafu, vypíše obsah textového řetězce **s** = ‘ Nazev grafu ’
- **xlabel (s)** … popis osy x, vypíše obsah textového řetězce **s**
- **ylabel (s)** … popis osy y, vypíše obsah textového řetězce **s**
- **text (x, y, s)** … vypíše obsah textového řetězce **s** na pozici určenou souřadnicemi **x, y**
- **legend (s_1, s_2, …, p)** … zobrazení legendy, postupně podle obsahů textových řetězců s_1, s_2, … v pořadí postupně vykreslovaných průběhů, příznak **p** určuje umístění : 1… vpravo nahoře (implicitní), 2 … vlevo nahoře, 3 … vlevo dole, 4 … vpravo dole

Příklad :




```
>> axis tight
>> axis tight, grid on
>> title ('Goniometricke funkce')
>> xlabel('osa x'), ylabel('osa y')
>> legend('sin(x)','cos(x)',3)
>> text(3.5,0.5,'Prubehy funkci')
```

Matlab nabízí celou řadu dalších možností (formátování textu, výběr fontů, řez, barvu, velikost, indexy, mat.symboly, speciální znaky, …).

Podrobnější informace viz učební text, help nebo manuál.

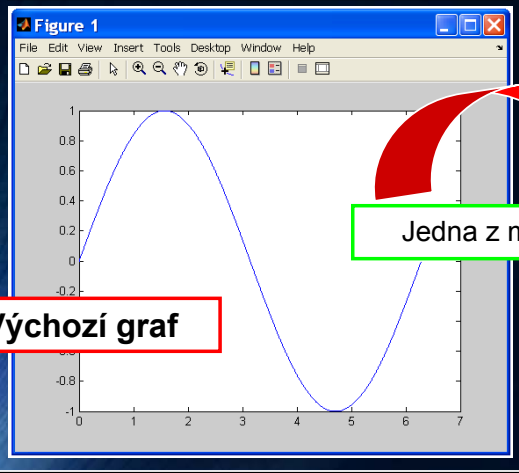
Práce v příkazovém okně (Command Window) :

2D grafika - manuální editace grafu

Užitečnou možností je manuální editace přímo v grafickém okně pomocí nabídek v ‘ toolbaru ’.

Příklad :

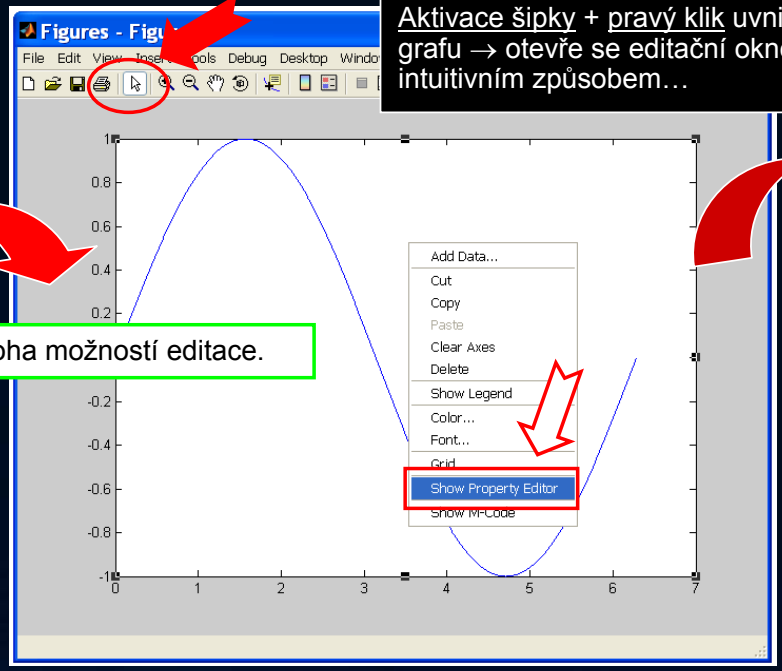
```
>> x=linspace(0,2*pi,1001);
>> y=sin(x); plot(x,y)
```



Výchozí graf

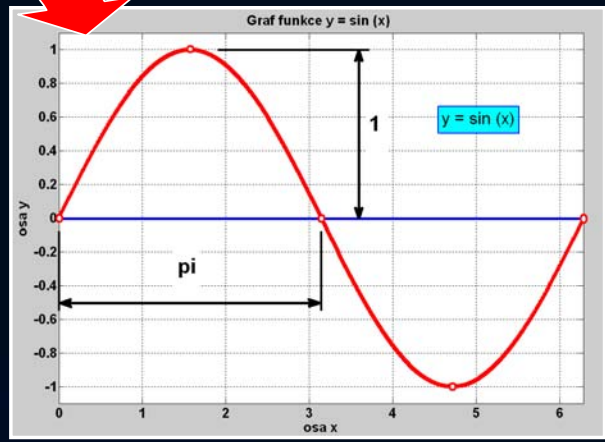
K editaci přistupujeme interaktivním způsobem pomocí nabídek menu.

Jedna z mnoha možností editace.



Aktivace šipky + pravý klik uvnitř rámečku grafu → otevře se editační okno os. Dále intuitivním způsobem…

Možný výsledek po editaci



Práce v příkazovém okně (Command Window) :

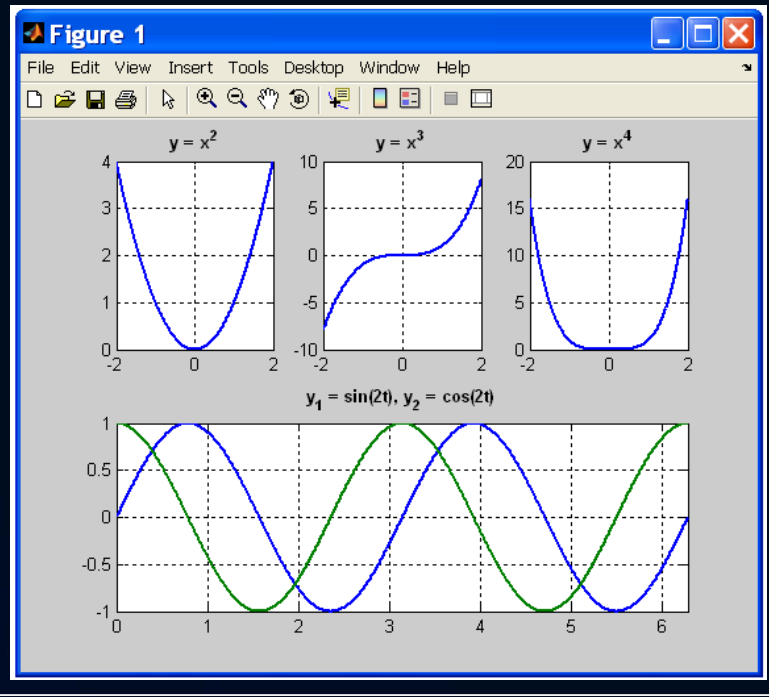
2D grafika - více grafů v jednom obrázku

- **subplot (m, n, p)** … kreslicí plochu rozdělí na **m** řádků a **n** sloupců parametr **p** určuje pozici vykreslovaného grafu (číslování po řádcích zleva doprava a po sloupcích shora dolů)

Příklad :

```
>> x=linspace(-2,2,1001); t=linspace(0,2*pi,1001);
>> subplot(2,3,1); plot(x,x.^2,'LineWidth',2);
>> grid on; title('\bfy = x^2')
>> subplot(2,3,2); plot(x,x.^3,'LineWidth',2);
>> grid on; title('\bfy = x^3')
>> subplot(2,3,3); plot(x,x.^4,'LineWidth',2);
>> grid on; title('\bfy = x^4')
>> subplot(2,1,2); plot(t,sin(2*t),t,cos(2*t),'LineWidth',2);
>> grid on; axis tight; title('\bfy_1 = sin(2t), y_2 = cos(2t)')
```

Za povšimnutí stojí zejména definice a umístění 4.grafu. Příkaz „subplot(2,1,2)“ rozčleňuje kreslicí plochu na dva řádky (v každém pouze jeden graf) a určuje vykreslení do druhé pozice (první pozicí je první, tedy horní řádek).



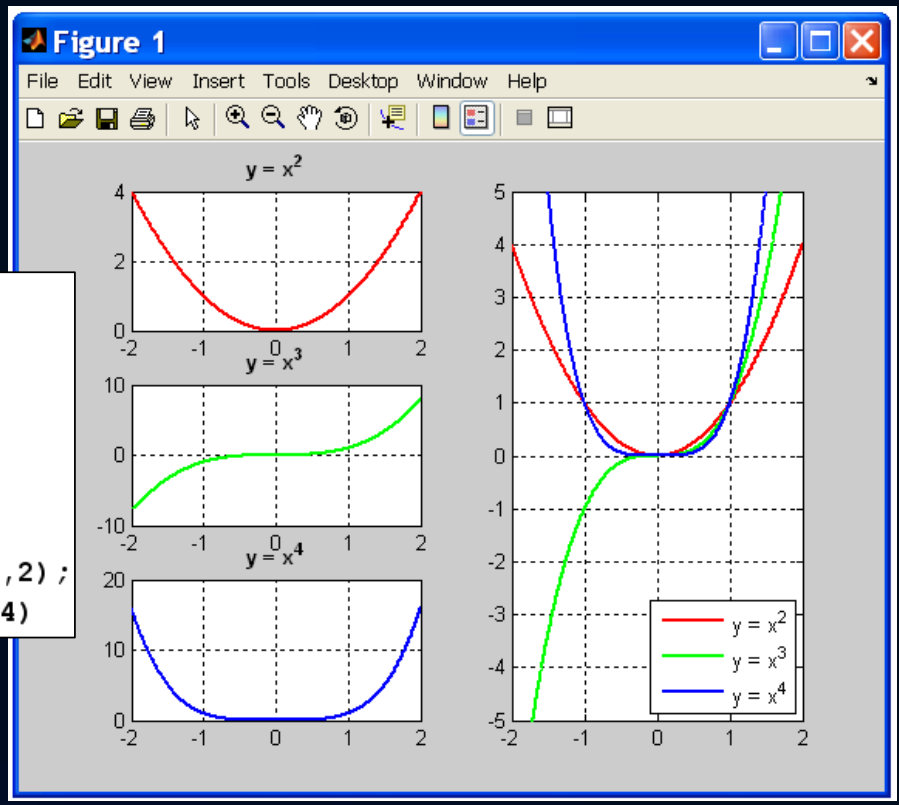
Práce v příkazovém okně (Command Window) :

2D grafika - více grafů v jednom obrázku

- **subplot (m, n, p)**

Jiný příklad rozdělení plochy obrázku :

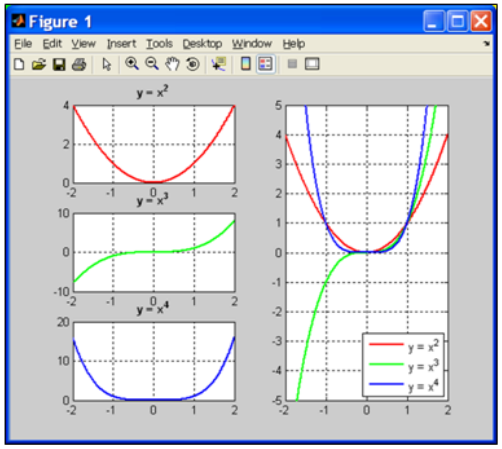
```
>> x=linspace(-2,2,1001);
>> subplot(3,2,1); plot(x,x.^2,'r','LineWidth',2);
>> grid on; title('\bfy = x^2')
>> subplot(3,2,3); plot(x,x.^3,'g','LineWidth',2);
>> grid on; title('\bfy = x^3')
>> subplot(3,2,5); plot(x,x.^4,'b','LineWidth',2);
>> grid on; title('\bfy = x^4')
>> subplot(1,2,2); plot(x,x.^2,'r',x,x.^3,'g',x,x.^4,'b','LineWidth',2);
>> grid on; axis([-2 2 -5 5]); legend('y = x^2','y = x^3','y = x^4',4)
```



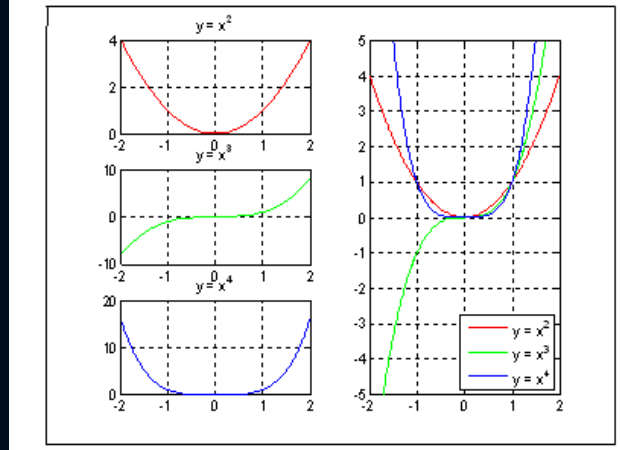
Všimněte si a pochopte význam parametrů příkazu subplot a jejich použití (je to poněkud komplikované).

Práce v příkazovém okně (Command Window) :

2D grafika - vložení obrázku do textového editoru



- Kombinací kláves :
  Alt + PrintScreen a **Ctrl + V**

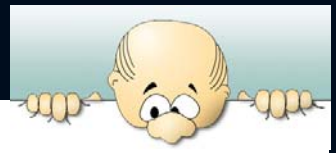


- V menu Figure :
  Edit → Copy Figure a **Ctrl + V**

V grafickém editoru lze samozřejmě obrázek ještě oříznout, měnit velikost, otočit atd.

2D grafika – příklady

Příklad 1



Nakreslete graf funkce :

$$y = K\, e^{\alpha t} \sin(\omega t) \quad , \quad t \in < 0, t_{\max} >$$

volte :

$$K = 2 \quad , \quad \alpha = -0{,}5 \quad , \quad \omega = 5 \quad , \quad t_{\max} = 4\pi$$

Dokážete si předem představit jaký má přibližně tato funkce průběh ?

2D grafika – příklady

Příklad 1

Nakreslete graf funkce :

$$y = K\, e^{\alpha t} \sin(\omega t) \quad , \quad t \in < 0, t_{max} >$$

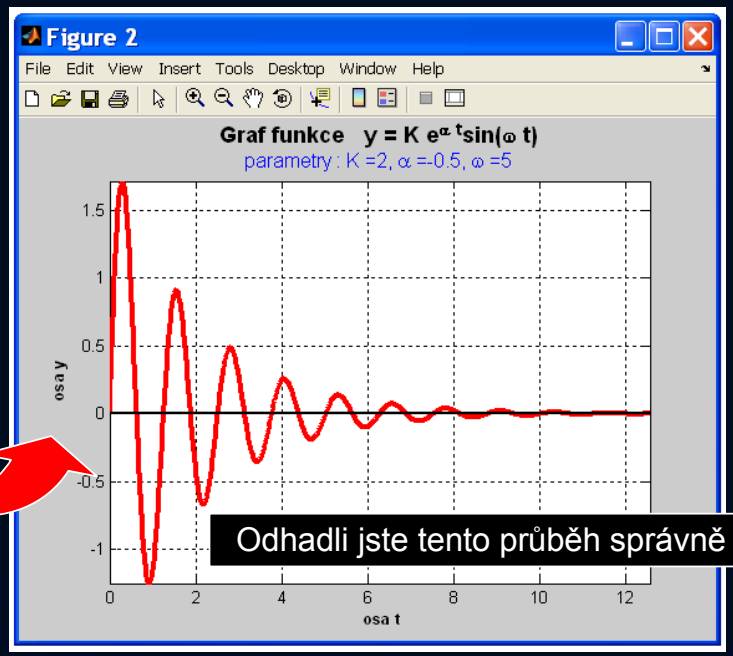
volte :

$$K = 2 \quad , \quad \alpha = -0{,}5 \quad , \quad \omega = 5 \quad , \quad t_{max} = 4\pi$$

Řešení :

Proč tam musí být tečka ?

```
>> K=2; alfa=-0.5; omega=5; tmax=4*pi;          %zadana data
>> t=linspace(0,tmax,1001);
>> y=K*exp(alfa*t).*sin(omega*t);               %vypocet funkce
>> plot(t,y,'r','LineWidth',3);                 %kresba grafu
>> hold on; grid on; axis tight                 %nastaveni
>> xlabel('\bf osa t'); ylabel('\bf osa y')     %popis os
>> plot([0 tmax],[0 0],'k','LineWidth',2)       %kresba osy t
>> %popis grafu :
>> title({'\bf\fontsize{14}Graf funkce   y = K e^{\alpha t}sin(\omega t)',...
   ['\rm\color{blue}\fontsize{12}parametry : K =',num2str(K),...
   ', \alpha =',num2str(alfa),', \omega =',num2str(omega)]})
```



Odhadli jste tento průběh správně ?

Všimněte si (a pochopte) techniku popisu grafu.

Zamyslete se nad průběhem závislosti i s jinými parametry a ověřte následně výpočtem.

2D grafika – příklady

Příklad 2

Nakreslete graf epicykloidy :

$$x = (a+b)\cos\left(\frac{b}{a}t\right) - c\cos\left(\frac{a+b}{a}t\right)$$

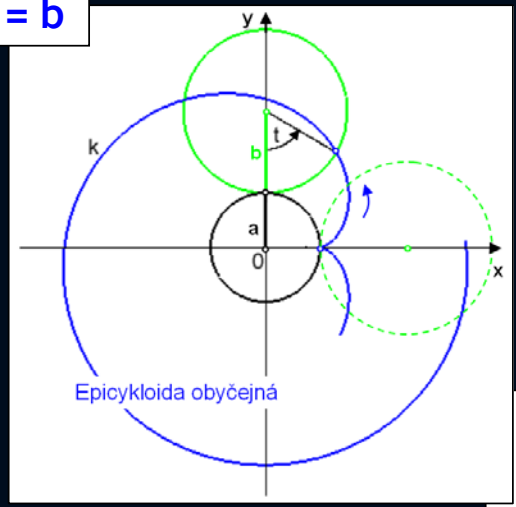
$$y = (a+b)\sin\left(\frac{b}{a}t\right) - c\sin\left(\frac{a+b}{a}t\right)$$

parametry a, b, c, t volte vhodně sami

c > b … epicykloida prodloužená
c < b … zkrácená
c = b … obyčejná

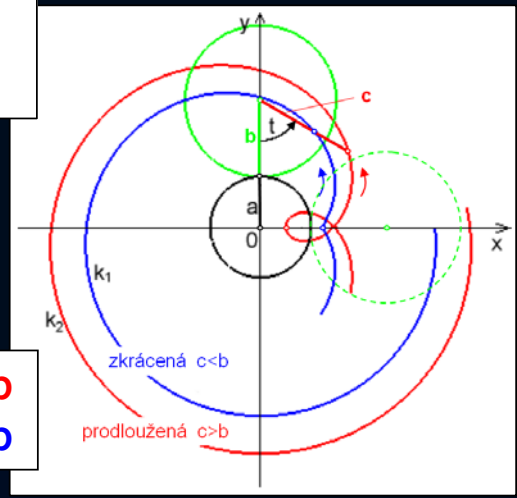
Epicykloida je uzavřenou křivkou, je-li podíl a / b racionální.

c = b



Epicykloida vzniká odvalováním kružnice s poloměrem b (zelená) po kružnici s poloměrem a (černá). Příslušný bod spojený s odvalující se kružnicí opisuje epicykloidu.

c > b
c < b



2D grafika – příklady

Příklad 2

Řešení :

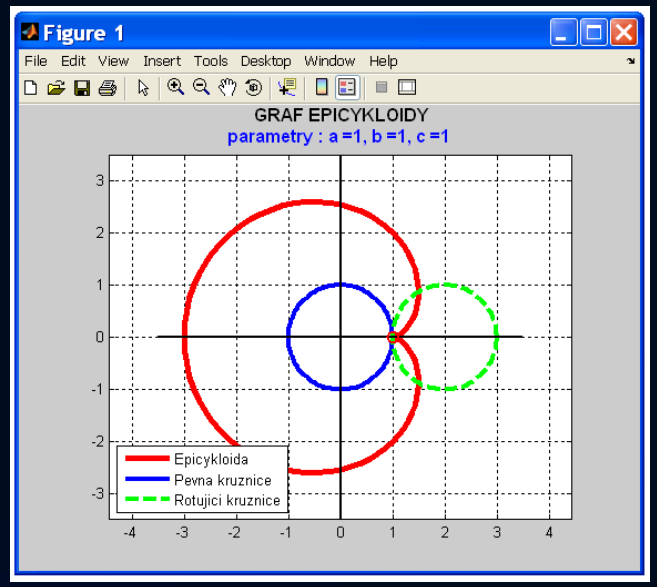
```
>> a=1; b=1; c=1; t=linspace(0,2*pi,101);    %zadane parametry
>> x=(a+b)*cos(b*t/a)-c*cos((a+b)*t/a);...   %vypocet souradnic
   y=(a+b)*sin(b*t/a)-c*sin((a+b)*t/a);
>> plot(x,y,'r','LineWidth',4); axis equal;  %kresba grafu
>> grid on, hold on                          %kresba mrizky
>> %kresba pevne a rotujici kruznice:
>> plot(a*cos(t),a*sin(t),'b','LineWidth',3)
>> plot(b*cos(t)+(a+b),b*sin(t),'--g','LineWidth',3)
>> %osy grafu:
>> plot([-(a+2*b+0.5)  (a+2*b+0.5)],[0 0],'k','LineWidth',2)
>> plot([0 0],[-(a+2*b+0.5)  (a+2*b+0.5)],'k','LineWidth',2)
>> plot(a+b-c,0,'r o','MarkerSize',8,'LineWidth',2)%tvorici bod
>> %popis grafu:
>> title({'\bf\fontsize{12} GRAF EPICYKLOIDY',...
   ['\color{blue}parametry : a =',num2str(a),...
   ', b =',num2str(b),', c =',num2str(c)]})
>> legend('Epicykloida','Pevna kruznice','Rotujici kruznice',3)
```

Všimněte si a pochopte techniku popisu grafu.

Nakreslete graf epicykloidy :

$$x = (a+b)\cos\left(\frac{b}{a}t\right) - c\cos\left(\frac{a+b}{a}t\right)$$
$$y = (a+b)\sin\left(\frac{b}{a}t\right) - c\sin\left(\frac{a+b}{a}t\right)$$

parametry a, b, c, t volte vhodně sami



Kardioida (srdcovka) = speciální případ a=b=c .

2D grafika – příklady

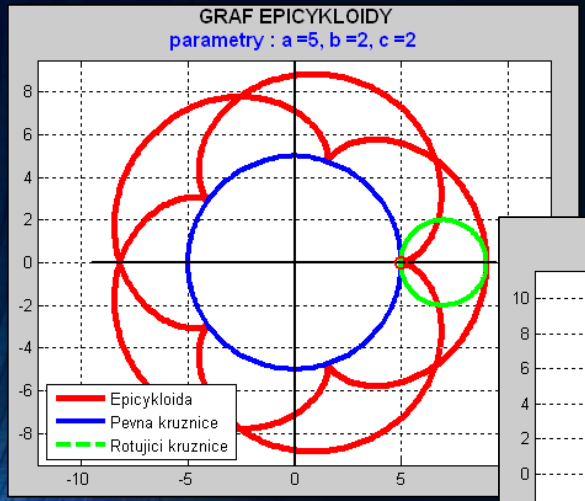
Příklad 2

Různé varianty epicykloidy :

Nakreslete graf epicykloidy :

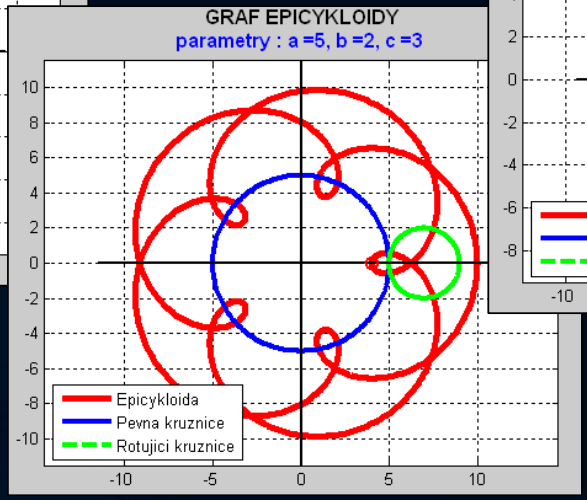
$$x = (a+b)\cos(\frac{b}{a}t) - c\cos(\frac{a+b}{a}t)$$
$$y = (a+b)\sin(\frac{b}{a}t) - c\sin(\frac{a+b}{a}t)$$

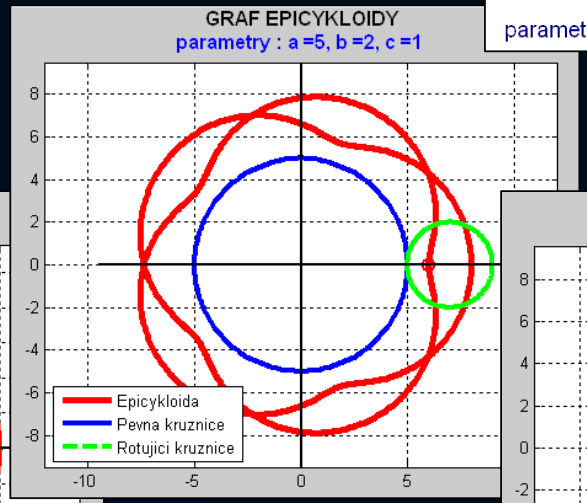
parametry a, b, c, t volte vhodně sami



a = 5, b = c = 2
epicykloida obyčejná

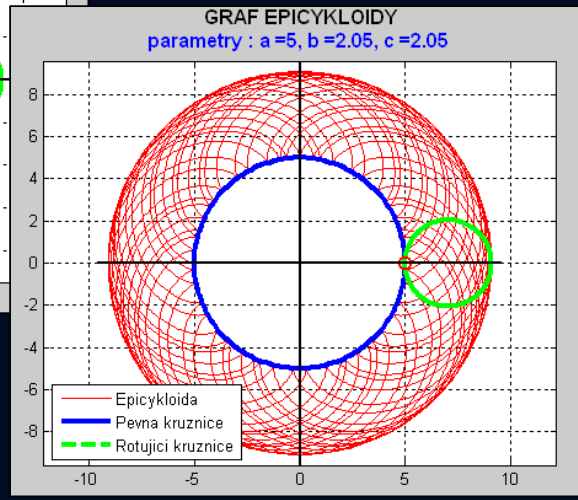
a = 5, b = 2, c = 3
epicykloida prodloužená





a = 5, b = 2, c = 1
epicykloida zkrácená

a = 5, b = c = 2,05
epicykloida obyčejná



2D grafika – příklady

Příklad 3

Nakreslete graf parametricky zadané funkce :

$$x = 16 \sin^3 t$$
$$y = 13 \cos t - 5 \cos 2t - 2 \cos 3t - \cos 4t$$
$$t \in < -\pi, \pi >$$

Samostatná práce – odhadnete jak bude graf vypadat ?

Pracujte týmově – zapojte do práce i své kolegy.

2D grafika – příklady

Příklad 3

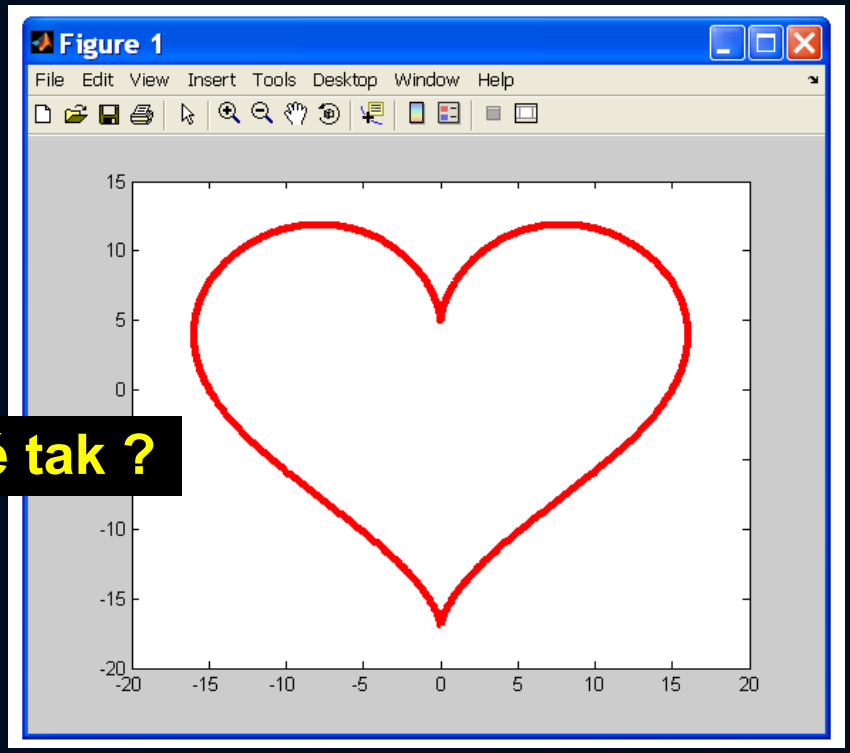
Nakreslete graf parametricky zadané funkce :

$$x = 16\sin^3 t$$
$$y = 13\cos t - 5\cos 2t - 2\cos 3t - \cos 4t$$
$$t \in \langle -\pi, \pi \rangle$$

Řešení :

```
>> t=linspace(-pi,pi,1001);
>> x=16*sin(t).^3;
>> y=13*cos(t)-5*cos(2*t)-2*cos(3*t)-cos(4*t);
>> plot(x,y,'r','LineWidth',4)
```

Vyšlo vám to také tak ?



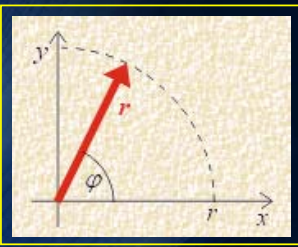
2D grafika – příklady

Příklad 4

Nakreslete graf parametricky zadané funkce v polárních souřadnicích :

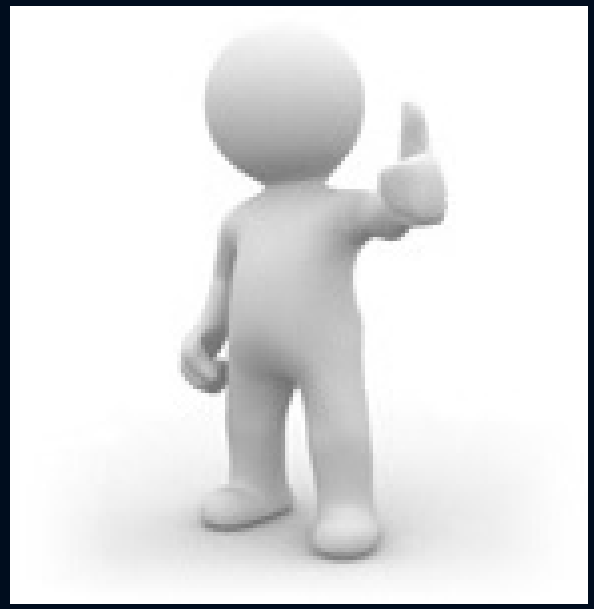
$$r(\varphi) = 2 - 2\sin\varphi + \sin\varphi \frac{\sqrt{|\cos\varphi|}}{\sin\varphi + 1{,}4}$$

$$\varphi \in \langle 0, 2\pi \rangle$$



Princip zobrazení v polárních souřadnicích :

$$x = r\cos\varphi \quad , \quad y = r\sin\varphi$$



Zkuste sami !

2D grafika – příklady

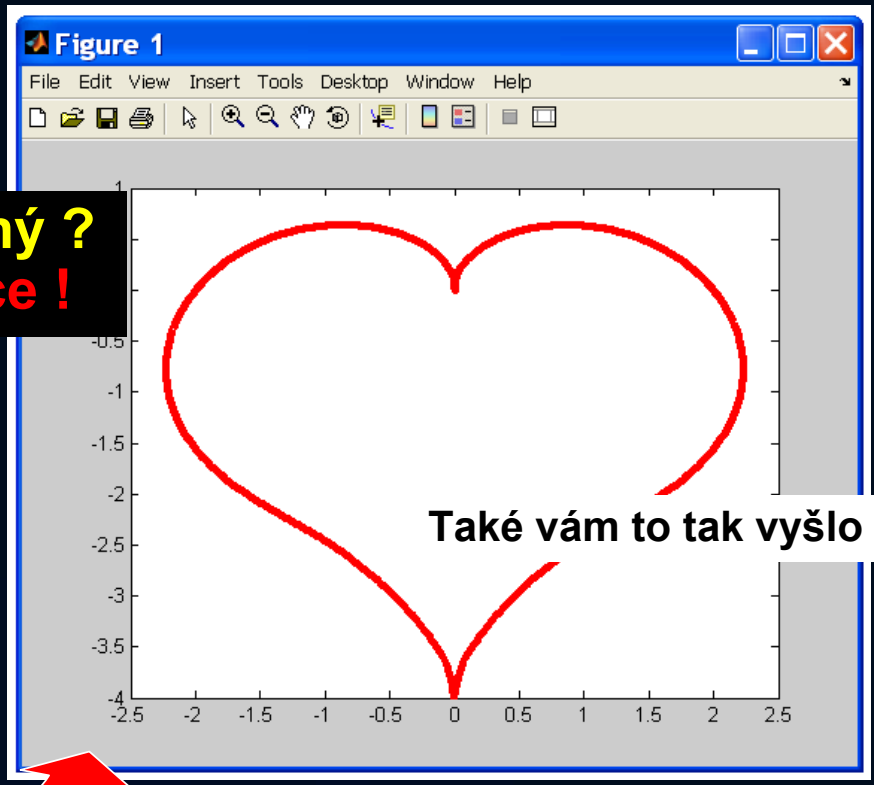
Příklad 4

Nakreslete graf parametricky zadané funkce v polárních souřadnicích :

$$r(\varphi) = 2 - 2\sin\varphi + \sin\varphi \frac{\sqrt{|\cos\varphi|}}{\sin\varphi + 1{,}4}$$

$$\varphi \in < 0, 2\pi >$$

Je to možný ?
Zase srdce !



Také vám to tak vyšlo ?

Řešení :

```
>> fi=linspace(0,2*pi,1001);
>> r=2-2*sin(fi)+sin(fi).*...
   (sqrt(abs(cos(fi)))./(1.4+sin(fi)));
>> plot(r.*cos(fi),r.*sin(fi),'r','LineWidth',4)
```

2D grafika – příklady

Příklad 4

Nakreslete graf parametricky zadané funkce v polárních souřadnicích :

$$r(\varphi) = 2 - 2\sin\varphi + \sin\varphi \frac{\sqrt{|\cos\varphi|}}{\sin\varphi + 1{,}4}$$

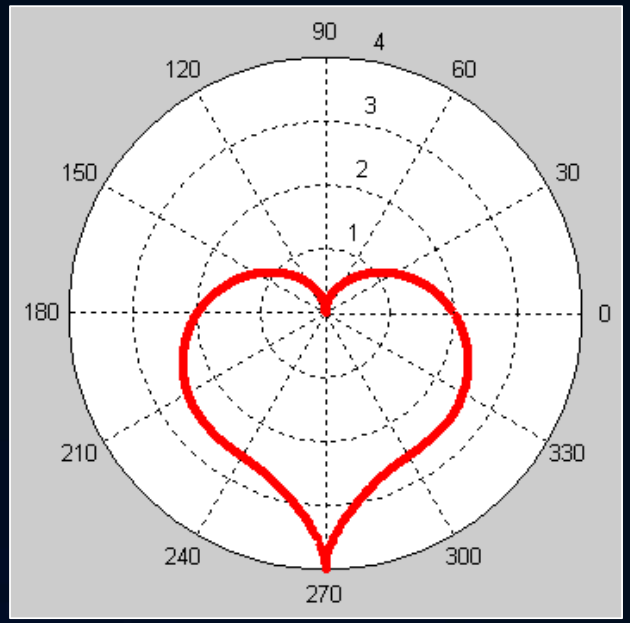
$$\varphi \in < 0,\ 2\pi >$$

Řešení (varianta s použitím příkazu **polar** pro kreslení v polárních souřadnicích) :

```
>> fi=linspace(0,2*pi,1001);
>> r=2-2*sin(fi)+sin(fi).*...
   (sqrt(abs(cos(fi)))./(1.4+sin(fi)));
>> polar(fi,r,'r')
```

```
>> %resp.silnejsi carou :
>> set(polar(fi,r,'r'),'Linewidth',4)
```

Je použito ve vedlejším obrázku.



Všimněte si použití popisovače (tzv. Handle Graphics), kterým lze pomocí příkazu **set** nastavit vlastnosti objektu.

Použijte nápovědu příkazů „polar“ a „set“.

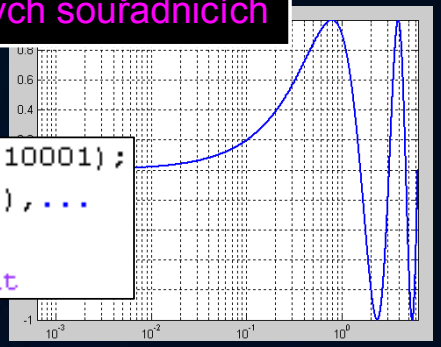
Práce v příkazovém okně (Command Window) :

2D grafika - speciální typy grafů

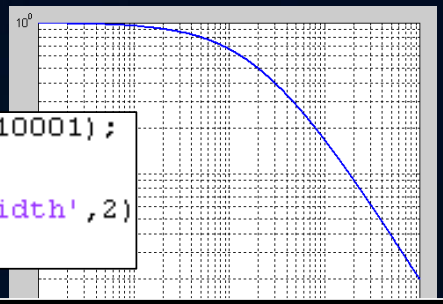
Příklady:

Kreslení v logaritmických souřadnicích

```
>> t=linspace(0,2*pi,10001);
>> semilogx(t,sin(2*t),...
    'LineWidth',2)
>> grid on, axis tight
```



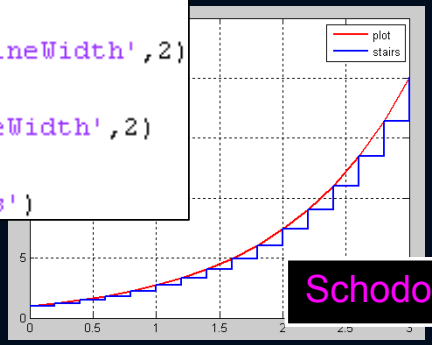
```
>> x=linspace(0,100,10001);
>> y=2./(x+2);
>> loglog(x,y,'LineWidth',2)
>> grid on
```



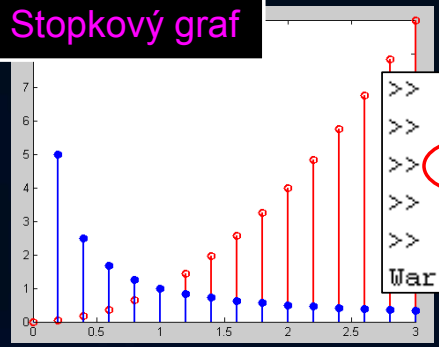
Použijte nápovědu příkazů „semilog“, „loglog“, „stairs“ a „stem“.

Schodový graf

```
>> dT=0.2;
>> t=0:dT:3;
>> plot(t,exp(t),'r','LineWidth',2)
>> hold on
>> stairs(t,exp(t),'LineWidth',2)
>> grid on
>> legend('plot','stairs')
```



Stopkový graf



```
>> dT=0.2;
>> t=0:dT:3;
>> stem(t,t.^2,'r','LineWidth',2)
>> hold on
>> stem(t,1./t,'fill','LineWidth',2)
Warning: Divide by zero.
```

Všimněte si jak se projevilo dělení nulou.

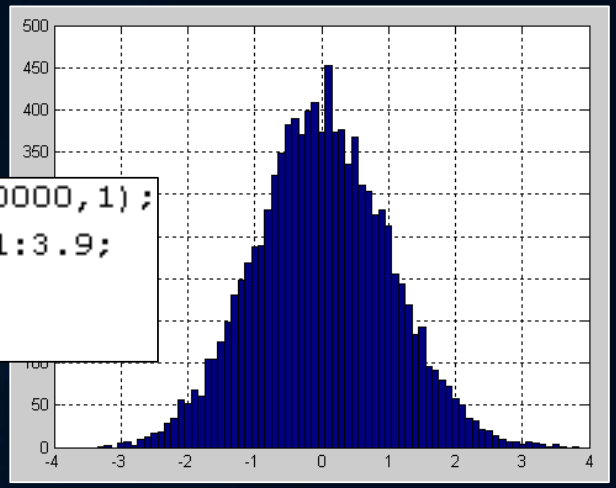
Práce v příkazovém okně (Command Window) :

2D grafika - speciální typy grafů

Příklady:

Použijte nápovědu příkazů „randn“, „hist“ a „pie“.

Histogram

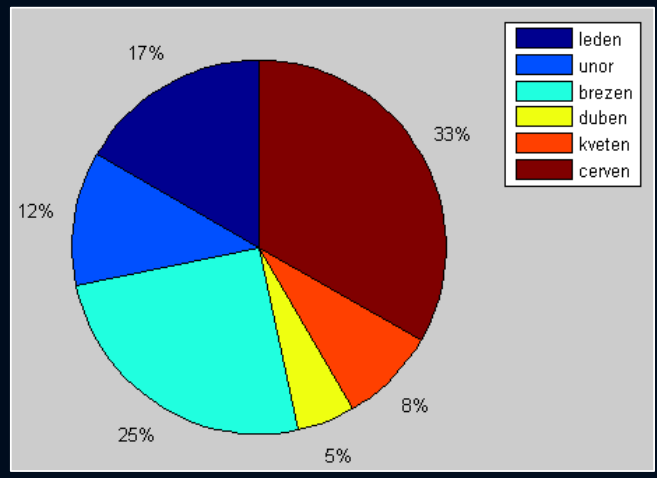



```
>> v=randn(10000,1);
>> x=-3.9:0.1:3.9;
>> hist(v,x)
>> grid on
```

Histogram vyjadřuje četnost výskytu numerických hodnot prvků souboru v daném intervalu.

V příkladu : vytvořen vektor náhodných čísel **v** s normálním rozložením a vykreslen na intervalu **x** ∈ <-3,9 ; +3,9> s krokem 0,1 histogram četnosti jejich výskytu.

Koláčový graf




```
>> v=[1 0.7 1.5 0.3 0.5 2];
>> pie(v)
>> legend('leden','unor','brezen',...
        'duben','kveten','cerven',-1)
```

Všimněte si parametru „-1“ v příkazu „legend“, vysvětlete jeho význam (viz nápověda) .

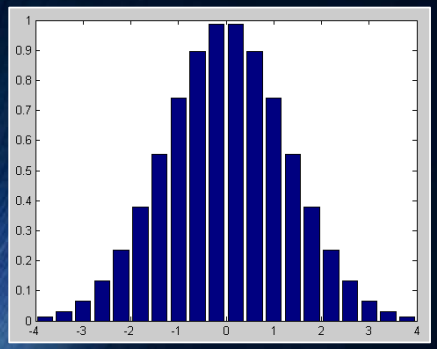
Práce v příkazovém okně (Command Window) :

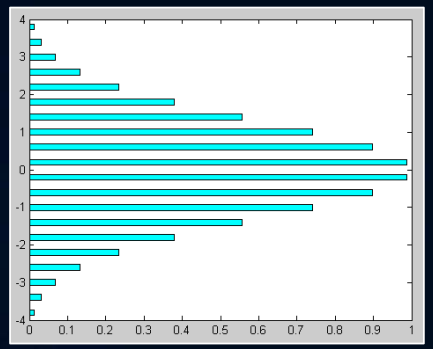
2D grafika - speciální typy grafů

Příklady:

Sloupcový graf

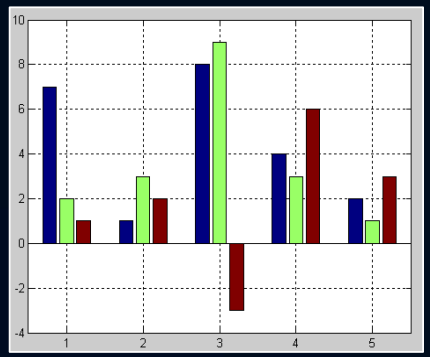
```
>> x=linspace(-3.8,3.8,20); y=exp(-0.3*x.^2);
>> bar(x,y), figure, barh(x,y,0.4,'c')
```

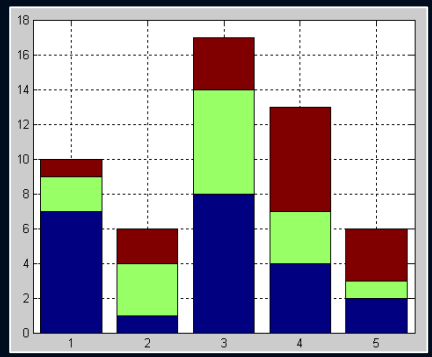




Sloupcový graf – vykreslení matice

```
>> M=[7 2 1;1 3 2;8 9 -3;4 3 6;2 1 3];
>> bar(M), grid on, figure, bar(M,'stack'), grid on
```





Použijte nápovědu příkazů „bar“ a „barh“.

Další grafické možnosti Matlabu viz učebnice nebo nápověda.

Práce v příkazovém okně (Command Window) :

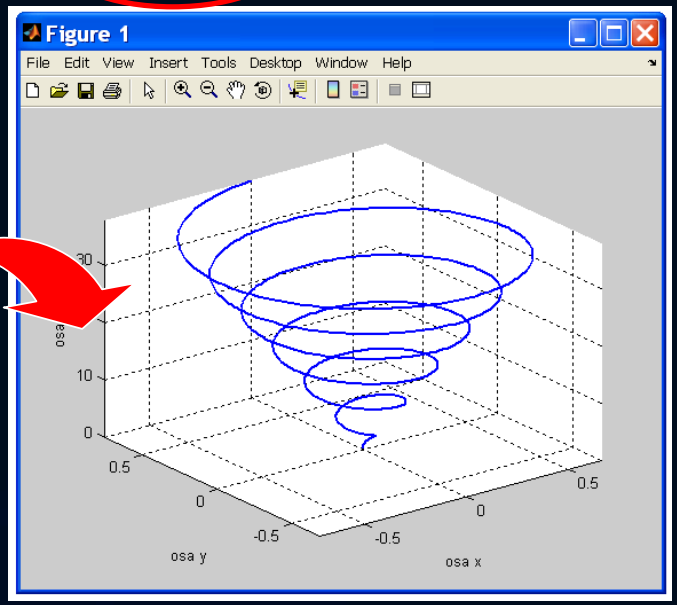
3D grafika - použití příkazu plot3

Základním příkazem pro vykreslení 3D grafu v prostředí Matlab je příkaz plot3 .

Syntaxe příkazu **plot3** je stejná jako příkazu **plot** , **pouze s rozšířením pro třetí souřadnici.**

Příklad :

```
>> t=linspace(0,12*pi,1001);
>> plot3(0.02*t.*sin(t),0.02*t.*cos(t),t,'LineWidth',2)
>> grid on; axis tight
>> xlabel('osa x'); ylabel('osa y'); zlabel('osa z')
```



Za povšimnutí stojí parametry použitého příkazu „plot3“ a nově použitý příkaz „zlabel“.

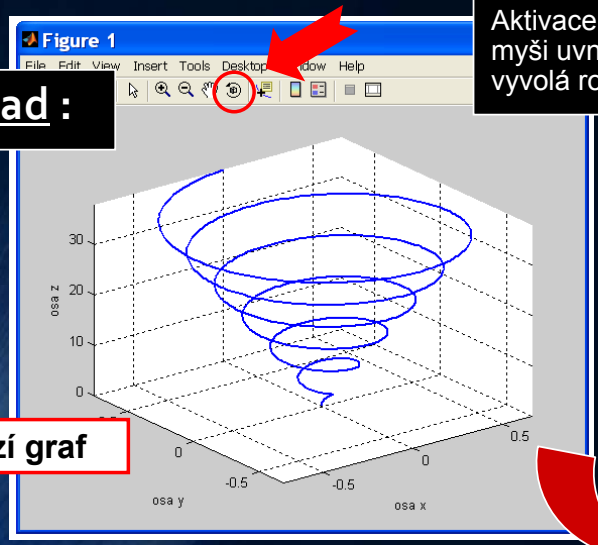
Práce v příkazovém okně (Command Window) :

3D grafika - manuální editace grafu

Užitečnou možností je manuální editace přímo v grafickém okně.
V ‘toolbaru’ grafického okna jsou k dispozici všechny funkce jako u 2D grafu.

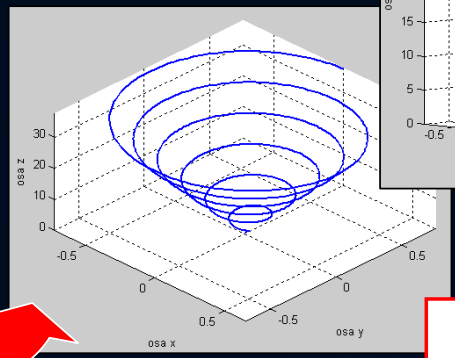
- **Často používaná a zajímavá funkce je rotace** (funguje i u 2D grafu, ale nemá valný smysl)

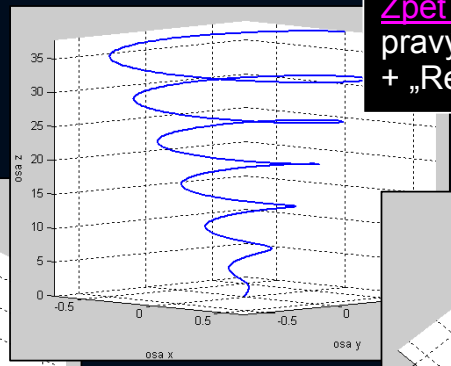
Příklad :



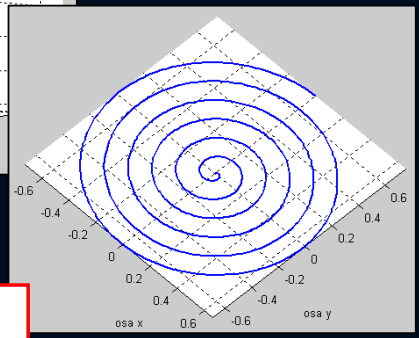
Výchozí graf

Aktivace ikony rotace + stisk levého tlačítka myši uvnitř rámečku grafu → pohyb myší vyvolá rotaci objektu.





Zpět k původnímu pohledu :
pravý klik do obrázku +
+ „Reset to Original View“



Příklady různých pohledů na objekt

Práce v příkazovém okně (Command Window) :

3D grafika - graf funkce dvou proměnných

$$z = f(x, y)$$

Funkce je vypočtena (a následně vykreslena) v bodech určených různými kombinacemi zadaných hodnot **x** a **y** .
Tento rastr vytvoříme pomocí příkazu **meshgrid** .

Syntaxe příkazu :

- **[Xm, Ym] = meshgrid (x, y)** … **x** , **y** vektory nezávisle proměnných ; **Xm**, **Ym** matice rastru

Příklad :

```
>> x=[1 3 5]; y=[2 4];
>> [Xm,Ym]=meshgrid(x,y)
Xm =
     1     3     5
     1     3     5
Ym =
     2     2     2
     4     4     4
```

Pomocné matice rastru **Xm**, **Ym** mají na stejnolehlých pozicích různé kombinace prvků zadaných vektorů nezávislých proměnných **x** a **y** .

(Tím je rastr definován.)

Rastr mřížky nezávisle proměnných NEMUSÍ být vytvořen příkazem meshgrid – může být zadán manuálně nebo tabulkou.

Práce v příkazovém okně (Command Window) :

3D grafika - graf funkce dvou proměnných z = f (x, y)

Nejčastější příkazy pro vykreslení síťového grafu funkce :

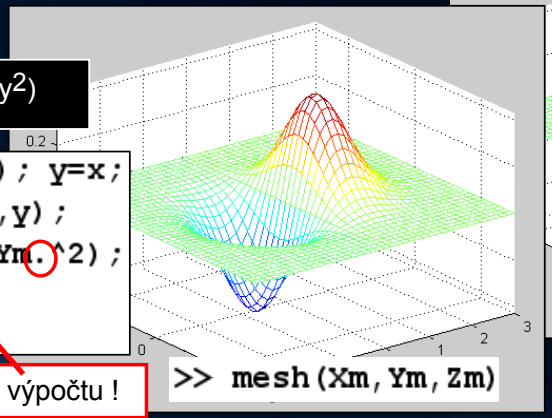
- **mesh** (**X**m, **Y**m , **Z**m) … vykreslí síťový graf propojením bodů pořadnic funkce **Zm** vypočtených na rastru určeném maticemi **Xm**, **Ym**
- **meshc** (**X**m, **Y**m , **Z**m) … varianta síťového grafu s vrstevnicemi
- **meshz** (**X**m, **Y**m , **Z**m) … varianta síťového grafu s nulovou rovinou
- **waterfall** (**X**m, **Y**m , **Z**m) … varianta tzv. „vodopádového“ grafu

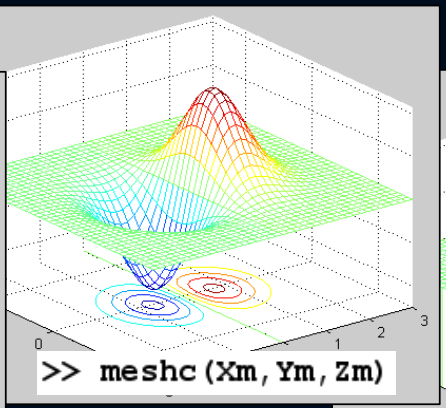
Příklady :

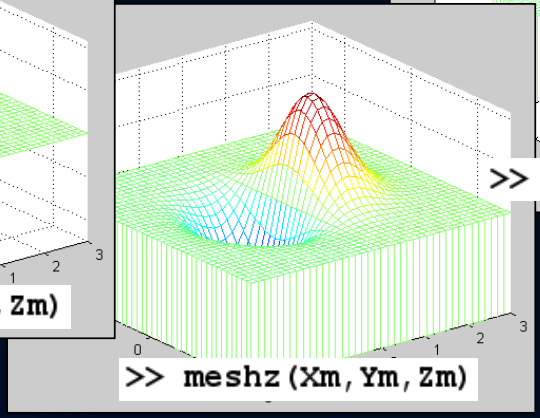
$z = x\, e^{(-x^2 - y^2)}$

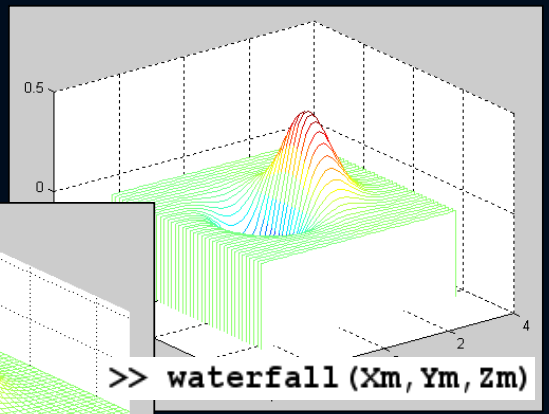
```
x=linspace(-3,3,41); y=x;
[Xm,Ym]=meshgrid(x,y);
Zm=Xm.*exp(-Xm.^2-Ym.^2);
mesh(Xm,Ym,Zm)
axis tight
```

Pozor na význam **tečky** ve výpočtu !









Další podrobnosti viz učebnice nebo manuál.

Práce v příkazovém okně (Command Window) :

3D grafika - graf funkce dvou proměnných z = f (x, y)

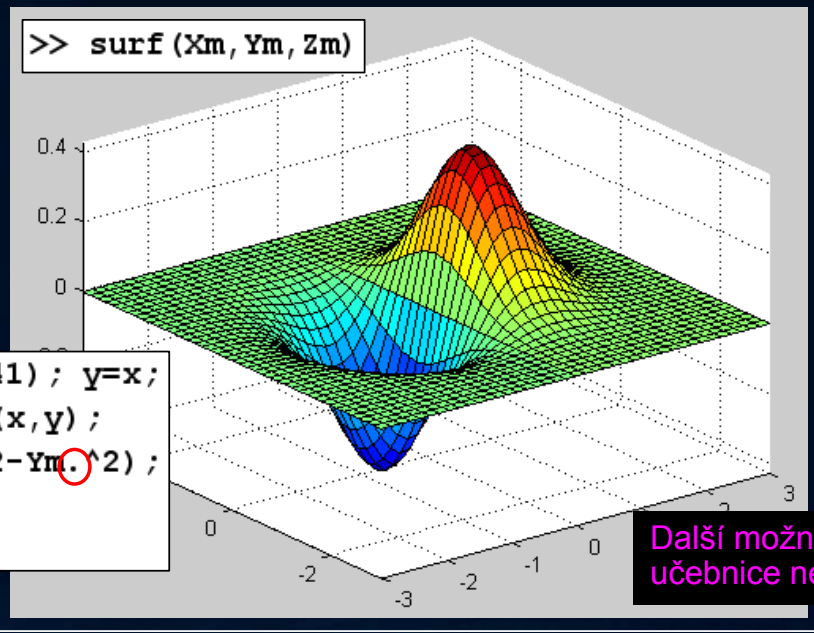
Nejčastější příkazy pro vykreslení plošného grafu funkce :

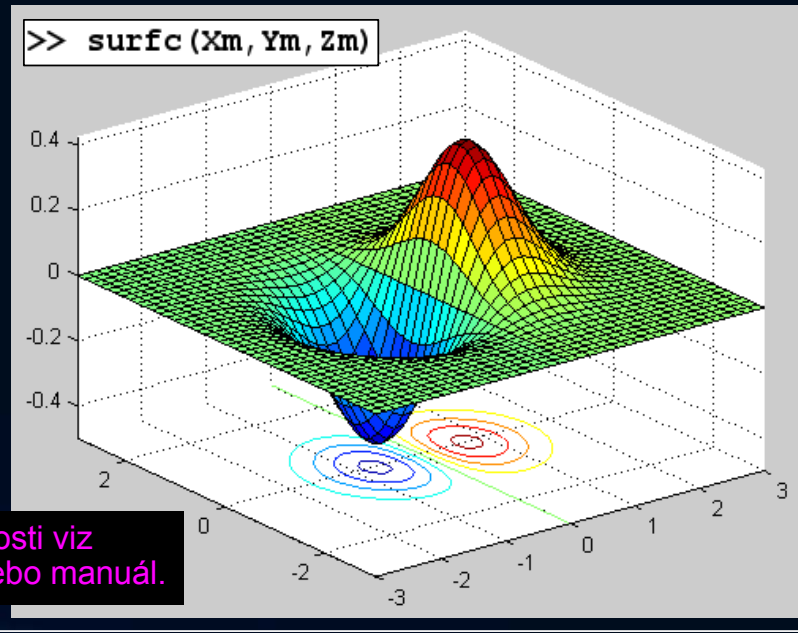
- **surf** (**X**m, **Y**m , **Z**m) … obdoba příkazu mesh pro vybarvený plošný graf funkce
- **surfc** (**X**m, **Y**m , **Z**m) … obdoba příkazu meshc pro graf s vrstevnicemi

Příklady :

$z = x\, e^{(-x^2 - y^2)}$

```
>> x=linspace(-3,3,41); y=x;
>> [Xm,Ym]=meshgrid(x,y);
>> Zm=Xm.*exp(-Xm.^2-Ym.^2);
>> mesh(Xm,Ym,Zm)
>> axis tight
```





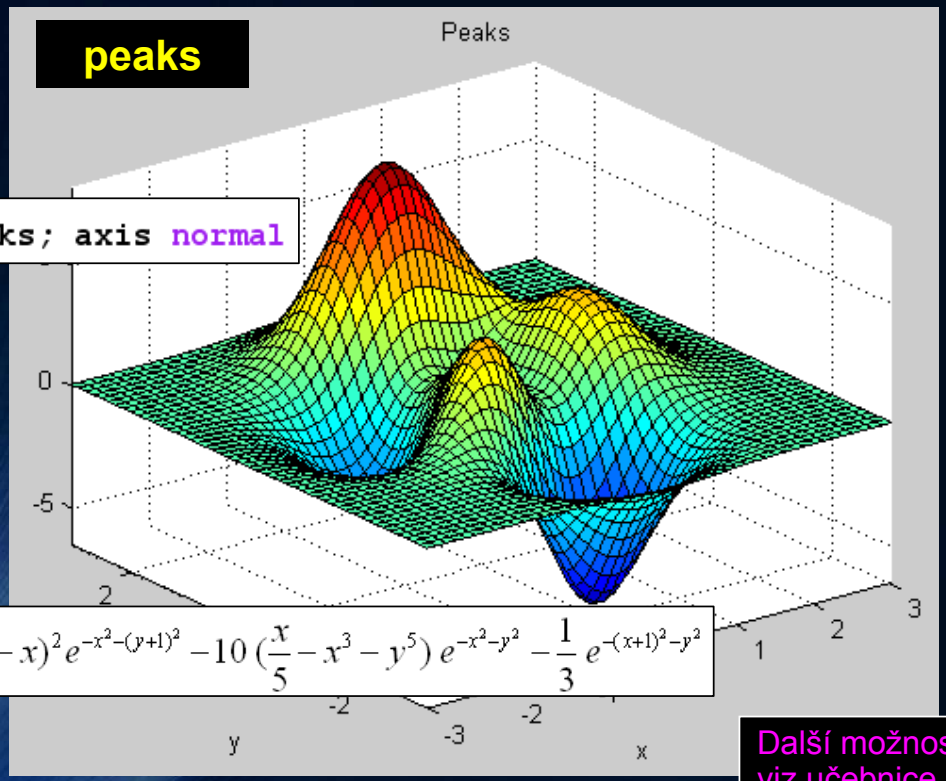
Další možnosti viz učebnice nebo manuál.

Práce v příkazovém okně (Command Window) :

3D grafika - předdefinované demonstrační funkce

peaks

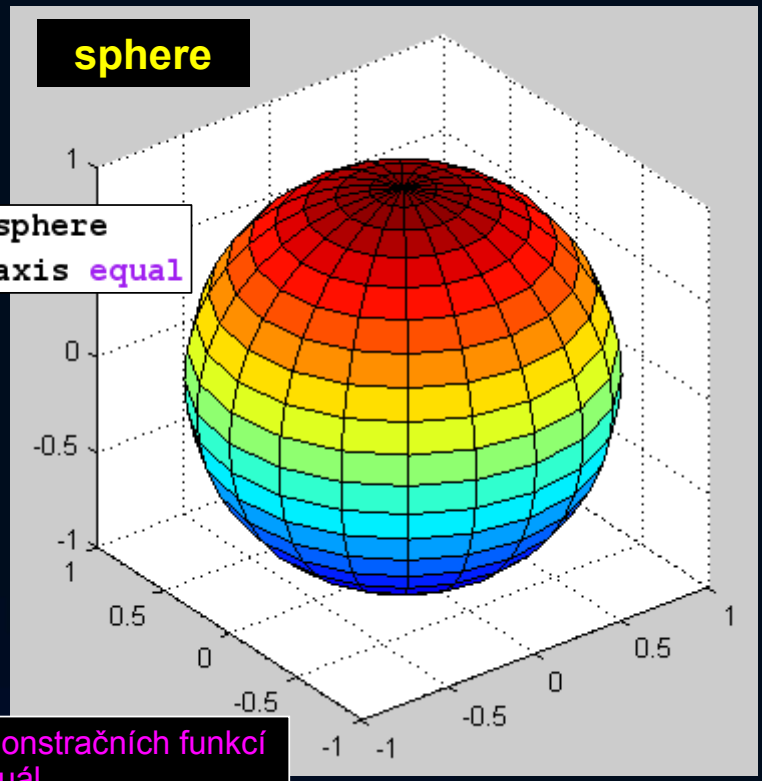
```
>> peaks; axis normal
```



$$z = 3\,(1-x)^2 e^{-x^2-(y+1)^2} - 10\,(\frac{x}{5} - x^3 - y^5)\,e^{-x^2-y^2} - \frac{1}{3}\,e^{-(x+1)^2-y^2}$$

sphere

```
>> sphere
>> axis equal
```

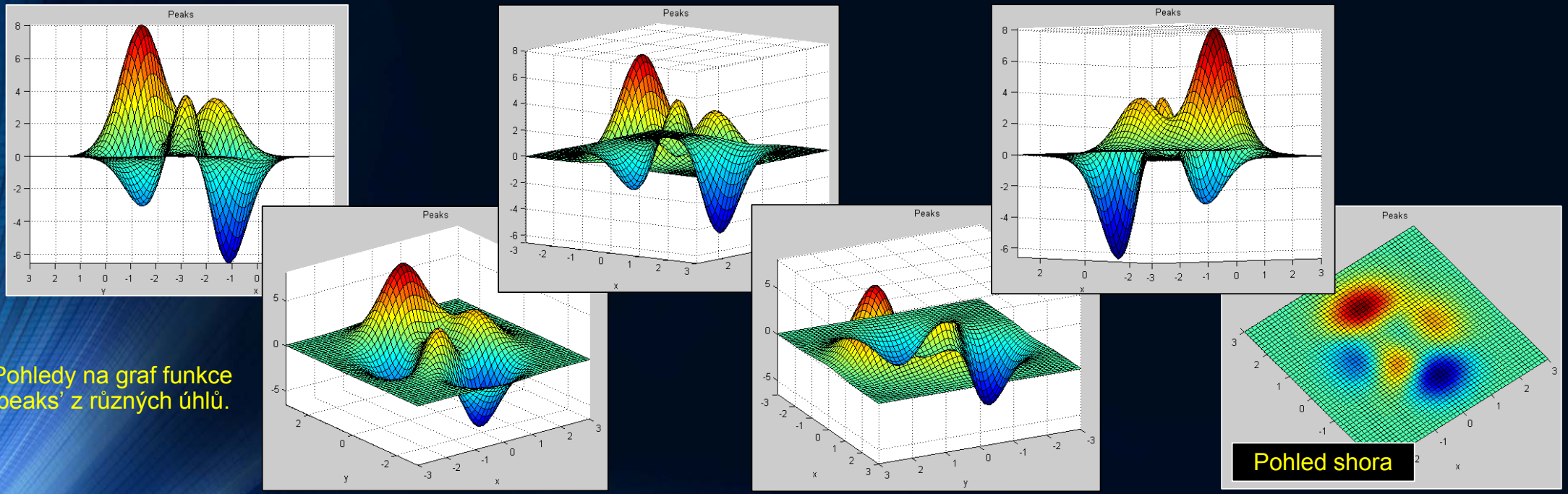


Další možnosti a varianty demonstračních funkcí viz učebnice, demo nebo manuál.

Práce v příkazovém okně (Command Window) :

3D grafika - různé pohledy na graf funkce

Lze dosáhnout manuálně pomocí menu v ‘toolbaru’ Figure nebo příkazem **view**
(viz učebnice, nápověda nebo manuál).



Pohledy na graf funkce ‘peaks’ z různých úhlů.

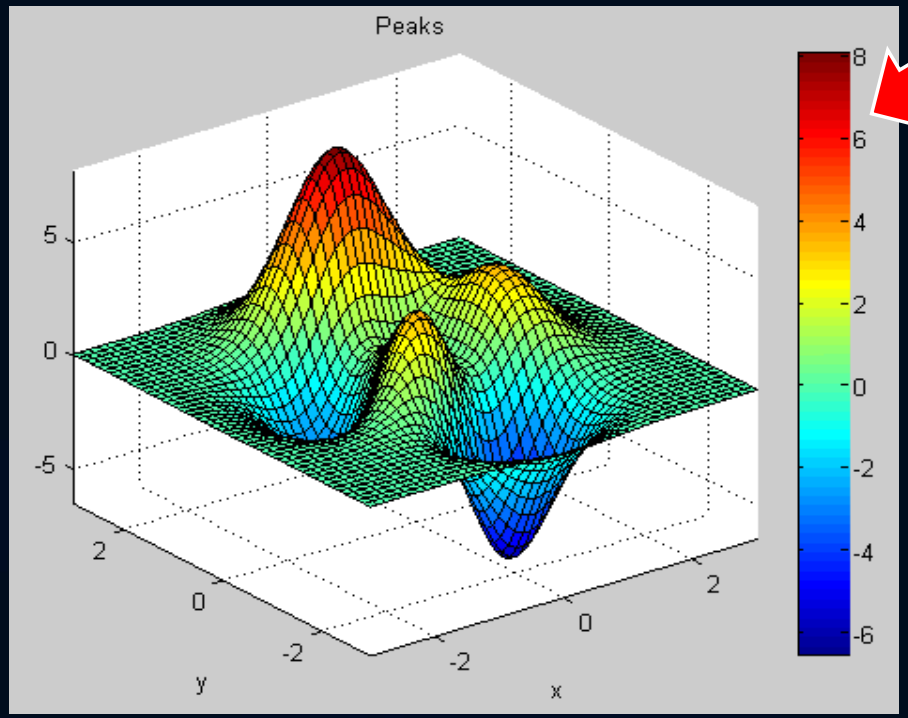
Práce v příkazovém okně (Command Window) :

3D grafika - doplňkové funkce grafiky

- **colorbar** … přidá informační sloupec s použitou barevnou paletou

```
>> peaks; colorbar
```

Prosté použití příkazu „colorbar“ přidá informační sloupec se standardně přednastavenou barevnou paletou.

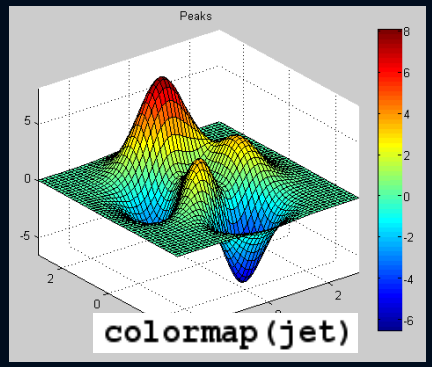


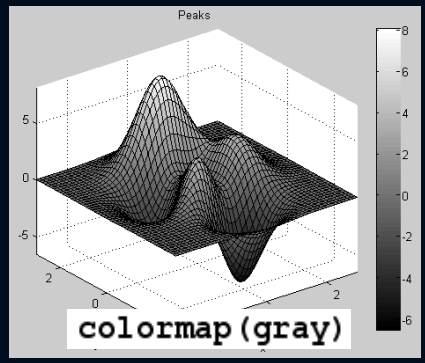
Práce v příkazovém okně (Command Window) :

3D grafika - doplňkové funkce grafiky

- **colormap (M)** … parametrem **M** lze zvolit barevnou paletu grafu

Matlab nabízí řadu přednastavených palet např. :
jet, grey, cool, autumn, …
(viz nápověda příkazu „graph3d“).





Příklady :

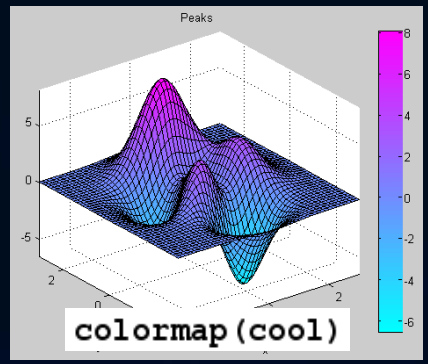
```
>> peaks; colorbar; colormap(jet)
```

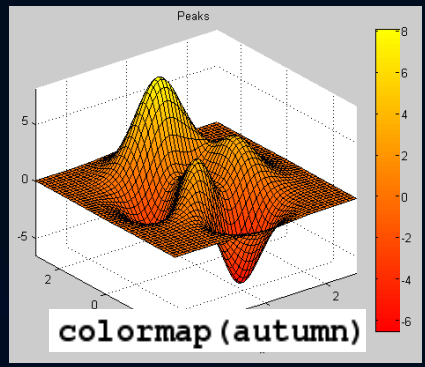
```
colormap(gray)
```

```
colormap(cool)
```

```
colormap(autumn)
```

```
colormap('default')
```
… standardní nastavení

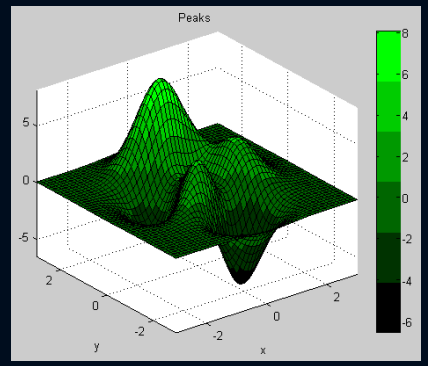




Práce v příkazovém okně (Command Window) :

3D grafika - doplňkové funkce grafiky

- **colormap (M)** … parametrem **M** lze také zvolit individuální barevnou paletu grafu



Příklady :

```
>> peaks; colorbar
```

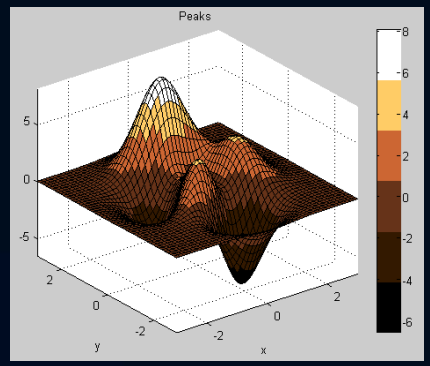
```
>> M=[0 0 0; 0 .2 0; 0 .4 0;...
      0 .6 0; 0 .8 0; 0 1 0];
>> colormap(M)
```

Barevnou paletu lze nastavit individuálně volbou matice **M** .
Matice má tolik řádků, kolik barev v paletě nastavíme.
Každý řádek obsahuje trojici čísel v rozsahu 0 (0%) až 1 (100%) udávající RGB (red-green-blue) kód barvy.

Tedy : 0 0 0 … černá
1 0 0 … červená
atd.
1 1 1 … bílá

```
>> peaks; colorbar
```

```
>> M=[0 0 0; .2 .1 0; .4 .2 .1;...
      .8 .4 .2; 1 .8 .4; 1 1 1];
>> colormap(M)
```



Práce v příkazovém okně (Command Window) :

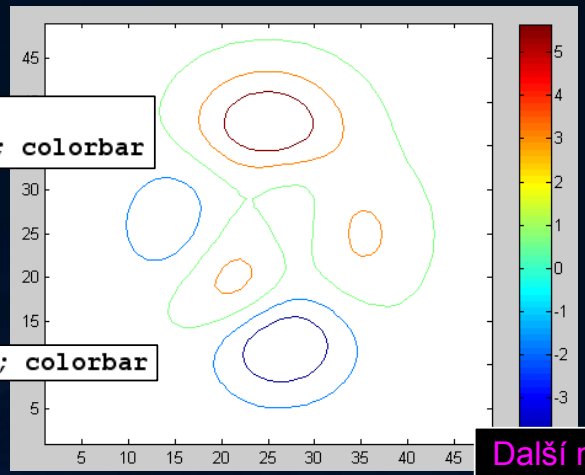
3D grafika - vrstevnicové grafy

- **contour** (**Xm, Ym , Zm, n**) … vykreslí 2D vrstevnicový graf funkce { **Xm**, **Ym**, **Zm** } s **n** rovnoměrně rozloženými hladinami
- **contour3** (**Xm, Ym , Zm, n**) … varianta 3D vrstevnicového grafu s **n** vrstevnicemi
- **contourf** (**Xm, Ym , Zm, n**) … varianta vyplněného 2D vrstevnicového grafu s **n** vrstevnicemi
- **clabel** (**C, h**) … umístí do grafu numerické popisky vrstevnic parametry **C** a **h** předem připravit např.: **[C, h] = contour** (**Xm, Ym , Zm, n**)

Příklady :

(další příklady i na následující straně)

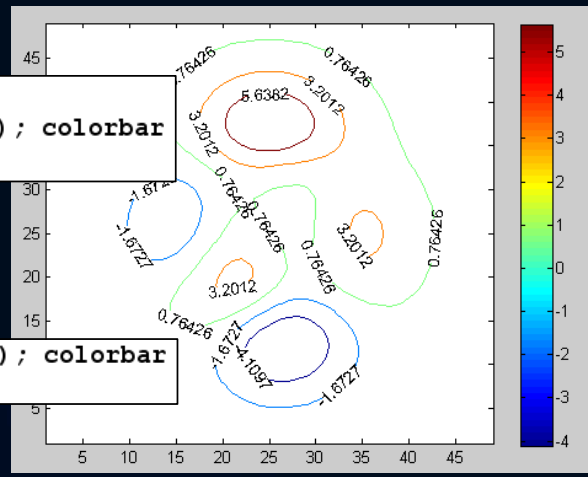
```
>> [x,y,z]=peaks;
>> contour(x,y,z,5); colorbar
```



resp. :

```
>> contour(peaks,5); colorbar
```

```
>> [x,y,z]=peaks;
>> [C,h]=contour(x,y,z,5); colorbar
>> clabel(C,h)
```



resp. :

```
>> [C,h]=contour(peaks,5); colorbar
>> clabel(C,h)
```

Další možnosti viz učebnice, help nebo manuál.

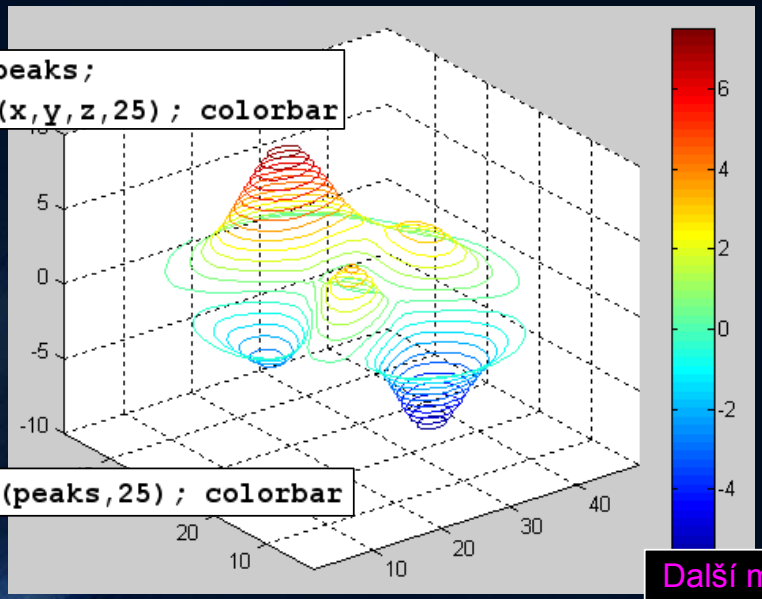
Práce v příkazovém okně (Command Window) :

3D grafika - vrstevnicové grafy

- **contour3** (**X**m, **Y**m , **Z**m, **n**) … varianta 3D vrstevnicového grafu s **n** vrstevnicemi
- **contourf** (**X**m, **Y**m , **Z**m, **n**) … varianta vyplněného 2D vrstevnicového grafu s **n** vrstevnicemi

Příklady :

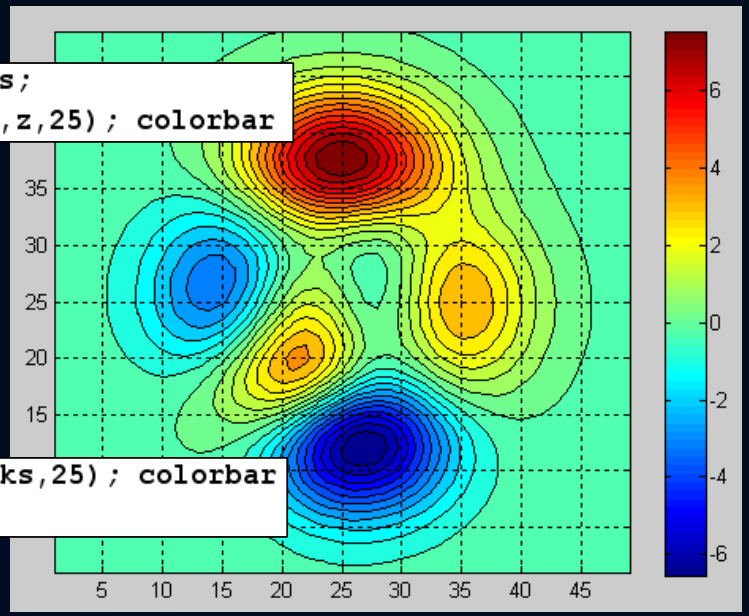
```
>> [x,y,z]=peaks;
>> contour3(x,y,z,25); colorbar
```



resp. :

```
>> contour3(peaks,25); colorbar
```

```
>> [x,y,z]=peaks;
>> contourf(x,y,z,25); colorbar
```



resp. :

```
>> contourf(peaks,25); colorbar
>> grid on
```

Další možnosti viz učebnice, help nebo manuál.

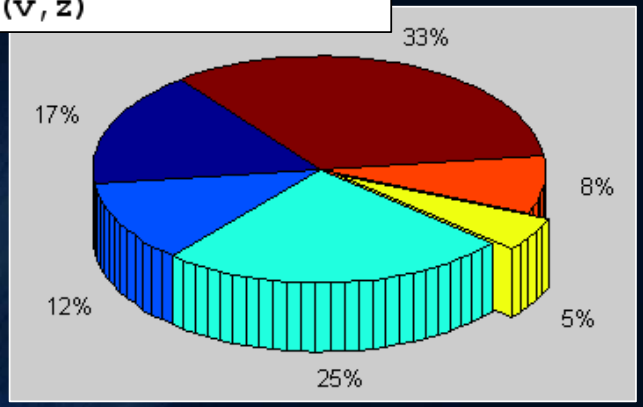
Práce v příkazovém okně (Command Window) :

3D grafika - speciální typy grafů

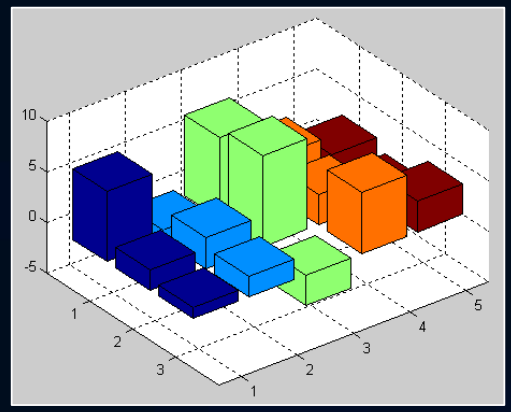
- **pie3** (**v**, **z**) … varianta 3D koláčového grafu, **v** parametr velikostí jednotlivých částí, nepovinný parametr **z** je vektor označující zvýrazněné části částečným vyčleněním
- **bar3** (**M**), resp. **bar3h** (**M**) … varianta 3D sloupcového grafu (ve vertikální, resp. horizontální dispozici)

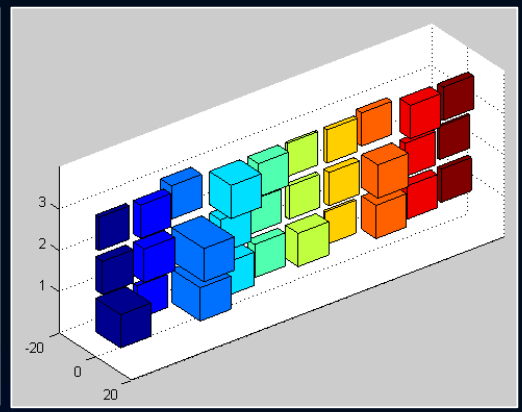
Příklady :

```
>> v=[1 0.7 1.5 0.3 0.5 2];
>> z=[0 0 0 1 0 0];
>> pie3(v,z)
```




```
>> M=[7 2 1;1 3 2;8 9 -3;4 3 6; 2 1 3];
>> bar3(M'), figure, bar3h([2*M;M]')
```





Další grafické možnosti Matlabu viz učebnice, manuál nebo nápověda.

3D grafika – příklady

Příklad 1

Nakreslete graf funkce :

$$z = e^{-x^2} + e^{-y^2} \quad , \quad x \in < -2 ; 2 >$$

$$, \quad y \in < -2{,}5 ; 2{,}5 >$$

vyznačte vrstevnice závislosti.

Dokážete si předem přibližně představit
jak bude vypadat graf této funkce ?

Spolupracujte při řešení se svými kolegy.
Raďte si, diskutujte, hledejte různé varianty.

3D grafika – příklady

Příklad 1

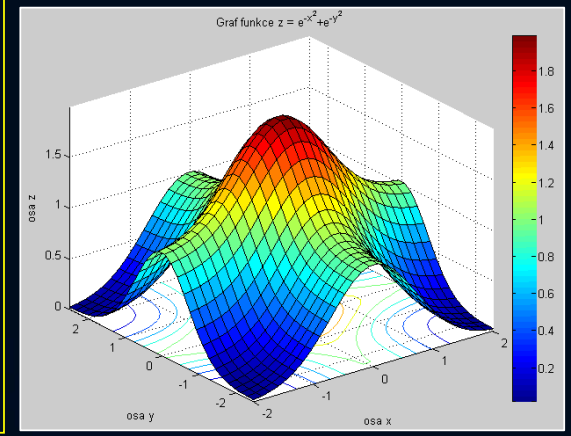
Nakreslete graf funkce :

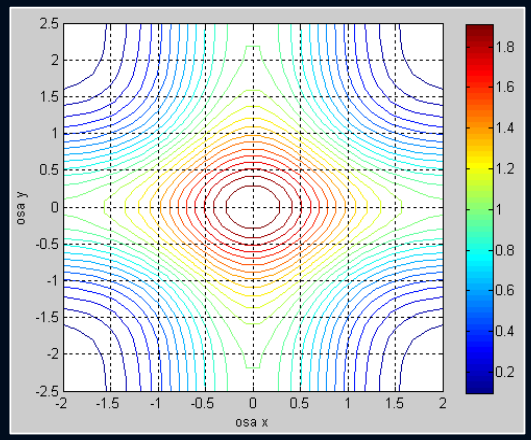
$$z = e^{-x^2} + e^{-y^2} \quad , \quad x \in \langle -2\,;\,2 \rangle \quad , \quad y \in \langle -2{,}5\,;\,2{,}5 \rangle$$

vyznačte vrstevnice závislosti.

Řešení :

```
>> x=linspace(-2,2,30);
>> y=linspace(-2.5,2.5,30);
>> [X,Y]=meshgrid(x,y);
>> Z=exp(-X.^2)+exp(-Y.^2);
>> surfc(X,Y,Z); colorbar
>> xlabel('osa x')
>> ylabel('osa y')
>> zlabel('osa z')
>> title('Graf funkce z = e^{-x^2}+e^{-y^2}')
>> axis tight
```






```
>> figure; contour(X,Y,Z,25)
>> axis tight; grid on; colorbar
>> xlabel('osa x'); ylabel('osa y')
```

Dokázali byste najít analyticky extrém této funkce ? Jaká je jeho hodnota ?
Srovnejte s vykresleným grafem.

3D grafika – příklady

Příklad 2

Nakreslete graf funkce :

$$f(x, y) = x^3 + y^3 - 3xy \quad , \quad x, y \in \langle -2 ; 3 \rangle$$

vyznačte vrstevnice závislosti
a rozhodněte o extrémech zadané funkce.

- Řešte pomocí grafických funkcí Matlabu
- Řešte analytickým výpočtem
- Oba postupy srovnejte

Spolupracujte při řešení se svými kolegy.
Raďte si, diskutujte, hledejte různé varianty.

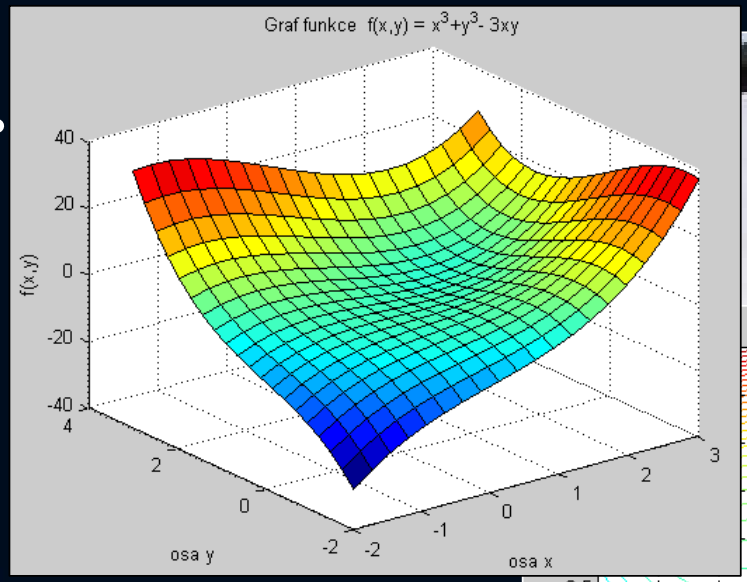
3D grafika – příklady

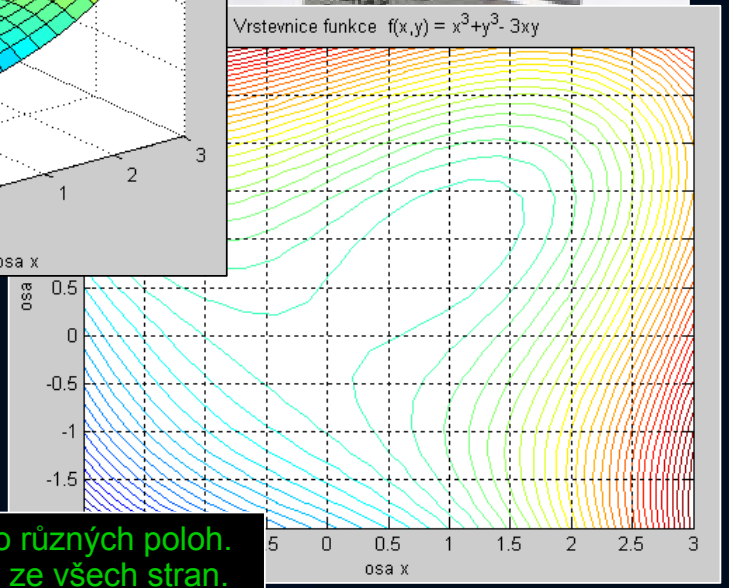
Příklad 2

Nakreslete graf funkce :

$$f(x,y) = x^3 + y^3 - 3xy \quad , \quad x, y \in \langle -2 ; 3 \rangle$$

vyznačte vrstevnice závislosti
a rozhodněte o extrémech zadané funkce.





Řešení :

(grafickými prostředky)

```
>> [a,b]=meshgrid(linspace(-2,3,20));
>> c=a.^3+b.^3-3*a.*b; surf(a,b,c)
>> xlabel('osa x'); ylabel('osa y')
>> zlabel('f(x,y)')
>> title('Graf funkce  f(x,y) = x^3+y^3- 3xy')
```

Zajímavá varianta použití příkazu meshgrid.

```
>> figure; contour(a,b,c,50); grid on
>> xlabel('osa x'); ylabel('osa y')
>> title('Vrstevnice funkce  f(x,y) = x^3+y^3- 3xy')
```

Otočte si manuálně graf do různých poloh.
Prohlédněte si ho pozorně ze všech stran.

3D grafika – příklady

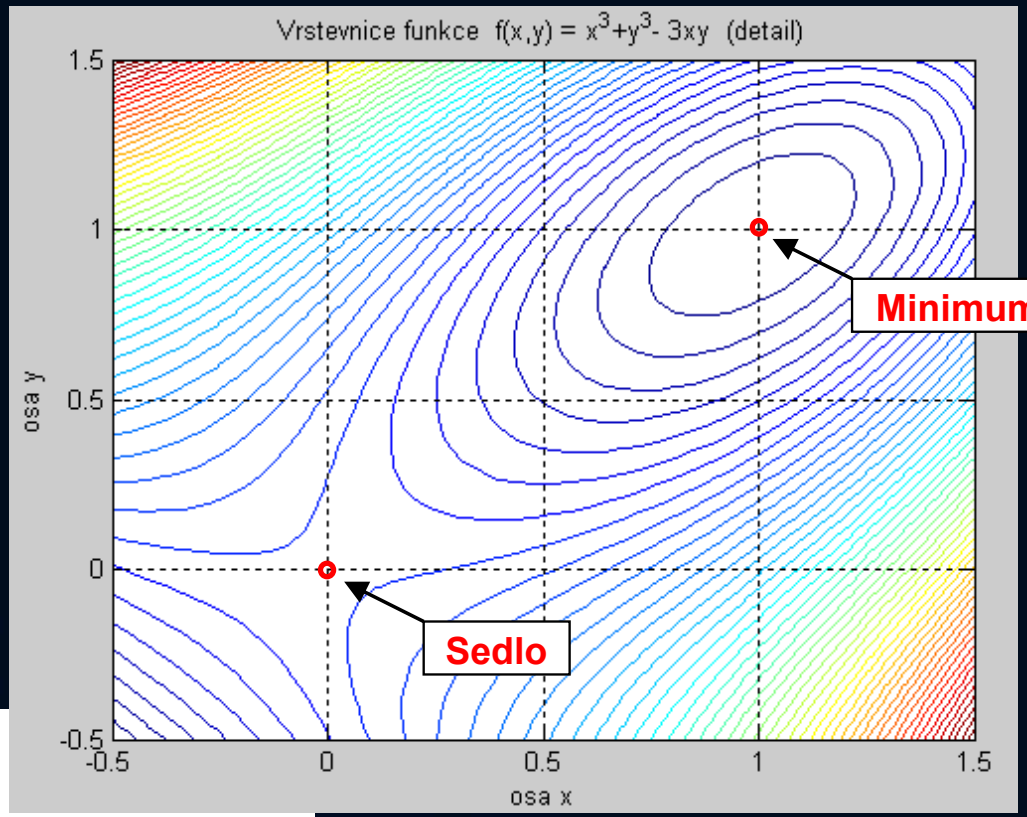
Příklad 2

Nakreslete graf funkce :

$$f(x,y) = x^3 + y^3 - 3xy \quad , \quad x, y \in \langle -2\,;\,3 \rangle$$

vyznačte vrstevnice závislosti
a rozhodněte o extrémech zadané funkce.

Detail chování funkce v singulárních bodech.




```
>> [x,y]=meshgrid(linspace(-0.5,1.5,50));
>> figure; contour(x,y,x.^3+y.^3-3*x.*y,50)
>> grid on; xlabel('osa x'); ylabel('osa y')
>> title('Vrstevnice funkce  f(x,y) = x^3+y^3- 3xy  (detail)')
```

3D grafika – příklady

Příklad 2

Nakreslete graf funkce :

$$f(x,y) = x^3 + y^3 - 3xy \quad , \quad x, y \in \langle -2 ; 3 \rangle$$

vyznačte vrstevnice závislosti
a rozhodněte o extrémech zadané funkce.

Řešení : (analytickými prostředky)

$$f(x,y) = x^3 + y^3 - 3xy$$

Dva singulární body :

$$\left.\begin{aligned} \frac{\partial f}{\partial x} &= 3x^2 - 3y = 0 \\ \frac{\partial f}{\partial y} &= 3y^2 - 3x = 0 \end{aligned}\right\} \rightarrow \begin{aligned} x &= y = 0 \\ x &= y = 1 \end{aligned}$$

Hessova matice :

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 6x & -3 \\ -3 & 6y \end{bmatrix}$$

$$H_{[0,0]} = \begin{bmatrix} 0 & -3 \\ -3 & 0 \end{bmatrix} \rightarrow \Delta(\lambda) = \det(\lambda E - H) = \det \begin{bmatrix} \lambda & 3 \\ 3 & \lambda \end{bmatrix} = \lambda^2 - 9 = 0$$

$$\lambda_1 = +3 \quad , \quad \lambda_2 = -3$$

V bodě [0, 0] je H indefinitní $\rightarrow$ lokální extrém typu sedlo.

$$H_{[1,1]} = \begin{bmatrix} 6 & -3 \\ -3 & 6 \end{bmatrix} \rightarrow \Delta(\lambda) = \det(\lambda E - H) = \det \begin{bmatrix} \lambda - 6 & 3 \\ 3 & \lambda - 6 \end{bmatrix} = \lambda^2 - 12\lambda + 27 = 0$$

$$\lambda_1 = +9 \quad , \quad \lambda_2 = +3$$

V bodě [1, 1] je H pozitivně definitní $\rightarrow$ lokální extrém typu minimum.

(Globální extrém funkce neexistuje.)

3D grafika – příklady

Příklad 3

Nakreslete graf funkce :

$$f(x,y)=\frac{2}{5+3x^2+4y}, \quad x \in <-2\,;\,2> \quad y \in <-3\,;\,0>$$

Dokážete alespoň přibližně předem odhadnout průběh této funkce ?

Není to zase tak jednoduché jak se na první pohled zdá.
Zamyslete se nad tím dřív než se podíváte na řešení.

3D grafika – příklady

Příklad 3

Řešení :

Nakreslete graf funkce :

$$f(x,y)=\frac{2}{5+3x^2+4y}, \quad x\in<-2;2>, \quad y\in<-3;0>$$

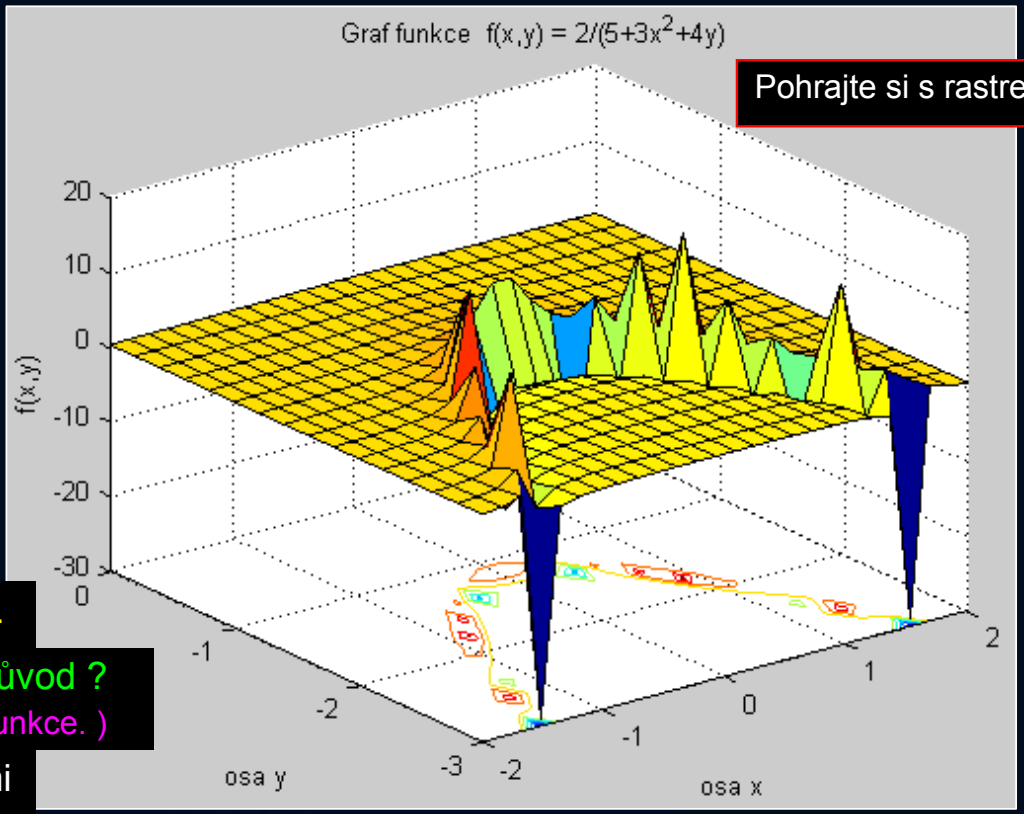
```
>> x=linspace(-2,2,26); y=linspace(-3,0,16);
>> [X,Y]=meshgrid(x,y); Z=2./(5+3*X.^2+4*Y);
>> surfc(X,Y,Z)
>> xlabel('osa x'); ylabel('osa y'); zlabel('f(x,y)')
>> title('Graf funkce  f(x,y) = 2/(5+3x^2+4y)')
```

Zobrazte graf s jinou sítí argumentů x, y.

Chová se to divně ? Dokážete vysvětlit důvod ?

(Klíčem je definiční obor funkce.)

Zamyslete se nad omezenými možnostmi diskrétní reprezentace.



Pohrajte si s rastrem.

3D grafika – příklady

Příklad 4

Rosenbrockova funkce

(tzv.banánová)

Nakreslete graf funkce :

$$f(x,y) = 100\,(y - x^2)^2 + (1-x)^2 \quad , \quad x, y \in \langle -2\,;\,2 \rangle$$

Minimum této funkce je v bodě [1, 1] . Nalezení jejího minima je poměrně obtížné, protože je v tomto bodě funkce velmi plochá.

Jedná se o standardní testovací funkci kvality a rychlosti konvergence optimalizačních algoritmů.

Dokážete určit extrém analytickými prostředky ?

3D grafika – příklady

Příklad 4

Rosenbrockova funkce

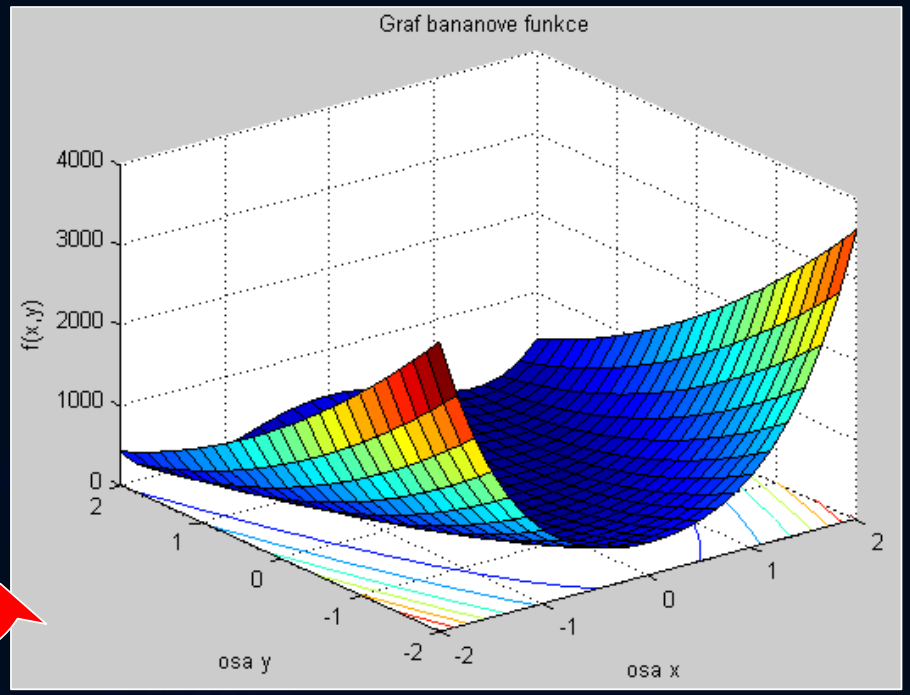
(tzv.banánová)

Nakreslete graf funkce :

$$f(x,y) = 100\,(y - x^2)^2 + (1-x)^2 \quad , \quad x, y \in \langle -2\,;\,2 \rangle$$

Řešení :

```
>> [X,Y]=meshgrid(linspace(-2,2,25));
>> Z=100*(Y-X.^2).^2+(1-X).^2;
>> surfc(X,Y,Z); xlabel('osa x')
>> ylabel('osa y'); zlabel('f(x,y)')
>> title('Graf bananove funkce')
>> figure; contour(X,Y,Z,40)
>> grid on; colorbar
>> xlabel('osa x'); ylabel('osa y')
>> title('Vrstevnice bananove funkce')
```

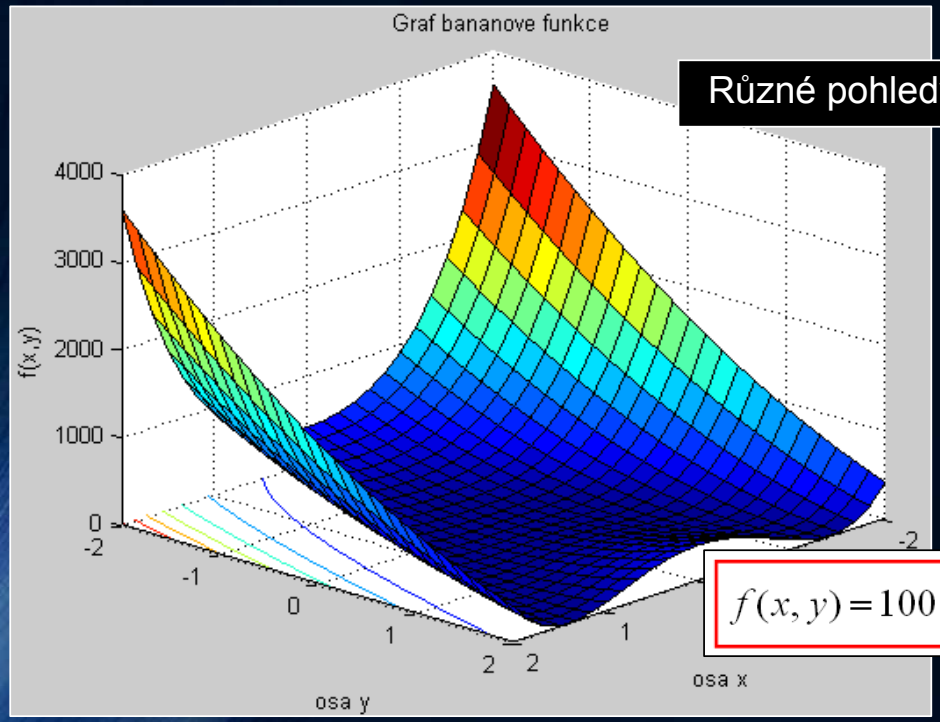


3D grafika – příklady

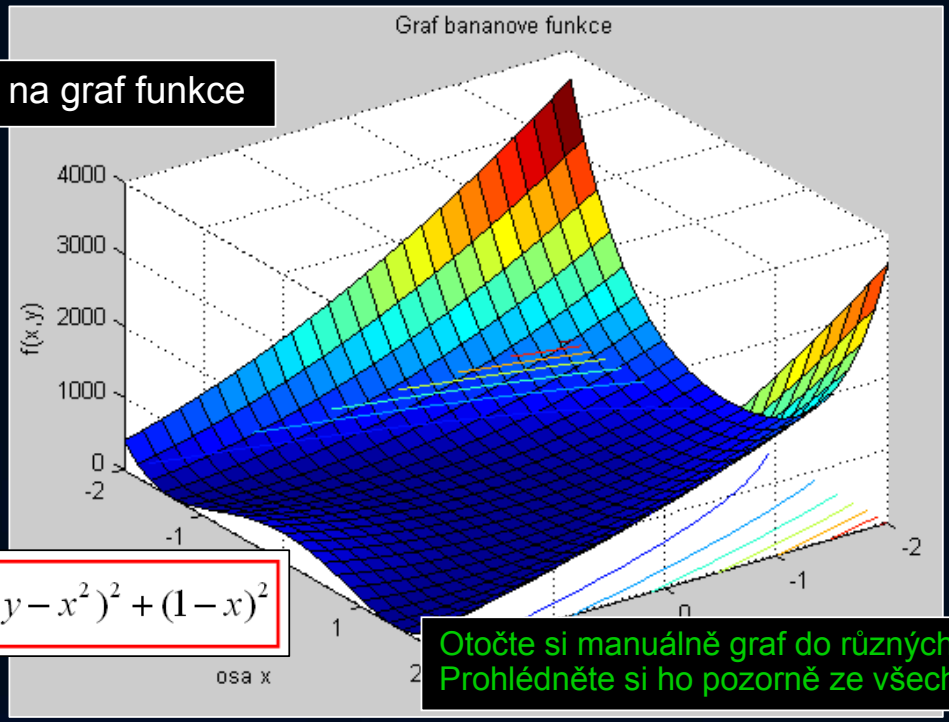
Příklad 4

Rosenbrockova funkce

(tzv.banánová)



Různé pohledy na graf funkce



$$f(x,y) = 100\,(y - x^2)^2 + (1-x)^2$$

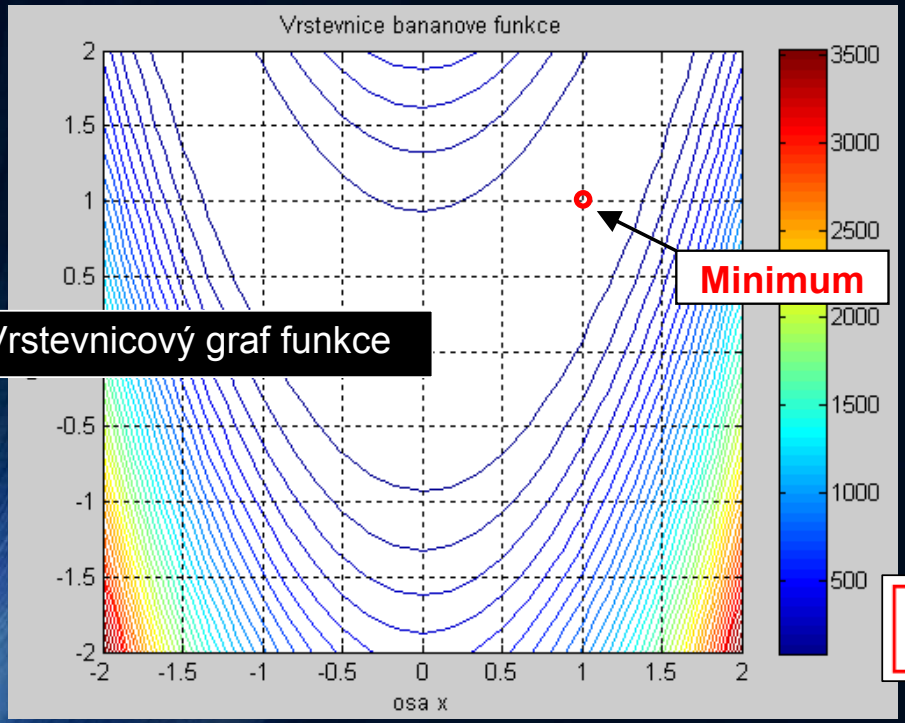
Otočte si manuálně graf do různých poloh.
Prohlédněte si ho pozorně ze všech stran.

3D grafika – příklady

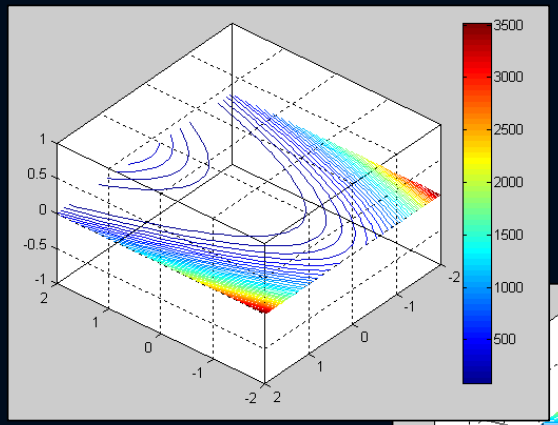
Příklad 4

Rosenbrockova funkce

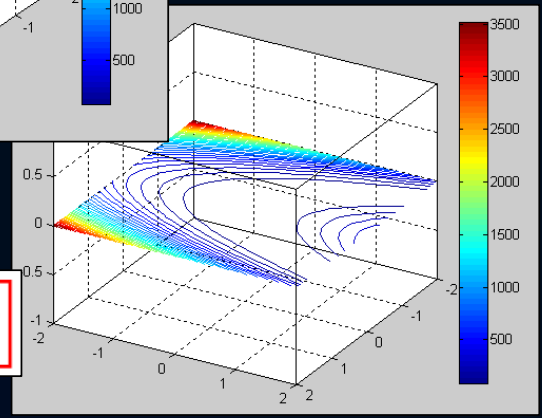
(tzv.banánová)



Vrstevnicový graf funkce



Různé pohledy



$$f(x,y) = 100\,(y - x^2)^2 + (1 - x)^2$$

3D grafika – příklady

Příklad 5 Rastriginova funkce

Nakreslete graf funkce :

$$f(x,y) = x^2 + y^2 - 10\,(\cos(2\pi x) + \cos(2\pi y)) \quad , \quad x, y \in \langle -2\,;\,2 \rangle$$

Opět další „oříšek“ pro numerické optimalizační metody.

Minimum této funkce je v bodě [0, 0] . Nalezení jejího absolutního minima je však poměrně obtížné, protože má mnoho lokálních extrémů.

Dokážete rozhodnout zda má tato funkce ve svém definičním oboru konečný počet extrémů ?

3D grafika – příklady

Příklad 5

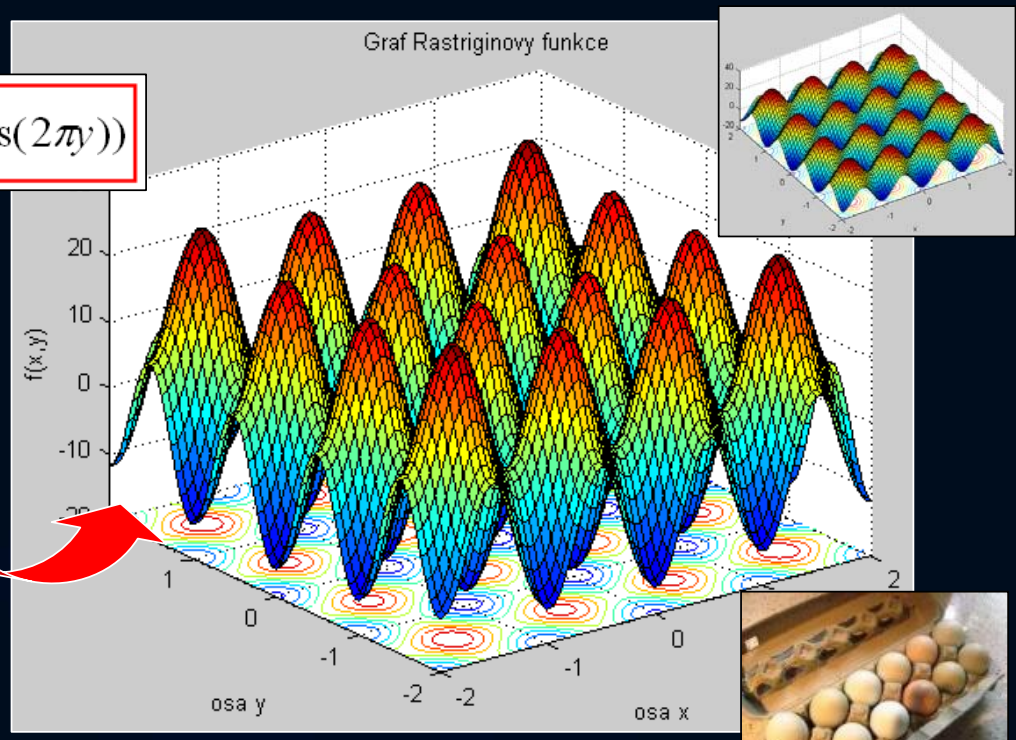
Rastriginova funkce

$$f(x,y) = x^2 + y^2 - 10\,(\cos(2\pi x) + \cos(2\pi y))$$

Řešení :

```
>> [X,Y]=meshgrid(linspace(-2,2,70));
>> Z=X.^2+Y.^2-10*(cos(2*pi*X)+cos(2*pi*Y));
>> surfc(X,Y,Z); xlabel('osa x')
>> ylabel('osa y'); zlabel('f(x,y)')
>> title('Graf Rastriginovy funkce')
```

```
>> figure; contour(X,Y,Z,20)
>> grid on; colorbar
>> xlabel('osa x'); ylabel('osa y')
>> title('Vrstevnice Rastriginovy funkce')
```

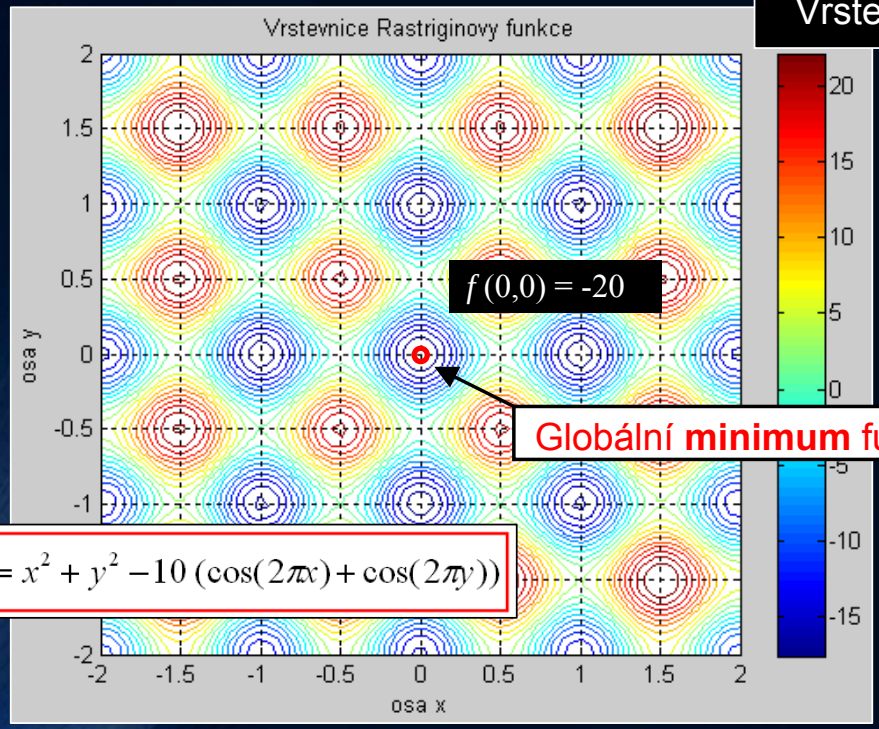


Nepřijde vám to jako futrál na vajíčka ?

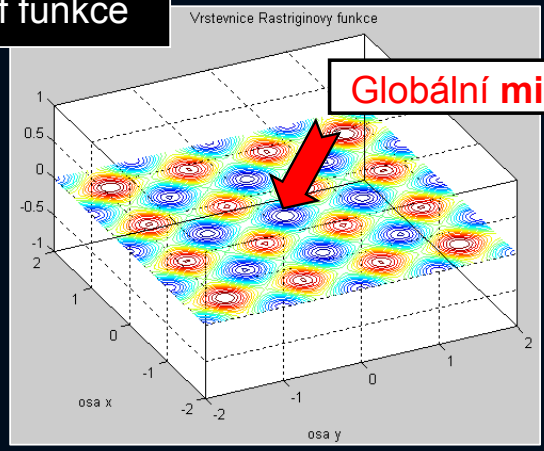
3D grafika – příklady

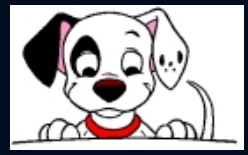
Příklad 5 Rastriginova funkce

Vrstevnicový graf funkce



$$f(x,y) = x^2 + y^2 - 10\,(\cos(2\pi x) + \cos(2\pi y))$$





Je zajímavé, že takto členitá funkce má pouze jediné globální (absolutní) minimum.

Většina numerických optimalizačních metod se ‘chytá’ pouze lokálních extrémů. Je proto nutné při optimalizaci takto členitých funkcí provést nejprve analýzu a klasifikaci lokálních extrémů funkce.