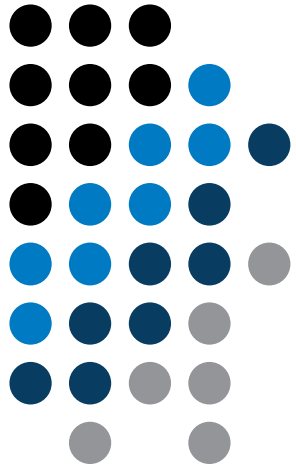
A0B17MTB – Matlab

Část #5

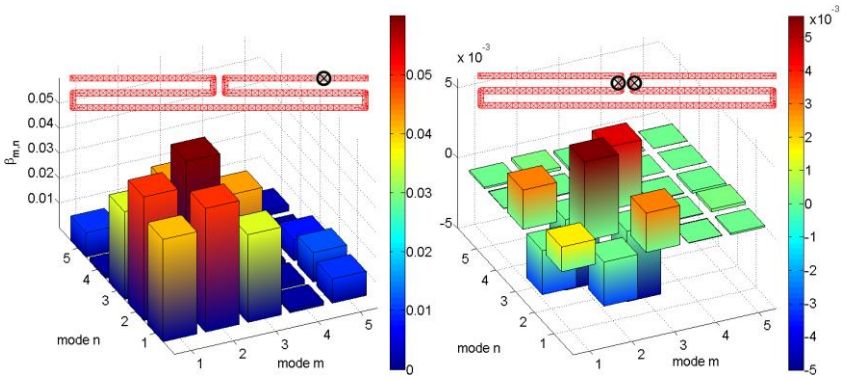
Miloslav Čapek
miloslav.capek@fel.cvut.cz

Filip Kozák a Viktor Adler

Katedra elektromagnetického pole
B2-626, Dejvice



Naučíte se …



Větvení programu #2

Vizualizace v Matlabu #1

Ladění běhu programu #1

Větvení programu – `switch` / `case`

- odpovídá `promenna` jednomu z (typicky mnoha) `hodnot`?
- výrazy v části `otherwise` jsou provedeny v případě, že žádný předchozí případ nenastal (srovnejte s `else` u příkazu `if`)

- vhodný pro zpracování textových podmínek
  - pokud Vás v detailech zajímá, kdy `if` a kdy `switch`, zkuste navštívit fóra na stránkách `blogs.mathworks.com`

- je vhodné konstrukci vždy zakončit případem `otherwise`

```
c = randi(1e2);
switch mod(c,2)
 case 1
    disp('c je liche');
 case 0 & c > 10
    disp('sude,>10');
 otherwise
    disp('sude,<=10')
end
```

Větvení programu – `switch / case`

450 s ↑

- vytvořte skript, který vypočte podle Pythagorovy věty zbývající stranu pravoúhlého trojúhelníka
  - zbývající dvě strany jsou zadané, je zadaný textový řetězec označující typ neznámé strany (odvěsna: `'od'`, přepona: `'prep'`)

```
%% HINT:
% input variables will be here
%(including type of unknown side)
switch aaa % aaa denotes the type of unknown side
  case yyy % calculation for the first type of side
%     calculation1
  case zzz % calculation for the second type of side
%     calculation2
  otherwise % unknown type
%     return empty (default) values
end
```

Co daný skript dělá?

300 s ↑

- pokuste se odhadnout, co daný skript vrací jako hodnotu `logResult` v závislosti na vstupní proměnné `vec` (jde o vektor)
  - dokázali byste říct, zda existuje funkce v Matlabu, která umí totéž?

```
% vec is a given vector

logResult = false;
m = 1;
while (m <= length(vec)) && (logResult == false)
    if vec(m) ~= 0
        logResult = true;
    end
    m = m + 1;
end
```

Co daný skript dělá?

300 s ↑

- pokuste se odhadnout, co daný skript vrací jako hodnotu `logResult` v závislosti na vstupní proměnné `mat` (jde o matici)
  - dokázali byste říct, zda existuje funkce v Matlabu, která umí totéž?

```
% mat is a given matrix
count  = 0;
[mRows, nColumns] = size(mat);
for m = 1:mRows
    for n = 1:nColumns
        if mat(m,n) ~= 0
            count = count + 1;
        end
    end
end
logResult = count == numel(mat);
```

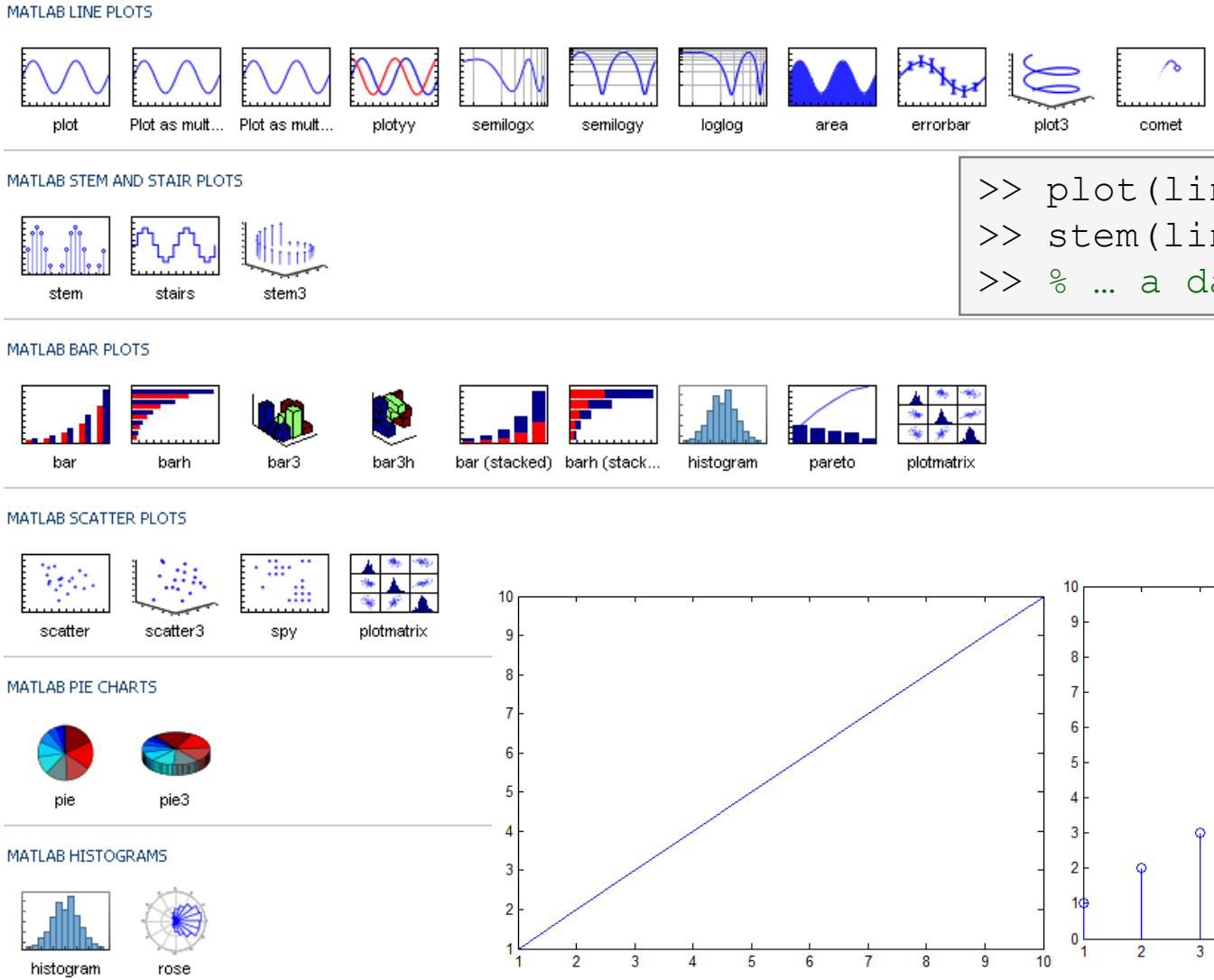
Úvod do vizualizace

- již jsme se okrajově seznámili s několika grafy v Matlabu
  - `plot`, `stem`, `bar`, `hist`, `surf`

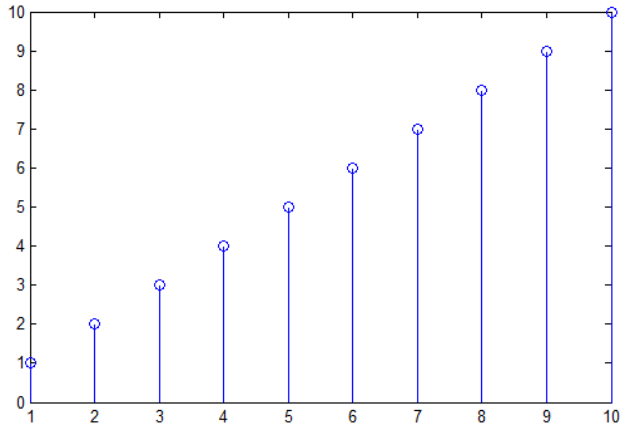
- obecně lze v Matlabu grafické funkce využívat na
  - vyšší úrovni
    - přistupujeme k jednotlivým funkcím a vlastnosti objektů ovlivňujeme pouze parametry volané funkce
    - do cca. 9-10 týdnu semestru
  - nižší úrovni
    - vyvoláváme přímo objekty a pracujeme s nimi
    - vyžaduje znalost handle grafiky v Matlabu (OOP)
    - otevírá rozsáhlé možnosti personalizace vizualizace

- podrobnosti naleznete:
  - Matlab → Graphics → 2-D and 3-D Plots → Plotting Basics

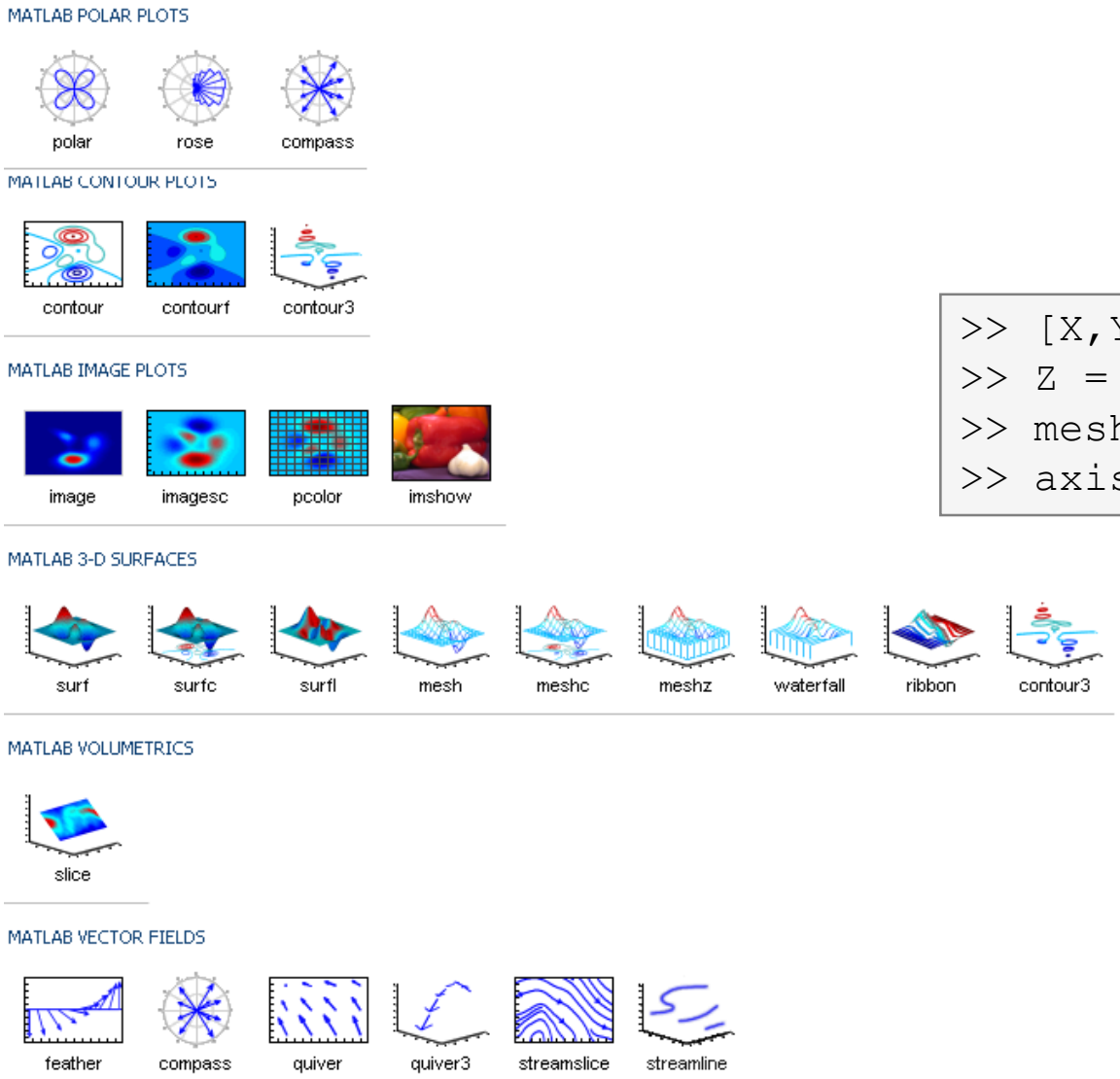
Vybrané grafy #1



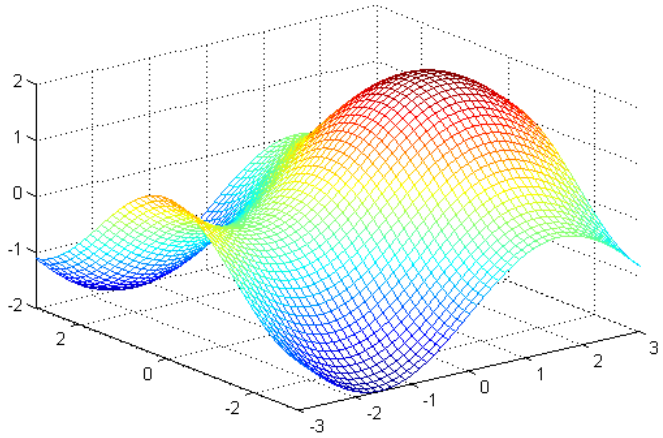

```
>> plot(linspace(1,10,10));
>> stem(linspace(1,10,10));
>> % … a dalsi
```

Vybrané grafy #2




```
>> [X,Y] = meshgrid(-3:.125:3);
>> Z = sin(X) + cos(Y);
>> mesh(X,Y,Z);
>> axis([-3 3 -3 3 -2 2]);
```



Vybrané funkce na úpravu grafů

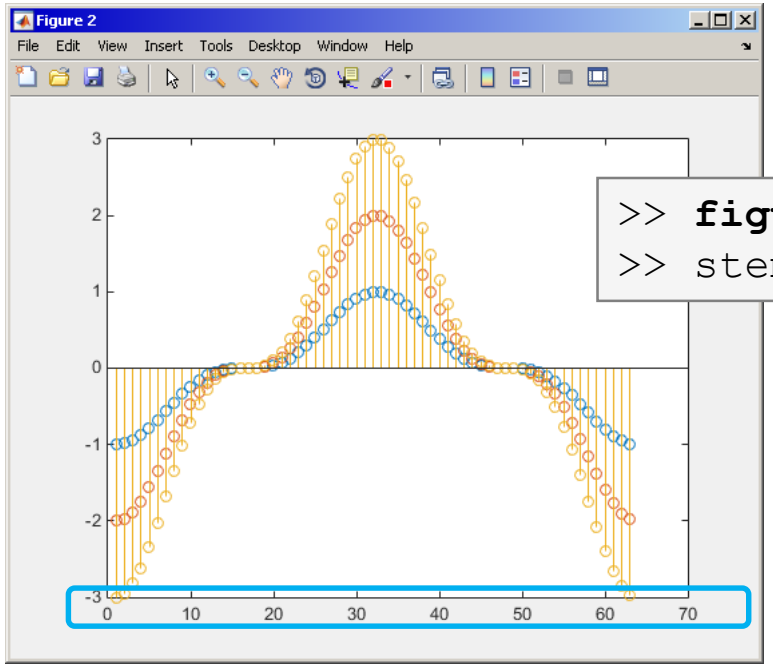
- v Matlabu lze vzniklé grafy upravovat mnoha způsoby, ty základní vlastnosti jsou:

funkce	popis
`title`	nadpis daných os (grafu)
`grid on, grid off`	zobrazí mřížku / schová mřížku
`xlim, ylim, zlim`	nastaví vlastní rozmezí pro osy
`xlabel, ylabel, ...`	popíše osy
`hold on`	umožní vykreslit další graf. prvky (grafy) do stejných os
`legend`	zobrazí legendu
`subplot`	funkce umožňuje otevření více grafů v jednom okně
`text`	přidá do grafu text
`gtext, ginput`	vložení textu pomocí myši, zadání bodu v grafu pomocí myši
a další	

figure

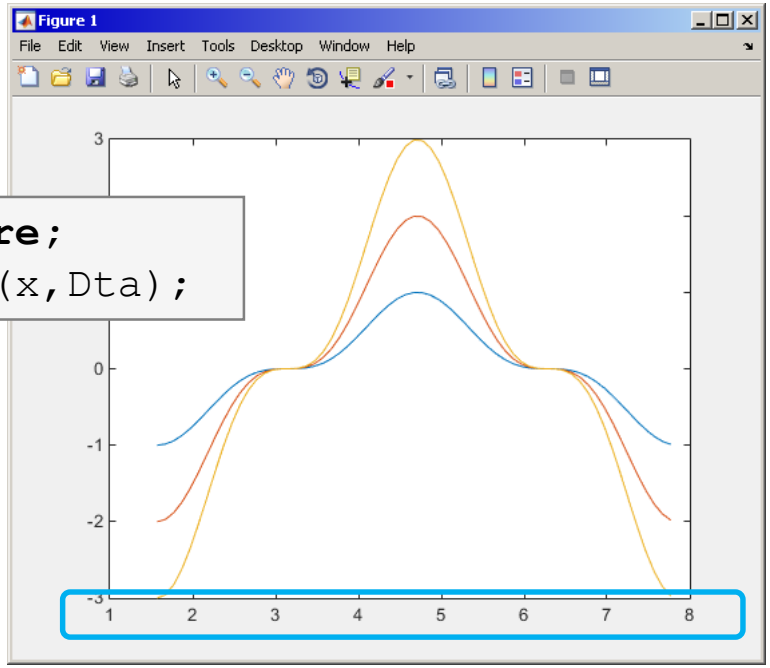
- `figure` otevře prázdné okno, do kterého lze vykreslovat grafy
  - funkci lze volat vč. přiřazení – pak vrací objekt třídy `Figure`

```
>> x = (0:0.1:2*pi) + pi/2;
>> Dta = -[1 2 3]'*sin(x).^3;
```




```
>> figure;
>> stem(Dta');
```

```
>> figure;
>> plot(x,Dta);
```

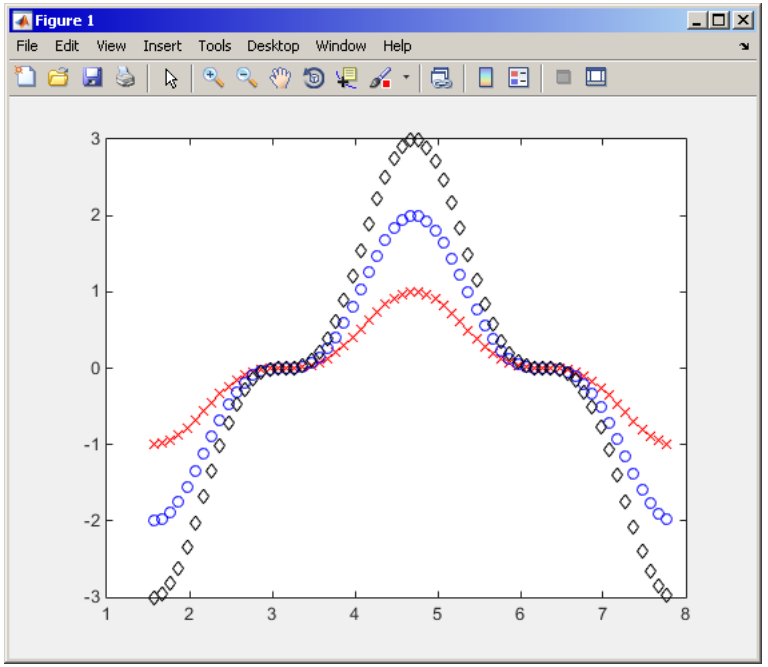


- lze vykreslovat i data v maticích (po sloupcích)
- nezapomínejte na data x-ové osy!

hold on

- funkce `hold on` umožní vykreslit více křivek do jednoho okna, lze vypnout jako `hold off`
- grafy `plot`, `plot3`, `stem` a celá řada další umožňují přidat volitelné parametry (jako textový řetězech)

```
x = (0:0.1:2*pi) + pi/2;
Dta = -[1 2 3]'*sin(x).^3;
figure;
plot(x, Dta(1,:), 'xr');
hold on;
plot(x, Dta(2,:), 'ob');
plot(x, Dta(3,:), 'dk');
```



LineSpec – nastavení křivek grafu

- co znamenají parametry u funkcí plot?
  - viz >> doc LineSpec
  - jde o nejčastěji nastavované parametry čar v grafech
    - barva čáry (lze zadávat i pomocí matice [R G B], kde R, G, B jsou mezi 0 a 1)
    - tvar značek křivky (*Markers*)
    - druh čáry
- od verze 2014b došlo k velkým změnám!

barva křivky	
'r'	červená
'g'	zelená
'b'	modrá
'c'	azurová
'm'	fialová
'y'	žlutá
'k'	černá
'w'	bíla

značka křivky	
'+'	plus
'o'	kroužek
'*'	hvězdička
'.'	plný bod
'x'	křížek
's'	čtverec
'd'	kosočtverec
'^'	trojúhelník
a další	>> doc LineSpec

```
plot(x,f,'bo-');
plot(x,f,'g*--');
```

```
figure('color', ...
    [.5 .1 .4]);
```

druh čáry	
'-'	{plná čára}
'--'	čárkovaná
':'	tečkovaná
'-.'	čerchovaná
'none'	bez čáry

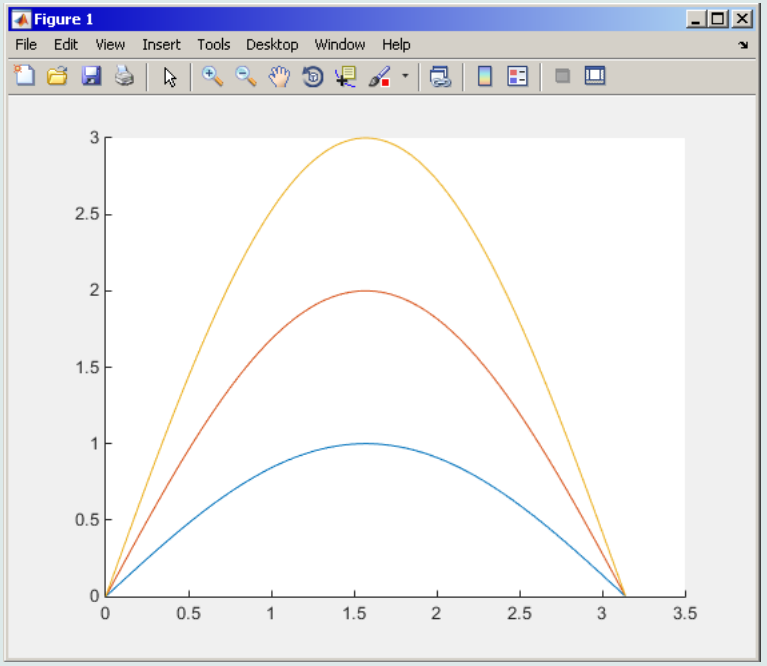
LineSpec – výchozí nastavení ve 2014b

- při vykreslování více křivek do jednoho grafu jsou automaticky použity barvy v předem daném pořadí
  - od verze 2014b došlo ke změně barevného schématu:
- při použití `hold on` již není nutné nastavovat barvu ke každé křivce zvlášť
  - je použito defaultní pořadí barev:

```
>> get(groot, 'DefaultAxesColorOrder')

% ans =
%
%            0    0.4470    0.7410
%       0.8500    0.3250    0.0980
%       0.9290    0.6940    0.1250
%       0.4940    0.1840    0.5560
%       0.4660    0.6740    0.1880
%       0.3010    0.7450    0.9330
%       0.6350    0.0780    0.1840
```

```
close all; clear; clc;
x = 0:0.01:pi;
figure;
hold on;
plot(x, 1*sin(x));
plot(x, 2*sin(x));
plot(x, 3*sin(x));
```

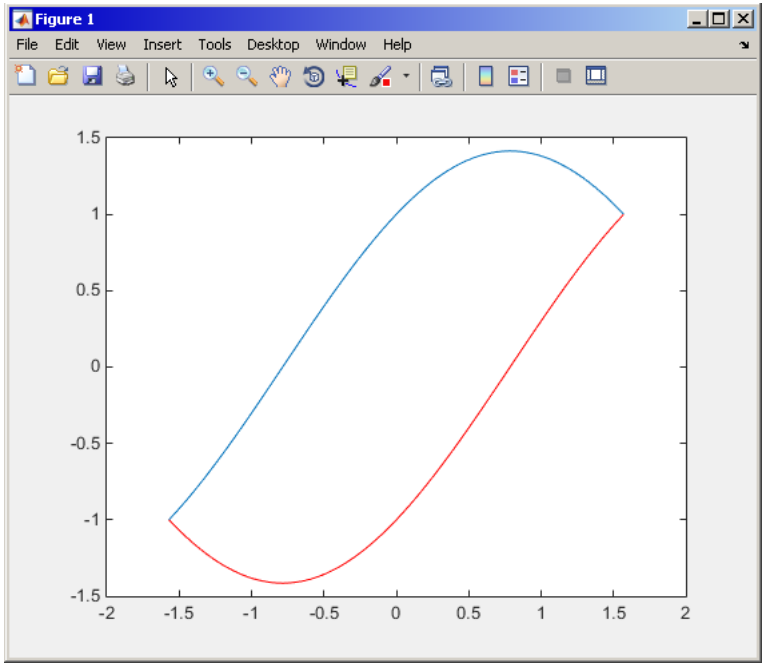


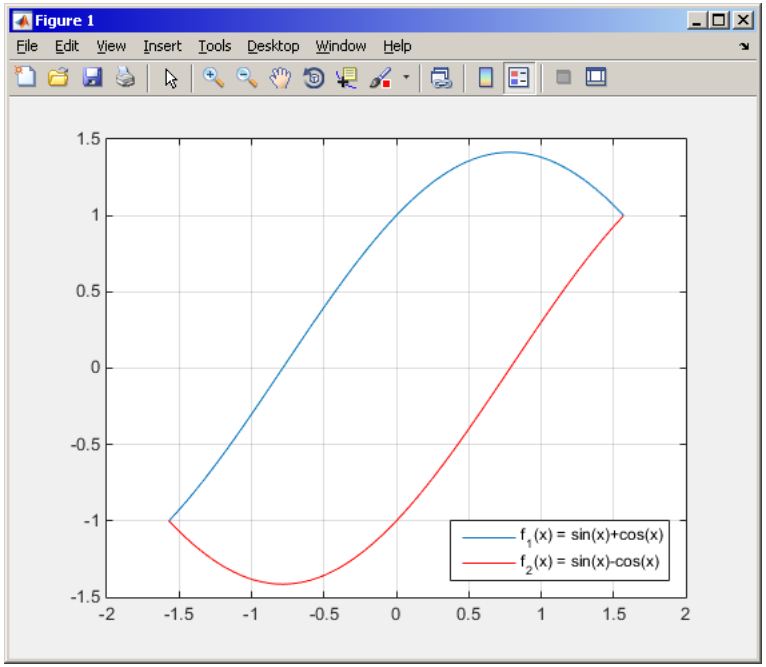
Vizualizace – legend, grid

```
x = -pi/2:0.01:pi/2;
f1 = sin(x) + cos(x);
f2 = sin(x) - cos(x);
```

```
plot(x, f1);
hold on;
plot(x, f2, 'r');
```

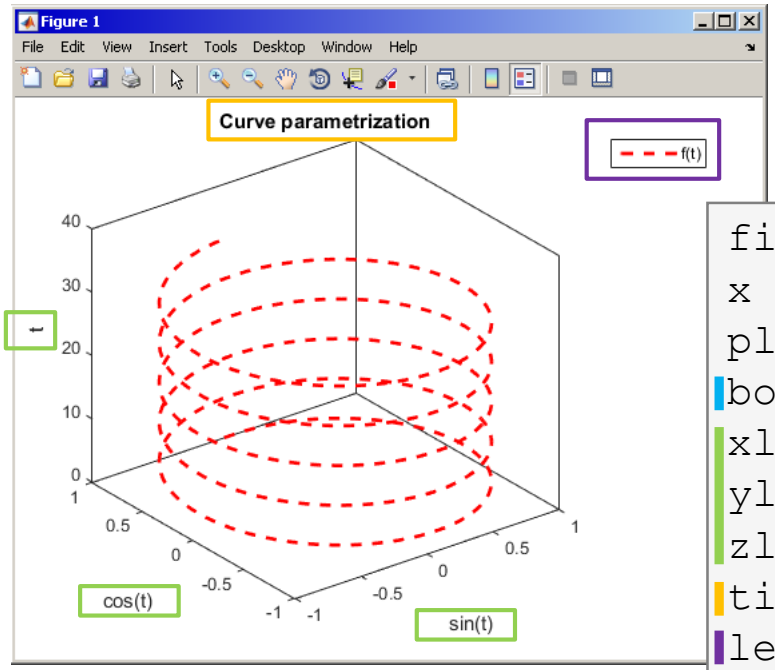
```
grid on;
legend('f_1(x) = sin(x)+cos(x)',...
    'f_2(x) = sin(x)-cos(x)',...
    'Location', 'southeast');
```





plot3

- příklad níže ukazuje vykreslení spirály pomocí parametrizace funkce
  - využíváme funkcí `xlabel`, `ylabel` a `zlabel` pro popis os
  - využíváme funkce `title` pro nadpis grafu
  - využíváme funkce `legend` pro popis křivky



- funkce `box` ohraničí celý graf

```
figure('color','w');
x = 0:0.05:10*pi;
plot3(sin(x),cos(x),x,'r--','LineWidth',2);
box on;
xlabel('sin(t)');
ylabel('cos(t)');
zlabel('t');
title('Curve parametrization')
legend('f(t)');
```

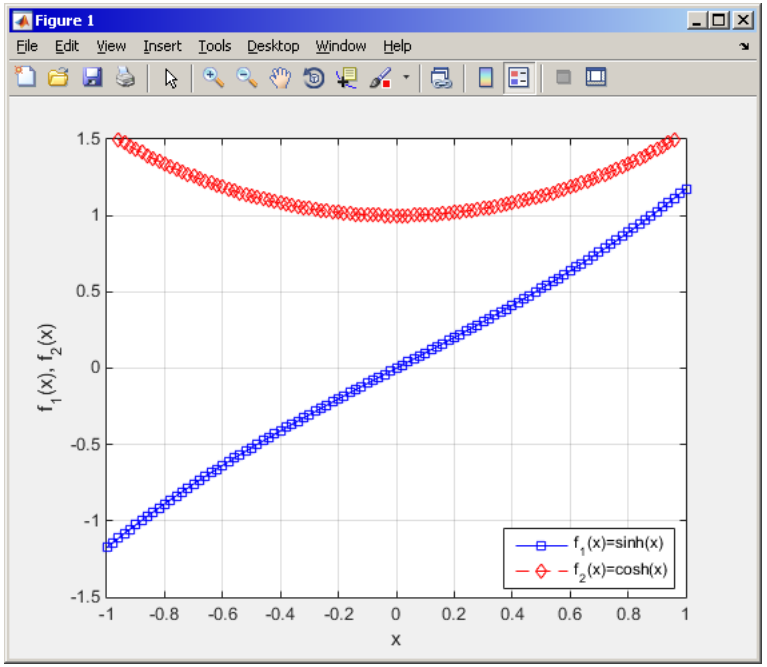
LineSpec – nastavení křivek grafu

450 s ↑

- vypočtěte následující dvě funkce v intervalu [-1,1] pro 101 hodnot:

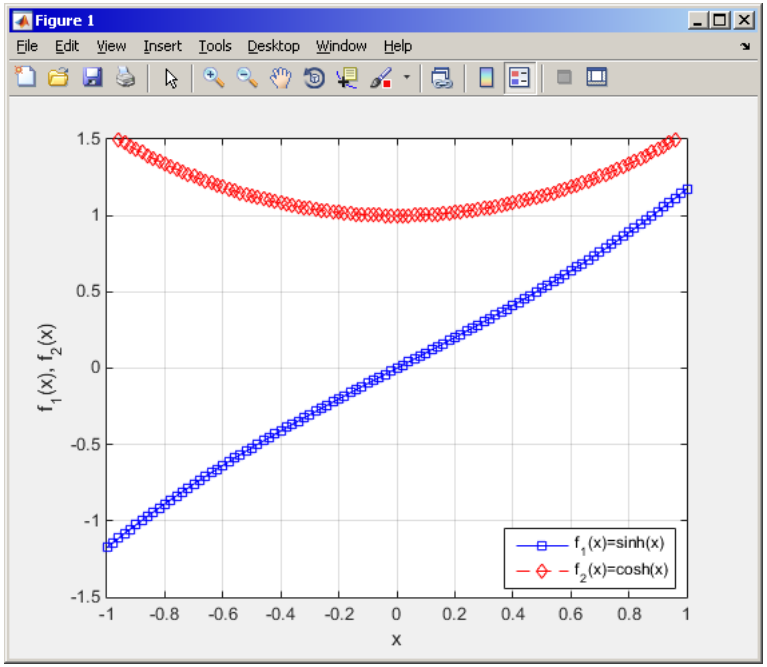
$$f_1(x) = \sinh(x), \qquad f_2(x) = \cosh(x)$$

- funkce vykreslete pomocí funkce `plot` tak, že
  - obě funkce budou vykresleny do jednoho grafu
  - první funkce bude vykreslena modře se značkami □ a plnou čarou
  - druhá funkce bude vykreslena červeně se značkami ◊ a čárkovaně
  - zobrazený rozsah osy *y* omezte na interval [-1.5, 1.5]
  - obrázku přidejte legendu, popisující obě funkce
  - popište osy (osa *x*: *x*, osa *y*: f_1, f_2)
  - graf bude mít mřížku



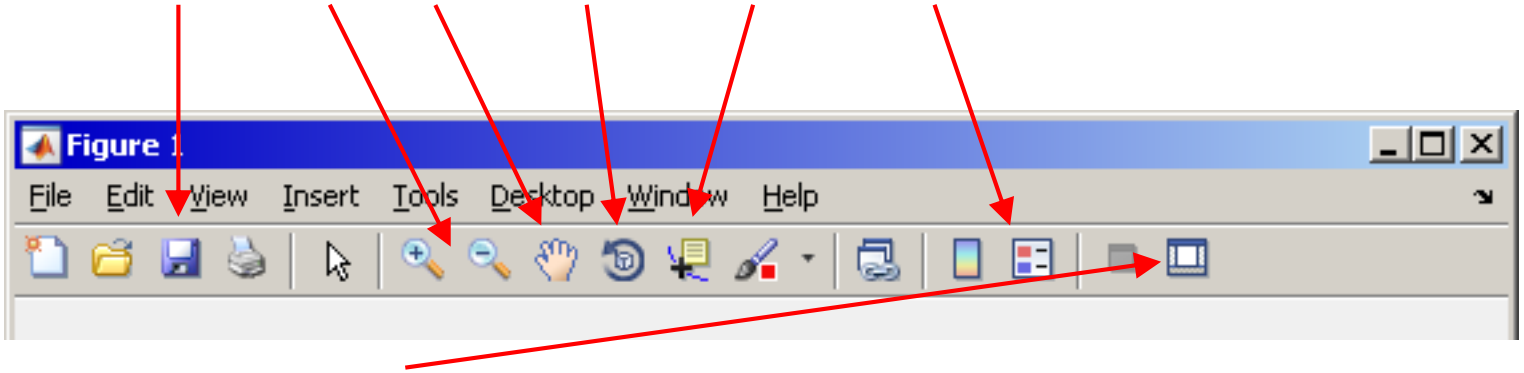
LineSpec – nastavení křivek grafu

$$f_1(x) = \sinh(x), \qquad f_2(x) = \cosh(x)$$



Vizualizace – `Plot tools`

- s grafem lze dále pracovat
  - `save`, `zoom`, `pan`, `rotate`, marker, `legend`



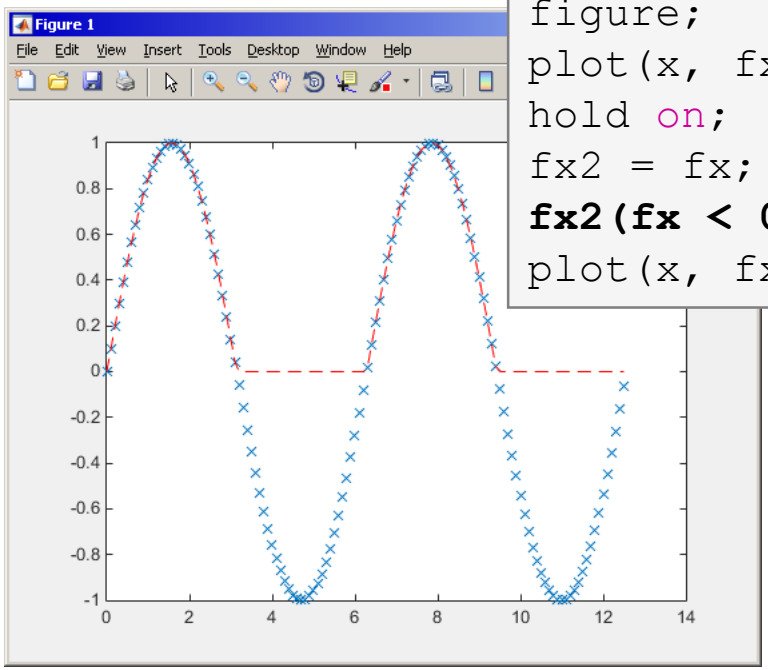
  - otevře `Matlab Property Editor` (probereme později)

- všechny tyto operace lze aktivovat skrze funkce Matlabu
  - probereme později (např. `rotate3d` aktivuje rotaci obrázku, `view(az,el)` nastaví 3D perspektivu grafu pro zadaný azimut `az` a elevaci `el`)

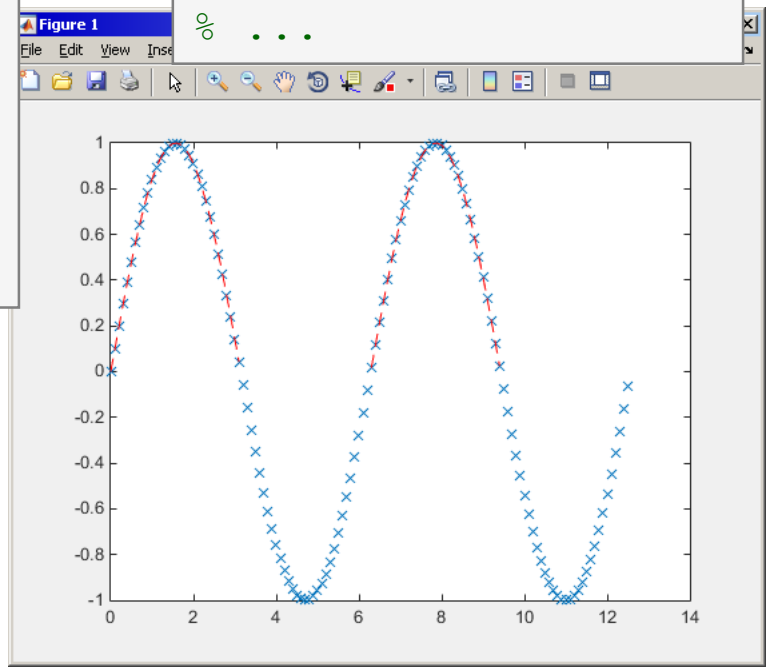
Vizualizace – využití hodnot NaN

- hodnoty NaN nejsou v grafech vykreslovány
  - často musíme odlišit např. nulové hodnoty a nedefinované hodnoty
  - vykreslování s pomocí NaN lze využívat ve všech vizualizačních funkcích

```
x = 0:0.1:4*pi;
fx = sin(x);
figure;
plot(x, fx, 'x');
hold on;
fx2 = fx;
fx2(fx < 0) = 0;
plot(x, fx2, 'r--');
```




```
% ...
fx2(fx < 0) = NaN;
% ...
```

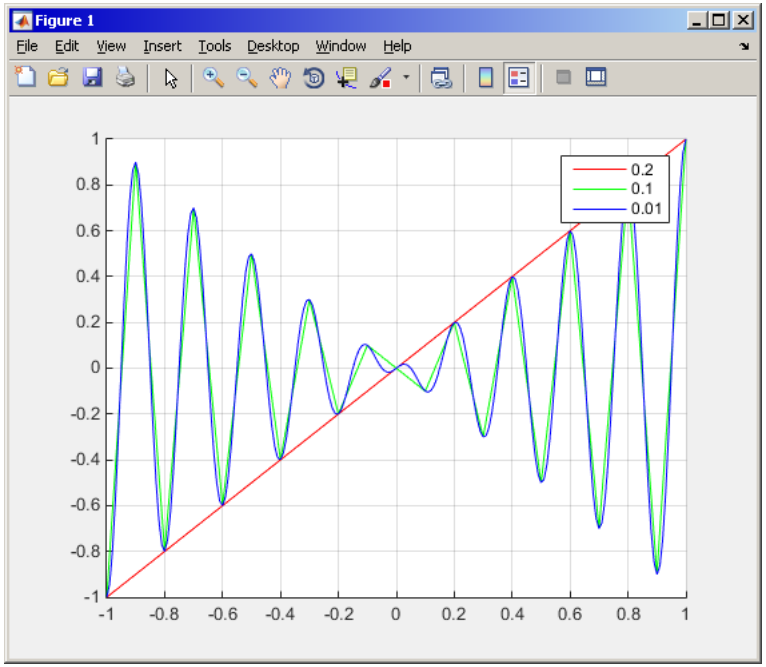


Příklad - vzorkování

300 s ↑

- Vykreslete funkci $f(x) = x \sin\left(\frac{\pi}{2}(1+20x)\right)$ v intervalu $\langle -1;\ 1 \rangle$

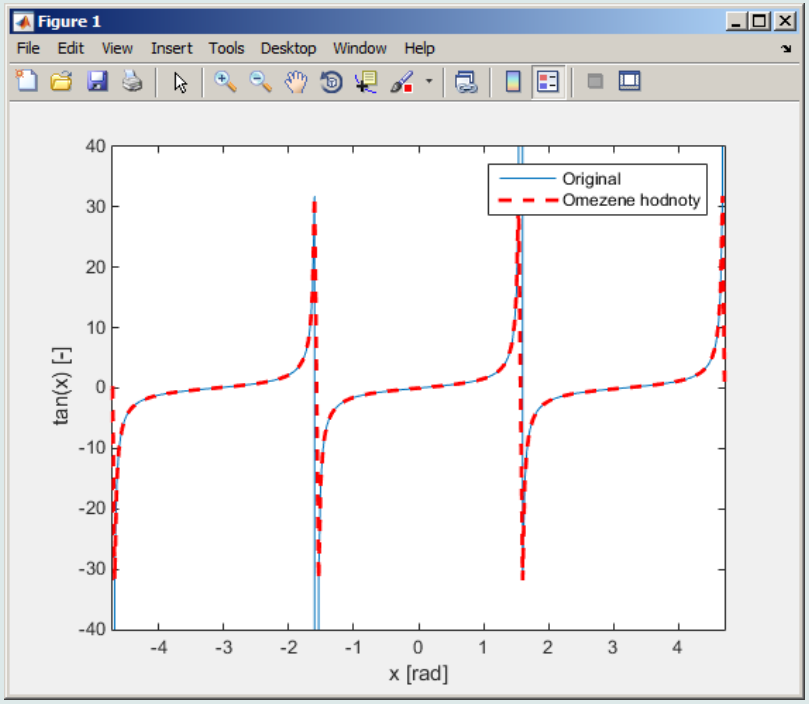
  s krokem 0.2, 0.1 a 0.01
- porovnejte výsledky!



Příklad - zaokrouhlování

300 s ↑

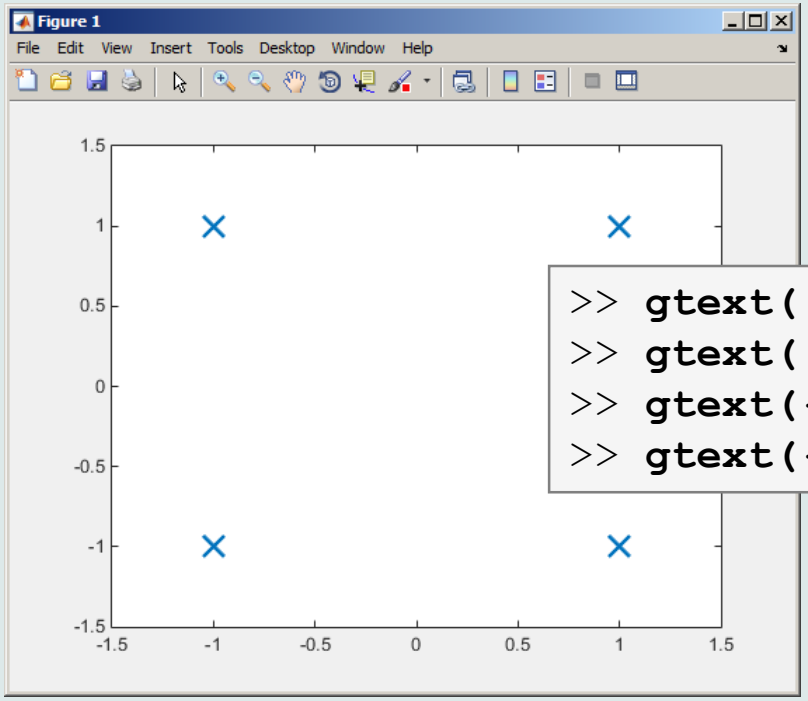
- vykreslete funkci `tan(x)` pro $x \in \langle -3/2\pi; 3/2\pi \rangle$ s krokem $\pi/100$
- omezte hodnoty zobrazené v grafu na ± 40
- funkční hodnoty s absolutní hodnotou větší než $1 \cdot 10^{10}$ nahraďte nulou
  - použijte logické indexování
- zobrazte oba výsledky a porovnejte



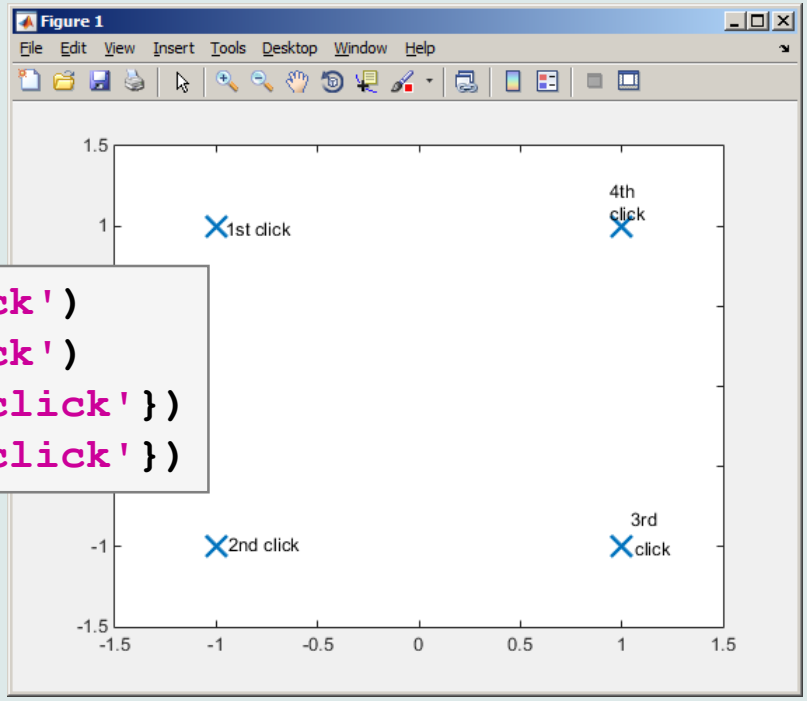
Funkce gtext

- funkce `gtext` umožňuje vložit zadaný text do grafu
  - umístění probíhá interaktivně pomocí osového kříže

```
>> plot([-1 1 1 -1], [-1 -1 1 1], ...
    'x','MarkerSize',15,'LineWidth',2);
>> xlim(3/2*[-1 1]); ylim(3/2*[-1 1]);
```

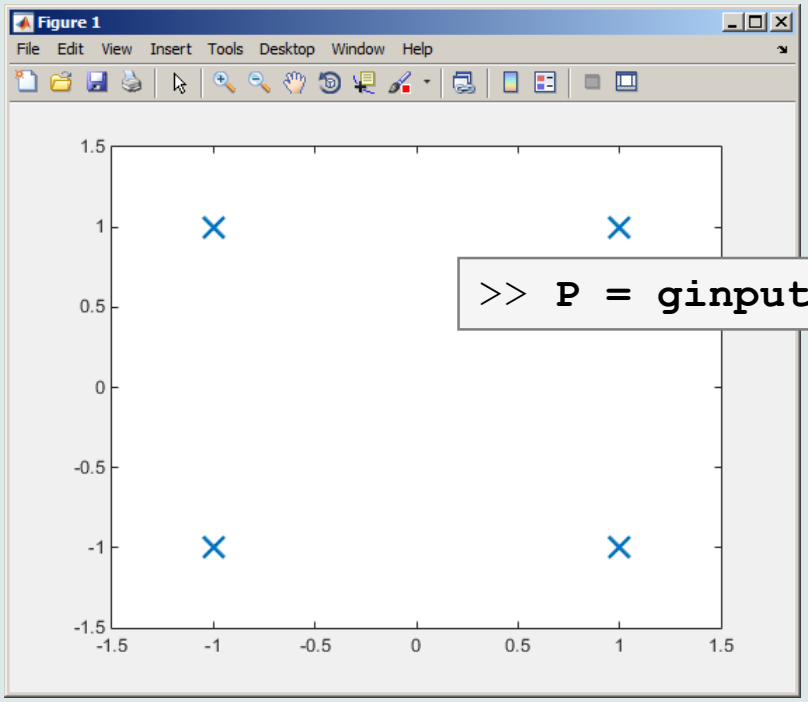



```
>> gtext('1st click')
>> gtext('2nd click')
>> gtext({'3rd';'click'})
>> gtext({'4th','click'})
```



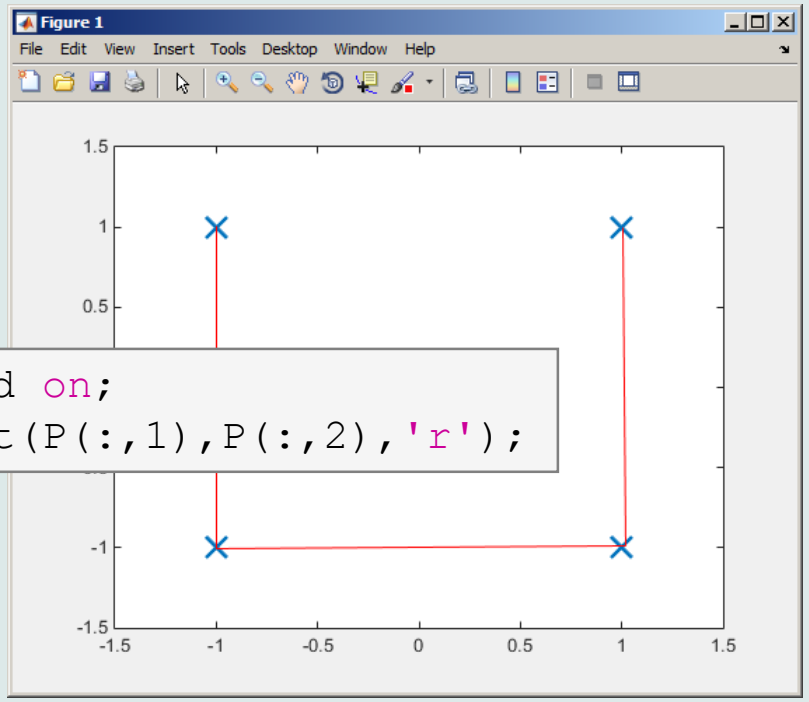
Funkce `ginput`

- funkce `ginput` umožňuje zadat body na grafu pomocí myši
  - buďto zadáme potřebný počet bodů (`P = ginput(x)`) nebo ukončíme stisknutím klávesy Enter




```
>> P = ginput(4);
```

```
>> hold on;
>> plot(P(:,1),P(:,2),'r');
```



Ladění běhu programu #1

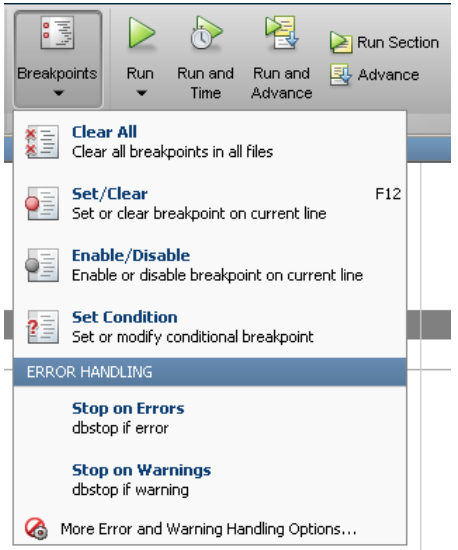
- *bug* $\Rightarrow$ *debugging*

- rozeznáváme:
  - chyby sémantické („logické“ chyby)
    - zpravidla těžko identifikovatelné
  - chyby syntaktické („gramatické“ chyby)
    - všímejte si obsahu chybových hlášení – usnadní to odstranění chyby
  - neočekávané události (viz dále)
    - např. problém se zápisem do otevřeného souboru, málo místa na disku, atp.
  - zaokrouhlovací chyby (vše počítá, jak má a přesto je výsledek chybný)
    - je nutné dopředu analyzovat algoritmus, určit dynamiku výpočtů atd.

- ladění a testování softwaru patří k nedílné součásti jeho vývoje
  - později probereme možnosti zrychlení kódu pomocí `Matlab profile`

Ladění běhu programu #2

- nejprve se omezíme na sémantické a syntaktické chyby ve skriptech
  - program vždy testujeme na případu kdy známe (tušíme) správný výsledek
- možné techniky:
  - využijeme funkce `who`, `whos`, `keyboard`, `disp`
  - využijeme ladící nástroje v `Matlab Editoru` (ukázka)



MATLAB Functions

dbclear	Clear breakpoints
dbcont	Resume execution
dbdown	Reverse workspace shift performed by dbup, while in debug mode
dbquit	Quit debug mode
dbstack	Function call stack
dbstatus	List all breakpoints
dbstep	Execute one or more lines from current breakpoint
dbstop	Set breakpoints for debugging
dbtype	List text file with line numbers
dbup	Shift current workspace to workspace of caller, while in debug mode
checkcode	Check MATLAB code files for possible problems
keyboard	Input from keyboard
mlintrpt	Run checkcode for file or folder, reporting results in browser

- využijeme Matlabu built-in funkce, které ladění provádějí

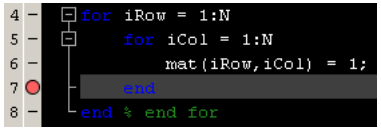
Ladění běhu programu

250 s ↑

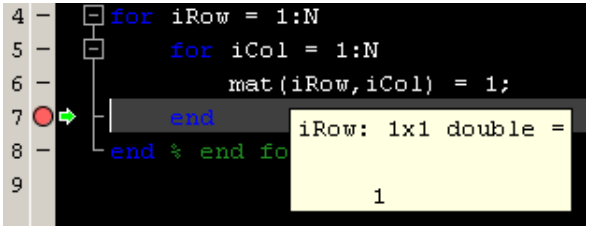
- pro vybraný skript, např.:

```
clear; clc;
N   = 5e2;
mat = zeros(N,N);
for iRow = 1:N
    for iCol = 1:N
        mat(iRow,iCol) = 1;
    end % end for
end % end for
```

- se pokuste v `Matlab Editoru`:
  - nastavit *Breakpoint* (ve sloupečku vedle čísla řádky)



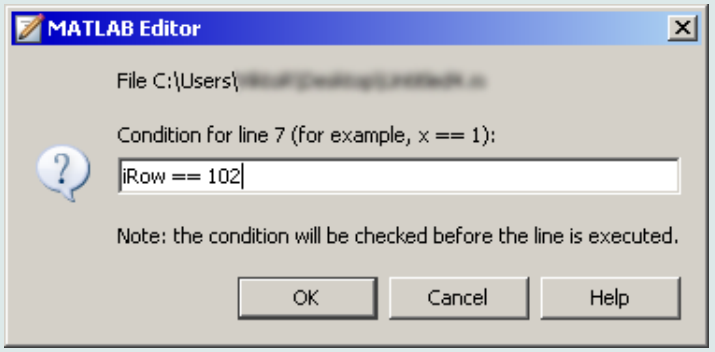
  - spustit skript (F5)
  - podívat se na stav proměnných (prostředí `keyboard`, příp. v Editoru najeďte myší nad proměnnou, která Vás zajímá)
  - krokujte skript dále
    - jaký je rozdíl mezi *Continue* a *Step* (F10)?

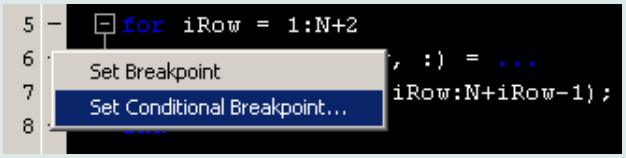


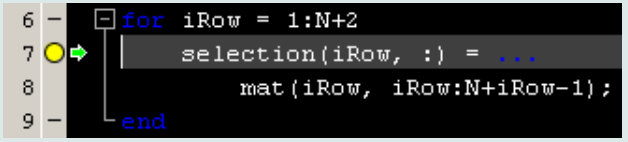
Advanced debugging

- *Conditional Breakpoints*
  - slouží k pozastavení běhu programu při splnění definované podmínky
    - podmínku může být náročné určit
  - uživatelsky jednodušší odhalení chyb v cyklech
    - běh programu lze zastavit v konkrétním cyklu
  - podmínka může být libovolný vyhodnotitelný logický výraz

```
% kod s chybou
clear; clc;
N   = 100;
mat = magic(2*N);
selection = zeros(N, N);
for iRow = 1:N+2
    selection(iRow, :) = ...
        mat(iRow, iRow:N+iRow-1);
end
```







Vybrané tipy pro přehledný kód #1

```
for iRow = 1:N
    mat(iRow,:) = 1;
end % end of ...
```

- využívejte odsazení těla cyklu, případně kódu uvnitř podmínek (TAB)
  - velikost odsazení lze nastavit v `Preferences` (obvykle 3 až 4 mezery)
- v případě podmínek používejte „pozitivní“
  - tj. využijte `isBigger`, příp. `isSmaller`, nikoliv `isNotBigger` (vede ke zmatení)
- složité konstrukce s logickými a relačními operátory vyhodnoťte samostatně → vyšší čitelnost kódu
  - srovnejte:

```
if (val>lowLim)&(val<upLim)&~ismember(val,valArray)
    % do something
end
```

vs.

```
isValid = (val > lowLim) & (val < upLim);
isNew   = ~ismember(val, valArray);
if isValid & isNew
    % do something
end
```

Vybrané tipy pro přehledný kód #2

- kód lze oddělovat volnou řádku tam, kde to zvýší přehlednost
- dvě volné řádky užijte pro významné oddělení (funkcionality)
  - lze užít i cellů, případně např. řádky `%----------`, atp.
- zvažte volné mezery kolem operátorů (= & |)
  - zvýšíte čitelnost kódu:

```
(val>lowLim)&(val<upLim)&~ismember(val,valArray)
```

vs.

```
(val > lowLim) & (val < upLim) & ~ismember(val, valArray)
```

- v případě hodně zanoření využijte komentářů za slovem `end`

Probrané funkce

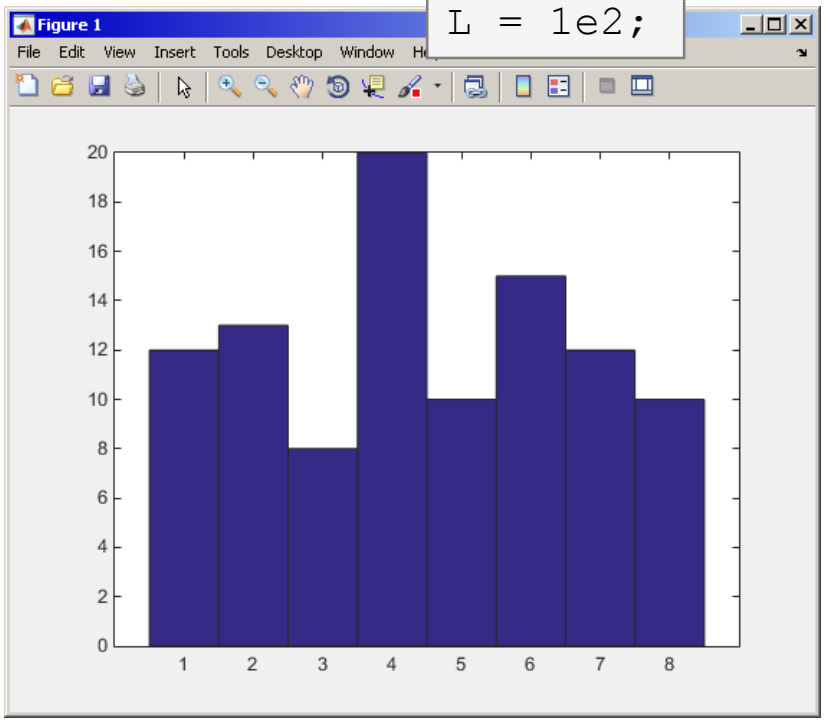
`switch-case-otherwise-end`	podmínka	•
`figure, hold`	otevře nové okno grafu, umožní vykreslení více křivek v jednom grafu	•
`title, xlim, ..., xlabel, ...`	nadpis grafu, limity os, popisy os	•
`legend, grid`	legenda k vykresleným datům, mřížka	•
`gtext, ginput`	interaktivní vložení textu, interaktivní zadání bodů v grafu	•

Cvičení #1

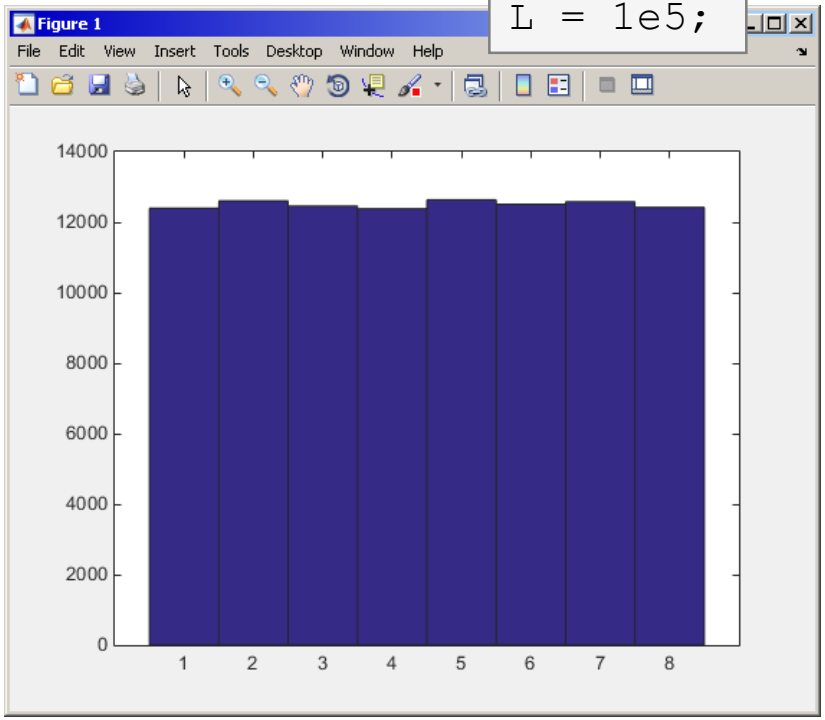
600 s ↑

- napište skript, který bude simulovat `L` hodů osmistěnnou kostkou
  - jaké očekáváte rozložení pravděpodobnosti?
  - pro vykreslení použijte `hist(hody, 1:8)`, kde `hody` je vektor hodů
  - počet hodů `L` uvažujte různě veliký (od desítek po miliony)

`L = 1e2;`



`L = 1e5;`



Cvičení #2

600 s ↑

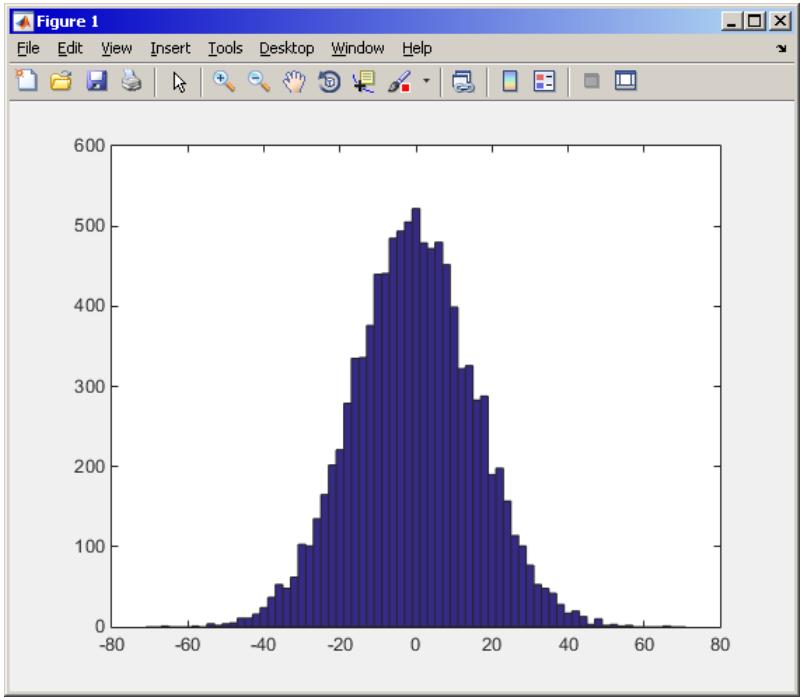
- napište skript, který bude simulovat `N` sérií pokusů, kdy v každé sérii `M` krát hodíme korunou (padne buď orel nebo panna)
  - vygenerujte matici s hody (o rozměrech M×N)
  - vypočítejte, kolikrát padla jednička v každé ze sérií (číslo mezi 0 a M)
  - vypočítejte o kolik padlo jedniček více (nebo méně) než je průměr (očekávání dané rovnoměrných rozložením pravděpodobnosti)
  - jaké očekáváte rozložení pravděpodobnosti?
  - vykreslete výsledné odchylky v počtu padlých jedniček do grafu
    - využijte příkaz `hist()`

Cvičení #3

- můžeme vypočítat střední hodnotu a směrodatnou odchylku:

$N = 1 \cdot 10^4$:

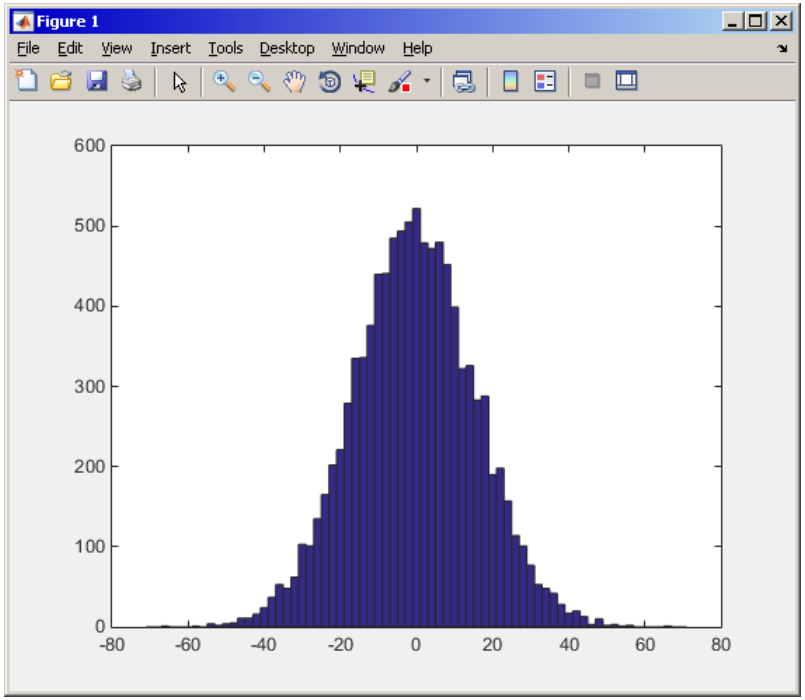
$$\mu = \frac{1}{N} \sum_i x_i \approx 0 \qquad \sigma = \sqrt{\frac{\sum_i (\mu - x_i)^2}{N}}$$



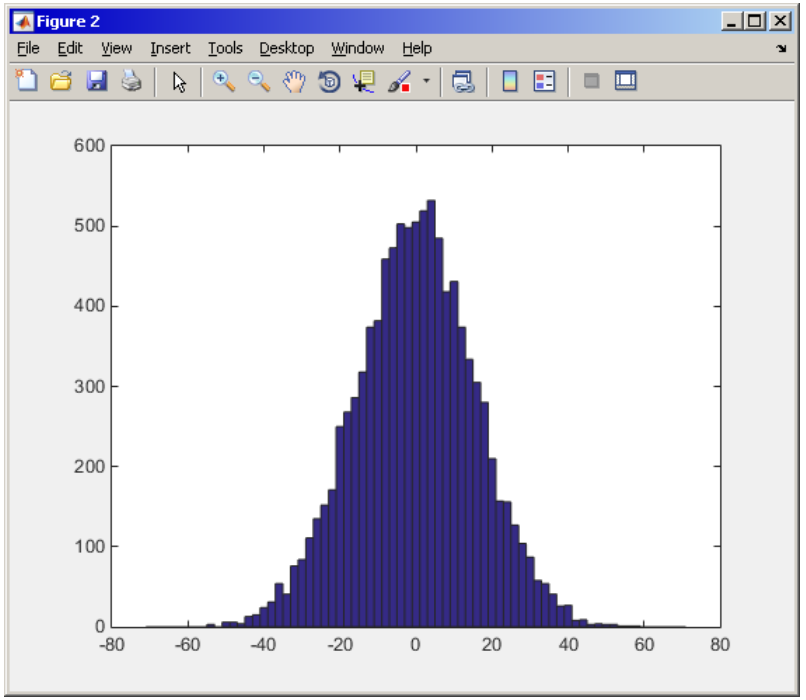
Cvičení #4

- otestujme, zda pro přímo generovaná data dostaneme podobnou distribuci:

hod kostkou:

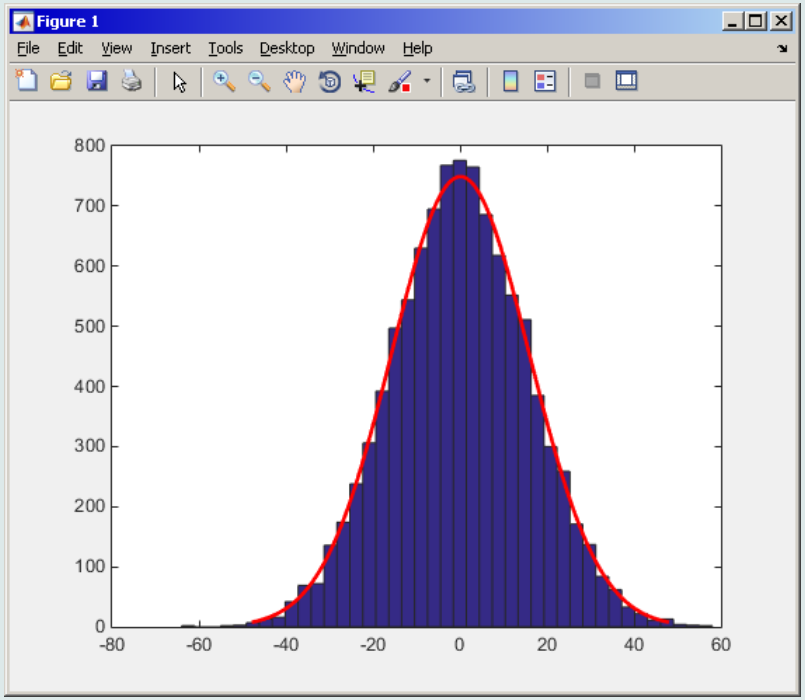


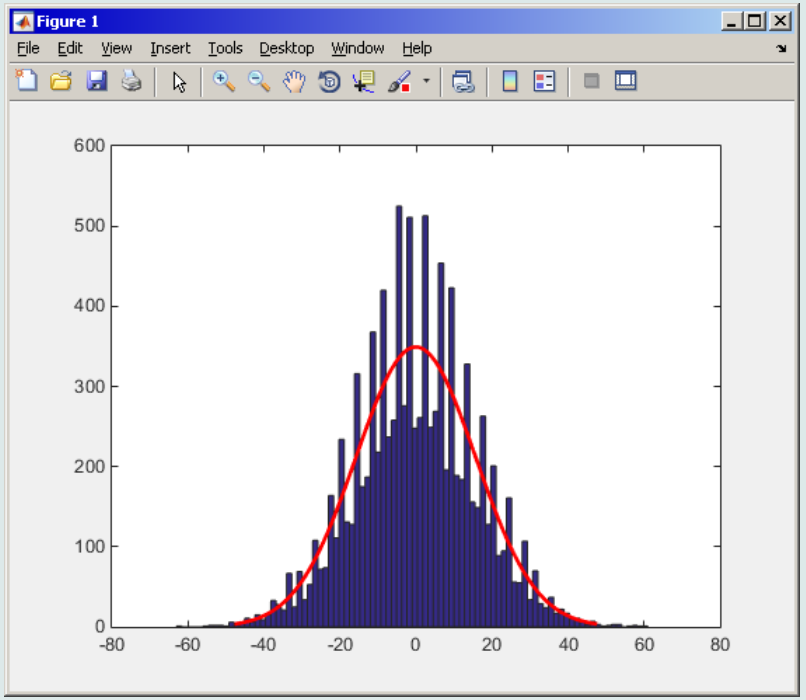
odpovídající, přímo generovaná data:



Cvičení #5

- pro vykreslení hustoty pravděpodobnosti k histogramu použijte funkci `histfit` (Statistics Toolbox)
  - vhodně nastavte parametr `nbins` pro korektní zobrazení histogramu nespojité náhodné veličiny

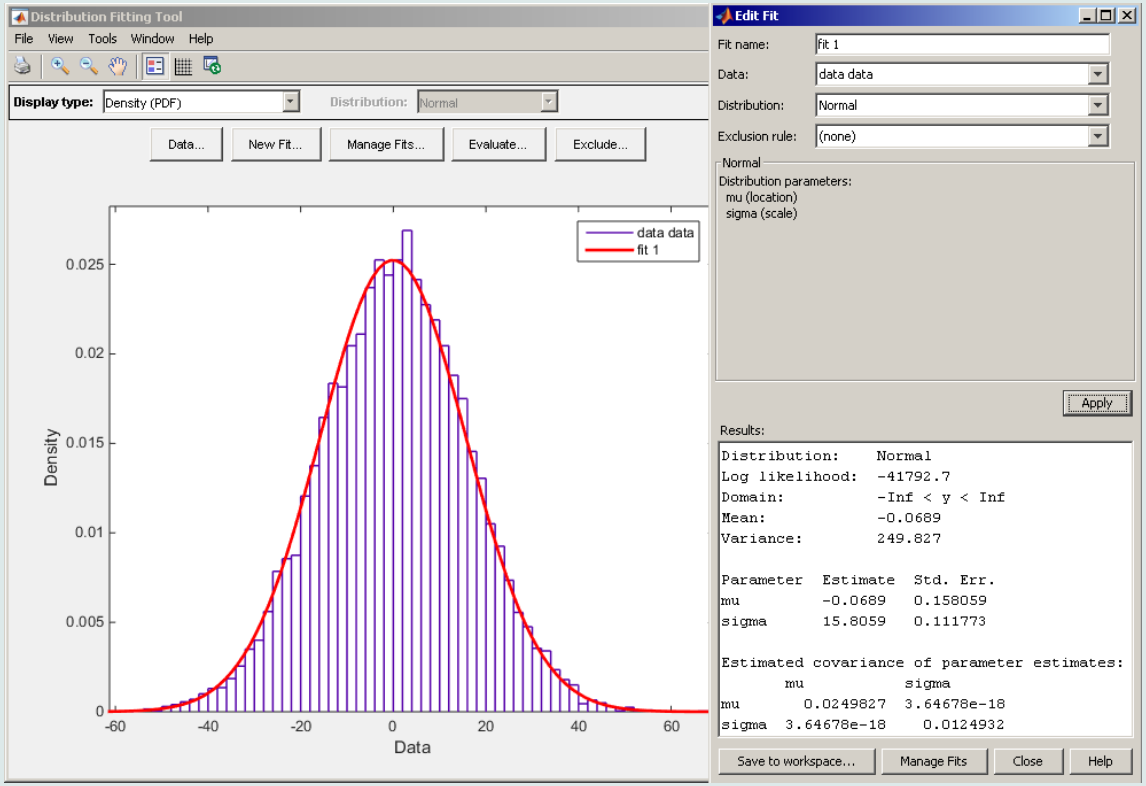




Cvičení #6

- použijte Distribution Fitting Tool (`dfittool`) k aproximaci distribuce pravděpodobnosti náhodných pokusů

```
dfittool(noOnesOverAverage);
```



Cvičení #7

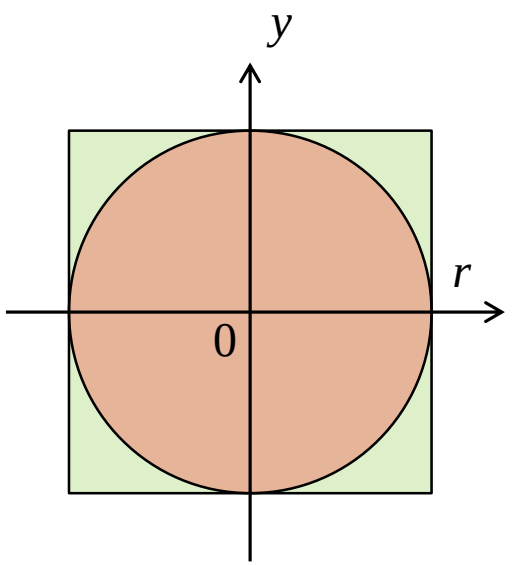
600 s ↑

- využijte metodu Monte Carlo a odhadněte velikost hodnoty čísla π
  - Monte Carlo je stochastická metoda využívající pseudonáhodná čísla
- postup:

  (1) v daném čtverci vygenerujme body (rovnoměrně náhodně rozmístěné)

  (2) pak porovnejte kolik náhodných bodů je v celém čtverci a kolik jich je v kruhu

$$\frac{S_o}{S_\square} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4} \approx \frac{\text{hits}}{\text{shots}}$$



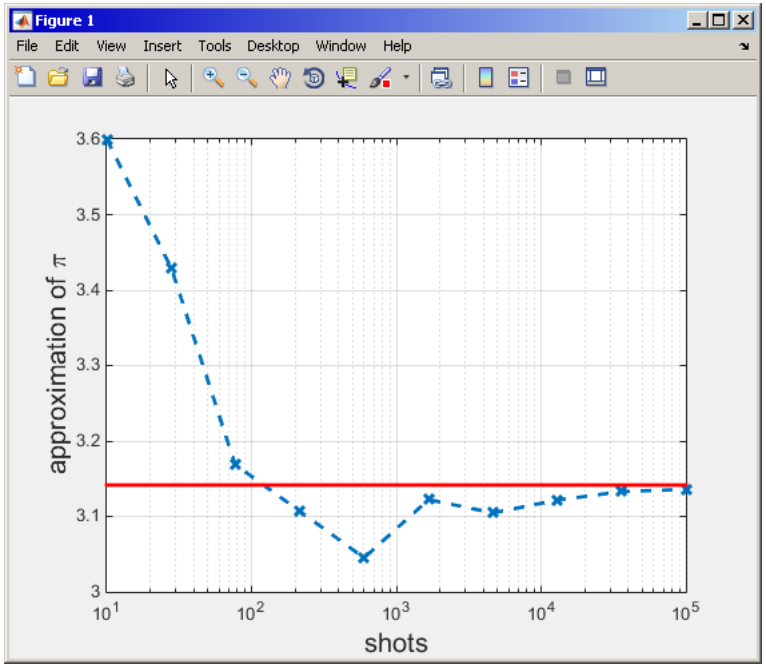
- vypracujte skript tak, aby šlo měnit počet bodů
  - všimněte si vlivu počtu bodů na přesnost řešení

Cvičení #7-řešení

- výsledný kód (poloměr kruhu $r = 1$):

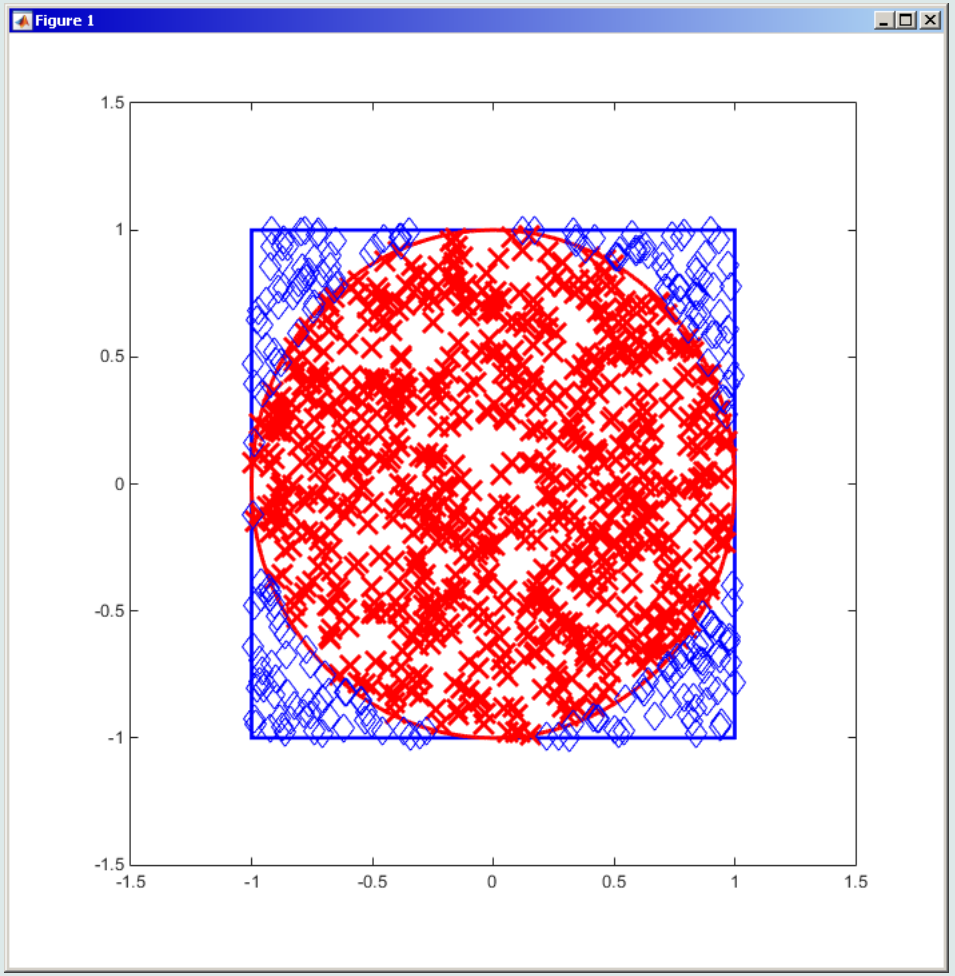
Cvičení #8

- aproximace Ludolfova čísla – vizualizace:



Cvičení #9

- vizualizace řešené úlohy:



Cvičení #10

600 s ↑

- platí následující rozvoj:

$$\arctan(x) = \sum_{n=0}^{\infty} (-1)^n \frac{(x)^{2n+1}}{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots$$

- na základě rozvoje pro $x = 1$ odhadněte hodnotu čísla π:

$$\arctan(1) = \frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$$

- zjistěte počet potřebných sčítanců a výpočetní čas pro přesnost odhadu lepší než $1 \cdot 10^{-6}$

Cvičení #11

600 s ↑

- pomocí rozvoje

$$\frac{\pi}{8} = \sum_{n=0}^{\infty} \frac{1}{(4n+1)(4n+3)} = \frac{1}{1\cdot 3} + \frac{1}{5\cdot 7} + \frac{1}{9\cdot 11} + \ldots$$

  odhadněte velikost čísla π

- zjistěte počet potřebných sčítanců a výpočetní čas pro přesnost odhadu lepší než $1\cdot 10^{-6}$

Cvičení #12

600 s ↑

- pro aproximace čísla π použijte vztah:

$$\frac{\pi}{4} = 6\arctan\left(\frac{1}{8}\right) + 2\arctan\left(\frac{1}{57}\right) + \arctan\left(\frac{1}{239}\right)$$

- pro implementaci funkce arctan použijte rozvoj:

$$\arctan(x) = \sum_{n=0}^{\infty} (-1)^n \frac{(x)^{2n+1}}{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots$$

- zjistěte počet potřebných sčítanců, výpočetní čas pro přesnost odhadu lepší než $1 \cdot 10^{-6}$ a porovnejte s předchozími řešeními

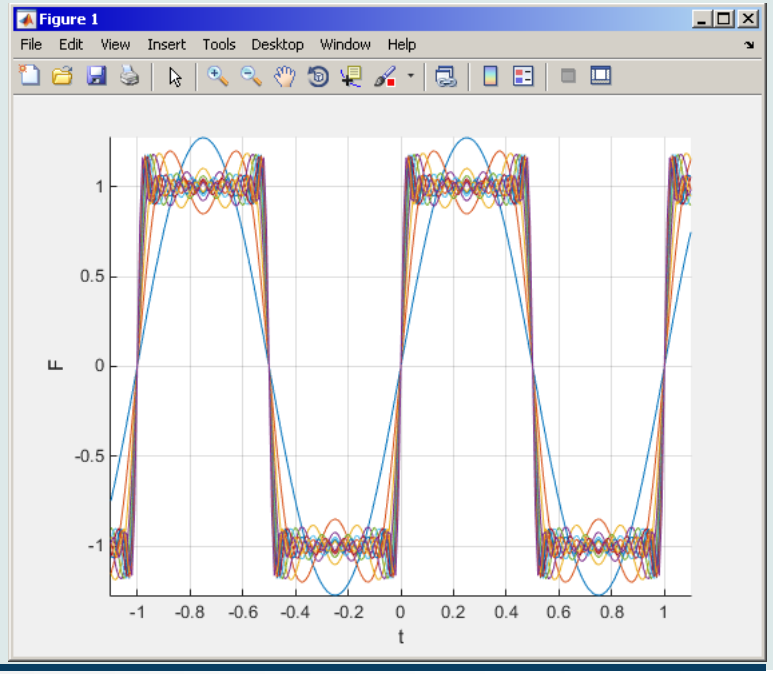
Cvičení #13

600 s ↑

- pro periodický obdélníkový signál s nulovou stejnosměrnou složkou, amplitudou A a periodou T se dá analyticky odvodit Fourierova řada

$$s(t) = \frac{4A}{\pi} \sum_{k=0}^{\infty} \frac{1}{2k+1} \sin\left(\frac{2\pi t (2k+1)}{T} \right)$$

- pro A=1 V a T=1 s vykreslete do jednoho obrázku výsledný signal $s(t)$ aproximovaný z jedné až deseti harmonických složek v úseku $t \in \langle -1.1; 1.1 \rangle$s



Děkuji!

ver. 3.7 (13/3/2014)
Miloslav Čapek
miloslav.capek@fel.cvut.cz