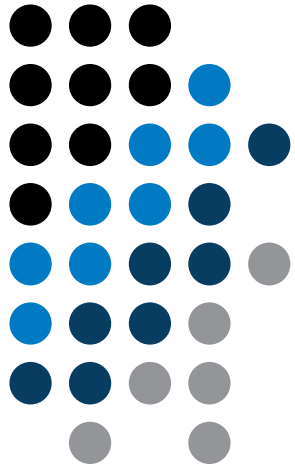
A0B17MTB – Matlab

Část #3

Miloslav Čapek
miloslav.capek@fel.cvut.cz

Filip Kozák, Viktor Adler

Katedra elektromagnetického pole
B2-626, Dejvice



Naučíte se …

Indexace

```
ResTable.data1(...
   PsoData.cond{crt}(spr,2),...
   PsoData.cond{crt}(spr,3) ...
   ) = ...
 bestPersDim(bestGlobNum, crt);
```

Velikost a typ dat

Formát výsledků

`Matlab Editor`

Indexace v Matlabu

- nyní již známe vše potřebné pro vyložení indexace v Matlabu
- zvládnutí indexace je zcela zásadní pro efektivní práci s Matlabem!!!
- dosud jsme pracovali pouze s celými maticemi, velmi často však vyžadujeme přístup pouze k jednotlivým prvkům
- pro matice / pole rozlišujeme
  - přístup pomocí kulatých závorek „ ( ) “
    - odkazuje na pořadí v prvků v matici
  - přístup pomocí hranatých závorek „ [ ] “
    - odkazuje na obsah (dané prvky) v matici

Indexace v Matlabu

600 s ↑

- uvažme následující trojici matic
  - zkuste si v Matlabu jednotlivé příkazy, odhalte jak pracují
  - postupujte vždy od vnitřních příkazů, problém si klidně nakreslete
  - povšimněte si funkce klíčového slova `end`

$$\mathbf{N}_1 = \begin{pmatrix} -5 \\ 0 \\ 5 \end{pmatrix} \qquad \mathbf{N}_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 2 & 3 & 5 & 7 & 11 \end{pmatrix} \qquad \mathbf{N}_3 = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 22 & 24 & 26 & 28 \\ 33 & 36 & 39 & 42 \\ 44 & 48 & 52 & 56 \end{pmatrix}$$

```
>> N1 = (-5:5:5)';N2 = [1:5;2:2:10;primes(11)];N3 = (1:4)'*(11:14);
```

```
>> N1
>> N1(1:3)
>> N1([1 2 3])
>> N1(1:2)
>> N1([1 3])
>> N1([1 3]')
>> N1([1 3])'
>> N1([1; 3])
```

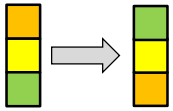
```
>> N2(1, 3)
>> N2(3, 1)
>> N2(1, end)
>> N2(end, end)
>> N2(1, :)
>> N2(1, :)'
>> N2(:, 2)
>> N2(:, 3:end)
```

```
>> N3(2:3, [1 1 1]) % jako repmat
>> N3(2:3, ones(1,3))
>> N3(2:3, ones(3,1))
>> N3([N2(2,1:2)/2 4], [2 3])
>> N3([1 end], [1:4 1:2:end])
>> N3(:, :, 2) = magic(4)
>> N3([1 3], 3:4, 3) = ...
       [1/2 -1/2; pi*ones(1, 2)]
```

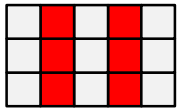
Indexace v Matlabu

420 s ↑

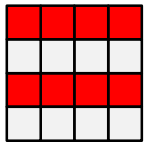
- zapamatujte si funkci klíčového slova `end` a využití dvojtečky „`:`“
- vyzkoušejte:
  - prohození prvků vektoru **N1**



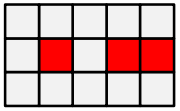
    - bez využití funkce `fliplr` / `flipud`
  - výběr pouze sudých sloupečků **N2**



  - pouze liché řádky **N3**



  - 2., 4. a 5. sloupec **N2** ve 2. řádku



  - vytvořte matici 4×3 **A**, obsahující zleva doprava, shora dolů čísla 1-12

Indexace v Matlabu

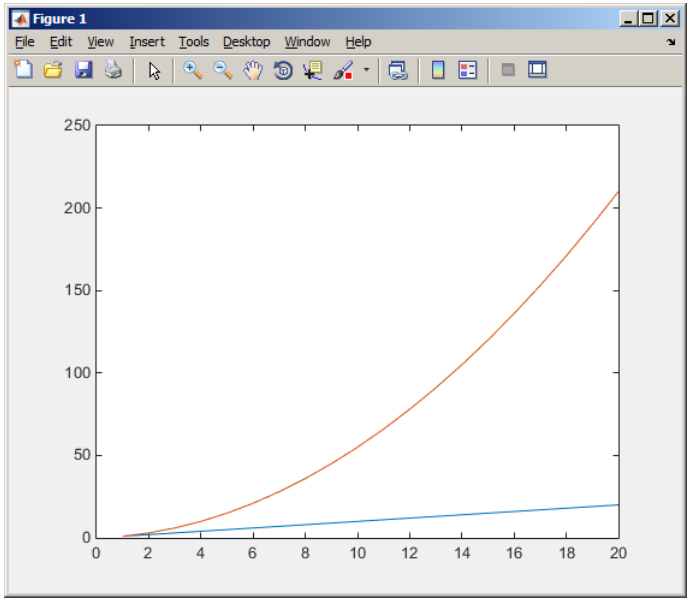
300 s ↑

- spočítejte kumulativní **S** sumu pro vektor **x** celých čísel od 1 do 20
  - pro nalezení potřebné funkce využijte nápovědy (*cumulative sum*)
  - pro vyčíslení součtu použijte funkci Matlabu

$$\mathbf{x} = \begin{pmatrix} 1 & 2 & \dots & 20 \end{pmatrix}$$

$$S = \begin{pmatrix} 1 & 1+2 & \dots & 1+2\cdots+20 \end{pmatrix}$$

  - spočítejte kumulativní sumu **L** a vyhodnoťte ji nad každým sudým elementem vektoru **x**
  - jaká je hodnota posledního prvku výsledného vektoru **L**?



Indexace v Matlabu

150 s ↑

- který příkaz vrátí rohové elementy matice **A** o rozměrech 10×10?

```
>> A([1,1], [end,end])                          % A.
>> A({[1,1], [1,end], [end,1], [end,end]})      % B.
>> A([1,end], [1,end])                          % C.
>> A(1:end, 1:end)                              % D.
```

Mazání prvků matice

- pro mazání prvků je stěžejní poznatek z tvorby prázdné matice

```
>> T = []
```

- tj. chceme-li
  - odebrat 2. řádek matice **A**

```
>> A(2, :) = []
```

  - odebrat 3. sloupec matice **A**

```
>> A(:, 3) = []
```

  - 1., 2. a 5. sloupec matice **A**

```
>> A(:, [1 2 5]) = []
```

Přidávání, náhrada prvků matic

- chceme-li nahradit
  - 3. sloupec matice **A** (velikosti M×N) vektorem **x** (velikosti N)

```
>> A(:, 3) = x
```

  - 2., 4. a 5. řádek matice **A** třemi řádky matice **B** (počet sloupců **A** i **B** je stejný – jak by se upravil výraz, kdyby stejný nebyl?)

```
>> A([2 4 5], :) = B(1:3, :)
```

- chceme-li prohodit
  - 2. řádek matice **A** s 5. sloupcem matice **B** (počet sloupců matice **A** je roven počtu řádek matice **B** – jak by se upravil výraz, kdyby stejný nebyl?)

```
>> A(2, :) = B(:, 5)'
```

- pamatujte, že se Vám vždy musejí rovnat rozměry matic!

Mazání, přidávání, náhrada matic

420 s ↑

- který příkaz vymaže první a poslední sloupec matice **A** velké 6×6?
  - pro vyzkoušení si vytvořte potřebnou matici

```
>> A[1,end] = 0       % A.
>> A(:,1,end) = []    % B.
>> A(:,[1:end]) = []  % C.
>> A(:,[1 end]) = []  % D.
```

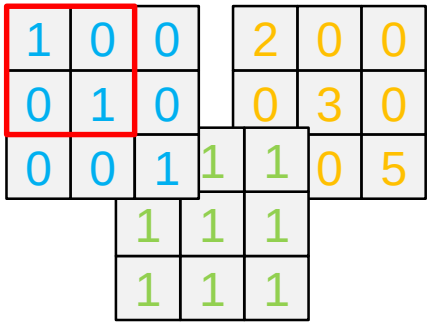
- nahraďte 2., 3. a 5. řádek matice **A** prvním řádkem matice **B**
  - předpokládejte, že počet sloupců **A** i **B** je stejný, matici **B** si vytvořte
  - ošetřete případ, kdy **B** má větší počet sloupců než **A**
  - jak se chová příkaz, pokud má **B** menší počet sloupců než **A**?

Tvorba matic, náhrada prvků

300 s ↑

- vytvořte následující 3D pole

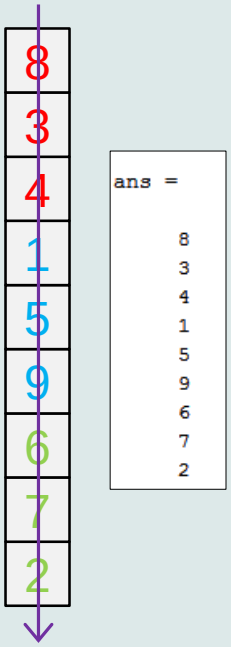
$$\mathbf{M}(:,:,1)=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\quad \mathbf{M}(:,:,2)=\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix},\quad \mathbf{M}(:,:,3)=\begin{pmatrix}2&0&0\\0&3&0\\0&0&5\end{pmatrix}$$



- nahraďte prvky, které jsou umístěny v prvních dvou sloupcích a řádcích prvního listu struktury (tj. matici `[1 0; 0 1]`) prvky `NaN`

Lineární indexace

- pole o libovolném počtu dimenzí a velikosti lze indexovat i pomocí jediného indexu
  - indexace probíhá postupně podle hlavní dimenze (ve sloupečku), potom vedlejší dimenze (podél řádek), atd.

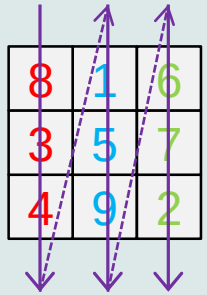



```
ans =

     8
     3
     4
     1
     5
     9
     6
     7
     2
```

```
>> A = magic(3)
```

```
>> A(:)
```

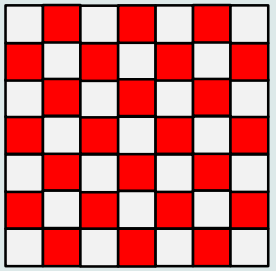
```
>> A = magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> A(:)
```



Lineární indexace - využití

- mějme následující matici:

```
>> MAT = rand(7);
```

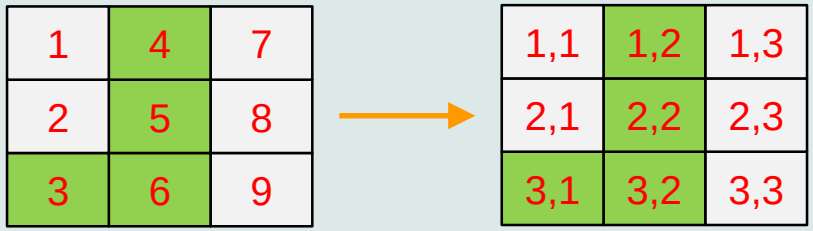


  - v matici nyní všechny prvky zvýrazněné červeně vynulujeme:

```
>> MAT(2:2:end) = 0
```

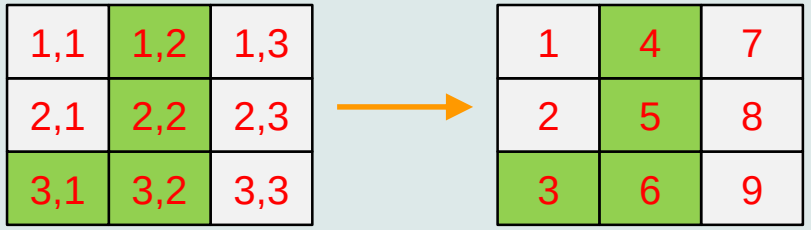
Lineární indexace – `ind2sub`, `sub2ind`

- `ind2sub`: přepočte lineární index na index odpovídající velikosti a dimenzi nové matice
  - podporuje pole libovolných rozměrů a dimenzí




```
>> ind = 3:6;
>> [rw, col] = ind2sub([3, 3], ind)
% rw  = [3 1 2 3]
% col = [1 2 2 2]
```

- `sub2ind`: přepočte dílčí indexy na lineární indexy
  - podporuje pole libovolných rozměrů a dimenzí




```
>> ind2 = sub2ind([3, 3], rw, col)
% ind2 = [3 4 5 6]
```

Lineární indexace

300 s ↑

- pro dvourozměrné pole, kde znáte polohu (tj. číslo řádku i číslo sloupečku), najděte vzoreček pro přepočet z indexů `row` (pořadí řádku), `col` (pořadí sloupečku) na lineární index
  - zkontrolujte pro matici `A` o velikosti 4×4, kde
    - `row = [2, 4, 1, 2]`
    - `col = [1, 2, 2, 4]`
  - a tedy
    - `ind = [2, 8, 5, 14]`

Funkce who, whos

- funkce `who` vypíše seznam všech proměnných v Matlab Workspace
  - celá řada možných nastavení
- funkce `whos` vypíše seznam + rozměr, velikost a datový typ proměnných, případně obsah souboru
  - celá řada možných nastavení

```
>> whos('-file', 'matlab.mat');
```

```
>> a = 15; b = true;
>> c = 'test'; d = 1 + 5j;
>> who
>> whos
>> Wt = what; A
>> Ws = whos; B
```

Funkce `what`, `which`, `delete`

- funkce `what` vypíše všechny Matlab soubory v aktuální složce

- funkce `which` je schopna lokalizovat (v tomto pořadí)
  - `.m` / `.p` / Simulink funkci
  - metodu Java třídy
  - Workspace proměnnou
  - libovolný soubor, je-li v aktuálním adresáři

```
>> which sin
built-in (C:\Program Files\MATLAB\R2013a\toolbox\matlab\elfun\@double\sin)  % double method
```

- funkce `delete` smaže
  - soubory
  - handle objekty (např. grafické objekty)

Funkce `cd`, `pwd`, `dir`

- funkce `cd` změní aktuální adresář
  - pokud zavolána bez parametru, vypíše aktuální adresář
  - „`cd ..`“ skočí o složku výše, „`cd /`“ vyskočí do rootu
- funkce `pwd` identifikuje aktuální adresář
- funkce `dir` vylistuje aktuální adresář
- další funkce (`mkdir`, `rmdir`, …) viz nápověda

Funkce `memory`, `ver`

- funkce `memory` vypíše informace o dostupné a využité paměti

```
>> memory
>> M = memory
```

```
>> memory
Maximum possible array:      4408 MB (4.622e+09 bytes) *
Memory available for all arrays:      4408 MB (4.622e+09 bytes) *
Memory used by MATLAB:       696 MB (7.294e+08 bytes)
Physical Memory (RAM):      3534 MB (3.705e+09 bytes)

*  Limited by System Memory (physical + swap file) available.
```

- funkce `ver` vypíše licenční informace
  - verze Matlabu, build
  - číslo licence
  - seznam toolboxů, jejich verze a určení Matlabu

```
>> ver
>> V = ver
```

- pokud potřebujete znát pouze verzi Matlab, využijte `version`

```
>> V = version
```

Formát výpisu výsledku do řádky

```
>> pi

ans =

    3.1416

>> sin(1.1)

ans =

    0.8912
```

- doposud jsme používali základní nastavení
- Matlab umožňuje i celou řadu dalších možností
  - využíváme příkaz `format nastav`
  - formát výstupu nemění přesnost s jakou jsou výsledky vypočteny nebo uchovány (stále platí `eps`, `realmax`, `realmin`, …)

`nastav`	popis formátu
`short`	fixně zobrazuje 4 desetinná čísla
`long`	15 desetinných míst pro přesnost double, 7 pro přesnost single
`shortE`	využívá plovoucí desetinné čárky a exponentu (vědecká notace)
`longE`	-//-
`bank`	pouze dvě desetinná místa (eura – centy)
`rat`	Matlab se pokusí výsledek zobrazit jako zlomek
a další	pozn.: vynecháním `nastav` v příkazu vracíme formát do počátečního nastavení

Formát výpisu výsledku do řádky

240 s ↑

- vyzkoušejte si následující změny formátování výstupů
  - každý formát je vhodný pro jiný typ úlohy

```
>> s = [5 1/2 1/3 10*pi sqrt(2)];
>> format long; s
>> format rat; s
>> format bank; s
>> format hex; s
>> format +; s
>> format; s
```

- existují další formáty s drobnými změnami
  - podívejte se do nápovědy `doc format`

- později se naučíme využívat kontrolované konverze do textových řetězců (příkazy `sprintf` a `fprintf`)

Seznam ASCII znaků

- ASCII znaky hodně využívané v Matlabu
  - všechny značky jsou na anglické klávesnici

[	ALT + 91	definice matic, indexace
]	ALT + 93	-//-
{	ALT + 123	indexace cell buněk
}	ALT + 125	-//-
@	ALT + 64	handle (symbolická matematika)
>	ALT + 62	relační operátor
<	ALT + 60	-//-
\	ALT + 92	dělení matic
|	ALT + 124	logický operátor
~	ALT + 126	-//-
^	ALT + 94	mocnina

  - více viz: `http://www.asciitable.com/`

Spouštění externích programů

- využijeme spíše zřídka

- externí programy spouštíme pomocí vykřičníku „ ! “
  - celý řádek o vykřičníku dále je zpracován jako příkaz operačního systému

```
>> !notepad poznamky.txt
>> !calc
```

  - pokud nechceme startem přerušovat běh Matlabu, přidáme „&“

```
>> !calc &
```

Práce s adresáři / soubory z řádky

- zkuste si následující
  - kód kopírujte řádku po řádce, všímejte si, co se stane
  - příkazy případně editujte s rozmyslem!!!

```
>> mkdir('My_experiment');
>> cd('My_experiment');
>> this_directory = pwd;
>> our_file = 'pathdef.m';
>> our_data = fullfile(matlabroot, 'toolbox', 'local', our_file);
>> copyfile(our_data, this_directory);
>> new_file = 'my_demo.txt';
>> movefile(our_file, new_file);
>> !write my_demo.txt
```

Příklad #1

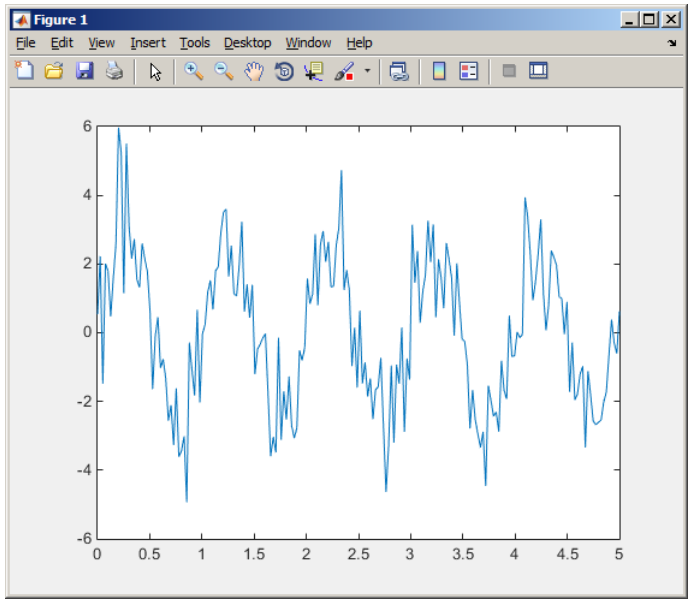
600 s ↑

- uvažujte signál ve tvaru: $s(t) = \sqrt{2\pi}\sin(2\omega_0 t) + n(\mu,\sigma), \quad \omega_0 = \pi,$
  kde střední hodnota a směrodatná odchylka normálního rozložení *n* je:

  `mu` $\mu = 0, \quad \sigma = 1$ `sigma`

  - vytvořte signál pro určitý časový úsek tak, aby bylo zobrazeno $N = 5$ period signálu s $V = 40$ hodnotami na jednu periodu
  - jedna perioda: $T = 1$: $t \in [k, k+1], \; k \in \mathbb{Z}^0$ (k volte např. 0)
  - funkce $n(\mu,\sigma)$ má v Matlabu syntax:

```
>> n = mu + sigma*randn(1, N*V)
```



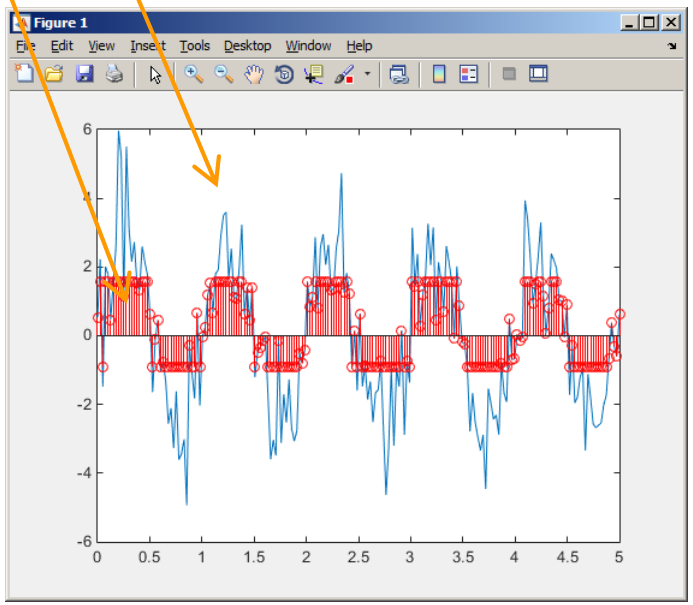
Příklad #2

600 s ↑

- na vygenerovaný signál nyní aplikujte prahovou funkci, která signál deformuje v maximech i minimech:

$$s_p(t) = \begin{cases} s_{min} \Leftrightarrow s(t) < s_{min} & s_{min} = -\frac{9}{10} \\ s_{max} \Leftrightarrow s(t) > s_{max} & \\ s(t) \ldots \text{ jinak} & s_{max} = \frac{\pi}{2} \end{cases}$$

  - výsledkem bude vektor `sp_t`
  - pro tyto účely využijte funkce `min` a `max` se dvěma vstupními parametry, podrobnosti najdete v nápovědě zmíněných funkcí
  - pro ověření využijte následujícího kódu:




```
>> close all;
>> plot(t, s_t); hold on;
>> stem(t, sp_t, 'r');
```

Matlab Editor

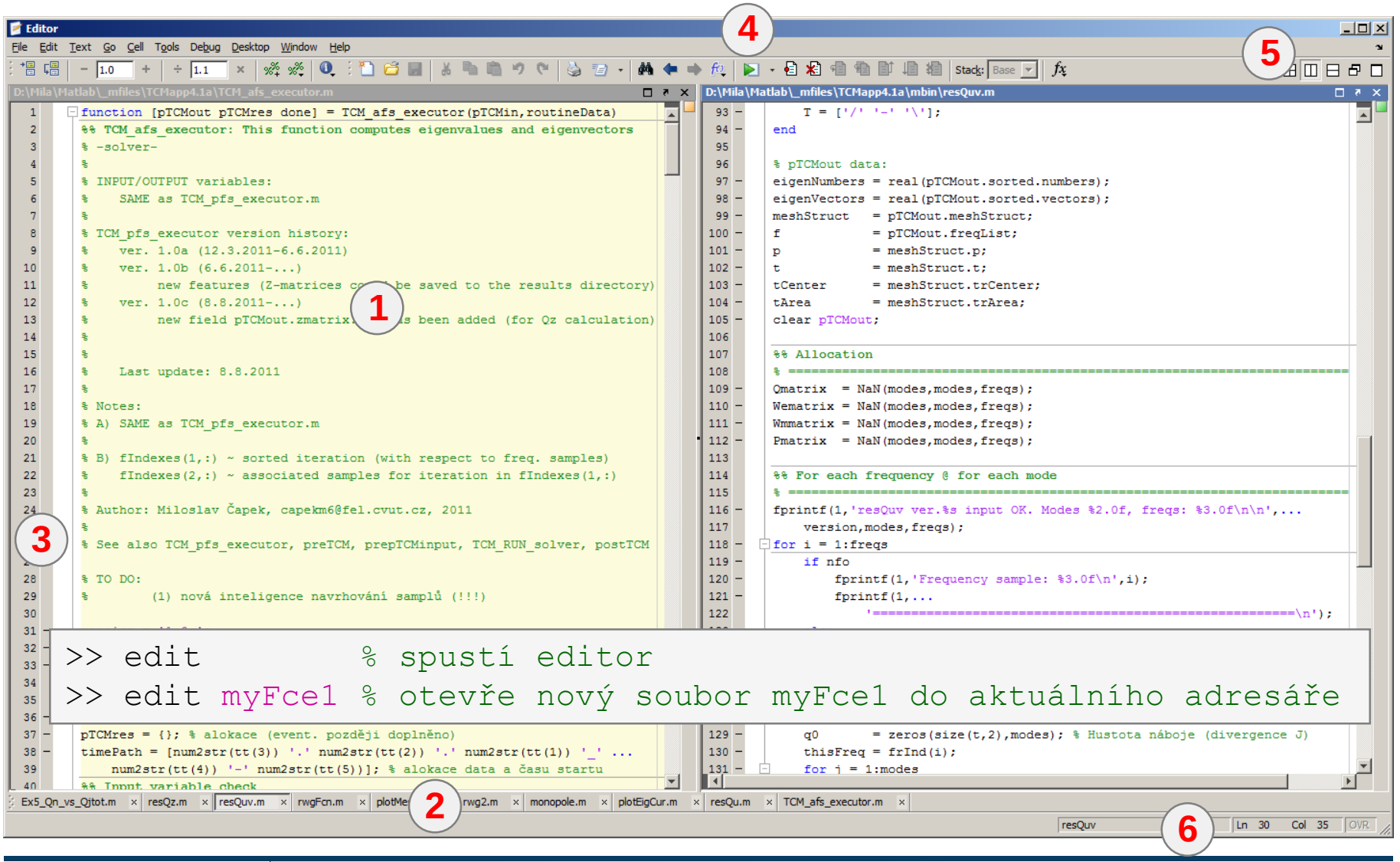
- často chceme určitou sekvenci příkazů vyhodnocovat opakovaně ⇒ využíváme skriptů v Matlabu (prosté ASCII kódování)
- ideální je využít `Matlab Editor`
  - otevřeme pomocí: `>> edit`
  - příp. v Matlabu < R2012a: `Start` → `Desktop Tools` → `Editor`
- skript si lze představit jako sekvenci výrazů, které jsme psali dříve do příkazové řádky (příklad: zpracování naměřených dat)
  - po spuštění skriptu jsou všechny příkazy postupně provedeny
  - skript operuje globálně s daty v `Matlab Workspace`
  - vhodné pro rychlé analýzy a řešení problémů s více příkazy
- názvy skriptů (stejně tak jako později názvy funkcí) mají svá specifika

Spuštění skriptu, m-soubory

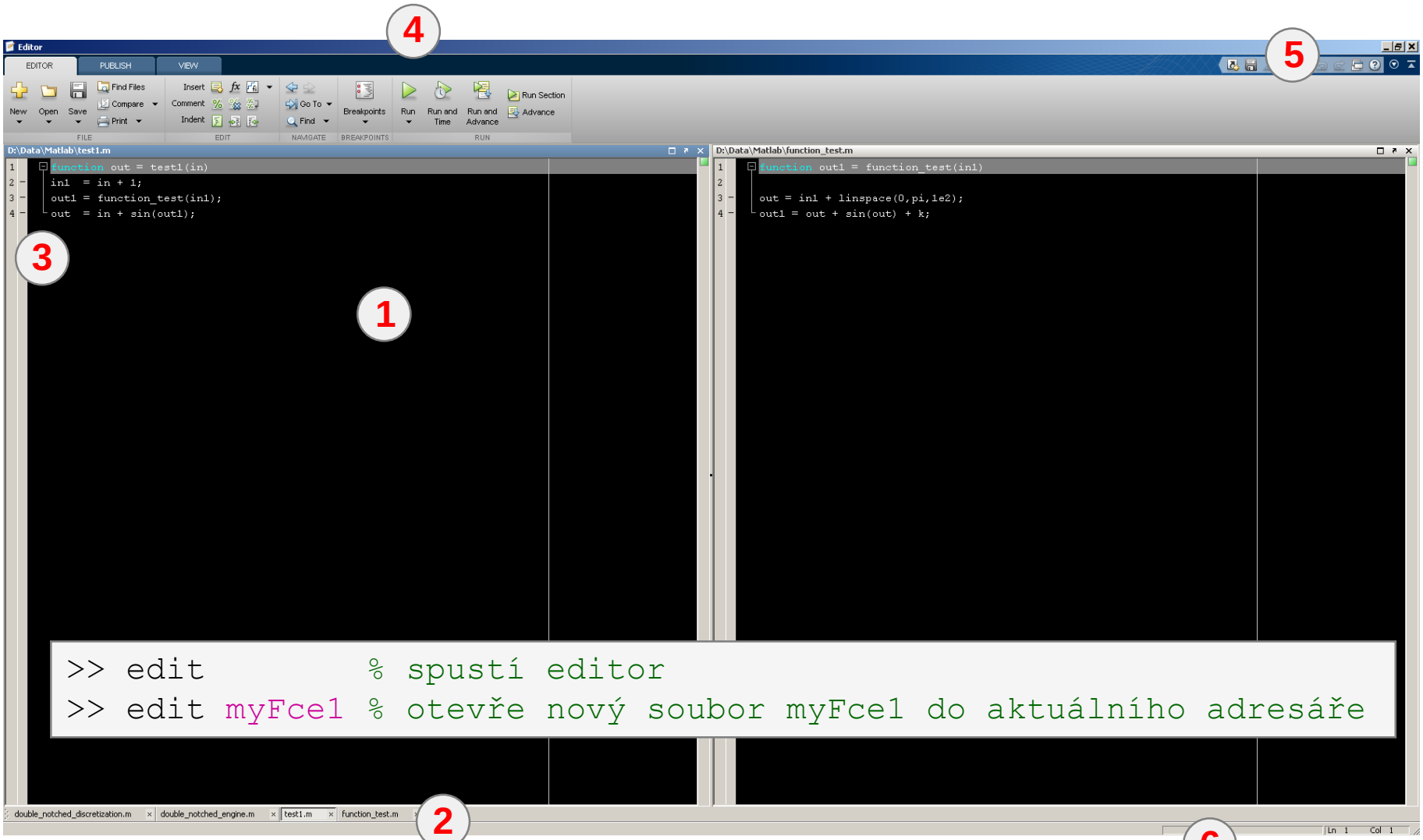
- skript můžeme spustit
  - klávesou F5 v Matlab Editoru
  - Current Folder → výběr skriptu → kontextová nabídka → Run
  - Current Folder → výběr skriptu → F9
  - z příkazové řádky: `>> jmeno_skriptu`

- skripty jsou uloženy jako tzv. m-soubory
  - přímo `.m`
  - pozor: máte-li nainstalovánu i Mathematicu, může se stát, že se Vám `m`-soubory spouštějí v ní

Matlab Editor, < R2012a



Matlab Editor, ≥ R2012a



Užitečné zkratky pro Matlab Editor

klávesa	význam
CTRL + Pg. UP	přepíná mezi všemi otevřenými m-file soubory - jedním směrem
CTRL + Pg. DOWN	- druhým směrem
CTRL + R	„zakomentuje“ označené řádky
CTRL + T	odstraní komentář z označených řádek
F5	spustí aktuální skript / funkci
CTRL + S	uloží aktuálně otevřený soubor (to je automaticky provedeno i po stisku klávesy F5)
CTRL + HOME	přeskočí na začátek otevřeného souboru
CTRL + END	přeskočí na konec otevřeného souboru
CTRL + → / ←	skáče po celých slovech nebo výrazech doprava / doleva
CTRL + W	uzavře aktuální soubor
CTRL + O	aktivuje dialog pro otevření souboru (soubory lze otevírat i z Matlabu poklepáním a přetažením)
CTRL + F	vyhledávací dialog, vč. možnosti nahradit
CTRL + G	„go to“, přejde na konkrétní řádku, nebo značené části kódu (jména funkcí / cell oddělovače)
CTRL + D	otevře v Editoru m-file s funkcí, na které je kurzor
CTRL + I	správně odsadí označené bloky kódu podle klíčových slov (`for` / `while`, `if` / `swich` – `case`)
F1	otevře kontextovou nápovědu nad funkcí

Matlab Editor

120 s ↑

- otevřete si `Matlab Editor` a připravte se k práci s novým skriptem, nazvěte ho například `signal1.m`
- jako skript využijte například příklad signálu a jeho prahování z minulé hodiny
- skript si uložte do aktuálního (příp. Vašeho vlastního) adresáře

- zkuste si ho spustit (F5)

```
>> edit signal1
```

```
%% script generates signal with noise
clear; clc;
t = linspace(0, 5, 5*40);
s_t = sqrt(2*pi)*sin(2*pi*t) + randn(1, 5*40);
plot(t, s_t);
```

- poznámka: kód uvnitř skriptů bude dále v rámečcích bez „>>“

Užitečné funkce pro psaní skriptů

- funkce `disp` vypíše hodnotu proměnné do okna `Command Window`
  - není však vypisováno jméno proměnné a rovnítko
  - lze kombinovat s textem (později)
  - častěji je vhodné využít složitější, ale robustnější funkce `sprintf`

```
>> a = 2^13-1;
b = [8*a 16*a];
b

b =

       65528      131056
```

```
a = 2^13-1;
b = [8*a 16*a];
b
```

vs.

```
a = 2^13-1;
b = [8*a 16*a];
disp(b);
```

```
>> a = 2^13-1;
b = [8*a 16*a];
disp(b);
       65528      131056
```

- funkce `input` slouží pro zadávání proměnných
  - pokud funkce skončí chybou, je výzva k zadání proměnné opakována

```
A = input('Zadejte parametr A: ');
```

```
>> A = input('Zadejte parametr A: ');
Zadejte parametr A: 10.153
>> str = input('Zadejte parametr str: ','s');
Zadejte parametr str: toto je test
>> whos
  Name        Size                Bytes  Class

  A           1x1                     8  double
  str         1x12                   24  char
```

  - lze zadávat i text:

```
str = input('Zadejte parametr str: ', 's');
```

Matlab Editor – Příklad

600 s ↑

- vytvořte skript, který vypočte složený úrok
  - složený úrok je dán vztahem:

$$P = \frac{rA\left(1+\frac{r}{n}\right)^{nk}}{n\left(\left(1+\frac{r}{n}\right)^{nk}-1\right)},$$

kde P je pravidelná splátka z dluhu A, zaplacená n-krát za rok po dobu k let za úroku r (desetinné číslo)

- vytvořte nový skript, uložte si ho
- na počátku vymažte proměnné a obrazovku
- začněte výpočtem, vstupy (`input`) a výstupy (`disp`) řešte až poté
- pokuste se kód vektorizovat, např. pro různé hodnoty P, nebo n či k

- ověřte si výsledky (pro $A = 1000$, $n = 12$, $k = 15$, $r = 0.1$ je $P = 10.7461$)

Matlab Editor – Příklad

- zkuste si kód vektorizovat , jak pro *r*, tak pro *k*

$$P = \frac{rA\left(1+\frac{r}{n}\right)^{nk}}{n\left(\left(1+\frac{r}{n}\right)^{nk}-1\right)}$$

- pro další práci s Matlabem si již vytvářejte skripty
  - pamatujte ale, že jednotlivé části lze rychle odladit v příkazové řádce

Lineární indexace

600 s ↑

- mějte následující matici:

```
>> A = magic(4);
```

  - využijte lineárního indexování tak, aby v každé řádce nové matice `B` zbyla pouze hodnota, která má v dané řádce nejvyšší hodnotu

```
>> B = zeros(size(A));
>> % dokoncete ...
```

Užitečné funkce pro psaní skriptů

- funkce `keyboard` přeruší běh programu a umožní přístup uživateli
  - funkce je široce využívána při odlaďování programů, neboť pozastaví program v místě, kde máme pochybnosti o funkcionalitě kódu

```
K>>
```

  - prostředí `keyboard` je indikováno značkou `K>>`
  - vstup uživatele ukončíme příkazem `return`
- funkce `pause` pozastaví běh programu,
  - `pause(x)` zastaví běh programu na `x` sekund

```
% kód; kód; kód;
pause;
```

- dále také: `echo`, `waitforbuttonpress`
  - funkce naleznou využití ve specifických případech

Matlab Editor – Příklad

360 s ↑

- upravte skript s výpočtem složeného úroku tak, že
  - hodnoty *A* a *n* lze zadat z příkazové řádky (funkce `input`)
  - vyzkoušejte si funkci `keyboard` (vložte ji těsně za zadání parametrů)
    - lze parametry zadané pomoci funkce `input` změnit v prostředí `keyboard`?
    - zajistěte návrat z prostředí `keyboard` (`K>>`), užijte funkci `return`
  - skript bude zastaven před vypsáním výsledků (funkce `pause`)
    - všimněte si varování „*Paused*“ v hlavním okně Matlabu vlevo dole

Komentáře skriptů

- **KOMENTUJTE !!**
  - důležité / obtížné části kódu
  - popis funkcionality, myšlenky, změny implementace

klasický komentář (jedno-/více- řádkový)

umožňuje rozčlenit funkci do více uživatelských bloků (%% …)

```
% A     = magic(3);
matX = dataIn(:,1);
SumX = sum(matX); % all members are summed
%% CELL mode (must be enabled in Editor)
disp(num2str(SumX));
Z = inv(ZZ);
%{
This is a multi-line comment.
Mostly, it is more appropriate to use more
single-line comments.
%}
```

víceřádkový komentář

Zkratky:
CTRL+R
CTRL+T

Když nekomentujete…

- …
  tak
  se
  v
  tom
  nikdo
  nevyzná!

```
edgTotal  = MeshStruct.edgTotal;
RHO_P     = zeros(3,9,edgTotal);
RHO_M     = zeros(3,9,edgTotal);
for m = 1:edgTotal
    RHO_P(:,:,m) = repmat(MeshStruct.Rho_Plus1(:,m),[1 9]);
    RHO_M(:,:,m) = repmat(MeshStruct.Rho_Minus1(:,m),[1 9]);
end
Z         = zeros(edgTotal,edgTotal) + 1j*zeros(edgTotal,edgTotal);
for p = 1:MeshStruct.trTotal
    Plus  = find(MeshStruct.TrianglePlus - p == 0);
    Minus = find(MeshStruct.TriangleMinus - p == 0);
    D     = MeshStruct.trCenter9 - ...
                repmat(MeshStruct.trCenter(:,p),[1 9 MeshStruct.trTotal]);
    R     = sqrt(sum(D.*D));
    g     = exp(-K*R)./R;
    gP    = g(:,:,MeshStruct.TrianglePlus);
    gM    = g(:,:,MeshStruct.TriangleMinus);
    Fi    = sum(gP) - sum(gM);
    ZF    = FactorFi.*reshape(Fi,edgTotal,1);
    for k = 1:length(Plus)
        n      = Plus(k);
        RP     = repmat(MeshStruct.Rho_Plus9(:,:,n),[1 1 edgTotal]);
        RPi    = repmat(MeshStruct.Rho_Minus9(:,:,n),[1 1 edgTotal]);
        A      = sum(gP.*sum(RP.*RHO_P)) + sum(gM.*sum(RP.*RHO_M));
        Z1     = FactorA.*reshape(A,edgTotal,1);
        Z(:,n) = Z(:,n) + MeshStruct.edgLength(n)*(Z1+ZF);
    end
    for k = 1:length(Minus)
        n      = Minus(k);
        RP     = repmat(MeshStruct.Rho_Minus9(:,:,n),[1 1 edgTotal]);
        RPi    = repmat(MeshStruct.Rho_Plus9(:,:,n),[1 1 edgTotal]);
        A      = sum(gP.*sum(RP.*RHO_P)) + sum(gM.*sum(RP.*RHO_M));
        Z1     = FactorA.*reshape(A,edgTotal,1);
        Z(:,n) = Z(:,n) + MeshStruct.edgLength(n)*(Z1-ZF);
    end
end
```

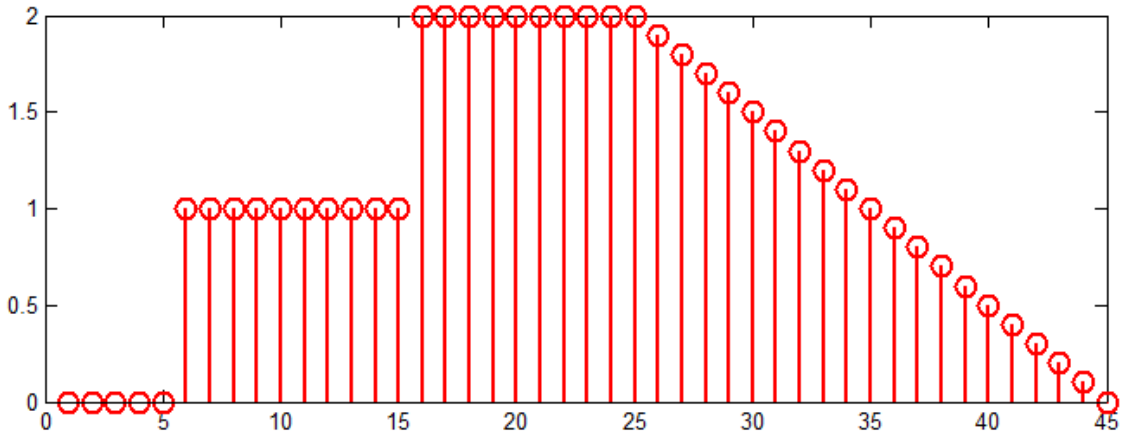
Probrané funkce

`edit`	otevře Matlab Editor	•
`disp, pause`	vypíše výsledek do příkazové řádky, pozastaví vyhodnocování kódu	•
`keyboard, return, input`	uživatelský vstup do vykonávaného skriptu, zadávání hodnot	•
`who, what, whos, which`	získávání informací o proměnných, souborech, adresářích	•
`cd, pwd, dir`	změna adresáře, výpis složky	•
`memory, ver`	paměť k dispozici Matlabu, verze Matlabu a toolboxů	•
`format, delete`	formát výpisu výsledků do příkazové řádky, smazání souborů / objektů	•

Cvičení #1

400 s ↑

- vygenerujte vektor obsahující následující posloupnost

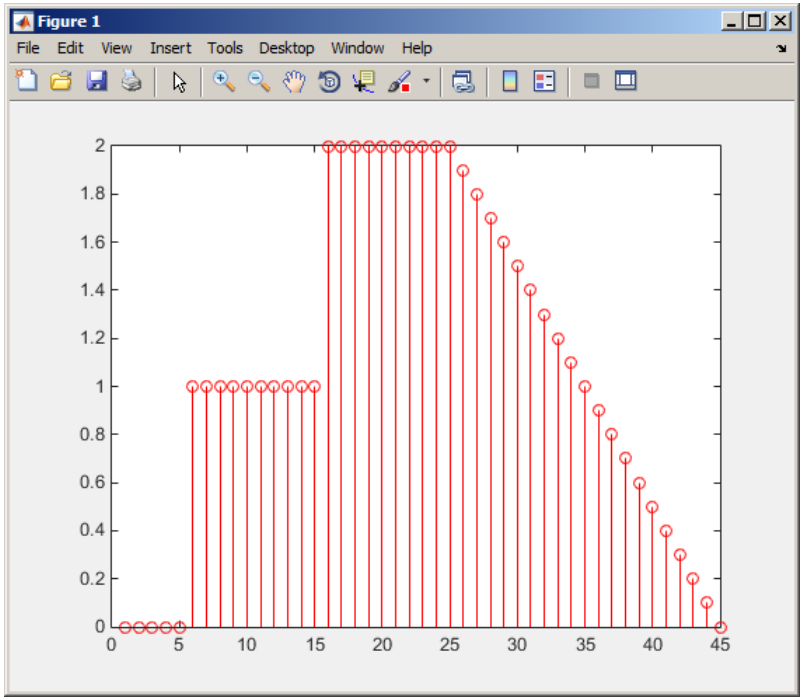


  - všimněte si osy x (intervalu, počtu vzorků)
  - rozdělte si problém na jednotlivé části, ty řešte samostatně
  - problém lze řešit několika způsoby
  - pro vykreslení využijte místo funkce `plot(x)` funkci `stem(x)`
- zkuste stejný signál generovat od nuly (včetně)…

Cvičení #2

- vygenerujte vektor obsahující následující posloupnost

- řešení např.:



- nebo

Cvičení #3

- uvažujte signál ve tvaru

$$s(t) = \sqrt{2\pi}\sin(2\omega_0 t) + n(\mu, \sigma)$$

kde střední hodnota normálního rozložení $n(\mu, \sigma)$ je $\mu = 0$ (`mu`) a směrodatná odchylka $\sigma = 1$ (`sigma`). Funkce má v Matlabu syntax:

```
n = mu + sigma*randn(1, N+V)
```

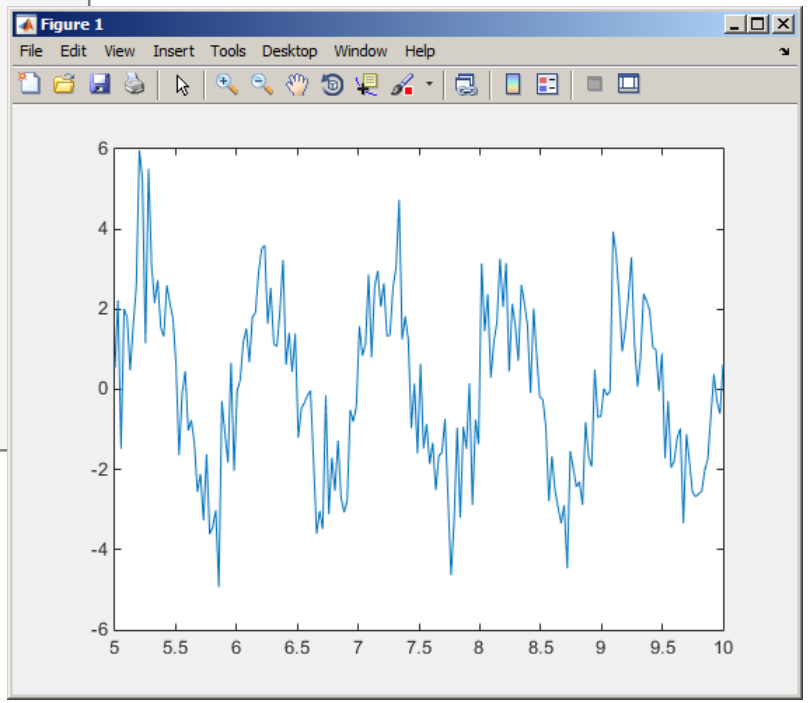
- vytvořte signál pro časový úsek $\langle 5;10 \rangle$tak, aby bylo zobrazeno $N = 5$ period signálu s $V = 40$ hodnotami na jednu periodu.

- využijte kódu na následujícím slajdu a opravte v něm chyby. Správné řešení se dozvíte příště.

Cvičení #4

400 s ↑

```
%% TVORBA VEKTORU CASU
N = 5;          % pocet period
40 = V;         % pocet vzorku na periodu
k == 5;         % zacatek interval
t = linspace[k, N+k+10), N*V];
clear;
%% TVORA VEKTORU SUMU
mu    = 2;      % stredni hodnota
sigma = 0;      % smerodatna odchylka
n     = mu + sigma*randn(1, N*V);
%% TVORBA VEKTORU ZASUMENEHO SIGNALU
omega = pi;     % uhlova frekvence
s_t = sqrt(2*pi)*Sin(2*omega*t)*n;
%% VYKRESLENI SIGNALU
plot(s_t, t)
```



- Správné řešení vykreslí toto:

Cvičení #5

400 s ↑

- koeficient odrazu S_{11} jednobranu o impedanci Z je dán vztahem:

$$S_{11} = 10\log_{10}\left(\left|\frac{Z - Z_0}{Z + Z_0}\right|\right)^2,$$

kde $Z_0 = 50\,\Omega$ a $Z = R + jX$.

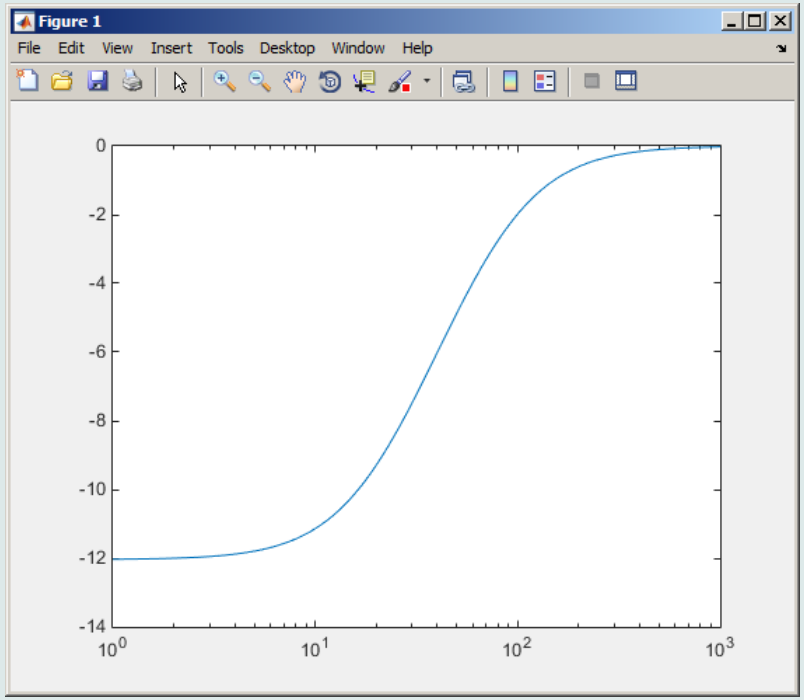
- spočtete a vykreslete závislost S_{11} pro $R = 30\,\Omega$ a X které leží v intervalu $\langle 1, 10^3 \rangle$ se sto logaritmicky rozmístěnými body

- využijte kódu níže a opravte v něm chyby. Správné řešení se dozvíte příště.

```
>> 500 = Z0;                          % vztazna impedance
>> R   == 30;                         % realna cast impedance
>> X   = Logspace(0, 3, 1e2);         % vektor reaktanci
>> clear;
>> Z   = i*(R + 1i*X);                % impedance
>> S11 = 10*log(abs(Z-Z0)./(Z+Z0))^2); % koef. odrazu v dB
>> semilogx(S11, X)                   % vykresleni s log. x-ovou osou
```

Cvičení #6

- Správné řešení vykreslí toto:



Cvičení #7

- elektromagnetická vlna o frekvenci $f = 5\,\mathrm{Hz}$ se šíří znečištěnou vodou v kladném směru osy z s konstantou šíření k. Okamžitá hodnota intenzity elektrické intenzity vlny v čase t a místě z se spočítá jako:

$$E = E_m e^{-\alpha z} \sin\left(\omega t - \beta z\right),$$

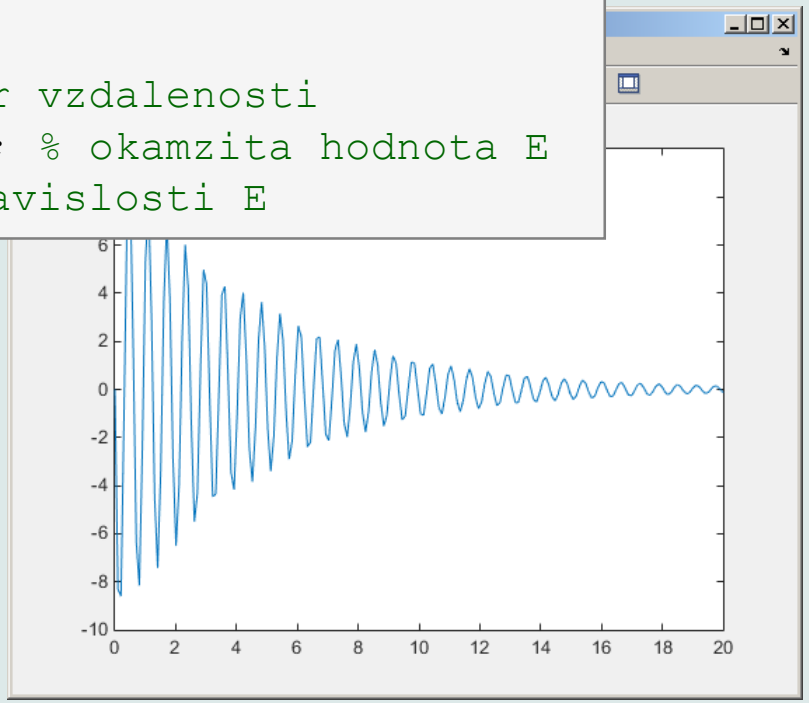
kde $E_m = 10\ \mathrm{V/m}$, $\omega = 2\pi f$ a $k = \beta - j\alpha = 10.1 - 0.21\mathrm{j}$.

  - Spočítejte a vykreslete okamžitou hodnotu E v čase $t = 10$ s na úseku osy v intervalu $\langle 0, 20 \rangle$ m ve dvou stech bodech.
  - Využijte kódu na následujícím slajdu a opravte v něm chyby. Správné řešení se dozvíte příště.

Cvičení #8

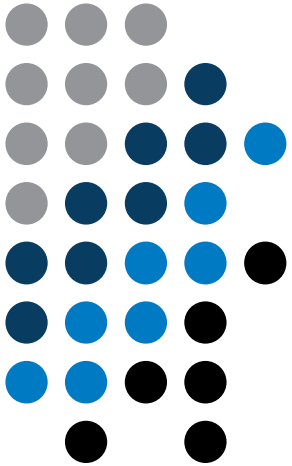
400 s ↑

```
clear;
f       = 5e9;              % frekvence
Em      = 10;               % max. hodnota intenzity E
omega   = 2pi*f;            % uhlova frekvence
k       = 1.1 - j0.21;      % konstanta sireni
beta    = real(k);          % fazova konstanta
alfa    = imag(k);          % merny utlum
t       = 10;               % cas
z       = linspace(0, 20, 20); % vektor vzdalenosti
E = Em*e^(-alfa*z)*sin(omega*t-Beta*z); % okamzita hodnota E
plot(E, z)                  % vykresleni zavislosti E
```



- Správné řešení vykreslí toto:

Děkuji!



ver. 3.4 (02/03/2015)
Miloslav Čapek
miloslav.capek@fel.cvut.cz