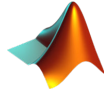


Úvod do MATLAB, Decibel



MATLAB je programové prostředí využívající skriptovací programovací jazyk. Jádrem programu je rychlý výpočet matic, které jsou základní datovou strukturou většiny aplikací. Maticových operací lze využít v širokém spektru datové analýzy, simulací a výpočtů. Programové prostředí obsahuje velké množství integrovaných i externích nástaveb pro 2D i 3D vizualizaci dat, tvorby vlastních samospustitelných programů apod. MATLAB je využíván zejména studentskou a vědeckou komunitou, ale své místo nachází i v soukromém sektoru ve vývojových a analytických centrech, kde není prioritou vytvoření konečné aplikace. Ačkoliv mnozí MATLAB nepovažují za samostatný programovací jazyk, tvorba skriptů vyžaduje znalost syntaxe, deklarace proměnných a povědomí o existujících funkcích.

Cíle úlohy:

1 Úvod do MATLAB:	1
1.1 Seznamte se s prostředím MATLAB	1
1.2 Práce s vektory a maticemi	3
1.3 Tvorba x-y grafu	16
1.4 Tvorba vlastních funkcí	19
2 Decibel [dB]	20
2.1 Úvod - Decibel	20
2.1.1 Hodnota 3 dB	22
2.1.2 Signal to Noise Ratio (SNR)	23
2.1.3 Decibely v nevykonných veličinách	23
2.2 Cvičení	24
3 Odevzdání	28

Užitečné funkce k prostudování (F1, doc, help): mean, std, median, sum, figure, clc, close, clear, plot, xlabel, ylabel, legend, hold, mldivide, mrdivide, zeros, ones, eye, rand, randn, num2str, size, length, title, disp

1 Úvod do MATLAB:

Komentář v kódu: %

Komentování jednoho nebo více označených řádků: **Ctrl + R**

Odkomentování jednoho nebo více označených řádků: **Ctrl + T**






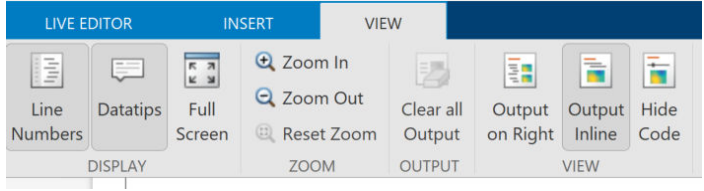
Automatické doplňování kódu: **Tab**

Zarovnání kódu: **Ctrl + I**

Zastavení výpočtu: **Ctrl + C** (při běhu programu, jinak standardně copy)

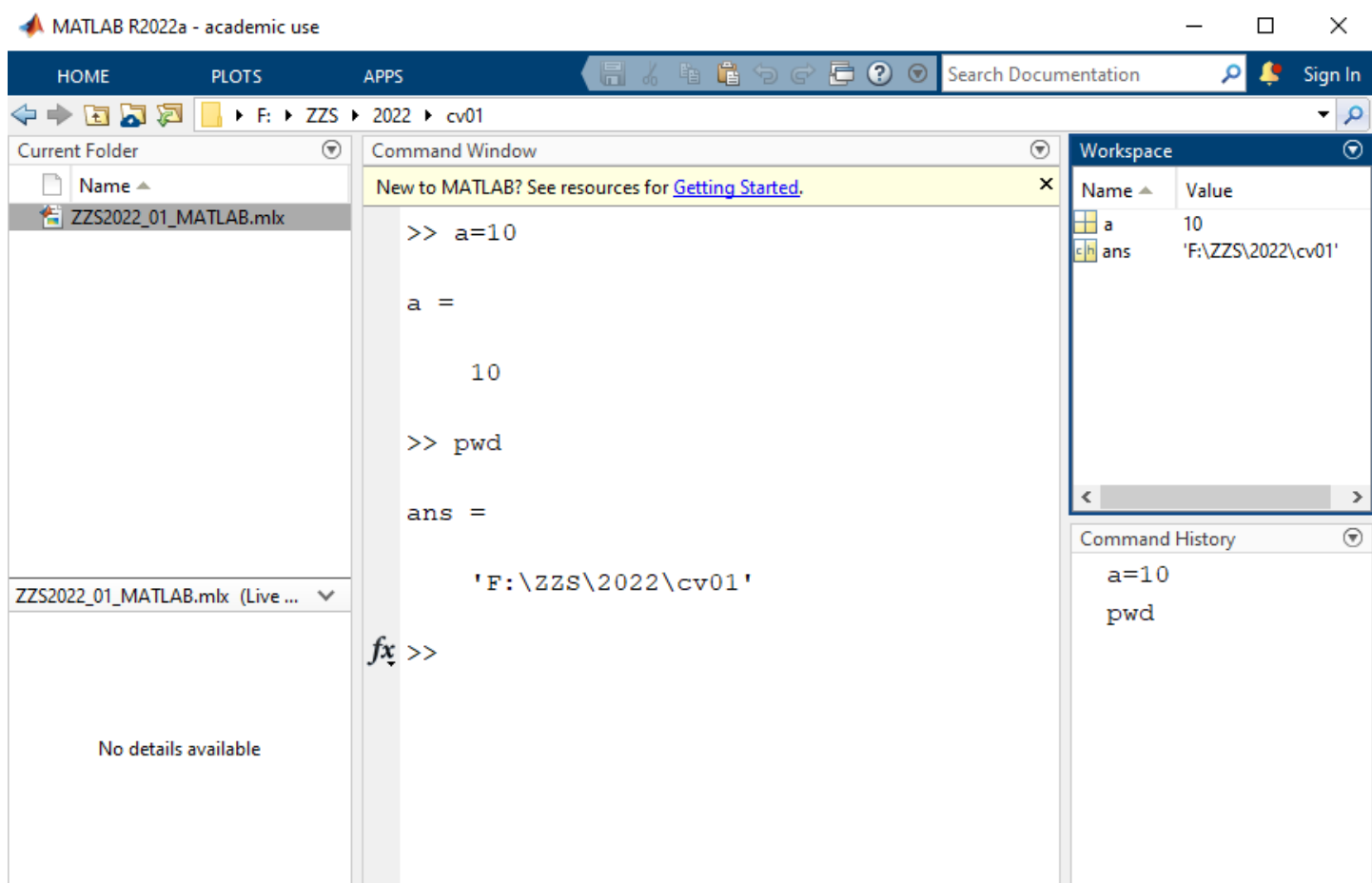
Pozn.: V MacOS/Linux se zkratky mohou lišit

1.1 Seznamte se s prostředím MATLAB

1. Vytvořte si složku a zvolte ji v MATLAB jako pracovní (Current Folder) . MATLAB při zadání příkazu nejprve pracuje s "Workspace", kde se dívá, jestli v příkazu není zadána proměnná. **Pozor**, lze si takto předefinovat některé existující funkce, které tím nadále nebudou plnit svůj účel (řešitelné smazáním proměnné z "Workspace" = kliknout na proměnnou a potom klávesu delete nebo příkazem v Command Window `clear`).
2. Při dalším kroku pracuje matlab právě s "Current Folder", kde můžete mít nadefinované vlastní funkce (stejně jako u proměnných, lze si tímto předefinovat některé existující funkce).
3. Na závěr MATLAB prochází jednotlivé složky, které jsou uloženy v "Set Path" (viz další bod). V této části jsou vidět všechny nainstalované ToolBoxy. Můžete si vytvořit složku, kde si budete ukládat data, ke kterým budete potřebovat přistupovat z několika různých skriptů. Tím, že složku s daty přidáte do "Set Path", budete mít přístup k datům ze všech Vašich skriptů. Složky se zde prochází v pořadí jak jsou seřazeny odshora dolů. Protože se funkce volají přímo jménem a nikoli jako v Pythonu, kde se importuje knihovna nebo přímo funkce, nastává zde nevýhoda MATLABu při definování stejného jména funkce. V takovém případě se volá první nalezená funkce.
4.  Preferences
 Set Path V "Preferences" si můžete nastavit vizuál prostředí MATLAB ale také nastavení Toolboxů. Vzhledem k obrázkům bez pozadí si nenastavujte pozadí na černo, protože byste potom neviděli popisky grafů přiložených v live skriptu.
5. Nový skript m-file (*.m) lze vytvořit pomocí ikony:
6.  New Script
 New Live Script Pro samostatné funkce, které budete volat v jiných skriptech, je výhodnější klasický script místo live skriptu.
7. Zde v live scriptu v záložce "VIEW" si zaškrtněte "Output Inline". Při původním nastavení "Output on Right" se Vám bude výstup (výpočty, grafy) zobrazovat napravo od kódu, což při malém rozlišení monitoru může být velice nepřehledné.
8. 

The image shows the MATLAB Live Editor toolbar with tabs for LIVE EDITOR, INSERT, and VIEW. The VIEW tab is active, showing options for zooming (Zoom In, Zoom Out, Reset Zoom), clearing output (Clear all Output), and output display (Output on Right, Output Inline, Hide Code). The Output Inline option is selected.
9. Příkazy můžete zadávat přímo v Command Window (a=10). Funkce `pwd` vypíše cestu do pracovního adresáře.
10. Části kódu lze spouštět i z editoru označením textu a stiskem (F9 pro Windows; Shift+F7 pro MacOS). Např. označte následující text a vyzkoušejte:

```
disp('Hello word!')
```



Spuštění bloku kódů se provádí modrým panelem vlevo nebo Ctrl + Entr označené buňky:

```
disp('Hello word!')
```

Hello word!

V případě livescriptu se výstup zobrazí v editoru. V případě práce v m-file se vypisuje v Command Window.

Mezi kódem a prostým textem můžete přepínat v záložce "LIVE EDITOR" pomocí tlačítek Code / Text.

Spuštění všech bloků (celého skriptu) využijte tlačítko na kartě editor "Run":



1.2 Práce s vektory a maticemi

Sledujte Workspace, kde se jednotlivé proměnné vytvářejí. Proměnná nesmí začínat číslici, ale může ji obsahovat.

```
a=10 % skalární hodnota
```

```
a =
    10
```

Při zápisu bez středníku na konci řádku se hodnota vypíše. Se zápisem se středníkem se proměnná pouze vytvoří (budeme v budoucnu preferovat). Text za % je komentář. %% odděluje celý blok (platí i pro scripty *.m, zde oddělí plnou čárou).

```
b=[1,2,3,4]; % řádkový vektor (zápis mezerou)
c=[1 2 3 4] % řádkový vektor (ekvivalentní zápis)
```

```
c = 1×4
     1     2     3     4
```

```
% pokud je na konci řádku ";" znamená to, že výstup se nevypíše v Command
% Window => při kontrole konkrétního výsledku nemusí být zapsán ";",
% nicméně při tvoření signálu o 10 000 vzorků doporučeno.
```

Vytvořte sloupcový vektor, hodnoty se oddělují středníkem.

```
d=[1;2;3;4]
```

```
d = 4×1
     1
     2
     3
     4
```

Vytvořte řádkový vektor d1 s hodnotami: 10 20 30:

```
d1 = [10 20 30]
```

```
d1 = 1×3
    10    20    30
```

Vytvořte sloupcový vektor d2 s hodnotami: 10 20 30:

```
d2 = [10;20;30]
```

```
d2 = 3×1
    10
    20
    30
```

Zápis matic využívá spojení řádkových a sloupcových vektorů. Čárka/mezera vedle sebe, středník pod sebe.

```
e=[1 2 3 4; 11 12 13 14] % [řádek ; řádek]
```

```
e = 2×4
     1     2     3     4
    11    12    13    14
```

```
% alternativní zápis:
f=[[1 2 3 4];[11 12 13 14]] % jiný zápis
```

```
f = 2×4
     1     2     3     4
    11    12    13    14
```

```
g=[1;11],[2;12],[3;13],[4;14]] % [sloupec, sloupec, sloupec, sloupec]
```

```
g = 2x4
    1     2     3     4
   11    12    13    14
```

Vytvořte matici g1 (3x4)

```
9 8 7 6
0 0 0 0
1 1 1 1
```

```
g1 = [9 8 7 6; 0 0 0 0; 1 1 1 1]
```

```
g1 = 3x4
    9     8     7     6
    0     0     0     0
    1     1     1     1
```

Skládání matic a vektorů využívá i symbolické reprezentace proměnných

```
% skládání matic
x=[10 20 30 40 50] % řádkový vektor 1x5
```

```
x = 1x5
    10    20    30    40    50
```

```
y=[0.1 0.2 0.3 0.4 0.5] % řádkový vektor 1x5
```

```
y = 1x5
    0.1000    0.2000    0.3000    0.4000    0.5000
```

```
XY_matice=[x;y] % matice: řádek y pod řádkem x
```

```
XY_matice = 2x5
    10.0000    20.0000    30.0000    40.0000    50.0000
     0.1000     0.2000     0.3000     0.4000     0.5000
```

```
XY_vektor=[x,y] % vektor: x následovaný y
```

```
XY_vektor = 1x10
    10.0000    20.0000    30.0000    40.0000    50.0000     0.1000     0.2000     0.3000 ...
```

Transpozice (')

```
% Hermitovská transpozice (u komplexních čísel budou po transpozici čísla komplexně sdružená)
ct=c' % transpozice vektoru c
```

```
ct = 4x1
    1
    2
    3
    4
```

```
et=e' % transpozice matice e
```

```
et = 4x2
     1    11
     2    12
     3    13
     4    14
```

```
% Transpozice bez komplexního sdružení:
```

```
ct=c.';
et=e.';
```

Geneze lineárně roustoucích/klesajících hodnot vektorů (:). Číselné hodnoty opět můžeme nahradit i symbolem jiné proměnné.

```
h=1:10 % od 1 do 10 s krokem 1
```

```
h = 1x10
     1     2     3     4     5     6     7     8     9    10
```

```
ii=1:1:10 % od 1 do 10 s krokem 1; i a j se používají jako imaginární jednotky pro
komplexní čísla
```

```
ii = 1x10
     1     2     3     4     5     6     7     8     9    10
```

```
jj=10:2:20 % od 10 do 20 s krokem 2
```

```
jj = 1x6
    10    12    14    16    18    20
```

```
k=10:2:19 % od 10 do 19 s krokem 2 (jaká je poslední hodnota?)
```

```
k = 1x5
    10    12    14    16    18
```

```
l=10:-1:1 % od 10 do 1 s krokem (-1)
```

```
l = 1x10
    10     9     8     7     6     5     4     3     2     1
```

```
pocatek=10;
krok=10;
konec=100;
m=pocatek:krok:konec % od 10 do 100 s krokem 10
```

```
m = 1x10
    10    20    30    40    50    60    70    80    90   100
```

Generování vektorů a matic pomocí funkcí

```
n=linspace(0,10,5) % řádkový vektor od 0 do 10 v pěti bodech
```

```
n = 1x5
     0    2.5000    5.0000    7.5000   10.0000
```

Pomocí skládání generování, transpozice a skládání vektorů vytvořte matici N1 (10x3), kde první sloupec bude nabývat hodnot v intervalu 1 až 10, druhý sloupec 100 až 10 a třetí sloupec 3 až 6.

```
a = 1:10;  
b = 100:-10:10;  
c = linspace(3,6,10);  
N1 = [a' b' c']
```

```
N1 = 10x3  
 1.0000 100.0000 3.0000  
 2.0000 90.0000 3.3333  
 3.0000 80.0000 3.6667  
 4.0000 70.0000 4.0000  
 5.0000 60.0000 4.3333  
 6.0000 50.0000 4.6667  
 7.0000 40.0000 5.0000  
 8.0000 30.0000 5.3333  
 9.0000 20.0000 5.6667  
10.0000 10.0000 6.0000
```

Generování speciálních typů vektorů a matic.

```
o=eye(5) % jednotková matice 5x5
```

```
o = 5x5  
 1 0 0 0 0  
 0 1 0 0 0  
 0 0 1 0 0  
 0 0 0 1 0  
 0 0 0 0 1
```

```
p=ones(5) % jedničková matice 5x5
```

```
p = 5x5  
 1 1 1 1 1  
 1 1 1 1 1  
 1 1 1 1 1  
 1 1 1 1 1  
 1 1 1 1 1
```

```
q=ones(5,3) % jedničková matice 5x3
```

```
q = 5x3  
 1 1 1  
 1 1 1  
 1 1 1  
 1 1 1  
 1 1 1
```

```
r=zeros(5,3,2) % nulová matice 5x3x2
```

```
r =  
r(:, :, 1) =  
  
 0 0 0  
 0 0 0
```

```
0    0    0
0    0    0
0    0    0
```

```
r(:, :, 2) =
```

```
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0
```

```
s=rand(3,5) % matice náhodných čísel 3x5 (rovnoměrná distribuce 0-1)
```

```
s = 3x5
    0.6456    0.5447    0.7210    0.2187    0.0636
    0.4795    0.6473    0.5225    0.1058    0.4046
    0.6393    0.5439    0.9937    0.1097    0.4484
```

```
s15=(5-1)*rand(3,5)+1 % matice náhodných čísel 3x5 (rovnoměrná distribuce čísel od 1 do 5)
```

```
s15 = 3x5
    2.4633    4.0879    1.7681    1.3753    4.4446
    4.0540    4.7314    1.5555    3.1016    2.9394
    3.5116    4.8910    3.7851    3.1214    2.5738
```

```
t=randn(3,5) % matice náhodných čísel 3x5 (Gaussovská/normální distribuce)
```

```
t = 3x5
    0.2358   -0.6086   -1.3429   -0.4189   -0.3001
    0.2458   -1.2226   -1.0322   -0.1403    1.0294
    0.0700    0.3165    1.3312    0.8998   -0.3451
```

```
mg = magic(4) % vytvoří matici NxN (4x4) pro N>3 o prvcích 1 až N^2
```

```
mg = 4x4
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
% - ideální na rychlé pokusy (např. s indexací)
```

Výběr prvků přímým indexováním.

```
u=10:2:30 % vektor
```

```
u = 1x11
    10    12    14    16    18    20    22    24    26    28    30
```

```
v=u(1) % 1. prvek vektoru
```

```
v =
    10
```



```
w=u([1 2 3]) % 1., 2. a 3. prvky vektoru
```

```
w = 1×3  
10    12    14
```

```
x=u(1:3) % 1. až 3. prvek (krok 1)
```

```
x = 1×3  
10    12    14
```

```
y=u(5:-1:1) % 5., 4., 3., 2. a 1.
```

```
y = 1×5  
18    16    14    12    10
```

```
z=u(10:end) % od 10. prvku až po konec s krokem 1
```

```
z = 1×2  
28    30
```

```
u_rev=u(end:-1:1) % vektor u pozpátku
```

```
u_rev = 1×11  
30    28    26    24    22    20    18    16    14    12    10
```

Indexování matic pomocí souřadnic (subscripts)

```
A=rand(10,5)
```

```
A = 10×5  
0.6834    0.8878    0.3276    0.5880    0.1117  
0.7040    0.3912    0.6713    0.1548    0.1363  
0.4423    0.7691    0.4386    0.1999    0.6787  
0.0196    0.3968    0.8335    0.4070    0.4952  
0.3309    0.8085    0.7689    0.7487    0.1897  
0.4243    0.7551    0.1673    0.8256    0.4950  
0.2703    0.3774    0.8620    0.7900    0.1476  
0.1971    0.2160    0.9899    0.3185    0.0550  
0.8217    0.7904    0.5144    0.5341    0.8507  
0.4299    0.9493    0.8843    0.0900    0.5606
```

```
B=A(5,2); % hodnota na 5. řádku a 2. sloupci
```

```
C=A(6:end,1); % hodnoty od 6. řádku po poslední v prvním sloupci
```

```
D1=A(8,:); % všechny hodnoty 8. řádku
```

```
D2=A(:, [2 4]); % všechny hodnoty 2. a 4. sloupce
```

```
E=A(1:7,4:5); % submatice mezi 1. až 7. řádkem a 4. až 5. sloupcem
```

```
F=A([1 3],1:3) % matice složená z 1. a 3. řádku mezi 1. a 3. sloupcem
```

```
F = 2×3  
0.6834    0.8878    0.3276  
0.4423    0.7691    0.4386
```

Indexování matic dle pořadí prvku (linear indices)

```
G=A(1:25); % vektor hodnot z prvních dvou sloupců (10+10) + 5 prvků 3. sloupce (+5), tj. vypíše prvních 25 prvků matice
```

Logické indexování. Vybere prvky, kde je true (1), vynechá false (0).

>, <, >=, <=, ==, ~=, &, |, &&, || % logické operátory

```
I=10:2:30; % vektor hodnot  
idx=I>20; % prvky větší než 20;
```

```
J=I(idx); % pouze prvky, kde na pozici je true, tj. >20;  
K=I(I>20) % ekvivalentní zápis
```

```
K = 1×5  
    22    24    26    28    30
```

Z matice A vyberte prvky 2. a 5. řádku a uložte do proměnné A1

```
A1 = A([2 5], :)
```

```
A1 = 2×5  
    0.7040    0.3912    0.6713    0.1548    0.1363  
    0.3309    0.8085    0.7689    0.7487    0.1897
```

Z matice A vyberte pouze rohové prvky a uložte je do proměnné A2 (existuje více způsobů, jak to udělat)

```
A2 = A([1 end], [1 end])
```

```
A2 = 2×2  
    0.6834    0.1117  
    0.4299    0.5606
```

V matice MG prohodte 1. sloupec s 3. sloupcem použitím *indexace*.

```
%%% Nápořěda:  
% Výběř prvního sloupce:  
MG(:,1)  
% Výběř prvního a druhého sloupce:  
MG(:,[1 2])  
% Výběř druhého a prvního sloupce:  
MG(:,[2 1])  
% lze takto vložit do rovnosti, a tedy rovnou prohodit sloupce
```

```
MG=magic(5)
```

```
MG = 5×5  
    17    24     1     8    15  
    23     5     7    14    16  
     4     6    13    20    22  
    10    12    19    21     3
```

11 18 25 2 9

```
MG = MG(:, [3 2 1 4 5])
```

MG = 5x5

1	24	17	8	15
7	5	23	14	16
13	6	4	20	22
19	12	10	21	3
25	18	11	2	9

Nastudujte si funkce (příkaz doc, F1): reshape, eye

Pozn. : Všimněte si, jakého typu je výstup funkce eye a proto jej nelze přímo použít jako indexace. Indexovat lze pouze logickou proměnnou (True/False) nebo čísla 1, 2, ..., N×M, kde N je počet řádků a M je počet sloupců matice (případně rozšířeno o další dimenze matice).

Vytvořte matici G (5x5) obsahující v řádcích hodnoty od 1 do 25. Pomocí logických operací a jednotkové matice eye vyberte prvky na digonále a uložte jako vektor G1. V praxi lze použít také funkce diag.

G = 5x5	G1= 5x1
1 2 3 4 5	1
6 7 8 9 10	7
11 12 13 14 15	13
16 17 18 19 20	19
21 22 23 24 25	25

```
G = 1:25;  
G = reshape(G,[5 5])'
```

G = 5x5

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

```
idx = eye(5)==1;  
G1 = G(idx)
```

G1 = 5x1

1
7
13
19
25

%...

Zápis prvků do matice využívá stejných pravidel indexování jako jsme používali výše: přímé, symbolické, logické.

Pro vektory nebo matice dle pořadí.

```
L=10:20
```

```
L = 1×11
    10    11    12    13    14    15    16    17    18    19    20
```

```
L(L>=15)=Inf % kde je L>15, nahraď nekonečnem
```

```
L = 1×11
    10    11    12    13    14    Inf    Inf    Inf    Inf    Inf    Inf ...
```

```
L(1:2)=0 % na 1. a 2. místo zapiš nulu
```

```
L = 1×11
     0     0    12    13    14    Inf    Inf    Inf    Inf    Inf    Inf ...
```

```
L(2:5)=[1 2 3 4] % 2. až 5. prvek nahraď vektorem
```

```
L = 1×11
     0     1     2     3     4    Inf    Inf    Inf    Inf    Inf    Inf ...
```

```
L(L>4)=[] % tam, kde je hodnota >4, vlož prázdnou hodnotu, tj. vymaž prvek
```

```
L = 1×5
     0     1     2     3     4
```

Pro matice indexováním souřadnic

```
LL=[1 2 3; 4 5 6; 7 8 9; 10 11 12]
```

```
LL = 4×3
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

```
LL(1:3,end)=100 % 1. až 3. řádek posledního sloupce nahraď hodnotou 100
```

```
LL = 4×3
     1     2    100
     4     5    100
     7     8    100
    10    11    12
```

Vygenerujte nulovou matici L1(6x6) a do čtvrtého kvadrantu (vpravo dole) vložte jednotkovou matici (3x3). Využijte funkcí zeros a eye.

```
L1 = 6×6
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

```
L1 = zeros(6);
```

```
L1(4:6,4:6) = eye(3)
```

```
L1 = 6x6
```

```
0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    1    0    0
0    0    0    0    1    0
0    0    0    0    0    1
```

Matematické operace. Sčítání (+), odčítání (-) a skalární násobení (.*) se provádí pro jednotlivé prvky matic, proto musí souhlasit jejich velikost, nebo alespoň rozměr v jedné dimenzi. Vyjímkou je operace s maticí a konstantou.

```
M=[1 2 3 4]
```

```
M = 1x4
```

```
1    2    3    4
```

```
N=[11 12 13 14]
```

```
N = 1x4
```

```
11    12    13    14
```

```
O=M+N % velikost matic musí souhlasit
```

```
O = 1x4
```

```
12    14    16    18
```

```
OO=M+10 % každému prvku M se přičte 10
```

```
OO = 1x4
```

```
11    12    13    14
```

```
P=N-M % velikost matic musí souhlasit
```

```
P = 1x4
```

```
10    10    10    10
```

```
% skalární násobení (po prvku)
```

```
Q=M.*N
```

```
Q = 1x4
```

```
11    24    39    56
```

```
S=M*10 % každý prvek M vynásobený 10
```

```
S = 1x4
```

```
10    20    30    40
```

```
% dělení (po prvku)
```

```
T=N./M
```

```
T = 1x4
```

11.0000 6.0000 4.3333 3.5000

U=M/10 % každý prvek M vydělený 10

U = 1×4
0.1000 0.2000 0.3000 0.4000

% 3. mocnina po prvku
V=M.^3

V = 1×4
1 8 27 64

% 2. odmocnina po prvku
W=M.^(1/2)

W = 1×4
1.0000 1.4142 1.7321 2.0000

Operace matic jiných velikostí, ale stejným rozměrem v jedné z dimenzí.

M24=[M;N] % 2×4

M24 = 2×4
1 2 3 4
11 12 13 14

K21=[2;10] % 2×1

K21 = 2×1
2
10

O24=M24+K21 % ke každému sloupci matice M24 přičte vektor K21

O24 = 2×4
3 4 5 6
21 22 23 24

P24=M24.*K21 % každý prvek sloupec matice M24 je násoben prvkem z K21

P24 = 2×4
2 4 6 8
110 120 130 140

Vektorové násobení.

%X=M*N % Error: oba jsou řádkové, matice nelze vynásobit - po vyzkoušení zakomentujte

Je potřeba, aby počet sloupců M byl shodný s počtem řádků N. Buď tedy transponujte N a vznikne skalární součin (skalár), nebo transponujte M a vznikne dyáda (outer product) o rozměru 4×4.

U násobení matic platí obdobná pravidla. Operace vrací matici jednotlivých vektorových násobení

```
Y=M24*M24'
```

```
Y = 2x2
    30    130
    130    630
```

Vektorové dělení.

Z lineární algebry víme, že operace maticového dělení je násobením s inverzní maticí:

```
MM = randn([4 4]) % matice náhodných čísel 4x4
```

```
MM = 4x4
    0.3984    0.6830    0.0226   -0.0029
    0.8840    1.1706   -0.0479    0.9199
    0.1803    0.4759    1.7013    0.1498
    0.5509    1.4122   -0.5097    1.4049
```

```
NN = randn([4 4]) % matice náhodných čísel 4x4
```

```
NN = 4x4
    1.0341   -1.3826    0.8797    0.6417
    0.2916    0.2445    2.0389    0.4255
   -0.7777    0.8084    0.9239   -1.3147
    0.5667    0.2130    0.2669   -0.4164
```

```
% U=inv(M)*N;
Z=MM\NN % též funkce mldivide (ml ~ multiply from left by inverse matrix)
```

```
Z = 4x4
   -0.6114    0.9493    3.7667    1.8399
    1.8903   -2.5954   -0.9238   -0.1112
   -0.7853    0.8627    0.4205   -0.8303
   -1.5420    2.7013   -0.2057   -1.2072
```

```
% V=N*inv(M);
ZZ=NN/MM % též funkce mrdivide (mr ~ multiply from right by inverse matrix)
```

```
ZZ = 4x4
   -4.7312    5.1408   -0.1453   -2.9033
   -1.0943    0.8112    1.1306   -0.3511
    4.0801   -3.8072    0.8243    1.4773
    0.6300    0.9359   -0.0941   -0.8978
```

Tento zápis s použitím operátorů \, / je v MATLAB efektivnější a přesnější než výpočet pomocí inverzní matice a násobení.

Vyřešte maticově soustavu rovnic o dvou neznámých x_1 a x_2 bez funkce inv.

$$2x_1 + x_2 = 3$$

$$x_1 - x_2 = 2$$

Převédeme do maticového zápisu

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}; a \in \mathbf{A}, x \in \mathbf{X}, b \in \mathbf{B}$$

$$\mathbf{Ax} = \mathbf{B}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{B}$$

```
x = 2x1
    1.6667
   -0.3333
```

```
AA = [2 1 ; 1 -1];
BB = [3 ; 2];
x= AA\BB
```

```
x = 2x1
    1.6667
   -0.3333
```

1.3 Tvorba x-y grafu

MATLAB obsahuje desítky [typů grafů](#) (lze označit proměnnou ve Workspace a následně v záložce PLOTS vybrat požadovaný graf).

My si prozatím vystačíme spojnicovým X-Y realizovaný funkcí `plot`. Detaily naleznete v dokumentaci.

```
% Zadejte do Command Window
doc plot
```

Funkce může mít jeden, dva nebo i více vstupů, tak jako většina MATLAB funkcí. Chování se poté může významně lišit, nebo umožňuje mnoho dalších nastavení. Proto je třeba pozorně studovat dokumentaci.

Grafy můžeme vytvářet v jednotlivých oknech - figure. Jednotlivé figury lze rozdělit do sekcí - subplot. Graf poté vykresluje v daném subplot. Toto členění oceníte zejména u psaní vlastního kódu v m-files (*.m). V livescript (*.mlx) se zobrazuje výstup uvnitř skriptu.

```
doc figure
doc subplot
```

```
figure(1); % vytvoří okno s číslem 1
clf; % vyčistí figuru 1, pokud něco obsahuje z předchozího spuštění
subplot(1,2,1) % vytvoří 2 podokna, 1 řádek, 2 sloupce. Poslední parametr je místo,
kam budu zapisovat
```

Zobrazme si čáru spojující dva o souřadnicích [0 0] a [2 3]

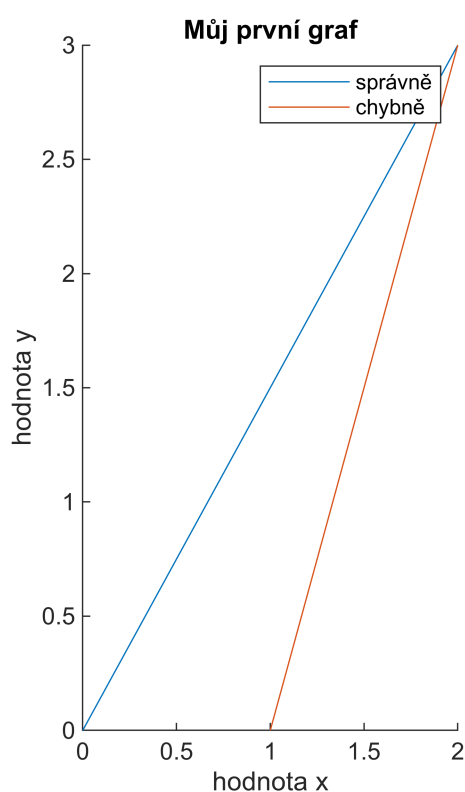
```
x=[0 2]; % x
y=[0 3]; % y=f(x)
hold on
plot(x,y) %
hold off
```


Pokud bychom do funkce zadali pouze jedne vstup y, hodnoty x by se předpokládaly dle pořadí [1 2]. Barva nového grafu se volí automaticky.

```
hold on % umožní uchování předchozího grafu a přidá do stejného grafu nový  
plot(y) % zobrazí hodnoty y, ale za x dá pořadí 1 2 ...  
hold off % při opětovném spuštění skriptu by se přidávalo znovu proměnné do obrázku,  
% hold off vypne přidávání dalších grafů
```

Přidáme popisky os:

```
xlabel('hodnota x')  
ylabel('hodnota y')  
title('Můj první graf')  
legend('správně', 'chybně') % přidá legendu
```



Pozn.: Popisy os a nadpis se nastavuje pro každý graf v subplot zvlášť.

Pozn. pro zájemce: Od verze MATLAB 2019b existuje alternativní způsob tvorby grafů a podgrafů pomocí funkcí `tiledlayout` a `nexttile`. Pro většinu běžných grafů jsou oba způsoby stejně vhodné, `tiledlayout` může být výhodnější u tvorby složitých grafů s více podgrafy a různými typy grafů. Je na Vás, který způsob budete využívat, v zadáních úkolů bude využit klasický způsob pomocí `figure` a `subplot`.

Nyní to zkusme znovu a do druhého podokna vykreslíme X-Y graf výsledku soustavy dvou rovnic z předchozí úlohy. Jedná se vlastně o parametrické vyjádření přímky.

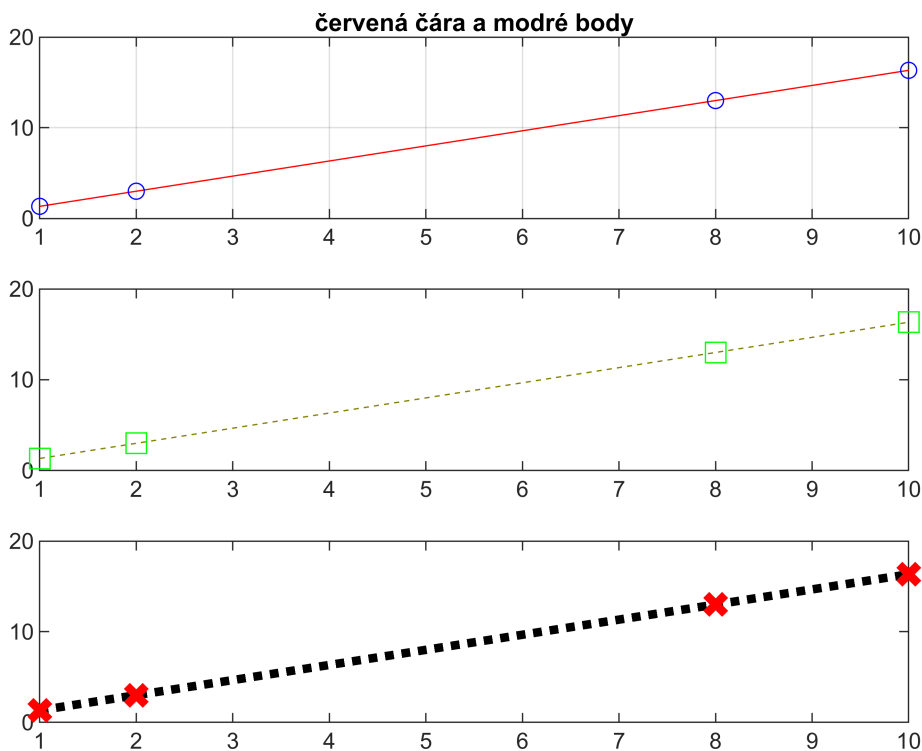
$$y = 1.6667x - 0.3333$$

Nyní si můžeme zvolit libovolné souřadnice x a vykreslíme k nim hodnotu y. Další parametry se mohou zadávat ve zkratkách, viz dokumentace. Můžete ale nastavit složitější parametry v [Line Properties](#)

```
x=[1 2 8 10];
y=1.6667*x-0.33;
figure(2)
subplot(3,1,1)
plot(x,y,'r') % písmenný třetí operátor definuje bravu (r-red)

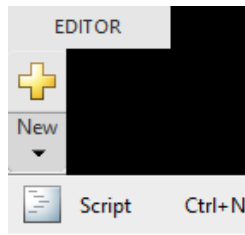
hold on
plot(x,y,'ob') % 'o' zobrazí body bez spojnice, (b-blue)
hold off
grid on % rastr
title('červená čára a modré body')

subplot(3,1,2)
plot(x,y,'Color',[0.5 0.5
0], 'LineStyle','--', 'Marker','s', 'MarkerSize',10, 'MarkerEdgeColor','g')
% typ LineStyle, Marker a Color lze napsat rovnou jako jeden samostatný parametr
% hned po vypsání os
subplot(3,1,3)
plot(x,y,':xk', 'MarkerSize',10, 'MarkerEdgeColor','r', 'LineWidth',3.2)% LineWidth
nastaví šířku čáry, zde na velikost 3.2 bodu
```



1.4 Tvorba vlastních funkcí

Při tvorbě **vlastních funkcí** se nejčastěji využívá m-file (*.m), který se musí jmenovat stejně jako volaná funkce.



Vytvořte nový skript:

Na první řádek vytvořte hlavičku funkce, její výtupy, název a vstupy:

```
function [out1, out2, out3]=kos_prumer(in1, in2) % počet vstupů a výstupů může být libovolný
```

Nyní si vytvořte funkci `kos_prumer.m`, která bude počítat ze zadaných známek `c` a kreditů `w` za předmět vážený průměr známek. Doplňte vnitřní kód funkce, který spočítá výstupy `avr` (aritmetický průměr), `sd` (směrodatnou odchylku), `med` (medián) a `wavr` (vážený průměr). Nalezněte v dokumentaci vhodné funkce, které `avr`, `sd` a `med` počítají. Pro výpočet váženého průměru `wavr` využijte skalárního násobení/dělení a funkce `sumy`.

```
function [avr,sd,med,wavr]=kos_prumer(c,w)
% popis funkce, který se vypíše, když zadáte help k funkci
% můžete zde psát rovnice v LaTeX stylu
% můžete také zadávat seznamy, příklady jak se má funkce správně volat, atd.
% na help se vypíše všechny řádky začínající procentem, které jsou přímo pod function

% tzn. že tento řádek a následující se již nevypíše
avr=...; % aritmetický průměr
sd=...; % směrodatná odchylka
med=...; % medián
wavr=...; % vážený průměr
```

$$W = \frac{\sum_{n=1}^N w(n)c(n)}{\sum_{n=1}^N w(n)}, \text{ kde } W \text{ je vážený průměr, } c \text{ hodnota, } w \text{ její váha a } N \text{ počet průměrovaných hodnot}$$

Funkci uložte do pracovního adresáře jako `kos_prumer.m`.

Nyní si vytvořte vektor `c` známek a odpovídajících kreditů `w` a funkci zavolejte:

```
c=[1 2 3 4 5];
w=[5 4 3 2 1];
[avr,sd,med,wavr]=kos_prumer(c,w);
```

```
avr =
3
sd =
1.2472
med =
3
wavr =
2.3333
```

Pozn.

V drtivé většině případů si budete vytvářet vlastní funkce a ukládat je jako samostatné .m soubory. Ty budete volat z příkazového řádku nebo jiného skriptu (.mlx nebo .m). Vyjímkou však mohou být jednoduché a jednorázově používané funkce, které lze do skriptu vložit jako anonymní funkci *@fun* před tím, než ji voláme.

Lze vytvářet anonymní funkce podobně jako v jazyce Python (lambda x:).

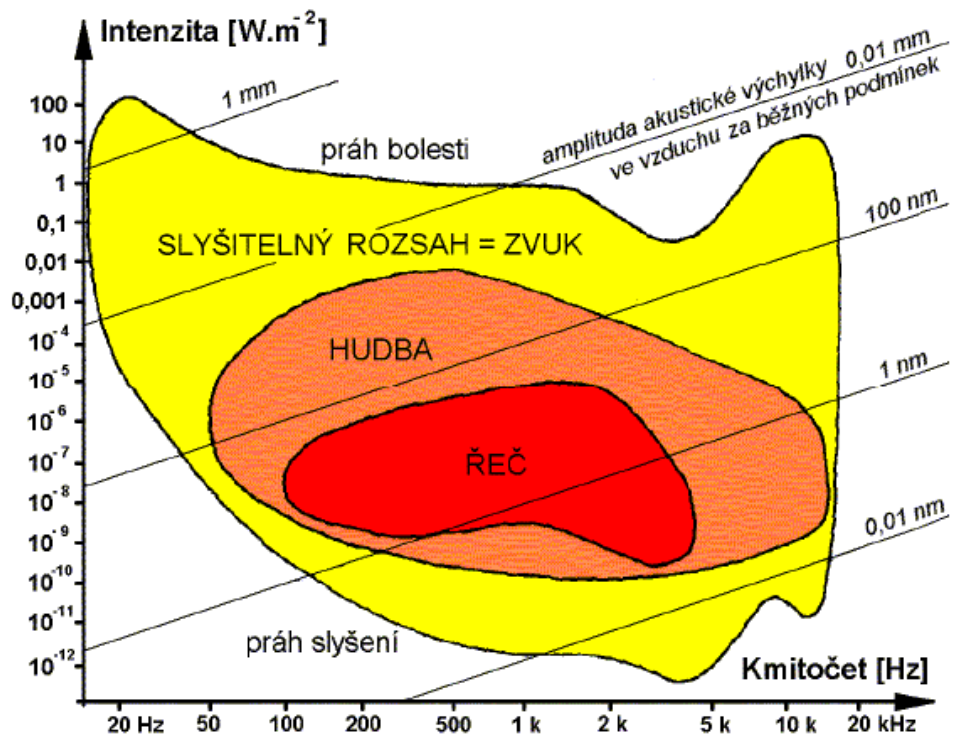
```
anonymni_fce_vypis_mean_std = @(x) ['value = ' num2str(mean(x)) ' +- ' num2str(std(x))]  
anonymni_fce_vice_promenych = @(x,y) sum(x) + sum(y)  
% jde tvořit odkaz na existující funkce  
odkaz_mean = @mean  
  
% příklady volání  
X1=anonymni_fce_vypis_mean_std([1 2 3 4 5])  
X2=anonymni_fce_vice_promenych([1 2 3],[-1 -2 -3])  
X3=odkaz_mean([10 20 30 40])
```

Pozn.: Pokud chceme mít funkci (jako je např. vytvořená *kos_prumer*) součástí spouštěného skriptu, musí být kód funkce vložen až na konci skriptu, avšak skript nelze poté spouštět po sekcích, ale jen jako celek.

2 Decibel [dB]

2.1 Úvod - Decibel

Ačkoli jednotka "bel", respektive "decibel", nepatří do jednotek SI, je v technice široce využívána kvůli logaritmické škále. Původně vznikla pro popis lidského sluchu, protože subjektivní vnímání člověka není lineární, nýbrž logaritmické. Člověk je schopen vnímat intenzitu zvuku od sotva slyšitelného šumu (práh slyšení) až po extrémní hluk (na prahu bolesti). Hranice slyšitelnosti odpovídá akustické intenzitě zhruba 1 pW/m^2 pro 1 kHz tón.



Zdroj: [Česká komora autorizovaných inženýrů a techniků činných ve výstavbě](#)

Při výpočtu dB se využívá vzorce:

$$Výsledek[dB] = 10 \cdot \log_{10} \frac{Měření}{Reference} [-]$$

Poměr proměnné (měřené) veličiny a její reference mají stejnou jednotku, je tedy logaritmován jejich bezrozměrný poměr. **Decibel je tedy ve výsledku také bezrozměrný** (stejně jako např. procenta).

Na frekvenci 1 kHz je člověk schopen od sebe rozeznat dvě různé intenzity hlasitosti právě s odstupem 0,1 bel (0,1 B). Pro snadnější zápis hodnot v celých číslech se zavedla podjednotka "decibel" (0,1 B = 1 dB), který je dosažen násobením logaritmu deseti ve vzorci.

Pozn.: V různých aplikacích se můžeme setkávat s dalšími variantami značení dB, kdy doplňuje i hodnotu použité reference. Například pro referenci v miliwattech (mW) se využívá značení dBm, zisk antény dBi aj.

Spočtete a odpovězte:

1) Převod na dB. Referenční hodnota je dle definice 1 pW/m². Kolik je 100 pW/m² v dB?

Funkce: log, exp, log10, disp

```
% Definice proměnných
reference = 1; % pW / m^2 ... je vhodné si do komentáře psát jednotky kvůli převodu
mereni = 100; % pW / m^2

% varianta 1. (zápis výpočtu)
```

```
% Vzorec pro dB
promenna_v_dB=10*log10(100/1); % log je logaritmus o základu e (přirozený logaritmus)

% varianta 2. (využití anonymní funkce)
% můžeme zadefinovat anonymní funkci, abychom mohli výpočet nadále snadno replikovat
vypocet_dB = @(x) 10*log10(x / reference); % definice anonymní funkce
promenna_v_dB = vypocet_dB(mereni) % kde mereni je hodnota v pW / m^2

% vypíše výsledek
disp([num2str(promenna) ' pW/m^2 = ' num2str(promenna_v_dB) ' dB'])
```

Přepočtete 100 pW/m² na dB:

```
vykon_db = 10*log10(100/1)
```

```
vykon_db =
20
```

2) Převod z dB. Pokud 0 dB odpovídá 1 pW/m², kolik je 130 dB v pW/m²? Výpočet proveďte v MATLABu.

Převod z decibelů provedeme odlogaritmováním $\log_{10}X \approx 10^X$:

$$\text{Měření} \left[\frac{\text{pW}}{\text{m}^2} \right] = \text{Reference} \cdot 10^{\frac{\text{Výsledek \{dB\}}{10}}$$

Pozn.: desetina v exponentu odpovídá převodu z decibel na bel.

```
% využijte zápis výpočtu nebo vytvořte anonymní funkci přepočtu dB -> pW / m^2
```

```
mereni = 1*10^(130*0.1)
```

```
mereni =
1.0000e+13
```

3) Zde stačí napsat přímo odpověď (např. **5krát větší**, atd.).

a) Výsledek je –3 dB. Kolikrát je výstupní výkon menší / větší oproti vstupu?

poloviční (0,5012krát)

b) Výsledek je 10 dB. Kolikrát je výstupní výkon menší / větší oproti vstupu?

10krát větší

c) Výsledek je 20 dB. Kolikrát je výstupní výkon menší / větší oproti vstupu?

100krát větší

2.1.1 Hodnota 3 dB

U sluchu jsou 3 dB minimální hodnotou intenzity, kdy je člověk schopen rozlišit hlasitost **dvou hladin šumu** (oproti 1 dB pro 1 kHz tón). Tato hodnota je velice subjektivní a někdy se uvádí i vyšší hodnota (např. 5 dB, [můžete vyzkoušet sami](#)). Je to hodnota, kdy se obecně **výkon** zvýší dvakrát (+3 dB), resp. zmenší na polovinu (-3 dB), tj. v závislosti na znaménku => odpověď u otázky 3.a budou mít všichni správně! :). V technice se tato hodnota ustálila jako arbitrární práh, který např. definuje místo zlomového kmitočtu u filtrů (např. RC, LC články apod., detailněji v dalších týdnech).

2.1.2 Signal to Noise Ratio (SNR)

V měření, záznamu a reprodukci signálu je vždy užitečná informace zkreslená všudypřítomným šumem. Pro kvantifikaci poměru informace a šumu se používá hodnota **Signal to Noise Ratio (SNR)**, česky **odstup signálu od šumu**, která vyjadřuje poměr **výkonu** užitečného signálu od **výkonu** šumu. Rozdíl mezi signálem a šumem chceme co největší (tj. maximálně potlačit šum). Hodnota SNR může tedy nabývat hodnot v široké škále, proto s výhodou také převádíme logaritmem na dB. Podobně jako u úpravy jednoty dB podle použité reference, i značení SNR se upravuje pro konkrétní případ o jaký šum se jedná. Např. pro kvantizační šum se veličina značí SNRQ. Výpočet se pro konkrétní případ také může upravovat nebo zjednodušovat, platí ale základní definovaný [výše uvedený vzorec](#).

2.1.3 Decibely v nevýkonných veličinách

Hodnoty dB jsou vždy definovány pro poměr veličin výkonu. Avšak v mnoha aplikacích je využíván poměr nevýkonných veličin, např. napěťové nebo proudové zesílení zesilovače $\left(\frac{U_{\text{out}}}{U_{\text{in}}}, \frac{I_{\text{out}}}{I_{\text{in}}}\right)$, přenos systému apod.

Např. elektrické napětí je veličina práce. Proto musíme vzorec výpočtu upravit, pokud chceme vypočítat hodnoty poměru napětí v dB. Elektrický výkon (P) je spočten jako násobek napětí (U) a proudu (I). Protože v biologických signálech většinou měříme jen jednu veličinu (typicky elektrický potenciál, tj. napětí), dosadíme za proud pomocí Ohmova zákona výpočet proudu pomocí elektrického odporu:

$$P = U \cdot I = U \cdot (U/R) = U^2/R$$

Výsledek dosadíme do původního vzorce:

$$\text{Proměnná[dB]} = 10 \cdot \log_{10} \left(\frac{U_{\text{Měření}}^2 / R[W]}{U_{\text{Reference}}^2 / R[W]} \right) = 10 \cdot \log_{10} \left(\frac{U_{\text{Měření}}[V]}{U_{\text{Reference}}[V]} \right)^2 = 20 \cdot \log_{10} \left(\frac{U_{\text{Měření}}[V]}{U_{\text{Reference}}[V]} \right)$$

Všimněte si, že kvadrát napětí se přesune před logaritmus, který je pro nevýkonných veličiny násoben 20. Dostáváme tak vzorec $X \text{ [dB]} = 20 \cdot \log_{10}(X)$.

Referenci bereme většinou 1 V (pokud není uvedeno jinak). Potom lze měřené napětí uvádět v dB:

$$U_{\text{Měření}}[\text{dB}] = 20 \cdot \log_{10} U_{\text{Měření}}[\text{V}]$$

Obdobně přenos systému (např. zesílení zesilovače, útlum filtru) zapisujeme v dB, kde referenční hodnotu

reprezentuje hodnota na vstupu a měřená na výstupu $A = \frac{U_{\text{out}}}{U_{\text{in}}}$ apod.:

$$A \text{ [dB]} = 20 \cdot \log_{10} A$$

Odpovězte (opět stačí rovnou slovně, pokud jste si jistí na 100 %):

1) Jaké hodnotě napětí odpovídá hodnotě 3 dB při referenci 1V?

2V

2) Jaké hodnotě napětí odpovídá hodnotě 6 dB při referenci 1V?

3,98V (vypočet v příkladu 2.1.2 Převod z dB)

3) Zesilovač zesílí vstupní napětí 10.000×. Jaký je zisk zesilovače v dB?

40dB (4 řády = 4B = 40dB)

2.2 Cvičení

Načtěte signál a vytvořte grafy podle vzoru.

1)

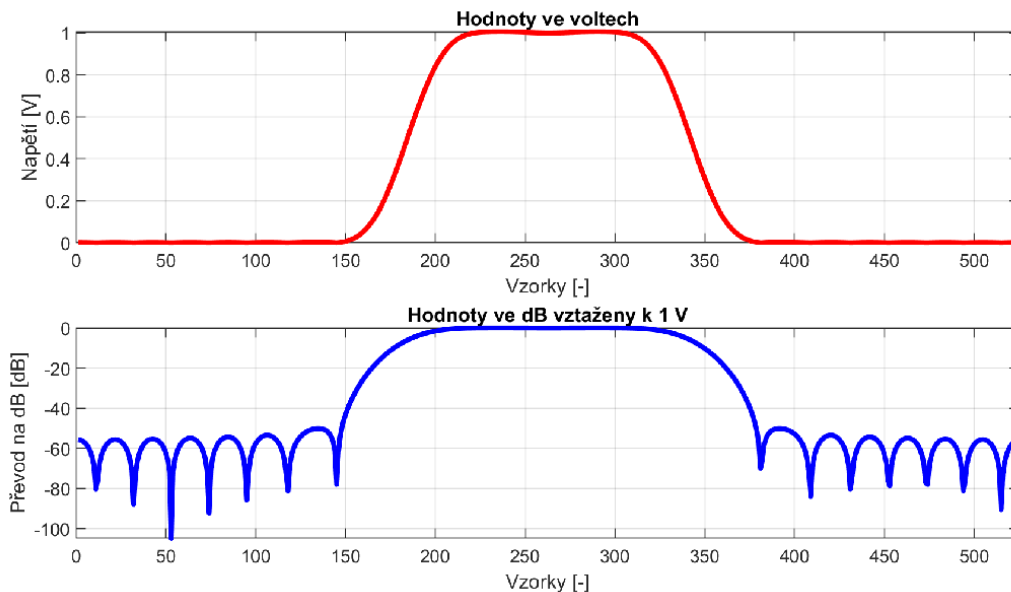
Nápověda: log10, figure, subplot, plot, hold, xlim, xlabel, ylim, ylabel, grid, legend, title

Napětíový průběh jako takový v praxi **nebudeme** převádět na dB. Cílem je zde vyzkoušet a procvičit si převod na dB a vykreslení grafů v MATLABu. Později budeme převod a toto vykreslení využívat u filtrů.

Načtení signálu z .mat souboru lze udělat přímo v kódu. Musíte ale mít soubor vidět v "Current Folder" nebo pomocí "Set Path" nastavit cestu k datům ([viz úvod](#)).

```
%% Načtení dat
load ZZS_cv01_data.mat
%% Soubor obsahuje jednu proměnnou y, která reprezentuje napětí ve Voltech.
%% Osu x můžete vygenerovat jako vektor nebo na přímo vykreslit hodnoty do
%% plot.
% x = 1:length(y);
```

Vytvořte graf podle vzoru



Obrázek má dva subploty.

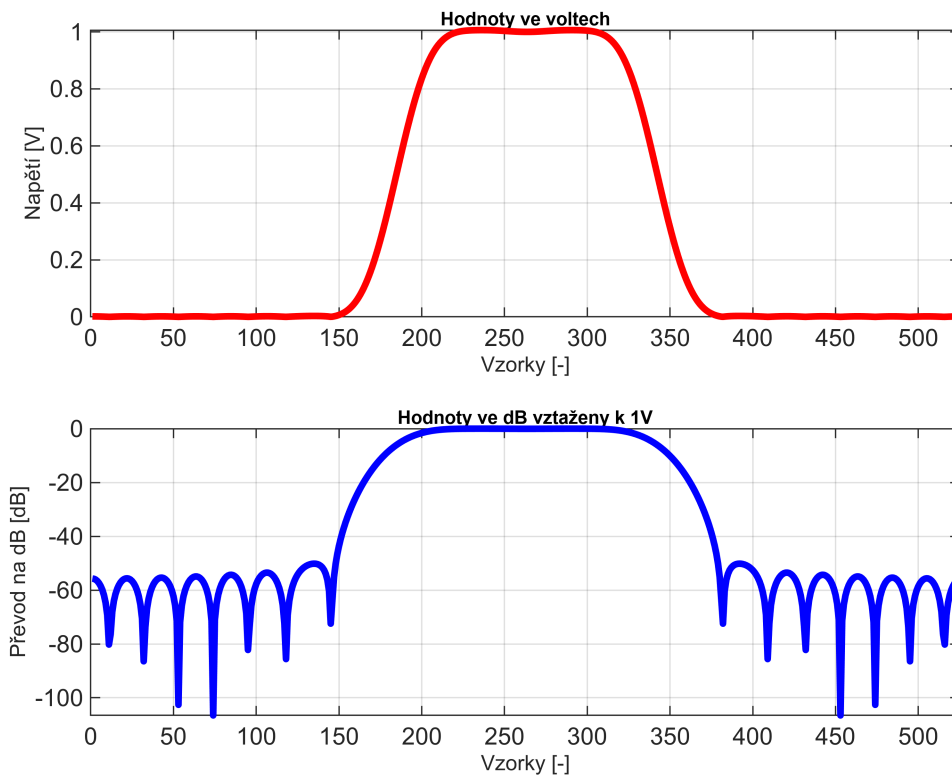
1. První subplot má hodnoty ve Voltech (přímo samotná proměnná y) bude červenou barvou s šířkou čáry 2.5.
2. Druhý subplot má hodnoty ve dB (nutno převést y na dB podle posledního [vztahu uvedeného v 2.1.3](#)) bude modrou barvou s šířkou čáry 2.5.

Popište osy a nadpis grafu, nastavte limity osy x od 0 po 525 a vykreslete rastr.

```
figure(3)

subplot(2,1,1)
plot(y,"LineWidth",2.5,"Color","r")
xlim([0 525])
grid on
title("Hodnoty ve voltech", "FontSize",7)
ylabel("Napětí [V]", "FontSize",8)
xlabel("Vzorky [-]", FontSize=8)

subplot(2,1,2)
data_db = 20*log10(y);
plot(data_db,LineWidth=2.5,Color="b")
xlim([0 525])
ylabel("Převod na dB [dB]", "FontSize",8)
xlabel("Vzorky [-]", FontSize=8)
title("Hodnoty ve dB vztaženy k 1V",FontSize=7)
grid on
hold off
```

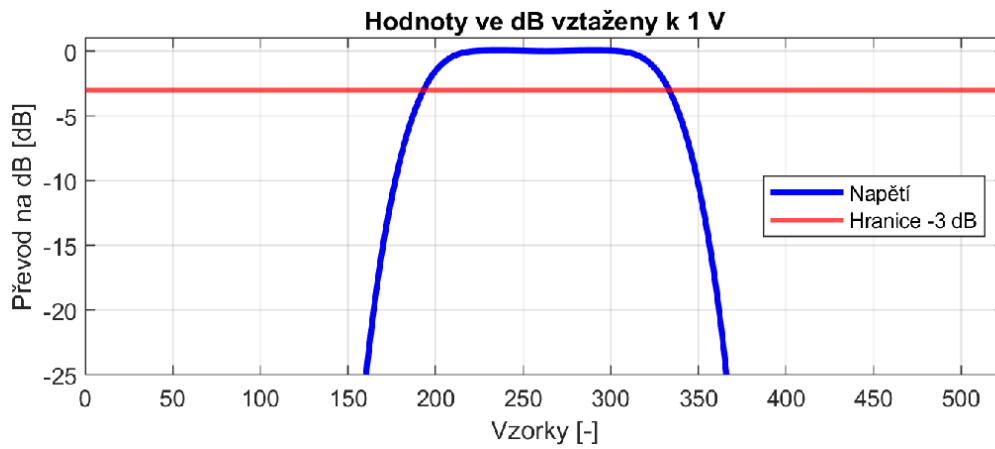


...

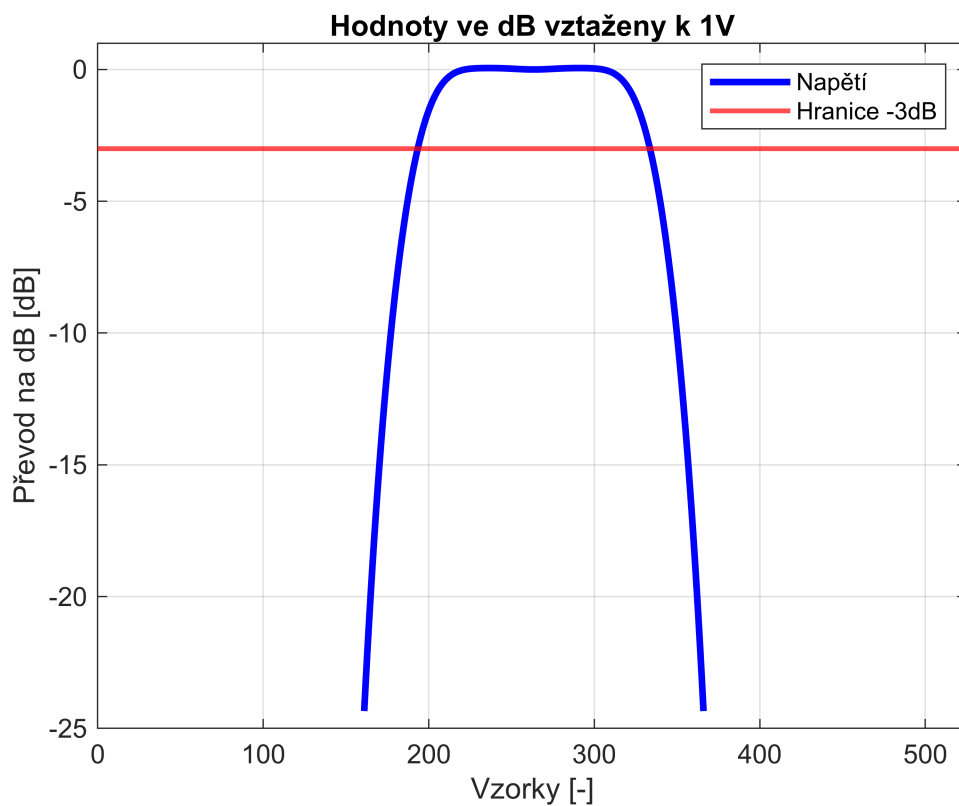
Pozn.: Všimněte si úseku v rozmezí vzorků 0 až 150 (380 až 525). V hodnotách ve Voltech nelze rozeznat detail oproti hodnotám v decibelech.

2)

- Vytvořte v novém obrázku detail na rozmezí hodnot od -25 do 1 dB (tj. detail druhého subplotu).
- Ponechte původní nastavení (šířku čáry, barvy, limit osy x).
- Do grafu přidejte konstantní funkci $y(x) = -3$ dB. Nejjednodušší způsob je využití funkce `yline`.
- Popište osy, nadpis a přidejte i legendu.

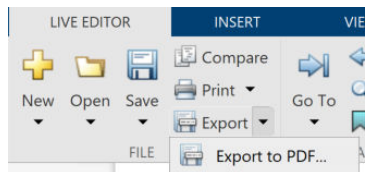


```
figure(4)
parsed_data = data_db;
parsed_data(data_db<-25|data_db>1) = NaN;
plot(parsed_data, LineWidth=2.5, Color="b")
xlim([0 525])
ylim([-25 1])
yline(-3, '-r', 'LineWidth', 1.8)
title("Hodnoty ve dB vztaženy k 1V")
ylabel("Převod na dB [dB]")
xlabel("Vzorky [-]")
legend("Napětí", "Hranice -3dB")
grid on
hold off
```



3 Odevzdání

Odevzdávejte tento soubor *mlx*, Vámi vytvořené funkce (v tomto cvičení: *kos_prumer*) a vytvořený PDF. PDF soubor vytvoříte tak, že nejprve necháte vygenerovat všechny výstupy ve skriptu (vykreslení obrázků, výpis pomocí `disp`, atd.). Nejsnáze přes tlačítko **Run** (ve skriptu samotném také ponechte výstupy). Poté v liště **"LIVE EDITOR"** v sekci **"FILE"** lze rozkliknout nabídku **"Export"**, kde zvolíte **"Export to PDF..."**.



Vše zabalte do ZIP a odešlete na MOODLE v sekci "Odevzdání protokolů".