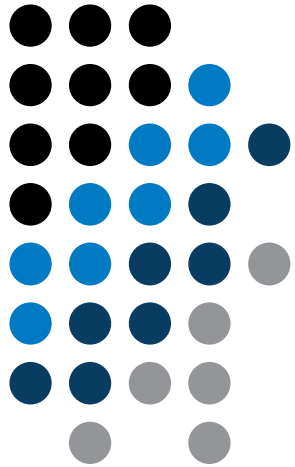
A0B17MTB – Matlab

Část #2

Miloslav Čapek
miloslav.capek@fel.cvut.cz

Filip Kozák, Viktor Adler

Katedra elektromagnetického pole
B2-626, Dejvice

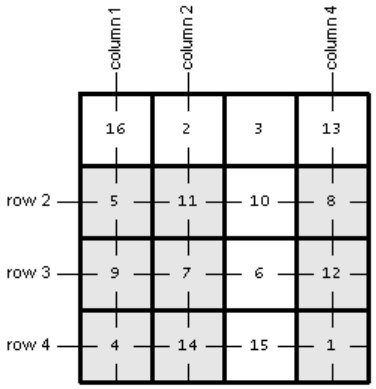


Naučíte se …

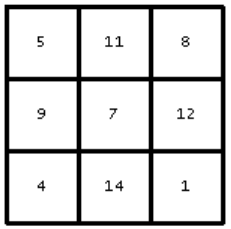
Komplexní čísla

Tvorba matic

Operace s maticemi



A



A([2 3 4], [1 2 4])

Komplexní čísla

- více možných zápisů v Matlabu
  - chceme se ale vyhnout nedorozumění
  - rychlost

```
>> C1 = 1 + 1j
>> C2 = 1 + 5i % preferováno
>> C3 = 1 + i
>> C4 = 1 + j5
```

```
>> C5 = sqrt(-1)
```

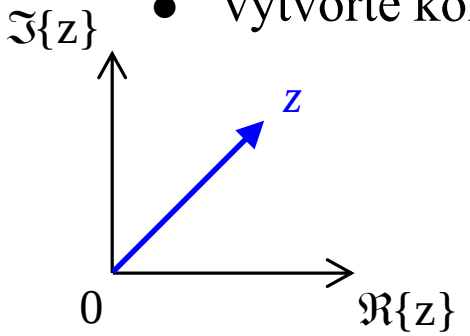
- často využívané funkce

`real, imag`	reálná a imagirární část komplexního čísla
`conj`	komplexně sdružené číslo
`abs`	absolutní hodnota komplexního čísla
`angle`	úhel v komplexní rovince (v [rad])
`complex`	vytvoří komplexní číslo z reál. a imag. části
`isreal`	je číslo reálné (více později)
`i, j`	komplexní jednotka
`cplxpair`	seřadí komplexní čísla do sdružených dvojic

Komplexní čísla

300 s ↑

- vytvořte komplexní číslo a číslo k němu komplexně sdružené



$$z = 1 + 1\mathrm{j}$$

$$s = z^*$$

- převádějte mezi čísly ve složkovém a polárním tvaru (najděte $|z|, \varphi$)

$$z = \Re\{z\} + \mathrm{j}\,\Im\{z\} = a + \mathrm{j}b$$

$$z = |z|\mathrm{e}^{\mathrm{j}\varphi},\ |z| = \sqrt{a^2 + b^2}$$

$$z = |z|\left(\cos(\varphi) + \mathrm{j}\sin(\varphi)\right)$$

- ověřte si Moivreovu větu

$$z^n = \left(|z|\mathrm{e}^{\mathrm{j}\varphi}\right)^n$$

$$z^n = |z|^n\left(\cos(n\varphi) + \mathrm{j}\sin(n\varphi)\right)$$

$$Z = |z| = \sqrt{2} \approx 1.4142$$

$$\varphi = \arctan\left(\frac{\Im\{z\}}{\Re\{z\}}\right) = \arctan\left(\frac{1}{1}\right) \approx 0.7854\,\mathrm{rad}$$

Komplexní čísla

300 s ↑

- najděte velikost komplexního vektoru (bez indexace)
  - spočítejte pomocí `abs`, `sqrt`

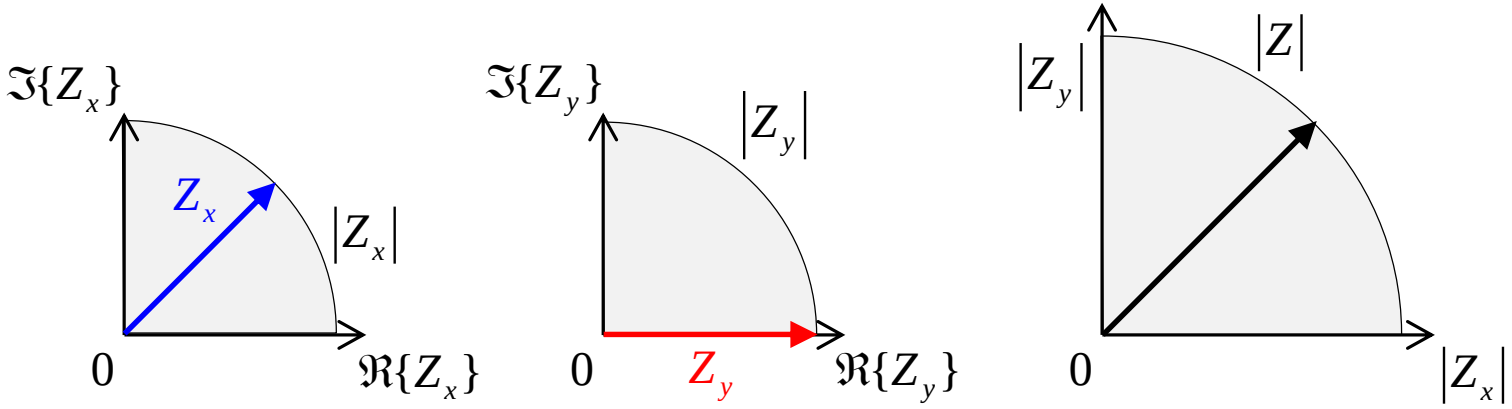
$$\mathbf{Z} = \begin{pmatrix} 1+1\mathrm{j} & \sqrt{2} \end{pmatrix}$$

$$\|\mathbf{Z}\| = ?, \quad \mathbf{Z} \in \mathbb{C}^2$$

(1) $|Z_x|, \ |Z_y|$

(2) $$|\mathbf{Z}| = \sqrt{|Z_x|^2 + |Z_y|^2} = \sqrt{Z_x Z_x^* + Z_y Z_y^*}$$
$$= \sqrt{\mathbf{Z} \cdot \mathbf{Z}^*} = \sqrt{|\mathbf{Z}|^2}$$

  - nebo využijte následujících funkcí:
    - `norm`
    - `dot` (z *dot product*)
    - `hypot` (z *hypotenuse*)



Transpozice a konjugace matice

- pozor na situace, kdy je matice komplexní, $\mathbf{A} \in \mathbb{C}^{M \times N}$
- rozlišujeme následující operace

transpozice	$\mathbf{A}^{\mathrm{T}} = \left[A_{ij} \right]^{\mathrm{T}} = \left[A_{ji} \right]$	`transpose(A) % <- nepoužívat`	`A.'`
transpozice + konjugace	$\mathbf{A}^{\mathrm{H}} = \mathbf{A}_{ij}^{\mathrm{H}} = \left[\mathbf{A}^{*} \right]^{\mathrm{T}}$	`conj(A) % <- nepoužívat`	`A'`

```
>> A = magic(2) + 1j*magic(2)'

A =

   1.0000 + 1.0000i   3.0000 + 4.0000i
   4.0000 + 3.0000i   2.0000 + 2.0000i
```

```
>> A.'

ans =

   1.0000 + 1.0000i   4.0000 + 3.0000i
   3.0000 + 4.0000i   2.0000 + 2.0000i
```

```
>> A'

ans =

   1.0000 - 1.0000i   4.0000 - 3.0000i
   3.0000 - 4.0000i   2.0000 - 2.0000i
```

Zadávání matic – „ : “

- rozsáhlé vektory a matice s pravidelně rostoucí hodnotou prvků lze zadat pomocí dvojtečkové notace
  - `a` je nejmenší prvek („od“) , `incr` je přírůstek, `b` je největší prvek („do“)

```
>> A = a:incr:b
```

```
>> A = 1:4:17

A =

     1     5     9    13    17
```

- `b` nemusí být prvek požadované řady
  - potom je maximální prvek $N \cdot incr$ takový, že:

$$|a + N \cdot incr| < |b|$$

```
>> A = 1:4:7

A =

     1     5
```

- je-li `incr` vynecháno, je generován vektor / matice s přírůstkem 1

```
>> A = a:b
```

```
>> A = 0:10

A =

     0     1     2     3     4     5     6     7     8     9    10
```

Zadávání matic

300 s ↑

- pomocí operace „ : “ vytvořte
  - vektory

$$\mathbf{u} = \begin{pmatrix} 1 & 3 & \dots & 99 \end{pmatrix}$$

$$\mathbf{v} = \begin{pmatrix} 25 & 20 & \dots & -5 \end{pmatrix}^{\mathrm{T}}$$

  - matici
    - pozor, třetí sloupec nelze zapsat pomocí „ : “ jednoduše

$$\mathbf{T} = \begin{pmatrix} -4 & 1 & \frac{\pi}{2} \\ -5 & 2 & \frac{\pi}{4} \\ -6 & 3 & \frac{\pi}{6} \end{pmatrix}$$

ale lze pomocí „ : “ a „ . “ (viz dále)

Zadávání matic – `linspace`, `logspace`

- dvojtečkou definujeme vektor s pevným krokem
- máme-li ale pevný počet prvků vektoru, používáme `linspace`:

```
>> A = linspace(0,2,5)

A =

         0    0.5000    1.0000    1.5000    2.0000
```

```
>> A = linspace(a, b, N);
```

  - vynecháme-li parametr `N`, je generováno vždy 100 prvků:

```
>> A = linspace(a, b);
```

- funkce `logspace` pracuje podobně, ale v logaritmické míře

```
>> A = logspace(a, b, N);
```

Zadávání matic

200 s ↑

- vytvořte vektor od -1.15 do 75.4 se 100 lineárně rozloženými prvky
- vytvořte vektor od 100 do -100 s 201 lineárně rozloženými prvky
- vytvořte vektor od 100 do -100 s krokem -10
  - zkuste poté varianty s pomocí `linspace`, i s pomocí „`:`“

Zadávání matic – pomocí funkcí

- velice často potřebujeme speciální typ matice o daných rozměrech
  - pro tyto účely je v Matlabu řada připravených funkcí
- příklad: matice nul
  - budeme ji velice často využívat

```
zeros(m)                % matice B je velká m×m
zeros(m, n)             % matice B je velká m×n
zeros(m, n, p,...)      % matice B je velká m×n×p×...
zeros([m n])            % matice B je velká m×n

B = zeros(m, 'single') % matice B je velká m×m, typu single)

% a další varianty (viz nápověda)
```

Zadávání matic – pomocí funkcí

- analogicky jako `zeros` lze využít i následující užitečné funkce:

`ones`	matice jedniček
`eye`	jednotková matice
`NaN, Inf`	matice hodnot NaN, matice hodnot Inf
`magic`	jedna z matic vhodných na pokusy s Matlabem, povšimněte si jejích zajímavých vlastností
`rand, randn, randi`	matice náhodných čísel s uniformním nebo normálním rozložením, matice náhodně generovaných celých čísel
`randperm`	vytvoří permutační vektor, prvky jsou umístěny náhodně (výsledek je vždy jen vektor!)
`diag`	vytvoří z vektoru diagonální matici (a opačně z matice vyjme zvolenou diagonálu)
`blkdiag`	vytvoří z jednotlivých matic blokovou matici podél hlavní diagonály
`cat`	složí dvě matice do jedné (je nutné zadat dimenzi, ve které jsou matice seskládány)
`true, false`	vytvoří matici logických jedniček nebo logických nul
`pascal, hankel`	Pascalova matice, Hankelova matice

- pro další funkce viz Matlab → Mathematics → Elementary Math → Constants and Test Matrices

Zadávání matic – pomocí funkcí

360 s ↑

- vytvořte následující matice
  - využijte pro tyto účely funkce Matlabu
  - začněte těmi maticemi, které zvládnete snadno

$$\mathbf{M}_1 = \begin{pmatrix} \mathrm{NaN} & \mathrm{NaN} \\ \mathrm{NaN} & \mathrm{NaN} \end{pmatrix}$$

$$\mathbf{M}_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{M}_3 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -5 \end{pmatrix}$$

$$\mathbf{M}_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Zadávání matic – pomocí funkcí

360 s ↑

- zkuste vytvořit pole typu `double` o 3 dimenzích, které je ale prázdné

- naleznete ještě jinou možnost?

  - `empty` je skrytá (ale veřejná) metoda všech neabstraktních tříd Matlabu

Zadávání matic

- velice často máme několik možností, jak potřebnou matici vytvořit
  - v Matlabu lze využívat výstup jedné funkce přímo jako vstup do další:
- zvažte
  - přehlednost
  - jednoduchost
  - rychlost
  - zvyklosti

```
>> plot(diag(randn(10, 1), 1))
```

- např. pásová matice s 1 na diagonále a 2, 3 na vedlejších diagonálách

```
>> N = 10;
>> diag(ones(N,1)) + diag(2*ones(N-1,1),1) + diag(3*ones(N-1,1),-1)
```

  - lze vyřešit i pomocí cyklu (viz dále), který může být i rychlejší…
  - napadne Vás i jiná varianta?

Práce s řídkými maticemi

- Matlab podporuje i práci s řídkými maticemi
  - řídké matice obsahují velký podíl nul a vyplatí se je tedy ukládat jiným – úspornějším – způsobem

- vytvoření řídké matice `S` z matice `A`:

```
S = sparse(A),
```

- převod z matice typu sparse na hustou matici:

```
B = full(S),
```

  - v případě potřeby viz nápověda pro další funkce

Operace nad maticí #1

- krom transpozice (viz dříve `transpose` příp. `'`) a diagonály matice (`diag`) se Vám může hodit:

```
P =

    0.3404    0.2551    0.9593    0.2575
    0.5853    0.5060    0.5472    0.8407
    0.2238    0.6991    0.1386    0.2543
    0.7513    0.8909    0.1493    0.8143
```

- horní trojúhelníková matice

```
>> U = triu(P),
```

```
>> U = triu(P)

U =

    0.3404    0.2551    0.9593    0.2575
         0    0.5060    0.5472    0.8407
         0         0    0.1386    0.2543
         0         0         0    0.8143
```

- dolní trojúhelníková matice

```
>> L = tril(P),
```

```
>> L = tril(P)

L =

    0.3404         0         0         0
    0.5853    0.5060         0         0
    0.2238    0.6991    0.1386         0
    0.7513    0.8909    0.1493    0.8143
```

- matici lze modifikovat i podle vedlejších diagonál

```
>> L = triu(P, -1),
```

```
>> U2 = triu(P,-1)

U2 =

    0.3404    0.2551    0.9593    0.2575
    0.5853    0.5060    0.5472    0.8407
         0    0.6991    0.1386    0.2543
         0         0    0.1493    0.8143
```

Operace nad maticí #2

- pro kopírování (části) matice využíváme funkci `repmat`

```
>> B = repmat(A, m, n),
```

  - např.

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \end{pmatrix}$$

```
>> B = repmat(A, 1, 2),
>> B = repmat(A, [2 1]),
```

$$\begin{matrix} A_{11} & A_{12} & A_{13} & A_{11} & A_{12} & A_{13} \end{matrix}$$

$$\begin{matrix} A_{11} & A_{12} & A_{13} \\ A_{11} & A_{12} & A_{13} \end{matrix}$$

- pamatujte si, že `repmat` je velice rychlá funkce
  - srovnání rychlosti tvorby matice plné nul o velikosti 1e4 × 1e4:

```
>> X = zeros(1e4, 1e4);      % computed in 0.18s
>> Y = repmat(0, 1e4, 1e4); % computed in 0.0004s
```

  - zvažte, jak alokovat nové matice …

Operace nad maticí #3

- pro přeskládání matice využíváme funkci `reshape`

```
>> B = reshape(A, m, n),
```

- např.

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

```
>> B = reshape(A, [4 1]),
>> B = reshape(A, 1, 4),
```

$$\begin{matrix} A_{11} \\ A_{21} \\ A_{12} \\ A_{22} \end{matrix}$$

$$A_{11} \quad A_{21} \quad A_{12} \quad A_{22}$$

Operace nad maticí #4

- následující funkce využijeme, potřebujeme-li prohodit pořadí
  - sloupců: `fliplr`

```
>> B = fliplr(A),
```

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} A_{13} & A_{12} & A_{11} \\ A_{23} & A_{22} & A_{21} \end{pmatrix}$$

  - řádek: `fliud`

```
>> B = flipud(A),
```

$$\mathbf{A} = \begin{pmatrix} A_{21} & A_{22} & A_{23} \\ A_{11} & A_{12} & A_{13} \end{pmatrix}$$

  - libovolné dimenze: `flipdim`

```
>> B = fliplr(A, 1),
>> B = fliplr(A, 2),
```

- stejný výsledek získáme s využitím indexace (viz dále)

Operace nad maticí #5

- krom prohození pořadí máme k dispozici i cyklickou záměnu
  - lze ji provést ve zvolené dimenzi (řádky / sloupce)
  - lze ji provést oběma směry (tam / zpět)

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

```
>> B = circshift(A, -2),
```

$$\mathbf{A} = \begin{pmatrix} A_{31} & A_{32} & A_{33} \\ A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{pmatrix}$$

```
>> B = circshift(A, [-2 1]),
```

$$\mathbf{A} = \begin{pmatrix} A_{33} & A_{31} & A_{32} \\ A_{13} & A_{11} & A_{12} \\ A_{23} & A_{21} & A_{22} \end{pmatrix}$$

- promyslete si rozdíl mezi `flipdim` a `circshift`

Operace nad maticí #1

450 s ↑

- matici $\mathbf{A} = \begin{pmatrix} 1 & \pi \\ \mathrm{e} & -\mathrm{i} \end{pmatrix}$ upravte do podoby matic $\mathbf{A}_1$ až $\mathbf{A}_4$
  - užijte funkcí `repmat`, `reshape`, `triu`, `tril`, příp. `conj`

$$\mathbf{A}_1 = \begin{pmatrix} 1 & \pi & 1 & \pi & 1 & \pi \\ \mathrm{e} & -\mathrm{i} & \mathrm{e} & -\mathrm{i} & \mathrm{e} & -\mathrm{i} \end{pmatrix}$$

$$\mathbf{A}_2 = \begin{pmatrix} 1 & \pi & \mathrm{e} & -\mathrm{i} \end{pmatrix}$$

$$\mathbf{A}_3 = \begin{pmatrix} 1 & \pi \\ \mathrm{e} & +\mathrm{i} \\ 1 & \pi \\ \mathrm{e} & +\mathrm{i} \\ 1 & \pi \\ \mathrm{e} & +\mathrm{i} \end{pmatrix}$$

$$\mathbf{A}_4 = \begin{pmatrix} 1 & \pi & 0 & 0 & 0 & 0 \\ \mathrm{e} & -\mathrm{i} & \mathrm{e} & 0 & 0 & 0 \\ 0 & \pi & 1 & \pi & 0 & 0 \\ 0 & 0 & \mathrm{e} & -\mathrm{i} & \mathrm{e} & 0 \\ 0 & 0 & 0 & \pi & 1 & \pi \\ 0 & 0 & 0 & 0 & \mathrm{e} & -\mathrm{i} \end{pmatrix}$$

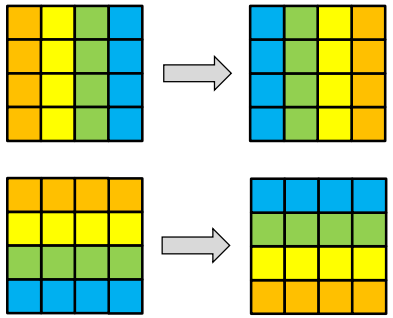
Operace nad maticí #2

450 s ↑

- vytvořte následující matici (použijte pokročilé techniky zadávání)

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 1 & 2 & 3 \\ 0 & 2 & 4 & 0 & 2 & 4 \\ 0 & 0 & 5 & 0 & 0 & 5 \end{pmatrix}$$

  - uložte ji do souboru se jménem `matice.mat`
  - z matice **A** vytvořte matici **B** pomocí záměny sloupců
  - z matice **B** vytvořte matici **C** pomocí záměny řádek
  - matice **B** a **C** přidejte do souboru `matice.mat`



Operace nad maticí #3

150 s ↑

- porovnejte a interpretujte následující příkazy:

```
>> x = (1:5)';                  % zadání vektoru

>> X = repmat(x, [1 10]),       % 1. možnost
>> X = x(:, ones(10, 1)),       % 2. možnost
```

```
>> x = (1:5)'

x =

     1
     2
     3
     4
     5
```

- `repmat` je mocná, ale ne vždy nejrychlejší funkce

```
>> X = repmat(x,[1 10])

X =

     1     1     1     1     1     1     1     1     1     1
     2     2     2     2     2     2     2     2     2     2
     3     3     3     3     3     3     3     3     3     3
     4     4     4     4     4     4     4     4     4     4
     5     5     5     5     5     5     5     5     5     5

>> X = x(:,ones(10,1))

X =

     1     1     1     1     1     1     1     1     1     1
     2     2     2     2     2     2     2     2     2     2
     3     3     3     3     3     3     3     3     3     3
     4     4     4     4     4     4     4     4     4     4
     5     5     5     5     5     5     5     5     5     5
```

Operace nad vektory a maticemi

- pamatujte, že maticové násobení není komutativní, tzn. $\mathbf{AB} \neq \mathbf{BA}$
- pamatujte, že násobení vektor × vektor produkuje

$$\mathbf{v}_{M,1}\mathbf{u}_{1,N} = \mathbf{w}_{M,N} \qquad \mathbf{v}_{1,M}\mathbf{u}_{M,1} = w_{1,1}$$

$$\begin{array}{c|cc} & u_{11} & u_{12} \\ \hline v_{11} & w_{11} & w_{12} \\ v_{21} & w_{21} & w_{22} \\ v_{31} & w_{31} & w_{32} \end{array} \qquad \begin{array}{ccc|c} & & & u_{11} \\ & & & u_{21} \\ & & & u_{31} \\ \hline v_{11} & v_{12} & v_{13} & w_{11} \end{array}$$

- tj. … pozor na rozměry matic v Matlabu!

Násobení vektorů po prvcích

- v Matlabu je možné násobit stejně velká pole tzv. prvek po prvku
  - výsledkem operace je opět pole
  - rozměry všech polí jsou stejná, příklad pro vektory 1×3:

$$\mathbf{a} = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$$

```
>> a*b
```

$$a_1 \quad a_2 \quad a_3 \,, \; b_1 \quad b_2 \quad b_3 \rightarrow$$

CHYBA

(Inner matrix dimensions must agree.)

```
>> a.*b
```

$$a_1 \quad a_2 \quad a_3 \,, \; b_1 \quad b_2 \quad b_3 \rightarrow a_1 b_1 \quad a_2 b_2 \quad a_3 b_3 = \left[a_i b_i \right]$$

Násobení matic po prvcích

- chceme-li vynásobit prvky dvou stejně velkých matic „prvek po prvku“, využijeme tzv. tečkovou notaci
  - tj. rozlišujeme násobení

```
>> A*B
```

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \rightarrow \begin{bmatrix} A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22} \\ A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22} \end{bmatrix}$$

```
>> A.*B
```

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \rightarrow \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} \\ A_{21}B_{21} & A_{22}B_{22} \end{bmatrix}$$

  - jde o tzv. *Hadamard product* / *element-wise product* / *Schur product*: $\mathbf{A} \circ \mathbf{B}$

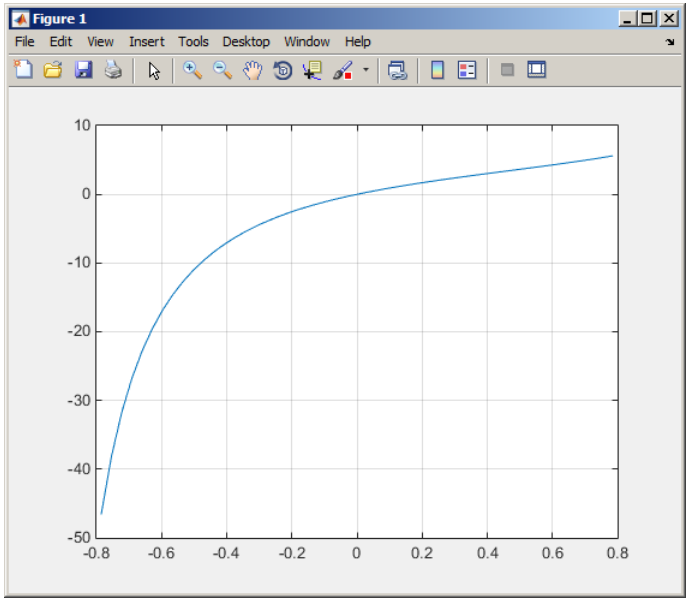
Operace po prvcích #1

- v Matlabu lze pracovat po prvcích i s vektory, skvěle se dá kombinovat s vektorovými funkcemi
- velice často lze eliminovat 1 nebo dokonce 2 cykly!!!
- tyto operace jsou mimořádně rychlé
  → otevírají možnosti pro tzv. vektorizaci (viz dále)

- např.: $f(x) = \frac{10}{(x+1)} \tan(x),$

$$x \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$$

```
>> x = -pi/4:pi/100:pi/4;
>> fx = 10./(1+x).*tan(x);
>> plot(x, fx);
>> grid on;
```



Operace po prvcích #1

300 s ↑

- vyhodnoťte funkce $f_1(x) = \sin(x)$ pro proměnnou $x \in [0, 2\pi]$

  $f_2(x) = \cos^2(x)$

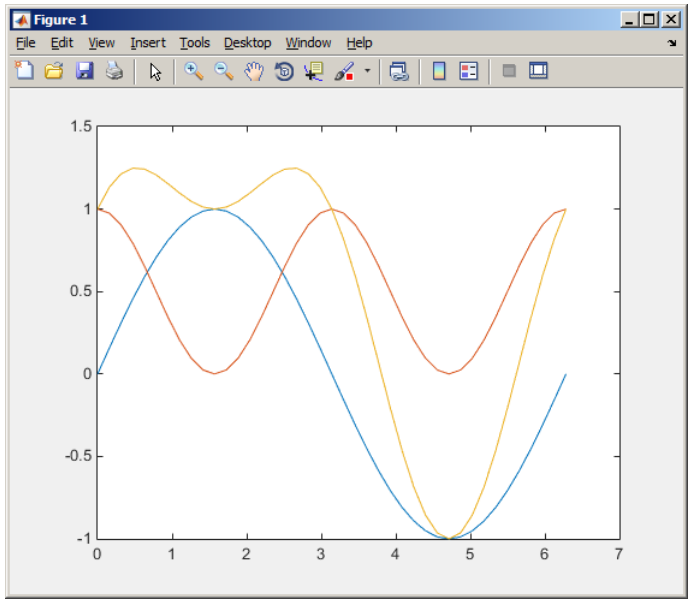
  $f_3(x) = f_1(x) + f_2(x)$

  - funkci vyhodnoťte v bodech uvedeného intervalu, které jsou ekvidistantně vzdáleny o $\Delta x \in \pi / 20$

  - vaše řešení můžete ověřit takto:

```
>> plot(x,f1, x,f2, x,f3),
```

  - Matlab umožňuje i symbolické řešení (probereme později)



Operace po prvcích #2

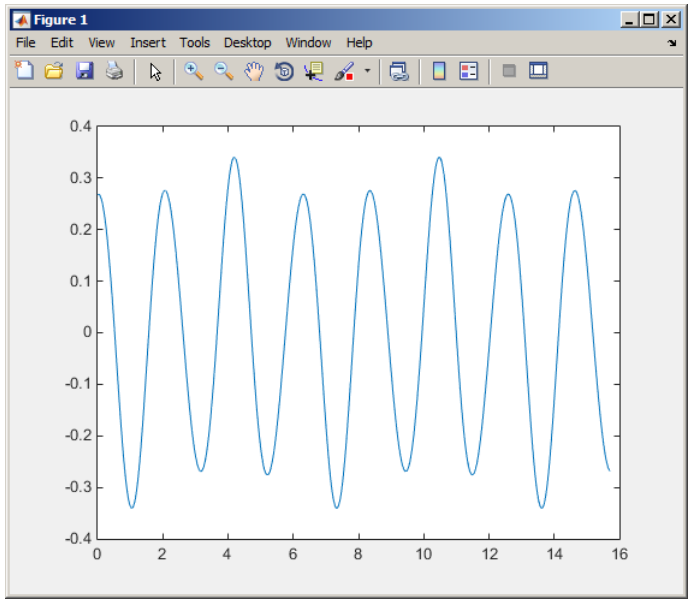
240 s ↑

- graficky znázorněte průběh funkce v intervalu $x \in [0, 5\pi]$

$$f_4(x) = \frac{-\cos(3x)}{\cos(x)\sin\left(x - \frac{\pi}{5}\right) - \pi}$$

  - výsledek vykreslete pomocí příkazu

```
>> plot(x, f4);
```



- vysvětlete rozdíl v násobení

```
>> A*B,
```

```
>> A.*B,
```

```
>> A'.*B,
```

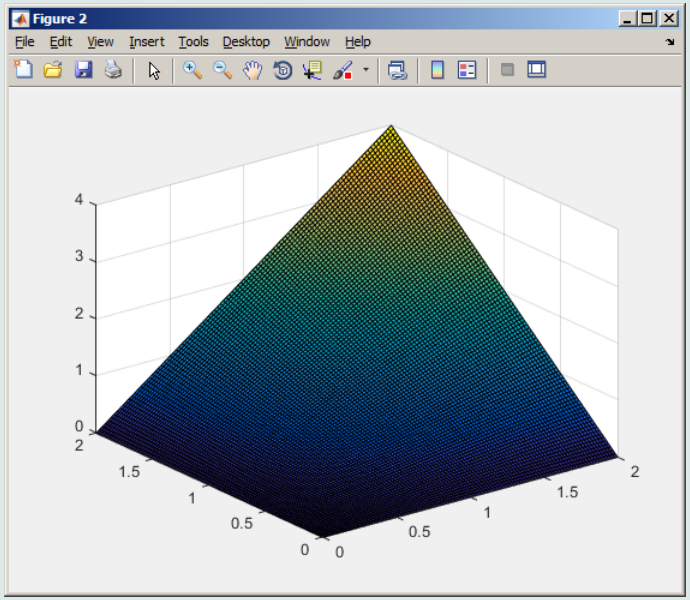
pro stejně velké čtvercové matice

Operace po prvcích #3

360 s ↑

- funkci $f(x,y) = xy, \quad x,y \in [0,\ 2]$ a vyhodnoťte ve 101 bodech
- její vyhodnocení lze provést buďto pomocí vektorů, vektorizace (nad maticemi prvek po prvku), nebo dvěma cykly
  - výsledek vykreslete pomocí příkazu `surf(x,y,f)`
  - máte-li hotovo, zkuste si funkci $f(x,y) = x^{0.5}y^2$ ve stejných mezích

nebo také (bude se nám hodit později):



Operace nad maticemi

- tvorba blokové diagonální matice: `blkdiag`

A_{11}

$\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

```
>> A = 1; B = [2 3; -4 -5];
>> C = blkdiag(B, A);
```

$\begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & 0 \\ 0 & 0 & A_{11} \end{bmatrix}$

- složení dvou stejně velkých matic: `cat`

$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

$\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

```
C = cat(2, A, B)
C = cat(1, A, B)
C = cat(3, A, B)
```

$\begin{bmatrix} A_{11} & A_{12} & B_{11} & B_{12} \\ A_{21} & A_{22} & B_{21} & B_{22} \end{bmatrix}$

$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

$\begin{bmatrix} A_{11} & A_{12} & \\ A_{21} & A_{22} & B_{12} \\ & B_{21} & B_{22} \end{bmatrix}$

Rozměry matic a dalších struktur

- v Matlabu často požadujeme znalost velikost matic a polí
- funkce `size` vrací vektor, který udává velikost matice / pole

```
>> A = randn(3,5);
>> d = size(A) % d = [3 5]
```

- funkce length vrací největší rozměr daného pole
  - tj. `length(A) = max(size(A))`

```
>> A = randn(3,5,8);
>> e = length(A) % e = 8
```

- funkce `ndims` vrací počet dimenzí matice / pole
  - tj. `ndims(A) = length(size(A))`

```
>> m = ndims(A) % m = 3
```

- funkce `numel` vrací počet prvků v matici / poli

```
>> n = numel(A) % n = 120
```

Rozměry matic a dalších struktur

250 s ↑

- vytvořte si libovolné 3D pole
  - např. využijte následujících příkazů:

```
>> A = rand(2+randi(10), 3+randi(5));
>> A(:,:,2) = flipud(fliplr(A)),
```

- pro všechny následující úkoly zkuste nejprve určit správný výsledek
  - nalezněte rozměry pole
  - nalezněte počet prvků pole
  - nalezněte počtu prvků podél „nejdelší“ dimenze
  - nalezněte počet dimenzí pole

Datové typy v Matlabu

- v Matlabu jde o téma, které lze odložit na později …

```
>> whos
  Name                    Size              Bytes  Class     Attributes

  D                      50x1                400  double
  DD                      1x20               160  double
  DWx                    20x20              3200  double
  DWy                    20x20              3200  double
  Eps                     1x1                  8  double
  KA                     20x20              3200  double
  L                       1x1                  8  double
  Lcheck                 20x20              3200  double
  N                       1x1                  8  double
  Nth                     1x1                  8  double
  OK                      1x1                  1  logical
  PR                     20x20              3200  double
  Pr                      1x1                  8  double
  SOL                    20x20               400  logical
  Tcross                  1x1                  4  single
  lam                     1x1                  8  double
  omWA                   20x20              3200  double
  psi                     1x1                  8  double
  type_of_connection      1x6                 12  char
```

```
>> class(type_of_connection)

ans =

char
```

Bonus: funkce `gallery`

- funkce umožňuje vytvořit celou řadu matic, které můžeme využít pro testování Matlab kódu
- většina matic je určena pro speciální využití
  - pro znalého uživatele Matlabu však funkce gallery představuje značnou časovou úsporu
- viz `help gallery / doc gallery`
- zkuste si např.:

```
>> gallery('pei', 5, 4)
>> gallery('leslie', 10)
>> gallery('clement', 8)
```

Funkce `why`

- je nutné vyzkoušet! :)
  - zkuste pro ní najít nápovědu
  - zkuste přijít na to, kolik variant odpovědí existuje

Probrané funkce

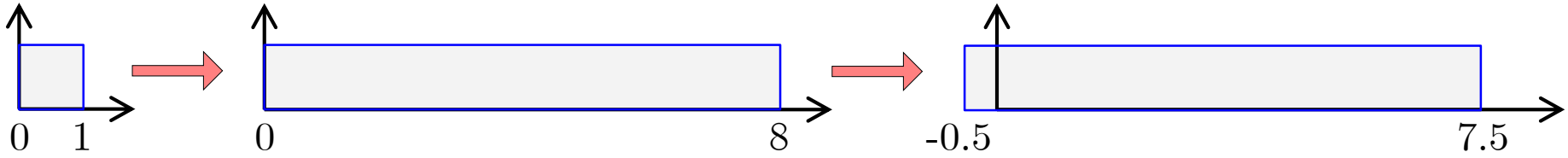
`real, imag, cong, angle, complex`	práce s komplexními čísly	
`norm, cumsum`	norma (matice / vektoru), kumulativní suma	●
`hypot`	vypočte přeponu (i pro komplexní čísla)	
`linspace, logspace`	generace vektoru – ekvidistantně v lineární / logaritmické míře	
`zeros, ones, eye, NaN, magic`	alokace matic	
`rand, randn, randi`	matice náhodných čísel s uniformním nebo normálním rozložením, matice náhodně generovaných celých čísel	
`randperm`	vytvoří permutační vektor, prvky jsou umístěny náhodně	
`true, false`	alokace matic (logické proměnné)	
`pascal, hankel, gallery`	speciální matice	●
`blkdiag, cat`	tvorba blokové matice, složení matic podél stejně velké dimenze	
`diag, triu, tril,`	práce s diagonální maticí, horní a dolní trojúhelníková matice	●
`fliplr, flipud, circshift`	prohození prvků, rotace prvků matice	
`repmat, reshape`	operace s maticí (replikace, změna dimenzí)	●
`length, size, ndims, numel`	délka vektoru, rozměr matice, počet dimenzí a prvků	
`sparse, full`	práce s řídkými maticemi	
`grid on, grid off`	zapne / vypne grid na grafu	
`figure, surf`	otevře nové okno grafu, 3D graf surf	●

Cvičení #1

360 s ↑

- vytvořte matici **M** o rozměrech `size(M) = [3 4 2]`, která bude obsahovat náhodně generovaná čísla s rovnoměrným rozložením v intervalu [-0.5,7.5]

$$I(x) = (I_{\max} - I_{\min})\,\mathrm{rand}(\ldots) + I_{\min}$$



Cvičení #2

200 s ↑

- zamyslete se nad operací `a1^a2`, lze provést pro
  - `a1` – matice, `a2` – skalár
  - `a1` – matice, `a2` – matice
  - `a1` – matice, `a2` – vektor
  - `a1` – skalár, `a2` – skalár
  - `a1` – skalár, `a2` – matice
  - `a1`, `a2` – matice, `a1.^a2`

v případě potřeby si vytvořte vhodné matice `a1`, `a2` a na nich vyzkoušejte…

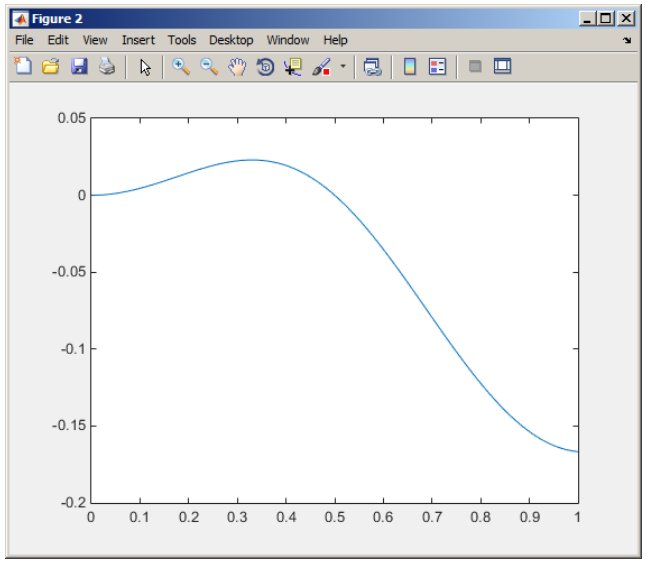
Cvičení #3

- opravte následující segment kódu tak, abyste získali hodnoty funkce $f(x)$ v intervalu [0, 1] pro 200 vzorků:

$$f(x) = \frac{x^2 \cos(\pi x)}{(x^3+1)(x+2)}$$

```
>> % chybný kód
>> x = linspace(0, 1);
>> clear all
>> g = x^3+1; H = x+2;
>> y = cos xpi; z = x.^2;
>> f = y*z/gh
```

  - přímým přístupem do vektoru zjistěte hodnotu funkce v bodě x = 1?
  - jaká je hodnota funkce v bodě x = 2?
  - pro kontrolu vykreslete průběh funkce $f(x)$



Cvičení #4

200 s ↑

- promyslete, kolika způsoby lze vypočítat přeponu ze dvou zadaných odvěsen
  - využijte různé operátory, případně různé funkce Matlabu, které naleznete
  - uvažujte také, že odvěsny mohou být komplexní čísla

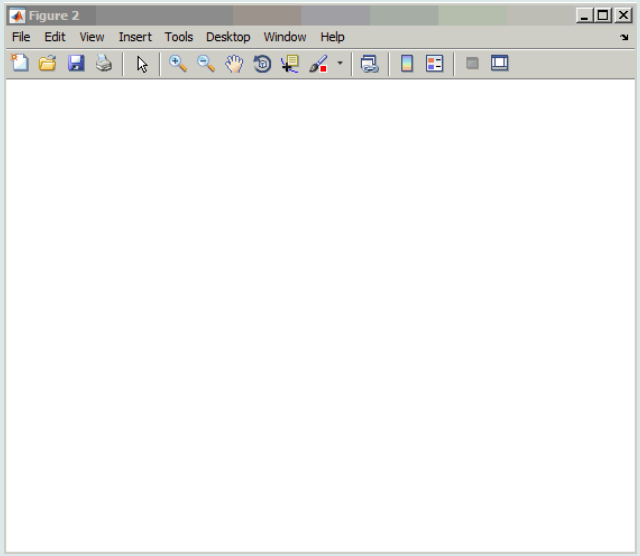
Cvičení #5

- Proton s nábojem $Q = 1.602 \cdot 10^{-19}\,\mathrm{C}$ a hmotností $m = 1.673 \cdot 10^{-31}\,\mathrm{kg}$ vletí počáteční rychlostí $v_0 = 1 \cdot 10^{7}\,\mathrm{m/s}$ ve směru osy z do homogenního magnetického pole o intenzitě $B = 0.1\,\mathrm{T}$ a zároveň do elektrického pole o intenzitě $E = 5 \cdot 10^{5}\,\mathrm{V/m}$ takovým způsobem, že se pohybuje po šroubovicové dráze.
  - rychlost letu podél osy z je $v = \frac{QE}{m} t + v_0$
  - kde t je čas, uletěná dráha podél osy z je $z = \frac{1}{2} \frac{QE}{m} t^2 + v_0 t$
  - poloměr šroubovice je $r = \frac{vm}{BQ}$
  - frekvence oběhu okolo šroubovice je $f = \frac{v}{2\pi r}$
  - souřadnice x a y letu se vypočte jako $x = r\cos(2\pi f t), \quad y = r\sin(2\pi f t)$

Cvičení #6

500 s ↑

- Vykreslete dráhu protonu v prostoru v časovém intervalu od 0ns do 1ns v 1001 bodech pomocí funkce `comet3(x,y,z)`




```
>> clear all; close all; clc;
```

```
>> % zde bude Váš kód
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
>> % ...
```

```
>> comet3(x, y, z)
```

Děkuji!

ver. 3.4 (23/02/2015)
Miloslav Čapek
miloslav.capek@fel.cvut.cz