



**TECHNICKÁ UNIVERZITA V LIBERCI**  
**Fakulta mechatroniky, informatiky  
a mezioborových studií**



# MTLB: přednáška 1

Programování a práce s proměnnými

**Jaroslav Čmejla**

# Předmět MTLB

- 6 přednášek
- 11 cvičení navazujících na látku předchozí přednášky
- 2 malé kontrolní práce na 5. a 10. cvičení + závěrečná zápočtová práce
- Cvičení, kontr./záp. práce → <https://elearning.tul.cz>
- Podmínky pro udělení zápočtu:
  - Povolena jedna neomluvená absence na cvičeních
  - Získání min. 700 bodů z 1000 z úkolů na cvičeních
  - Získání min. 10 bodů z 20 z kontrolních prací
  - Získání min.  $6+3x$  bodů z 10 v zápočtové práci, kde  $x$  je počet nepovolených a neomluvených absencí na cvičeních
  - Odevzdání individuální GUI aplikace (10. cvičení)
- Zápočtová písemná práce
  - bude vypsáný dostatečný počet termínů v zápočtovém týdnu a během zkouškového období
  - maximálně 2 pokusy
  - obsahuje výhradně látku probíranou na cvičeních

# Část I

## Program a programovací jazyk MATLAB

# Matlab - základní informace

- Skriptovací programovací jazyk
- Program Matlab: interpret příkazů, komerční produkt firmy MathWorks, Inc.
- Neprovádí se překlad do strojového kódu
- Nevýhody:
  - menší rychlost
  - menší obecnost využití
  - závislost na interpretu (Matlabu)
- Výhody:
  - Přenositelnost (UNIX, MAC, WINDOWS, LINUX,...)
  - Velmi snadné ladění (debugging, profiling)
  - Stabilita (automatická alokace proměnných)
- Instalace licence TAH:  
<https://liane.tul.cz/software/MATLAB>
- Alternativní programy: GNU Octave, Scilab, FreeMat

# Matlab - základní informace

## Spouštění:

- Původně konzolová aplikace, dnes GUI (grafické uživatelské rozhraní) - systém je ale stejný, tj. ovládáme standardní vstup a pozorujeme standardní výstup jádra
- Příkazová řádka »
  - Zadávají se příkazy, volají proměnné, funkce, skripty, ...
  - Více příkazů oddělujeme “;” (nevypisuje se výsledek) nebo “ ”  
“ ”  
“ ”
  - Našeptávání a historie pomocí tabulátoru a šipek ↑ a ↓
- Pracujeme v aktuálním pracovním adresáři `pwd`, `cd`

```
>> pwd  
ans =  
e:\tmp\b
```
- Cesty k souborům: `path`, `addpath`
- Ukončení příkazem `exit`, `quit`

# Matlab - základní informace

- Příkazy: výrazy, proměnné, funkce a skripty
- Komentář je oddělený znakem "%"

Nápověda:

- `>> help funkce % toto je komentář`
- Jak to funguje: `help` vypíše komentář pod hlavičkou souboru `funkce.m`
- Snadné vytváření nápovědy k vlastním funkcím
- Hypertextová nápověda

`>> doc funkce`

Editor a Live Editor:

- `>> edit funkce % otevře v textovém editoru soubor funkce.m`
- Pomocí Live Editoru si lze dělat elektronické poznámky během přednášky

# Odkazy

## WWW

- <https://elearning.tul.cz>
- <https://liane.tul.cz/software/MATLAB>

## Literatura

- Náповѣда а електронікѣ курзы Matlab (viz stránky LIANE)

# Část II

## Programování skriptů a funkcí



# Funkce a skripty v Matlabu

- ASCII soubory s koncovkou \*.m
- Lze editovat v libovolné editoru
- Vlastní editor prostředí Matlabu spustíme příkladem `edit`

# Skript

- ASCII soubor s koncovkou \*.m obsahující posloupnost příkazů
- Skript nemá vlastní datový segment “Workspace”, pracujeme v základním “Base”
- Globální proměnné: deklarace pomocí příkazu `global`

# Funkce

- ASCII soubory s koncovkou \*.m začínající klíčovým slovem `function`
- Vlastní datový segment “Workspace” (proměnné jsou lokální)
- Deklarace funkce, vstupních a výstupních proměnných  
`function [a,b,c]=mojefunkce(x,y)`  
*% tady je nápověda*
- Netřeba definovat datové typy, počet vstupů a výstupů (deklarací si vstupy a výstupy pouze pojmenováváme, nemusí existovat, může jich být víc)
- Počet vstupních a výstupních proměnných: proměnné `nargin` a `nargout`, pro vstup a výstup lze použít paměťové pole `varargin`, `varargout`
- Vložené funkce: dalším klíčovým slovem `function` definujeme lokální funkci, která ale není vidět navenek

# Debugging

- Breakpointy vkládáme v programovacím editoru, z příkazové řádky je to též možné (příkaz `dbstop`) avšak...
- Lze vkládat i podmíněné breakpointy
- Pomáháme si příkazy `echo`, `disp`, `keyboard`, `return`, `warning`, ...
- Během debuggingu můžeme dělat prakticky cokoliv: sledovat a měnit proměnné, vytvářet nové, volat příkazy ... zde je zásadní výhoda toho, že Matlab je interpret nikoliv překladač

# Cykly

## ■ Cyklus for

```
for i=ind % ind obsahuje hodnoty, které i nabývá  
          % často např 1:50  
% tělo cyklu  
end
```

Lze měnit řídící proměnnou v průběhu, avšak v dalším průběhu bude mít další hodnotu z pole ind

## ■ Cyklus while

```
while podminka % podminka je logická hodnota  
              % např. a>1  
% tělo cyklu  
end
```

# Podmínky

## Podmínka

```
if podminka
    % tělo podmínky
elseif podminka2
    % tělo druhé podmínky
elseif podminka3
    % tělo třetí podmínky
else
    % jinak
end
```

# Podmínka switch

## Podmínka switch

```
switch vyraz  
  case 1  
    % tělo příkazu  
  case {2, 3, 4}  
    % tělo příkazu  
  otherwise % jinak  
    % tělo příkazu  
end
```

Provádí se pouze pravdivé případy, není třeba ukončovat případy pomocí `break` jako je tomu např. v C++

# Část III

## Proměnné a operátory



# Čísla, matice, vektory, pole

- Každá proměnná v Matlabu je n-rozměrné pole nějakého typu
- Automatická alokace paměti a doalokovávání
- Základním typem je `double`, popř. `complex` `double` (přestože třeba zadáme celočíselnou hodnotu)
- Funguje i počítání s hodnotami `Inf`, `-Inf`, `NaN` podle standardních pravidel
- Speciální význam (hlavně kvůli operátorům)
  - 0-rozměrné pole - skalár
  - 1-rozměrné pole - vektor (řádkový nebo sloupcový)
  - 2-rozměrné pole - matice
- `whos` - detailní výpis proměnných, jejich typů atd. (též `Workspace`)

## Vytváření vektorů a matic pomocí [ ]

### ■ Příklad:

```
>> A=[1 2 3; 5 10,15.3; 7+5i 8 9]';
```

- “;” uvnitř hranatých závorek znamená další řádek
- “,” uvnitř hranatých závorek znamená další prvek
- “,” na konci příkazu znamená, že se nevypisuje výsledek
- $i$  je imaginární jednotka, ale může být i názvem pro proměnnou.  $5i$  je ale vždy komplexní číslo, protože název proměnné nemůže začínat číslem.
- Transpozice a komplexní sdružení (hermitovská transpozice): apostrof

```
>> A'
```

- Pouze transpozice: tečka + apostrof. Viz

```
>> A.'
```

# Indexování

- Indexujeme pomocí oblých závorek () za názvem proměnné
- Např.  $A(1,2)$  znamená prvek A v prvním řádku a druhém sloupci.
- Pokud prvek neexistuje, je to chyba. Neexistujícímu prvku ale lze přiřadit hodnotu: automatická doalokace.
- Operátor dvojtečka “:”

```
>> 1:5 % vektor 1 až 5
```

```
ans =
```

```
1 2 3 4 5
```

```
>> 1:2:10 % vektor 1 až 10 po 2
```

```
ans =
```

```
1 3 5 7 9
```

```
>> 0:0.2:1 % vektor 0 až 1 po 0.2
```

```
ans =
```

```
0 0.2000 0.4000 0.6000 0.8000 1.0000
```

# Indexování

- Použití: indexování celé obdélníkové části pole. Např.

```
>> A(1:3,4:5) % ve skutečnosti se jedná o složený výraz  
% A([1 2 3],[4 5])
```

- Samostatný operátor ":" uvnitř oblých závorek má význam "všechny prvky".

```
>> A(:,5) % všechny prvky A v pátém sloupci  
% čili pátý sloupec A
```

```
>> A(2,:) % všechny prvky A v druhém řádku
```

```
>> A(:,,:) % celá matice
```

```
>> A(:,:,3) % třetí matice v 3D poli A
```

# Indexování

- Speciální význam má použití jediného indexovacího parametru. Tím můžeme indexovat prvky libovolně rozměrného pole.

```
>> A(8) % prvek pole A, který je osmý v pořadí v paměti
```

```
>> A(:) % všechny prvky A řazené podle pořadí v paměti  
      % do sloupcového vektoru
```

- Další příklady:

```
>> A(end,:) % poslední řádek, klíčové slovo end
```

```
>> A(:,end-1) % předposlední sloupec
```

```
>> p=[1 2 7 2 2 3 3]; % indexy se mohou i opakovat  
      % a mít různé pořadí
```

```
>> A(p,end,1:4,:[1:6 9:16]) % složený výraz,  
      % vícerozměrné pole
```

# Operátory

- Standardní operátory  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\backslash$ ,  $($ ,  $)$ ,  $^$
- Závorky pro změnu priority ve výrazech jsou  $( )$ , nikoliv  $[ ]$ ,  $\{ \}$
- Většina operátorů má stejný význam jako v (lineární) algebře.
- Příklady

```
>> A+1 % ke každému prvku A přičte 1
```

```
>> A-B % pole musí mít stejné rozměry
```

```
>> 8*A % každý prvek A vynásobí osmi
```

```
>> A*B % Pozor! Jsou-li A a B matice či vektory, jedná se o  
% maticové násobení!!! Rozměry musí být pro takovou  
% operaci kompatibilní, jinak vzniká chyba
```

# Operátory

## Další příklady

```
>> 4^5.6 % 4 na 5.6
```

```
>> A^3 % stejné jako A*A*A, tedy může to znamenat maticovou  
      % operaci je-li A matice!
```

```
>> 2/3 % dělení
```

```
>> 2\3 % opačné dělení, tedy 3/2
```

```
>> A/B % A*inv(B) jsou-li A a B čtvercové matice stejných  
      % rozměrů a B je regulární  
      % není však obecně stejné jako inv(B)*A
```

```
>> A\B % inv(A)*B je-li A regulární
```

# Operátory

- Operátory fungující "po prvcích": +, -, .\*, ./, .\, .^

```
>> A.*B % A a B musí mít stejný rozměr až na výjimku od  
% verze jádra 9.0  
% každý prvek A je vynásoben příslušným prvkem B  
% platí pro libovolně rozměrná pole
```

```
>> A./B % dělení polí po prvcích
```

```
>> 1./A % převrácené hodnoty všech prvků A  
% výsledkem je pole stejných rozměrů jako A
```

```
>> A.^2 % druhá mocnina všech prvků A  
% jiné než A^2
```

- Výjimka od verze jádra 9.0 (R2016b): Rozměry matic musí být shodné nebo rovny 1.



# Operátory

- Vektorizované výrazy - výhoda jazyku Matlab! Např. pro všechny prvky polí A, B, C, D chceme spočítat výraz

$$\frac{a \cdot b}{(c + 1)^2} - d$$

```
>> A.*B./(C+1).^2 - D % pozor na prioritu operátorů
```

- Logické operátory: ==, ~=, >, <, <=, >=, &, |, ~ fungují též "po prvcích" a vrací logické 0 nebo 1. Např.

```
>> A>7 % prvky A větší než 7  
      % výsledkem je pole stejných rozměrů jako A
```

```
>> A>B % A a B musí mít stejný rozměr
```

```
>> ~A % nulové prvky A
```

# Práce s maticemi

## ■ Zjišťování rozměrů

```
>> size(A) % vrací vektor rozměrů  
ans =
```

```
4      6
```

```
>> size(A,2) % druhý rozměr A
```

```
>> length(A) % vrací délku A (největší rozměr)  
ans =
```

```
6
```

## ■ Generování speciálních matic: zeros, ones, rand, randn, eye

```
>> A=zeros(4,5,6) % 3D tenzor nul o rozměrech 4x5x6  
>> B=ones(size(A)) % B má stejné rozměry jako A
```

# Práce s maticemi

- Spojování matic:

```
>> A=[A,B;C] % rozměry musí "pasovat"
```

- Změna rozměrů: příkaz reshape

- Operátor přiřazování - "lze kombinovat s indexováním. Výraz na pravé straně musí být přiřaditelný výrazu na straně levé (rozměry, atp.)

```
>> A(1:2,4:5)=3; % všechny prvky na pravé straně budou 3
```

```
>> A(1:2,4:5)=[4 5; 6 7]; % pravá strana je matice 2x2,  
                        % levá také
```

```
>> A(:,4)=[]; % levá strana je prázdná matice, takto se  
              % odstraní z A čtvrtý sloupec
```

- Operace lineární algebry: det, inv, diag, triu, tril

# Standardní matematické funkce v Matlabu

- Standardní názvy: `sin`, `cos`, `tan`, `log`, `exp`, `abs`, `sign` ...
- Funkce fungují obecně v komplexním oboru
- Funkce se standardně aplikují “po prvcích”, výstup má stejný rozměr jako vstup
- Zaokrouhlování: `round`, `ceil`, `fix`, `floor`
- Maximální a minimální prvek a medián: `max`, `min`, `median`. V případě vektoru je jedno je-li sloupcový nebo řádkový, u matic fungují “po sloupcích”, dále viz `help`. Neslouží k hledání minim a maxim funkcí (Optimization Toolbox)!
- Třídění: `sort`

Tento materiál vznikl v rámci projektu ESF CZ.1.07/2.2.00/28.0050

**Modernizace didaktických metod a inovace výuky technických předmětů,**  
který je spolufinancován Evropským sociálním fondem a státním rozpočtem ČR.