# 11MAMY – Cvičení 1

## Opakování Matlabu

Lucie Kárná, Jan Přikryl, Jakub Ševčík, Martin Janda

ČVUT FD (Kárná, Přikryl), ZČU FEL (Přikryl, Ševčík, Janda)

17. března 2022

# Obsah cvičení

## Opakování Matlabu

- m-soubory
- Větvení
- Cykly

## Model epidemie

- Kermack-McKendrickův SIR model
- Model v Matlabu
- Grafický výstup

# m-soubory

Tímto pojmem označujeme textové soubory s příkazy Matlabu uložené s příponou `.m`

## Typy m-souborů

- **skripty** . . . sekvence příkazů
  - všechny proměnné globální
  - volají se jménem souboru

# m-soubory

Tímto pojmem označujeme textové soubory s příkazy Matlabu uložené s příponou `.m`

## Typy m-souborů

- **skripty** . . . sekvence příkazů
  - všechny proměnné globální
  - volají se jménem souboru
- **m-funkce** . . . funkce
  - všechny proměnné lokální
  - vstupní a výstupní parametry
  - volají se jménem funkce a parametry
  - jméno souboru udává jméno funkce

# Větvení programu – podmínka `if`

Tři možnosti:

- `if` podmínka, příkazy, `end`
- `if` podmínka, příkazy1, `else`, příkazy2, `end`
- `if` podmínka1, příkazy1, `elseif` podmínka2, příkazy2, `else`, příkazy3, `end`

# Větvení programu – podmínka `if`

Tři možnosti:

- `if` podmínka, příkazy, `end`
- `if` podmínka, příkazy1, `else`, příkazy2, `end`
- `if` podmínka1, příkazy1, `elseif` podmínka2, příkazy2, `else`, příkazy3, `end`

## Příklad 1

```
if a>0
  disp('a je kladne');
end
```

# Větvení programu – podmínka if

Tři možnosti:

- if podmínka, příkazy, end
- if podmínka, příkazy1, else, příkazy2, end
- if podmínka1, příkazy1, elseif podmínka2, příkazy2, else, příkazy3, end

## Příklad 1

```
if a>0
  disp('a je kladne');
end
```

## Příklad 2

```
if a==b
  disp('a rovno b');
else
  disp('a nerovno b');
end
```

# Větvení programu – podmínka if

Tři možnosti:

- if podmínka, příkazy, end
- if podmínka, příkazy1, else, příkazy2, end
- if podmínka1, příkazy1, elseif podmínka2, příkazy2, else, příkazy3, end

## Příklad 1

```
if a>0
  disp('a je kladne');
end
```

## Příklad 2

```
if a==b
  disp('a rovno b');
else
  disp('a nerovno b');
end
```

## Příklad 3

```
if a<b
  disp('a < b');
elseif a>b
  disp('a > b');
else
  disp('a rovno b');
end
```

# Opakované vykonávání kódu – cykly

**Q:** V jakých případech potřebujeme opakovat části kódu?

# Opakované vykonávání kódu – cykly

**Q:** V jakých případech potřebujeme opakovat části kódu?
**A:** Iterativní výpočet, práce s polem, ...

# Opakované vykonávání kódu – cykly

**Q:** V jakých případech potřebujeme opakovat části kódu?
**A:** Iterativní výpočet, práce s polem, ...

**Q:** Jaké dva základní typy cyklů v Matlabu známe a v čem se liší?

# Opakované vykonávání kódu – cykly

**Q:** V jakých případech potřebujeme opakovat části kódu?
**A:** Iterativní výpočet, práce s polem, ...

**Q:** Jaké dva základní typy cyklů v Matlabu známe a v čem se liší?
**A:** `for` cyklus – pevný počet opakování, `while` cyklus – počet opakování nemusí být předem dán

# Opakované vykonávání kódu – cykly

**Q:** V jakých případech potřebujeme opakovat části kódu?
**A:** Iterativní výpočet, práce s polem, ...

**Q:** Jaké dva základní typy cyklů v Matlabu známe a v čem se liší?
**A:** `for` cyklus – pevný počet opakování, `while` cyklus – počet opakování nemusí být předem dán

Případy použití:

- `for` cyklus – práce s polem, procházení tabulky (numerická integrace ODE Eulerovou metodou)
- `while` cyklus – iterativní výpočty (hledání kořene, hledání minima), výpočty s předem neznámým počtem opakování

# `while` cyklus

Cyklus, jenž se opakuje, dokud není splněna daná podmínka:

- `while` podmínka, příkazy, `end`

# `while` cyklus

Cyklus, jenž se opakuje, dokud není splněna daná podmínka:

- `while` podmínka, příkazy, `end`

## Příklad 4 (Zbytek po dělení)

Naprogramujte skript, který pro dvě přirozená čísla `delenec` a `delitel` najde zbytek po dělení dělence dělitelem pomocí opakovaného odčítání

# while cyklus

Cyklus, jenž se opakuje, dokud není splněna daná podmínka:

- `while` podmínka, příkazy, `end`

## Příklad 4 (Zbytek po dělení)

Naprogramujte skript, který pro dvě přirozená čísla `delenec` a `delitel` najde zbytek po dělení dělence dělitelem pomocí opakovaného odčítání

### Řešení

```
% máme zadáno třeba delenec=123 a delitel=4
zbytek = delenec;
while zbytek >= delitel
   zbytek = zbytek - delitel;
end
% proměnná zbytek teď obsahuje zbytek po dělení 123/4, tj. 3
```

# `for` cyklus

Cyklus, jenž má předem daný pevný počet opakování:

- `for proměnná=pole`, příkazy, `end`

## Příklad 5 (Faktoriál)

Naprogramujte výpočet faktoriálu $n!$ pro dané přirozené číslo $n$.

# for cyklus

Cyklus, jenž má předem daný pevný počet opakování:

- `for proměnná=pole`, příkazy, `end`

## Příklad 5 (Faktoriál)

Naprogramujte výpočet faktoriálu $n!$ pro dané přirozené číslo $n$.

### Řešení

```
% můžeme zkusit třeba n=6
factorial = 1;
for i = 1:n % všimněte si -- 1:n je pro Matlab pole!
  factorial = factorial * i;
end
% a máme factorial = 1*2*3*4*5*6 = 720
```

# Kermack-McKendrickův SIR model

Model šíření epidemie s třemi komponentami:

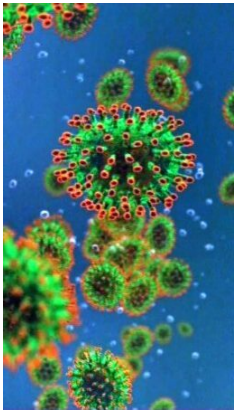
- vnímaví jedinci (angl. ***S**usceptible*) – $S(t)$
- nakažení jedinci (angl. ***I**nfected*) – $I(t)$
- jedinci mimo hru (angl. ***R**emoved*) – $R(t)$

Předpoklady

- uzavřená homogenní populace
- konstantní velikost populace $S(t) + I(t) + R(t) = \mathrm{c}$
- nulová inkubační doba
- infekční po celou dobu nemoci

Viz také http://mathworld.wolfram.com/Kermack-McKendrickModel.html

# Kermack-McKendrickův SIR model



Model je spojitý v čase s rovnicemi

$$\frac{\mathrm{d}}{\mathrm{d}t}S(t) = -\alpha I(t)S(t),$$
$$\frac{\mathrm{d}}{\mathrm{d}t}I(t) = \alpha I(t)S(t) - \beta I(t),$$
$$\frac{\mathrm{d}}{\mathrm{d}t}R(t) = \beta I(t),$$

kde

- $\alpha$ je koeficient nakažlivosti,
- $\beta$ je koeficient uzdravení,
- $1/\beta$ je doba trvání nemoci.

# Epidemiologický práh

Z druhé rovnice lze vyjádřit změnu počtu nakažených jako

$$I'(t) = I(t)(\alpha S(t) - \beta)$$

Počet infikovaných roste, pokud $\alpha S(t) - \beta > 0$.

Práh epidemie $R_0 = \dfrac{\alpha S(t)}{\beta}$

- $R_0 > 1$: počet nemocných roste $\Rightarrow$ epidemie
- $R_0 < 1$: počet nemocných klesá

Hodnota $R_0$ udává počet lidí, které nakazí jeden nemocný

- spalničky asi 15
- koronavirus SARS-CoV-2 cca 2,2 podle mutace (interval 1,4–3,8 bez ostatních opatření)

# Diskrétní verze modelu

Epidemiologové ale neměří data spojitě, měří je po dnech, potřebujeme tedy **diskrétní verzi modelu**.

Původní model počítá derivaci $\Rightarrow$ změnu hodnot $S(t)$, $I(t)$ a $R(t)$. Ty můžeme v delším časovém horizontu (nepříliš přesně) použít jako indikátory denních změn hodnot $S_k \to S_{k+1}$, $I_k \to I_{k+1}$, $R_k \to R_{k+1}$ pro dny $k = 0, 1, 2, \ldots, n$

# Diskrétní verze modelu

Epidemiologové ale neměří data spojitě, měří je po dnech, potřebujeme tedy **diskrétní verzi modelu**.

Původní model počítá derivaci $\Rightarrow$ změnu hodnot $S(t)$, $I(t)$ a $R(t)$. Ty můžeme v delším časovém horizontu (nepříliš přesně) použít jako indikátory denních změn hodnot $S_k \to S_{k+1}$, $I_k \to I_{k+1}$, $R_k \to R_{k+1}$ pro dny $k = 0, 1, 2, \ldots, n$

Pro diskrétní aproximaci řešení rovnic SIR modelu použijeme vlastně vám jíž známé Eulerovo schéma – a dostaneme

$$\begin{aligned} S_{k+1} &= S_k + \Delta S_k, & S_{k+1} &= S_k - \alpha I_k S_k \\ I_{k+1} &= I_k + \Delta I_k, & I_{k+1} &= I_k + \alpha I_k S_k - \beta I_k \\ R_{k+1} &= R_k + \Delta R_k, & R_{k+1} &= R_k + \beta I_k \end{aligned}$$

To už dokážeme zprogramovat.

# Proměnné a konstanty

- `alpha` ... koeficient nakažlivosti
- `beta` ... koeficient uzdravení
- `n` ... počet iterací
- `S` ... vnímaví jedinci
- `S0` ... jejich počáteční hodnota
- `I` ... infekční jedinci
- `I0` ... jejich počáteční hodnota
- `R` ... uzdravení jedinci
- `R0` ... `R0 = 0` (není to $R_0$)

# Proměnné a konstanty

- `alpha` ... koeficient nakažlivosti
- `beta` ... koeficient uzdravení
- `n` ... počet iterací
- `S` ... vnímaví jedinci
- `S0` ... jejich počáteční hodnota
- `I` ... infekční jedinci
- `I0` ... jejich počáteční hodnota
- `R` ... uzdravení jedinci
- `R0` ... `R0 = 0` (není to $R_0$)

V *Command Window* zadáme:

```
S0 = 10000
I0 = 10
alpha = 2e-5 % nebo 6e-5
beta = 0.07
n = ... % asi 60
```

# Vlastní skript

Do souboru `sirscript.m` vložíme kód pro diskrétní simulaci epidemie:

```
S = zeros(1,n+1); % počáteční hodnota + n iterací
I = zeros(1,n+1);
R = zeros(1,n+1); % R0 = 0, nebudeme nastavovat
S(1) = S0; % počáteční počet vnímavých
I(1) = I0; % počáteční počet infikovaných

for k = 1:n % ‘n‘ dní
  S(k+1) = S(k) - alpha*I(k)*S(k);
  I(k+1) = I(k) + alpha*I(k)*S(k) - beta*I(k);
  R(k+1) = R(k) + beta*I(k);
end
```

# Výstupy skriptu `sirscript.m`

V proměnných prostředí Matlabu se po dokončení výpočtu objeví tři nové: pole `S` udávající počet vnímavých v daném dni, pole `I` udávající počet infikovaných v daném dni, a pole `R` udávající počet uzdravených v daném dni.

# Jednoduchý graf

Hodnoty si nyní vykreslíme do grafu. Základní příkaz je `plot`:

- `plot(v)`, kde `v` je vektor:
  - na vodorovné ose index `i`
  - na svislé ose hodnoty `v(i)`

# Jednoduchý graf

Hodnoty si nyní vykreslíme do grafu. Základní příkaz je `plot`:

- `plot(v)`, kde `v` je vektor:
  - na vodorovné ose index `i`
  - na svislé ose hodnoty `v(i)`
- `plot(A)`, kde `A` je matice (tabulka hodnot):
  - na vodorovné ose řádkový index `i`
  - na svislé ose hodnoty `A(i,j)` postupně pro jednotlivá `j`
  - tj. pro každý sloupec `j` vznikne jeden graf
- `plot(x,y)`, pro `x` a `y` vektory stejné délky: XY-graf
  - na vodorovné ose hodnoty `x(i)`
  - na svislé ose hodnoty `y(i)`

# Výsledek naší simulace

Použijeme například maticovou verzi příkazu `plot`:

```
A = [ S' I' R' ]; % matice se třemi sloupci a ‘n‘ řádky
plot(A); % vykreslíme graf
```

# Výsledek naší simulace

Použijeme například maticovou verzi příkazu `plot`:

```
A = [ S' I' R' ]; % matice se třemi sloupci a ‘n‘ řádky
plot(A); % vykreslíme graf
```

Limity os jsou špatně, upravíme

```
xlim([0,n]); % rozsah osy ‘x‘
ylim([0,S0]); % rozsah osy ‘y‘
```

# Výsledek naší simulace

Použijeme například maticovou verzi příkazu `plot`:

```
A = [ S' I' R' ]; % matice se třemi sloupci a ‘n‘ řádky
plot(A); % vykreslíme graf
```

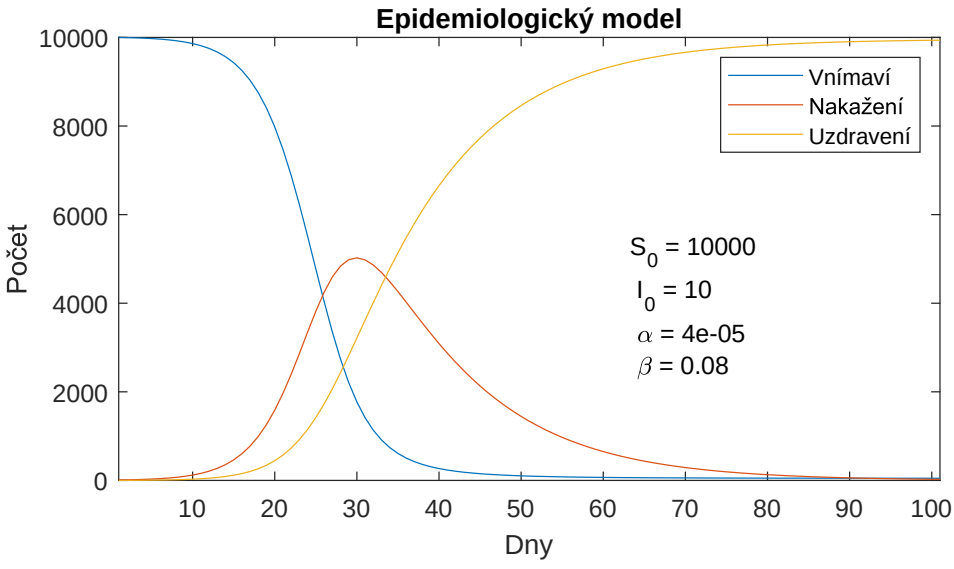
Limity os jsou špatně, upravíme

```
xlim([0,n]); % rozsah osy ‘x‘
ylim([0,S0]); % rozsah osy ‘y‘
```

V samotném grafu ale nevidíme, co je co, přidáme proto popis os, legendu k jednotlivým čarám, a titulek:

```
title('Epidemiologicky␣model');
xlabel('Dny');
ylabel('Pocet');
legend('Vnimavi' ,'Nakazeni', 'Uzdraveni');
```

# Výsledný graf



# Dodatky

# Doplňkové info

Mohli bychom ještě udělat, pokud zbyde čas:

- SIR pomocí m-funkce
- Ukládání a formátování obrázků

# SIR jako funkce

Pro opakované vyhodnocování je lepší mít náš SIR model implementovaný jako funkci a vola jej například jako

```
[S,I,R] = sirmodel(S0,I0,alpha,beta,n); % funkce může vrace více,
než jednu hodnotu
```

M-funkce v Matlabu se od skriptu odlišuje tím, že má záhlaví/hlavičku, které říká, co funkce přijímá za parametry a jaké parametry navrací:

```
function [S,I,R] = sirmodel(S0,I0,alpha,beta,n);
% Sem se případně píše popis parametrů
... % sem přijde tělo skriptu sirscript.m
```

**Q:** Jak musíme nazvat m-soubor s funkcí `sirmodel()`?

# SIR jako funkce

Pro opakované vyhodnocování je lepší mít náš SIR model implementovaný jako funkci a vola jej například jako

```
[S,I,R] = sirmodel(S0,I0,alpha,beta,n); % funkce může vrace více,
než jednu hodnotu
```

M-funkce v Matlabu se od skriptu odlišuje tím, že má záhlaví/hlavičku, které říká, co funkce přijímá za parametry a jaké parametry navrací:

```
function [S,I,R] = sirmodel(S0,I0,alpha,beta,n);
% Sem se případně píše popis parametrů
... % sem přijde tělo skriptu sirscript.m
```

**Q:** Jak musíme nazvat m-soubor s funkcí `sirmodel()`? **A:** `sirmodel.m`

# Jak uložit graf do PDF/SVG/EPS

Matlab ve verzi 2020b (a možná i dříve, to zjistíme) umí konečně smysluplně ukládat grafické výstupy do vektorových formátů i jiných, než je EPS. Stačí na to použít metodu `saveas`:

```
f = figure(); % ‘f‘ odkazuje na daný obrázek
plot(...); % vykreslíme graf
... % další kouzla (legenda, titulek, anotace, ...)
saveas(f, 'figure.eps', 'epsc'); % tohle šlo vždy, 'epsc' = barva
saveas(f, 'figure.pdf'); % tohle kreslilo na A4, nově už OK
saveas(f, 'figure.svg'); % tohle umí 2020b => HTML, Inkscape
```

# Jak změnit rozměry obrázku

Pokud potřebujeme nestandardní rozměry obrázku, musíme měnit vlastnosti obrázku, na něž nově odkazují složky proměnné `f`:

```
f.PaperUnits = 'centimeters'; % lze i 'inches', viz dokumentace
f.PaperSize = [15,8]; % šíře 15, výška 8 centimetrů
f.PaperPositionMode = 'manual'; % poloha obrázku je dána ručně
f.PaperPosition = [0,0,15,8]; % obrázek je 15x8cm a začíná v [0,0]
```

Další informace viz `doc figure` a odtud “Figure Properties”.

# Literatura



JIRKOVSKÝ Jaroslav. Úvod do prostředí MATLAB [online]. Webový seminář. Praha: Humusoft, 2013. Dostupné z https://www.mathworks.com/videos/introduction-to-matlab-81427.html


KOVÁŘ Bohumil, PŘIKRYL Jan, PĚNIČKA Martin, VLČEK Miroslav, HODNÝ Lukáš. Jemný úvod do Matlabu a Simulinku [online]. Praha: FD ČVUT, 2007. Dostupné z https://zolotarev.fd.cvut.cz/msp/ctrl.php?act=show,file,23


LUDVÍK Pavel, MORÁVKOVÁ Zuzana. Numerická matematika: Řešené příklady s Matlabem a aplikované úlohy [online]. Ostrava: VŠB-TU Ostrava, 2016. Dostupné z http://mdg.vsb.cz/portal/nm/nmsm.pdf


NAGY Ivan. Úvod do Matlabu [online]. Praha: ČVUT FD, 2012. Dostupné z https://www.fd.cvut.cz/department/k611/PEDAGOG/11THOS/Uvod_do_Matlabu_Nagy.pdf


VONDRÁK Vít. Úvod do Matlabu [online]. Ostrava: FEI VŠB-TU Ostrava. Dostupné z https://homel.vsb.cz/~luk76/Teaching/FMMI/matlab.pdf