Cvičení 2 – Matlab

Modelování systémů a procesů

Mgr. Lucie Kárná, PhD

karna@fd.cvut.cz

March 3, 2020

1. Začínáme programovat v Matlabu
   - m-files
   - Větvení
   - Cykly

2. Model epidemie
   - Kermack-McKendrickův SIR model
   - Model v Matlabu
   - Grafický výstup

m-files

= příkazy uložené v textovém souboru s příponou `.m`

Typy m-souborů

scripty sekvence příkazů

- všechny proměnné globální
- volají se jménem souboru

m-files

= příkazy uložené v textovém souboru s příponou `.m`

Typy m-souborů

scripty 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 musí být totožné se jménem funkce

Větvení

```
if <podmínka> <příkazy> end
if <podmínka> <příkazy1> else <příkazy2> end
```

Větvení

```
if <podmínka> <příkazy> end
if <podmínka> <příkazy1> else <příkazy2> end
```

Příklad 1

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

Větvení

if *<podmínka>* *<příkazy>* end

if *<podmínka>* *<příkazy1>* else *<příkazy2>* end

Příklad 1

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

Příklad 2

```
if a==b disp('cisla se rovnaji')
else disp('cisla se nerovnaji') end
```

while-cyklus

`while` *<podmínka>* *<příkazy>* `end`

cyklus, kde není předem známý počet opakování

while-cyklus

`while` *<podmínka>* *<příkazy>* `end`

cyklus, kde není předem známý počet opakování

Příklad 3

Naprogramujte script, který pro dvě přirozená čísla `velke` a `male` najde zbytek po dělení prvního čísla druhým.

while-cyklus

`while` *<podmínka>* *<příkazy>* `end`

cyklus, kde není předem známý počet opakování

Příklad 3

Naprogramujte script, který pro dvě přirozená čísla `velke` a `male` najde zbytek po dělení prvního čísla druhým.

Řešení

```
zbytek = velke;
while zbytek >= male
zbytek = zbytek - male;
end
```

for-cyklus

`for i=1:n` *<příkazy>* `end`

cyklus, u kterého je předem známý počet opakování

for-cyklus

`for i=1:n` <*příkazy*> `end`

cyklus, u kterého je předem známý počet opakování

Příklad 4

Naprogramujte výpočet faktoriálu $n!$.

for-cyklus

`for i=1:n` <*příkazy*> `end`
cyklus, u kterého je předem známý počet opakování

Příklad 4

Naprogramujte výpočet faktoriálu $n!$.

Řešení

```
factorial = 1;
for i = 1:n
factorial = factorial * i;
end
```

Kermack-McKendrickův SIR model

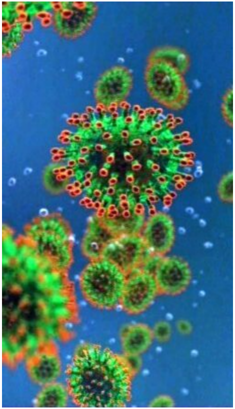
$S(t)$ vnímaví jedinci (**S**usceptible)
$I(t)$ nakažení jedinci (**I**nfected)
$R(t)$ jedinci mimo hru (**R**emoved)

Předpoklady

- uzavřená homogenní populace
- velikost populace $S(t) + I(t) + R(t) = const.$
- nulová inkubační doba
- infekční po celou dobu nemoci

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

Kermack-McKendrickův SIR model



Rovnice modelu

$$
\begin{aligned}
S'(t) &= -\alpha I(t)S(t) \\
I'(t) &= \alpha I(t)S(t) - \beta I(t) \\
R'(t) &= \beta I(t)
\end{aligned}
$$

α koeficient nakažlivosti

β koeficient uzdravení

$1/\beta$ doba trvání nemoci

Epidemiologický práh

Z druhé rovnice vyjádříme: $I'(t) = I(t)(\alpha S(t) - \beta)$
počet infikovaných roste, pokud $\alpha S(t) - \beta > 0$

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

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

α … počet lidí, které nakazí jeden nemocný

- spalničky asi 15
- koronavirus SARS-CoV-2 cca $2,2$ (interval $1,4$–$3,8$)

Proměnné a konstanty

```
% alpha ... koef. nakazlivosti
% beta ... koeficient uzdraveni
% n ... pocet iteraci
```

```
% S ... vnimavi jedinci
% S0 ... pocatecni hodnota
% I ... infekcni jedinci
% I0 ... pocatecni hodnota
% R ... uzdraveni jedinci
% R0 = 0
```

Proměnné a konstanty

```
% alpha ... koef. nakazlivosti
% beta ... koeficient uzdraveni
% n ... pocet iteraci


% S ... vnimavi jedinci
% S0 ... pocatecni hodnota
% I ... infekcni jedinci
% I0 ... pocatecni hodnota
% R ... uzdraveni jedinci
% R0 = 0
```

V *Command Window* zadáme:

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

Vlastní script

```
S = zeros(1,n+1); % pocatecni hodnota + n iteraci
S(1) = S0;
I = zeros(1,n+1);
I(1) = I0;
R = zeros(1,n+1); % R0 = 0

for j = 1:n
S(j+1) = S(j) - alpha*I(j)*S(j);
I(j+1) = I(j) + alpha*I(j)*S(j) - beta*I(j);
R(j+1) = R(j) + beta*I(j);
end
```

Jednoduchý graf

Základní příkaz: `plot`

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

Jednoduchý graf

Základní příkaz: `plot`

- `plot(v)`, `v` je vektor:
  - na vodorovné ose index `i`
  - na svislé ose hodnoty `v(i)`
- `plot(A)`, `A` je matice (= tabulka):
  - na vodorovné ose řádkový index `i`
  - na svislé ose hodnoty `A(i,j)`
  - tj. pro každý **sloupec** `j` jeden graf

Jednoduchý graf

Základní příkaz: `plot`

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

Výsledek naší simulace

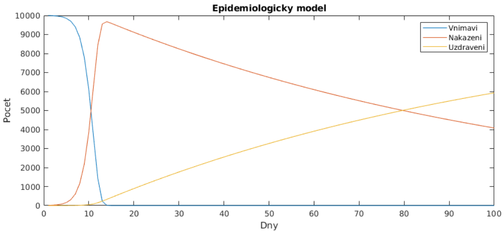
```
» A = [ S' I' R' ];
» plot(A);
```

Výsledek naší simulace

```
» A = [ S' I' R' ];
» plot(A);
» title('Epidemiologicky model');
» xlabel('Dny');
» ylabel('Pocet');
» legend('Vnimavi' ,'Nakazeni', 'Uzdraveni');
```

Výsledek naší simulace

```
» A = [ S' I' R' ];
» plot(A);
» title('Epidemiologicky model');
» xlabel('Dny');
» ylabel('Pocet');
» legend('Vnimavi' ,'Nakazeni', 'Uzdraveni');
```




That's all Folks!