EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

ALGORITMIZACE I

...aneb trocha teorie

CHARAKTERISTIKY

- Algoritmus je…
  pracovní postup, který má následující vlastnosti

  - Konečnost – finitnost: musí skončit v *konečném* počtu kroků
  - Elementárnost: základních typů kroků existuje konečný počet, složitější problém lze vždy řešit pomocí těchto kroků
  - Obecnost – universalita: řeší třídu problémů, nikoli jeden konkrétní problém
  - Určitost – determinovanost: vždy musí být jasné, co dělat
  - Výstup – resultativnost: musí mít aspoň jeden výstup (jinak vytvoříme „černou díru“, do níž cpeme informace a ono nic ☺)

METODY VYTVÁŘENÍ

- Shora dolů:
  - postupuje se od složitějších problémů k jednodušším
  - problém rozkládáme na jednotlivé podproblémy
  - skončíme u elementárních kroků
- Zdola nahoru:
  - z elementárních kroků vytváříme složitější prvky (viz „Karel“)
- Kombinace obou:

VYJÁDŘENÍ

- **Algoritmus lze vyjádřit různě**
  dle toho, jak jsou popsány jeho jednotlivé kroky
  - **slovně:** větami v přirozeném jazyce
  - **graficky:** grafickými značkami se slovním popisem, např. pomocí vývojových diagramů
  - **matematicky:** soustava rovnic, vztah mezi veličinami
  - **programem:** instrukcemi určitého procesoru

EFEKTIVNOST ALGORITMU

- Pokud danou úlohu řeší více algoritmů, vybírá efektivnější podle určitých kritérií:
  - **časové:** úloha vyřešena v kratším čase (uvažujeme strojový čas tj. počet instrukcí procesoru)
  - **paměťové:** spotřeba paměti
  - **přehlednost, srozumitelnost:** (důležité pro další vývoj a úpravy)

POJMY

- **Algoritmizace:** proces vytváření a sestavování algoritmů
- **Programování:** zakódování algoritmu do zvoleného programovacího jazyka
- **Programovací jazyk:** umělý jazyk používaný pro definování sekvence programových příkazů, které lze zpracovat na počítači.
- Algoritmus má obecnou povahu, zatímco implementace algoritmu v určitém programovacím jazyku je ryze konkrétní.

PŘÍPRAVA ČAJE

- Nejjednodušší varianta, algoritmus popsán slovně
  1. Vstup: Připrav si hrnek, varnou konvici, sáček čaje, lžičku
  2. Natoč vodu do konvice po rysku odpovídající objemu hrnku
  3. Zapni konvici
  4. Počkej do vypnutí konvice
  5. Vlož sáček do hrnku
  6. Zalej vodou z konvice
  7. Čekej 3-5 min.
  8. Vyndej sáček
  9. Výstup: čaj

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 1 – PROSTŘEDÍ, REŽIMY...

Seznámení s prostředím

K dispozici máme:

- *Command Window* (příkazové onno) – spouštějí se v něm příkazy či programy, vypisují se výsledky a chybové hlášky
- *Current folder* – okno pro správu adresářů
- *Command History* – okno historie příkazů, hodí se při vytváření a zkoušení části programů
- *Workspace* – informace o definovaných proměnných (typ, rozsah, obsah)

Začínáme:

- **nastavení pracovní složky**
  provádí se pomocí editačního pole **Current Directory** na panelů nástrojů okna MATLAB při každém spuštění aplikace!!!
- **práce v dialogovém režimu**
  ≡ v podokně **Command Window** se zapisují příkazy do příkazového řádku uvozeného znaky >> a potvrzují se klávesou ENTER
- práce v grafickém režimu
  ≡ v podokně Figure, které se otevírá:
  a) příkazem **figure** v podokně Command Window nebo
  b) nabídkou **File/New/Figure** v aplikačním okně MATLAB
  edituje se graf a ukládá do souboru, úpravy je možno uložit jako podprogram (APPS)
- **práce v programovém režimu**
  ≡ v kódovém editoru, který se spouští nabídkou **File/New/M-file**, zapisuje a ukládá příkazy do m-souboru
- **ukončení**
  ≡ quit exit zavření okna

Nápověda

- **help**
  vypíše textovou nápovědu v příkazovém podokně Command Window
  >> **help**
  vypíše seznam instalovaných knihoven a toolboxů
  >> **help matlab\general**
  vypíše nápovědu ke knihovně funkcí »general«
  >> **help who**
  vypíše nápovědu k funkci »who«
- **helpwin**
  otevře textovou nápovědu v novém aplikačním okně Help
- **helpdesk**
  otevře hypertextovou dokumentaci v aplikačním okně Help

- **demo**
  otevře hypertextovou nabídku demonstračních ukázek v aplikačním okně Help
- **doc příkaz**
  otevře podrobný popis příkazu v novém okně, podrobnější, uvádí i příklady, odkazy na další

HELP topics:
documents\MATLAB -
matlab\testframework - (No table of contents file)
matlab\demos – příklady použití
matlab\graph2d – 2D grafika
matlab\graph3d – 3D grafika
matlab\graphics – rozhraní, aktivní prvky
matlab\plottools – editace vlastností grafických násrojů

Proměnné

- Název smí obsahovat alfanumerické znaky, je case sensitive (rozlišuje malá a velká písmena)
- Název musí začínat písmenem
- Desetinným oddělovačem je tečka
- Název nesmí být shodný se jmény vestavěných funkcí či proměnných, jinak se přepisují jejich hodnoty
- Název nesmí začínat číslicí

Defaultně definované:

- ans – proměnná používaná k zobrazení výsledků jinak nepřiřazeného jiné proměnné
- pi – Ludolfovo číslo
- i, j – komplexní jednotka
- eps – přesnost
- realmin, realmax – nejmenší a největší možné kladné reálné číslo
- Inf – nekonečno
- NaN – neurčitý výraz (např. 0/0)
- nargin, nargout – počet vstupních či výstupních parametrů funkce

Formát zobrazování čísel

Matlab pracuje v režimu dvojnásobné přesnosti (double precission), implicitně zobrazuje 4 desetinná místa

Tab. 1 Vybrané formáty čísel v Matlabu

Typ formátu	zobrazuje	Příklad
format short	5 číslic (4 desetinná místa)	3.1416
format short e	minimálně 5 číslic + exponent	3.1416e+000
format long	15 číslic (14 desetinných míst)	3.14159265358979
format long e	15 číslic + exponent	3.14159265358979e+000
format hex	hexadecimální číslo	400921fb54442d18
format bank	2 desetinná místa	3.14
format +	kladná i záporná čísla se znaménkem	+3.1416

Syntaxe

Přiřazovací příkaz: **proměnná = výraz**

- Potlačit zobrazení výsledku lze středníkem za příkazem (pozn. jsou místa, kde se středník psát NESMÍ)
- Argumenty funkcí se dávají vždy do kulatých závorek
- Je možno jednoduše generovat vektory či matice, řádky se oddělují středníkem, sloupce čárkou, při výčtu prvků se používají hranaté závorky
- Operace – základní +, -, *, /, ^ (umocnění)
- Jednoduše je možno matice či vektory transponovat pomocí apostrofu

- Generování vektoru či matice **od:krok:do**
- Jiná možnost **linspace(od, do, počet**)
- Matice do hranatých závorek

Dialogový režim

- Je přístupný v okně *command window*
- Příkazy se ihned vykonávají
- V tomto režimu lze použít MATLAB jako inteligentní kalkulátor
- Jsou dostupné elementární matematické funkce help matlab\elfun
- Použité proměnné se zachovávají v paměti, lze je vypsat příkazem **who,** s jejich rozměry pak příkazem **whos**
- Proměnné je možno mazat příkazem **clear název_proměnné**, všechny pak příkazem **clear all** (uplatní se později v programovém režimu)
- Pozn.: k cíli, kterého chceme dosáhnout, může vést více postupů

Speciální příkazy pro tvorbu matic

- Příkazy: **zeros**, **ones**, **eye**

Submatice

- Z každé matice lze separovat její různé submatice a naopak lze i různé matice skládat jako jednotlivé submatice do nějaké jiné větší
- Submatice se získá pomocí zápisu **A(od:do,od:do)**, kde od značí číslo řádku (sloupce), odkud se začíná submatice vytvářet, do číslo řádku (sloupce), kde se bude končit
- Pokud se použije místo rozsahu **od:do** pouhá dvojtečka (:), znamená to, že do výběru spadají všechny řádky (sloupce) matice
- Sdružované submatice zapisujeme do hranatých závorek, např. **A=[A1, A2]** nebo **A=[A1; A2]**
- Změna rozměru matice **reshape(matice, řádky,sloupce)**
- Použití pro konec matice/vektoru označení **end**
- Nedefinovaný prvek **NaN** (not a number)
- Velikost matice lze zjistit pomocí příkazu **size**

Příklady pro řešení v dialogovém režimu

Příklad 01.01:

Přepočtěte teplotu 50°C na stupně Fahrenheita

$$F = \frac{9}{5}C + 32$$

Příklad 01.02:

Spočtěte 3., 10. a 50. člen Fibonacciho posloupnosti dle vzorce

$$F_n = \frac{\sqrt{5}}{5}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right]$$

Pozn.: využijte chytře okna historie příkazů

Příklad 01.03:

Jaký je tlak vzduchu v pneumatice nákladního automobilu při teplotě 20 °C a hustotě 8 kg·m^{-3}
Molární hmotnost vzduchu $M_m \doteq 29.10^{-3}$ kg·mol^{-1}

$$p = \frac{mRT}{M_m V}$$

Příklad 01.04:

Určete tepelnou kapacitu vzorku pro teplotu 293 K dle rovnice

$c_p = a + bT + cT^{-2} + dT^{-\frac{1}{2}}$

$a = 104{,}35 \;\; b = 6{,}07.10^{-3} \;\; c = 3{,}4.10^{4} \;\; d = -1070$

Příklad 01.05:

Generujte řádkový vektor $\mathbf{a}$ přirozených čísel od 1 do 10. Totéž pro sloupcový vektor

Příklad 01.06:

Generujte vektor $\mathbf{b}$, který bude obsahovat racionální čísla od 1 do 10 s krokem 0,2

Příklad 01.07:

Generujte matici $\mathbf{X}$ o rozměrech 3x3, která bude obsahovat samé jednotky a nulovou matici **Y** stejných rozměrů

Příklad 01.08:

Generujte matici $\mathbf{C}$ o rozměrech 3x3, která bude obsahovat následující prvky:

1, 2, 3

4, 5, 6

7, 8, 1

Příklad 01.09:

Generujte matice

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} -1 & -2 & 3 \\ -6 & 7 & 4 \\ 7 & 8 & 9 \end{bmatrix}$$

Definujte následující maticové operace

$$\mathbf{A}_1 + \mathbf{A}_2; \;\; [\mathbf{A}_1 \quad \mathbf{A}_2]; \;\; \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}; \;\; \mathbf{A}_1.\mathbf{A}_{2;}\, \mathbf{A}_2.\mathbf{A}_1$$

Z matice $\mathbf{A}_1$ vyberte 2. a 3. prvek 2. řádku

Programový režim

- Pracuje se zvláštním editorem
- Je možno a doporučeno používat komentáře, uvozují se znakem **%** (procento). Vše za procentem do konce řádku je považováno za komentář a ignoruje se, pokud se použijí 2 % za sebou, editor to chápe jako komentář oddělující novou sekci programu, lze tedy takto program přehledněji rozčlenit
- Příkazy je možno oddělovat čárkou nebo středníkem
- Do programového režimu se lze dostat následujícími cestami:
  a) ikona či příkaz menu **Command window** pro otevření nového souboru
  b) otevření již existujícího programového souboru v Matlabu (dvojklik myší)
  c) ikona či příkaz menu **Command window** pro otevření již existujícího programového souboru v Matlabu
- Soubor je před spuštěním nutno uložit na disk a pojmenovat
- Soubor má standardní příponu *.m
- Soubor se spouští v okně **Command window** zapsáním názvu souboru
- Vyučující vás seznámí s tím, na které disky máte povoleno své soubory ukládat
- Je vhodné nastavit si na příslušný disk cestu pomocí ikony **Path Browser**
- Pro jednoduché zobrazení výsledků lze použít příkaz **plot(x,y)**

Grafický režim – úvod

- Grafika bude podrobně probrána samostatně
- Používá se samostatné grafické okno **figure, p**říkazem **figure** lze otevřít nové okno (otvírá se defaultně po zadání některého příkazu pro zobrazení)
- Lze použít také položku menu **plots**

- Příkazem **close** lze zavřít poslední aktivní grafické okno, vyplatí se používat na počátku programů příkaz **close all** (v nové verzi Matlabu již stačí napsat jen close)
- Pro základní zobrazení se používá příkaz **plot(x,y),** kde **x** a **y** jsou sloupcové vektory
- Podrobnější informace lze nalézt pomocí příkazu **help plot**
- Pro určení barvy zobrazovaných dat se příkaz modifikuje do tvaru **plot(x,y,'l')**, kde **l** je zkratka barvy zobrazovaných dat
- Pro modifikaci typu čáry či volbu zobrazení pomocí značek se příkaz modifikuje do tvaru **plot(x,y,'z')** , kde **z** je zkratka typu čáry či značky
- Oba předchozí příkazy lze kombinovat jako **plot(x,y,'lz')**
- Pro přidání pomocného měřítka se použije příkaz **grid**
- Pro změnu rozsahu zobrazení se použije příkaz **axis([xmin, xmax, ymin, ymax])**
- Osy lze popsat pomocí příkazů **xlabel('text'), ylabel('text')**
- Celý graf lze popsat pomocí příkazu **title('text')**

Příklady pro řešení v programovém režimu

Příklad 01.10:

Tabelujte funkce $y_1 = \sin t,\ y_2 = \cos t\ ,\ t \in \langle -\pi;\ \pi\rangle,\ \Delta t = \pi/30$

Příklad 01.11:

Tabelujte funkce

$$f_1 = \exp(-x),\ f_2 = \exp(-3x),\ f_3 = \exp\left(-\frac{x}{3}\right), x \in \langle -1;\ 1\rangle, \Delta x = 0{,}1$$

Příklad 01.12:

Tabelujte funkci $y = 5x + 1;\ \ x \in \langle -5; 5\rangle;\ \Delta x = 0{,}1$

Příklad 01.13:

Je k dispozici balónek naplněný kryptonem. Do jaké hloubky by bylo třeba tento balónek v mořské vodě ponořit, aby se přestal vznášet k vodní hladině a začal klesat ke dnu? Předpoklady: teplota 25 °C, atmosférický tlak 101,3 kPa, hustota mořské vody 1,04 g/cm^3 nezávislá na hloubce a ideální chování plynu. Vliv pryžového obalu se zanedbává. Tíhové zrychlení g = 9,81 m/s^2, M =83,7 g/mol

$$h = \frac{1}{\rho \mathrm{g}}\left(\frac{\mathrm{R}T}{V_m} - p_a\right)$$

Příklad 01.14:

Generujte sloupcový vektor $x \in \langle -5; 5\rangle$, tak,aby obsahoval 200 hodnot. Spočtěte funkci $y = -5x + 1$

Příklad 01.15:

Je dána matice

$$\mathbf{A} = [\mathbf{A}_1 \quad \mathbf{A}_2]$$

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} -1 & -2 & 3 \\ -6 & 7 & 4 \\ 7 & 8 & 9 \end{bmatrix}$$

Vytvořte z matice $\mathbf{A}$ matici $\mathbf{B}$ tak, aby měla 9 řádků a 2 sloupce. Sledujte, jak se prvky řadí za sebou. Vytvořte matici $\mathbf{C}$ tak, aby měla 2 řádky a 9 sloupců. Pomocí příkazů **size** se přesvědčte o velikosti matic.

Příklad 01.16:

Generování datových proměnných – příkaz **datetime**
příklad datetime(rok, měsíc, den, hodina, minuta sekunda).
Generujte následující proměnné:
d1: 29.2.2016;
d2: 1.11.2020 17:00;
d3: 1. únor pro roky 1999 až 2010 jako sloupcový vektor;
d4: 1. každého měsíce roku 2020 jako sloupcový vektor;
d5: všechny dny října 2021

Řešení úloh v dialogovém režimu

Příklad 01.01:

```
>> C = 50; F = 9/5*50 + 32
F =   122
```

Příklad 01.02:

```
>> n = 3;
>> F = sqrt(5)/5*(((1+sqrt(5))/2)^(n+1)-((1-sqrt(5))/2)^(n+1))
F =    3.0000
>> n = 10;
>> F = sqrt(5)/5*(((1+sqrt(5))/2)^(n+1)-((1-sqrt(5))/2)^(n+1))
F =   89.0000
>> n = 50;
>> F = sqrt(5)/5*(((1+sqrt(5))/2)^(n+1)-((1-sqrt(5))/2)^(n+1))
F =   2.0365e+10
```

Příklad 01.03:

```
>> T = 20 + 273.15; R = 8.314; rho = 8; Mr = 29e-3;
>> p = rho * R * T / Mr
p =   6.7234e+05
```

Příklad 01.04:

```
>> a = 104.35; b = 6.07e-3; c = 3.4e4; d = -1070;
>> T = 293;
>> cp = a + b*T + c*T^-2 + d*T^-0.5
cp =   44.0145
```

Příklad 01.05:

```
>> a = 1:10
a =     1     2     3     4     5     6     7     8     9    10
>> a = a'
```

Příklad 01.06:

```
>> b = 1:0.2:10
```

Příklad 01.07:

```
>> X = ones(3,3),  Y = zeros(3,3)
```

Příklad 01.08:

```
>> C = [1,2,3;4,5,6;7,8,1]
C =
     1     2     3
     4     5     6
     7     8     1
```

Příklad 01.09:

```
>> A1 = [1 2 3;6 4 5;7 8 9]; A2 = [-1 -2 3;-6 7 5;7 8 9];
>> A1 + A2
ans =
     0     0     6
     0    11    10
    14    16    18
>> [A1 A2]
ans =
     1     2     3    -1    -2     3
     6     4     5    -6     7     5
     7     8     9     7     8     9
>> [A1; A2]
ans =
     1     2     3
     6     4     5
     7     8     9
```

```
    -1    -2     3
    -6     7     5
     7     8     9
>> A1*A2
ans =
     8    36    40
     5    56    83
     8   114   142
>> A2*A1
ans =
     8    14    14
    71    56    62
   118   118   142
>> A1(3,2:3)
ans =     8     9
```

Řešení úloh v programovém režimu

```
%% Program 01.01
  clear, close, clc
  C = 50; % C
  F = 9/5 * 50 + 32 % Fahrenheit

%% Program 01.02
   clear, close, clc
%  vzorec nepřehledný, je rozumné zavést pomocné proměnné
%  pokud se budou počítat všechny členy naráz, nutno uvažovat tečkové
%  operace
   n = [1 3 10]; % zadat ty členy, které chceme spočíst
   p = sqrt(5);  % pomocné proměnné
   p1 = ((1 + p)/2).^(n + 1);
   p2 = ((1 - p)/2).^(n + 1);
   F = p * (p1 - p2) % elegantní vzorec nakonec

%% Program 01.03
   clear, close, clc
%  Vstupní data
   T = 20 + 273.15; % K
   R = 8.314;       % J/mol/K
   rho = 8;         % kg/m^3
   Mr = 29e-3;      % kg/mol
   p = rho * R * T / Mr;
   p = p / 1e6      % MPa

%% Program 01.04
   clear, close, clc
%  Vstupní data
   a = 104.35;
   b = 6.07e-3;
   c = 3.4e4;
   d = -1070;
   T = 293;
   cp = a + b * T + c * T ^ -2 + d * T^ -0.5

%% alternativa (data do matice koeficientů)
%  má cenu, pokud bychom chtěli srovnat vlastnosti více látek
%  k = [ a b c d]
   k = [104.35, 6.07e-3, 3.4e4, -1070];
   cp = k(1) + k(2) * T + k(3) * T ^ -2 + k(4) * T^ -0.5
```

```
%% Program 01.09
   clear, close, clc
% elegantní zápis matice
   A1 = [1 2 3;...
         6 5 4;...
         7 8 9];
   A2 = [-1 -2 3;...
         -6 7 4;...
          7 8 9];
% maticové operace
  A1 + A2
  [A1 A2]
  [A1; A2]
  A1 * A2
  A2 * A1
  A1(2,2:3)

%% Program 01.10
   clear, close, clc
   t = -pi:pi/30:pi;
   y1 = sin(t);
   y2 = cos(t);
   plot(t,y1,t,y2)
   xlabel('t'),ylabel('y')

%% Program 01.11
   clear, close, clc
   x=-1:0.1:1;
   f1 = exp(-x);
   f2 = exp(-3*x);
   f3 = exp(-x/3);
   plot(x,f1,x,f2,x,f3)
   xlabel('x'),ylabel('y')

%% Program 01.12
   clear, close, clc
   x = -5:0.1:5;
   y = 5*x+1;
   plot(x,y,'b')
   xlabel('x'),ylabel('y')

%% Program 01.13
   clear, close, clc
   g = 9.81;     % ms-2
   pa = 101.3e3; % Pa
   t = 25;       % C
   rho = 1.04;   % g cm-3
   Mr = 83.7;    % g mol-1
   R = 8.314;
   V = Mr/rho;
   V = V*1e-6;
   p = R*(t+273.15)/V;
   h = (p-pa)/g/rho/1000 % 3 km

%% Program 01.14
   clear, close, clc
   x = linspace(-5,5,200);
   y = -5*x+1;
   plot(x,y,'b')
   xlabel('x'),ylabel('y')
```

```
%% Program 01.15
   clear, close, clc
   A1 = [1 2 3;6 4 5;7 8 9];
   A2 = [-1 -2 3;-6 7 5;7 8 9];
   A = [A1 A2]
   B = reshape(A,9,2)
   C = reshape(A,2,9)
   size(B), size(C)

%% Program 01.16
   clear, close, clc
   d1 = datetime(2016,2,29)
   d2 = datetime(2020,11,1,17,0,0)
   d3 = datetime((1999:2010)',2,1)
   d4 = datetime(2020,(1:12)',1)
   d5 = datetime(2021,10,(1:30))
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

ALGORITMIZACE II

...aneb vývojové diagramy

CHARAKTERISTIKA

- **Vývojový diagram** slouží ke grafickému znázornění jednotlivých kroků procesu, pracovního postupu obecně či algoritmu

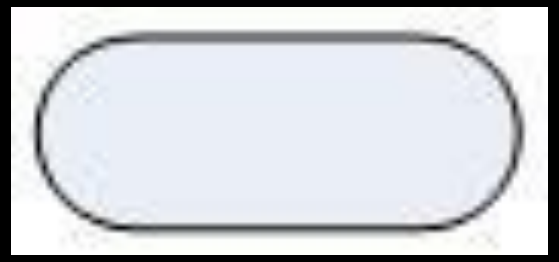
  - Obsahuje obrazce různého tvaru navzájem propojené pomocí šipek
  - Obrazce reprezentují jednotlivé kroky, šipky tok řízení.
  - Standardně nezobrazují tok dat, ten je zobrazován pomocí data flow diagramů
  - Využíván v informatice pro analýzu, návrh, dokumentaci nebo řízení procesů

POUŽÍVANÉ ZNAČKY

- Vývojové diagramy používají následující značky:

ZAČÁTEK, KONEC

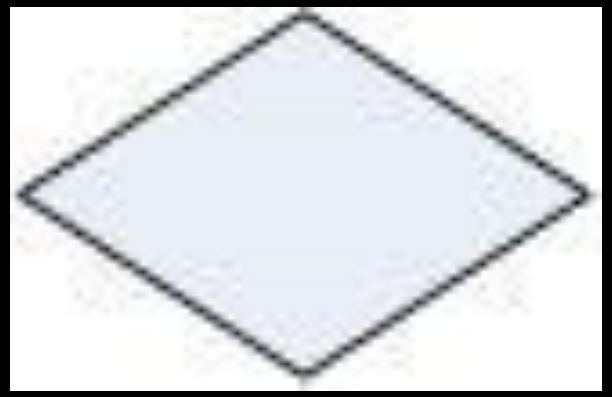
- Začátek a konec algoritmu
- Může obsahovat text, který říká, o co se jedná



PŘIŘAZOVACÍ PŘÍKAZ

- Používá se pro jeden krok zpracování algoritmu
- Většinou jde o tzv. přiřazovací příkaz vezmi něco, udělej s tím něco, schovej si to někam
- Př. A=1
  (v paměti vytvoř proměnnou s označením A a ulož do ní hodnotu 1)

ROZHODOVÁNÍ



- Větvení algoritmu
- Je zadána podmínka
- Pokud je splněna, pokračuje algoritmus dále jednou větví
- Pokud splněna není, vydá se druhou
- Př.
  Máš žízeň?
  Ano: dej si pivo
  Ne: makej dál

CYKLUS S PEVNÝM POČTEM OPAKOVÁNÍ

- Značka se používá pro začátek a konec cyklu
- Počet opakování řídí nějaké počítadlo průchodů cyklem
- Př.
  Pro počítadlo rovno 1,2,…,10
  Dej si pivo

CYKLUS S PODMÍNKOU

- Podmínka může být na počátku cyklu, nebo na konci (v tom případě proběhne vždy alespoň jednou)
- Značky se moc nepoužívají, lze obejít rozhodovací strukturou
- …(ale pro fajnšmekry uvádíme)…

VSTUPY A VÝSTUPY

- Načítání ze souboru

- Uložení do souboru

VSTUPY A VÝSTUPY

- Ruční vstup (zadává se z klávesnice v dialogovém režimu)

- Obecný vstup či výstup

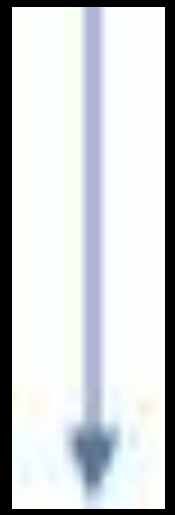
PODPROGRAM

- Část algoritmu, která řeší dílčí (relativně samostatný) problém
- Př. Výpočet faktoriálu při řešení výpočtu kombinačních čísel

SPOJOVACÍ ZNAČKA

- Pokud je algoritmus velký a potřebuji jej rozdělit např. na více listů papíru či osamostatnit jednotlivé větve rozhodování
- Značka se zapisuje tam, kde se přeruší, i tam, kde se naváže
- Do obou značek se píše číslo, aby bylo jasné, co na co navazuje

SPOJOVACÍ ČÁRA



- Je orientovaná úsečka
- Ukazuje, odkud kam se postupuje

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 2 – OPERÁTORY, FUNKCE

Operátory

- Aritmetické –
  a) maticové: + (sčítání), - (odčítání), * (násobení), / (dělení – to není chyba, matematicky je maticové dělení násobení inverzní maticí), ^ (umocnění), ‘ (transpozice)
  b) vektorové – provádí se po jednotlivých elementech, v zápisu se před operátor napíše tečka: .* (násobení), ./ (dělení), .^ (umocnění)
- Relační: Tab. 2 – výsledkem je buď nepravda (nula), nebo pravda (jednička)
- Logické: Tab. 3
- Přehled pomocí nápověd help ops, help arith, help relop

Funkce

- Jako operátory, i funkce se dělí do kategorií:
  a) skalární – aplikují se na každý prvek matice, běžné funkce typu sin(x), e^x,… Přehled pomocí nápovědy **help elfun**
  b) vektorové – aplikují se na každý sloupec matice, převážně statistické funkce typu suma (sum), minimum (min), maximum (max), směrodatná odchylka (std), průměr (mean), medián… Přehled pomocí nápovědy **help stats**
  c) maticové – aplikují se na celou matici, funkce typu determinant (det), inverse (inv)… Přehled pomocí nápovědy **help matfun, help elmat, help specfun**

Tab. 2 Srovnávací operace

Operátor	Význam	Operátor	Význam
==	Je rovno	~=	Není rovno
>	Je větší než	<	Je menší než
>=	Je větší nebo rovno	<=	Je menší nebo rovno

Tab. 3 Logické operátory

Operátor	Význam	Operátor	Význam
&	and	|	or
~	not	xor	exclusive or

Tab. 4 Vybrané funkce v Matlabu

Trigonometrické			
Funkce	**Popis**	**Funkce**	**popis**
sin	sinus, argument v radiánech	sind	sinus, argument ve stupních
cos	kosinus, argument v radiánech	cosd	kosinus, argument ve stupních
tan	tangens, argument v radiánech	tand	tangens, argument ve stupních
cot	kotangens, argument v radiánech	cotd	kotangens, argument ve stupních
asin	Inverzní funkce, argument v radiánech	asind	Inverzní funkce, argument ve stupních
acos		acosd	
atan		atand	
acot		acotd	
sinh	Hyperbolické funkce	asinh	Inverzní hyperbolické funkce
cosh		acosh	
tanh		atanh	
coth		acoth	
Exponenciální			
Funkce	**Popis**	**Funkce**	**Popis**
exp	exponenciála	log	přirozený logaritmus
sqrt	2. odmocnina	log10	dekadický logaritmus
Zaokrohlovací			
Funkce	**Popis**	**Funkce**	**Popis**
fix	směrem k nule	floor	směrem k -∞
round	směrem k nejbližšímu celému číslu	ceil	směrem k +∞
mod	funkce modulo	rem	zbytek po celočíselném dělení
sign	znaménková funkce		
Komplexní			
Funkce	**Popis**	**Funkce**	**Popis**
real	reálná část	abs	modul, velikost
imag	imaginární část	angle	fázový úhle
conj	komplexně sdružená část	isreal	testovací funkce reálných čísel
Maticové			
Funkce	**Popis**	**Funkce**	**Popis**
cross	vektorový součin	dot	skalární součin
inv	inverzní matice	eig	vektor vlastních čísel
sum	součet sloupců matice	diag	prvky diagonály
triu	horní trojúhelníková matice z původní	tril	dolní trojúhelníková matice z původní

Příklady pro samostatné řešení

Příklad 02.01:

Graficky znázorněte průběh tlaku pro H_2O_2 podle Antoineovu rovnici pro rozsah teplot 20 až 150 °C, s krokem 5°C

$\log p = A - \frac{B}{C+t}$ (°C, kPa), A = 5,817; B = 1264,74; C = 171,561

Příklad 02.02:

Vypočtěte průběh rychlosti v (m.s^{-1}) pro časový interval $\langle 0; 5 \rangle$ min s krokem 0,1 minuty. Rychlost je určena vztahem

$$v = \frac{t^3}{\sqrt[4]{1+3t^4}}$$

Příklad 02.03:

Pomocí determinantů řešte soustavu rovnic:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 6 \\ 4x_1 + 5x_2 + 6x_3 &= 15 \\ 7x_1 + 8x_2 + x_3 &= 16 \end{aligned}$$

Řešte tuto soustavu elegantněji pomocí inverze a ještě elegantněji pomocí maticového dělení

Příklad 02.04:

Pro matici $\mathbf{A}$=[1 1,1 1,2; 1,5 1,7 1,9; 2,1 2,4 3] určete: její determinant, je-li možno tak inverzní matici $\mathbf{A}$, sumu jednotlivých řádků, sumu jednotlivých sloupců, průměr řádků i sloupců, minimální a maximální prvek matice a součet všech prvků matice.

Příklad 02.05:

Tom je pyšný, že našel o 2 valouny zlata více než Joe. Jima mrzí, že má o 2 valouny méně, než Sam. Fred našel tolik, kolik mají Jim a Sam dohromady. Tom s Joem si za svých společných 24 valounů chtějí koupit vilu na Canaria Islands. Celá parta vytěžila dohromady 44 valounů. Kolik tedy kdo má?

Příklad 02.06:

Pro data: 1; 1,1; 1,11; 0,98; 1,35; 0,8; 1,25; 1,1; 1,1; 1,2; 0,9 spočtěte průměr a směrodatnou odchylku, určete minimum a maximum.

Příklad 02.07:

Svobodník čekatel Karotka na své pravidelné noční obchůzce Stínovem zatknul 17 podezřelých zločinců. Do rukou spravedlnosti bylo předáno třikrát méně nepoctivých hostinských než neurvalých opilců. Počet zatčených falešných švadlen byl o 2 menší než opilců a počet neopatrných kapsářů byl o jeden větší než falešných švadlen. Zjistěte, kolik a kterých narušitelů veřejného pořádku svobodník čekatel Karotka zadržel.

Příklad 02.08:

Řešte soustavu rovnic

$$\begin{aligned} 5x_1+8x_2+5x_3+2x_4+5x_5+5x_6 &= 0 \\ 3x_1+2x_2+4x_3+8x_4+3x_5+10x_6 &= -10 \\ x_1-x_2+x_3-2x_4+x_5-4x_6 &= 10 \\ 4x_1-4x_2+2x_3-2x_4+8x_5+10x_6 &= 10 \\ 5x_1+x_2-x_3+x_4-x_5 &= 1 \\ 2x_1-2x_2+4x_3-x_4+3x_5-11x_6 &= 22 \end{aligned}$$

Příklad 02.09:

Tabelujte funkci

$$y = \frac{\sqrt{x}}{x+1};,\ x \in \langle 0; 5\rangle, \Delta x = 0{,}01$$

Příklad 02.10:

Tabelujte funkci

$$y = 5x^3 - 3x^5;,\ x \in \langle -1{,}5; 1{,}5\rangle, \Delta x = 0{,}01$$

Příklad 02.11:

Bratři Jarda, Pepa a Honza jsou přezdívaní jako Dlouhý, Široký a Krátkozraký. Honza je totiž nejmenší, měří 140 cm, nosí silné brýle a i s postelí váží tolik co Pepa. Pepa je zase velmi tělnatý a na svou střední výšku 150 cm má tak vysoké BMI, že je stejné jako Jardovo a Honzovo BMI dohromady. Jarda je naproti tomu hubený a z bratrů nejvyšší, měří 168 cm. Zjistěte hmotnosti bratrů s přesností na kilogramy, když víte, že jejich průměrná hmotnost je 55 kg, a že všichni dohromady i se svými postelemi, které jsou všechny stejné, váží 300 kg. Kolik váží Jarda, Pepa a Honza dohromady? Kolik váží jejich postele? Zjistěte také BMI každého z bratrů.

Poznámka: BMI je číselné vyjádření míry obezity člověka a vypočítá se jako podíl hmotnosti člověka v kilogramech a kvadrátu jeho výšky v metrech. Například člověk s hmotnostní 70 kg a výškou 180 cm má BMI $70 \div (1{,}8)^2$, což je přibližně 21,6.

Příklad 02.12:

Upravte matici z příkladu 2.08 následovně: nahraďte všechny prvky větší než 5 hodnotou ∞ a všechny prvky menší než -5 hodnotou $-\infty$

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

ALGORITMIZACE III

...aneb vytváříme diagramy a rozhodujeme

SEKVENCE

- Jedná se o nejjednodušší verzi algoritmu
- Příkazy jdou po sobě bez oklik, skoků a větvení
- Postupně se provedou
- Př. Vaření čaje z Algoritmizace I

VAŘÍME ČAJ A)

- Úkol: znázorněte graficky vývojovým diagramem

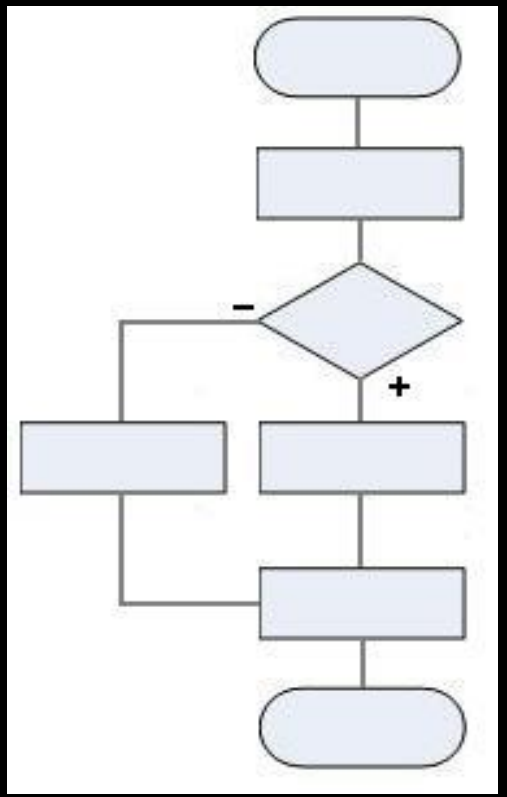
1. Vstup: Připrav si hrnek, varnou konvici, sáček čaje, lžičku
2. Natoč vodu do konvice po rysku odpovídající objemu hrnku
3. Zapni konvici
4. Počkej do vypnutí konvice
5. Vlož sáček do hrnku
6. Zalej vodou z konvice
7. Čekej 3-5 min.
8. Vyndej sáček
9. Výstup: čaj

VĚTVENÍ

- Algoritmus se větví na několik částí pomocí rozhodování
- Pokud je podmínka splněna, provede se jiná sekvence příkazů, než v případě, že splněna není

ÚPLNÁ ALTERNATIVA

- Algoritmus se větví do dvou částí:
  - pokud je podmínka splněna, provede se sekvence, pokud ne, provede se jiná část
  - Př. Rozhodujeme, zda do čaje dáme mléko nebo citron (je jasné, že obojí zároveň by vytvořilo poněkud nepoživatelnou směs



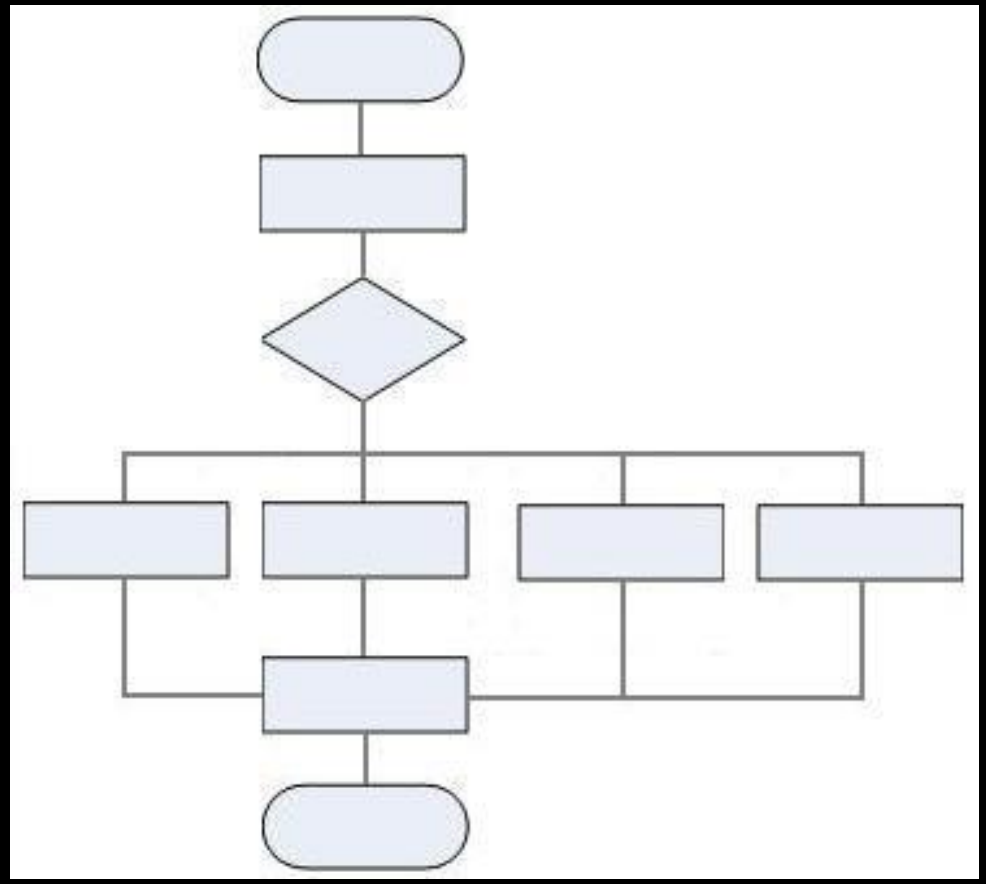
VAŘÍME ČAJ B)

- Úkol: znázorněte graficky vývojovým diagramem

1. Vstup: Připrav si hrnek, varnou konvici, sáček čaje, lžičku, mléko, citron
2. Natoč vodu do konvice po rysku odpovídající objemu hrnku
3. Zapni konvici
4. Počkej do vypnutí konvice
5. Vlož sáček do hrnku
6. Zalej vodou z konvice
7. Čekej 3-5 min.
8. Vyndej sáček
9. Mléko (+) nebo citron (-) ?
   - (+) Přidej mléko
   - (-) Přidej citron
10. Výstup: čaj

VÍCENÁSOBNÉ VĚTVENÍ

- Algoritmus se větví do několika částí:
  - To, kterou větví se pustí, řídí nějaký přepínač
  - Př. Rozhodujeme, zda do čaje dáme:
    - mléko (anglická varianta)
    - citron (nachlazená varianta)
    - rum (chlapská zmrzlá varianta)
    - růžový květ (romantická varianta)



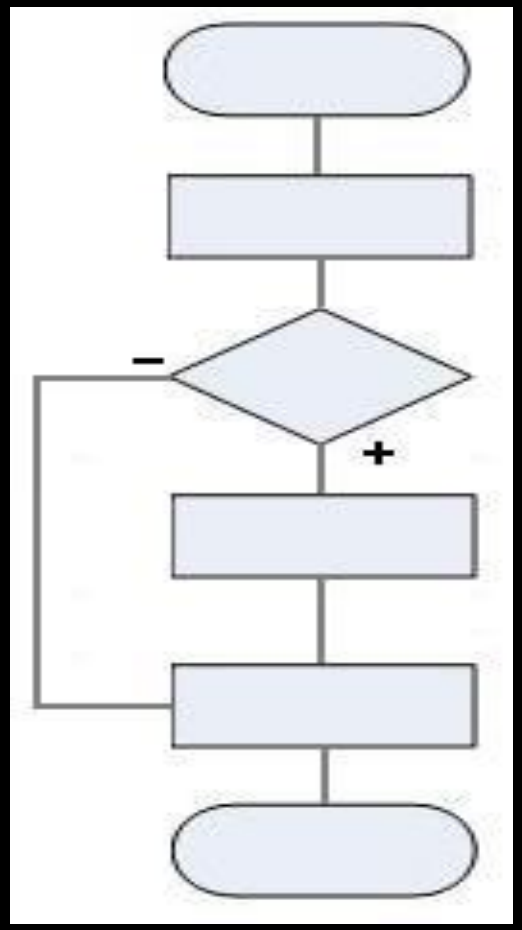
VAŘÍME ČAJ C)

- Úkol: znázorněte graficky vývojovým diagramem

1. Vstup: Připrav si hrnek, varnou konvici, sáček čaje, lžičku, mléko, citron, rum, růžový květ
2. Natoč vodu do konvice po rysku odpovídající objemu hrnku
3. Zapni konvici
4. Počkej do vypnutí konvice
5. Vlož sáček do hrnku
6. Zalej vodou z konvice
7. Čekej 3-5 min.
8. Vyndej sáček
9. Co přidáš ?
   (a) Přidej mléko (b) Přidej citron (c) Přidej rum (d) Přidej růžový květ (e) Nepruď a dej sem ten čaj
10. Výstup: čaj

NEÚPLNÁ VARIANTA

- Algoritmus se větví do dvou částí:
  - V případě, že je podmínka splněna, se provede nějaká sekvence příkazů
  - V případě že splněna není, se tato sekvence neprovede a algoritmus pokračuje dál
  - Př. Rozhodujeme se, zda čaj osladíme (buď sladíme nebo nic)



VAŘÍME ČAJ D)

- Úkol: znázorněte graficky vývojovým diagramem

1. Vstup: Připrav si hrnek, varnou konvici, sáček čaje, lžičku, cukr
2. Natoč vodu do konvice po rysku odpovídající objemu hrnku
3. Zapni konvici
4. Počkej do vypnutí konvice
5. Vlož sáček do hrnku
6. Zalej vodou z konvice
7. Čekej 3-5 min.
8. Vyndej sáček
9. Sladíš?
   (+) Přidej cukr
10. Výstup: čaj

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 3 – 2D GRAFIKA

- Obecný tvar příkazu pro práci s objektem:
  název = příkaz (vlastnost, hodnota)
- Nastavit vlastnost objektu lze příkazem **set(objekt, vlastnost, hodnota)**
- Lze upravovat více vlastností naráz (někdy je to dokonce nutné)
- Příkaz pro otevření grafického okna (viz lekce 1) je **figure**
- Používané jednotky (Units) jsou doporučovány v režimu Normal: je to interval <0,1> x <0,1>, kde pozice <0,0> je levý dolní a pozice <1,1> pravý horní roh obrazovky
- Pozici a velikost okna lze nastavit příkazem
  set(gfc, 'Units', 'Normal', 'Position', [x_1, y_1, x_2, y_2])
  kde **x_1, y_1** je pozice levého dolního rohu grafického okna od levého dolního rohu obrazovky a **x_2, y_2** šířka a výška grafického okna v týchž jednotkách
- V rámci prostoru okna figure lze umístit podobně i souřadný systém příkazem
  souradnice = axes('Position', [x_1, y_1, x_2, y_2])
- Příkaz **get(objekt)** vypíše všechny existující vlastnosti objektu s jejich aktuálními hodnotami
- Barvy: každá z barev je definována kombinací 3 základních barev (červená – zelená – modrá) v rozsahu <0;1>
- Pozn. Příkaz **pause** je na zastavení výpočtu, abychom si mohli pozorně prohlédnout výsledky, pokračuje se stiskem libovolné klávesy

Příklad 03.01:

```
%% Program 03.01
%  Modelový příklad 2D grafika
   clear, close, clc
%  definice umístění grafického okna a jeho vlastností
   figure;  % otevření okna
%  pozice okna
   set(gcf, 'Units', 'Normal', 'Position', [0.4,0.5,0.5,0.3])
%  pozice souřadného systému v okně
   s1 = axes('Position', [0.1,0.1,0.8,0.7])
   y = sin(0:pi/10:2*pi);
   plot(y)      % graf fce
   get(gcf) % zjistí vlastnosti grafického okna
   get(s1)  % zjistí vlastnosti grafu v rámci okna
%  Vlastnosti zjišťujeme proto, aby bylo jasné, co měnit a jak

%% Vybrané vlastnosti objektu s1 a jejich přenastavení
%  Color [x y z] barva plochy grafu
   set(s1,'Color',[1 0 0]), pause   % R
   set(s1,'Color',[0 1 0]), pause   % G
   set(s1,'Color',[0 0 1]), pause   % B
```

```
   set(s1,'Color',[1 1 1]), pause   % W

%  Box on/off obrys grafu
   set(s1,'Box','on'), pause, set(s1,'Box','off')

%  Další, např. pro popis grafu
%  FontName 'jmeno', FontSize [velikost],
%  LineWidth [velikost], NextPlot ‘replace’
%  Visible on/off viditelnost grafu
%  XColor, YColor, ZColor barva os, vztahuje se i na grid
%  Nelze měnit barvu čáry grafu, je to jiný objekt!!!
```

Tab. 5 Grafické zobrazení objektů

Barva čáry		Typ čáry		Značka			
r	červená	-	plná	.	bod	v	trojúhelníček dolů
g	zelená	:	tečkovaná	o	kroužek	^	trojúhelníček nahoru
b	modrá	-.	čerchovaná	x	křížek	<	trojúhelníček vlevo
c	tyrkys	--	čárkovaná	+	křížek plus	>	trojúhelníček vpravo
m	purpur			*	hvězdička	p	pětiúhelníček
y	žlutá			s	čtvereček	h	šestiúhelníček
k	černá			d	kosočtverec		

Parametry zobrazovaných čar se udávají v pořadí **plot (x1, y1,<barva> <značky> <typ čáry>)**, podrobněji viz Tab. 5.

- Šířku čáry lze nastavit pomocí parametru **LineWidth**, velikost značky pak pomocí parametru **MarkerSize**
- Často je potřeba umístit do téhož grafického okna více grafů, tj. je zde nutnost okno rozdělit. Příkaz, který to umožní, je **subplot(yxz)**, kde **y** udává počet grafů, které se do okna umístí ve směru osy y, **x** počet grafů ve směru osy x a **z** pořadové číslo grafu (čísluje se zleva shora)

Příklad 03.02:

```
%% Program 03.02
%  Modelový příklad 2D grafika subplot
   clear, close, clc
   y1 = sin(0:pi/10:2*pi);
   y2 = cos(0:pi/10:2*pi);
   xlabel('\it x'), ylabel('\it y')

%  a) 2 grafy pod sebou
   subplot(211), plot(y1)
   xlabel('\it x'), ylabel('\it y')
   subplot(212), plot(y2)
   xlabel('\it x'), ylabel('\it y')

%  b) 2 grafy vedle sebe
   figure;
   subplot(121), plot(y1)
   xlabel('\it x'), ylabel('\it y')
   subplot(122), plot(y2)
   xlabel('\it x'), ylabel('\it y')

%  c) 4 grafy
   figure;
   subplot(221), plot(y1)
```

```
    xlabel('\it x'), ylabel('\it y')
    subplot(222), plot(y2)
    xlabel('\it x'), ylabel('\it y')
    subplot(223), plot(2*y1)
    xlabel('\it x'), ylabel('\it y')
    subplot(224), plot(2*y2)
    xlabel('\it x'), ylabel('\it y')

%   d) 3 g. spodní větší
    figure;
    subplot(221), plot(y1)
    xlabel('\it x'), ylabel('\it y')
    subplot(222), plot(y2)
    xlabel('\it x'), ylabel('\it y')
    subplot(212), plot(y1+y2)
    xlabel('\it x'), ylabel('\it y')
```

Poznámka: v nových verzích MATLAB je adekvátním příkazem: **tiledlayout(rows,columns)**, k dalšímu okénku se posouvá příkazem **nexttile**. Pokud není zadáno dělení, bude si MATLAB okna organizovat sám. Výhodou je, že lze globálně popsat všechny grafy naráz

```
%%  Alternativní způsob
    clear, close, clc
    t = 0:pi/10:2*pi;
    y1 = sin(t);
    y2 = cos(t);

   figure; tiledlayout(3,1)
   nexttile, plot(t,y1), axis('tight')
   nexttile, plot(t,y2), axis('tight')
   nexttile, plot(t,y1+y2), axis('tight')

   xlabel('\it x'), ylabel('\it y')
```

Popis grafu

- Popisky os do grafu se vloží pomocí příkazu **xlabel** a **ylabel**, titulek pomocí **title**.
- Pokud má příslušná popiska obsahovat více řádek, je třeba jednotlivé řádky této popisky zapsat do složených závorek.
- Mřížka se zobrazí pomocí **grid**.
- K zobrazení legendy se používá příkaz **legend.** Zadá-li se **legend boxoff**, dojde k odstranění rámečku a výplně plochy legendy. Navrácení původního stavu nastane po zadání **legend boxon**.
- Umístění legendy pomocí **Location**, více v **help legend**
- Pro zobrazení několika průběhů v jednom grafu je použit příkaz **hold**, implicitní nastavení je **hold off**. Vykreslení další funkce do stejného osového systému je možné po zadání příkazu **hold on**.
- Do grafu je také možné pomocí příkazu **text(x,y,string)** vkládat jakýkoliv text, který je pak v grafu umístěn na zadanou pozici vztahující se ke konkrétním souřadnicím.
- Rozměry os se ovládají příkazem **axis**

Tab. 6 Některé speciální 2D grafy

Příkaz	Popis
plotyy	2 různé osy y (sekundární osa)
semilogx	logaritmická x, lineární y
semilogy	logaritmická y, lineární x
loglog	logaritmická x i y
bar	sloupečkový ekonomický graf
histogram	vykreslí histogram hodnot
scatter	bodový, data znázorněna kroužky
area	vybarvuje plochu pod křivkou

Ostatní typy v help a v okně PLOTS

Výstup grafů do souborů

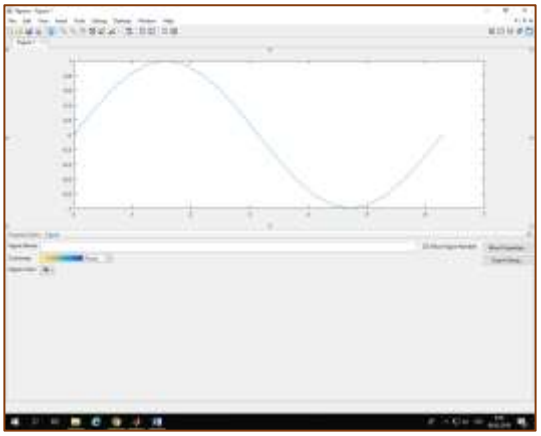
- Pomocí příkazu print; formát **print(soubor,typ)** – soubor udává jméno souboru, typ typ fomátu grafu (uvozeno –d), např. –dpng
- **Pozn**. Starší verze Matlabu umožňují zápis **print soubor typ** (současná verze ho připouští jako alternativu)
- Typy souborů, které Matlab umí, se získají pomocí **help print** nebo **doc print**

Ovládání 2D grafiky pro líné programátory

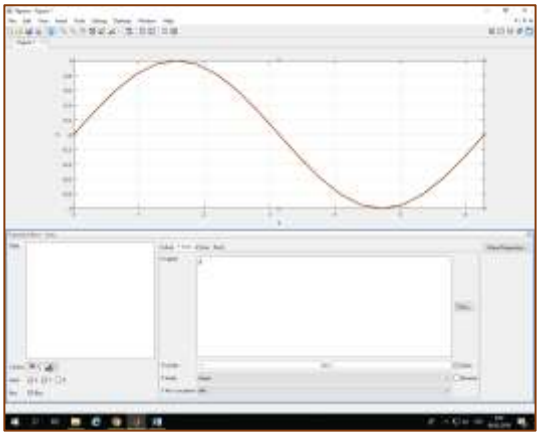
- Je potřeba udělat alespoň základní zobrazení dat, tedy např.

```
clear, close, clc
x = 0:pi/10:2*pi;  y = sin(x);
plot(x,y)
```

- V grafickém okně rozkliknout poslední položku: **ShowPlot Tools and Dock Figure**
- Ještě líněji: je potřeba vytvořit aspoň nezávislou a závislou proměnnou, označit si je a pokračovat v menu PLOTS (platí pro novější verze MATLAB, od 2016 výše)



Obr. 1 ShowPlot Tools and Dock Figure



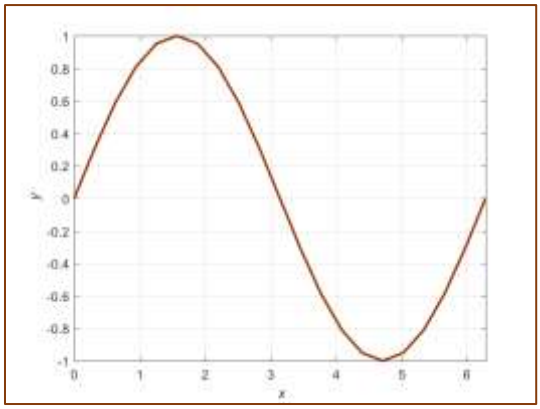
Obr. 2 Graf po ruční úpravě

- Nyní lze myší vybírat jednotlivé prvky grafu (osy, křivka…) a upravovat je
- Pokud se bude jednat o standardní úpravy, které se budou opakovat, má cenu vygenerovat kód pomocí **File – generate Code** a vzniklou funkci `createfigure` upravit, uložit do stejného adresáře jako program (neměnit název, příště budeme vědět, proč) a v programu místo příkazu plot použít `createfigure(x,y)` – za nezávislou a závislou proměnnou se dosadí jména, která jsme použili

```
function createfigure(X1, Y1)
% CREATEFIGURE(X1, Y1)
%  X1:  vector of x data
%  Y1:  vector of y data

  figure1 = figure; % Create figure
  axes1 = axes('Parent',figure1); % Create axes
  hold(axes1,'on');
% Create plot
  plot(X1,Y1,'DisplayName','x','LineWidth',2,...
    'Color',[0.600000023841858 0.200000002980232 0]);
  xlabel('\it x'); ylabel('\it y'); % Create labels
  xlim(axes1,[0 6.2832]); box(axes1,'on'); % X-limits of the axes
  set(axes1,'XGrid','on','YGrid','on'); % Set the remaining axes properties
end
```

- Upravený program

```
%% Program 03_02b
   clear all, close all, clc
   x = 0:pi/50:2*pi;
   y = sin(x);
%  plot(x,y)        % tohle se teď nahradí generovaným kódem
   createfigure(x, y)
   print Fig0302 -dpng
```



Obr. 3 Výsledek upraveného programu

Příklad 03.03:

Programátorsky ošetřený graf

```
%% Program 03.03
   clear, close, clc
   figure
   x = 0:pi/50:2*pi; y1 = sin(x); y2 = cos(x);
   plot(x,y1,'m-.','LineWidth',2.5), hold on
   plot(x,y2,'b:','LineWidth',2.5)
   grid, xlabel('\it x'), ylabel('\it y')
   title('Graf funkce sinus a cosinus')
   legend('sin(\it{x})','cos(\it{x})',...
          'Location', 'BestOutside')
   text(pi,0,'\leftarrow sin(\pi)') % text do grafu
   axis tight % aby nezůstávalo volné místo u os
```

Příklady pro samostatné řešení

Příklad 03.04:

Pro příklady **02.01; 02.02**; **02.09**; **02.10** zobrazte průběhy funkcí do jednoho grafického okna (4 podgrafy), nastavte správně popisy grafu, přidejte tloušťky a barvy čar, popis os, legendu

Příklad 03.05:

Síla působící na těleso byla vyjádřena závislostí

$$F = 4{,}2\ m\frac{\ln(t+1)}{t+1}$$

Spočtěte tuto sílu pro tělesa o hmotnosti 50 a 100 kg v časovém rozmezí 0 až 0,1 hodiny (převeďte na sekundy), graficky znázorněte do jednoho okna – dva samostatné grafy a 3. graf pro srovnání účinků síly. Zobrazte totéž v logaritmických souřadnicích pro osu x.

Příklad 03.06:

Zobrazte průběhy funkcí $y_1 = 2\,x^2 - \ln|x|$; $y_2 = x^2\,(4 - x)$; $y_3 = \sin x\,(1 + \cos x)$ na intervalu $x \in \langle -6;\ 6\rangle$; $\Delta x = 0{,}01$. Znázorněte do jednoho grafického okna do 3 grafů pod sebou. Graf uložte do souboru s názvem Graf03_06 jako typ souboru .png

Příklad 03.07:

Zobrazte průběh funkce zadané parametricky pro $t \in \langle -1;\ 1\rangle$; $\Delta t = 0{,}01$; $x = t\exp(t)$; $y = t^3 + 6t$. Do grafu vyneste závislost y na x.

Příklad 03.08:

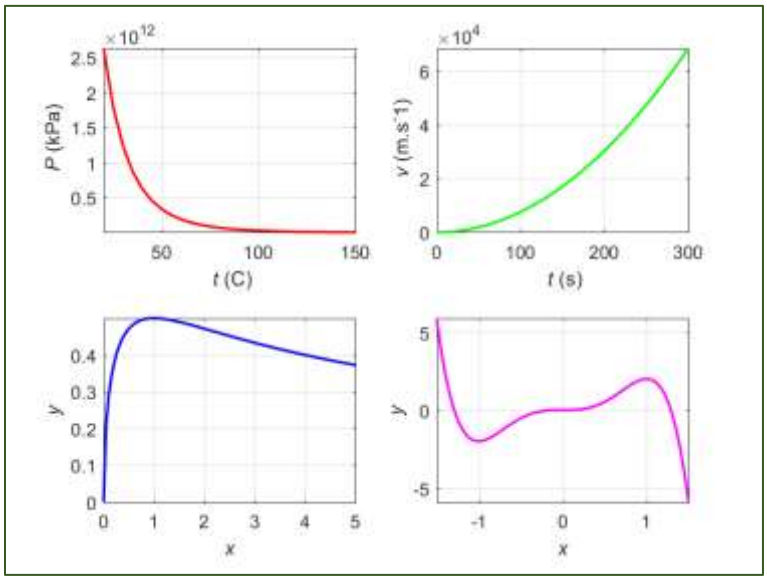
Zobrazte průběh funkce zadané parametricky pro $t \in \langle 0;\ 100\rangle$; $\Delta t = 0{,}01$;
$x = t\cos t$; $y = t\sin t$. Do grafu vyneste závislost y na x.

Příklad 03.09:

Vytvořte program pro výpočet vrhu šikmého. Vstupními údaji budou: počáteční výška (m), počáteční rychlost (m/s), úhel vrhu (ve stupních, přepočte se v programu na radiány). Výstupem bude graf, který má na ose x vzdálenost (m) a na ose y výšku (m). Ošetřete záporné hodnoty y (nahraďte pomocí logického příkazu nulami). Bez znalosti derivací najděte maximální výšku a zjistěte, v jakém čase nastane.

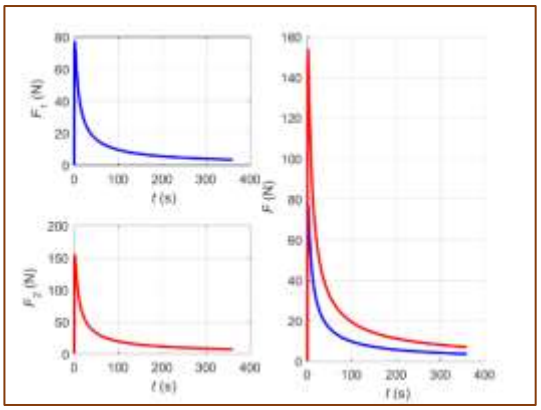
Řešení úloh 3

```
%% Program 03.04
   clear, close, clc
   figure

%  a) 2.01
   A = 5.817; B = 1264.74; C = 171.56; % def. konstant
   t = 20:5:150; logP = A+B./(C+t); P = 10.^logP;
   subplot(221)
   plot(t,P,'r','LineWidth',1.5),
   xlabel('{\it t} (C)'),ylabel('{\it P} (kPa)')
   grid, axis tight

%  b) 2.02
   t = 0:.1:5; t = t*60;
   v = t.^3./((1+3*t.^4).^(1/4));
   subplot(222)
   plot(t,v,'g','LineWidth',1.5),
   xlabel('{\it t} (s)'),ylabel('{\it v} (m.s^-1)')
   grid, axis tight

%  c) 2.09
   x = 0:.05:5; y = x.^(1/2)./(x+1);
   subplot(223)
   plot(x,y,'b','LineWidth',1.5)
   xlabel('\it{x}'),ylabel('\it{y}')
   grid, axis tight

%  c) 2.10
    x= -1.5:.01:1.5; y = 5*x.^3-3*x.^5;
   subplot(224)
   plot(x,y,'m','LineWidth',1.5)
   xlabel('\it{x}'),ylabel('\it{y}')
   grid, axis tight
```

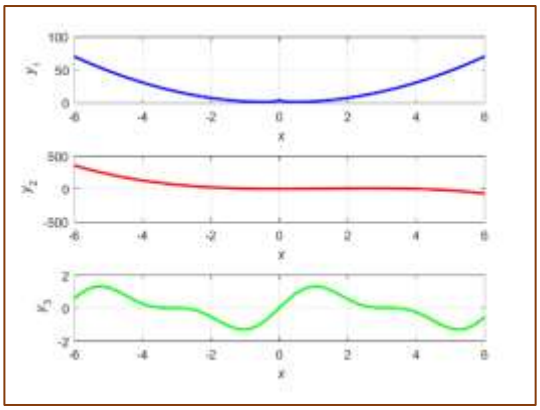


Obr. 4 Graf příkladu 03.04

```
%% Program 03.05
   clear all, close all, clc
   m = [50; 100]; % kg
   t = (0:1:.1*3600)'; % hodiny po minutě převod na s
   F=  4.2*log(t+1)./(t+1).*m';
%% v lineárních souřadnicích
   figure
   subplot(221)
   plot(t,F(:,1),'b','LineWidth',2),grid
   xlabel('{\it t} (s)'), ylabel('{\it F}_1 (N)')

   subplot(223)
   plot(t,F(:,2),'r','LineWidth',2),grid
   xlabel('{\it t} (s)'), ylabel('{\it F}_2 (N)')
   subplot(122)

   hold on
   plot(t,F(:,1),'b','LineWidth',2),plot(t,F(:,2),'r','LineWidth',2)
   xlabel('{\it t} (s)'), ylabel('{\it F} (N)')
   grid, hold off

%% v semilogaritmických souřadnicích
   figure
   subplot(221)
   semilogx(t,F(:,1),'b','LineWidth',2),grid
   xlabel('ln {\it t} '), ylabel('{\it F}_1 (N)')

   subplot(223)
   semilogx(t,F(:,2),'r','LineWidth',2),grid
   xlabel('ln {\it t}'), ylabel('{\it F}_2 (N)')

   subplot(122)
   semilogx(t,F(:,1),'b','LineWidth',2),hold on
   semilogx(t,F(:,2),'r','LineWidth',2)
   xlabel('ln {\it t}'), ylabel('{\it F} (N)')
   grid, hold off


%% Program 03.06
   clear, close, clc
   x = 1:0.01:6;
   y1 = 2*x.^2-log(x);
   y2 = x.^2.*(4-x);
   y3 = sin(x).*(1+cos(x));

   figure
   subplot(311)
   plot(x,y1,'b','LineWidth',2), grid
   xlabel('{\it x}'), ylabel('{\it y}_1')
   subplot(312)
   plot(x,y2,'r','LineWidth',2), grid
   xlabel('{\it x}'), ylabel('{\it y}_2')
   subplot(313)
   plot(x,y3,'g','LineWidth',2), grid
   xlabel('{\it x}'), ylabel('{\it y}_3')
   print Fig0306 -dpng
```

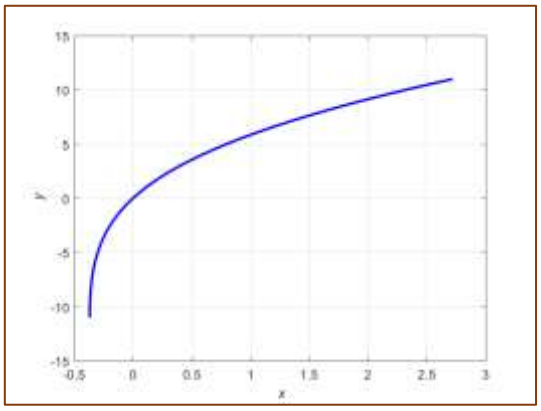


Obr. 5 Graf př. 03.05

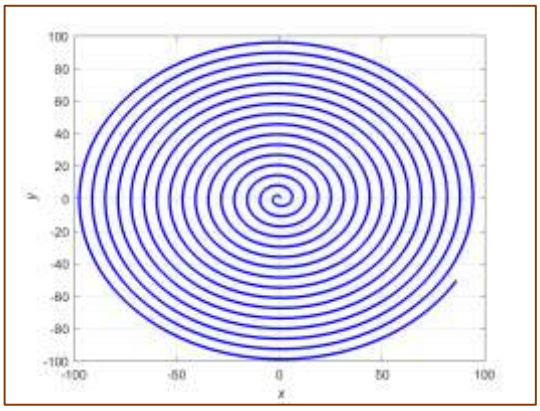


Obr. 6 Výstup grafu 03.06 do souboru

```
%% Program 03.07
   clear, close, clc
   t = -1:.01:1;
   x = t.*exp(t); y = t.^3 + 10*t;
   plot(x,y,'b','LineWidth',2),grid
   xlabel('{\it x}'), ylabel('{\it y}'

%% Program 3.08
   clear, close, clc
   t = 0:.01:100;
   x = t.*cos(t); y = t.*sin(t);
   plot(x,y,'b','LineWidth',2),grid
   xlabel('{\it x}'), ylabel('{\it y}')
   print Fig0308 -dpng
```



Obr. 7 Graf př. 03.07



Obr. 8 Graf př. 03.08

```
%% Program 3.09
%  vrh šikmý
   clear, close, clc
   alpha = input('zadej úhel (stupně): ');
   y0 = input('zadej výšku (m): ');
   v0 = input('zadej počáteční rychlost (m/s): ');
   g = 9.81; % m/s^2
   x0 = 0;
```

```
alpha = alpha/180*pi; % radiány
t = 0:0.01:3; % čas (s)
x = x0 + v0*t*cos(alpha);
y = -g*t.^2/2 + v0*t*sin(alpha) + y0;
y(y<0) = 0;
plot(x,y)
xlabel('{\it x} (m)'), ylabel('{\it y} (m)')
title('Vrh šikmý')
[ymax,imax] = max(y);
disp('Maximální výška: '), ymax
disp('Čas dosažení: '), t(imax)
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

ALGORITMIZACE IV

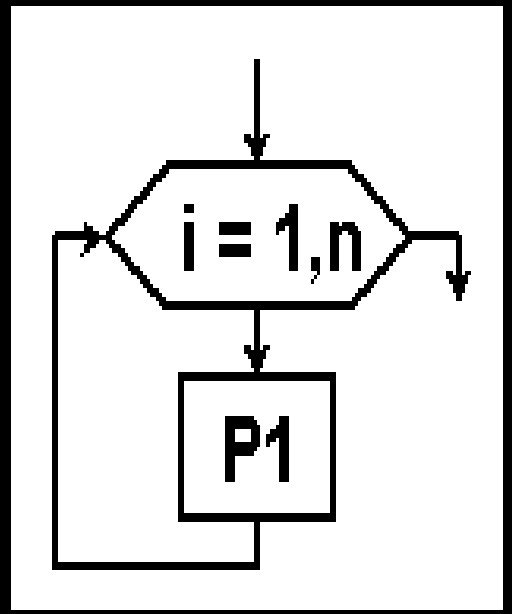
...aneb cykly

CYKLUS

- Opakování určité sekvence příkazů
- Buď je na počátku známo, kolikrát se bude tato sekvence opakovat: pak se jedná o cyklus s konečným, přesně daným počtem opakování
- Nebo je počet cyklů řízen nějakou podmínkou
  - Ta může být vyhodnocena na počátku, v tom případě cyklus nemusí proběhnout ani jednou
  - Nebo je vyhodnocena na konci, v tom případě cyklus proběhne aspoň 1x

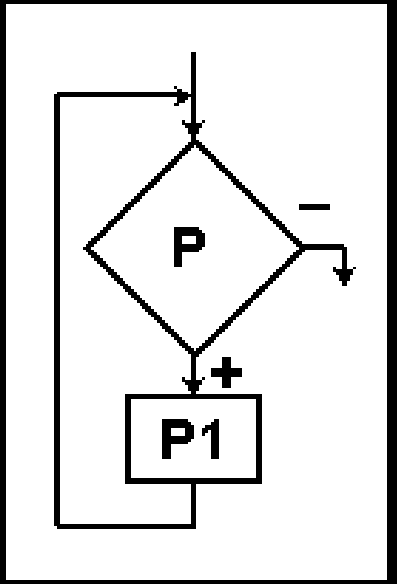
S KONEČNÝM POČTEM OPAKOVÁNÍ

- Prováděj sekvenci příkazů právě n-krát
- V programovacích jazycích typicky cyklus typu „for“
- Obecně cyklus proběhne tolikrát, kolik je rozdíl mezi konečnou a počáteční hodnotou celočíselného počítadla
- Pokud by byla počáteční hodnota počítadla vyšší, než konečná, cyklus neproběhne



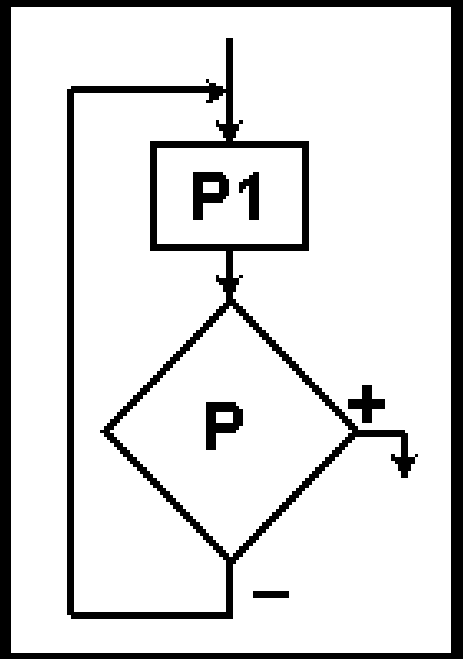
S PODMÍNKOU NA POČÁTKU

- Dokud platí podmínka P, prováděj sekvenci příkazů
- Cyklus nemusí proběhnout ani jednou v případě, že podmínka není splněna
- V programovacích jazycích typicky cyklus typu „while“
- Př.
  Dokud máš žízeň, dej si pivo
  (počet vypitých piv je úměrný žízni, pokud žízeň není, nevypije subjekt žádné pivo)



S PODMÍNKOU NA KONCI

- Proveď sekvenci příkazů
- Zjisti, zda už není sIněna ukončovací podmínka
- Pokud ne, vrať se a prováděj sekvenci příkazů znovu
- V programovacích jazycích typicky cyklus typu „repeat“
- Př.
  Dej si pivo
  Pokud máš stále žízeň, dej si další
- (počet vypitých piv je úměrný žízni, počítá se s tím, že subjekt vypije aspoň 1 pivo)



PŘÍKLADY

- Sestavte vývojové diagramy pro
  - A) výpočet faktoriálu zadaného čísla n
  - B) hledání kořene nelineární rovnice Newtonovou metodou (podmínkou bude dosažení dané přesnosti řešení)
  - C) načtení určitého počtu kladných čísel z klávesnice a uložení do indexované proměnné a (poslední hodnotou bude -1)

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 4 – UŽIVATELEM DEFINOVANÉ FUNKCE

- **Funkce** – definována jako pojmenovatelná posloupnost příkazů (porovnej s matematikou)
- Syntaxe: function [vystupní_proměnne]=jméno_funkce(vstupní_proměnné)
  jméno – výstižný a jednoznačný, pod tímto názvem se funkce ukládá jako soubor (s příponou *.m) V editoru je možné využít příkaz New Function
  vstupní_proměnné – seznam vstupních parametrů, v kulatých závorkách, použije se čárka jako oddělovač
  výstupní_proměnné – seznam výstupních parametrů, v hranatých závorkách, opět čárka jako oddělovač
  seznamy parametrů mohou být prázdné nebo proměnné délky
- komentář – začíná znakem **%** a končí s koncem řádku, nejen pro osobní sklerózu, ale pomůže těm, kdo budou Vaše funkce používat po Vás, aby vůbec pochopili, co jste měli na mysli
  program bez komentářů je jako velbloud bez hrbů, nedá se na něj dlouho dívat
  první komentářový řádek – podstatné informace – hledání pomocí **lookfor**
  druhý komentářový řádek – obecný popis komunikace s funkcí
  další komentářové řádky – význam všech parametrů funkce a další informace
- následuje posloupnost příkazů funkce
- musí obsahovat alespoň 1 příkaz přiřazení něčeho do výstupních proměnných
- koncové **end** je nepovinné
- dekompozice – rozklad systému na podsystémy (užitečný)
- modularita – složitou funkci vhodně dekomponujeme na několik jednoduchých funkcí
- Heslo kapitoly: Kdo nekrokuje s námi, krokuje proti sobě.
  Poznámka 1: Pokud funkce musí fungovat, ale přesto nefunguje, krokuj.
  Poznámka 2: Pokud funkce vypadá jako zcela nefunkční, ale přesto funguje, krokuj.
  Poznámka 3: Editor, debugger (Debug), F10, F11, F12, průzkum pomocí myši.
- Podfunkce – definují se za vlastní funkcí, fungují pak pouze v jejím rámci
- Příkaz **feval** – se použije v případě, že chci vyčíslit existující funkci, ale neznám předem její název
- Nové verze MATLAB upřednostňují tzv. anonymní funkce pro jednodušší funkce a podporují jejich vykreslení pomocí příkazu **fplot**. Za zavináč se dávají vstupní proměnné, funkce se potom zadává ve tvaru fce = 0. Příkaz fplot defaultně vykresluje funkci na intervalu [-5 5], pokud chce uživatel jinak, zadává si rozsah jako parametr

Příklad 04.01:

Je dána modelová funkce pro výpočet c_p ideálního plynu (J/kmol/K)

$$c_P = \mathrm{R}\left[\frac{7}{2} + \frac{x^2 \exp x}{(\exp x - 1)^2}\right]; x = \frac{\mathrm{h}\nu}{\mathrm{k}T}$$

kde R je univerzální plynová konstanta (8314,3 J/kmol/K),
h Planckova konstanta (0,66262 10^{-33} J/s),
k Boltzmannova konstanta (1,3806 10^{-23} J/K),
T teplota (K), ν frekvence vibrací pro danou látku (8,67 10^{13} Hz pro HCl)

```
function [Cp]=Fce04_01(ni,t)
%  Molární teplo Cp pro dvouatomovou molekulu
%  podle statistické termodynamiky
%  [Cp]=Fce04_01(ni,t);
%  Cp ... molární teplo při konstantním tlaku (J/kmol/K)
%  ni ... frekvence vibrací (8.67e13 Hz pro HCl)
%  t .... teplota (st.C) resp. vektor teplot

%% Konstanty
   R = 8314.3;      % plynová konstanta (J/kmol/K)
   H = 0.66262e-33; % Planckova konstanta (J/s)
   k = 1.3806e-23;  % Boltzmanova konstanta (J/K)
   T = t+273.15;    % absolutní teplota
   x = h*ni/k./T;   % pomocná proměnná

   Cp = 7/2*R + R*x.^2.*exp(x)./(exp(x)-1).^2;
end
```

Volání funkce proběhne v dialogovém nebo programovém režimu s konkrétními parametry, zde např. pro teplotu 25 °C

```
>> [Cp] = Fce04_01(8.67e13,25)
Cp =   2.9101e+04
```

Příklad 04.01a:

Tatáž úloha řešená pomocí anonymní funkce přímo v programu. V tomto případě to není úplně šikovné, kvůli pomocné proměnné a kvůli zadávání frekvence vibrací z klávesnice

```
%% Program 04.01
   clear, close, clc
   t = input('zadej teplotu (C): ');
   ni = input('zadej frekvenci vibrací: ');
%% Konstanty
   R = 8314.3;      % plynová konstanta (J/kmol/K)
   h = 0.66262e-33; % Planckova konstanta (J/s)
   k = 1.3806e-23;  % Boltzmanova konstanta (J/K)
   T = t+273.15;    % absolutní teplota
   x = h*ni/k./T;   % pomocná proměnná, předpoklad vektoru
   Cp = @(x) 7/2*R+R*x.^2.*exp(x)./(exp(x)-1).^2;
   yT = Cp(x)
```

Příklady pro samostatné řešení

Příklad 04.02:

Proveďte volání předchozí funkce pro rozsah teplot od 20 do 100 °C s krokem 2 °C, výsledek graficky zobrazte

Příklad 04.03:

Vytvořte funkci pro výpočet Antoineovy rovnice a testujte její správné fungování pro čtyři organické látky v rozsahu teplot 20 do 200 °C. Krok volte sami.

$\log p = A - \frac{B}{C+t}$ (°C, kPa)

		A	B	C
Methan	CH_4	5,820510	405,42	267,777
Ethan	C_2H_6	5,959420	663,70	256,470
Propan	C_3H_8	5,928880	803,81	246,990
Butan	C_4H_{10}	5,933860	935,86	238,73

Příklad 04.04:

Vytvořte funkci pro výpočet numerické integrace lichoběžníkovou metodou s pevným krokem h. Otestujte její funkčnost na výpočtu integrálu z funkce $y = \exp x$ pro $x \in \langle 0; 1 \rangle$

Příklad 04.05:

Závislost přijatého molárního tepla na teplotě je dána rovnicí

$$Q_{pm} = \int_{T_1}^{T_2} \left(a + bT + cT^2 + \frac{d}{T^2} \right) \mathrm{d}T$$

kde Q_{pm} je přijaté teplo na 1 mol látky [J mol^{-1}], T absolutní teplota [K], a, b, c, d empirické konstanty, charakteristické pro danou látku. Vypočtěte teplo přijaté oxidem dusným při ohřevu ze 100 °C na 1200 °C. Použijte funkci z předchozího příkladu. Hodnoty konstant jsou: a = 7,681; b = 1,44.10^{-3}; c = 2,53.10^{-6}; d = -0,951.10^{5}. Integrovanou funkci znázorněte grafem.

Příklad 04.06:

Vytvořte funkci pro výpočet povrchu a objemu kvádru. Vypočtěte, kolik m^3 vody je v plaveckém bazénu o rozměrech 15 x 15 x 2 metry, pokud je naplněn z 90 %? A kolik vody bude v zahradním bazénku o rozměrech 2 x 0,5 x 0,3 m, naplněném z poloviny?

Příklad 04.07:

Vytvořte funkci pro výpočet BMI, jestliže víte, že $BMI = \frac{m}{v^2}$ (kg, m^2). Spočtěte svůj BMI.

Příklad 04.08:

Vytvořte funkci, která spočte obsah, objem, velikost stranové a tělesové úhlopříčky pro krychli o zadané velikosti strany. V jejím rámci vytvořte podfunkci pro výpočet přepony trojúhelníku podle Pythagorovy věty.

Příklad 04.09:

Modelový způsob vyčíslení funkce pomocí **feval** pro funkci $\sin x$ v bodě 0. Zápisy jsou ekvivalentní

```
y = sin(0)
y = feval(@sin,0)
y = feval('sin',0)
nazev='sin'; y = feval(nazev,0)
fhand=@sin; y = feval(fhand,0)
```

Příklad 04.10:

Vytvořte funkci, která úhel zadaný ve stupních, minutách a vteřinách (vstup) převede na radiány (výstup)

Příklad 04.11:

Vytvořte funkci, která vrátí zadaný počet setříděných nejmenších hodnot vektoru náhodných čísel. Využijte vestavěné funkce **sort** a **rand**.

```
x = rand(1,10)
[vektorOut] Fce04_11(x,5)

function [vektorOut] = Fce04_11(vektorIn,n)
% [vektorOut] = Fce04_11(vektorIn,n)
%  vektorOut ... výstup
%  vektorIn ... vstup, generovaná čísla
%  n ... kolik jich chci vrátit
%  tVektor ... setříděný vektor vzestupně

   tVektor = sort(vektorIn,"ascend");
   vektorOut = tVektor(1:n);
end
```

Příklad 04.12:

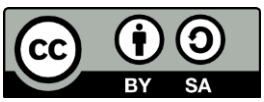
Vytvořte funkci, která vypočte střední odchylku naměřených a ideálních hodnot a zaokrouhlí ji na 2 desetinná místa. Testujte na posloupnosti $y_{id} = 1{:}10$; $y = y_{id}$ + náhodné malé číslo.

$$e = \frac{1}{n}\sum_{i=1}^{n}(y - y_{id})^2$$

Řešení úloh 4

```
%% Program 04.02
   clear, close, clc
   t = 20:2:100;
   ni = 8.67e13;
   [Cp] = Fce04_01(ni,t);
   plot(t,Cp,'r','LineWidth',2)
   grid, xlabel('{\it t} (C)')
   ylabel('{\it C}_P (J/kmol/K)')

%% Program 04.03
%  Výpočet Antoineovy rovnice
   clear, close, clc

%% Zadání dat metan etan propan butan
   ABC = [5.820510, 405.42, 267.777;...
          5.959420, 663.70, 256.470;...
          5.928880, 803.81, 246.990;...
          5.933860, 935.86, 238.73];
   t = 25:200;
   logP1 = Fce04_03(t,ABC(1,:)); logP2 = Fce04_03(t,ABC(2,:));
   logP3 = Fce04_03(t,ABC(3,:)); logP4 = Fce04_03(t,ABC(4,:));

   figure
   plot(t,logP1,'r-','LineWidth',1.5), hold on
   plot(t,logP2,'b-.','LineWidth',1.5)
   plot(t,logP3,'m:','LineWidth',1.5)
   plot(t,logP4,'c-','LineWidth',1.5)
   grid
   legend('metan', 'etan', 'propan', 'butan','Location','Best')
   xlabel('{\it t} (C)'), ylabel(' log {\it P}')
   title('Antoineova rovnice')
   hold off

function [logP] = Fce04_03(t,ABC)
%  Výpočet Antoineovy rovnice
%  [logP] = Fce04_03(t,ABC)
%  logP ... logaritmus tlaku
%  t ... teplota (C)
%  A, B, C ... specifické látkové koeficienty
   A = ABC(1);   B = ABC(2);   C = ABC(3);
   logP = A-B./(C+t);
end

%% Program 04.04
   clear, close, clc
   h = 0.01; t = 0:h:1; y = exp(t);
   integral = Fce04_04(h,y)

function [integral] = Fce04_04(h,y)
% Funkce pro výpočet integrálu lichoběžníkem s pevným krokem
% [integral] = Fce04_04(x,y)
% integral ... výsledný integrál
% y ... závislá proměnná
% h ... krok integrace
  integral = 2*sum(y(2:(end-1)));
  integral = integral + y(1) + y(end);
  integral = integral*h/2;
end
```

```
%% Program 04.05
   clear, close, clc
%% data
   a = 7.681; b = 1.44e-3; c = 2.53e-6; d = -0.951e5;
   t1 = 100; t2 = 1200;
   h = (t2-t1)/1000;
   T = t1:h:t2; T = T+273.15;
   cp = a + b*T + c*T.^2 + d./T.^2;
   plot(T,cp)
   xlabel('{\it T} K'), ylabel('\it Cp')
   axis('tight')
   Q = Fce04_05(h,cp)

function [integral] = Fce04_05(h,y)
% fce definovaná přímo pod skriptem
  integral = 2*sum(y(2:(end-1)));
  integral = integral + y(1) + y(end);
  integral = integral*h/2;
end

%% Program 04.06
%  Vypočtěte, kolik m3 vody je v bazénu o rozměrech 15 x 15 x 2 metry,
%  pokud je naplněn z 90 %

   clear, close, clc
   a = 15; b = 15; c = 2;        % plavecký bazén
   V1 = Fce04_06(a,b,c)*.9       % v m3

   a = 2; b = 0.5; c = 0.3;      % zahradní bazének
   V2 = Fce04_06(a,b,c)*.5

function [V,S] = Fce04_06(a,b,c)
% výpočet objemu a povrchu kvádru
% [V,S] = Fce04_06(a,b,c)
% V ... objem
% S ... povrch
% a, b, c ... rozměry
  V = a*b*c;
  S = 2*(a*b+b*c+a*c);
end


function [BMI] = Fce04_07(m,v)
% výpočet BMI
% [BMI] = Fce04_07(m,v)
% BMI ... body mass index
% m ... hmotnost (kg)
% v ... výška (m)
  BMI = m/v^2;
end

%% Program 04.08
   clear, close, clc
%  data
   a = input('zadej velikost strany krychle: '); % velikost strany krychle
   [V,S,us,ut] = Fce04_08a(a)
```



```
function [V,S,us,ut] = Fce04_08a(a)
% Funkce na výpočet vlastností krychle
% [V,S,us,ut] = Fce04_08(a)
% V ... objem
% S ... povrch
% us, ut ... úhlopříčky stranové a tělesové
% a ... strana krychle
  V = a^3; S = 6*a^2;
  us = Fce04_08b(a,a);
  ut = Fce04_08b(a,us);
end

function [c] = Fce04_08b(a,b)
% Funkce na výpočet přepony dle Pythagora
  c = sqrt(a^2+b^2);
end % konec podfunkce

function [uhelr] = Fce04_10(uhels, uhelm, uhelv)
% Funkce na přepočet úhlu zadaného
% ve stupních, minutách a vteřinách
% na radiány
% [uhelr] = Fce04_10(uhels, uhelm, uhelv)
% uhelr ... úhel v radiánech
% uhels, uhelm, uhelv ... stupně, minuty, vteřiny
  uhelr = uhels + uhelm/60 + uhelv/3600;
  uhelr = uhelr*pi/180;
end

function e = Fce04_12 (y0,y)
% Funkce na výpočet střední odchylky
% [e] = Fce04_12(y0, y)
% y0 ... ideální hodnoty
% y  ... naměřené hodnoty
% e  ... chyba
    e = sum((y0-y).^2)/length(y);
    e = round(e,2); % zaokrouhlení
end
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

ALGORITMIZACE V

...aneb podprogramy

O CO JDE?

- Používají se v případě, že vytváříme složitější úlohu
- Úloha se rozdělí na jednotlivé části – **dekompozice**
- Části se realizují samostatně
- Př. Cesta do práce
    1. Dojdi na metro
    2. Jeď metrem
    3. Dojdi od metra ke škole
    4. Vstup do budovy
    5. Jdi od vchodu do kanceláře

JEDNOTLIVÉ ČÁSTI A)

1. Dojdi na metro
   a) Vyjdi z bytu
   b) Zamkni dveře
   c) Sejdi po schodech
   d) Vyjdi z domu
   e) Jdi na stanici metra

2. Jeď metrem
   a) Vstup do stanice
   b) Sjeď po eskalátoru
   c) Čekej na nástupišti do centra
   d) Nastup do vlaku
   e) Dojeď na Dejvickou
   f) Vystup ze soupravy
   g) Dojeď eskalátorem k východu

JEDNOTLIVÉ ČÁSTI B)

3. Dojdi od metra ke škole
   a) Vyjdi ze stanice
   b) Jdi k budově VŠCHT A

4. Vstup do budovy
   a) Vyndej kartu
   b) Přilož kartu ke snímači
   c) Otevři dveře
   d) Vstup
   e) Vystoupej po schodech
   f) Přilož kartu ke snímači
   g) Projdi turniketem

- Jednotlivé kroky příkladu by se daly dekomponovat dále
- Př.
  2e) Dojeď na Dejvickou
  může být složitější pro toho, kdo cestou přestupuje
- Některé kroky v algoritmu se opakují
- Př.
  4b) a 4f) Přilož kartu ke snímači – jde o stejnou činnost

PODPROGRAMY

- Výhody:
  - Ušetříme čas v případě, že se vykonává tatáž činnost s různými parametry
  - Př. Přilož kartu ke snímači
  - Př. Výpočet binomického koeficientu potřebuje několik faktoriálů
    - Tj. bylo by dobré mít samostatně vyřešený výpočet faktoriálu a volat ho různými parametry

CO PODPROGRAM POTŘEBUJE?

- Vstupní a výstupní parametry
- Výstupním parametrům se také říká návratová informace
- Může existovat podprogram bez vstupních parametrů (typicky definice nějakého objektu)
- Podprogram musí mít vždy výstup (může být v různé podobě) – pokud ne, byla vytvořena černá díra!!! ☺
- Parametry se většinou předávají přes hlavičku, ale lze pracovat i tzv. globálními proměnnými
- V MATLAB typicky struktury uvedené klíčovým slovem „function“

PŘÍKLADY

- Vytvořte algoritmy podprogramů pro:
  - Výpočet faktoriálu
  - Přehození pozice dvou čísel v indexované proměnné

- S použitím výše uvedených vytvořte algoritmy pro
  - Výpočet binomického čísla
  - Setřídění posloupnosti čísle uložených v indexované proměnné

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 5 – ROZHODOVÁNÍ

- Rozhodovací funkce slouží k tomu, aby se provedla jistá množina příkazů v případě splnění nějaké podmínky
- Příklad:
  když se naučíš programovat
  dostaneš zápočet,
  dostaneš kredity,
  postoupíš do dalšího semestru,
  …..doma budou mít radost,
  konec
- **Syntaxe úplné podmínky**:
  if podmínka
  % příkazy pozitivní části
  else
  % příkazy negativní části
  end
  Syntaxe neúplné podmínky:
  if podmínka
  % příkazy pozitivní části
  end
  Syntaxe úplné vícenásobné podmínky:
  if podmínka
  % příkazy splnění podmínky 1
  elseif
  % příkazy splnění podmínky 2, lze přidávat další a další
  else
  % příkazy, které se vykonají, pokud neplatí žádná z podmínek
  end

- Rozhodování a větvení lze do sebe vnořit (ale nesmí dojít ke křížení)
- Matlab umí rozhodovat „skrytě“, tj. obsahuje příkazy, které v sobě již podmínku mají; např. funkce **min** de facto porovnává velikost jednotlivých prvků vektoru, dokud nenajde nejmenší
- Operátory logické viz přednáška 2
- Vícenásobné rozhodování je možné pomocí příkazu **switch,** používá se také k tvorbě položek menu
- **Syntaxe**
  switch *přepínač*
  case *výraz 1*
  % příkazy pro výraz 1

case *výraz 2*
% příkazy pro výraz 2
otherwise % když nic z toho neplatí
% příkazy
end

- Příkazy, kterými lze při rozhodování a ošetření běhu programu pomoci
  Komplexněji u vstupů dat
  vstup dat pomocí boxu: proměnná = **inputdlg**("text"),
  př. a= = inputdlg("zadej proměnnou a: ")
  text varování, že je něco v nepořádku, **warning**("text"),
  př. warning("špatný datový typ")
  jiná možnost **warndlg**("text")- rozdíl zjistíte sami ☺
  chyba (a po ní by měl být asi výpočet zastaven) **error**("text")
  př. **error**("soubor s daty nenalezen")
  jiná možnost **errordlg**("text") - rozdíl zjistíte sami ☺
  př. **errordlg**("soubor s daty nenalezen")
  zpráva do vyskakovacího okna: **msgbox**("text")
  př. msgbox("výpočet hotov")

Příklad 05.01:

Modelový příklad. Vytvořte funkci, která spočte kořeny kvadratické rovnice.

```
function [x] = Fce05_01(a,b,c)
% Modelová funkce
% Funkce na výpočet kořenů kvadratické rovnice
% x ... kořeny
% a, b, c ... koeficienty rovnice
  if a == 0
      disp('lineární rovnice');
      x = -c/b;
  else
      disp('kvadratická a MATLAB umí vše');
      D = b^2-4*a*c;
      x(1) = (-b+sqrt(D))/2/a;
      x(2) = (-b-sqrt(D))/2/a;
      % netřeba rozhodovat detailněji
      % Matlab umí počítat s komplexními čísly
      % ale x musí být vektor
  end
end
```

- Vytvoření jednoduchého rozhodovacího menu:
  volba = menu('název menu ','položka 1', 'položka 2', 'položka n');
  jméno volba se použije jako proměnná pro switch

Příklady pro samostatné řešení

Příklad 05.02:

Sestavte funkci na výpočet řešení lineární rovnice. Funkci řádně otestujte. Funguje i pro nulové a komplexní hodnoty parametrů?

Příklad 05.03:

Sestavte funkci na úplnou diskusi řešení kvadratické rovnice. Využijte funkci z předchozího příkladu v případě, že $a = 0$. Funkci otestujte pro všechny možné případy.

Příklad 05.04:

Sestavte funkci na testování, zda je číslo sudé či liché

Příklad 05.05:

Sestavte program, který vás v závislosti na zadané hodině pozdraví. Ošetřete i případy, že bude zadána hodina mimo povolený rozsah:

Hodina	Pozdrav
6 až 9	Dobré ráno
9 až 12	Pěkné dopoledne
12 až 13	Dobré poledne
13 až 18	Dobré odpoledne
18 až 21	Dobrý večer
21 až 6	Proč nespíš?

Příklad 05.06:

Sestavte funkci na výpočet podílu sinu x a uvedeného x. Instrukce: chování v okolí nuly je také důležité, Taylorova řada, dodefinování funkce v okolí nuly, krokování.

Příklad 05.07:

Sestavte funkci pro posouzení sestrojitelnosti trojúhelníka.

Příklad 05.08:

Vytvořte funkci na výpočet plochy trojúhelníka pomocí Heronova vzorce. Využijte modifikaci předchozí funkce

$$S = \sqrt{s(s-a)(s-b)(s-c)};\quad s = \frac{a+b+c}{2}$$

Příklad 05.09:

Sestavte funkci na určení polohy minima ze 3 prvků. Pozn. Jde to chytře!

Příklad 05.10:

Otestujte, zda je číslo kladné, záporné nebo nula. Využijte příkaz switch.

Příklad 05.11:

Vytvořte program, který navrhne menu a podle jeho obsahu menu vykreslí sinus nebo kosinus t na intervalu $\langle 0; 2\pi \rangle$, případně se ukončí

Příklad 05.12:

Vytvořte funkci, která zjistí, zda jsou všechny prvky zadané matice kladné. Pokuste se vyřešit úlohu s co nejmenším počtem příkazů IF (stačí jeden)

Příklad 05.13:

Sestavte program pro výpočet jarního úplňku a data Velikonoc. Vstupem bude daný rok. Instrukce k výpočtu

a) Výpočet gregoriánské epakty (stáří fáze Měsíce na počátku roku):
   Vydělte rok 19 a zapište zbytek po dělení, k němu přičtěte 1, výsledek je **Zlaté číslo**
b) Zlaté číslo vynásobte 11, výsledek dělte 30 a zapište zbytek po dělení, výsledkem je **juliánská epakta**
c) Výpočet jednotlivých oprav pro gregoriánskou epaktu: Rok dělte 100 a přičtěte 1, výsledek nazveme století,
d) Od století odečtěte 16, násobte 3 a pak dělte 4, zapište si celou část podílu, což je **Sluneční oprava**
e) Od století odečtěte 15, násobte 8 a dělte 25, zapište si celou část podílu, což je **Měsíční oprava**
f) Od juliánské epakty odečtěte 10 a Sluneční opravu, přičtěte Měsíční opravu
g) Výsledné číslo musí být v intervalu 0-29, je-li výsledek menší než 0, přičtěte 30; je-li větší než 29, odečtěte 30. Získáte je **gregoriánskou epaktu**
h) Ke gregoriánské epaktě je třeba přiřadit datum cyklického úplňku (Paschal Full Moon, zkratka PFM)
   - pro epaktu 0 - 23 platí: PFM = 44 – epakta
   - pro epaktu 24 platí: PFM = 49
   - pro epaktu 25 a Zlaté číslo < 12 platí: PFM = 49
   - pro epaktu 25 a Zlaté číslo >= 12 platí: PFM = 48
   - pro epaktu 26 - 29 platí: PFM = 74 - epakta
i) Je-li číslo PFM menší než 32, cyklický úplněk nastává v **březnu** a číslo ukazuje den v měsíci. Je-li PFM větší než 31, odečtete od něj 31 a dostanete den v **dubnu**, kdy nastává cyklický úplněk
j) Zjištění, ve kterém dni týdne je cyklický úplněk:
k) K číslu **10** přičtěte **Sluneční opravu**, výsledek je **gregoriánská oprava** (rozdíl ve dnech mezi oběma kalendáři)
l) K roku přičtěte rok/4, odečtěte gregoriánskou opravu a přičtěte PFM
m) Výsledek dělte 7, celou část zbytku po dělení (0…neděle, 1…pondělí, atd.), získáte den v týdnu, kdy úplněk nastane
n) Velikonoční neděle je první nedělí po PFM

Příklad 05.14:

Sestavte funkci, která vypočtěte chybějící hodnotu do rovnice ideálního plynu. Hodnoty jsou zadávány jako třísloupcový vektor x s pevným pořadím P, V, T. Předpokládejte, že data jsou zadána v jednotkách SI. Dopočte tu z hodnot, která není definovaná (je tedy zadána jako NaN), použijte funkci **isnan**. Otestujte, že zadávaná data jsou v požadovaném rozsahu (P: 10^5 až 3.10^5 Pa; V: 1 až 10 m^3; T: 300 až 400 K). Zobrazte dopočtenou hodnotu na 2 desetinná místa s adekvátními fyzikálními rozměry, můžete použít funkci **sprintf** . Př. `sprintf('%0.2f',X)`

Řešení úloh 5

```
function [x] = Fce05_02(a,b)
% Řešení lineární rce a*x+b=0
% [x]=Fce05_02(a,b);
% x ... vektor řešení
% a ... lineární člen
% b ... absolutní člen
   if a == 0 % degenerovaná
      if b ~= 0
         x = []; % nemá řešení
      else
         x = inf; % má nekonečně mnoho řešení
      end
   else
      x = -b/a; % právě jedno řešení
   end
end

function [x] = Fce05_03(a,b,c)
% Funkce na výpočet kořenů kvadratické rovnice
% [x] = Fce05_03(a,b,c)
% x ... kořeny
% a, b, c ... koeficienty rovnice
   if a == 0
         disp('lineární rovnice')
         [x]=Fce05_02(a,b)
   else
         D = b^2-4*a*c;
         if D == 0
             disp('1 reálný dvojnásobný kořen')
         elseif D > 0
             disp('2 reálné kořeny')
         else
             disp('2 komplexní kořeny')
         end
         x = [(-b+sqrt(D))/2/a (-b-sqrt(D))/2/a];
   end
end

function Fce05_04(a)
% Test sudosti čísla, bez výstupní proměnné
% Fce05_04(a)
% a .. testované číslo
  if mod(a,2) == 0
      disp('sudé');
  else
      disp('liché');
  end
end

%% Program 05.05
   clear, close, clc
   a = input('Zadej hodinu: ');
   if (a <= 0) | (a > 24)
       disp('Neumíš hodiny? To jsou kladná čísla od 0 do 24')
   elseif (a <= 6) | (a > 22)
       disp('Mazej spát!')
   elseif (a <= 9)
        disp('Dobré ráno!')
```

```
   elseif (a <= 12)
       disp('Pěkné dopoledne!')
   elseif (a <= 13)
       disp('Dobré poledne!')
   elseif (a <= 18)
       disp('Dobré odpoledne!')
   else
       disp('Dobrý večer!')
   end

function z = Fce05_06(x)
%  z = Fce05_06(x)
%  sin x/x
   if x == 0
       z = 1;
   else
       z = sin(x)/x;
   end
end

function l = Fce05_07(a,b,c)
%  Fce05_07(a,b,c)
%  Posuzuje sestrojitelnost trojúhelníka
%  a, b, c ... strany
  if (a <= 0) | (b <= 0) | (c <= 0)
      disp('Trojúhelník nelze sestrojit ')
      l = 0;
  elseif ((a+b) > c) & ((a+c) > b) & ((b+c) > a)
      disp('Trojúhelník lze sestrojit ')
      l = 1;
   else
      disp('Trojúhelník nelze sestrojit ')
      l = 0;
   end
end

function [S] = Fce05_08(a,b,c)
% Plocha trojúhelníka dle Herona
% [S] = Fce05_08(a,b,c)
% S ... plocha
% a, b, c ... strany
  S = [];
  l = Fce05_07(a,b,c); % testuje se sestrojitelnost
  if l == 0
     return
  end
  s = (a+b+c)/2;
  S = sqrt(s*(s-a)*(s-b)*(s-c));
end

function [poloha] = Fce05_09(a,b,c)
% Poloha minima 3 prvků s chytrým využitím vestavěných fcí
  [m,poloha] = min([a b c]);
end

function Fce05_10(x)
% Fce05_10(x) test čísla
  n = sign(x);
  switch n
      case -1
```

```
            disp('číslo je záporné')
        case 1
            disp('číslo je kladné')
        otherwise
            disp('číslo je nula')
    end
end

%% Program 05.11
    clear, close, clc
%   definice menu
    volba = menu('Volba funkce','sinus', 'cosinus', 'konec');
    t = 0: pi/100 :2*pi;
    switch (volba)
        case 1 % sin
            y = sin(t);
        case 2 % cos
            y = cos(t);
        otherwise
           disp('Konec'); return;
    end
    plot(t,y,'LineWidth',2); grid
    xlabel('\it x'), ylabel('\it y')
    axis("tight")

%% Program 05.12
    clear, close, clc
    A = [1 2 3; 4 5 -7]; % matice
    Fce05_12(A)

function Fce05_12(A)
% Fce05_12(A)
% testuje, zda jsou prvky matice kladné
    B = (A > 0);
    if sum(B(:)) == length(A(:))
            disp('všechny prvky matice jsou kladné ')
    else
            disp('všechny prvky matice nejsou kladné ')
    end
end


%% Program 05.13
%   Výpočet data Velikonoc
    clear, close, clc
    R = input('zadej rok: ');
%% Juliánská epakta a opravy
    Z = mod(R,19) + 1;        % zlaté číslo, zbytek po dělení 19 +1
%   juliánská epakta, 11*zlaté číslo a zbytek po dělení 30
    Z1 = 11*Z; J = mod(Z1,30);
    S = fix(R/100)+1;       % století
    Z2 = fix((S-16)*3/4);   % sluneční oprava, celá část podílu
    Z3 = fix((S-15)*8/25); % měsíční oprava, celá část podílu
    G = J - 10 - Z2 + Z3;   % gregoriánská epakta v rozmezí 0-29
%   juliánská -10 + sluneční – měsíční
    if G < 0
        G = G + 30;
    elseif G > 29
        G = G-30;
    end;
```

```
%% Datum jarního úplňku (PFM)
   disp('První jarní úplněk: ')
   if (G > 0) & (G <= 23)
       PFM = 44-G;
   elseif (G == 24) |((G == 25) & (Z < 12))
       PFM = 49;
   elseif (G == 25) & (Z >= 12)
       PFM = 48;
   else
       PFM = 74-G;
   end

   if (PFM < 32)
       s1 = num2str(PFM); s2='. březen';
   else
       s1 = num2str(PFM-31); s2='. duben';
   end
   [s1 s2] % PFM nastane v tento den

%% Den v týdnu, kdy nastane PFM
   Z4 = 10 + Z2; % gregoriánská oprava
   Den = fix(mod((R + R/4 - Z4 + PFM),7));
   switch Den
       case 0
           disp('neděle'), Velikonoce = str2num(s1) + 7;
       case 1
           disp('pondělí'), Velikonoce = str2num(s1) + 6;
       case 2
           disp('úterý'), Velikonoce = str2num(s1) + 5;
       case 3
           disp('středa'), Velikonoce = str2num(s1) + 4;
       case 4
           disp('čtvrtek'), Velikonoce = str2num(s1) + 3;
       case 5
           disp('pátek'), Velikonoce = str2num(s1) + 2;
      otherwise
           disp('sobota'), Velikonoce = str2num(s1) + 1;
   end;

%% Velikonoční neděle
   if Velikonoce > 31
       Velikonoce = Velikonoce-31;
       s2='. duben';
   end
   disp('Velikonoce')
   [num2str(Velikonoce) s2]


%% AAP05 14
%  ideal gas
   clear, close, clc
    x = [233424.06 NaN 300.02]; % test
    y = Fce05_14(x)

function y = Fce05_14(x)
  P = x(1);
  V = x(2);
  T = x(3);
  R = 8.314;
```

```
  n = 100;
  y=' ';
  p1 = (V >= 1) & (V <= 10) & (T >= 300) & (T <= 400);
  p2 = (P >= 1e5) & (P <= 3e5) & (T >= 300) & (T <= 400);
  p3 = (P >= 1e5) & (P <= 3e5) & (V >= 1) & (V <= 10);
  if isnan(P)
      if p1
            P = n*R*T/V;
            y = [sprintf('%0.2f',P) ' Pa'];
       end
  elseif isnan(V)
     if p2
         V = n*R*T/P;
         y = [sprintf('%0.2f',V) ' m^3'];
      end
  else % isnan(T)
       if p3
          T = P*V/R/n;
          y = [sprintf('%0.2',T) ' K'];
       end
   end
end
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 6 – CYKLY

- Bez cyklu – každý příkaz se provede nejvýše jednou
  S cyklem – každý příkaz uvnitř cyklu se provede nejvýše N-krát (v případě programátorské katastrofy je N rovno nekonečnu)
- Důsledek pro programátora: Nechápu-li možnosti cyklu, pak doba potřebná k vytvoření programu je nejen vyšší než doba výpočtu, ale i vyšší než užitek z něj.
- Cyklus má svůj začátek, cyklus má svůj konec , cyklus má své tělo (libovolné příkazy mezi začátkem a koncem)
- Příkaz **return** – v těle větvení můžeme realizovat předčasný únik z funkce. (Existují i jiné příkazy pro předčasné ukončení cyklu, ale kazí programátorský styl.)
- Cyklus s pevným počtem opakování – **for** cyklus;
  cyklus s proměnným počtem opakování – **while** cyklus
- Obecná syntaxe for cyklu:

```
for proměnná = vektor
    % tělo_cyklu
end
```

- Poznámky k for cyklu: proměnnou, který řídí počet průchodů cyklem lze použít jako počítadlo (index)
- Obecná syntaxe while cyklu:

```
while logická_podmínka
  % tělo_cyklu
end
```

- Poznámky k while cyklu – pokud není podmínka splněna, nemusí proběhnout vůbec (a je nutno si na to dát pozor)
  cyklus se typicky používá tam, kde je nutno dosáhnout nějaké přesnosti výpočtu
- Matlab, žel, **neumí** počítat cyklus pro indexy vektorů (matic) od nuly. Je nutno se s tím smířit.
- Příklady cyklu ze života:

```
zalez_do_postele;
while unaven
   zdrimni_si;
end
vylez_z_postele;
```

- Příklad větvení uvnitř cyklu:

```
jdi_do_prace;
while cas < 16.00
   if nekdo_jde_kolem
      predstirej_praci;
```

```
   else
      zdrimni_si;
   end
end
jdi_domu;
```

- Příklad for cyklu ze života:

```
for n = 1:10
   dej_si_pivo;
end
```

- Matlab umí cykly schovat do předdefinovaných funkcí

Příklady pro samostatné řešení

Příklad 06.01:

Jistě znáte omšelou historku z 19. století, jak neposlušný žák Gauss dostal za trest spočítat součet všech čísel od 1 do 100 a on to měl hned hotové. Sestavte funkci pro výpočet součtu prvních n přirozených čísel. Instrukce: Krokujte a místo proměnné k si představte počet čárek na tácku. Použijte for cyklus.

Příklad 06.02:

Ještě jednou totéž, pro změnu s while cyklem

Příklad 06.03:

Do třetice s vestavěnou funkcí Matlabu

Příklad 06.04:

Sestavte funkci pro výpočet faktoriálu malého nezáporného celého čísla (testujte). Srovnejte s vestavěnou funkcí **factorial**.

Příklad 06.05:

Harmonická posloupnost je tvořena převrácenými hodnotami přirozených čísel. Sestavte funkci pro výpočet součtu prvních n členů harmonické posloupnosti.

Příklad 06.06:

Pyramida je stavěna z kostek a bez dutin. Výška pyramidy je H. Základnou pyramidy je čtverec o straně H. Sestavte funkci pro spotřebu kostek na její stavbu. Vypočtěte objem materiálu k její stavbě, jestliže kostka má velikost strany a. Vypočtěte počet kostek nutný k postavení duté pyramidy a objem materiálu.

Příklad 06.07:

Vytvořte funkci pro realizaci Newtonovy metody pro řešení rovnice $f(x) = 0$ za předpokladu, že derivace je dána analyticky jako druhá funkce. (Jde o nespolehlivou metodu, ale druhého řádu)

Příklad 06.08:

Vytvořte funkci pro realizaci metody sečen pro řešení rovnice $f(x) = 0$ za předpokladu, že derivace je dána analyticky jako druhá funkce. (Jde o nespolehlivou, ale superlineární metodu řádu 1,618)

Příklad 06.09:

Vytvořte funkci pro řešení rovnice $f(x) = 0$ metodou půlení intervalu za předpokladu, že funkce je spojitá a v krajních bodech intervalu mění znaménko. (Jde o spolehlivou, ale pomalou metodu prvního řádu)

Příklad 06.10:

Vytvořte funkci pro výpočet první a druhé derivace libovolné matematické funkce. Vzorce jsou následující (tříbodové metody). Funkčnost ověřte na výpočtu derivací známých funkcí. Výsledky znázorněte graficky.

$$y_1' = \frac{-3y_1+4y_2-y_3}{2h};\ y_i' = \frac{y_{i+1}-y_{i-1}}{2h};\ y_n' = \frac{3y_n-4y_{n-1}+y_{n-2}}{2h}$$

$$y_1'' = \frac{y_1-2y_2+y_3}{h^2};\ y_i'' = \frac{y_{i-1}-2y_i+y_{i+1}}{h^2};\ y_n'' = \frac{y_{n-2}-2y_{n-1}+y_n}{h^2}$$

Příklad 06.11:

Vypočtěte hodnotu funkce $\ln(1+x)$ jako součet nekonečné řady se zadanou přesností. Pro výpočet vytvořte funkci, jejímž vstupem bude zadané číslo x a požadovaná přesnost.

$$\ln(x+1) = \sum_{i=1}^{\infty} (-1)^i \frac{x^{i+1}}{i+1}$$

Příklad 06.12:

Vytvořte funkci pro numerickou integraci, uvažujte, že krok integrace nemusí být konstantní. Vstupem funkce budou hodnoty x, y; výstupem hodnota integrálu.

Příklad 06.13:

Vytvořte program pro výpočet a znázornění impulsní a přechodové funkce koncentrace v kaskádě ideálních mísičů, jestliže n je počet stupňů kaskády (10), τ je střední doba prodlení částic v kaskádě (400 s), doba výpočtu $t \in \langle 0;\ 2\,\tau \rangle$

$$\mathrm{g}(t) = \frac{1}{(n-1)!}\left(\frac{nt}{\tau}\right)^{n-1}\frac{n}{\tau}\exp\left(-\frac{nt}{\tau}\right)$$

$$\mathrm{h}(t) = 1 - \exp\left(-\frac{nt}{\tau}\right)\sum_{k=0}^{n-1}\frac{1}{k!}\left(\frac{nt}{\tau}\right)^k$$

Příklad 06.14:

Sestavte funkci pro výpočet binomického čísla (vstup n, k). Využijte již hotových poznatků z funkce pro výpočet faktoriálu.

Příklad 06.15:

Jsou dány kinetické konstanty μ_{max} (hod^{-1}) = 0,45; K_S ($g.l^{-1}$) = 7,2; K_I ($g.l^{-1}$) = 12,5 a rovnice $\mu = \mu_{\max}\frac{S}{S+K_s}$, $\mu_I = \mu_{\max}\frac{S}{S+K_s+\frac{S^2}{K_I}}$.

Pro hodnoty S v intervalu $\langle 0;\ 250 \rangle$ hodin napište program pro výpočet a zobrazení μ a μ_I bez použití a s použitím cyklu. Porovnejte si pro sebe výhody a nevýhody obou metod. Upgradujete postup s použitím funkcí pro μ a μ_I.

Řešení úloh 6

```
function [s] = Fce06_01(n)
%  Spočte for cyklem sumu n čísel
%  [s] = Fce06_01(n)
   s = 0; % soucet začíná od 0
   for i = 1:n % dělám to nx, pokud je n =1, neproběhne
       s = s + i; % přičtu další číslo
   end
end

function [s] = Fce06_02(n)
%  Spočte while cyklem sumu n čísel
%  [s] = Fce06_02(n)
   k = 1; % první člen součtu
   s = 0; % soucet začíná od 0
   while k <= n    % mám ještě přičítat?
      s = s + k;   % přičtu
      k = k + 1;   % zvětším na další člen
   end
end

function [s] = Fce06_03(n)
% Spočte sumu n čísel
% [s] = Fce06_03(n)
   s = sum(1:n);
end

function [f] = Fce06_04(n)
% Spočte faktoriál čísla n
% [f] = Fce06_04(n)
   f = [];
   if n >= 0
      f = 1;
      for i = 1:n
          f = f*i;
      end
   else
       disp('záporné, faktoriál nemá smysl')
       return
   end
end

function [h] = Fce06_05(n)
% Součet prvních n členů harmonické posloupnosti
% [h]=Fce06_05(n)
% h ... hodnota harmonického součtu
% n ... pocet členů řady
  h = 0; % součet
  for k =1 :n
      h = h + 1/k; % pričítám 1,1/2,1/3...
  end % konec cyklu
end
```

```
function [pocet] = Fce06_06(H)
% Počet kostek pyramidy
% [pocet]=Fce06_06(H)
% pocet ... výsledný počet kostek
% H ... počet pater pyramidy
  pocet = 0;
  for i = 1:H
      pocet = pocet + i^2;
  end
end

function [x] = Fce06_07(jmenof,jmenod,a,b,x0,nmax,epsx,epsf)
% Obecne řešení rovnice f(x)=0 Newtonovou metodou
% [x]=Fce06_07(jmenof,jmenod,a,b,x0,nmax,epsx,epsf);
% x ... řešení rovnice
% jmenof ... název funkce f(x) jako řetězec znaků
% jmenod ... název funkce f'(x) jako řetězec znaků
% a  ... dolní mez definičního oboru f(x)
% b  ... horní mez definičního oboru f(x)
% x0 ... počáteční odhad řešení
% nmax ... maxcimální počet vyčíslení fce f(x)
% epsx ... postačující šířka intervalu <a;b>
% epsf ... postačující hodnota funkce f(x)
  if a >= b   % je dán interval správně?
     x = inf; % smůla!
     return;
  end
  for k = 1:nmax % do vyčerpání trpělivosti, ochrana proti nekonečnému cyklu
      if (x0 < a) | (x0 > b) % mimo definiční obor?
         x = inf;    % smůla!
         return;
      end
      f = feval(jmenof,x0); % vyčíslení funkce
      if abs(f)<=epsf       % nejsem už u cíle?
         x = x0;            % hurá!!!
         return;
      end
      d = feval(jmenod,x0); % vyčíslení derivace
      if d == 0             % dělím, musím se zeptat
         x = inf;           % riziko Newtonovy metody
         return;
      end
      x = x0 - f/d;         % vlastní vzorec
      if abs(x-x0) <= epsx  % jak daleko jsou x od sebe?
         return;
      end
      x0 = x; % škatule, hejbejte se a jedem znovu od začátku
   end
end

function [x] = Fce06_08(jmenof,a,b,nmax,epsx,epsf)
% Obecne řešení rovnice f(x)=0 metodou sečen
% [x]=Fce06_08(jmenof,a,b,nmax,epsx,epsf);
% x ... řešení rovnice
% jmenof ... název funkce f(x) jako řetězec znaků
% a  ... dolní mez definičního oboru f(x)
% b  ... horní mez definičního oboru f(x)
% nmax ... maxcimální počet vyčíslení fce f(x)
% epsx ... postačující šířka intervalu <a;b>
```

```
% epsf ... postačující hodnota funkce f(x)
  if a >= b   % je dán interval správně?
     x = inf; % smůla!
     return;
  end
  x1 = a; x2 = b;
  f1 = feval(jmenof,x1); % levý okraj
  if abs(f1) <= epsf     % není to už ono?
     x = x1;             % hurá!!!
     return;
  end
  f2 = feval(jmenof,x2); % pravý okraj
  if abs(f2) <= epsf     % není to už ono?
     x = x2;             % hurá!!!
     return;
  end
  if f1*f2 > 0
     x = inf;            % chybně, jsem mimo
     return;
  end
  for k = 3:nmax         % do vyčerpání trpělivosti
      x = (x1*f2-x2*f1)/(f2-f1); % vlastní metoda
      if (x<a) | (x>b)           % mimo definiční obor?
          x = inf;               % riziko metody
          return;
      end
      f = feval(jmenof,x);       % další bod
      if abs(f) <= epsf          % už je to ono?
         return;
      end
      if abs(x-x2) <= epsx       % jak daleko jsou od sebe?
         return;
      end
      x1 = x2; f1 = f2; % škatule, hejbejte se a jedem znovu od začátku
      x2 = x; f2 = f;
   end
end

function [x] = Fce06_09(jmenof,a,b)
% Obecné řešení rovnice f(x)=0 metodou půlení intervalu
% [x]=Fce06_09(jmenof,a,b);
% x ... řešení rovnice
% jmenof ... název funkce f(x) jako řetězec znaků
% a  ... dolní mez intervalu
% b  ... horní mez intervalu
% nmax ... maximální počet vyčíslení fce f(x)
  nmax = 40;
  fa = feval(jmenof,a); % levý okraj
  fb = feval(jmenof,b); % pravý okraj
  if fa*fb>0
     x = inf;  % jsem mimo
     return;
  end
  for k = 3:nmax        % ochrana proti nekonečnému cyklu
      x = (a+b)/2;      % střed intervalu
      f = feval(jmenof,x);
      if fa*f <= 0      % je to v intervalu <a;x>?
         b = x;         % zahození intervalu (x;b>
      else
```

```
        a = x;          % zahození intervalu <a;x)
      end
  end
  x = (a+b)/2;          % naposled
end

%% Program 06.10
   clear, close, clc
   h = 0.01;    % krok výpočtu
   t = 0:h:2;   % vektor nezávislé proměnné
   y = exp(-t); % téměř chronicky známá funkce
   [dy, d2y] = Fce06_10(h,y);
   figure
   plot(t,y,'r','LineWidth',1.5), hold on
   plot(t,dy,'b','LineWidth',1.5)
   plot(t,d2y,'c:','LineWidth',1.5)
   grid,   xlabel('{\it t}')
   ylabel('{\it y}, d{\it y}, d^2{\it y}')
   title("Funkce a její derivace numericky")
   hold off

function [dy, d2y] = Fce06_10(h,y)
% Funkce pro výpočet numerické derivace
% [dy,d2y] = Fce06_10(h,y)
%  dy, d2y ... vektor derivací
%  y ... vektor funkce
%  h ... krok
%% První derivace
   n = length(y);
   dy(1) = (-3*y(1)+4*y(2)-y(3))/(2*h);
   for i = 2:n-1
       dy(i)=(y(i+1)-y(i-1))/(2*h);
   end;
   dy(n) = (3*y(n)-4*y(n-1)+y(n-2))/(2*h);
%% Druhá derivace
   d2y(1) = (y(1)-2*y(2)+y(3))/(h^2);
   for i = 2:n-1
          d2y(i) = (y(i-1)-2*y(i)+y(i+1))/(h^2);
   end;
   d2y(n) = (y(n)-2*y(n-1)+y(n-2))/(h^2);
end

function [y] = Fce06_11(x, tol)
% Funkce na výpočet součtu nekonečné řady
% [y] = Fce06_11(x, presnost)
% y ... výsledek
% x ... z čeho počítám
% tol ... zadaná přesnost
  y = x;       % na počátku rovnou 1. hodnotu
  clen = x;    % toto bude další člen Taylorova rozvoje
  nmax = 1000; % odhad, že tolik kroků mi bude stačit
  i = 1;
  while abs(clen) > tol
      clen = ((-1)^(i+1))*x^i/i;
      y = y + clen;  i = i + 1;
      if i > nmax
         disp('vyčerpáno nmax'), return
      end
  end
end
```

```
%% Program 06.12
   clear, close, clc
   h = 0.01;    % krok výpočtu
   t = 0:h:2;   % vektor nezávislé proměnné
   y = exp(-t); % téměř chronicky známá funkce
   S = Fce06_12(t,y)

function S = Fce06_12(x,y)
% S = Fce06_12(x,y)
% výpočet integrálu ichoběžníkem s nestejným krokem
  S = 0;
  N = length(x); % počet kroků
  for i = 1:N-1
      S = S + (y(i+1)+y(i))*(x(i+1)-x(i));
  end
  S = S/2;
end

%% Program 06.13
   clear, close, clc
%  Data
   n = 10; % počet stupňů kaskády
   tau = 400;
%% Analytické řešení kaskády
   t = [0:tau/50:2*tau];
   g= 1/factorial(n-1).*(n*t/tau).^(n-1).*n/tau.*exp(-n*t/tau);
   suma = 0;
   for k = 0:(n-1)
       suma = suma + (n*t./tau).^k/factorial(k);
   end
   h = 1-exp(-n*t./tau).*suma;
   tiledlayout(2,1)
   nexttile, plot(t,g,'r-','LineWidth',1), grid
   xlabel('{\it t} (s)'), ylabel('g({\it t})')
   nexttile, plot(t,h,'b-','LineWidth',1), grid
   xlabel('{\it t} (s)'), ylabel('h({\it t})')

function [bin] = Fce06_14(n,k)
% [bin] = Fce06_14(n,k)
% Výpočet binomického čísla
% n,k ... zadání
% bin ... binomický koeficient
  if k > n
      error('n musí být větší než k')
      return;
  end
  bin = factorial(n)/factorial(n-k)/factorial(k);
end


%% AAP06_15
   clear, close, clc
   mi_max = 0.45; % hod-1
   Ks = 7.2; Kl = 12.5; % g/l
   S =  0:250;
%  a) bez cyklů
   mi = mi_max*S./(Ks+S);
   mil= mi_max*S./(Ks+S+S.^2/Kl);
   subplot(211)
   plot(S,mi), hold on
   plot(S,mil), grid
```

```
    xlabel('\it S'), ylabel('\it \mu')
    hold off
%   b) cyklem
    for i=1:length(S)
        mi(i) = mi_max*S(i)/(Ks+S(i));
        mil(i) = mi_max*S(i)/(Ks+S(i)+S(i)^2/Kl);
    end
    subplot(212)
    plot(S,mi), hold on
    plot(S,mil), grid
    xlabel('\it S'), ylabel('\it \mu')
    hold off
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 7 – SYMBOLICKÁ MATEMATIKA

- Přehled všech funkcí v **help symbolic**
- Ve vyšších verzích se někde změnila syntaxe oproti nižším
- Základní definice symbolických proměnných **syms proměnné_oddělené_mezerou** (čárka není povolena!)
- Alternativně voláním funkce **sym('proměnná'),** tj. `x = sym('x')`
- Symbolická matematika zná běžné funkce, zaokrouhlování
- Náhrada symbolických proměnných za hodnotu či jinou proměnnou pomocí funkce **subs**
```
syms x y
>> f = x^2 -2*x + 1
>> subs(f,1)
ans = 0
```
- Převod symbolických proměnných na číselné pomocí příkazu **double**
```
a1 = sym(1/2+1/4+1/6) % symb. proměnná 11/12
>> a2 = double(a1)
```
- Lepší znázornění výsledku pomocí funkce **pretty(výraz)**
- Zjednodušení výsledků funkcí **simplify(výraz)** – přirozeně jen tehdy, když je to možné (nelze na to spoléhat, občas zjednodušení, které vidíme, zkrátka nenajde)
- Funkce **collect(výraz, proměnná)** sloučí části obsahující danou proměnnou
- Funkce **expand(výraz)** roznásobí mnohočleny
- Symbolické funkce lze nově vykreslit pomocí příkazu **ezplot(funkce)**
```
syms x; ezplot(cos(x))
```
`ezplot(funkce , [a b])` ... graf funkce pro nezávisle proměnnou z intervalu $[a;b]$.
`ezplot(x,y)` ... graf parametricky zadané křivky $x = x(t), y = y(t)$ pro t z intervalu $[0;2\pi]$;
`ezplot(x,y,[a b])` ... graf parametricky zadané křivky $x = x(t), y = y(t)$ pro t z intervalu $[a;b]$.
- 3D grafy budou podobné jako ostatní, jen mají předponu „ez“ (např. **ezplot3**)
- Řešení rovnic příkaz **solve**
```
syms x; xx = solve(x^2+5*x+6==0)
```
- Symbolická integrace **int**
```
int(exp(x))
```
- Symbolická derivace **diff**
```
diff(exp(-x))
```
`diff(výraz,prom,řád)` … derivace vyššího řádu
- Limita funkce **limit(výraz)**
- Součet řady symsum(výraz,počet)
Taylorův rozvoj taylor(funkce)
- Řešení diferenciálních rovnic pomocí **dsolve**
```
r = dsolve('rce1,rce2,...', 'x1,x2,...', 'proměnná')
```
Změnila se syntaxe zadávání rovnic! Výstupní proměnnou pro řešení ODE je třeba zadat jako

funkci času, tedy **syms proměnná(t)**. Pro derivace se již neužívá samostatné označení, ale počítají se pomocí funkce **diff**. Pokud chceme zachovat systém z předchozích verzí, je třeba zavést substituce

Pozn.: Symbolický toolbox prochází podobně jako samotný MATLAB vývojem, do verze 3.2.3 bylo používáno jádro MAPLE (existoval také rozšířený symbolický toolbox – extended symbolic math toolbox). Od r. 2008 vstoupila v platnost nová smlouva, podle které je výpočetní jádro MAPLE v prodeji exkluzivně u firmy Maplesoft. Proto je symbolický toolbox od verze 4.9 (MATLAB R2007b+) postavený na jádru MuPAD. Změny byly poměrně velké. Zpočátku toolbox obsahoval hodně chyb. V současnosti je k dispozici Symbolic Math Toolbox ve verzi 5.5 či vyšší, kde je opět mnoho změn oproti předchozí verzi. Proto je potřeba brát v potaz, v jaké verzi MATLAB se pracuje a podle toho uzpůsobit výklad, případně odkaz na **help**.

Příklady pro samostatné řešení

Příklad 07.01:

Pomocí symbolické matematiky definujte funkci pro průběh tlaku pro H_2O_2 podle Antoineovy rovnice. Dosaďte hodnoty pro rozsah teplot 20 až 500 °C, s krokem 5°C (viz **Příklad 02.01**), vykreslete graf.

$\log p = A - \frac{B}{C+t}$ (°C, kPa), A = 5,817; B = 1264,74; C = 171,561

Příklad 07.02:

Vypočtěte průběh rychlosti v (m.s^{-1}) pro časový interval $\langle 0; 5 \rangle$ min s krokem 0,1 minuty (viz **Příklad 02.02**). Rychlost je určena vztahem

$$v = \frac{t^3}{\sqrt[4]{1+3t^4}}$$

Spočtěte dráhu pomocí symbolické integrace

Příklad 07.03:

Pomocí symbolické matematiky řešte soustavu rovnic (viz **Příklad 02.03**)

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 6 \\ 4x_1 + 5x_2 + 6x_3 &= 15 \\ 7x_1 + 8x_2 + x_3 &= 16 \end{aligned}$$

Příklad 07.04:

Pomocí symbolické matematiky řešte rovnici $x^2+5x+6 = 0$.

Příklad 07.05:

Vypočtěte 1., 2. a 3. derivaci funkce $f(x) = (5x^5 e^x)/250$

Příklad 07.06:

Pomocí symbolické matematiky řešte soustavu rovnic $x^2+xy+y = 3$, $x^2-4x+3 = 0$

Příklad 07.07:

Pomocí symbolické matematiky řešte diferenciální rovnici $y' = -5x$ s počáteční podmínkou $y(0) = 1$.

Příklad 07.08:

Pomocí symbolické matematiky řešte nelineární diferenciální rovnici $(y')^2+y^2 = 1$, s počáteční podmínkou $y(0) = 0$

Příklad 07.09:

Pomocí symbolické matematiky řešte diferenciální rovnici 2. řádu $y'' = -5\,y$, s počátečními podmínkami $y(0) = 1$, $y'(\pi/5) = 0$

Příklad 07.10:

Řešení ODE vyššího řádu se zavedením substitucí (modelový příklad):

$y'' = -y$, s počátečními podmínkami $y(0) = 1, y'(0) = 0$

```
%% Program 07.10
%  Modelový příklad řešení ODE vyššího řádu se substitucemi
%  tak, jak bylo běžné v předchozích verzích toolboxu
   clear, close,clc
   syms y(t)
% zavádí se příslušné derivace
   Dy = diff(y);
   D2y = diff(y,2);
   y(t) = dsolve(D2y == -y, y(0) == 1, Dy(0) == 0) % řešení rce

% alternativa
   syms y
   y = dsolve('D2y = -y', 'y(0) = 1', 'Dy(0) = 0') % řešení rce
```

Příklad 07.11:

Je dána kinetická rovnice rozkladu látky A: $\frac{\mathrm{d}c_\mathrm{A}}{\mathrm{d}t} = -kc_\mathrm{A}$ s počáteční podmínkou $c_{\mathrm{A}0}$. Řešte pomocí symbolické matematiky, jestliže $k = 10^{-2}$ s^{-1} a $c_{\mathrm{A}0} = 0{,}1$ mol/dm^3. Zobrazte výsledek pro časový rozsah $t \in \langle 0; 500 \rangle$ s

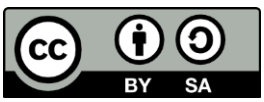
Příklad 07.12:

Je dána kinetická rovnice 2. řádu rozkladu látky A: $\frac{\mathrm{d}c_\mathrm{A}}{\mathrm{d}t} = -kc_\mathrm{A}^2$ s počáteční podmínkou $c_{\mathrm{A}0}$. Řešte pomocí symbolické matematiky, jestliže $k = 10^{-2}$ s^{-1} a $c_{\mathrm{A}0} = 0{,}1$ mol/dm^3. Zobrazte výsledek pro časový rozsah $t \in \langle 0; 5000 \rangle$ s

Řešení úloh 7

```
%% Program 07.01
%  Funkce symbolicky
   clear, close, clc
   syms t p
   A =  5.817; B = 1264.74; C = 171.561;
   p = A-B/(t+C);                                % def. logaritmu fce symbolicky
   p = 10^p;
   ezplot(p,[20,500])                             % vykreslení přímo
   figure
   tt = 20:5:500;  pp = double(subs(p,tt));       % dosazení
   plot(tt,pp), xlabel('\it x'), ylabel('\it y') % zobrazení


%% Program 07.02
%  Výpočet rychlosti a dráhy symbolicky
   clear, close, clc
   syms t v s
   v = t^3/((1+3*t^4)^(1/4));
   ezplot(t,v),                          % zobrazení symbolické fce
   s = int(v,t); pretty(s)               % integrace
   si = double(subs(s,300)-subs(s,0)) % určitý integrál


%% Příklad 07.03
%  Soustava lineárních rovnic symbolicky
   clear, close, clc
   syms x1 x2 x3
   [xx1,xx2,xx3] = solve(x1 + 2*x2 + 3*x3 == 6,...
                         4*x1 + 5*x2 + 6*x3 == 15,...
                         7*x1 + 8*x2 + x3 == 16,...
            x1, x2, x3)
   double([xx1, xx2, xx3])

%  alternativa (live script)
   eq1 = x1+2*x2+3*x3==6;
   eq2 = 4*x1+5*x2+6*x3==15;
   eq3 = 7*x1+8*x2+x3==16;
   eq = [eq1, eq2, eq3];
   vars = [x1, x2, x3];
   [A,B] = equationsToMatrix(eq,vars )
   X = linsolve(A,B)

%  alternativa (starší Matlab)
   [xx1,xx2,xx3] = solve('x1 + 2*x2 + 3*x3 = 6',...
                         '4*x1 + 5*x2 + 6*x3 = 15',...
                         '7*x1 + 8*x2 + x3 = 16',...
            x1, x2, x3);
   double([xx1, xx2, xx3])


%% Program 07.04
%  Nelineární rovnice symbolicky
   clear, close, clc
   syms x
   xx = solve(x^2 + 5*x + 6 == 0);
   double(xx)
%  alternativa live
   eq = x^2+5*x+6==0
   xx = solve(eq)
```

```
%  alternativa (starší Matlab)
   xx = solve('2*x^2 + 5*x + 6 = 0')


%% Program 07.05
%  Symbolický výpočet derivací
   clear, close, clc
   syms x f
   f = (5*x^5*exp(x))/250;
   f1 = diff(f,x); simplify (f1)
   f2 = diff(f,x,2); simplify (f2)
   f3 = diff(f,x,3); simplify (f3)


%% Program 07.06
%  Řešení soustavy nelineárních rovnic
   clear, close, clc
   syms x y
   [x,y] = solve(x^2 + x*y + y == 3, ...
                 x^2 - 4*x + 3 == 0)

%  alternativa live
    eq1 = x^2 + x*y + y == 3;
    eq2 = x^2 - 4*x + 3 == 0;
    eq =[eq1,eq2];
    [x,y] = solve(eq, [x,y])

%  alternativa
    [x,y] = solve('x^2 + x*y + y = 3',...
                  'x^2 - 4*x + 3 = 0')


%% Program 07.07
%  Řešení ODE 1. řádu s počáteční podmínkou symbolicky
   clear, close, clc
   syms y(t)
   y = dsolve(diff(y) == -5*y, y(0) == 1)
   ezplot(y, [0 10]), grid % vykreslení

% alternativa live script
   eq = diff(y,x) == -5*x
   cond = y(0) == 1
   y = dsolve(eq,cond)
   ezplot(y, [0 10]), grid % vykreslení

%  alternativa
   syms y x
   yy = dsolve('Dy =-5*x','y(0) = 1')
   ezplot(yy, [0 10]), grid % vykreslení



%% Program 07.08
%  ODE nelineární
   clear, close, clc
   syms y(t)
   eqn = diff(y,t)^2 + y^2==1;
   cond = y(0)==0;
   y = dsolve(eqn,cond)
   pretty(simplify(y))
```



```
% alternativa live script

%  alternativa
   syms y
   yy = dsolve('Dy = sqrt(1-y^2)', 'y(0) = 0')


%% Program 07.09
%  ODE 2. řád
   clear, close, clc
   syms y(t)
   yy = dsolve(diff(y,2) == -5*y, y(0) == 1, diff(y(pi/5)) == 0)
%  alternativa
   Dy = diff(y);
   D2y = diff(y,2);
   y(t) = dsolve(D2y == -5*y, y(0) == 1, Dy(pi/5) == 0)

%  alternativa
   syms y
   yy = dsolve('D2y = -5*y', 'y(0) = 1', 'Dy(pi/5) = 0')


%% Program 07.11
%  symbolické řešení kinetické rovnice 1. řádu
%  dcA/dt = -k cA
   clear, close, clc
   syms ca(t) k ca0 t
%  obecné symbolické řešení
   Dca = diff(ca,t);
   cca = dsolve(Dca == -k*ca, ca(0) == ca0)
%  dosazení konstant ze zadání
   k = 1e-2;
   ca0 = 0.1;
   cca = (dsolve(Dca == -k*ca, ca(0) == ca0))
%  vyčíslení pro konkrétní hodnoty
   tt = 0:500; cca1 = double(subs(cca,tt));
%  zobrazení
   ezplot(cca, tt)
   plot(tt,cca1,'LineWidth',1), grid
   xlabel('{\it t} (s)'),
   ylabel('{\it c}_A (mol/dm^3)')

%  alternativa
syms ca t
%  obecné symbolické řešení
   cca = dsolve('Dca = -1e-2*ca', 'ca(0) = 0.1')
%  vyčíslení pro konkrétní hodnoty
   tt = 0:.1:500;
   cca1 = double(subs(cca,tt));
%  zobrazení
   plot(tt,cca1,'LineWidth',1), grid
   xlabel('{\it t} (s)'),
   ylabel('{\it c}_A (mol/dm^3)')


%% Program 07.12
%  symbolické řešení kinetické rovnice 2. řádu
%  dcA/dt = -k cA^2
   clear, close , clc
   syms ca(t) k ca0 t
```

```
   Dca = diff(ca);
%  obecné řešení
   cca = dsolve(Dca == -k*ca^2, ca(0) == ca0)
%  dosazení konstant ze zadání
   k = 1e-2;
   ca0 = 0.1;
   cca = dsolve(Dca == -k*ca^2, ca(0) == ca0)
%  vyčíslení pro konkrétní hodnoty
   tt = 0:5000;
   cca1 = double(subs(cca,tt));
%  zobrazení
   plot(tt,cca1,'LineWidth',1), grid
   xlabel('{\it t} (s)')
   ylabel('{\it c}_A (dm^3 mol^{-1} s^{-1})')

%  alternativa
   syms ca t
   cca = dsolve('Dca = -1e-2*ca^2', 'ca(0) = 0.1')
%  vyčíslení pro konkrétní hodnoty
   tt = 0:5000;
   cca1 = double(subs(cca,tt));
%  zobrazení
   plot(tt,cca1,'LineWidth',1), grid
   xlabel('{\it t} (s)')
   ylabel('{\it c}_A (dm^3 mol^{-1} s^{-1})')
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 8 – EXPORT A IMPORT DAT, APROXIMACE POLYNOMEM

Práce s řetězci znaků

Nové verze MATLABu rozlišují mezi datovou položkou **char** a **string**. Charakter je jednotlivý znak, uzavřený v apostrofech, jejich pole je potom posloupnost, pokud se uvažuje jako jeden objekt, je to string, uzavřený v uvozovkách (zjednodušeně). Nižší verze pracovaly jen s typem char. Více `help strfun`

Funkce, použitelné pro práci s těmito typy:

Tab. 7 Funkce pro řetězce

Funkce	Popis	Funkce	Popis
string	Definice řetězce	char	Definice znaku
join	Kombinace stringů	blanks(n)	Přidej n mezer
plus	Přidej číslo	append	Kombinace
num2str	Konverze čísla na řežezec	str2num	Konverze řetězce na číslo
isletter	Testuje, zda se jedná o daný znak	isnumeric	Testuje, zda je znak číslo
double	Převod na ASCII (char)	strcat	Spojuje řetězce dané na řádku
strncmp	Porovná, zda jsou řetězce stejné	strvcat	Spojuje sloupec řetězců
findstr	Vyhledá kratší řetězec v delším	strrep	Nástroj pro náhradu jedné sekvence jinou
upper	Převod malých písmen na velká	lower	Převod velkých písmen na malá

Vstup a výstup programu

- Nejjednodušší zadání vstupu do programu je pomocí příkazu **input** s doprovodnou výzvou
  ```
  proměnná = input('text výzvy')
  ```
  Příklad
  ```
  a = input('zadej a: ');
  ```
  Pokud je potřeba zadat víceřádkovou výzvu, zlom řádku se řídí příkazem \n
  ```
  a = input('zadej prvek posloupnosti, \n který bude kladný: ');
  ```
- Nejjednodušší zobrazení výstupu z programu do pracovní plochy je pomocí příkazu bez středníku, doprovozeným komentářem pomocí příkazu **disp**
  ```
  disp('text komentáře')
  disp(proměnná)
  ```
- Sofistikovanější způsob vstupu je pomocí vstupního dialogového boxu a příkazu **inputdlg**, vstup je realizován jako textový řetězec, na data musí bát převeden
  ```
  proměnná = inputdlg('Text výzvy: ', 'Název dialogu', parametry)
  prevod = str2num(proměnná{1})
  ```
  Příklad – načítá data oddělená mezerou, max. délka 50 znaků
  ```
  answer = inputdlg('Zadej čísla oddělená mezerou:',...
  ```

```
                    'Data', [1 50])
user_val = str2num(answer{1})
```

Data ve formátu MATLAB

- Vstup a výstup pracovního prostoru MATLAB (nebo jeho části) se realizuje příkazy **save**, respektive **load.** Proměnné jsou uloženy do souboru zadaného názvu s příponou *.MAT.
  Syntaxe:
  `save filename`
  `save filename proměnné` (oddělené mezerou)
  `load filename`
  Příklad (uloží proměnnou a do souboru, po vymyzání paměti ji znovu načte)
  `save afile a, clear all, load afile`
- Alternativní syntaxe ve formě volání funkce
  `save(filename)`
  `save(filename, proměnné)`
  `load(filename)`
- Další možností uložení proměnných do souboru s příponou *.MAT je příkaz matfile. Využívá se více v objektovém programování. Syntaxe (konstruuje objekt, který je souborem):
  `matobjekt = matfile(filename)`
- Pomocí příkazu save a load lze vytvářet, respektive načítat i ASCII soubory, syntaxe je pak:
  `save -ascii filename proměnné`
  `load filename -ascii`

Data z textových souborů

- Data v textovém souboru (předpokládá se sloupcové uspořádání) se importují příkazem **dlmread**. Soubor musí existovat v adresáři, z něhož jej MATLAB volá (jinak by bylo nutné specifikovat cestu k souboru). Syntaxe:
  `data=dlmread('filename.přípona')`
- Zápis obdobně příkazem **dlmwrite**.

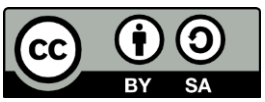
Data z tabulkových kalkulátorů

- Data z tabulkových kalkulátorů se načítají příkazy **readtable** (jakýkoli formát), **xlsread** (Excel) a ukládají příkazy **writetable**, **xlswrite**. Syntaxe příkazu pro načtení dat z Excelu:
  `Data = xlsread(filename, list, rozsah)`
  Příklad
  `Data = xlsread('mydata.xls', 3, 'RC1:RC2')`

Tab. 8 Podporované formáty importu a exportu dat

Typ souboru	Přípona	Poznámka	Import	Export
MATLAB	MAT	Z pracovního prostoru	`load`	`save`
		Vybrané proměnné	`matfile`	`matfile`
Text	Jakékoliv, včetně CSV, TXT	Čísla oddělená čárkou	`csvread`	`csvwrite`
		Čísla	`dlmread`	`dlmwrite`
		Čísla nebo i s textem	`textscan`	`none`
		Sloupce čísel (i s textem)	`readtable`	`writetable`
Tabulkový kalkulátor	XLS, XLSX, XLSM, XLSB XLTM, XLTX (jen import)	Tabulka Excel	`xlsread`	`xlswrite`
		Data v tabulce uspořádaná do sloupců	`readtable`	`writetable`

- Konečně, ve vyšších verzích MATLAB existují v hlavním menu (karta HOME) tlačítka **Import Data** a Save Workspace. Pokud jsou data jen v trochu čitelném formátu, dostanou se do pracovního prostoru i tudy



Aproximace dat polynomem

- Aproximace jednorozměrných dat polynomem je poměrně jednoduchou záležitostí. Existuje vestavěná funkce **polyfit**, která určí příslušné koeficienty metodou nejmenších čtverců, pomocí funkce **polyval** lze pak získat požadovaný průběh regresní křivky.
  Syntaxe:
  `vektor_koeficientů=polyfit(nezávislá,závislá,řád)`
  `závislá_aprox=polyval(vektor_koeficientů, nezávislá)`
- Matematicky – pro polynom 3. stupně (odvození)
  $y_v = a_3x^3 + a_2x^2 + a_1x + a_0$
  účelová funkce pro metodu nejmenších čtverců:
  $$S = \sum_{i=1}^{n}(y_i - y_{iv})^2 = \sum_{i=1}^{n}\left(y_i - a_3x_i^3 - a_2x_i^2 - a_1x_i - a_0\right)^2$$
- Je potřeba zajistit, aby parciální derivace podle všech hledaných koeficientů byly rovny nule
  $\frac{\partial S}{\partial a_3} = 0; \frac{\partial S}{\partial a_2} = 0; \frac{\partial S}{\partial a_1} = 0; \frac{\partial S}{\partial a_0} = 0$
- Výpočet parciálních derivací vede k soustavě rovnic (derivuje se složená funkce)
  $$\frac{\partial S}{\partial a_3} = 2\sum_{i=1}^{n}\left(y_i - a_3x_i^3 - a_2x_i^2 - a_1x_i - a_0\right)\ x_i^3$$
  $$\frac{\partial S}{\partial a_2} = 2\sum_{i=1}^{n}\left(y_i - a_3x_i^3 - a_2x_i^2 - a_1x_i - a_0\right)\ x_i^2$$
  $$\frac{\partial S}{\partial a_1} = 2\sum_{i=1}^{n}\left(y_i - a_3x_i^3 - a_2x_i^2 - a_1x_i - a_0\right)\ x_i$$
  $$\frac{\partial S}{\partial a_0} = 2\sum_{i=1}^{n}\left(y_i - a_3x_i^3 - a_2x_i^2 - a_1x_i - a_0\right)$$
- Soustava se dělí 2 a násobí se jednotlivými součty mocnin x
  $$\frac{\partial S}{\partial a_3} = \sum_{i=1}^{n}\left(x_i^3y_i - a_3x_i^6 - a_2x_i^5 - a_1x_i^4 - a_0x_i^3\right)$$
  $$\frac{\partial S}{\partial a_2} = \sum_{i=1}^{n}\left(x_i^2y_i - a_3x_i^5 - a_2x_i^4 - a_1x_i^3 - a_0x_i^2\right)$$
  $$\frac{\partial S}{\partial a_1} = \sum_{i=1}^{n}\left(x_i\ y_i - a_3x_i^4 - a_2x_i^3 - a_1x_i^2 - a_0x_i\right)$$
  $$\frac{\partial S}{\partial a_0} = \sum_{i=1}^{n}\left(y_i - a_3x_i^3 - a_2x_i^2 - a_1x_i - a_0\right)$$
- Dále se osamostatní jednotlivé součty a přejde se na maticový tvar
  $$\begin{bmatrix} \sum_{i=1}^{n}x_i^6 & \sum_{i=1}^{n}x_i^5 & \sum_{i=1}^{n}x_i^4 & \sum_{i=1}^{n}x_i^3 \\ \sum_{i=1}^{n}x_i^5 & \sum_{i=1}^{n}x_i^4 & \sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^2 \\ \sum_{i=1}^{n}x_i^4 & \sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i \\ \sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}x_i^3y_i \\ \sum_{i=1}^{n}x_i^2y_i \\ \sum_{i=1}^{n}x_i\ y_i \\ \sum_{i=1}^{n}y_i \end{bmatrix}$$

- Alternativní způsob řešení v novějších verzích MATLAB pomocí příkazu fit
  `koeficienty = fit(x,y,“polyX“),`
  kde X je řád polynomu. Pokud je potřeba získat i hodnotu koeficientu R^2, funkce se volá s dalším výstupem
  `[koeficienty, statistika] = fit(x,y,“polyX“),`
  a příslušný koeficient se pak získá jako parametr `statistika.rsquare`

Nalezení kořenů funkce

- Pro polynomickou funkci funguje příkaz **roots**. Syntaxe: `roots(C)`, kde `C` je vektor koeficientů polynomu
- Pro další funkce existuje příkaz **fzero**. Syntaxe:
  `řesení = fzero(@funkce, nástřel)`

Příklad 08.05:

(Modelový příklad) Jsou dána data

x = [0 0.2 0.5 0.7 0.8 1 1.2 1.6 1.9 2];

y = [5.2 4.3 4.4 4.9 5.5 6 6.2 8.4 8.9 10.9].

Zobrazte je v grafu modrými kolečky. Dále aproximujte průběh

a) polynomem 3. řádu pomocí funkcí polyfit a polyval;

b) řešením maticové rovnice $\mathbf{A}\,\mathbf{x} = \mathbf{b}$

c) rovnou vytvořením matice mocnin x a vektoru pravých stran $\mathbf{M}\,\mathbf{x} = \mathbf{y}$

```
%% Program 08.05
%  Aproximace funkcí lineární MNČ
   clear, close, clc

%% Data
   x = [0 0.2 0.5 0.7 0.8 1 1.2 1.6 1.9 2]';
   y = [5.2 4.3 4.4 4.9 5.5 6 6.2 8.4 8.9 10.9]';
   scatter(x,y), hold on
   grid, xlabel('\it x'), ylabel('\it y')

%% Vestavěná fce polyfit(x,y,řád)
   n = 3; a1 = polyfit(x,y,n) % koeficienty regresní křivky
%  pomocná nezávislá proměnná pro jemnější vykreslení
   xx = [min(x):(max(x)-min(x))/1000:max(x)];
   yy = polyval(a1,xx);
   plot(xx,yy,'r','LineWidth',1), hold off

%% Maticí
%  alokace matice a vektoru pravých stran
   A = zeros(n+1,n+1);
   b = zeros(n+1,1);
   aa = zeros(2*n+1,1);
   m = length(aa); % pomocný vektor pro sumy
%  vytvoření všech sum x
   for i = 1:m
       aa(i) = sum(x.^(2*n-i+1));
   end
%  obecné naplnění matice
   for i = 1:n+1
       p = i;
       for j=1:n+1
          A(i,j) = aa(p);
          p = p + 1;
       end
   end
 % naplnění vektoru b
   for i = 1:n+1
       b(i) = sum(x.^(n-i+1).*y);
   end
   a2 = A\b

%% Rovnou podle počtu koeficientů
%  Pokud by některý koeficient nebyl přítomen,
%  je třeba matici upravit a příslušný sloupec vypustit
   M = [];
   for i = 1:n+1
       M = [M x.^(n-i+1)];
   end
   a3 = M\y
```

```
%% Pomocí fit
   [a4, stst] = fit(x,y,"poly3");
   figure
   plot(a4,x,y); grid
   title(['R^2 ' num2str(stst.rsquare)]), stst.sse
```

Příklad 08.11:
(Modelový příklad)

```
%% AAP 08.11 hrátky se znaky a řetězci
   clear, close, clc
   a = char('abcd'); a(3)
   b = string("abcd"); b(1)
   double(a)
   strncmp(a,b,4)

   t1c = 'Matlab je ten'; t2c = 'nejlepsi nastroj'; t3c = 'na svete';
   t1s = "Matlab je ten"; t2s = "nejlepsi nastroj"; t3s = "na svete";

   Tc = [t1c t2c t3c]
   Tc = [t1c blanks(1) t2c blanks(1) t3c]

   Ts = t1s + blanks(1) + t2s+blanks(1) + t3s
   Ts1 = [t1s; t2s; t3s]
   strvcat(Ts1)
   findstr(t1c,"e") % určí pozice
```

Příklady pro samostatné řešení

Příklad 08.01:

Zadejte z klávesnice počet čísel a posloupnost čísel (s příslušnými výzvami) a spočtěte průměr těchto čísel.

Příklad 08.02:

Řešte předchozí příklad pomocí dialogového boxu (netřeba zadávat počet)

Příklad 08.03:

Načtěte data z textového souboru Data08_03.txt a zobrazte je v grafu jako modrá kolečka. Graf popište. Program si uložte k dalšímu použití.

Příklad 08.04:

Načtěte data ze souboru MS Excel Data08_04.xls. Data jsou umístěna na 2. listu v rozsahu sloupců B5:C365. Zobrazte data v grafu jako modré křížky o velikosti značky 1.5. Graf popište. Program si uložte k dalšímu použití.

Příklad 08.06:

Aproximujte data z příkladu 08.03 polynomem 2. řádu. Zobrazte do téhož grafu. Spočtěte sumu čtverců odchylek

Příklad 08.07:

Aproximujte data z příkladu 08.04 polynomem 1. řádu. Zobrazte do téhož grafu. Spočtěte sumu čtverců odchylek

Příklad 08.08:

Načtěte data ze souboru MS Excel data08_08.xls. Data jsou umístěna na 2. listu v rozsahu sloupců B3:C27. Aproximujte data funkcí $X = c_1 \exp(c_2\, t)$. Použijte metodu linearizace $\ln X = \ln c_1 + c_2\, t$. Vypočtete koeficienty, sumu čtverců odchylek a výsledek zobrazte do grafu.

Příklad 08.09:

Najděte kořeny rovnic pomocí funkce roots:

a) $x^6 + 2\,x^5 - x^4 - 3\,x^3 - 5\,x^2 - x - 1 = 0$
b) $x^4 - 7\,x^3 + 10\,x^2 - 20\,x - 8 = 0$

Příklad 08.10:

Najděte kořeny rovnic pomocí funkce fzero:

a) $e^{-x} - 0{,}2\,x = 0,\ x_0 = 2$
b) $\sin x^2 - 0{,}5 = 0,\ x_0 = 2$
c) $2\cos x + 0{,}5\,x = 0,\ x_0 = 0$

Analyzátor	
Čas (min)	X (V)
0	1,5400
1	1,5509
2	1,5619
3	1,5729
4	1,5840
5	1,5953
6	1,6065
7	1,6179
8	1,6294
9	1,6409
10	1,6525
11	1,6642
12	1,6760
13	1,6878
14	1,6998
15	1,7118
16	1,7239
17	1,7361
18	1,7484
19	1,7607
20	1,7732
21	1,7857
22	1,7984
23	1,8111
24	1,8239
25	1,8368
26	1,8498
27	1,8629
28	1,8761
29	1,8893
30	1,9027
31	1,9162
32	1,9297
33	1,9434
34	1,9571
35	1,9710
36	1,9849
37	1,9990
38	2,0131
39	2,0274
40	2,0417
41	2,0561
42	2,0707
43	2,0853
44	2,1001
45	2,1150
46	2,1299
47	2,1450
48	2,1602
49	2,1754
50	2,1908
51	2,2063

Gravimetricky	
Čas (min)	X (g/l)
0	1,89
59	2,66
125	3,95
190	6,51
240	7,98
320	13,24
360	15,46

52	2,2219
53	2,2377
54	2,2535
55	2,2694
56	2,2855
57	2,3017
58	2,3179
59	2,3343
60	2,3509
61	2,3675
62	2,3842
63	2,4011
64	2,4181
65	2,4352
66	2,4524
67	2,4698
68	2,4873
69	2,5049
70	2,5226
71	2,5404
72	2,5584
73	2,5765
74	2,5947
75	2,6131
76	2,6316
77	2,6502
78	2,6689
79	2,6878
80	2,7068
81	2,7260
82	2,7453
83	2,7647
84	2,7843
85	2,8040
86	2,8238
87	2,8438
88	2,8639
89	2,8842
90	2,9046
91	2,9251
92	2,9458
93	2,9666
94	2,9876
95	3,0088
96	3,0301
97	3,0515
98	3,0731
99	3,0948
100	3,1167
101	3,1388
102	3,1610
103	3,1833
104	3,2059
105	3,2285
106	3,2514
107	3,2744

108	3,2976
109	3,3209
110	3,3444
111	3,3680
112	3,3919
113	3,4159
114	3,4400
115	3,4644
116	3,4889
117	3,5136
118	3,5384
119	3,5635
120	3,5887
121	3,6141
122	3,6396
123	3,6654
124	3,6913
125	3,7174
126	3,7437
127	3,7702
128	3,7969
129	3,8238
130	3,8508
131	3,8780
132	3,9055
133	3,9331
134	3,9609
135	3,9890
136	4,0172
137	4,0456
138	4,0742
139	4,1031
140	4,1321
141	4,1613
142	4,1908
143	4,2204
144	4,2503
145	4,2803
146	4,3106
147	4,3411
148	4,3718
149	4,4028
150	4,4339
151	4,4653
152	4,4969
153	4,5287
154	4,5607
155	4,5930
156	4,6255
157	4,6582
158	4,6912
159	4,7244
160	4,7578
161	4,7914
162	4,8253
163	4,8595

164	4,8939
165	4,9285
166	4,9634
167	4,9985
168	5,0338
169	5,0695
170	5,1053
171	5,1414
172	5,1778
173	5,2144
174	5,2513
175	5,2885
176	5,3259
177	5,3636
178	5,4015
179	5,4397
180	5,4782
181	5,5170
182	5,5560
183	5,5953
184	5,6349
185	5,6748
186	5,7149
187	5,7554
188	5,7961
189	5,8371
190	5,8784
191	5,9200
192	5,9619
193	6,0040
194	6,0465
195	6,0893
196	6,1324
197	6,1758
198	6,2195
199	6,2635
200	6,3078
201	6,3524
202	6,3973
203	6,4426
204	6,4882
205	6,5341
206	6,5803
207	6,6269
208	6,6738
209	6,7210
210	6,7685
211	6,8164
212	6,8646
213	6,9132
214	6,9621
215	7,0114
216	7,0610
217	7,1109
218	7,1612
219	7,2119

220	7,2629
221	7,3143
222	7,3661
223	7,4182
224	7,4707
225	7,5235
226	7,5767
227	7,6303
228	7,6843
229	7,7387
230	7,7934
231	7,8486
232	7,9041
233	7,9600
234	8,0163
235	8,0731
236	8,1302
237	8,1877
238	8,2456
239	8,3040
240	8,3627
241	8,4219
242	8,4815
243	8,5415
244	8,6019
245	8,6628
246	8,7240
247	8,7858
248	8,8479
249	8,9105
250	8,9736
251	9,0370
252	9,1010
253	9,1654
254	9,2302
255	9,2955
256	9,3613
257	9,4275
258	9,4942
259	9,5614
260	9,6290
261	9,6972
262	9,7658
263	9,8349
264	9,9044
265	9,9745
266	10,0451
267	10,1161
268	10,1877
269	10,2598
270	10,3324
271	10,4055
272	10,4791
273	10,5532
274	10,6279
275	10,7031

276	10,7788
277	10,8551
278	10,9319
279	11,0092
280	11,0871
281	11,1655
282	11,2445
283	11,3241
284	11,4042
285	11,4849
286	11,5661
287	11,6480
288	11,7304
289	11,8134
290	11,8970
291	11,9811
292	12,0659
293	12,1513
294	12,2372
295	12,3238
296	12,4110
297	12,4988
298	12,5872
299	12,6763
300	12,7660
301	12,8563
302	12,9472
303	13,0388
304	13,1311
305	13,2240
306	13,3175
307	13,4118
308	13,5067
309	13,6022
310	13,6984
311	13,7954
312	13,8930
313	13,9913
314	14,0902
315	14,1899
316	14,2903
317	14,3914
318	14,4932
319	14,5958
320	14,6990
321	14,8030
322	14,9078
323	15,0132
324	15,1195
325	15,2264
326	15,3342
327	15,4426
328	15,5519
329	15,6619
330	15,7727
331	15,8843

332	15,9967
333	16,1099
334	16,2239
335	16,3386
336	16,4542
337	16,5706
338	16,6879
339	16,8059
340	16,9248
341	17,0446
342	17,1652
343	17,2866
344	17,4089
345	17,5321
346	17,6561
347	17,7810
348	17,9068
349	18,0335
350	18,1611
351	18,2896
352	18,4190
353	18,5493
354	18,6805
355	18,8127
356	18,9458
357	19,0798
358	19,2148
359	19,3508
360	19,4877

Standardní atmosféra

Výška (m)	Teplota (°C)	Tlak (Pa)	Hustota (kg/m³)	Rychlost zvuku (m/s)
0,00	15,00	101325,00	1,23	340,20
500,00	11,75	95457,00	1,17	338,30
1000,00	8,50	89871,00	1,11	336,40
1500,00	5,25	84550,00	1,06	334,40
2000,00	2,00	79490,00	1,01	332,50
2500,00	-1,25	74675,00	0,96	330,50
3000,00	-4,50	70099,00	0,91	328,50
3500,00	-7,75	65753,00	0,86	326,50
4000,00	-11,00	61629,00	0,82	324,50
4500,00	-14,25	57715,00	0,78	322,50
5000,00	-17,50	54007,00	0,74	320,50
6000,00	-24,00	47165,00	0,66	316,30
7000,00	-30,50	41046,00	0,59	312,20
8000,00	-37,00	35582,00	0,53	308,00
9000,00	-43,50	30724,00	0,47	303,70
10000,00	-50,00	26419,00	0,41	299,40
12000,00	-55,00	19338,00	0,31	296,00
15000,00	-55,00	12086,00	0,19	296,00
20000,00	-55,00	5521,00	0,09	296,00
25000,00	-55,00	2522,00	0,04	296,00
30000,00	-55,00	1152,00	0,02	296,00
40000,00	4,00	277,50	0,003 4	335,50
50000,00	77,00	92,82	0,000 9	375,10
75000,00	-15,00	6,41	0,000 08	322,20

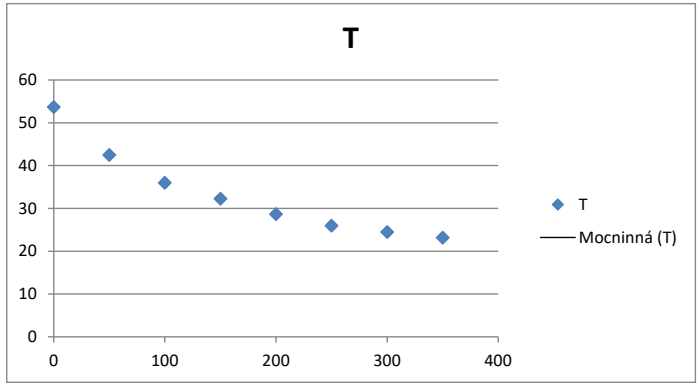
Měděný drát (α = 1,7.10-5K-1) měl při teplotě-50C délku 21,55 m.. Jakou délku má při teplotě 300C. O kolik cm se drát prodloužil? http://www.priklady.eu/cs/Fyzika/Tepelna-roztaznost.alej#sthash.yqNy7mT7.dpuf

$l=l0(1+a\ dt)$

Ochlazování teploměru

$T-Tokolí=T0\ exp(-kt)$

t	T
0	53,7
50	42,5
100	36
150	32,3
200	28,7
250	26
300	24,5
350	23,2
Tokolí	22,5



$A\ exp(-Bx)+C$

Analyzátor	
Čas (min)	X (V)
0	1,5400
1	1,5509
2	1,5619
3	1,5729
4	1,5840
5	1,5953
6	1,6065
7	1,6179
8	1,6294
9	1,6409
10	1,6525
11	1,6642
12	1,6760
13	1,6878
14	1,6998
15	1,7118
16	1,7239
17	1,7361
18	1,7484
19	1,7607
20	1,7732
21	1,7857
22	1,7984
23	1,8111
24	1,8239
25	1,8368
26	1,8498
27	1,8629
28	1,8761
29	1,8893
30	1,9027
31	1,9162
32	1,9297
33	1,9434
34	1,9571
35	1,9710
36	1,9849
37	1,9990
38	2,0131
39	2,0274
40	2,0417
41	2,0561
42	2,0707
43	2,0853
44	2,1001
45	2,1150
46	2,1299
47	2,1450
48	2,1602
49	2,1754
50	2,1908
51	2,2063

Gravimetricky	
Čas (min)	X (g/l)
0	1,89
59	2,66
125	3,95
190	6,51
240	7,98
320	13,24
360	15,46

52	2,2219
53	2,2377
54	2,2535
55	2,2694
56	2,2855
57	2,3017
58	2,3179
59	2,3343
60	2,3509
61	2,3675
62	2,3842
63	2,4011
64	2,4181
65	2,4352
66	2,4524
67	2,4698
68	2,4873
69	2,5049
70	2,5226
71	2,5404
72	2,5584
73	2,5765
74	2,5947
75	2,6131
76	2,6316
77	2,6502
78	2,6689
79	2,6878
80	2,7068
81	2,7260
82	2,7453
83	2,7647
84	2,7843
85	2,8040
86	2,8238
87	2,8438
88	2,8639
89	2,8842
90	2,9046
91	2,9251
92	2,9458
93	2,9666
94	2,9876
95	3,0088
96	3,0301
97	3,0515
98	3,0731
99	3,0948
100	3,1167
101	3,1388
102	3,1610
103	3,1833
104	3,2059
105	3,2285
106	3,2514
107	3,2744

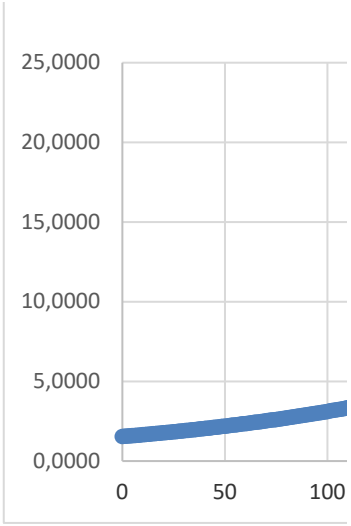
108	3,2976
109	3,3209
110	3,3444
111	3,3680
112	3,3919
113	3,4159
114	3,4400
115	3,4644
116	3,4889
117	3,5136
118	3,5384
119	3,5635
120	3,5887
121	3,6141
122	3,6396
123	3,6654
124	3,6913
125	3,7174
126	3,7437
127	3,7702
128	3,7969
129	3,8238
130	3,8508
131	3,8780
132	3,9055
133	3,9331
134	3,9609
135	3,9890
136	4,0172
137	4,0456
138	4,0742
139	4,1031
140	4,1321
141	4,1613
142	4,1908
143	4,2204
144	4,2503
145	4,2803
146	4,3106
147	4,3411
148	4,3718
149	4,4028
150	4,4339
151	4,4653
152	4,4969
153	4,5287
154	4,5607
155	4,5930
156	4,6255
157	4,6582
158	4,6912
159	4,7244
160	4,7578
161	4,7914
162	4,8253
163	4,8595

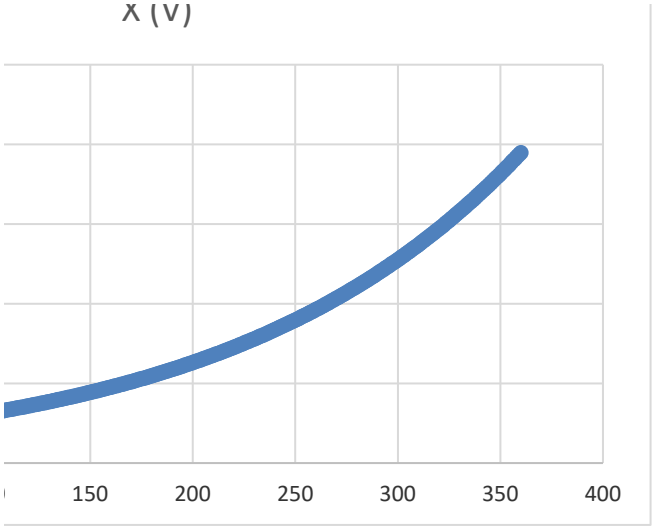
164	4,8939
165	4,9285
166	4,9634
167	4,9985
168	5,0338
169	5,0695
170	5,1053
171	5,1414
172	5,1778
173	5,2144
174	5,2513
175	5,2885
176	5,3259
177	5,3636
178	5,4015
179	5,4397
180	5,4782
181	5,5170
182	5,5560
183	5,5953
184	5,6349
185	5,6748
186	5,7149
187	5,7554
188	5,7961
189	5,8371
190	5,8784
191	5,9200
192	5,9619
193	6,0040
194	6,0465
195	6,0893
196	6,1324
197	6,1758
198	6,2195
199	6,2635
200	6,3078
201	6,3524
202	6,3973
203	6,4426
204	6,4882
205	6,5341
206	6,5803
207	6,6269
208	6,6738
209	6,7210
210	6,7685
211	6,8164
212	6,8646
213	6,9132
214	6,9621
215	7,0114
216	7,0610
217	7,1109
218	7,1612
219	7,2119

220	7,2629
221	7,3143
222	7,3661
223	7,4182
224	7,4707
225	7,5235
226	7,5767
227	7,6303
228	7,6843
229	7,7387
230	7,7934
231	7,8486
232	7,9041
233	7,9600
234	8,0163
235	8,0731
236	8,1302
237	8,1877
238	8,2456
239	8,3040
240	8,3627
241	8,4219
242	8,4815
243	8,5415
244	8,6019
245	8,6628
246	8,7240
247	8,7858
248	8,8479
249	8,9105
250	8,9736
251	9,0370
252	9,1010
253	9,1654
254	9,2302
255	9,2955
256	9,3613
257	9,4275
258	9,4942
259	9,5614
260	9,6290
261	9,6972
262	9,7658
263	9,8349
264	9,9044
265	9,9745
266	10,0451
267	10,1161
268	10,1877
269	10,2598
270	10,3324
271	10,4055
272	10,4791
273	10,5532
274	10,6279
275	10,7031

276	10,7788
277	10,8551
278	10,9319
279	11,0092
280	11,0871
281	11,1655
282	11,2445
283	11,3241
284	11,4042
285	11,4849
286	11,5661
287	11,6480
288	11,7304
289	11,8134
290	11,8970
291	11,9811
292	12,0659
293	12,1513
294	12,2372
295	12,3238
296	12,4110
297	12,4988
298	12,5872
299	12,6763
300	12,7660
301	12,8563
302	12,9472
303	13,0388
304	13,1311
305	13,2240
306	13,3175
307	13,4118
308	13,5067
309	13,6022
310	13,6984
311	13,7954
312	13,8930
313	13,9913
314	14,0902
315	14,1899
316	14,2903
317	14,3914
318	14,4932
319	14,5958
320	14,6990
321	14,8030
322	14,9078
323	15,0132
324	15,1195
325	15,2264
326	15,3342
327	15,4426
328	15,5519
329	15,6619
330	15,7727
331	15,8843

332	15,9967
333	16,1099
334	16,2239
335	16,3386
336	16,4542
337	16,5706
338	16,6879
339	16,8059
340	16,9248
341	17,0446
342	17,1652
343	17,2866
344	17,4089
345	17,5321
346	17,6561
347	17,7810
348	17,9068
349	18,0335
350	18,1611
351	18,2896
352	18,4190
353	18,5493
354	18,6805
355	18,8127
356	18,9458
357	19,0798
358	19,2148
359	19,3508
360	19,4877




X (V)
150
200
250
300
350
400

1.00 0.818730753
1.10 0.802518798
1.20 0.786627861
1.30 0.771051586
1.40 0.755783741
1.50 0.740818221
1.60 0.726149037
1.70 0.711770323
1.80 0.697676326
1.90 0.683861409
2.00 0.670320046
2.10 0.65704682
2.20 0.644036421
2.30 0.631283646
2.40 0.618783392
2.50 0.60653066
2.60 0.594520548
2.70 0.582748252
2.80 0.571209064
2.90 0.559898367
3.00 0.548811636

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 9 – 3D GRAFIKA

- 3D grafika se používá se pro zobrazení funkcí dvou proměnných z=f(x,y)
- Souřadnice jednotlivých bodů funkce se generují pomocí příkazu **meshgrid**
- Možností zobrazení v 3D je mnoho – viz příklad, příkazy: **plot3, mesh, meshc, surf, surfc, surfl, surface**
- Příkaz **surface** tvoří objekt, s kterým mohu pracovat dále tak, jako s jinými objekty, tj. měnit jeho vlastnosti…
- Průmět do roviny x,y se dělá pomocí příkazu **contour** (výsledkem jsou jakési „vrstevnice“)
- Barvy: každá z barev je definována kombinací 3 základních barev (červená – zelená – modrá) v rozsahu <0;1>
- Aplikace barev pomocí příkazu **colormap**, škála pomocí příkazu **colorbar**
- Lze si pomoci vlastnostmi grafického okna

Příklad 09.01:

Modelový příklad, zobrazení funkce $f(x,y) = -x\exp(-x^2 - y^2)$ na intervalu $x, y \in \langle -2{,}5;\ 2{,}5 \rangle$; $\Delta x = \Delta y = 0{,}1$

```
%% Program 09.01
%  3f grafika modelový program
%  f(x,y) = -x exp(-x^2-y^2)
%  x = -2.5:.1:2.5     y = -2.5:.1:2.5
   clear, close, clc
   [x,y] = meshgrid(-2.5:.1:2.5);  % generace bodů
   z = -x.*exp(-x.^2-y.^2);        % definice funkce
%  nejjednodušší čárové zobrazení
   figure, plot3(x,y,z)
%  zobrazení v barvách
   figure, mesh(x,y,z), title('graf fce'),
   xlabel('\it x'), ylabel('\it y')
%  změna barevné škály
   pause, colormap(spring(3)) % počet barev v daném schématu
   pause, colormap([0 0 1])   % modrá
   pause, colormap([0 0 1;1 0 1;0 1 1]) % změna matice barev
%  průmět do roviny xy
   figure, meshc(x,y,z)
%  zobrazení pomocí povrchu
   figure,surf(x,y,z)
%  s průmětem
   pause, surfc(x,y,z)
%  nasvětlení objektu 45 stupňů (default)
   pause, surfl(x,y,z)
%  vlastní nasvětlení pomocí s=[azimut, elevace]
   pause, s = [80,10]; surfl(x,y,z,s)
%  lze definovat surface jako objekt
```

```
%  a dál s ním pracovat samostatně
   s1 = surface(x,y,z); get(s1)
%  průmět do roviny,"vrstevnice"
   contour(x,y,z)
%  odrazivost
   figure, s = [0,0,0]; o = [1,1,1,1];
   surfl(x,y,z,s,o)
```

Animace

V Matlabu je možné animovat křivky. Zde alespoň nejjednodušší přístup pro 2- i 3D animace

Animace pomocí příkazů **addpoint**, **drawnow. V případě 3D animace je zapotřebí nastavit si pohled na graf pomocí příkazu view.**

Příklad 09.08:

Modelový příklad: Zobrazte průběh funkce zadané parametricky pro $t \in \langle 0;\ 100 \rangle$; $\Delta t = 0{,}01$; $x = t \cos t$; $y = t \sin t$. Do grafu vyneste závislost y na x. Animujte (2D)

```
%% P09_08
%  Animace
   close, clear, clc
   h = animatedline('LineWidth',1);
   axis([-100 100 -100 100])
   t = 0:0.1:100;
   x = t.*cos(t);      % definice parametrické křivky
   y = t.*sin(t);
   for k = 1:length(t) % vykreselení
       addpoints(h,x(k),y(k)); % další bod
       drawnow       % animuj
   end
   xlabel('\it x'), ylabel('\it y')
```

Příklad 09.09:

Modelový příklad: Zobrazte průběh funkce zadané parametricky pro $t \in \langle 0;\ 8\pi \rangle$; $\Delta t = \pi/20$; $x = \cos t$; $y = \sin t$; $z = t/10$. Do grafu vyneste závislost z na x a y. Pohled zvolte: azimut i elevaci 50°. Animujte (3D)

```
%% P09_09
%  Animace 3D
   close, clear, clc
   t = 0:pi/20:8*pi;            % definice parametrické křivky
   x = cos(t);
   y = sin(t);
   z = t/10;
   h = animatedline;
   az = 50; el = 50; view(az,el); % pohled
   axis([-1 1 -1 1 0 10])     % rozsah os
   for k = 1:length(t)
       addpoints(h,x(k),y(k),z(k));
       drawnow % animuj
   end
   xlabel('\it x'), ylabel('\it y'),zlabel('\it z')
```

3D grafika pro líné programátory

Nové verze MATLAB mají vestavěné funkce: fplot3, fsurf, fmesh, fimplicit3, pomocí nichž se dají funkce vykreslovat rychleji bez většího programátorského úsilí. Př.

```
fsurf(@(x,y) x.*exp(-x.^2-y.^2))
fmesh(@(x,y) x.*exp(-x.^2-y.^2))
```

Lze tak vykreslovat i parametrické křivky, pokud jsou známy funkce pro jednotlivé veličiny, př.

```
fplot3(@(t) sin(2*t),@(t) cos(2*t), @(t) sin(3*t+2), [-pi,pi], '--')
```

Příklady pro samostatné řešení

Příklad 09.02:

Znázorněte vhodně průběh funkce $z = \sin Q/Q$, kde $Q = \sqrt{x^2 + y^2}$ pro $x, y \in \langle -8; 8 \rangle$; $\Delta x = \Delta y = 0{,}1$. Do téhož grafu zobrazte i vrstevnicové znázornění.

Příklad 09.03:

Vykreslete dráhu bodu pohybujícího se po spirále, jsou-li zadány následující parametry: $v_0 = 0{,}2$ m/s; $r = 1$ m; $\omega = 0{,}3$ rad/s. Čas uvažujte s vhodně malým krokem v rozmezí $t \in \langle 0; 120 \rangle$ s. Dráha je udána parametricky rovnicemi $x = r \sin \omega t$ $y = r \cos \omega t$ $z = v_0 t$. Animujte.

Příklad 09.04:

Graficky znázorněte funkci $f(x, y) = 2^{x^2+y^2}$ pro $x, y \in \langle -1; 1 \rangle$; $\Delta x = \Delta y = 0{,}05$. Pro graf zvolte novou paletu barev složenou ze tří barev. Použijte editor vlastností objektu.

Příklad 09.05:

Graficky znázorněte funkci $f(x, y) = \frac{2}{5+3x^2+4y^2}$ pro $x, y \in \langle -5; 5 \rangle$; $\Delta x = \Delta y = 0{,}2$.

Zobrazte 4 varianty grafu, které se budou lišit bodem nasvětlení a odrazivostí povrchu:

a) osvětlení z pozice [1 0 3]; odrazivost [0.7 0.7 0.9 10]

b) osvětlení z pozice [1 0 3]; odrazivost [0.7 0.7 0.9 2]

c) osvětlení z pozice [5 0 3]; odrazivost [0.7 0.7 0.9 10]

d) osvětlení z pozice [1 0 3]; odrazivost [0 0.2 0.2 10]

Příklad 09.06:

Do 4 podoken grafického okna vykreslete s vhodným krokem funkci $f(x, y) = \cos x^2 + \sin y^3$ pro $x, y \in \langle -2; 2 \rangle$; pomocí příkazů

a) plot3

b) mesh

c) surf

d) contour.

Příklad 09.07:

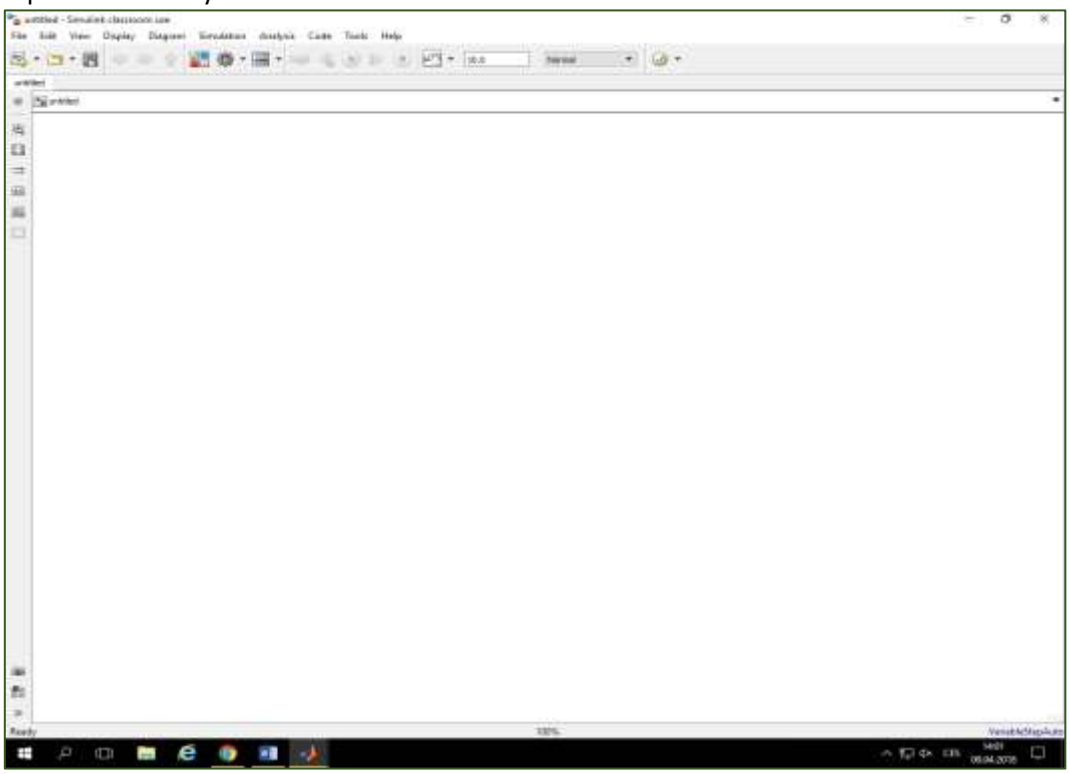
Graficky znázorněte funkci $f(x, y) = \sin 5x \cos 5y \,/5$ pro $x, y \in \langle -2; 2 \rangle$; $\Delta x = \Delta y = 0{,}05$. Nasvětlení a odrazivost povrchu budou [-1 0 3]; [1 1 1 10].

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 10 – SIMULINK

- Simulink je nadstavba MATLAB, určená pro simulaci a modelování dynamických systémů. Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat („Lego pro dospělé“)
- Modely mohou být popsány rovnicemi nebo mohou být sestavené z bloků reprezentujících prvky reálných systémů. Kromě modelů fyzikálních soustav je možné modelovat také algoritmy řídicích systémů včetně jejich automatického ladění, systémy pro zpracování signálu, komunikace a zpracování obrazu.
- Volání z Matlabu příkazem **simulink**, systém dává výběr, s čím se bude pracovat (vybírá se Blank Model), starší verze měly příponu modelů *.MDL, stávají verze příponu. SLX. Je třeba dodržovat konvenci pojmenování souborů jako v Matlabu, model se nesmí jmenovat stejně jako program, kterým případně budeme dopočítávat nějaké parametry s modelem související!!!
- Barevné tlačítko na liště (umístěné pod příkazem Simulation) otevírá knihovnu předdefinovaných bloků



Obr. 1 Okno modelu v SIMULINKU

- Každý blok reprezentuje nějakou vlastnost, funkci či operaci, kterou systém se signály jím procházejícími provádí. Všechny funkce v Simulinku jsou závislé na (simulačním) čase (ten běží jinak, než reálný, simulace vyjadřující dění v intervalu např. dvou hodin minut může být vypočítána v několika sekundách).
- Jednotlivé bloky z knihovny do okna modelu se umístí přetažením myší (funguje i kopírování a vložení)
- U většiny bloků je nutno nastavit jejich vnitřní parametry (dialogová okna se otevírají pomocí double-click na blok)

Vybrané bloky

Sources

Zdroje signálů

- **Clock** – hodiny; výstup odpovídá uplynulému simulačnímu času
- **Constant** – generuje výstup se zadanou konstantní hodnotou
- **Step** – generátor skokové funkce (z hodnoty Initial value na hodnotu Final value v zadaném simulačním čase Step time)
- **Sine Wave** – generuje funkci sinus s následujícími parametry výstupu: $y(t) = A \cdot \sin(2\pi \cdot f\, t + P) + B$, kde A je amplituda, f frekvence, P fázový posun a B posun po vertikální ose (funkce by šla realizovat i jinak)
- **In (In1)** – vstupní svorka pro signál do subsystému
- **Ramp** – generuje rampovou funkci (signál o průběhu $y(t) = k\,t + q$). Parametry: **Slope** je směrnice přímky; Start time udává hodnotu simulačního času, od které má začít generování rampového výstupu. Initial output je posun po vertikální ose.
- **Ground** – používá se k „zaslepení“ případných nepoužitých vstupních svorek u jiných bloků. Tím se zamezuje, aby Simulink vypisoval varovné zprávy upozorňující na nezapojené vstupní svorky. Výstupem bloku je nula.
- **From File/From Workspace/From SpreadSheat** – vstup dat ze souboru či z prostředí MATLAB, popřípadě z tabulky dat

Sources

Výstupy

- **Scope** – rychlé grafické zobrazení výstupu
- **XY Graph** – souřadnicový graf (horní vstup pro nezávislou, spodní pro závislou proměnnou)
- **Display** – zobrazuje okamžitou číselnou hodnotu signálu, který je do něj přiveden (v případě vektoru ukazuje hodnotu každé složky v samostatném okénku)
- **Terminator** – viz Ground pro výstupy
- **To Workspace/To File** – ukládá hodnoty signálu do definované proměnné v pracovním prostoru Matlabu, případně do souboru (pozor na nastavení struktury dat!)
- **Out (Out1)** – výstupní svorka signálu ze subsystému

Math Operations
Realizace matematických operací

- **Abs** – funkce pro absolutní hodnotu
- **Add** – součet; **Subtract** – rozdíl; **Divide** – podíl; **Product** – násobení; **Dot Product** – násobení po složkách
- **Sum** – součet/rozdíl
- **Gain** – násobení konstantou
- **Sqrt**, **Trigonometric Function**, **Sign**, **Sine Wave** – vybrané matematické funkce
- **Math Function** – jakákoliv matematická funkce

Continuous
Vybrané funkce pro výpočet diferenciálních rovnic a práci ve stavovém prostoru

- **Integrator** – numericky integruje hodnoty vstupního signálu v závislosti na čase. Inital condition – zadání počátečních podmínek
- **Derivative** – numericky derivuje vstupní signál podle času. Výstupem derivátoru v daném čase je hodnota derivace vstupního signálu v tomto čase (derivace v bodě)
- **Transfer Fcn** –přenos ve tvaru Laplaceovy transformace
- **State-Space** – systém ve formě stavového modelu. Do políček A, B, C a D se zadají matice stavového modelu

Signal Routing
- **Demux** - rozdělování vícerozměrných signálů na jednotlivé složky
- **Mux** – opak demus,směšuje jednotlivé signály do jediného vícerozměrného signálu
- **Goto** - „bezdrátový“ přenos signálu v modelu. Signál přivedený na vstupní svorku zasílá do korespondujícího **From** bloku. V kolonce Tag se nastavuje název tohoto From bloku.
- **From** - „bezdrátově“ přijímá signál z bloku Goto definovaného v kolonce Goto Tag
- **Manual Switch** - propouští signál z jedné ze dvou vstupních svorek, přepnutí mezi svorkami se provádí dvojklikem na blok, funguje i během simulace

User-Defined Functions
- **Fcn** – lze vytvořit libovolný matematický výraz, jehož proměnou je vstup tohoto bloku (ve výrazu značen u). Výstupem je pak výsledek matematického výrazu. Podle rozměru má vstup, se lze odkazovat na jednotlivé jeho složky $u(1)$... $u(n)$.

Parametry modelu
- Modelling – Model settings (CTRL+E) – Solver – Solver Details – Relative tolerance, případně nastavení maximální a minimální velikosti kroku
- Komentovat – double click na blok

Příklady pro samostatné řešení

Příklad 10.01:

Realizujte časový průběh funkce $y = \sin(3t)$, pro $t \in \langle 0;\ 10 \rangle$ (existuje více možností, prozkoumejte všechny)

Příklad 10.02:

Realizujte v jediném modelu časový průběh funkcí $y_1 = \exp(-t)\sin(3t)$; $y_2 = \exp(-3t)\sin(t)$; $t \in \langle 0;\ 10 \rangle$. Výstup zobrazte do téhož grafu.

Příklad 10.03:

Realizujte v jediném modelu časový průběh funkcí $y_1 = \sin(t)$; $y_2 = \cos(t)$; $t \in \langle 0;\ 20 \rangle$. Výstup zobrazte do téhož grafu typu XY.

Příklad 10.04:

Realizujte v jediném modelu časový průběh funkcí $y_1 = 0.2\sin(t)$; $y_2 = 0.5\cos(t)$; $t \in \langle 0;\ 20 \rangle$. Výstup zobrazte do grafu XY.

Příklad 10.05:

Realizujte systém, který modeluje logaritmickou spirálu danou rovnicemi $y_1 = \exp(-\mathrm{k}\,t)\sin(t)$; $y_2 = \exp(-\mathrm{k}\,t)\cos(t)$; $t \in \langle 0;\ 30 \rangle$. Parametr k je volitelná konstanta. Výstup zobrazte do grafu XY (upravte parametry zobrazení).

Příklad 10.06:

Realizujte systém, který modeluje Archimédovu spirálu danou rovnicemi
$y_1 = t\sin(t)$; $y_2 = t\cos(t)$; $t \in \langle 0;\ 35 \rangle$. Výstup zobrazte do grafu XY (upravte parametry zobrazení).

Příklad 10.07:

Řešte pomocí Simulinku ODR $y' = 2y$, s počáteční podmínkou $y(0) = 1$ pro simulační čas $t \in \langle 0;\ 2 \rangle$.

Příklad 10.08:

Řešte pomocí Simulinku ODR $y'' + 4y' + 3y = 8\exp(t)$, s počátečními podmínkami $y(0) = 1$, $y'(0) = 1$, pro simulační čas $t \in \langle 0;\ 3 \rangle$.

Příklad 10.09:

Řešte pomocí Simulinku soustavu ODR s počátečními podmínkami pro simulační čas $t \in \langle 0;\ 9 \rangle$. Zobrazte pomocí XY grafu (upravte si vhodně rozsah zobrazení) závislost z na y:

$y' = y - yz,\ y(0) = 4$

$z' = -z + yz,\ z(0) = 1$

Příklad 10.10:

Řešte pomocí Simulinku ODR $y'' + 3y = 4\sin(t) - 5\cos(t)$, s počátečními podmínkami $y(0) = 0, y'(0) = 1$, pro simulační čas $t \in \langle 0;\ 15 \rangle$.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 11 – ŘEŠENÍ DIFERENCIÁLNÍCH ROVNIC

- Řešení ODR pomocí řešitelů **odeXX**. Obecná syntaxe
  `[t,y]=odeXX(odefun,tspan,y0,podm)`
  V praxi nejčastěji používána metoda Runge-Kutta 4. řádu, tj. ode45, `odefun` je odkaz na funkci, která má být řešena, `tspan` je doba řešení, y0 vektor počátečních podmínek, podm jsou doplňující podmínky (např. přesnost řešení)
- Pokud bude diferenciální rovnice vyššího řádu, je třeba zavést substituci a převést ji na soustavu rovnic 1. řádu

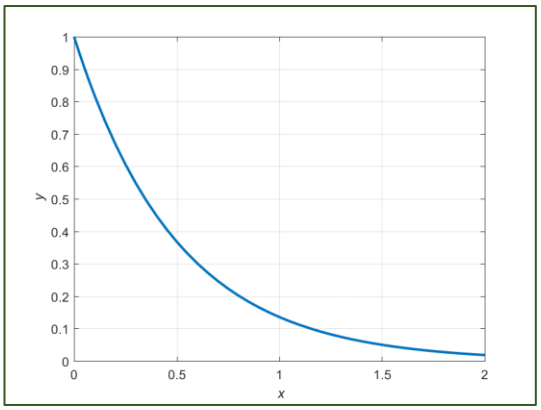
Příklad 11.01:

Modelový příklad: Řešte pomocí řešitele ODE diferenciální rovnici $y' = -2\,y$, s počáteční podmínkou $y(0) = 1$ pro $t \in \langle 0; 2 \rangle$.

Řešení

a) Nejjednodušší způsob, kdy je funkce jednoduchá a lze vše napsat naráz
b) Sofistikovanější způsob s funkčním podprogramem

```
%% AAP 11_01
% Řešení ODE
   clear, close, clc
%  a) nejjednodušší, vše do jednoho řádku
   [t,y] = ode45(@(t,y) -2*y,[0 2],[1]);
   plot(t,y,'LineWidth',2), grid
   xlabel('\it x'), ylabel('\it y')

%  b) sofistikovanější pomocí funkce
   y0 = 1;
   tspan = [0:0.001:2];
   [t,y] = ode45(@Fce11_01,tspan,y0);
   plot(t,y,'LineWidth',2), grid
   xlabel('\it x'), ylabel('\it y')

function dydt=Fce11_01(t,y)
% Funkce pro řešení pomocí ODE
% dy = -2y
  dydt = -2*y;
end
```



Obr. 12 Řešení ODE 11.01

Příklad 11.02:

Modelový příklad: Řešte pomocí řešitele ODE diferenciální rovnici $y'' + 4\,y' + 3\,y = 8\,\exp(t)$, s počátečními podmínkami $y(0) = 1, y'(0) = 1$, pro simulační čas $t \in \langle 0; 3 \rangle$.

Řešení – osamostatnit nejvyšší derivaci a pak substituce $y'' = -4\,y' - 3\,y + 8\,\exp(t)$. Alokovat prostor pro všechny derivace jako nulové vektory. Substituuje se $y_1 = y; y_2 = y'$. Potom se zavede soustava rovnic

$y_1' = y_2$;

$y_2' = -4\,y_2 - 3\,y_1 + 8\,\exp(t)$.

```
%% Příklad 11.02
%  Řešení ODE
   clear, close, clc
   y0 = [1,1];
   tspan = [0:0.001:3];
   [t,y] = ode45(@Fce11_02,tspan,y0);
%  vykreslení obou křivek
   plot(t,y,'LineWidth',2), grid, pause
%  vykreslení pouze poslední křivky
   plot(t,y(:,2),'LineWidth',2), grid
   xlabel('\it x'), ylabel('\it y')

function dydt=Fce11_02(t,y)
% y''=-4y'-3y+8exp(t), substiuce
    dydt = zeros(2,1); % alokace prostoru (řád rovnice, 1)
    dydt(1) = y(2);
    dydt(2) = -4*y(2)-3*y(1)+8*exp(t);
end
```

Příklady pro samostatné řešení

Příklad 11.03:

Řešte pomocí řešitele ODE soustavu diferenciálních rovnic s počátečními podmínkami pro simulační čas $t \in \langle 0; 9 \rangle$. Graficky znázorněte závislost z na y

$y' = y - yz,\ y(0) = 4$
$z' = -z + yz,\ z(0) = 1$

Příklad 11.04:

Řešte pomocí řešitele ODE diferenciální rovnici $y'' + 3\ y = 4\ \sin\ t - 5\ \cos\ t$, s počátečními podmínkami $y(0) = 0, y'(0) = 1$, pro simulační čas $t \in \langle 0; 15 \rangle$.

Příklad 11.05:

Řešte pomocí řešitele ODE diferenciální rovnici kinetiky s daty k = 2,4 10^{-6}; p_{B0} = 240 kPa, $t \in \langle 0; 5000 \rangle$. Graficky znázorněte.

$$\frac{\mathrm{d}p_B}{\mathrm{d}t} = -3kp_B^{3/2}$$

Příklad 11.06:

Je dána válcová nádrž o průměru dna 1,5 m. Na výtoku nádrže je ventil, jehož rovnice je

$$Q = 0{,}1K_v \frac{z}{z_r}\sqrt{\mathrm{g}h}$$

Význam proměnných: K_v je jmenovitý průtok ventilu (specifická tabulková charakteristika $m^3.h^{-1}$), z je momentální zdvih ventilu (mm), z_r maximální regulační zdvih (mm), g gravitační konstanta, h výška hladiny v nádrži (m), Q objemový průtok ventilem za předpokladu konstantní hustoty ($m^3.s^{-1}$). Data: K_v = 16 m^3/h; z = 9 mm ; z_r = 21 mm; Q = 2 m^3/h. Spočtěte počáteční výšku hladiny. Řešte diferenciální rovnici pro případ, že se přítok Q změní na 2,5 m^3/h, $t \in \langle 0; 2{,}5\ 10^4 \rangle$

$$A\frac{\mathrm{d}h}{\mathrm{d}t} = Q - 0{,}1K_v \frac{z}{z_r}\sqrt{\mathrm{g}h}$$

Graficky znázorněte.

Příklad 11.07:

Je dána kinetická rovnice rozkladu látky A: $\frac{\mathrm{d}c_\mathrm{A}}{\mathrm{d}t} = -kc_\mathrm{A}$ s počáteční podmínkou c_{A0}. Řešte pomocí pomocí řešitele ODE, jestliže k = 10^{-2} s^{-1} a c_{A0} = 0,1 mol/dm^3. Zobrazte výsledek pro časový rozsah $t \in \langle 0; 500 \rangle$ s (viz příklad **7.11**) Graficky znázorněte.

Příklad 11.08:

Je dána kinetická rovnice 2. řádu rozkladu látky A: $\frac{\mathrm{d}c_\mathrm{A}}{\mathrm{d}t} = -kc_\mathrm{A}^2$ s počáteční podmínkou c_{A0}. Řešte pomocí pomocí řešitele ODE, jestliže k = 10^{-2} s^{-1} a c_{A0} = 0,1 mol/dm^3. Zobrazte výsledek pro časový rozsah $t \in \langle 0; 1{,}2\ .\ 10^5 \rangle$ s (viz příklad **7.12**) Graficky znázorněte.

Příklad 11.09:

V reaktoru probíhají následné reakce $\mathrm{A} \rightarrow \mathrm{B} \rightarrow \mathrm{C}$. Vypočtěte jednotlivé koncentrace jako funkce času, jestliže následné reakce mají rychlostní konstanty $k_1 = 6.10^{-4}$ a $k_2 = 5.10^{-4}$; $c_{A0} = 2.5\ \mathrm{mol/dm^3}$; $c_{B0} = 0\ \mathrm{mol/dm^3}$; $c_{c0} = 0\ \mathrm{mol/dm^3}$. Řešte pomocí pomocí řešitele ODE, jsou-li dány diferenciální rovnice kinetického chování soustavy, pro $t \in \langle 0;\ 15000 \rangle$ s, s krokem $\Delta t = 60$ s. Graficky znázorněte.

$$\frac{\mathrm{d}c_A}{\mathrm{d}t} = -k_1 c_A;\ \frac{\mathrm{d}c_B}{\mathrm{d}t} = k_1 c_A - k_2 c_B;\ \frac{\mathrm{d}c_C}{\mathrm{d}t} = k_2 c_B$$

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 12 – SAMOSTATNÝ PROJEKT I

Část I – Hrátky s π

U příkladů určete také chybu od počítačem daného čísla π v případě, že použijete 10, 30, 100 a 1000 členů (v případě příkladů používajících faktoriál jen 30 členů). Každý ze studentů realizuje minimálně 2 ze zadaných úkolů, předkládá programy a výsledky shrnuté do přehledné tabulky

Příklad 12.01:

Sestavte program na výpočet čísla π Vieteovým rozvojem

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2}\frac{\sqrt{2+\sqrt{2}}}{2}\dots$$

Z klávesnice je zadán počet členů rozvoje, formát zobrazení volte long

Řešte tentýž problém s tím, že místo počtu členů je zadána přesnost řešení (10^{-15}). Vypište, kolik členů je potřeba k dosažení dané přesnosti.

Příklad 12.02:

Sestavte program na výpočet čísla π pomocí Leibnitzovy řady

$$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}$$

Z klávesnice je zadán počet členů řady, formát zobrazení volte long

Řešte tentýž problém s tím, že místo počtu členů je zadána přesnost řešení (10^{-8}). Vypište, kolik členů je potřeba k dosažení dané přesnosti.

Příklad 12.03:

Sestavte program na výpočet čísla π pomocí Madhav-Leibnitzovy řady

$$\pi = \sqrt{12}\sum_{k=0}^{\infty}\frac{(-3)^{-k}}{2k+1}$$

Z klávesnice je zadán počet členů řady, formát zobrazení volte long

Řešte tentýž problém s tím, že místo počtu členů je zadána přesnost řešení (10^{-15}). Vypište, kolik členů je potřeba k dosažení dané přesnosti.

Příklad 12.04:

Sestavte program na výpočet čísla π pomocí Lambertovy řady

$$\frac{\pi^2}{6} = \sum_{k=1}^{\infty} \frac{1}{k^2}$$

Z klávesnice je zadán počet členů řady, formát zobrazení volte long

Řešte tentýž problém s tím, že místo počtu členů je zadána přesnost řešení (10^{-5}). Vypište, kolik členů je potřeba k dosažení dané přesnosti.

Příklad 12.05:

Sestavte program na výpočet čísla π pomocí Wallisova součinu

$$\frac{\pi}{2} = \prod_{k=1}^{\infty} \frac{(2k)^2}{(2k)^2 - 1}$$

Z klávesnice je zadán počet členů součinu, formát zobrazení volte long

Řešte tentýž problém s tím, že místo počtu členů je zadána přesnost řešení (10^{-5}). Vypište, kolik členů je potřeba k dosažení dané přesnosti.

Příklad 12.06:

Sestavte program na výpočet čísla π pomocí Rámanudžanovy řady

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!\,(1103 + 26390k)}{(k!)^4 396^{4k}}$$

Z klávesnice je zadán počet členů řady, formát zobrazení volte long

Příklad 12.07:

Sestavte program na výpočet čísla π pomocí Čudovských řady

$$\frac{426880\sqrt{10005}}{\pi} = \sum_{k=0}^{\infty} \frac{(6k)!\,(13591409 + 545140134k)}{(3k)!\,(k!)^3 (-640320)^{3k}}$$

Z klávesnice je zadán počet členů řady, formát zobrazení volte long

Příklad 12.08:

Sestavte program na výpočet čísla π pomocí BBP vzorce (Bailey-Borwein-Plouffe)

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

Z klávesnice je zadán počet členů řady, formát zobrazení volte long

Řešte tentýž problém s tím, že místo počtu členů je zadána přesnost řešení (10^{-15}). Vypište, kolik členů je potřeba k dosažení dané přesnosti.

Část II – Rozhodování

Každý ze studentů odladí následující program pro různá zadání vstupních dat. Výsledky přehledně shrne.

Příklad 12.09:

Z klávesnice je zadáno celé číslo n (počet pohybů na účtu), dále n reálných čísel (lze i jako dialogové menu, v tom případě se předpokládá, že pokud bude zadáno méně čísel, zbývající se doplní nulami). Počáteční stav účtu považujte za nulový. Vypište, zda a kolikrát se účet dostal do záporných hodnot (tedy zda se čerpal kontokorent). Dále zobrazte konečný stav účtu.

Část III – Algoritmizace

Příklad 12.10:

Sestavte a znázorněte algoritmus, který:

a) Načte řadu čísel a uloží do indexované proměnné $a(i)$
b) Nalezne maximum a minimum hodnot číselné řady a uloží do proměnných $amax$, $amin$
c) Přepočte pomocí minima a maxima hodnoty číselné řady na bezrozměrné hodnoty pomocí vzorce $a(i) = \frac{a(i)-amin}{amax-amin}$

d) Realizujte v MATLABu pro zadanou číselnou řadu co nejjednodušeji a nejelegantněji

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 13 – KOMPLEXNÍ PROBLÉMY

- Příkaz **linspace** – generuje N bodů na intervalu A, B; syntaxe `linspace(A,B,N)`
- Příkaz **pcolor** – pseudobarva matice (pro rychlé grafické znázornění), hodnoty jsou barevně odstupňovány, spolu s příkazem `shading interp`, má každá buňka matice barvu vypočtenou bilineární interpolací 4-spojitého prostoru. Syntaxe `pcolor(x, y, F)`, kde x a y jsou nezávislé proměnné a F jejich funkce

Příklad 13.01:

Ustálené vedení tepla ve vícedimensionálním prostoru je dáno řešením Fourierovy rovnice s okrajovými podmínkami (odvození viz **Chemické inženýrství**). Uvažujme dlouhé těleso obdélníkového průřezu o rozměrech a×b, umístěné v prostředí s konstantními vlastnostmi. Řešením Fourierovy rovnice se získají vztahy

$$\mathrm{T}(x,y) = \sum_{n=1}^{\infty} \mathrm{T}_n(y)\Psi_n(x)$$

$$\mathrm{T}_n(y) = k_n \exp(-\lambda_n y)\,[\exp(\lambda_n y) - 1][\exp(\lambda_n y) - \exp(\lambda_n b)]\lambda_n^{-3}[1 + \exp(\lambda_n b)]^{-1}$$

$$\Psi_n(x) = \sqrt{\frac{2}{a}}\sin(\lambda_n x)$$

$$k_n = \sqrt{\frac{2}{a}}[-1 + (-1)^n]\alpha;\ \lambda_n = \frac{n\pi}{a}$$

Naprogramuje výpočet tak, aby z klávesnice byly zadávány rozměry a, b, počet dílků, na které se budou intervaly dělit (může být různý ve směru osy x a y), koeficient přestupu tepla α (W.m^{-2}.K^{-1}) a počet členů posloupnosti N, který bude pro výpočet požadován, vlastní výpočet pak programujte jako funkci. Výstup graficky znázorněte

Příklad 13.02:

Sestavte program pro hledání minima funkce gradientní metodou. Jako testovací funkci využijte např.

$$\mathrm{f}(x,y) = \frac{1}{\exp(a_1 x + a_2) + 1}$$

Příklad 13.03:

Napište GUI pro výpočet Velikonoc (viz **Příklad 5.13**) – rozšiřující látka pro pokročilé

Příklad 13.04:

(Modelová úloha)

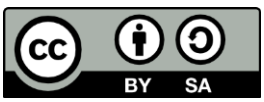
Sestavte program pro výpočet jednoho z chybějících parametrů rovnice ideálního plynu:

```
%% Program 13.04
   clear, close,clc
   x = [233424.06 NaN 300.02]; % data
   y = Fce05_14(x)

   function y = Fce05_14(x)
        P = x(1);    V = x(2);   T = x(3);
        R = 8.314;   n = 100;  y=' ';
        % podmínky
        p1 = (V>=1) & (V<=10) & (T>=300) & (T<=500);
        p2 = (P>=1e5) & (P<=3e5) & (T>=300) & (T<=500);
        p3 = (P>=1e5) & (P<=3e5) & (V>=1) & (V<=10);
        if isnan(P) % chybí P
            if p1
                    P = n*R*T/V;
                    y = [sprintf('%0.2f',P) ' Pa'];
            end
        elseif isnan(V) % chybí V
            if p2
                V = n*R*T/P;
                y = [sprintf('%0.2f',V) ' m^3'];
            end
        else % isnan(T) chybí T
            if p3
                T = P*V/R/n;
                y = [sprintf('%0.2',T) ' K'];
            end
        end
   end
```

Řešení úloh 13

```
%% Program 13.01
%% Data
   clear, close, clc
   a = input('zadej rozměr a: ');
   b = input('zadej rozměr b: ');
   na = input('zadej počet dílků a: ');
   nb = input('zadej počet dílků b: ');
   alfa= input('zadej koeficient alfa: ');
   N = input('zadej počet členů nekonečné posloupnosti: ');

%% Výpočet
   %  dělení rozměru a,b na příslušný počet dílků
   x = linspace(0,a,na);
   y = linspace(0,b,nb);
   konst = sqrt(2/a);
   for i = 1:length(x)
       for j = 1:length(y)
           T(i,j) = 0;
           for n = 1:N
               lambdan = n*pi/a; % výpočet dané lambdy
               kn = konst*(-1+(-1)^n)*alfa; % pomocná konstanta
               Psin = konst*sin(lambdan*x(i));
               Tn = kn*exp(-lambdan*y(j))*...
                  (-1+exp(lambdan*y(j)))*...
                  (-exp(b*lambdan) + exp(lambdan*y(j)))/...
                  ((1 + exp(b*lambdan))*lambdan^3);
               T(i,j) = T(i,j)+Tn*Psin;
           end
       end
   end

%% Zobrazení
   figure
   pcolor(x,y,T); xlabel('\it x'), ylabel('\it y'),
   shading interp, colorbar, axis equal
```

```
%% Program 13.02
%  Nelineární gradientní metoda
   clear, close, clc
   syms aa1 aa2 ff      % def. symbolických pro derivaci fce
%% Data
   x = [-2 -1 1 2];
   y = Fce_1302([-0.6 0.4],x);  % data se známými koeficienty
   ff = 1./(exp(aa1*x+aa2)+1);  % symbolická fce
   R1 = -2:0.05:0; R2=0:0.05:1; % rozsahy hledání parametrů

%% Účelová fce
   for i = 1:length(R1)
       for j = 1:length(R2)
           S(i,j) = sum((y-Fce_1302([R1(i),R2(j)],x)).^2);
       end
   end

%% Zobrazení účelové fce v prostoru
   figure
   subplot(121), meshc(R2,R1,S), axis tight
   xlabel('{\it a}_2'), ylabel('{\it a}_1'),
   zlabel('S({\it a}_1, {\it a}_2)')

%% Gradientní metoda
   a1(1) = 1; a2(1) = 1;  % nástřel hledaných parametrů
   alpha = 0.7;           % váha
   nmax = 500;            % max počet iterací
   tol = 1e-5;            % přesnost

   SS = (y-ff).^2;        % účelová fce symbolicky
%  výpočet derivací
   for i = 1:nmax
      % derivace dle hledaných konstant symbolicky
        Da1 = diff(SS,aa1); Da2 = diff(SS,aa2);
      % dosazení za parametry
        Da1 = subs(Da1,aa1,a1(i)); Da1 = subs(Da1,aa2,a2(i));
        Da2 = subs(Da2,aa1,a1(i)); Da2 = subs(Da2,aa2,a2(i));
      % sumy
        da1 = double(Da1); da1 = sum(da1);
        da2 = double(Da2); da2 = sum(da2);
      % výpočet normy parametrů a ověření přesnosti
        da = [da1 da2];
        err = norm(da)
      % zobrazuje se, aby bylo vidět, zda konverguje
      % pokud ne, je nutné změnit rozsahy nebo alfu
        if err < tol   % pokud je dosaženo přesnosti, končíme
            break
        end
      % posun
        a1(i+1) = a1(i)-alpha*da1; a2(i+1) = a2(i)-alpha*da2;
   end;

%% Zobrazení
%  účelová fce, vrstevnice a posloupnost iterací
   subplot(224), contour(R2,R1,S,50),
   hold on, plot(a2,a1,'ko','MarkerSize',2.5);
   xlabel('{\it a}_2'), ylabel('{\it a}_1'),
   hold off
```



```
%  výsledek aproximace, data kolečka (stem), fce červená čára
   subplot(222), stem(x,y), hold on
   x1 = min(x):(max(x)-min(x))/1000:max(x);
   plot(x1,Fce_1302([a1(end), a2(end)],x1),'r','LineWidth',2);
   xlabel('{\it x}'), ylabel('{\it y}'), grid on
   hold off

function f = Fce_1302(a,x)
% f=fa(a,x) funkce pro nelineární aproximaci
% f ... výstupní hodnoty
% x ... vstup
% a(i) ... koeficienty
  f = 1./(exp(a(1)*x+a(2))+1);
end
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

TÉMA 14 – SAMOSTATNÝ PROJEKT II

Pozn.: každému studentovi bude zadána jedna z variant úlohy. Hodnotí se funkčnost, složitost a „elegance“ zpracování. Výsledky shrnout do tabulek, grafy, obrázky atd. tisknout a vložit do protokolu), spolu s programy uložit do souboru s uživatelským jménem studenta a dodat domluvenou formou (mail, úložiště)

Příklad 14.01:

Graficky znázorněte danou funkci, nastavte barevné schéma na **copper**, osvětlení a odrazivost. Znázorněte pomocí příkazu **subplot** do 4 podoken:
(první defaultní osvětlení i odrazivost; druhé defaultní odrazivost, osvětlení na pozici [10 10 50]; třetí osvětlení na pozici [10 10 50], odrazivost [0 0 0 20]; čtvrté osvětlení na pozici [0 0 100], odrazivost [1 1 1 20])

a) $f(x,y) = 0.75/\exp(625x^2y^2)$ pro $x, y \in \langle -2; 2\rangle$; $\Delta x = \Delta y = 0{,}05$;
b) $f(x,y) = 1 - |x+y| - |y-x|$ pro $x, y \in \langle -3; 3\rangle$; $\Delta x = \Delta y = 0{,}05$;
c) $f(x,y) = 1 - |y|$ pro $x, y \in \langle -2; 2\rangle$; $\Delta x = \Delta y = 0{,}05$;
d) $f(x,y) = -\sqrt{x^2+y^2}$ pro $x, y \in \langle -5; 5\rangle$; $\Delta x = \Delta y = 0{,}05$;
e) $f(x,y) = (x^2-y^2)/100$ pro $x, y \in \langle -5; 5\rangle$; $\Delta x = \Delta y = 0{,}1$;
f) $f(x,y) = \sqrt{1+(2-\sqrt{x^2+y^2})^2}$ pro $x, y \in \langle -5; 5\rangle$; $\Delta x = \Delta y = 0{,}1$;
g) $f(x,y) = \sqrt{x^2+y^2} + 3\cos\sqrt{x^2+y^2}$ pro $x, y \in \langle -5; 5\rangle$; $\Delta x = \Delta y = 0{,}1$;
h) $f(x,y) = \frac{1}{(\sin|x|+x)} + \frac{1}{(\cos|y|+y)}$ pro $x \in \langle -1; 0\rangle$, $y \in \langle -2; 4\rangle$; $\Delta x = 0{,}1$; $\Delta y = 0{,}05$;
i) $f(x,y) = \sin 5x \cos 5y + x^2 + 1$ pro $x, y \in \langle -5; 5\rangle$; $\Delta x = \Delta y = 0{,}1$;
j) $f(x,y) = x^2 + y^2 + x\,y \sin(x+y)$ pro $x, y \in \langle -20; 20\rangle$; $\Delta x = \Delta y = 0{,}25$;
k) $f(x,y) = \cos xy$ pro $x, y \in \langle -\pi; \pi\rangle$; $\Delta x = \Delta y = \pi/30$;
l) $f(x,y) = \cos(x^2+y^2)$ pro $x, y \in \langle -\pi; \pi\rangle$; $\Delta x = \Delta y = \pi/25$;

Příklad 14.02:

Řešte zadanou diferenciální rovnici

a) pomocí řešitele ODE
b) pomocí symbolické matematiky
c) pomocí Simulinku

Rovnice:

a) $y' = 2\sin 3t - 4y$, $y(0) = 1$; $t \in \langle 0; 2\rangle$;
b) $y' = 2y + t^2$, $y(0) = 1$; $t \in \langle 0; 1\rangle$;
c) $y' = 2y + \sqrt{t}$, $y(0) = 1$; $t \in \langle 0; 1\rangle$;
d) $y' = -ty(2-y)$, $y(0) = 1$; $t \in \langle 0; 2\rangle$;

e) $y' = -t(2 - y)$, $y(0) = 1$; $t \in \langle 0; 1{,}5 \rangle$;
f) $y' = -ty$, $y(0) = 1$; $t \in \langle 0; 3 \rangle$;
g) $y' = 2y(3 - y)$, $y(0) = 1$; $t \in \langle 0; 1 \rangle$;
h) $y' = (y - 4)(y^2 - 4)$, $y(0) = 1$; $t \in \langle 0; 1{,}2 \rangle$;
i) $y'' = 2 \sin t + y$, $y(0) = 1$; $y'(0) = 1$; $t \in \langle 0; 2 \rangle$;
j) $y'' = 2 \sin t + y - 0{,}5 \cos t$, $y(0) = 1$; $y'(0) = 0$; $t \in \langle 0; 1{,}5 \rangle$;
k) $y'' = 2 \exp t + 2y - y'$, $y(0) = 1$; $y'(0) = 0$; $t \in \langle 0; 1 \rangle$;
l) $y'' = 1/3 \exp -t + 1/3y - y'$, $y(0) = 1$; $y'(0) = 1$; $t \in \langle 0; 4 \rangle$;

Příklad 14.03:

Načtěte data ze zadaného datového souboru (Data14_03.xlsx) a aproximujte polynomem 3. stupně. Výsledek graficky znázorněte a spočtěte sumu čtverců odchylek.

Data pro aproximaci

a)

x	*y*
0,00	7,00
0,20	7,00
0,40	7,00
0,60	6,80
0,80	7,40
1,00	8,00
1,20	10,00
1,40	12,50
1,60	16,50
1,80	22,40
2,00	30,00
2,20	50,00
2,40	65,00
2,60	81,00
2,80	100,70
3,00	124,00

b)

x	*y*
0,00	5,00
0,50	7,20
1,00	8,00
1,50	6,70
2,00	3,00
2,50	-4,50
3,00	-16,00
3,50	-32,00
4,00	-55,00
4,50	-84,00
5,00	-120,00
5,50	-163,50
6,00	-217,00
6,50	-279,50
7,00	-350,00
7,50	-435,60

c)

x	*y*
-10,00	-1950,00
-9,00	-1420,00
-8,00	-980,00
-7,00	-653,00
-6,00	-400,00
-5,00	-220,00
-4,00	-100,00
-3,00	-30,00
-2,00	10,00
-1,00	20,00
0,00	20,00
1,00	21,00
2,00	32,00
3,00	70,00
4,00	140,00
5,00	260,00

d)

x	*y*
-1,00	7,00
-0,90	5,70
-0,80	4,70
-0,70	3,70
-0,60	3,00
-0,50	2,20
-0,40	1,80
-0,30	1,60
-0,20	1,30
-0,10	1,10
0,00	1,00
0,10	1,00
0,20	0,80
0,30	0,80
0,40	0,70
0,50	0,60

e)

x	*y*
1,00	-1,25
1,25	-0,50
1,50	0,52
1,75	1,64
2,00	3,00
2,25	4,70
2,50	6,80
2,75	9,20
3,00	12,30
3,25	15,70
3,50	20,00
3,75	25,00
4,00	31,00
4,25	38,00
4,50	45,40
4,75	54,00

f)

x	*y*
-2,00	-20,00
-1,80	-17,70
-1,60	-15,20
-1,40	-13,00
-1,20	-11,50
-1,00	-9,50
-0,80	-7,80
-0,60	-6,20
-0,40	-5,00
-0,20	-3,50
0,00	-2,00
0,20	-0,70
0,40	0,70
0,60	2,20
0,80	4,00
1,00	5,50

g)

x	*y*	h)	*x*	*y*	i)	*x*	*y*	j)	*x*	*y*	k)	*x*	*y*	l)	*x*	*y*
0,10	-5,00		-5,00	-150,00		1,00	4,20		1,00	-3,50		1,00	-3,50		1,00	-7,00
0,30	-5,00		-4,75	-125,00		1,20	6,60		1,20	-3,10		1,20	-3,80		1,20	-5,00
0,50	-4,70		-4,50	-100,00		1,40	9,60		1,40	-2,50		1,40	-3,60		1,40	-4,70
0,70	-4,00		-4,25	-85,00		1,60	13,00		1,60	-1,30		1,60	-3,00		1,60	-4,00
0,90	-2,00		-4,00	-69,00		1,80	18,00		1,80	0,60		1,80	-1,90		1,80	-2,00
1,10	0,70		-3,75	-52,00		2,00	23,60		2,00	3,00		2,00	0,00		2,00	1,00
1,30	4,50		-3,50	-42,00		2,20	30,30		2,20	6,20		2,20	2,50		2,20	4,50
1,50	10,00		-3,25	-32,00		2,40	38,00		2,40	10,10		2,40	6,00		2,40	10,00
1,70	17,90		-3,00	-24,00		2,60	46,50		2,60	15,30		2,60	10,40		2,60	17,90
1,90	27,00		-2,75	-16,00		2,80	57,00		2,80	21,40		2,80	16,00		2,80	27,00
2,10	39,50		-2,50	-11,00		3,00	68,20		3,00	28,50		3,00	22,20		3,00	39,50
2,30	53,00		-2,25	-7,00		3,20	81,00		3,20	37,00		3,20	30,30		3,20	53,00
2,50	70,00		-2,00	-4,00		3,40	95,80		3,40	46,50		3,40	39,60		3,40	70,00
2,70	90,25		-1,75	-2,00		3,60	111,50		3,60	58,00		3,60	50,10		3,60	90,25
2,90	115,10		-1,50	0,00		3,80	130,00		3,80	70,50		3,80	62,20		3,80	115,10
3,10	140,00		-1,25	1,00		4,00	149,00		4,00	85,00		4,00	76,00		4,00	135,00