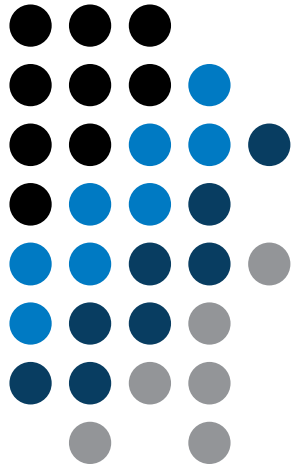
A0B17MTB – Matlab

Část #11

Miloslav Čapek
miloslav.capek@fel.cvut.cz

Katedra elektromagnetického pole
B2-626, Dejvice



Naučíte se …

Datové typy `struct`, `categorical`, `table`

Import / export v Matlabu

Časové funkce

`warning`, `error`, `try-catch`

Základy symbolické matematiky

$$I = \iint_S f(x,y)\mathrm{d}S \qquad f(x,y) = x + y$$

$$x \in (0,2),$$
$$y \geq 0 \wedge y \leq 2 - x$$

Strukturovaná proměnná, `struct`

- data jsou uložena v polích, které jsou soustředěny do jedné struktury
- pojetí je blízké OOP (bez vlastností OOP)

- Př. sklad se zásobami:

```
>> stock(1).id = 1;
>> stock(1).thing = 'fridge';
>> stock(1).price = 750;
>> stock(1).units = 'USD';
>> stock(2).id = 2;
>> stock(2).thing = 'Bowmore_12yr';
>> stock(2).price = 1100;
>> stock(2).units = 'CZK';
>> stock
```

Funkce pro práci se strukturami

- tvorba nového pole
  - přímým příkazem

```
>> stock(1).newField = 'test';
```

  - jméno pole známe jako textový řetězec – funkce

```
>> setfield(stock(1), 'newField', 'test');
```

- nastavení hodnoty pole
  - přímým příkazem

```
>> stock(1).id = 3;
```

  - jméno pole a hodnotu zadám pomocí funkce

```
>> stock(1).('id') = 3;
```

Funkce pro práci se strukturami

- seznam všech polí dané struktury – `fieldnames`

```
>> fieldnames(stock)

ans =

    'id'
    'thing'
    'price'
    'units'
    'test'
```

```
>> fieldnames(stock)
```

- hodnota daného pole

```
>> id2 = stock(2).id
>> id2 = stock(2).('id')
>> id2 = getfield(stock(2),'id')
```

- existuje dané pole?

```
>> isfield(stock,'id')   % = 1
>> isfield(stock,'ID')   % = 0
```

- je daná proměnná struktura?

```
>> isstruct(stock)       % = 1
```

Funkce pro práci se strukturami

- smazání polí

```
>> rmfield(stock,'id')
```

- složitější indexace struktur
  - struktura může mít více úrovní

```
>> stock(1).subsection(1).order = 1
>> stock(1).subsection(2).order = 2
```

  - lze kombinovat s cell strukturami

```
>> stock(1).subsection(3).check = [1; 2]
>> K{1} = stock;
```

  - některé pole lze indexovat pomocí názvu uloženého jako textový řetězec

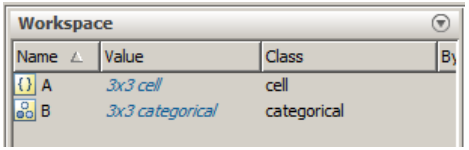
```
>> K{1}(1).subsection(3).('check')(2)
```

Typické využití struktury

- výsledek exportu dat do Matlabu
- všechny komplikované vnitřní proměnné (výjimky, chyby, proměnné callback funkcí, …)
- přiřazením `get(0)` do nějaké proměnné vzniká struktura vlastností
  - `root = get(0)`

Datový typ, `categorical arrays`

- pole kvalitativních dat s hodnotami z konečné množiny diskrétních, nečíselných dat
  - získáme pole nečíselných hodnot, které odpovídají nějaké kategorii (např. kategorii dopravní prostředek budou odpovídat hodnoty: koloběžka, jednokolka, dvoukolák…)
  - hodnoty lze specifikovat jménem (např. hodnoty `'r'`, `'g'`, `'b'`, mohou být příznakem pro jména `'red'`, `'green'`, `'blue'`)
  - `categorical arrays` má ve `Workspace` svou vlastní ikonu



Tvorba categorical arrays

- vytvoření `categorical array` z libovolného pole hodnot (např. `cell array` textových řetězců)

```
>> A = {'r' 'b' 'g'; 'g' 'r' 'b'; 'b' 'r' 'g'} % cell array stringů
>> B = categorical(A)    % categorical arrays
>> categories(B)         % výpis jednotlivých kategorií
```

- široká škála nástrojů pro slučovaní, přidávání, odebírání, přejmenování, řazení,…

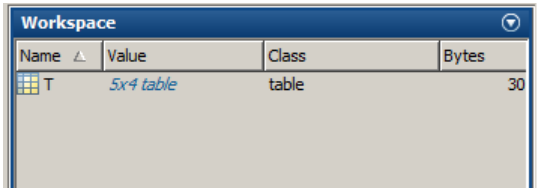
```
>> doc categorical arrays
```

Výhody `categorical arrays`

- přirozenější řazení dat podle názvů (jmen)
  - pozn. pro porovnávání stringů v `categorical arrays` se opět používá logického operátoru eq(==) stejně tak, jako u číselných polí, namísto funkce `strcmp()` používané při práci s textovými řetězci
- matematické řazení stringů
  - nastavení „velikosti“ stringu jinak než abecedně (např. `small < medium < large`)
- data jsou v paměti efektivně udržována
  - data nejsou v paměti uchovávána jako textový řetězec
  - v paměti se jako textový řetězec uchovávají pouze kategorie

Datový typ `tables`

- pole tabulkové formy, které umožňuje mít sloupce různých datových typů a velikostí (podobně jako `cell array`)
  - každý sloupec musí mít stejný počet řádků (stejně jako matice)
  - `tables` má ve `Workspace` svou vlastní ikonu



Tvorba tables

- vytváří se vkládáním jednotlivých vektorů jakožto sloupce tabulky (je třeba dodržet stejnou délku)

```
>> name = {'Miloslav'; 'Filip'; 'Viktor'; 'Pavel'};
>> matlabSemester = [3; 3; 2; 1];
>> favoriteDrink = categorical({'b'; 'm'; 'w'; 'w'}, ...
      {'w'; 'm'; 'b'}, ...
      {'water'; 'milk'; 'beer'});

>> T = table(semester, favoriteDrink, 'RowNames', name)
```

- více >> doc tables array

```
T =

                semester    favoriteDrink
                ________    _____________

    Miloslav    3           beer
    Filip       3           milk
    Viktor      2           water
    Pavel       1           water
```

Výhody `tables`

- Výhodně uložené data různých datových typů
- Přístup k datům přes číselné i jmenné indexování
  - např. vypsat všechny „Kozáky“ kteří v tabulce jsou a zobrazit jejich „věk“
- Možnost uložit metadata do vlastností tabulky
  - např. pro sloupec „věk“ je možné nastavit jednotku na „rok“

Import a export dat

- Matlab podporuje celou řadu souborů
  - `mat`, `txt`, `xls`, `jpeg`, `bmp`, `png`, `wav`, `avi` a další, viz
    - Matlab → Data and File Management → Data Import and Export → Import and Export Basics
  - existují otevřené balíky pro práci např. s `dwg` a podobnými formáty
  - lze načítat i zcela obecný soubor obsahující ASCII znaky

- v tomto kurzu si ukážeme
  - načítání dat ze souboru, načítání obrázku, čtení souboru po řádcích
  - ukládání do souboru, zápis do souboru
  - načítání z Excelu
  - ukládání do Excelu

Import a export dat

- pro velkou skupinu formátů lze import komfortně realizovat
  - starý Matlab: pomocí `File` → `Import Data`
  - nový Matlab: `Home` → `Import Data`

  - příkaz `uiimport` + následující rozhraní
  - stačí i vybraný soubor přetáhnout do okna Workspace

- ukládání do různých formátů viz různé funkce
  - `save`, `dlmwrite`, `xlswrite`, `imwrite`, `audiowrite`, `VideoWriter`

Import z Excelu

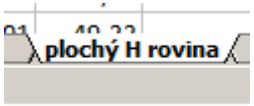
- k načítání využijeme funkci `xlsread`
  - můžete však využít již zmíněnou funkci `uiimport`

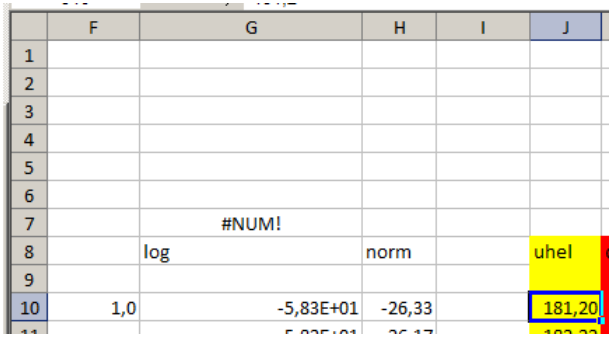
```
>> num = xlsread('directivity.xls','plochý H rovina','J10:K370');
```

jméno souboru
(Matlab ho musí vidět)

jméno listu
v souboru

rozsah buněk





	F	G	H	I	J
1					
2					
3					
4					
5					
6					
7		#NUM!			
8		log	norm		uhel
9					
10	1,0	-5,83E+01	-26,33		181,20

Import z Excelu

420 s ↑

- načtete všechna číselná data z Excel souboru na webu předmětu
  - poté vykreslete sloupec `hodnoty` v závislosti na sloupci `experiment`
  - ověřte si velikost načtených dat

Export do Excelu

- k uložení z Matlabu do Excelu využijeme `xlswrite`
  - např. do souboru `file.xlsx` listu `List1` uložíme data `fx` do řádku `A`

```
>> fx = 1:10;
>> xlswrite('file.xlsx',fx,1,'A1');
```

  - např. do souboru `file2.xlsx`, listu `list` uložíme data `fx` do sloupce `B` od řádku `1`

```
>> fx = 1:10;
>> xlswrite('file2.xlsx',fx','list','B1');
```

Export do Excelu

420 s ↑

- vyhodnoťte funkci $f(x) = \cos(x) + \frac{\cosh(x)}{10}$ v intervalu $x \in \langle -\pi, \pi \rangle$ s krokem 0.01
  - výsledné proměnné *x*, *f*(*x*) vypište do souboru `Excel_file.xlsx` do 1. listu, proměnná *x* bude ve sloupci `A`, proměnná *f*(*x*) bude ve sloupci `B`
  - ověřte, zda jsou data zapsaná do sešitu v pořádku

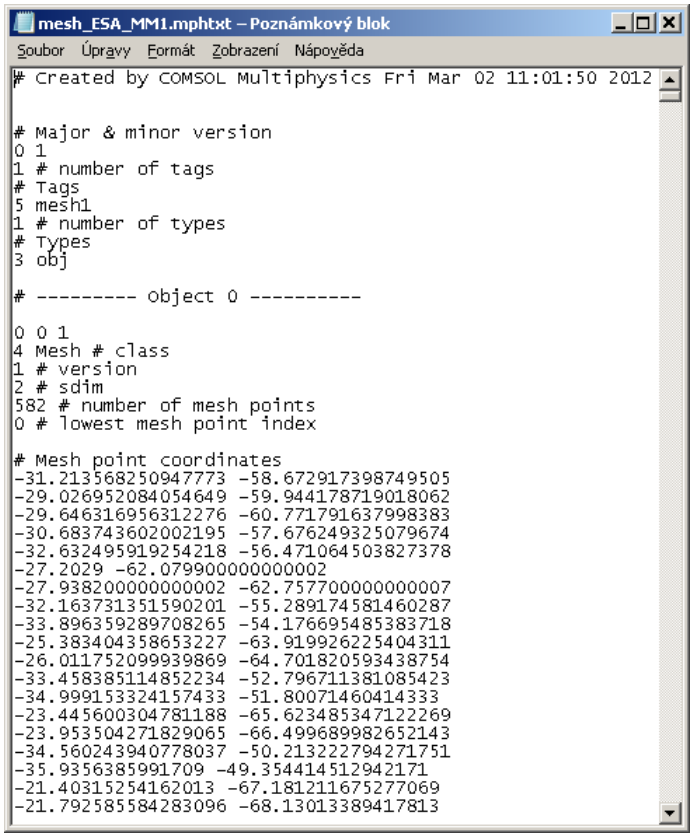
Binární načítání ze souboru #1

- využijeme, co jsme se naučili dříve (`while`, `str2double`, …)
  - navíc je potřeba soubor otevřít (`fopen`) a poté uzavřít (`fclose`)

```
>> fid = fopen('mesh_ESA_MM1.mphtxt');
```

```
% alokace
while ~feof(fid)
    % načítání
end
```

```
>> fclose(fid);
```



Binární načítání ze souboru #2

Běh programu

mesh_ESA_MM1.mphtxt – Poznámkový blok

Soubor Úpravy Formát Zobrazení Nápověda

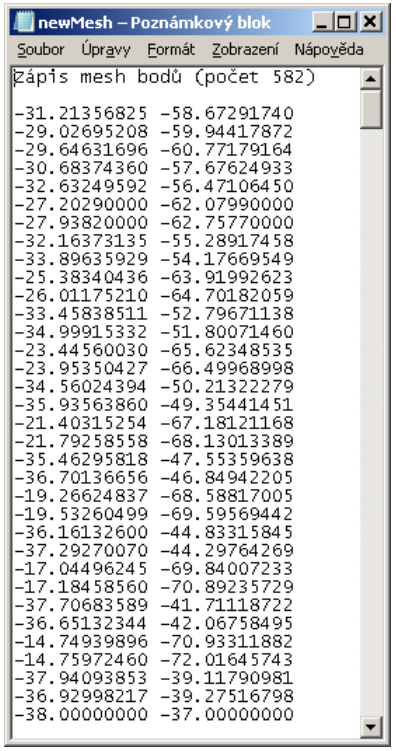
```
# Created by COMSOL Multiphysics Fri Mar 02 11:01:50 2012


# Major & minor version
0 1
1 # number of tags
# Tags
5 mesh1
1 # number of types
# Types
3 obj

# --------- Object 0 ----------

0 0 1
4 Mesh # class
1 # version
2 # sdim
582 # number of mesh points
0 # lowest mesh point index

# Mesh point coordinates
-31.213568250947773 -58.672917398749505
-29.026952084054649 -59.944178719018062
-29.646316956312276 -60.771791637998383
-30.683743602002195 -57.676249325079674
-32.632495919254218 -56.471064503827378
-27.2029 -62.079900000000002
-27.938200000000002 -62.757700000000007
-32.163731351590201 -55.289174581460287
-33.896359289708265 -54.176695485383718
-25.383404358653227 -63.919926225404311
-26.011752099939869 -64.701820593438754
-33.458385114852234 -52.796711381085423
-34.999153324157433 -51.80071460414333
-23.445600304781188 -65.623485347122269
-23.953504271829065 -66.499689982652143
-34.560243940778037 -50.213222794271751
-35.9356385991709 -49.354414512942171
-21.40315254162013 -67.181211675277069
-21.792585584283096 -68.13013389417813
```

```
>> size(Data)

ans =

   582     2
```

Zapisování do souboru #1

- pokusíme se zapsat do souboru proměnnou `Data`, přičemž úvodní řádka bude obsahovat informační text (hlavičku)

Zapisování do souboru #2



Upozornění v Matlabu – `warning`

- varování v Matlabu zobrazíme pomocí funkce `warning`

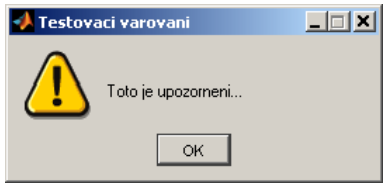
```
a = 1e3;
if a > 1e2
    warning('Input coefficient has to be smaller than 10!');
end
```

  - funkce je využívána i Matlabem, lze tedy např. dočasně deaktivovat vybraná vnitřní upozornění

- funkce `lastwarn` vrátí poslední varování, které bylo aktivováno

- v GUI s výhodou využijeme funkce `warndiag`
  - jde však skutečně jen o výpis, více viz předchozí přednáška

```
f = warndlg('Toto je upozorneni...', ...
    'Testovaci varovani', 'modal');
```



Chybový výpis v Matlabu – `error`

- chybové hlášení (červeně) získáme pomocí funkce `error`

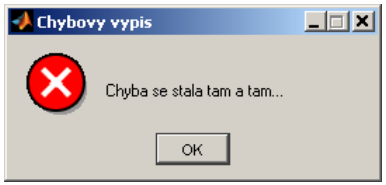
```
a = 100;
if a > 10
    error('Input has to be equal of smaller than 10!');
end
```

  - ukončuje běh programu
  - můžeme připojit identifikátor

```
error('Input has to be equal of smaller than 10!');
```

- v GUI s výhodou využijeme funkce `warndiag`
  - jde však skutečně jen o výpis, více viz předchozí přednáška

```
f = errordlg('Chyba se stala tam a tam...', ...
    'Chybovy vypis', 'modal');
```



Zachycení chyby #1

- využijeme zejm. tam, kde může dojít k neočekávané události
  - obecně práce se soubory (načítání, ukládání)
  - vyhodnocování zapouzdřeného kódu (funkce eval, assignin)
  - práce s proměnnými, jejichž vlastnosti (např. velikost) ještě neznáme
  - vyhodnocování kódu nad objektem, který již potenciálně neexistuje (GUI)
  - …

```
try
    % regular piece of code
catch
    % code that is evaluated if the regular code failed
end
```

- lze (resp. bychom měli) využít i identifikátoru dané chyby

Zachycení chyby #2

- identifikátor lze využít pro rozhodnutí, co s chybou uděláme
  - př.: pokud jde o chybu násobení způsobenou různou velikostí vektorů, lze vypsat vlastní varování
  - rovněž lze chybu kdykoliv později opět „vyhodit“, a to buď jako vyvolání poslední vzniklé chyby, nebo jako novou chybu s daným identifikátorem

```
try
    A = [1 1 1];
    B = [1 1];
    c = A.*B;
catch exc
    if strcmp(exc.identifier,'MATLAB:dimagree')
        disp('Pozor na velikost vektoru!');
    end
    % throw;
    % rethrow;
end
```

Časové funkce v Matlabu

- je jich celá řady, na měření běhu funkce nám stačí ale jediná

Funkce	Popis
`tic – toc`	změří délku časového úseku mezi klíčovými slovy `tic` a `toc`
`clock`	vrací šesti prvkový vektor `[year month day hour minute seconds]`
`date`	vrací datum ve formátu `dd-mmm-yyyy`, proměnná je typu `char` (text)
`etime`	vrací časový úsek mezi intervaly `t1` a `t2`, `etime(t2,t1)`
`cputime`	
`now`	vrací aktuální datum a čas jako celočíselnou hodnotu

```
>> tic
>>   %% program
>> toc
```

```
>> t0 = tic;
>>   %% program
>> t1 = toc(t0)
```

Časové funkce v Matlabu – příklad

- jak tedy změřit, jak dlouho trvá výpočet programu?
  - časově náročnější program × velmi rychlý program

```
tic
    % volani programu
toc
```

```
tic
    for k = 1:100
        % volani programu
    end
toc
```

  - další možnosti – která je nejlepší?
  - Mathworks doporučuje funkce `tic-toc` zejm. pro ≥P4@hyperthreading

```
t0a = tic;
fft(x);
toc(t0a)
```

```
t0b = clock;
fft(x);
etime(clock,t0b)
```

```
t0c = cputime;
fft(x);
e = cputime - t0c
```

Časové funkce v Matlabu – specialitky

- konverze mezi jednotlivými způsoby vyjádření datumu v Matlabu
  - `datavec`, `datanum`, `datastr`
  - např. takto převedeme datum do standardní podoby

```
>> datevec(now)
```

- den v týdnu:

```
>> weekday(date)
```

  - pozor, je počítáno podle US (sobota ~ poslední den v týdnu)

- poslední den v měsíci:

```
>> eomday(2014,1:12)
```

- kalendář

```
>> calendar
```

  - pozor, poslední den v týdnu je opět sobota!

Časové funkce v Matlabu

420 s ↑

- zkuste implementovat vybrané časové funkce do Vašeho projektu

Vyšší matematika

- rozlišujeme dva různé přístupy
  - symbolická matematika
  - numerická matematika
    - numerické chyby
  - možná klasifikace: analytický výsledek v principu umožňuje získat výsledek na nekonečný počet desetinných míst
- v Matlabu existuje celá řady technik (symbolické i numerické)
  - ukážeme si pouze vybrané techniky

Handle funkce – opakování

- umožňuje nepřímé volání funkce
- do `handle` si uložíte referenci na funkci

```
handle1 = @jmeno_funkce
handle2 = @(args) jmeno_funkce
```

- jde o velice mocný, byť již komplikovanější, nástroj
  - umožňuje volat funkci i z míst, kde ji Matlab normálně nevidí
  - function handle je v Matlabu datový typ (viz `whos`)

```
>> clear,clc;
>> doc function_handle

>> fxy = @(x,y) x^2 + y^2 - 5
>> fxy(2,-2)

>> fcos = @(alpha) cos(alpha)
>> fcos(pi)
```

Polynomy #1

- reprezentace polynomů v Matlabu

$$\mathrm{P} = C_n x^n + C_{n-1} x^{x-1} + \ldots + C_1 x + C_0 = \begin{bmatrix} C_n & C_{n-1} & \ldots & C_1 & C_0 \end{bmatrix}$$

```
>> x = roots([1 0 -1]);
>> x1 = x(1)
>> x2 = x(2)
```

- příkaz `roots` najde kořeny polynomu
- vyhodnocení polynomu: `polyval`

```
>> x = 2
>> p1 = 3*x^5 - 7*x^3 + 1/2*x^2 - 5)
>> polyval([3 0 -7 1/2 0 -5],2)
```

- násobení polynomů: `conv`

$$\mathrm{A}_1 = x - 1$$

$$\mathrm{A}_2 = x + 1$$

$$\mathrm{A}_1 \cdot \mathrm{A}_2 = (x-1)\cdot(x+1) = x^2 - 1$$

```
>> A1 = [1 -1]
>> A2 = [1 1]
>> conv(A1,A2)
% = [1 0 -1]
```

Polynomy #2

- dělení polynomů: `deconv`

```
>> deconv([1 0 -1],[1 1]) % = [1 -1]
```

$$\frac{x^2-1}{x+1}=\frac{(x-1)\cdot(x+1)}{x+1}=x-1$$

- další příkazy (matematika3 a dále), pouze výběr několika:
  - `residue`: vypočte s podílu dvou polynomů rezidua
  - `polyfit`: aproximace dat polynomem řádu `n`
  - `polyint`: integruje polynom
  - `polyder`: derivuje polynom

```
>> S = [1 1];
>> T = polyint(S) % = [0.5 1 0]
>> U = polyder(T) % = S = [1 1]
>> polyder(U)     % = 1
```

$$\int (x+1)\mathrm{d}x=\frac{1}{2}x^2+x \qquad \frac{\mathrm{d}\left(\frac{1}{2}x^2+x\right)}{\mathrm{d}x}=x+1$$

Polynomy #3

- roznásobení polynomů

$$P1 = A + Bx \qquad P2 = 4x^2 + 2x - 4$$

```
>> syms A B x
>> P1 = A + B*x;           % zadáni 1. polynomu
>> P2 = 4*x^2 + 2*x - 4;   % 2. polynomu
>> P0 = P1*P2;             % součin
>> P  = expand(P0)         % roznásobení
```

- pozn.: funkce `expand` vyžaduje Symbolic Math Toolbox

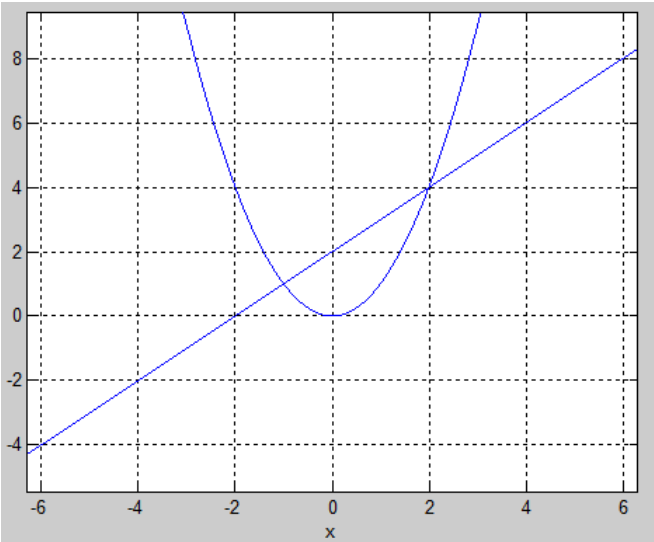
x = ?: f(x) == g(x)

- máme dvě funkce, chceme analyticky nalézt kdy jsou si rovny

$f(x) = x^2$ $g(x) = x + 2$ $x = ? : \{f(x) = g(x)\}$

zadání

```
>> clear,clc;
>> syms x y;
>> f = x^2;
>> g = y + 2;
```



řešení

```
>> x0 = solve(f-g) % = 2; -1
```

ověření

```
>> ezplot(f);
>> hold on;
>> grid on;
>> ezplot(g);
```

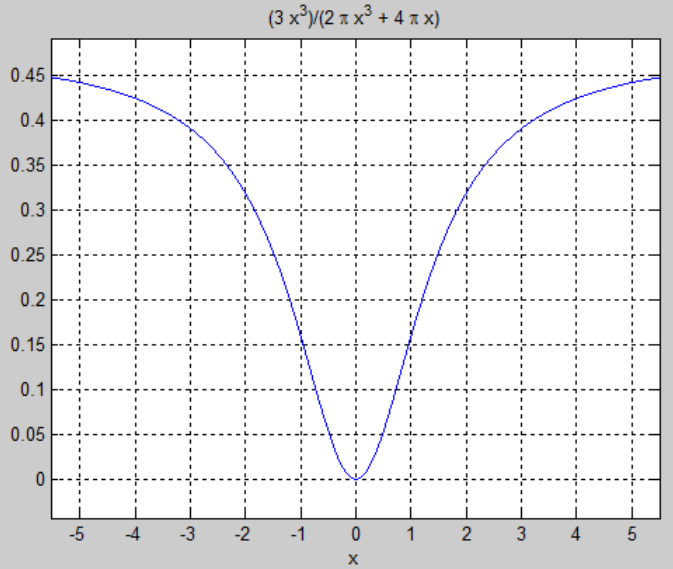
Limita

- chceme vyšetřit limity funkce

$$f(x)=\frac{3x^3}{2\pi x^3+4\pi x} \qquad f(x)=\frac{3}{2\pi}\left(\frac{x^2}{x^2+2}\right) \qquad \lim_{x\to-\infty} f(x)=\lim_{x\to\infty} f(x) \overset{L'H.P.}{=} \frac{3}{2\pi}=0.4775$$

zadání

```
>> clear, clc, close all;
>> syms x real;
>> f = 3*x^3/(2*pi*x^3 + 4*pi*x)
```



řešení

```
>> lim1 = limit(f, x, -inf)
>> lim2 = limit(f, x, inf)

>> double(lim1) % = 0.4775
>> double(lim2) % = 0.4775
```

ověření

```
>> figure;
>> ezplot(f);
>> grid on;
```

Derivace funkce #1

- zkusme si nyní L'Hospitalovo pravidlo na předchozí příklad
  - ve funkci $f(x)$ figuruje x^3, budeme tedy 3× derivovat (zvlášť jmenovatel a čitatel) podle x

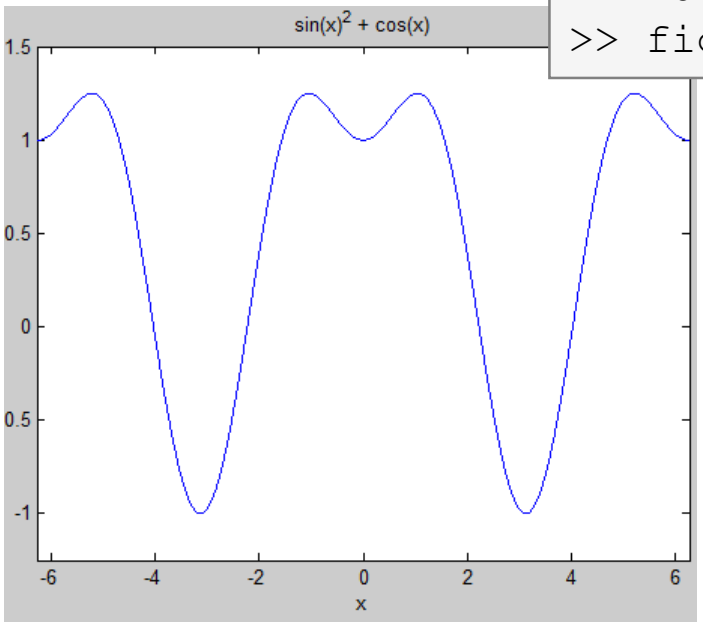
$$f(x) = \frac{3x^3}{2\pi x^3 + 4\pi x} \qquad f_1(x) = 3x^3 \qquad f_2(x) = 2\pi x^3 + 4\pi x$$

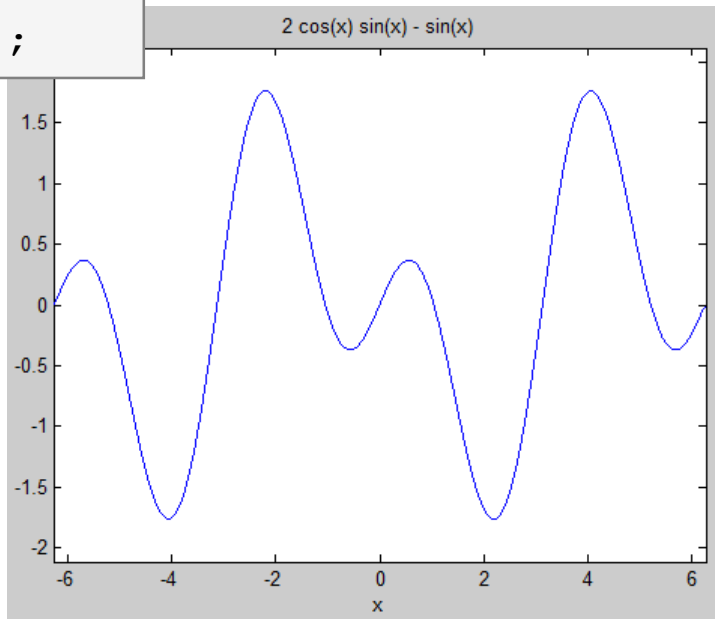
```
>> f1 = 3*x^3;
>> f2 = 2*pi*x^3 + 4*pi*x;
>> A1 = diff(f1,3)
>> A2 = diff(f2,3)
>> double(A1/A2) % = 0.4775
```

Derivace funkce #2

- derivujte následující funkci podle x $f(x) = \sin^2(x) + \cos(x)$
  - porovnejte výsledky, vykreslete

```
>> clear, clc;
>> syms x;
>> f = sin(x)^2 + cos(x);
>> figure; ezplot(f);
>> fd = diff(f);
>> figure; ezplot(fd);
```





Integrace #1

- nejprve symbolicky derivujme funkci $f(x) = \sin(x) + 2$
- druhou derivaci uložte jako funkci *g*, porovnejte výsledky
- nyní integrujte funkci *g* (1×, 2×), dostanete opět funkci *f* ?
  - pro naše účely ignorujte integrační konstanty

```
>> clear, clc;
>> x = sym('x');

>> f = sin(x) + 2
>> figure; ezplot(f);

>> fd = diff(f)
>> figure; ezplot(fd);

>> fdd = diff(f,2)
>> figure; ezplot(fdd);
```

```
>> g = fdd;
>> gi = int(g)
>> figure; ezplot(gi);

>> gii = int(gi);
>> err = f-gii

figure;
subplot(1,2,1);
ezplot(f);
subplot(1,2,2);
ezplot(gii);
```

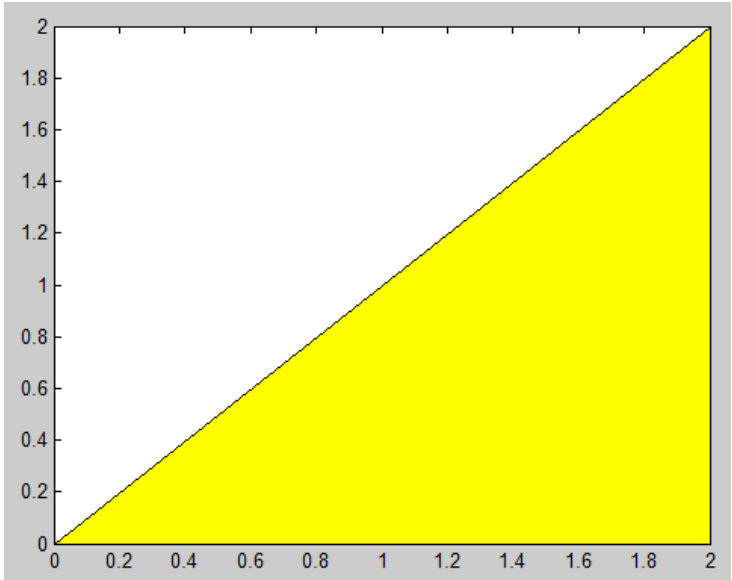
Integrace #2

- integrál funkce $f(x) = x$
  - zobrazte funkce a vyčíslete následující integrál $I = \int_0^2 f(x)\mathrm{d}x$
  - vypočtěte v ruce, nakreslete průběh funkce
  - vypočtěte neurčitý integrál v Matlabu
  - vyčíslete určitý integrál v mezích (0, 2), použijte např. funkci `int`

$$I = \int_0^2 f(x)\,\mathrm{d}x = \int_0^2 x\,\mathrm{d}x = \left[\frac{x^2}{2}\right]_0^2 = \frac{4}{2} - 0 = 2$$

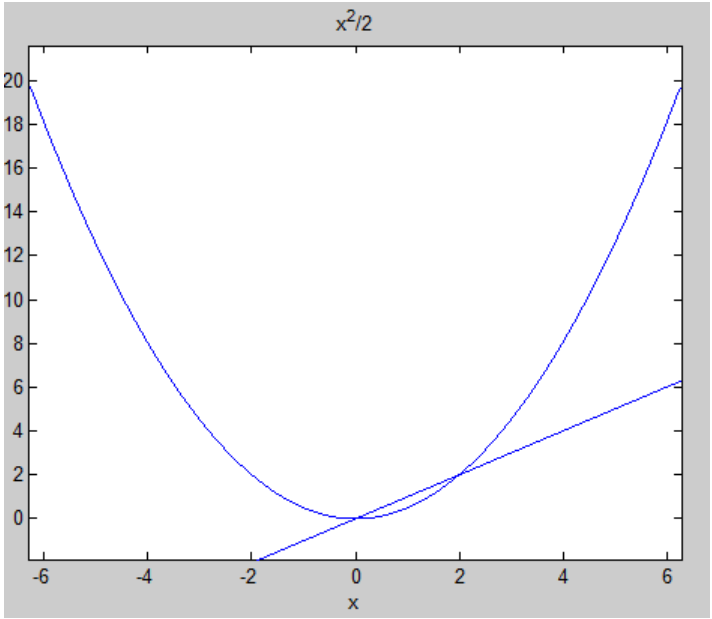
$$I = \frac{2 \cdot 2}{2} = 2$$

```
>> fill([0 2 2],[0 0 2],'y')
```



Integrace #3

- integrál funkce




```
>> clear, clc;
>> syms x;
>> f = x;
>> g = int(x);

>> figure;
>> ezplot(f);
>> hold on;
>> ezplot(g);

>> int(f,x,0,2)                  % = 2
>> polyarea([0 2 2],[0 0 2])     % = 2

% ALE!:
>> f = @(x) x   % function_handle!
>> I = quad(f,0,2)               % = 2
```

Numerická integrace #1

- numericky integrujeme vždy, když neznáme uzavřený tvar řešení (analytický), což je v technice často (skoro vždy)

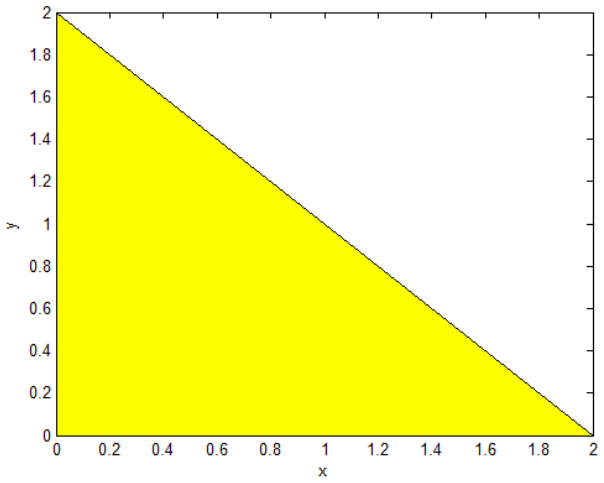
- lze napsat různé numerické metody integrace, více viz odborná literatura
- můžeme však využít funkcí Matlabu
  - `quad`, `dblquad`, `triplequad` a další
    - v novém Matlabu jde o funkce `integral`, `integral2`, `integral3`
  - definujeme funkci, kterou chceme integrovat (napíšeme vlastní uživatelskou funkci nebo použijeme *function handle*)

Numerická integrace #2

- řešme následující integrál na tomto intervalu

$$x \in (0,2),$$
$$y \geq 0 \wedge y \leq 2 - x$$

$$I = \iint_S f(x,y)\mathrm{d}S \qquad f(x,y) = x + y$$



$$I = \int_0^2 \int_0^{y_{\max}} f(x,y)\mathrm{d}x\mathrm{d}y = \int_0^2 \int_0^{2-x} (\mathrm{x}+\mathrm{y})\mathrm{d}x\mathrm{d}y = \int_0^2 \left(x\left[\mathrm{y}\right]_0^{2\text{-}\mathrm{x}} + \left[\frac{y^2}{2}\right]_0^{2\text{-}\mathrm{x}} \right)\mathrm{d}x$$

$$= \int_0^2 \left(x(2-x) + \frac{(2-x)^2}{2} \right)\mathrm{d}x = \int_0^2 \left(2x - x^2 + 2 - 2x + \frac{x^2}{2} \right)\mathrm{d}x$$

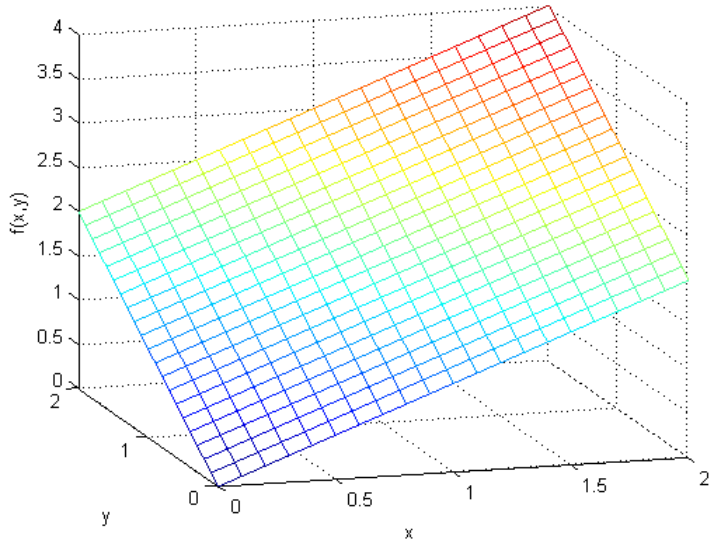
$$= \int_0^2 \left(2 - \frac{x^2}{2} \right)\mathrm{d}x = 2\left[x\right]_0^2 - \frac{1}{2}\left[\frac{x^3}{3}\right]_0^2 = 4 - 8\cdot\frac{1}{6} = \frac{12-4}{3} = \frac{8}{3} = \underline{\underline{2.\overline{666}}}$$

Numerická integrace #3

- řešme následující integrál na tomto intervalu

$$x \in (0,2),$$
$$y \geq 0 \wedge y \leq 2 - x$$

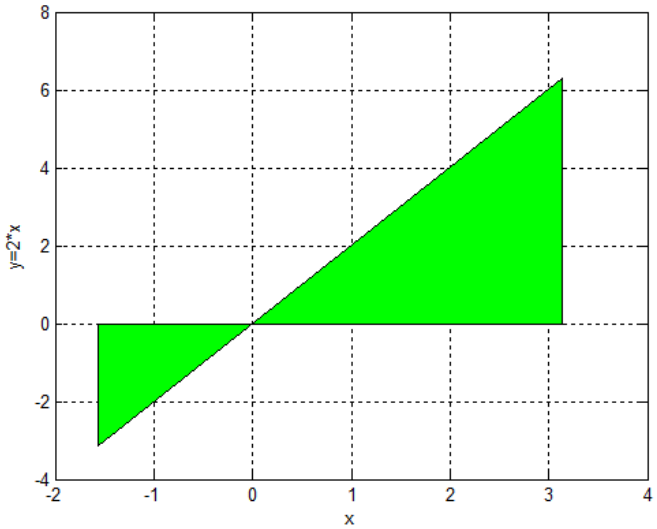
$$I = \iint_S f(x,y)\mathrm{d}S \qquad f(x,y) = x + y$$



Numerická integrace #4

- můžeme pracovat i s externími skripty, máme např. „složitější“ vztah, který nechceme zpracovat jako handle:

$$I = \int_x f(x)\mathrm{d}x = \int_{-\frac{\pi}{2}}^{\pi} 2x\mathrm{d}x = 2\int_{-\frac{\pi}{2}}^{\pi} x\mathrm{d}x = 2\left[\frac{x^2}{2}\right]_{-\frac{\pi}{2}}^{\pi} = \pi^2 - \frac{\pi^2}{4} = \underline{\underline{\frac{3}{4}\pi^2}}$$



Numerická integrace #1

- obecně problém derivace (nemůžeme jít k nule)

$$\lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

- využívají se různě sofistikované (a různě složité) numerické metody

- stránky, které toto komplexně řeší:
  - `http://www.matrixlab-examples.com/derivative.html`

Závěrečná poznámka

- pokud máte symbolických výpočtů více nebo narážíte na možnosti Matlabu, zkuste jiné matematické prostředí (pro analytické výpočty se hodí zejm. Maple, Mathematica)

- na numerické výpočet je však Matlab excelentní volbou (byť Mathematica má i numerické jádro výborné)

Vyšší matematika

- práce s polynomy
  - http://www.matrixlab-examples.com/polynomials.html
- jednoduchá a dvojitá integrace (symbolická, do příkladů)
  - http://www.matrixlab-examples.com/definite-integrals.html
- derivace (numerická)
  - analytické zadání:
    - http://www.matrixlab-examples.com/derivative.html
  - numerické zadání
    - ruční derivace

Shrnutí funkcí is*

- hvězdička nahrazuje celou řadu funkcí
  - výraz potom vrací logickou hodnotu (`true` / `false`)
- namátkou ty zajímavé (některé mají i více parametrů)

Funkce	Popis
`ischar`	obsahuje pole pouze textové řetězce? (probereme je za chvilku)
`isempty`	testuje zda je dané pole prázdné
`isfinite`	obsahuje pole konečně velké elementy?
`isnan`	obsahuje pole hodnoty `NaN`?
`isletter`	detekuje zda jsou prvky v poli písmena (a-z, A-Z)
`islogical`	obsahuje pole hodnoty `true` / `false`
`isnumeric`	obsahuje pole číselné hodnoty (reálné, komplexní skaláry, matice, vektory, celá pole)
`isreal`	jsou prvky pole reálná čísla
`isstudent`	je spuštěna verze Matlabu licencována jako studentská verze?
a další	koukněte na `>> doc is*`

Funkce `is*`

- vyzkoušejte si následující příklady
  - zvažte, za jakých případů by se Vám mohli hodit…

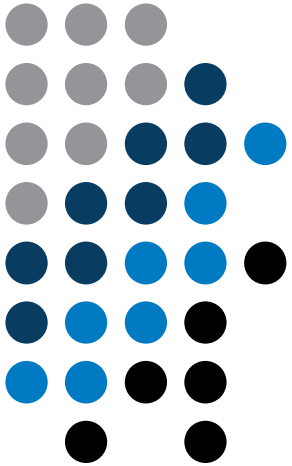
```
>> A = 'pi5_7';
>> B = pi;
>> C = [Inf NaN 5.31 true false pi];
>> D = [[] []];
>> ischar(A), ischar(B),
>> isstudent, isunix, computer,
>> isnan(A)
>> isnan(C)
>> ischar(A), ischar(B),
>> isempty(C), isempty(D),
>> isfinite(A), isfinite(C),
>> isletter(A),
>> islogical(C), islogical([true false]),
>> isnumeric(A), isnumeric(C)
```

Probrané funkce

`tic, toc, clock, date, etime, cputime, now`	časové funkce, měření rychlosti kódu	
`datevec, weekday, eomday, calendar`	časové funkce (dni v týdnu, měsíci, kalendář)	
`warning, error, try-catch`	varování, chybové výpisy, zachytávání chyb	●
`throw, rethrow`	generování chyby	●
`cell, celldisp, cellplot`	proměnná `cell` (alokace, zobrazení)	
`setfield, fieldnames, getfield, rmfield`	práce se strukturou	
`isfield, isstruct`	má proměnná dané pole?, je typem `struct`?	
`uiimport`	rozhraní na import dat do Matlabu	●
`xlsread, xlswrite`	čtení z Excelového šešitu, zápis do Excelu	●
`fopen, feof, fclose, fgetl`	otevření souboru, test na konec, uzavření, čtení řádky	●
`sym, syms`	tvorba symbolické proměnné (proměnných)	
`roots, polyval, conv, deconv`	práce s polynomy 1	
`residue, polyfit, polyder, polyint, expand`	práce s polynomy 2	
`solve`	řešení symbolických rovnic / nerovnic	●
`limit, diff, int`	limita funkce, derivace, integrace funkce	
`ezplot`	vykreslení funkce zadané symbolicky	
`quad (integral), quad2d (integral2)`	numerická integrace	●

Děkuji!

ver. 2.0 (06/09/2014)
Miloslav Čapek
miloslav.capek@fel.cvut.cz



Materiál vytvořen v rámci předmětu A0B17MTB.