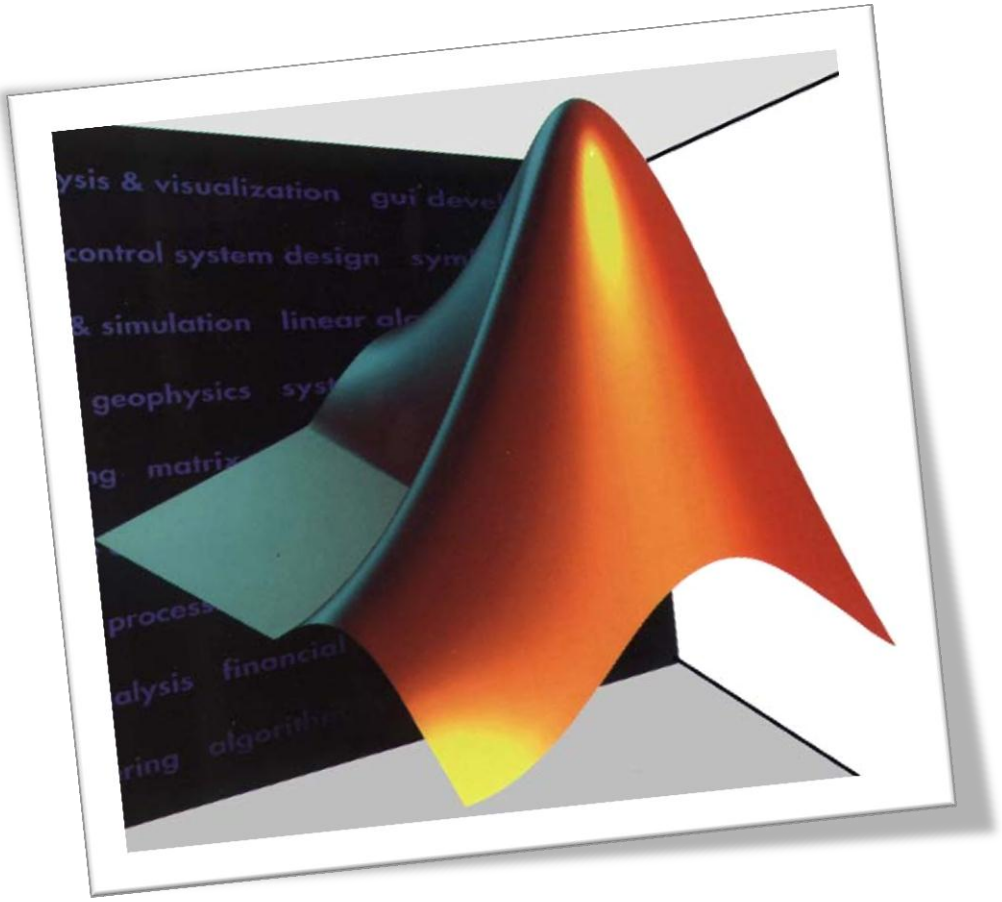
ÚVOD DO PROGRAMOVÉHO PROSTŘEDÍ MATLAB

KREJČÍ A., REITINGER J., TIHELKA D. & VANĚK J.



FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Verze: 5.2.2014

Obsah

1 ÚVOD

Název programu vznikl z prvních písmen z původně dlouhého názvu **MAT**rix **LAB**oratory (ve volném překladu maticová laboratoř). Jedná se o interaktivní programové prostředí a skriptovací programovací jazyk. Program je vyvíjen společností MathWorks a v září 2013 vyšla zatím poslední verze R2013b. Společnost nabízí uvedený SW ve verzích 32bit a 64bit pro Windows i Linux.

Matlab nabízí celou řadu funkcí jako počítání s maticemi, vykreslování 2D a 3D grafů funkcí, implementaci algoritmů, počítačovou simulaci, analýzu a prezentaci dat i vytváření aplikací včetně uživatelského rozhraní.

Původně byl jazyk určen pro matematické účely, ale postupem času docházelo k přidávání stále nových funkcí až do dosavadní podoby. Dnes je Matlab využíván v široké škále aplikací, hlavními oblastmi využití jsou technické obory a ekonomie.

1.1 Stručně o historii

Matlab byl vytvořen profesorem Cleverem Molerem na konci sedmdesátých let. Tento profesor působil na univerzitě na katedře informačních technologií v Novém Mexiku. Navrhl Matlab, aby studenti mohli využívat LINPACK a EISPACK bez nutnosti se učit programovací jazyk Fortran, který se mnoho let využíval pro matematické výpočty. Matlab se velmi rychle rozšířil i na další univerzity.

V roce 1983 se o Matlab začal zajímat Jack Little, který v softwaru viděl značný ekonomický potenciál, do této doby byl Matlab zdarma. Jack Little přepsal Matlab do jazyka C, přidal další funkce a knihovny a v roce 1984 založili Little, Moller a Steve Bergert společnost MathWorks. První verze pro PC TX byla vydána koncem roku 1985. Elementárním problémem byl nedostatek paměti a z toho plynoucí omezení na maximální velikost matic. Po příchodu PC AT společnost rychle zareagovala a vydala novou verzi pro tento počítač.

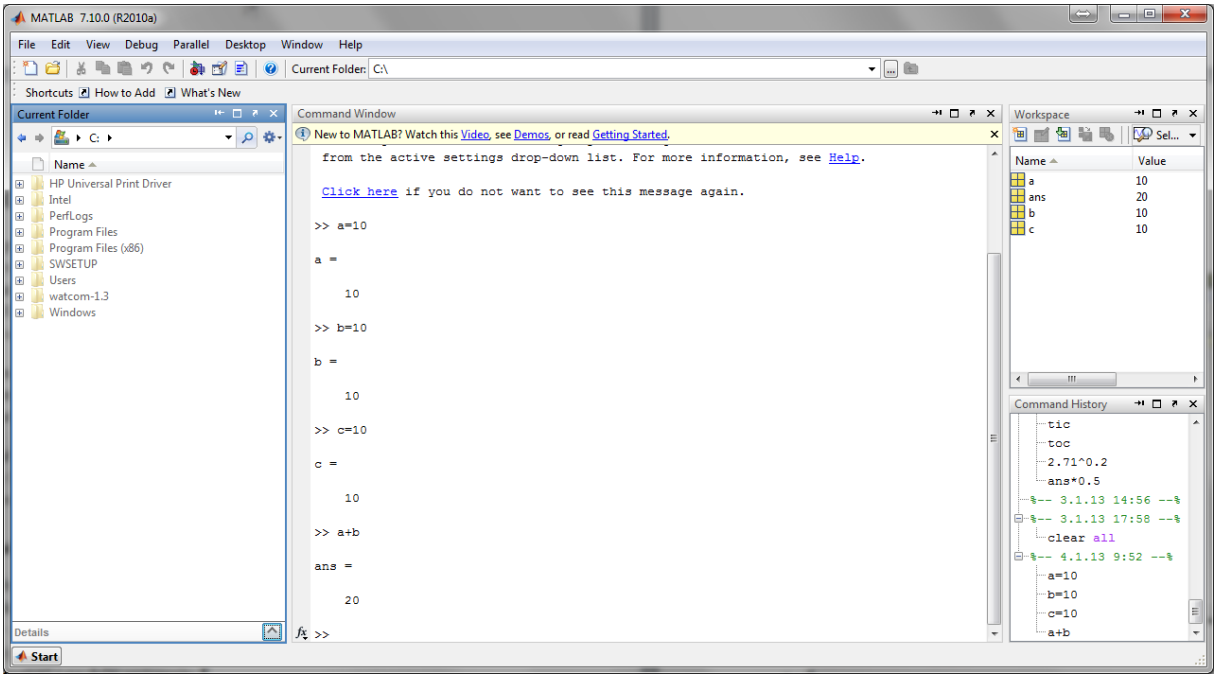
1.2 Vlastnosti

Programovací jazyk Matlab je integrované prostředí, které je určeno pro vědeckotechnické účely, paralelní výpočty, simulace atd.

Typické oblasti použití jsou:

- Tvorba algoritmů
- Inženýrské výpočty
- Modelování a simulace
- Analýza dat
- Tvorba aplikací (včetně GUI)

2 „PRVNÍ SPUŠTĚNÍ“



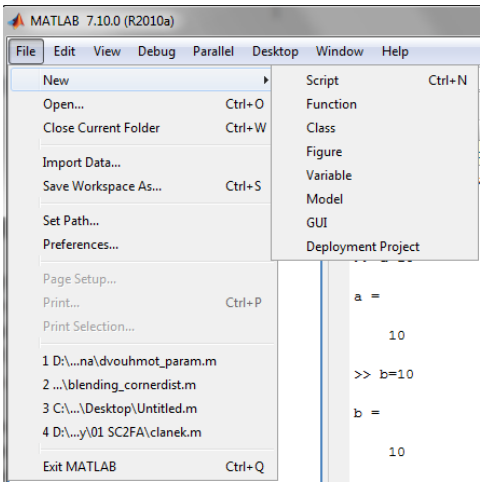
- Current folder – pracovní složka
- Command Window – práce v dialogovém režimu („používání Matlabu jako kalkulačky“), odeslání příkazu pomocí ENTER
- Workspace – zobrazuje všechny dostupné proměnné pracovního prostředí, umožňuje jejich mazání či zobrazení
- Command History – přehled použitých příkazů

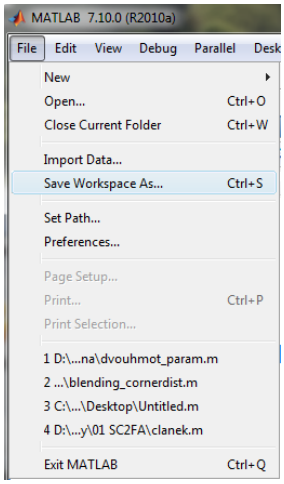
Poznámka: okno *Command History* nemusíme používat, protože v *Command Window* lze **listovat použitými příkazy** s použitím šipek (nahoru, dolu). Pokud před stiskem šipky napíšeme začátek hledaného příkazu (alespoň jeden znak), listuje se jen v názvech těch příkazů, které začínají napsaným textem.

Další klávesy v příkazovém řádku:

- Ctrl-C – přerušení výpočtu
- Esc – smazání obsahu řádky
- Šipky vlevo, vpravo – standartní pohyb v řádku
- clc – smaže obrazovku příkazového řádku

Matlab umožňuje psát programy a ty pak spouštět. Zdrojové kódy se píší do tzv. *m-filů*, které nalezneme v pravém horním menu: *File/New/Script* či zkrat CTRL+N.





Data uložená ve Workspace lze uložit, pomocí *File/Save Workspace As*, po opětovném otevření programu lze data načíst a dále s nimi pracovat. (pozn. Command Window se neuloží, proto používat m-file).

Základní příkazy:

- *Clear seznam proměnných oddělených mezerou* – smaže proměnné uvedené za příkazem *clear*, pokud nejsou uvedeny, smaže všechny
- *Who, whos* – vypíše seznam proměnných a jejich velikost (whose)
- *Help nějaký příkaz* – např: *help plot*

Speciální znaky a operátory:

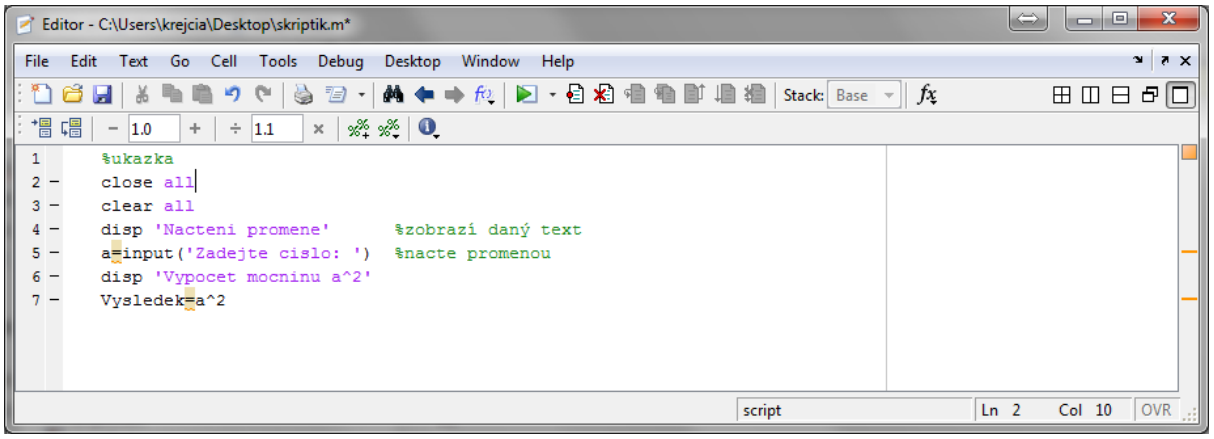
%	komentář, platí do konce řádky
.	prvková operace
,	oddělovač indexů
;	konec řádky v matici, konec příkazu s potlačením výstupu na obrazovku
:	dvojtečkový operátor, generování vektorů
()	závorky výrazů a indexování matic
[]	maticové závorky
=	operátor přiřazení
+ - * / \	matematické operátory
== ~=	operátor rovná se, nerovná se
< >	operátor porovnání
<= >=	operátory porovnání
^	mocnina

Detailní seznam: *help ops*

2.1 M-File základní pravidla (skripty)

Obdobně jako v různých programovacích jazycích píšeme jednotlivé příkazy na řádky, tedy: *co příkaz, to jedna řádka*. Matice a vektory můžeme zadávat i na více řádků.

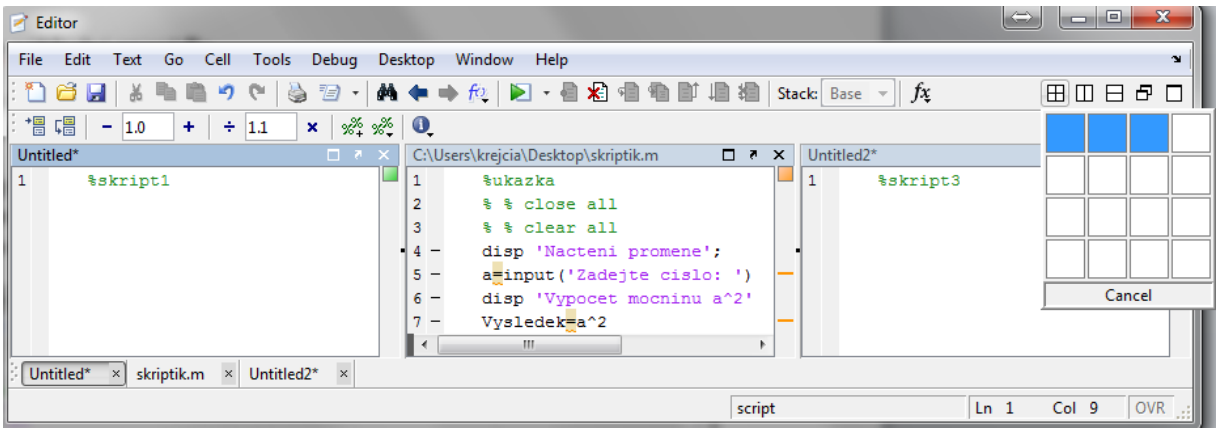
M-file (program) se uloží pod nějakým jménem, spustí se napsáním příslušného jména, musíme být ale ve složce, kam jsme program uložili. Spuštění je samozřejmě možné i ze skriptu, pomocí klávesy F5 či spuštěním z menu skriptu v nabídce *Debug*.



Proměnné, které jsme ve skriptu naplnili, zůstávají v Matlabu definované (pokud ukončíme celý Matlab, dojde k jejich smazání)! Pozor tedy při psaní, abychom nepoužili proměnnou s jinými hodnotami, než chceme. Řešení je uvést příkaz *clear all* na začátku skriptu.

Užitečné rady:

- Jak již bylo uvedeno, skript rychle a efektivně spustíme pomocí klávesy F5
- Pokud chceme zakomentovat nějakou sekci programu, použijeme CTRL+R pro zakomentování a CTRL+T pro odkomentování – sekce musí být označena, příkazy lze též nalézt v menu *Text*
- V menu v záložce *Windows* nalezneme řadu zobrazení pro více oken s m-file, rychlé zobrazení je též v levém horním rohu



run('skript')	spustí skript zadaný jménem (nikoli volání funkce!), zadává se bez přípony .m, musíme být v adresáři, kde se nachází skript.
error('text')	zobrazí chybovou zprávu a ukončí skript
warning('text')	zobrazí varovnou zprávu, ale pokračuje
lasterr	proměnná s poslední chybovou zprávou
lastwarn	proměnná s poslední varovnou zprávou

3 PROMĚNNÉ

Proměnná je objekt, který má svůj název, typ a hodnotu (obsah).

Název proměnné může obsahovat až 31 znaků. Musíme dodržovat následující pravidla - jsou **povoleny POUZE tyto znaky**: písmena anglické abecedy (a-z, A-Z), číslice (0-9) a podtržítko (_). Číslicí název začínat nesmí.

V názvech jsou rozlišována velká a malá písmena (tzv. vlastnost **case-sensitive**), tedy proměnná `pomocna` může existovat současně s proměnnou `Pomocna` i třeba `PomocnA`. Každá z uvedených proměnných má svou vlastní hodnotu.

Správné názvy: `z, Z, x1, x2, jednotkova_matice, gx`

Chybné názvy: `1x, jednotkova matice, rovnice.prvni, pom-4`

Pro vytvoření proměnné se používá výraz: název_proměnné = výraz.

Desetinná čísla zadáváme s desetinou tečkou (ne čárkou) nebo pomocí zlomku, pokud je před desetinou čárkou nula, lze ji vynechat (1.5, 3/2, 0.5, .5).

Pokud umístíme na konec příkazu středník, nedojde k vypsání dat.

4 POLE, MATICE, DVOJTEČKOVÝ OPERÁTOR

Jak již bylo řečeno, Matlab ve svém základu je zaměřen především na snadnou práci s maticemi. Matice lze skládat z prvků nebo matic po vložení mezi závorky []. Jednotlivé symboly se oddělují ve vodorovném směru znakem mezera nebo čárka a ve svislém směru znakem konce řádky či středníkem. Zápis s čárkou a středníkem je úspornější, zápis s mezerami a konci řádku přehlednější.

Pro skládání prvků platí některá pravidla: výsledná matice musí být čtvercová či obdélníková, musí obsahovat ve všech řádcích počet shodných základních prvků – komplexních nebo reálných hodnot.

Poznámka: **r = input('Zadejte číslo');** Načte číslo, které zadá uživatel

4.1 Pole

Pole skládáme pouze použitím čárek či mezer a hranatých závorek. Indexujeme pomocí kulatých závorek, pozor Matlab indexuje od 1, prvek (0) neexistuje.

Pole – základní operace

```
pole1=[1,2,3,4,5]            %naplnění pole hodnoty 1-5
pole2=[6 7 8 9 10]           %naplnění pole hodnoty 6-10
pole_slouceni=[pole1,pole2] %sloučení pole1, pole2, hodnoty 1-10
prvek_paty=pole_slouceni(5) %paty prvek z pole - pětka
```

Výpis

```
pole1 = 1     2     3     4     5

pole2 = 6     7     8     9    10

pole_slouceni = 1     2     3     4     5     6     7     8     9    10

prvek_paty = 5
```

V Matlabu je pole reprezentováno jedno-dimenzionální maticí.

4.2 Matice

Zadávání matic

```
maticeA=[1,1,1;2,2,2;3,3,3]            %naplnění maticeA
maticeB=[1,2,3;1,2,3;1,2,3]            %naplnění maticeB
```

Výpis

```
maticeA =

     1     1     1
     2     2     2
     3     3     3


maticeB =

     1     2     3
     1     2     3
     1     2     3

pole_slouceni = 1     2     3     4     5     6     7     8     9    10

prvek_paty = 5
```

Operace s maticemi

```
%-------------------------------------------------------------
% operace +  -  .*  ./ .\ .^ jsou operace, které probíhají po prvcích
% pro operace +  -  .*  ./ .\ musí mít matice stejný rozměr, tj. stejný
% počet řádků a sloupců
%-------------------------------------------------------------
soucet=maticeA + maticeB
rozdil=maticeA - maticeB
soucin = maticeA .* maticeB          % násobeni matic prvek po prvku
podil = maticeA ./ maticeB           % dělení zprava matic prvek po prvku,
dělení prvků matice A prvky matice B

druhaMocnina=maticeA.^2              % umocnění jednotlivých prvků matice A
na druhou
transponovana=maticeA'               % transpozice matice A
inverzni = inv(maticeA)              % inverzní matice
determinant = det(maticeA)           % determinant matice A - čtvercová
hlavniDiagonala=diag(maticeA)        % hlavní diagonála matice A
lambda = eig(maticeA)                % vlastni čísla - čtvercová
alambda = abs(lambda)                % velikost vlastních čísel
hodnost=rank(maticeA)                % nemusí byt čtvercová
norm(maticeA-maticeA')               % zda je matice symetrická, v případě
že vyjde nula
%-------------------------------------------------------------
% operace  *  /  \  ^  jsou maticové operace
% pro maticové násobení musí být počet sloupců v 1. matici stejný jako
počet řádků v 2. matici!
% výsledek má počet řádků jako 1. matice a počet sloupců jako 2. matice
% pro maticové operace nemusí mít tedy matice stejný rozměr
%-------------------------------------------------------------
```

4.2.1 Speciální matice

Speciální matice

```
% generování speciálních matic
nulova_matice = zeros(4,6)    % nulová matice 4x6
nulova_ctver = zeros(5)       % nulová čtvercová matice 5x5
velikost=size(nulova_ctver)   % kontrola velikosti
matice_jednicek = ones(3,3)   % matice jedniček 3x3

jednicky_diagonala = eye(5)   % 5x5 jedničky na diagonále
jednicky_hlav_dig= eye(6,4)   % matice 6x4 s jedničkami na hlavní diagonále
nahodna_matice = rand(3,4)    % matice 5x7 vyplněná náhodnými čísly
(rovnoměrné rozložení)
```

Detailní seznam: *help elmat*

4.2.2 Cell pole

Pole, jejichž prvky jsou kopie jiných polí, obecně různé velikosti. Vytvářejí se přes složené závorky {}

Např.: A = [1 2 3; 1 2 3; 1 2 3];

C = {A sum(A) det(A) };

Přístup k prvkům je opět přes složené závorky, takže C{1} vrátí matici A. Ukládají se kopie, ne ukazatele, takže změna A neovlivní C{1}.

4.2.3 Práce s částí vektoru

Číslo(3,3)	číslo z matice na pozici 3,3
Submat(2:4,3:6)	vybere sub-matici
Řádek(2,:)	vybere druhý řádek

4.3 Vícerozměrná pole:

V Matlabu je též možné pracovat s vícerozměrnými poli.

- **R1 = rand(3,4,2)** vygeneruje 2 matice 3x4 naplněné náhodnými hodnotami
- **R1 = rand(3,4,2,2)** vygeneruje matici 2x2 obsahující matice 3x4

4.4 Dvojtečkový operátor

V Matlabu se velmi často používá tzv. dvojtečkový operátor ':', který slouží ke generování vektoru po sobě jdoucích čísel.

Syntaxe vypadá následovně: *start:konec* pro posloupnost s krokem 1, případně *start:krok:konec*, pokud chceme jiný krok než 1. Krok může být záporné číslo či číslo s desetinou tečkou.

Dvojtečkový operátor

```
%Dvojtečkový operátor
vektor1=1:10              %vygeneruje čísla 1,2...10
vektor2=-10:0             %vygeneruje čísla -10,-11...0
vektor3=0:0.1:1           %vygeneruje čísla 0,0.1,0.2...1
vektor4=-10:0.05:10     %vygeneruje čísla -10,-9.95...10
```

5 2D A 3D GRAFIKA V MATLABU

V Matlabu jde veškerý grafický výstup do figury, jinak řečeno do grafického okna. Těchto grafických oken může být samozřejmě více než jedno, každé má své číslo. Okna lze jak vytvářet tak se mezi nimi přepínat pomocí příkazu figure. Příkaz figure sám o sobě vytvoří prázdný graf.

„V Matlabu můžeme vykreslovat více méně vše, co nás napadne.“

5.1 Funkce pro vykreslování

- *plot* - Nejčastěji používaná funkce pro vykreslování vektorů a matic do dvourozměrného grafu. Elementární syntaxe vypadá následujícím způsobem *plot(a,b)* s tím, že vektory *a*, *b* musí být stejné délky. Přímo v této funkci lze ovlivňovat, jakým způsobem se vektory vykreslují (barva, šířka, volba bodů atd.)
- *loglog* - Pracuje obdobně jako plot, pouze vykreslí graf v logaritmickém měřítku.
- *semilogx* – Logaritmické měřítko pouze pro horizontální osu.
- *semilogy* – Logaritmické měřítko pouze pro vertikální osu.
- *polar* – Vykreslování v polárních souřadnicích.

5.2 Popis a úprava grafů

Úpravu lze provádět pomocí funkcí v nabídce grafického okna nebo pomocí následujících příkazů přímo ve skriptu (pracovním okně).

- *title* – nadpis grafu
- *xlabel* – popis horizontální osy
- *ylabel* – popis vertikální osy
- *legend* – vytvoření legendy grafu (zejména při více čarách v grafu)
- *grid* – mřížka v grafu
- *text* – libovolný text v grafu

Speciální znaky					
Znak	Zápis	Znak	Zápis	Znak	Zápis
α	\alpha	β	\beta	γ	\gamma
δ	\delta	ε	\epsilon	ω	\omega
λ	\lambda	ξ	\xi	π	\pi
ρ	\rho	σ	\sigma	τ	\tau
σ	\sigma	Δ	\Delta	Σ	\Sigma
∇	\nabla	∂	\partial	∞	\infty
√	\surd	∫	\int	≠	\neq
∈	\in	⊂	\subset	⊆	\subseteq
≤	\leq	≥	\geq	↑	\uparrow
∧	\wedge	∨	\vee	↓	\downarrow
↔	\leftrightarrow	←	\leftarrow	→	\rightarrow
¬	\neg	∀	\forall	∃	\exists

5.3 Další často používané příkazy

- *subplot* – Více oken v jedné figuře (v jednom grafu)
- *hold on/off* – Umožňuje do jednoho grafu vykreslovat více čar, pokud nezadáme příkaz *hold on* a zavoláme 2x po sobě *plot*, zůstane nám jen poslední graf (data z posledního *plot*)
- *zoom* – zvětšování a zmenšování
- *xlim, ylim* – vlastní měřítka jednotlivých os – *xlim(0,10)*

Celou řadu dalších funkcí lze nalézt v nápovědě (help graph2d a help specgraph).

Pro kreslení 3D grafů používáme příkaz *plot3(x,y,z)*.

5.4 Vlastnosti grafického objektu

5.4.1 Barvy

y	žlutá (yellow)
m	fialová (magenta)
c	modrozelená (cyan)
r	červená (red)
g	zelená (green)
b	modrá (blue)
w	bílá (white)
k	černá (black)

5.4.2 Styly pro čáru jsou:

-	plnou čarou
--	čárkovaně
:	tečkovaně
-.	čerchovaně

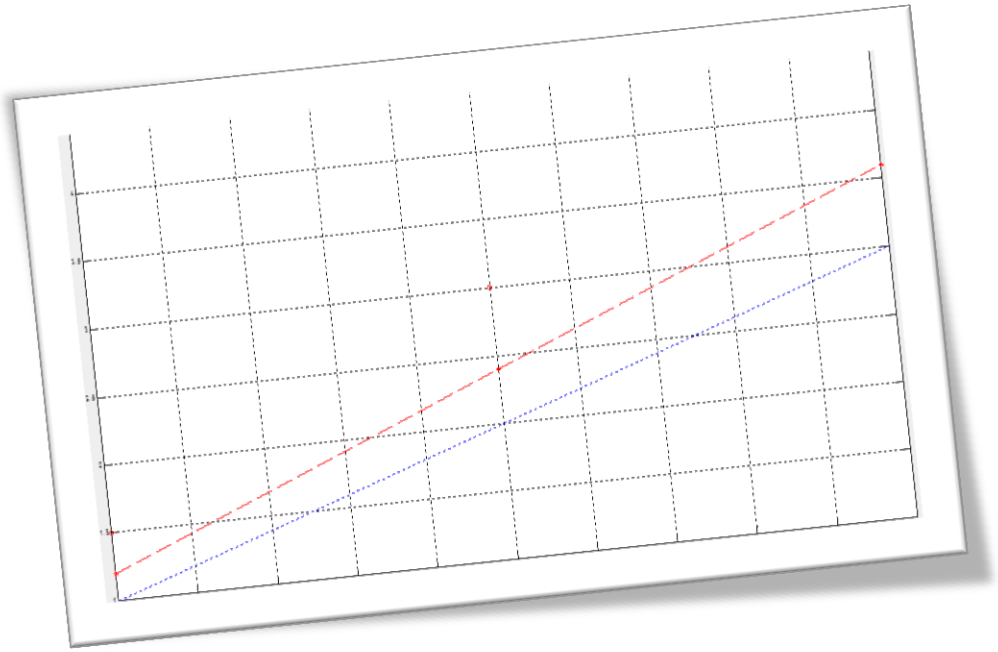
5.4.3 Jednotlivé znaky mohou být nastaveny pomocí:

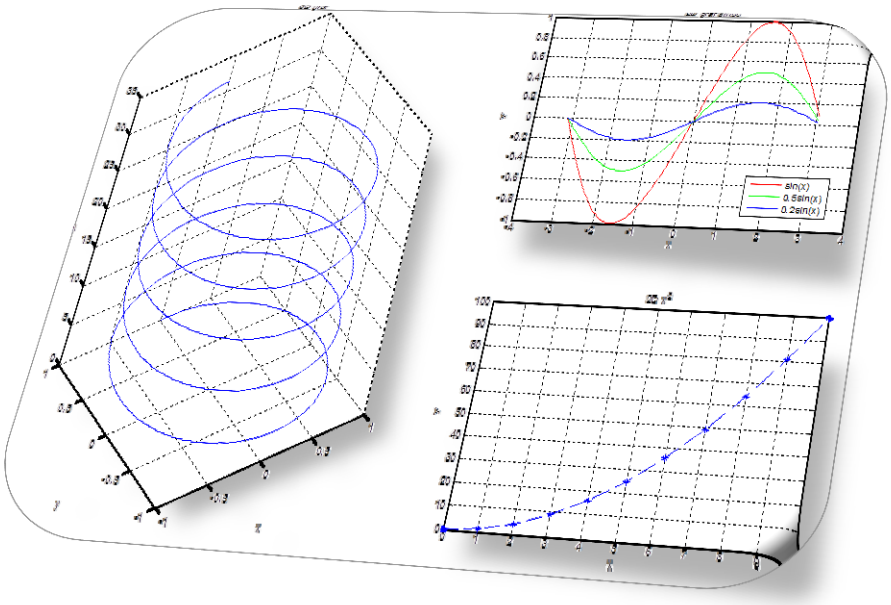
.	tečka
o	kroužek
+	křížek
*	hvězdička
s	čtvereček (square)
d	kosočtverec (diamond)
v	trojúhelník (otočený dolů)
^	trojúhelník (otočený nahoru)
<	trojúhelník (otočený doleva)
>	trojúhelník (otočený doprava)
p	pentagram
h	hexagram

Výše uvedené parametry lze samozřejmě kombinovat. Pro porozumění uvedeme několik příkladů.

Příklady

```
x=[1,2,3];
y=[1,2,3];
plot(x,y,'b:')            %vykreslení modré tečkované čáry
hold on                   %kreslení více grafů do jednoho
plot(x,1.2*y,'r--*')      %vykreslení červené čerchované čáry plus body
plot(x,1.5*y,'ro')        %vykreslení bodu kolečky
grid                      %mřížka
```





6 FUNKCE

Každá funkce se v Matlabu zapisuje do separátních m-filů. M-file se skriptem nesmí obsahovat definici funkce(í) a stejně tak nelze definovat fukci(e) přímo v příkazové řádce.

Funkce obsahují znovupoužitelný kód, nezávislý na konkrétních datech. Platí pro ně stejná pravidla jako pro skripty.

Funkce musí být ve stejném adresáři, jako „spouštěcí skript“.

Function out1 = jmeno_funkce(arg1, arg2, …)

nebo

Function [out1, out2, …] = jmeno_funkce(arg1, arg2, …)

- Určuje, že v m-filu je funkce
- Proměnné vytvořené ve funkci jsou lokální, tzn. neprojeví se ve workspace.
- Jméno funkce musí být shodné se jménem souboru, nesmí obsahovat mezeru.
- Bezprostředně za definicí funkce se píše komentář o tom, co funkce dělá, jaký je seznam vstupních a jaký výstupních parametrů. Tento komentář se vypisuje při zadání příkazu help.

out1 = hodnota Uloží nějakou hodnotu do výstupní proměnné

return Okamžitě ukončí funkci

- *nargchk* Otestuje počet zadaných argumentů funkce
- *nargin* Počet vstupních parametrů funkce
- *nargout* Počet výstupních parametrů funkce
- *varargin* Seznam vstupních parametrů funkce, můžou být matice i řetězce
- *varargout* Seznam výstupních parametrů funkce¨

Funkce sčítání a odčítání

```
function [soucet,rozdil] = funkce(cislo1,cislo2)
%
% Toto je help k funkci funkce. Všechny komentované řádky
% (začínající znakem %) pod hlavičkou funkce až do prvního prázdného řádku
% se vypíši zadáním příkazu 'help funkce'.
% Výstupem funkce je součet a rozdíl zadaných čísel cislo1,cislo2
% Volání funkce: [soucet,rozdil]=funkce(cislo1,cislo2)
% Výstupy funkce součet a rozdíl se mohou jmenovat libovolně, např.:x,y

    soucet = cislo1+cislo2;
    rozdil = cislo1-cislo2;

end % Nepovinné
```

Volání funkce

```
%skript pro praci s funkci
%help funkce
[soucet,rozdil]=funkce(10,20);  %volání funkce
soucet                          %výpis součtu
[soucet,rozdil]=funkce(100,50); %volání funkce
rozdil                          %výpis rozdílu
```

6.1 Vnitřní funkce

Každý m-file s funkcí musí obsahovat jednu hlavní funkci (tj. funkci se jménem stejným jako m-file), ta je public.

Dále pak je možné dodat libovolné množství sub-funkcí, které jsou private – je možné je volat pouze z hlavní funkce nebo subfunkcí.

6.2 Ukazatele na funkce

Je možné uložit si ukazatel na funkci a pak pracovat s tímto ukazatelem. Ukazatel na funkci pak může být použit jako parametr další funkce.

Ukazatel se získá pomocí operátoru **@.**

Ukazatel na funkce

```
funkce_sinus=@sin;
x=0:0.1:2*pi
y=funkce_sinus(x)
plot(x,y)
```

6.3 Funkce feval

Funkce feval se používá v případech, kdy potřebujeme zavolat existující funkci, ale předem nevíme, jak se bude volaná funkce jmenovat. Nejčastěji v případech, kdy vytváříme nějaký univerzální nástroj.

Syntaxe vypadá následující způsobem: [y1,y2,...] = feval(fhandle,x1,...,xn), kde handle je ukazatel na funkci (x vstupní parametry, y jsou výstupní parametry).

Poznámka: Funkci, kterou voláme pomocí feval, musí být buď knihovnou MATLABu, nebo uživatelskou funkcí uloženou v M-souboru v aktuálním adresáři.

Funkce fevalFcn

```
function[mocnina_soucet] = fevalFcn(x)
%funkce1 reprezentuje součet 1-5 mocniny x
%vstupem je hodnota x
mocnina_soucet = x + x^2 + x^3 + x^4 + x^5;
end
```

Volání funkce pomocí feval

```
y = fevalFcn(2)                                  %první možnost
y = feval(@fevalFcn,2)                           %druhá ...
y = feval('fevalFcn',2)                          %třetí ...
mocnina='fevalFcn'; x=2; y = feval(mocnina,x)    %čtvrtá ...
fhand=@fevalFcn; y = feval(fhand,2)              %pátá možnost
```

6.4 Funkce funkcí

Jedná se o třídu, která pracuje s jinými funkcemi. Tyto funkce zahrnují:

- hledání minim
- hledání kořenů funkce
- numerická integrace
- diferenciální rovnice

Příslušné operace je třeba provádět s analyticky zadanými funkcemi.

7 CYKLY, LOGICKÉ OPERÁTORY, VĚTVENÍ

Cykly slouží pro zápis příkazů, které mají být prováděny několikrát za sebou (opakovaně). Počet opakování může být předem známý nebo může záviset na nějaké podmínce (počet není předem znám). Cyklus s předem známým počtem opakování je v Matlabu realizován pomocí *for*, pro druhý případ, kde neznáme dopředu počet opakování, použijeme *while*.

break přeruší provádění cyklu

continue pokračuje v provádění cyklu od for/while

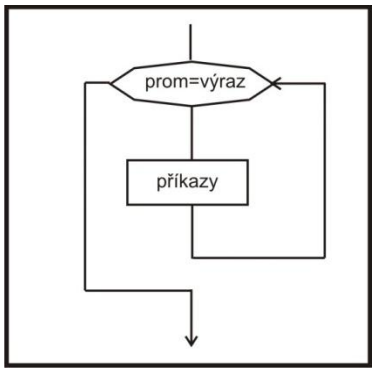
7.1 Cyklus for

Tento cyklus začíná klíčovým slovem „for“ a končí klíčovým slovem „end“

Cyklus for

```
%cyklus for
for x=1:10      %cyklus poběží 10x
    disp('Počet opakovaní cyklu for:')
    disp(x)
end
```

Činnost lze vyjádřit snadno větou: n-krát proveď příkazy.

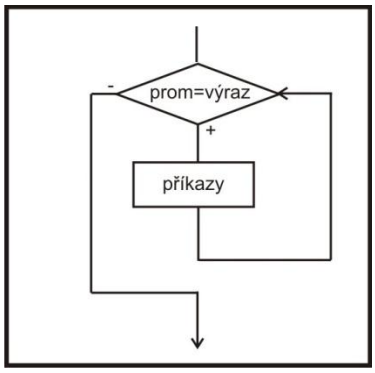


7.2 Cyklus while

Tento cyklus začíná klíčovým slovem „while“ a končí klíčovým slovem „end“

Cyklus while

```
n=1;  %počáteční hodnota, nutná!
while n<=10
    disp('Počet opakovaní cyklu while:')
    disp(n)
    n=n+1;  %nutné, musíme měnit, jinak nekonečný cyklus
end
```



Cyklus while testuje podmínku opakování cyklu vždy na počátku průběhu těla cyklu, počet průchodů cyklem může být nulový, pokud při prvním vykonání cyklu je podmínka neplatná.

7.3 Logické operátory

V Matlabu rozeznáváme následující sadu logických operátorů: AND, OR a negace.

A	B	A or B	A and B	negace A	negace B
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	1
1	1	1	1	0	0

7.4 Příkazy větvení

Větvení během programu lze provádět v zásadě dvěma způsoby a to pomocí if-else a switch-case.

7.4.1 If větvení

If větvení

```
if soucet < 10
    disp('Soucet je mensi nez 10')
elseif soucet > 10
    disp('Soucet je roven 10')
else
    disp('Soucet je vetsi nez 10')
end
```

7.4.2 Case větvení

Case větvení

```
switch prepni
    case {1,4} %zde může být více čísel
        disp('prepni je rovno 1 nebo 4');
    case 2
        disp('prepni je rovna 2');
    case 3
        disp('prepni je rovno 3')
    otherwise
        disp('prepni má jinou hodnotu')
end
```

8 TEXTOVÉ ŘETĚZCE

Textové řetězce jsou posloupnosti znaků, které zapisujeme do apostrofů (').

disp 'text' zobrazí daný text do Command Window

8.1 Vytváření textových řetězců

Vnitřně jsou textové řetězce reprezentovány jako pole čísel.

Tvorba textových řetězců

```
retezec='Byl jsem zde, Fantomas!'            % řetězec
retezec_matice=['mapa';'lana']               % všechny řádky musí mít
stejnou délku!
retezec_radky=char('a','ab','abc','abcd')    % řádky nemusí mít stejnou
délku
druha_radka=retezec_radky(2,:)               % přístup k jednotlivým řádkám
retezece_na_radce=['a','abc','abcef']        % více řetězců na jedné řádce
retezce_na_radce1=['Cislo pi=',num2str(pi)] % textu a převod čísla
na string
```

8.2 Manipulace s řetězci

Tvorba textových řetězců

```
% převod obecného textu na čísla dle ASCII a zpět
text_velka='ABCDEF'
text_mala='abcdef'
cisla1=double(text_velka)       % převod na číslo, dle ASCII tabulky 0-255
cisla2=double(text_mala)        % převod na číslo, dle ASCII tabulky 0-255
text_velka_zpet=char(cisla1)    % zpětný převod na text
text_mala_zpet=char(cisla2)     % zpětný převod na text
mezera=' '
ascii_mezera=double(mezera)     % mezera v ASCII, číselné podobě

%převod čísel na řetězce a zpět
cislo3=987.49                   % zadaní čísla
text_cislo3=num2str(cislo3)     % číslo jako text
tetx_cislo3_int=int2str(cislo3) % číslo jako text, zaokrouhlené na int
text_cislo4='512.796'           % číslo jako text
cislo4=str2num(text_cislo4)     % číslo, lze s ním počítat
soucet=cislo4+100               % správný výsledek
soucet=text_cislo4+100          % špatně
```

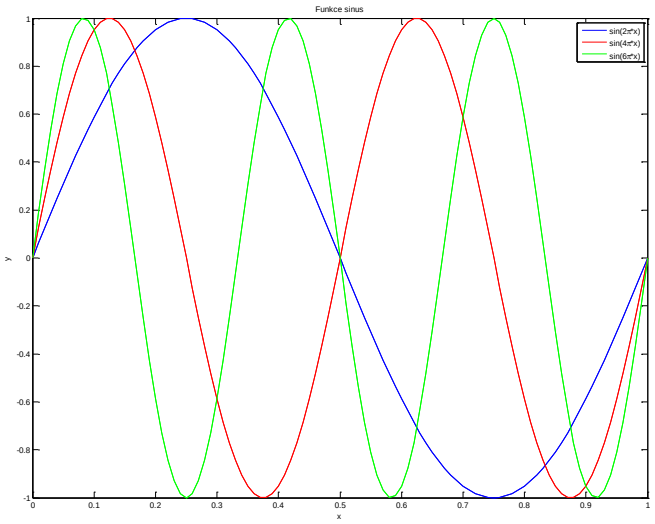
ASCII tabulku lze nalézt v přílohách (14.3 ASCII tabulka).

8.3 Eval

Funkce Eval pracuje s textovými proměnnými a provádí vyhodnocení proměnné.

Funkce Eval

```
%funkce eval
funkce_text='sin(2*pi*a*x)'; %funkce zadaná v textovém řetězci
x=0:0.01:1;
a=1;
y=eval(funkce_text);          %výpočet funkce
plot(x,y)                     %vykreslení funkce sinus
a=2;
y=eval(s);
hold on
plot(x,y,'r')                %vykreslení funkce sinus červeně
a=3;
y=eval(s);
hold on
plot(x,y,'g')                %vykreslení funkce sinus zeleně
legend('sin(2\pi*x)','sin(4\pi*x)','sin(6\pi*x)')
xlabel('x')
ylabel('y')
title('Funkce sinus')
```



9 ŘEŠENÍ ROVNIC, VÝPOČET HODNOTY INTEGRÁLU

9.1 Řešení rovnic

Soustavu rovnic Ax=b lze řešit pomocí operátoru \ nebo pomocí funkce inv a operátoru *. V každém případě je nutno nejdříve ověřit řešitelnost soustavy, jeden z možných příkladů je pomocí determinantu.

Řešení rovnic

```
% řešení rovnic v Matlabu
% A ... matice soustavy
% b ... vektor pravých stran
% x ... vektor řešení soustavy
%----------------------
%1. rovnice: x1+4x2=9
%2. rovnice: 3x1+5x2=13
% hledáme neznámé x1, x2 - sloupcový vektor x se 2 prvky
A=[1,4;3,5];
b=[9;13];
if det(A)==0 %ověření, zda existuje řešení
   disp('Soustava nemá řešení!')
else
    x=A\b % vypočet řešení
end
%1. rovnice: 10x1+13x2+42x3=-55
%2. rovnice: 8x1-x2-18x3=12
%3. rovnice: -5x1+13x2+2x3=108
% hledáme neznámé x1, x2, x3 - sloupcový vektor x se 3 prvky
A=[10,13,42;8,-1,-18;-5,13,2];
b=[-55;12;108];
if det(A)==0 %ověření, zda existuje řešení
   disp('Soustava nemá řešení!')
else
    x=inv(A)*b %druhá možnost řešení
end
%1. rovnice: x1+x2=2
%2. rovnice: 2x1+2x2=4
% hledáme neznámé x1, x2 - sloupcový vektor x se 2 prvky
A=[1,1;2,2];
b=[2;4];
if det(A)==0 %ověření, zda existuje řešení
   disp('Soustava nemá řešení!')
else
    x=A\b % vypočet řešení
end
```

9.2 Výpočty hodnot integrálu

Jedná se o numerické řešení, nutno napsat vlastní funkci, složité. Vhodné použít Toolbox symbolic více v kapitole 12 na straně 29.

10 VSTUPNĚ-VÝSTUPNÍ OPERACE

10.1 Záznam práce

V Matlabu můžeme prováděné příkazy a výsledky ukládat do souboru (tzv. žurnálu). K tomuto účelu se používá příkaz *diary* (respektive *diary on* a *diary off*). *Diary on* zapne ukládání a všechny následující příkazy a výsledky budou uloženy v pracovním adresáři do souboru diary, *diary off* toto ukládání přeruší. Pokud znovu spustíme *diary on*, nedojde k smazání dat v souboru diary, ale zapisuje se dále na jeho konec. Použití samotného diary buď vypne, či zapne ukládání. To záleží na předchozím stavu, pokud bylo ukládání aktivní, vypne se a naopak.

Použitím diary jmeno_souboru dosáhneme ukládání do námi zvoleného souboru.

10.2 Ukládání a načítání proměnných

V celé řadě případů, potřebujeme ukládat data na disk, či je načítat a pracovat s nimi. Data můžeme ukládat na disk v binárním kódu (nelze je přímo číst) nebo v textovém (ASCII) kódování. Úplně nejjednodušší způsob ukládání dat je příkazem *save* (uložit) s adekvátním příkazem *load* (načti).

Pro sofistikovanější práce se soubory jsou k dispozici příkazy fwrite a fread (binary) a fprintf a fscanf (ASCII).

- *save* – příkaz uloží všechny proměnné do souboru „matlab.mat“ do aktuálního adresáře
- *save jmeno_souboru* – všechny proměnné budou uložené do souboru „jmeno_souboru.mat“
- *save jmeno_souboru c1 c2 c3* – do souboru „jmeno_souboru.mat“ budou uloženy uvedené proměnné c1,c2,c3
- *save* (jmeno souboru, promenne) – *ascii* - data jsou uložena ve formátu ASCII. Příponu si lze libovolně zvolit, nejčastěji txt
- *load* – načte data ze souboru „matlab.mat“, soubor musí existovat a musíme být v adresáři, kde je umístěn
- *load jmeno_souboru* – načte všechny proměnné ze souboru „jmeno_souboru.mat“
- load *jmeno_souboru c1 c2 c3* – načte ze souboru „jmeno_souboru.mat“ uvedené proměnné c1,c2,c3
- *load (jmeno souboru, promenne)* – *ascii* - čtení z ASCII souboru (POZOR – v tomto souboru není uloženo jméno proměnné, pouze její obsah, z tohoto souboru lze načíst pouze jednu proměnnou, která má navíc stejné jméno, jako původní název souboru bez přípony)

11 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

Grafické uživatelské rozhraní (GUI) slouží k ovládání počítače pomocí interaktivních grafických prvků (jako jsou tlačítka, slidery, atd.). K tvorbě GUI se v Matlabu používá nástroj GUIDE (GUI Development Environment), který lze spustit přes File → New → GUI nebo příkazem *guide*. Při otevírání budeme volit možnost Blank GUI (Default).

V GUIDE se nám otevře prázdná pracovní plocha, která bude sloužit jako GUI, a na kterou lze umisťovat jednotlivé komponenty z levé části pomocí jednoduchého Drag&Drop. Upravíme tedy nejprve základní vzhled budoucího GUI (množinu komponent, jejich umístění a velikost) podle našich potřeb a poté se budeme věnovat rozšířenému nastavení, kam také patří vzájemné propojení komponent.

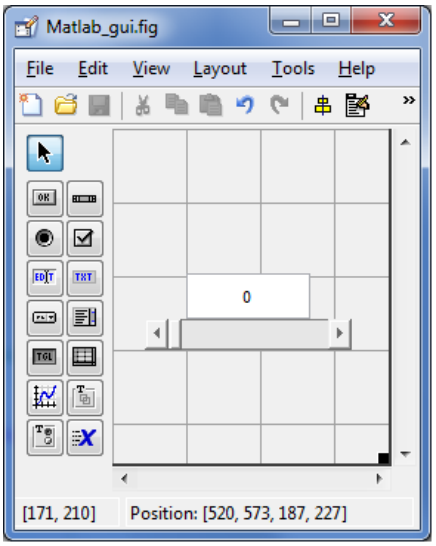
Po umístění komponent je vhodné program uložit! Při ukládání se vygeneruje druhý soubor, který obsahuje funkční stránku naší budoucí aplikace. Je to soubor m-file a budeme se mu věnovat níže.

Rozšířené nastavení každého bloku spravuje Property inspector (pravý klik na blok). Zde se nachází několik dalších parametrů bloku, které jsou závislé na typu bloku (tlačítko nebude mít minimální a maximální hodnotu rozsahu, kterou obsahuje slider, atd.). Důležitá vlastnost každého bloku je jeho název (Tag), pomocí kterého se na něj budeme odkazovat. V Property inspectoru je také u každého bloku kolonka Callback, která odkazuje na funkci ve vygenerovaném m-file. Tato funkce definuje akci, která se provede vždy při události bloku (stisk tlačítka, uložení hodnoty do Edit Text, atd.).

V Callback lze také definovat akce, které ovlivňují jiné komponenty a tím se komponenty vzájemně „provazují“. Odkazování na jiné komponenty se provádí pomocí struktury *handles*, která obsahuje globální proměnné, mezi něž patří i odkazy na všechny vytvořené komponenty ve formě *handles.tag_komponenty*. V zásadě budeme chtít hodnoty buď číst, nebo zapisovat. To se provádí pomocí „getrů“ a „setrů“. Například tedy

- ```
  set(handles.edit1,'String','Hello');
  ```
  nastaví obsah edit1 na hodnotu Hello a

- ```
  get(handles.edit1,'String');
  ```
  tuto hodnotu přečte.



Příklad:

Umístit slider a textEdit a po pohybu slideru aktualizovat hodnotu v textEditu a při zadání hodnoty do textEditu aktualizovat slider. Testovat, aby hodnota v textEditu nepřekročila meze slideru.

Pomoc:

- Min a Max slideru obsahují čísla.
- Hodnota slideru je uložena v atributu Value.
- textEdit obsahuje text, převod pomocí str2double()

GUI příklad

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
set( handles.edit1, 'String', num2str( get( handles.slider1, 'Value' ) ));

function edit1_Callback(hObject, eventdata, handles)
% Zjisti minimum a maximum nastaveni
min = get( handles.slider1, 'Min' );
max = get( handles.slider1, 'Max' );
% Hodnota z edit-boxu
val = str2double( get( handles.edit1, 'String' ) );
% Je hodnota v rozsahu?
if val > max
    val = max;
end
if val < min
    val = min;
end
% Nastavi hodnotu slide-baru
set( handles.slider1, 'Value', val );
% Updatuje hodnotu edit-boxu
set( handles.edit1, 'String', val );
```

Odkazy:

http://www.mathworks.com/discovery/matlab-gui.html

12 TOOLBOX SYMBOLIC

Symbolic math toolbox je nástroj pro práci se symbolickými výrazy. Může tedy sloužit například pro úpravu funkcí, výpočet kořenů, derivování a mnoho dalších analytických, nikoli numerických úkonů. K výpočtu je nutné nejdříve definovat **proměnné**, což lze udělat například pomocí

Definice proměnných a dosazení do proměnné

```
clear all
syms a b c  %definice proměnných
%Když máme definované proměnné, můžeme je umístit do funkce.
%To lze intuitivně udělat
f = a + b + c   %funkce
nahrada_a=subs(f, a, 3)%Dosazení za prom. můžeme prov. pomocí substituce
nahrada_b=subs(f, a, 'b')%Dosazení za prom. můžeme prov. pomocí substituce
hromanda_nahrada=subs(f, [a,b,c], [1,2,3])  %Hromadné dosazení
```

Funkce lze převádět do různých tvarů (roznásobovat nebo zjednodušovat):

Základní práce s funkcemi

```
%Funkce lze různě faktorizovat, rozvíjet nebo zjednodušovat:
clear all
syms a b
f = a^2 - 2*a*b + b^2;  %funkce
zjednoduseni = simple(f)    % zjednodušení
roznasobeni=expand(zjednoduseni)  % opětovné roznásobení
```

Symbolic toolbox lze využít také k řešení algebraických rovnic:

Řešení algebraických rovnic

```
%Symbolic tools lze využít také k řešení algebraických rovnic
clear all
syms x
reseni_x=solve('x^2 = 16', x)   %řešení rovnice
```

či jejich soustav:

Řešení soustav rovnic

```
%Soustavy rovnic
clear all
syms a b
f(1) = a + b - 3;   %první rovnice
f(2) = a + 2*b - 6; %druha rovnice
[a0,b0] = solve(f(1), f(2)) %řešení
```

Největší uplatnění tohoto toolboxu je při analytickém výpočtu diferenciálu, či integrálu funkce:

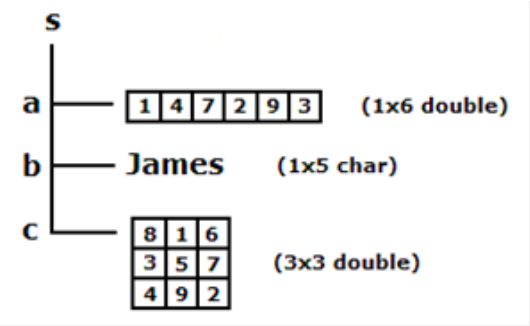
Výpočet diferenciálu a integrálu

```
%Výpočet dif. a integrálu funkce se provádí funkcemi diff() a int():
clear all
syms x
f = x^2 - 3*x + 4;
diferencial=diff(f) %výpočet diferenciálů
integral=int(f)     %výpočet integrálu
```

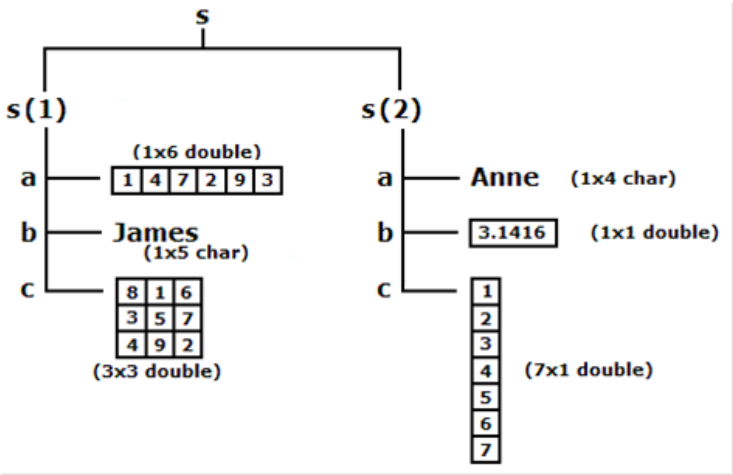
http://en.wikibooks.org/wiki/MATLAB_Programming/Symbolic_Toolbox

13 STRUKTURY V MATLABU

Struktury jsou datové typy, které uchovávají data v hierarchické struktuře v jedné proměnné. Každá struktura může obsahovat různý počet různě velkých polí odlišných typů. Jména a obsah těchto polí se definují při jejich vytváření.



Jako vše v Matlabu jsou i struktury definovány jako pole. Na obrázku výše je tedy pole s jedním prvkem a přirozeně můžeme vytvořit i pole s více prvky, jak je vidět na obrázku níže.



Struktury lze vytvářet například pomocí tečkové konvence

```
s.a = [1 4 7 2 9 3];
s.b = 'James';
s.c = magic(3);
```

nebo v jednom příkazu jako

```
s = struct('a',[1 4 7 2 9 3],'b','James','c',magic(3));
```

Čtení jednotlivých položek opět probíhá pomoc tečkové konvence.

http://www.mathworks.com/videos/introducing-structures-and-cell-arrays-68992.html

14 PŘÍLOHY

14.1 Matematické funkce

Trigonometrické funkce

- acos Inverzní kosinus
- acosh Inverzní hyperbolický kosinus
- acot Inverzní kotangents
- acoth Inverzní hyperbolický kotangents
- acsc Inverzní kosecant
- acsch Inverzní hyperbolický kosecant
- asec Inverzní sekant
- asech Inverzní hyperbolický sekant
- asin Inverzní sinus
- asinh Inverzní hyperbolický sinus
- atan Inverzní hyperbolický tangents
- atan2 Inverzní tangents
- atanh Inverzní hyperbolický tangents
- cos Kosinus
- cosh Hyperbolický kosinus
- cot Kotangents
- coth Hyperbolický kotangents
- csc Kosekant
- csch Hyperbolický kosekant
- sec Sekant
- sech Hyperbolický sekant
- sin Sinus
- sinh Hyperbolický sinus
- tan Tangents
- tanh Hyperbolický tangents

Exponenciální funkce

- exp Exponenciála
- log Přirozený logaritmus
- log10 Dekadický logaritmus
- log2 Logaritmus při základu 2
- pow2 Mocnina při základu 2
- sqrt Druhá odmocnina
- nextpow2 Nejbližší vyšší mocnina při základu 2

Komplexní funkce

- abs Absolutní hodnota nebo modul
- angle Fázový úhel
- conj Komplexně sdružená hodnota
- imag Imaginární část
- real Reálná část
- unwrap ’Rozbalení’ fázového úhlu
- isreal Test pro reálná pole
- cplxpair Setřídění komplexně sdružených párů

Zaokrouhlovací funkce

- fix Zaokrouhlování k nule
- floor Zaokrouhlování k $-\infty$
- ceil Zaokrouhlování k $+\infty$
- round Zaokrouhlování k nejbližšímu celému číslu
- mod Modulo
- rem Zbytek po dělení
- sign Signum

Transformace souřadnic

- cart2sph Transformace kartézských souřadnic na sférické
- cart2pol Transformace kartézských souřadnic na polární
- pol2cart Transformace polárních souřadnic na kartézské
- sph2cart Transformace sférických souřadnic na kartézské

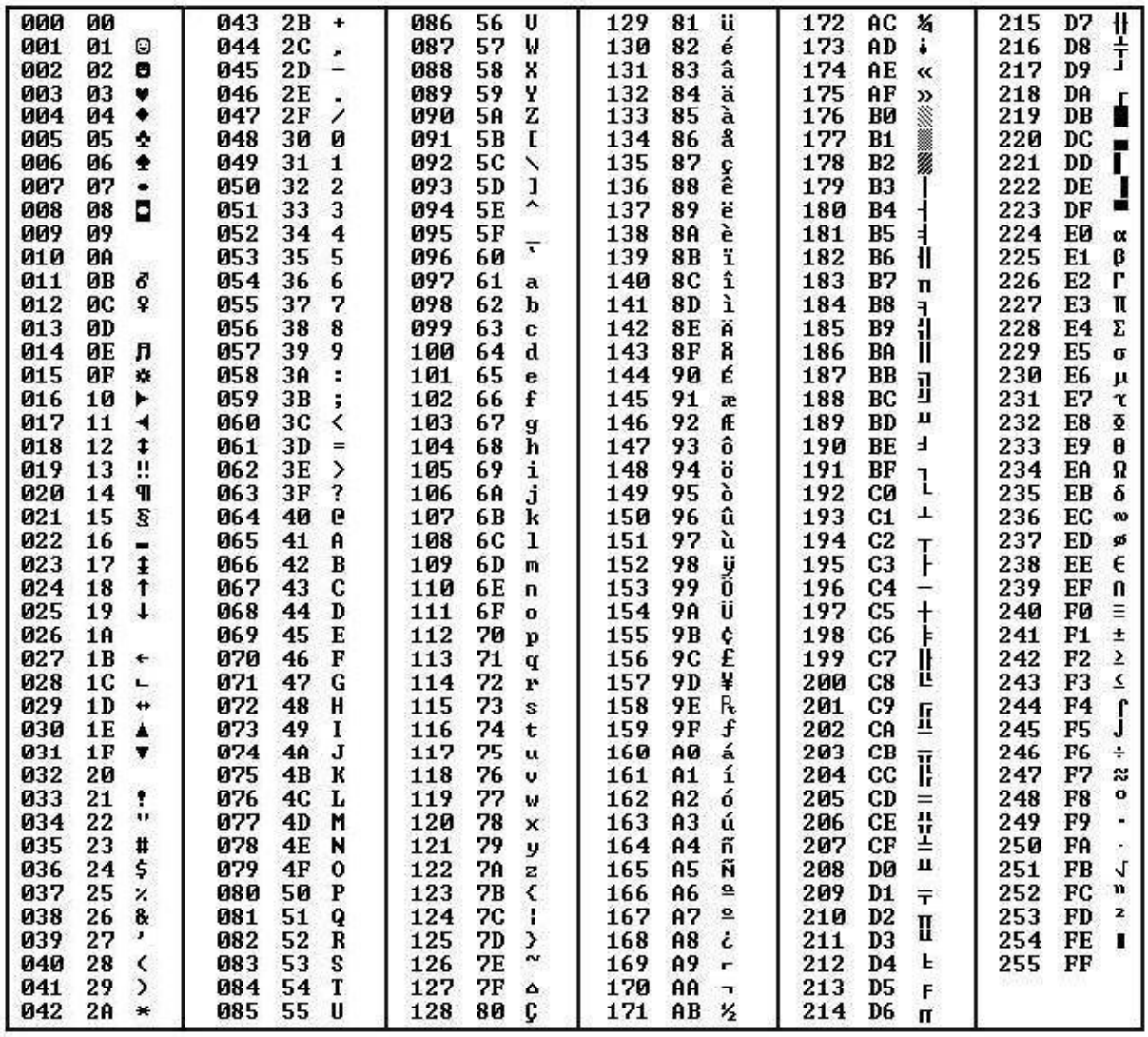
Funkce z teorie čísel

- factor Rozklad na prvočísla
- isprime Testování prvočísel
- primes Generování prvočísel
- gcd Největší společný dělitel
- lcm Nejmenší společný násobek
- rat Racionální aproximace
- rats Výstup racionálních čísel
- perms Všechny možné permutace
- nchoosek Všechny kombinace N nad K

14.2 Formáty zobrazení čísel

Příkaz	pí	Poznámka
format	3.1416	5 číslic
format short	3.1416	5 číslic
format long	3.14159265358979	15 číslic
format short e	3.1416e+00	5 číslic s exponentem
format long e	3.141592653589793e+00	16 číslic s exponentem
format short g	3.1416	zvolí short nebo short e
format long g	3.14159265358979	zvolí long nebo long e
format hex	400921fb54442d18	hexadecimální formát
format bank	3.14	číslo zobrazí na dvě desetinná místa
format rat	355/113	racionální aproximace

14.3 ASCII tabulka



Zdroj: http://www.pcdays.cz/wp-content/uploads/2011/09/200608081906_tabulkaASCII.jpg

15 REFERENCE

[1] *http://www.mathworks.com*, Oficiální stránky společnosti MathWorks

[2] *http://www.humusoft.cz*, Výhradní zástupce americké firmy MathWorks, Inc. pro Českou republiku a Slovensko

[3] *http://uprt.vscht.cz/majerova/matlab/*, Elektronické materiály předmětu Matlab, VŠCHT Praha, Ing. Diana Majerová

[4] KARBAN, K., *Výpočty a simulace v programech Matlab a Simulink*, Brno, Computer Press, 2006

[5] ZAPLATÍLEK, K., DONAŘ, B., *MATLAB tvorba uživatelských aplikací*, Praha, Ben, 2004

[6] SEDLÁČEK, M., ŠMÍD, R., *Matlab v měření*, Praha, Vydavatelstvá ČVUT, 2005

[7] CAMPBELL, S., CHANCELIER, J., NIKOUKHAH, R., *Modeling and Simulation inScilab/Scicos with ScicosLab 4.4*, Springers Science, 2010