1. Spuštení MATLABu

Ve většině operačních systému lze MATLAB spustit vypsáním příkazu matlab na příkazový řádek a ukončit příkazem exit nebo quit. Na PC lze MATLAB například spustit (pokud je nainstalován) takto:

```
C> matlab
```

a ukončit takto:

```
>> quit
```

V systémech podporujících multitasking, jako např. Unix, je z důvodů uvedených v kapitole 14 výhodné ponechat spuštěný jak MATLAB, tak i textový editor.

2. Zadávání matic

MATLAB v podstatě pracuje pouze s jedním druhem objektů - obdélníkovými číselnými maticemi, které mohou mít za prvky komlexní čísla. Všechny proměnné reprezentují matice, specielně maticí 1x1 se rozumí číslo (skalár), matice s jediným řádkem (sloupcem) je potom chápána jako vektor.

Matice mohou být v MATLABu zadávány různě:

- Zadány přímo výčtem prvků,
- Generovány vestavěnými funkcemi,
- Vytvořeny v M-souborech (viz kap. 12 a 14),
- Importovány z externích datových souborů (viz manuál).

Například každý z příkazů

```
A = [1 2 3; 4 5 6; 7 8 9]
```

a

```
A = [
1 2 3
4 5 6
7 8 9 ]
```

vytvoří čtvercovou matici 3x3 a přiřadí ji proměnné **A**. Vyzkoušejte. Prvky řádku matice lze oddělit čárkou nebo jen mezerou.

Při zadávání čísla v exponenciálním tvaru (např. 2.34e-9) je nutno provést celý zápis čísla bez mezer. Zadávání rozsáhlých matic je lepší provést pomocí M-souborů, kde se lépe opravují případné chyby (viz kap. 12 a 14).

Vestavěné funkce rand, magic, a hilb nabízejí například snadné generování matic pro pokusné účely. Příkaz rand(n) vytvoří matici **n x n** s náhodně generovanými prvky z intervalu (0,1), zatímco rand(m,n) vytvoří stejnou matici obdélníkovou **m x n**. magic(n) vytvoří celočíselnou matici **n x n**, která je magickým čtvercem (řádky a sloupce mají stejný součet); hilb(n) vytvoří Hilbertovu matici **n x n** (**m** a **n** jsou samozřejmě kladná celá čísla). Matice lze také generovat for-cyklem (viz kap. 6).

Jednotlivé prvky matice a vektoru jsou přístupné pomocí obvyklého indexování, například **A(2,3)** označuje prvek ze druhého řádku a třetího sloupce matice **A** a **x(3)** značí třetí složku vektoru **x**. Vyzkoušejte. Matice nebo vektor může být indexována pouze *kladnými* celými čísly.

3. Maticové a skalární operace

V MATLABu lze používat následující operace:

+	sčítání
-	odčítání
*	násobení
^	mocnění
'	transpozice
\	dělení zleva
/	dělení zprava

Tyto maticové operace fungují samozřejmě i pro skaláry (matice 1x1). Pokud rozměr matice nedovoluje provést příslušnou operaci, objeví se chybové hlášení. Výjimkou jsou skalární operace aplikované na matice (sčítání, odčítání, dělení, násobení). V tomto případě je operace provedena po prvcích (po složkách).

"Maticové dělení" je operace zasluhující speciální komentář. Je-li **A** invertibilní čtvercová matice a **b** vhodný řádkový (resp. sloupcový) vektor, pak

x = A \ b je řešení soustavy **A * x = b** a
x = b / A je řešení soustavy **x * A = b**.

Při dělení zleva, je-li **A** čtvercová, je upravena Gaussovou eliminací a poté použita k řešení **A * x = b**. Pokud **A** není čtvercová, je rozložena pomocí Householderovy ortogonalizace se sloupcovou pivotací a rozklad je použit k řešení s využitím metody nejmenších čtverců. Dělení zprava je definováno pomocí dělení zleva jako **b / A = (A' \ b')'**.

Skalární operace. Maticové operace sčítání a odčítání vždy probíhají po složkách, ale ostatní výše zmíněné operace nikoli -- jsou to *maticové* operace. Je důležité, že i tyto operace *, ^, \ a / lze nechat provést po složkách připsáním tečky před operaci. Například [1,2,3,4].*[1,2,3,4] nebo [1,2,3,4].\^2 dá výsledek [1,4,9,16]. Vyzkoušejte. Této vlastnosti lze využít zejména při používání matlabovské grafiky.

4. Příkazy, výrazy a proměnné; uložení práce

MATLAB je *"výrazový"* jazyk; výrazy, které zadáte jsou přečteny a vyhodnoceny. Příkazy MATLABu mají obvykle tvar

proměnná = *výraz*, nebo jen jednoduše
výraz

Výrazy se obvykle skládají z operátorů, funkcí a označení proměnných. Vyhodnocením výrazu se vytvoří matice, která je zobrazena na monitor a přiřazena do proměnné pro pozdější použití. Pokud jsou jméno proměnné a znak = vynechány, automaticky se vytvoří proměnná ans (answer = odpověď), do které se přiřadí výsledek.

Příkaz je běžně ukončen koncem řádku (Enter, Return). Lze však zajistit, aby příkaz pokračoval na dalším řádku, a to připsáním tří nebo více teček před ukončením řádku.

Pokud je poslední znak na řádku středník, potlačí se zobrazení na monitor, ale přiřazení do proměnné zůstane zachováno. Lze s výhodou využít k potlačení zobrazování mezivýsledků a podobně.

MATLAB v názvech příkazů, proměnných a funkcí rozlišuje mezi velkými a malými písmeny. Například solveUT není totéž co solveut.

Příkaz who vypíše seznam používaných proměnných. Proměnnou lze zrušit pčíkazem clear *jméno proměnné*. Samotný příkaz clear smaže všechny nepředddefinované proměnné.

Předdefinovaná proměnná eps (epsilon) udává přesnost - na většině počítačů 10^(-16). Užitečné při určování tolerance u konvergence iterovaných procesů.

Probíhající zobrazování nebo výpočet lze na většině počítačů zastavit bez ukončení MATLABu stiskem CTRL-C (CTRL-BREAK na PC).

Uložení práce. Je-li MATLAB ukončen, všechny proměnné definované při práci se ztrácí. Vyvoláním příkazu save před ukončením MATLABu dojde k zapsání všech proměnných do diskového souboru matlab.mat, který je pro člověka nesrozumitelný. Při dalším otevření MATLABu mohou být znovu vyvolány příkazem load.

5. Funkce pro generování matic

Užitečné funkce pro tvorbu matic jsou:

eye	jednotková matice
zeros	matice nul
ones	matice jedniček
diag	viz níže
triu	horní trojúhelníková matice
tril	dolní trojúhelníková matice
rand	náhodně generovaná matice
hilb	Hilbertova matice
magic	magický čtverec
toeplitz	viz help toeplitz

Například zeros(m,n) vytvoří matici nul **m** x **n** a zeros(n) vytvoří matici nul **n** x **n**; Je-li **A** matice, pak zeros(A) vytvoří matici nul stejných rozměrů, jaké má **A**.

Je-li **x** vektor, diag(x) je diagonální matice, která má na diagonále prvky **x** (shora dolů); je-li **A** čtvercová matice, pak diag(A) je vektor sestavený z diagonálních prvků matice **A**. Co je diag(diag(A))? Vyzkoušejte.

Matice mohou být blokové, například je-li **A** matice 3x3, pak

```
B = [A, zeros(3,2); zeros(2,3), eye(2)]
```

vytvoří jistou :-) matici 5x5. Vyzkoušejte.

6. Cykly, podmínky a relace

MATLAB umožňuje využívání těchto nástrojů (v základní podobě) tak, jak je obvyklé ve většině programovacích jazyků.

For. Například **for a** dvojtečka **n** - příkaz

```
x = []; for i = 1:n, x=[x,i^2], end
```

nebo

```
x = [];
for i = 1:n

    x = [x,i^2]

end
```

vytvoří jistý **n**-rozměrný vektor a příkaz

```
x = []; for i = n:-1:1, x=[x,i^2], end
```

vytvoří tentýž vektor v opačném pořadí. Vyzkoušejte. Pozor - matice může být prázdná (jako např. x = []). Příkazy

```
for i = 1:m

    for j = 1:n

        H(i, j) = 1/(i+j-1);

    end

end
H
```

vytvoří a zobrazí Hilbertovu matici **m** x **n**. Středník za vnitřním příkazem potlačí výstup mezivýslekdu, naopak koncové **H** zobrazí konečný výsledek.

While. Základní podoba while cyklu je

```
while relace

    příkazy

end
```

Příkazy jsou opakovaně prováděny tak dlouho, dokud platí relace (podmínka). Například následující cyklus pro dané číslo **a** spočítá a vypíše nejmenší nezáporné číslo **n**, pro které platí **2^n>= a**:

```
n = 0;
while 2^n < a

    n = n + 1;

end
n
```

If. Základní podoba jednoduchého příkazu if je

```
if relace

    příkazy

end
```

Příkazy budou provedeny pouze pokud relace platí. Je možné i větvení, jak ukazuje následující příklad:

```
if n < 0

    parity = 0;
```

```
elseif rem(n,2) == 0
    parity = 2;
else
    parity = 1;
end
```

Při větvení na dvě části lze pochopitelně část elseif vynechat.

Relace. Relační operátory v MATLABu jsou

<	menší než
>	větší než
<=	menší nebo rovno
>=	větší nebo rovno
==	rovno
~=	není rovno.

Pozor! Znak "=" se používá v přiřazovacím příkazu, zatímco pro rovnost je zaveden znak "==". Relace mohou být spojovány nebo kvantifikovány pomocí logických operátorů

&	and (a)
|	or (nebo)
~	not (negace)

Při aplikaci na skaláry je výsledek relace také skalár s hodnotou 1 nebo 0 v závislosti na tom, zda platí nebo ne. Vyzkoušejte 3 < 5, 3 > 5, 3 == 5, a 3 == 3. Při aplikaci na matice stejných rozměrů je výsledkem relace matice nul a jedniček, která udává výsledky relace pro jednotlivé vzájemně si odpovídající prvky. Vyzkoušejte a = rand(5), b = triu(a), a == b.

Relace mezi maticemi je v příkazech while a if chápána jako true (1, splněno), pouze pokud všechny prvky relační matice jsou nenulové. Chcete-li tedy provést *příkaz* když se matice **A** a **B** rovnají, stačí napsat

```
if A == B
    příkaz
end
```

pokud však chcete *příkaz* provést, když **A** není rovno **B**, použijte zápis

```
if any(any(A ~= B))
    příkaz
end
```

nebo jednodušeji

```
if A == B else
    příkaz
```

end

Pozor! Zdánlivě logicky správné

if A ~= B, *příkaz*, end

nevrátí zamýšlený výsledek, neboť se *příkaz* provede pouze v případě, ze *každá* dvojice vzájemně si odpovídajících prvků matic **A** a **B** relaci == porušuje. Funkce any a all mohou být využity k redukci maticových relací na vektorové a skalární. Ve zmíněném příkladě je nutno psát dvakrát any, protože any je vektorový operátor (viz kap. 8).

Příkaz for povoluje použít namísto zápisu 1:n libovolnou matici. Pro podrobnější popis tohoto použití příkazu for viz manuál.

7. Skalární funkce

Jisté funkce jsou v MATLABu zavedeny v podstatě pouze pro skaláry. Při aplikaci na matici se aplikují po prvcích (po složkách). Nejběžnější z těchto funkcí jsou

sin	asin	exp	abs	round
cos	acos	log (přirozeny log)	sqrt	floor
tan	atan	rem (zbytek)	sign	ceil

8. Vektorové funkce

Jiné funkce jsou v MATLABu definovány pro vektor (řádkový nebo sloupcový), ale lze je aplikovat i na matici **m** x **n** (**m >= 2**) - aplikace probíhá po sloupcích, výsledkem je řádkový vektor obsahující výsledky z jednotlivých sloupců. Aplikaci po řádcích lze zajistit tranpozicí, například mean(A')'. Některé z těchto funkcí jsou

max	sum	median	any
min	prod	mean	all
sort		std	

Například maximální prvek matice **A** lze určit jako max(max(A)), ale pozor na nesprávný zápis max(A). Vyzkoušejte.

9. Maticové funkce

Obrovský potenciál MATLABu spočívá v jeho maticových funkcích. Nejpoužívanější z nich jsou

eig	vlastní čísla a vlastní vektory
chol	Choleskeho rozklad
svd	rozklad podle singulárních hodnot
inv	inverze
lu	LU rozklad
qr	QR rozklad
hess	Hessenbergův tvar

schur	Schurova dekompozice
rref	Viz manuál
expm	mocnina matice
sqrtm	odmocninová matice
poly	charakteristický polynom
det	determinant
size	rozměr
norm	1-norma, 2-norma, F-norma, <infinity>-norma*
cond	Viz manuál
rank	hodnost

* Poznámka: <infinity> je symbol pro hodnotu nekonečno, v HTML pro ni zatím není patřičný zápis

Funkce mohou v MATLABu mít jeden nebo více výstupních argumentů, např.

y = eig(A), nebo jednoduše eig(A)

vytvoří sloupcový vektor obsahující vlastní čísla matice **A**, zatímco

[U,D] = eig(A)

vytvoří matici **U**, jejíž sloupce jsou vlastní vektory **A**, a diagonální matici **D**, která má na diagonále vlastní čísla **A**. Vyzkoušejte.

10. Příkazový řádek - úpravy a opětovné zadávání příkazů

Příkazový řádek lze v MATLABu editovat velice snadno. Šipkami umístíme kurzor na příslušné místo a použitím kláves Backspace a Delete smažeme nahrazované znaky. Na PC vyzkoušejte funkci kláves Home, End, Delete, jinde viz help cedit nebo type cedit.

Velice užitečná je funkce kláves šipka nahoru a šipka dolů. Umožňují pohyb po zásobníku předchozích příkazů. Lze tedy vyvolat předešlý příkazový řádek, upravit jej a odeslat jako nový příkaz. Tento postup je o mnoho pohodlnější než používání M-souborů, které vyžaduje přesouvání mezi MATLABem a editorem (viz kap. 12 a 14). Například počty operací (viz kap. 15) při počítání inverzních matic různých rozměrů mohou být porovnávány opakovaným voláním, úpravou a prováděním příkazu

a = rand(8); flops(0), inv(a); flops

Pokud bychom chtěli porovnat grafy funkcí **y = sin mx** a **y = sin nx** na intervalu **[0,2\pi]** pro různá **m** a **n**, bylo by to možné provést upravováním jednoho příkazového řádku:

m=2; n=3; x=0:.01:2*pi; y=sin(m*x); z=sin(n*x); plot(x,y,x,z)

11. Submatice a dvojtečkový zápis

Vektory a submatice se v MATLABu často používají k dosažení efektů komplexní manipulace s daty. Klíčem k účinné manipulaci s těmito objekty jsou "Dvojtečkový zápis" (který se používá jak pro generování vektorů tak i pro odkazování na submatice) a indexování pomocí vektorů. Jejich využívání dovoluje omezit používání cyklů, které MATLAB zpomalují, a zapsat kód jednoduše a čitelně. *Pokuste se vynaložit na seznámení s nimi o něco více námahy.*

Výraz 1:5 (již jsme se s ním setkali v příkazu for) je vlastně řádkový vektor [1 2 3 4 5]. Čísla nemusí být celá a posloupnost rostoucí, například

```
0.2:0.2:1.2
```

dává [0.2, 0.4, 0.6, 0.8, 1.0, 1.2], a

```
5:-1:1
```

dává [5 4 3 2 1]. Následující příkazy např. vytvoří tabulku hodnot funkce sinus. Vyzkoušejte.

```
x = [0.0:0.1:2.0]';
y = sin(x);
[x y]
```

Protože se funkce sin aplikuje po složkách, je výsledkem vektor **y** vytvořený z vektoru **x**.

Dvojtečkový zápis lze použít jako odkaz na submatici matice (indexování). Například

```
A(1:4,3)
```

je sloupcový vektor skládající se z prvních čtyř prvků třetího sloupce matice **A**. Samotná dvojtečka znamená celý řádek nebo sloupec:

```
A(:,3)
```

je třetí sloupec **A** a A(1:4,:) značí první čtyři řádky. Libovolný celočíselný vektor lze použít jako index:

```
A(:,[2 4])
```

obsahuje (jako sloupce) druhý a čtvrtý sloupec **A**. Takovéto indexování je dovoleno na obou stranách přiřazovacího příkazu:

```
A(:,[2 4 5]) = B(:,1:3)
```

nahradí 2., 4. a 5. sloupec matice **A** prvními třemi sloupci matice **B**. Nezapomeňte, že přiřazena a zobrazena je *celá* pozměněná matice **A** Vyzkoušejte.

2. a 4. sloupec matice **A** lze zprava vynásobit maticí [1 2;3 4] (matice 2x2):

```
A(:,[2,4]) = A(:,[2,4])*[1 2;3 4]
```

Přiřazena (do A) a zobrazena je opět celá pozměněná matice.

Co se stane po zadání příkazu x = x(n:-1:1), je-li **x** **n**-složkový vektor? Vyzkoušejte.

Tyto funkce oceníte zejména pokud příkazy MATLABu porovnáte se zápisem stejné operace v Pascalu, Fortranu nebo C.

12. M-soubory

MATLAB umožňuje provést posloupnost příkazů uložených v souborech na disku. Tyto soubory se nazývají "M-soubory", protože musí být typu ".m". Podstatná část Vaší práce v MATLABu bude tvorba a ladění M-souborů.

Existují dva typy M-souborů: *skriptové soubory* a *funkční soubory*.

Skriptové soubory. Skriptový soubor se skládá z posloupnosti běžných MATLABovských příkazů. Má-li např. jméno rotate.m, pak příkaz rotate v MATLABu způsobí provedení příkazů obsažených v souboru.

Proměnné ve skriptovém souboru jsou chápány jako globální a vždy změní hodnotu proměnných se stejným jménem, které byly definovány lokálně během práce v MATLABu.

Skriptové soubory se obvykle používají k zadávání dat do rozsáhlých matic. V souboru lze totiž chyby opravit lépe než při přímém zadávání. Je-li např. na disku soubor data.m obsahující

```
A = [
1 2 3 4
5 6 7 8
];
```

pak MATLABovský příkaz data provede přiřazení zadané v data.m.

M-soubor může obsahovat odkazy na jiné M-soubory včetně rekurzivního odkazu na sebe sama.

Funkční soubory. Funkční soubory vnášejí do MATLABu pružnost a přizpůsobivot. Můžete si vytvořit nové, vlastní funkce specielně pro Váš problém, které potom budou mít stejné postavení, jako běžné funkce v MATLABu. Proměnné jsou ve funkčních souborech standartně lokální, avšak od verze 4.0 je povoleno definovat proměnnou jako globální.

Pro ilustraci si uveďme krátký příklad funkčního souboru.

```
function a = randint(m,n)
%RANDINT Randomly generated integral matrix.
% randint(m,n) returns an m-by-n such matrix with entries
% between 0 and 9.
a = floor(10*rand(m,n));
```

Obecnější zápis této funkce by vypadal:

```
function a = randint(m,n,a,b)
%RANDINT Randomly generated integral matrix.
% randint(m,n) returns an m-by-n such matrix with entries
% between 0 and 9.
% rand(m,n,a,b) return entries between integers a and b.
if nargin < 3, a = 0, b = 9; end
a = floor((b-a+1)*rand(m,n)) + a;
```

Toto by na disku mělo být uloženo v souboru randint.m (aby jméno souboru odpovídalo jménu funkce). První řádek obsahuje deklaraci jména funkce a vstupních a výstupních argumentů. Bez této řádky by soubor byl pouze souborem skriptovým. Příkaz

```
z = randint(4,5),
```

potom například způsobí, že čísla 4 a 5 budou uložena do proměnných **m** a **n** ve funkčním souboru a výsledek se vloží do proměnné **z**. Jelikož proměnné ve funkčním souboru jsou lokální, jsou jejich názvy nezávislé na názvech proměnných definovaných z příkazové řádky při práci v MATLABu

Poznámka: Použití nargin ("number of input arguments (počet vstupních argumentů)") dovoluje předdefinovat hodnotu vstupního argumentu, který nebyl zadán -- stejně jako **a** a **b** v příkladu výše.

Funkce může také mít více výstupních argumentů, například:

```
function [mean, stdev] = stat(x)
% STAT Mean and standard deviation
% For a vector x, stat(x) returns the
% mean and standard deviation of x.
% For a matrix x, stat(x) returns two row vectors containing,
% respectively, the mean and standard deviation of each column.
```

```
[m n] = size(x);
if m == 1

    m = n; % handle case of a row vector

end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```

Je-li toto umístěno v souboru stat.m, provede např. příkaz [xm, xd] = stat(x) přiřazení průměru a standartní odchylky vstupních dat (z vektoru **x**) do proměnných **xm** a **xd**. Lze také provést pouze jedno přiřazení i u funkce, která má více výstupních argumentů. Například xm = stat(x) (okolo **xm** nemusí být závorky) přiřadí průměr **x** do **xm**.

Symbol % znamená, že zbytek řádku je komentář. MATLAB zbytek řádku ignoruje. Několik prvních řádků s komentářem, které popisují M-soubor, bychom do funkčního souboru *vždy* měli napsat. Jsou přístupné např. pro on-line hlep a zobrazí se např zadáním help stat.

Tato funkce ilustruje některé z nástrojů MATLABu, které lze použít ke konstrukci efektivního kódu. Při zápisu pozor na technické problémy, např. že x.^2 je matice druhých mocnin prvků **x**, že sum je vektorová funkce (kap.8), že sqrt je skalární funkce (kap. 7) a že dělení v sum(x)/m je maticově-skalární operace.

Následující funkce, která počítá největší společný dělitel dvou celých čísel Eukleidovým algoritmem, je příkladem na použití chybových hlášení (viz příští kapitola).

```
function a = gcd(a,b)
% GCD Greatest common divisor
% gcd(a,b) is the greatest common divisor of
% the integers a and b, not both zero.
a = round(abs(a)); b = round(abs(b));
if a == 0 & b == 0

    error('Nejv. spol. dělitel není definován, jsou-li obě čísla rovna nule.')

else

    while b ~= 0

        r = rem(a,b)
        a = b; b = r;

    end

end
```

Některé vyšší nástroje jsou předvedeny v následující funkci. Připomínáme, že některé ze vstupních argumentů funkce -- jako např. *tol* v ukázce, mohou být předdefinovány použitím nargin. Podobným způsobem může být použita proměnná nargout. Pozor na to, že relace je číslo (1 pro true, 0 pro false) a že příkazy while nebo if vyhodnotí relaci "nenula" znamená "true" a *0* znamená "false". Konečně MATLABovská funkce feval dovoluje použít jako vstupní proměnnou řetězec (string) odkazující na jinou funkci.

```
function [b, steps] = bisect(fun, x, tol)
%BISECT Zero of a function of one variable via the bisection method.
% bisect(fun,x) returns a zero of the function. fun is a string
% containing the name of a real-valued function of a single
% real variable; ordinarily functions are defined in M-files.
% x is a starting guess. The value returned is near a point
% where fun changes sign. For example,
```

```
% bisect('sin',3) is pi. Note the quotes around sin.
%
% An optional third input argument sets a tolerence for the
% relative accuracy of the result. The default is eps.
% An optional second output argument gives a matrix containing a
% trace of the steps; the rows are of form [c f(c)].

% Initialization
if nargin < 3, tol = eps; end
trace = (nargout == 2);
if x ~= 0, dx = x/20; else, dx = 1/20; end
a = x - dx; fa = feval(fun,a);
b = x + dx; fb = feval(fun,b);

% Find change of sign.
while (fa > 0) == (fb > 0)

    dx = 2.0*dx;
    a = x - dx; fa = feval(fun,a);
    if (fa > 0) ~= (fb > 0), break, end
    b = x + dx; fb = feval(fun,b);

end
if trace, steps = [a fa; b fb] end

% Main loop
while abs(b - a) > 2.0*tol*max(abs(b),1.0)

    c = a + 0.5*(b - a); fc = feval(fun,c);
    if trace, steps = [steps; [c fc]];
    if (fb > 0) == (fc > 0)

        b = c; fb = fc;

    else

        a = c; fa = fc;

    end

end
```

Některé z MATLABovských funkcí jsou vestavěny, zatímco ostatní jsou distribuovány jako M-soubory. Kód jakéhokoli M-souboru (MATLABovského nebo Vašeho vlastního) lze prohlížet zadáním příkazu type *jméno funkce*. Vyzkoušejte zadat type eig, type vander a type rank.

13. Textové řetězce, chybová hlášení, input

Textové řetězce se v MATLABu zadávají obklopeny apostrofy. Například

```
s = 'This is a test'
```

přiřadí zadaný text do proměnné s.

Textové řetězce mohou být zobrazeny funkcí disp. Například:

```
disp('this message is hereby displayed')
```

Chybová hlášení je nejlepší zobrazovat pomocí funkce error,

```
error('Sorry, the matrix must be symmetric')
```

neboť je-li takto zapsáno v M-souboru, způsobí ukončení jeho činnosti.

Z M-souboru může být uživatel interaktivně požádán, aby zadal data, funkcí input. Když se například provádí příkaz

```
iter = input('Enter the number of iterations: '),
```

zobrazí se žádost o data (zde počet iterací) a do doby, než uživatel data zadá, je provádění výpočtu pozastaveno. Po stisku klávesy Enter (Return) jsou data přiřazena do proměnné iter a výpočet pokračuje.

14. Dočasná práce s M-soubory

Při práci s MATLABem je často třeba vytvořit nebo upravit nějaký M-soubor a pak se vrátit zpátky do MATLABu. Během editace M-souboru by však MATLAB měl zůstat spuštěný, protože jinak bychom při jeho ukončení ztratili všechny proměnné.

To lze jednoduše zařídit použitím znaku !. Jste-li v MATLABu a napíšete před jakýmkoli systémovým příkazem (editace, kopírování, tisk...) znak !, můžete tento příkaz provést, aniž byste ukončili MATLAB. Je-li např. příkaz ed příkazem pro spuštění editoru, pak v MATLABu zadaný příkaz

```
>> !ed rotate.m
```

umožní upravit soubor rotate.m ve Vašem textovém editoru. Po skončení práce v editoru se vrátíte do MATLABu do okamžiku, kdy jste jej opustili.

Jak již bylo poznamenáno v kapitole 1, v systémech s podporou více procesů (např. Unix) se vyplatí ponechat spuštěný MATLAB i editor. Zatímco v jednom procesu pracujeme, je druhý suspendován. Pokud mohou být tyto procesy otevřeny v různých oknech, ponechte v jednom otevřený MATLAB a ve druhém editor.

Pro podrobnější informace o Vašem systému a o lokální instalaci kontaktujte Vašeho správce.

Verze 4.0 obsahuje mnoho odlaďovacích nástrojů. Viz help dbtype a zde uvedené odkazy.

Jste-li v MATLABu, pak příkaz dir vypíše obsah aktuálního adresáře, zatímco příkaz what zobrazí pouze seznam M-souborů uložených v aktuálním adresáři. Příkazy delete a type lze použít ke smazání souboru z disku a k vypsání souboru na obrazovku, příkazem chdir lze změnit aktuální adresář. Tyto příkazy mohou nahradit systémové příkazy, ale nevyžadují použití !.

M-soubor musí být pro MATLAB dostupný. Soukromé M-soubory jsou většinou uloženy v podadresáři s názvem matlab v domovském adresáři uživatele a jsou tak pro MATLAB dostupné z jakéhokoli pracovního adresáře. Další informace viz manuál - kapitola MATLABPATH.

15. Porovnávání efektivity algoritmů, efektivní čas

Pro efektivitu algoritmu existují dvě kritéria. Počet vykonaných operací a čas, který uběhl od zahájení výpočtu.

MATLABovská funkce flops uchovává průběžně počet vykonaných operací. Příkaz flops(0) (ne flops = 0!) její hodnotu anuluje. Zadáme-li tedy flops(0) bezprostředně před zahájením běhu algoritmu a flops bezprostředně po jeho ukončení, udává hodnota počet vykonaných operací pro tento algoritmus.

MATLABovská funkce clock udává aktuální čas s přesností na setiny sekund (viz help clock). Máme-li dva časy, t1 a t2, pak etime(t2,t1) udává čas, který uběhl od t1 do t2. Dá se tedy např. změřit čas potřebný k vyřešení dané soustavy lineárních rovnic $A\ x = b$ Gaussovou eliminací, a to následovně:

```
t = clock; x = A\b; time = etime(clock,t)
```

Chtěli byste porovnat toto číslo a počet vykonaných operací při řešení stejné soustavy vzorcem x = inv(A)*b;? Vyzkoušejte. Verze 4.0 obsahuje výhodnější tic a toc.

Je nutno poznamenat, že etime nemusí být objektivní kritérium na počítačích, které umožňují sdílení času. Zde totiž čas potřebný k provedení algoritmu závisí na momentální celkové vytíženosti počítače.

16. Formát výstupu

Formát zobrazovaných výsledků lze v MATLABu ovlivnit následujícími příkazy:

format short	zobrazení na 4 desetinná místa (default)
format long	zobrazení na 14 desetinných míst
format short e	scient. zápis na 4 des. místa
format long e	scient. zápis na 15 des. mist

Vybraný formát je pro výstupy uplatňován až do té doby, kdy je změněn na jiný.

Příkaz format compact zabraňuje zobrazování zbytečných prázdných řádků a tím dovoluje zahustit zápis informací na obrazovce. Jeho použití nezávisí na žádném jiném příkazu format.

17. Hardcopy

Hardcopy nejsnáze vyvoláme příkazem diary. Příkazem

diary *jméno souboru*

se zahájí zápis do diskového souboru daného jména, pokud jméno není uvedeno, vytvoří se automaticky soubor s názvem diary). Zapisováno je vše, co se objeví na obrazovce (kromě grafiky), a to zž do té doby, kdy zadáme příkaz diary off; příkaz diary on vyvolá pokračování přerušeného zápisu. Po ukončení práce lze tento soubor upravovat, tisknout atd. dle libosti buď ze systému nebo přímo z MATLABu použitím znaku ! (viz kap. 14).

18. Grafika

V MATLABu je možné zobrazovat jak rovinné, tak i 3-D (síťové) grafy. Příkazem plotdemo si můžete prohlédnout ukázky některých grafických funkcí.

Rovinné grafy. Příkaz plot vytváří grafy závislosti x a y; jsou-li **x** a **y** vektory o stejné délce, pak plot(x,y) otevře grafické okno a vykreslí x-y graf (jako prvky **x** versus prvky **y**). Například graf funkce sinus na intervalu od -4 do 4 lze získat pomocí příkazů:

```
x = -4:.01:4; y = sin(x); plot(x,y)
```

Vyzkoušejte. Vektor **x** je chápán jako část definičního oboru, 0.1 je zvolený krok a **y** je vektor udávající hodnoty funkce sinus ve zvolených bodech (krok 0,1), protože funkce sin aplikovaná na vektor působí po složkách.

Jste-li v grafickém modu, můžete se stiskem libovolné klávesy vrátit zpět na příkazový řádek MATLABu. Naopak příkaz shg (show graph = zobraz graf) vrátí aktuální grafickou obrazovku. Pokud Váš počítač podporuje víceokenní umisťování procesů se zvláštním grafickým oknem, je výhodné ponechat grafické okno otevřené někde po straně obrazovky.

Jako druhý příklad si můžete nakreslit graf funkce *y = e^(-x^2)* na intervalu od -1.5 do 1.5 :

```
x = -1.5:.01:1.5; y = exp(-x.^2); plot(x,y)
```

Poznámka: Raději před znak ^ připište tečku. Budete mít jistotu, že umocnění skutečně proběhne po složkách (viz kap. 3).

Lze vykreslit i grafy parametrizovaných křivek, vyzkoušejte např.

```
t=0:.001:2*pi; x=cos(3*t); y=sin(2*t); plot(x,y)
```

Příkaz grid přikreslí do grafu čtvercovou síť.

Ke grafům lze samozřejmě připojit titulky, popisky os a je možné přidat i text přímo do grafu. Slouží k tomu následující příkazy (všechny mají jako argument řetězec (string)).

title	titulek grafu
xlabel	popis osy x
ylabel	popis osy y
gtext	interaktivně vkládaný text
text	umístění textu na zadané souřadnice

Například příkaz

```
title('Best Least Squares Fit')
```

vyrobí ke grafu titulek, příkaz gtext('The Spot') umožní vybrat (myší, šipkami) místo, kam se daný text umístí po stisku nějaké jiné klávesy.

Předdefinované nastavení pro osy je takové, že se měřitko volí automaticky. To lze změnit příkazem axis. Je-li *c = [xmin,xmax,ymin,ymax]* čtyřprvkový vektor, pak axis(c) nastaví měřítko tak, jak je předepsáno v c. Samotné axis zmrazí aktuální nastavení pro všechny další grafy do odvolání, zadáme-li axis ještě jednou potom, vrátí původní nastavení na automatické měřítko. Příkaz axis('square') zajistí, že na obou osách bude použito stejné měřítko. Ve verzi 4.0 byl význam axis zásadně pozměněn, viz help axis.

Dvě možnosti, jak vykreslit dva grafy do jednoho obrázku si ukážeme na příkladě

```
x=0:.01:2*pi;y1=sin(x);y2=sin(2*x);y3=sin(4*x);plot(x,y1,x,y2,x,y3)
```

a při sestavování matice Y, která jako sloupce obsahuje funkční hodnoty.

```
x=0:.01:2*pi; Y=[sin(x)', sin(2*x)', sin(4*x)']; plot(x,Y)
```

Jiný způsob je hold. Příkaz hold zmrazí aktuální grafickou obrazovku a všechny následující grafické výstupy do ní přikresluje. Opětovné zadání příkazu hold ruší příkaz "hold" původní. Ve verzi 4.0 lze použít i příkazy hold on a hold off. Je možné měnit i předdefinované nastavení typů čar a způsob vykreslování bodů, např.

```
x=0:.01:2*pi; y1=sin(x); y2=sin(2*x); y3=sin(4*x); plot(x,y1,'--',x,y2,':',x,y3,'+')
```

vrátí na výstupu první graf čárkovaně, druhý tečkovaně a třetí jen jako bodový - na každém bodě zobrazí +. Druhy čar a znaků pro grafiku jsou:

Čáry: plná (-), čárkovaná (--) tečkovaná (:) a čerchovaná (-.)

Znaky: bod (.), plus (+), hvězdička (*), kroužek (o) a křížek (písmeno x) (x)

Pro nastavení barev viz help plot.

Pro zobrazení grafu jen do části graf. obrazovky lze použít příkaz subplot a prohlížet tak až 4 grafy najednou. Viz help subplot.

Uložení - kopie grafiky

Pozn. překladatele: V originále následuje složitý popis ukládání a tisku grafického výstupu. V novějších verzích MATLABu však stačí zvolit vhodné uložení grafiky v menu a uložený soubor potom upravit nebo odeslat na tiskárnu.

3-D grafika. Trojdimenzionální síťové grafy jsou kresleny funkcí mesh. Příkaz mesh(z) vykreslí trojdimenzoinální perspektivní zobrazení prvků matice **z**. Potah sítě je definován pomocí z-souřadnic bodů nad pravoúhlou rovinou x-y. Vyzkoušejte mesh(eye(10)).

Chceme-li vykreslit graf funkce *z = f(x,y)* na obdélníku, musíme nejprve definovat vektory **xx** a **yy**, které budou udávat dělení stran obdélníka. Funkcí meshdom (mesh domain; ve verzi 4.0 meshgrid) se potom vytvoří matice **x**, Každý její řádek je roven **xx** a délka jejího sloupce je rovna **yy**. Podobně se vytvoří matice **y**, jejíž každý sloupec je roven **yy**. Příkaz:

```
[x,y] = meshdom(xx,yy);
```

Dále se spočítá matice **z**, tu získáme vypočtením hodnot funkce **f** po složkách přes matice **x** a **y**. Na tuto matici **z** již lze aplikovat funkci mesh.

Například graf funkce *z = e^(-x^2-y^2)* nad čtvercem *[-2,2] x [-2,2]* můžete nakreslit takto (vyzkoušejte):

```
xx = -2:.1:2;
yy = xx;
[x,y] = meshdom(xx,yy);
z = exp(-x.^2 - y.^2);
mesh(z)
```

První tři řádky kódu lze samozřejmě nahradit zápisem

```
[x,y] = meshdom(-2:.1:2, -2:.1:2);
```

Další informace týkající se mesh viz manuál.

Ve verzi 4.0 se možnosti MATLABu pro 3-D grafiku výrazně rozšířily, viz on-line help k příkazům plot3, mesh a surf.

19. Přehled

Samozřejmě, že v MATLABu existuje mnoho nástrojů, o kterých jsme se v tomto úvodu nezmínili. Níže jsou uvedeny některé v MATLABu dostupné funkce roztříděné podle skupin. *[Zdroj: MATLAB User's Guide, version 3.5]* Pro jejich bližší vysvětlení použijte on-line help nebo manuál.

Kromě zde vypsaných existuje ještě spousta dalších funkcí, zvláště pak několik "toolboxů" - knihoven funkcí se specifickým použitím (např. pro spliny, optimalizaci a pod). *[Některé rozšiřující knihovny možná nemáte nainstalovány.]* Prostudovat je můžete pomocí příkazu help.

[Poznámka: Některé z následujících tabulek vyžadují prohlížeč, který je schopen pracovat s tabulkami, např. Netscape 1.1 a vyšší.]

Obecné

help	help
demo	spustí demo
who	výpis proměnných v paměti
what	výpis M-souborů z disku
size	rozměr
length	délka vektoru
clear	vymazání proměnných
computer	typ počítače
^C	přerušení práce
exit	ukončení MATLABu
quit	jako exit

Maticové operátory		Skalární operátory	
+	sčítání	+	sčítání
-	odčítání	-	odčítání
*	násobení	.*	násobení
/	dělení	./	dělení
\	dělení zleva	.\	dělení zleva
^	mocnina	.^	mocnina
'	transpozice	.'	transpozice

Relační a logické operátory	
< menší než	& a
<= menší nebo rovno	| nebo
> větší než	~ not
>= větší nebo rovno	
== rovno	
~= není rovno	

Speciální znaky	
=	přiřazovací příkaz
[	používáno k zadávání vektorů a matic
]	viz [
(	priorita aritmetických operátorů
)	viz (
.	desetinná tečka
...	příkaz bude pokračovat na dalším řádku
,	oddělovač hodnot a argumentů funkcí
;	konec řádku s potlačením výstupu
%	komentáře
:	indexování, generování vektorů
!	provádění systémových příkazů

Speciální hodnoty	
ans	proměnná pro výsledek, není-li přiřazena
eps	přesnost
pi	3.14159...
i, j	komplexní jednotka, sqrt(-1)
inf	nekonečno
NaN	Not-a-Number = Není to číslo
clock	čas
date	datum
flops	počet vykonaných operací
nargin	počet vstupních argumentů funkce
nargout	počet výstupních argumentů funkce

Diskové soubory	
chdir	změna aktuálního adresáře
delete	smazání souboru
diary	zaznamenání výstupu
dir	seznam souborů na disku
load	vyvolání proměnných ze souboru
save	zapsání proměnných do souboru
type	vypsání funkce nebo souboru
what	zobrazení M-souborů na disku
fprintf	zápis do souboru
pack	úspora paměti pomocí **save**

Speciální matice	
compan	??? companion ??? - adjungovaná?
diag	diagonální
eye	jednotková
gallery	??? esoteric ???
hadamard	Hadamardova
hankel	Hankelova
hilb	Hilbertova
invhilb	inverzní Hilbertova
linspace	lineární obal vektorů
logspace	logaritmický obal vektorů
magic	magický čtverec
meshdom	dělení pro mesh (viz kap. 18)
ones	konstantní (jedničková)
pascal	Pascalova
rand	náhodné prvky
toeplitz	Toeplitzova
vander	Vandermondeho
zeros	nulová

Manipulace s maticemi	
rot90	otočení
fliplr	převrácení zprava doleva
flipud	převrácení shora dolů
diag	vybrání diagonály
tril	dolní trojúhelníková
triu	horní trojúhelníková
reshape	??? reshape ???
.'	transpozice
:	převedení matice na jediný sloupec; A(:)

Relační a logické funkce	
any	log. podmínky
all	log. podmínky
find	??? find array indices of logical values ???
isnan	detekce nečíselných prvků
finite	detekce hodnot nekonečno
isempty	detekce prázdných matic
isstr	detekce řetězcových proměnných
strcmp	porovnávání řetězcových proměnných

Kontrola běhu výpočtu	
if	podmíněné příkazy
elseif	používáno spolu s if
else	používáno spolu s if
end	ukončení if, for, while
for	daný počet opakování příkazů
while	prováděj dokud
break	vyskočení z for a while cyklů
return	návrat z funkcí
pause	pozastavení, dokud není stisknuta klávesa

Programování a M-soubory	
input	zadávání čísel z klávesnice
keyboard	klávesnice jako M-soubor
error	zobrazení chybového hlášení
function	definice funkce
eval	interpretace textu v proměnné
feval	vyhodnocení funkce zadané řetězcem
echo	umožnění zpráv od příkazů (echo)
exist	test existence proměnných
casesen	nastavení citlivosti na malá/velká písmena
global	definice globálních proměnných
startup	startovní M-soubor
getenv	nastavení prostředí

menu	výběr položky z menu
etime	uplynulý čas

Text a řetězce	
abs	převod řetězce do ASCII hodnot
eval	vyhodnocení textového makra
num2str	převod čísla na řetězec
int2str	převod celého čísla na řetězec
setstr	nastavení značky udávající, že matice je řetězec
sprintf	převod čísla na řetězec
isstr	detekce řetězcových proměnných
strcomp	porovnávání řetězcových proměnných
hex2num	převod hex-řetězce na číslo

Příkazové okno	
clc	smazání příkazové obrazovky
home	přenesení kursoru home (začátek řádku)
format	nastavení formátu výstupu
disp	zobrazení matice nebo textu
fprintf	tisk (výstup) naformátovaného čísla
echo	umožnění zpráv od příkazů (echo)

Grafy - prostředí	
plot	lineární graf X-Y
loglog	loglog graf X-Y
semilogx	semi-log graf X-Y
semilogy	semi-log graf X-Y
polar	polar plot
mesh	3-D síť
contour	contour plot (nákres profilu)
meshdom	dělení pro mesh (viz kap. 18)
bar	sloupcový graf
stairs	hvězdičkovitý graf
errorbar	přidání sloupců chyb (odchylek)

Grafy - popisky	
title	titulek
xlabel	popis osy x
ylabel	popis osy y
grid	vykreslit čtvercovou síť
text	text na zadané souřadnice
gtext	text umístitelný myší
ginput	grafický vstup

Ovládání grafického okna	
axis	ruční nastavení měřítka
hold	ponechání nákresu v graf. okně
shg	zobrazení graf. okna
clg	smazání graf. okna
subplot	rozdělení graf. okna

Uložení grafického okna	
print	odeslání grafu na tiskárnu
prtsc	kopie obrazovky na tiskárnu
meta	grafický metasoubor

Elementární matematické funkce	
abs	absolutní hodnota (velikost komplex. čísla)
angle	úhel fáze
sqrt	odmocnina
real	reálná část
imag	imaginární část
conj	komplexně sdružené číslo
round	zaokrouhlení k nejbližšímu celému č.
fix	zaokrouhlení směrem k nule (celá část)
floor	zaokrouhlení směrem k -nekonečnu (celá část)
ceil	zaokrouhlení směrem k nekonečnu (celá část)
sign	funkce signum
rem	zbytek
exp	mocnění se základem e
log	přirozený logaritmus
log10	dekadický logaritmus

Trigonometrické funkce	
sin	sinus
cos	cosinus
tan	tangens
asin	arcsinus
acos	arccosinus
atan	arctangens
atan2	??? four quadrant arctangent ???
sinh	hyperbolický sinus
cosh	hyperbolický cosinus
tanh	hyperbolický tangens
asinh	hyperbolický arcsinus
acosh	hyperbolický arccosinus
atanh	hyperbolický arctangens

Speciální funkce

bessel	Besselova funkce
gamma	gamma funkce
rat	aproximace racionálním číslem
erf	chybová funkce
inverf	inverzní chybová funkce
ellipk	eliptický integrál prvního druhu
ellipj	Jacobiův eliptický integrál

Dekompozice a faktorizace	
balance	??? balanced form ???
backsub	zpětná substituce
cdf2rdf	??? convert complex-diagonal to real-diagonal ???
chol	Choleskeho rozklad
eig	vlastní čísla a vlastní vektory
hess	Hessenbergův tvar
inv	inverze
lu	factory z Gaussovy eliminace
nnls	nezáporné nejmenší čtverce
null	nulový prostor
orth	ortogonalizace
pinv	pseudoinverze
qr	ortogonálně-triangulační dekompozice
qz	QZ algoritmus
rref	??? reduced row echelon form ???
schur	Schurova decompozice
svd	rozklad podle singulárních hodnot

Maticové podmínky	
cond	??? condition number in 2-norm ???
norm	1-norma,2-norma,F-norma,<infty>-norma
rank	hodnost
rcond	??? condition estimate (reciprocal) ???

Elementární maticové funkce	
expm	meticové mocnění
logm	maticový logaritmus
sqrtm	odmocninová matice
funm	libovolná maticová funkce
poly	charakteristický polynom
det	determinant
trace	stopa
kron	Kroneckerův součin

Polynomy

poly	charakteristický polynom
roots	kořeny polynomu --- ??? companion matrix method ???
roots1	kořeny polynomu --- Laguerrova metoda
polyval	vyhodnocení polynomu
polyvalm	maticové vyhodnocení polynomu
conv	násobení
deconv	dělení
residue	rozvoj na parciální zlomky
polyfit	vložení pol. křivky

Analýza dat po sloupcích	
max	největší hodnota
min	nejmenší hodnota
mean	průměr
median	medián
std	standartní odchylka
sort	třídění
sum	součet prvků
prod	součin prvků
cumsum	celkový součet prvků
cumprod	celkový součin prvků
diff	aproximace derivací
hist	histogramy
corrcoef	korelační koeficienty
cov	kovarianční matice
cplxpair	přeuspořádání do komplexních dvojic

Informace o datech	
abs	velikost komplexního čísla
angle	úhel fáze
conv	konvoluce
corrcoef	korelační koeficienty
cov	kovariance
deconv	dekonvoluce
fft	Fourierova transformace 2. řádu
fft2	dvojdimenzionální FFT
ifft	inverzní Fourierova transformace
ifft2	inverzní 2-D FFT
fftshift	FFT přerovnání

Numerická integrace	
quad	numerická integrace funkce
quad8	numerická integrace funkce

Řešení diferenciálních rovnic

ode23	Runge-Kutteho metoda 2/3 řádu
ode45	Runge-Kutteho-Fehlbergova metoda 4/5 řádu

Nelineární rovnice a optimalizace	
fmin	minimum funkce jedné proměnné
fmins	minimum funkce více proměnných
fsolve	řešení soustavy nelineárních rovnic (nulové hodnoty funkce více proměnných)
fzero	nulová hodnota funkce jedné proměnné

Interpolace	
spline	kubický spline
table1	1-D tabulka
table2	2-D tabulka