Jak pracovat s MATLABem

Jiří Zelinka

11. března 2004

Obsah

Abstrakt

Tento návod je pracovním textem určeným zejména pro studenty mající zapsaný předmět Výpočetní matematické systémy, ale je k dispozici i všem dalším zájemcům o práci s programovým systémem MATLAB. Text bude postupně doplňován. Autor uvítá jakékoliv dobře míněné podněty a připomínky.

Návod je psán pro verzi MATLABu 5.3, u níž se téměř výlučně pracuje s příkazovou řádkou. Od verze 6 má MATLAB okenní strukturu s bohatšími možnostmi, ale příkazová řádka zůstává pro uživatele i nadále základním komunikačním prostředkem. Dá se tedy očekávat, že všechny popsané příkazy budou bez problémů fungovat i ve verzích 6.x, případně vyšších. Většina příkazů by měla být funkční i ve verzích starších.

V textu jsou příkazy MATLABu a jména proměnných uvedeny v uvozovkách, pokud příkaz neleží na samostatném řádku. V apostrofech jsou jednotlivé znaky a textové řetězce.

Kapitola 1

Úvod

1.1 Základní ovládání

Spouštění MATLABu je závislé na použitém operačním systému. Např. v Linuxu zpravidla spouštíme MATLAB zadáním příkazu "`matlab`" v příkazovém řádku. Ve Windows klikneme na příslušnou ikonu nebo program najdeme někde v měnu. MATLAB ukončíme příkazem "`exit`" nebo zavřením okna, v němž MATLAB běží. Příkazy MATLABu zadáváme do příkazového řádku, přičemž můžeme využívat v MATLABu definovaných i vlastních funkci a procedur. Pokud příkaz ukončíme středníkem, MATLAB nezobrazí výsledek provedeného příkazu. Na jeden řádek je možné žádat více příkazu, pak je oddělujeme čárkou nebo středníkem. Pokud chceme příkaz rozdělit na více řádku, můžeme použít tři tečky jako pokračovací znaky. Zadávané příkazy je možné editovat běžným způsobem, předchozí příkaz vyvoláme šipky "`NAHORU`".

1.2 Nápověda

Nápovědu získáme příkazem "`help`", např. "`help sin`" nám dá nápovědu k použití známé goniometrické funkce. Příkaz "`help`" samotný vypíše seznam všech tematicky oblasti, na než je pak možné získat nápovědu opět příkazem "`help`" - např. "`help matfun`". Samostatné okno s nápovědou otevřeme příkazem "`helpwin`". HTML stránky s dokumentaci získáme příkazem "`doc`" a demonstrační program k MATLABu vyvoláme zadáním příkazu "`demo`".

1.3 Jednoduché výpočty

V rámci MATLABu lze provádět libovolně výpočty, které lze počítat na kalkulačce: např. "`3*(15.8+3^"8)-sin(pi/12)+sqrt(4.8)` dá výsledek uvedeného výrazu. Přitom "`pi`" je známé Ludolfovo číslo, "`sqrt`" je odmocnina. Další definované konstanty jsou "`i,j`" pro komplexní jednotku. Známé funkce použitelné v MATLABu jsou například:

`sin, cos, tan, cot, asin, acos, atan, acot,`
`sinh, cosh, tanh, coth, asinh, acosh, atanh, acoth,`
`exp, log, log10, log2, pow2, sqrt, abs.`

Běžné operátory: +, −, *, /, ˆ

Pomoci kulatých závorek řídíme prioritu operátorů a umisťujeme do nich argumenty funkci.

Pro zaokrouhlování se používají následující funkce:

"`round`"	zaokrouhlení k nejbližšímu celému číslu
"`floor`"	zaokrouhlení směrem k minus nekonečnu
"`ceil`"	zaokrouhlení směrem k plus nekonečnu
"`fix`"	zaokrouhlení směrem k nule

1.4 Proměnné, matice a jejich definování

Výsledek jakékoliv operace můžeme uložit do proměnné, kterou lze použít při dalších výpočtech. Název proměnné je libovolná posloupnost písmen, číslic a znaku '_' začínající písmenem. Velikost písmen je důležitá, tj. "`A`" je jiná proměnná než "`a`".

Do proměnné přiřadíme výsledek výrazu pomoci znaku '='. Např. příkaz

```
x=12*sin(pi/20)
```

definuje proměnnou "`x`", kterou můžeme dále použít: "`x8=8*x`".

Proměnné mohou být i maticemi, vlastně i běžná čísla se pokládají za matice typu 1x1. Matice můžeme zadat výčtem jejich prvků v hranatých závorkách, přičemž jednotlivé prvky na řádku oddělujeme mezerou nebo čárkou, jednotlivé řádky matice oddělujeme středníkem. Příklad:

```
A=[1 -1 2 -3; 3 0 4 5;3.2, 5, -6 12]
u=[1 2 3 4]
v=[-1;-2;-3]
```

Prázdnou matici vytvoříme příkazem

```
o=[]
```

Matice lze vytvářet pomoci už definovaných proměnných, je potřeba kontrolovat, aby souhlasily typy jednotlivých proměnných. Př.:

```
y=[x, 2*x, 3*x]
B=[A; u]
C=[A, v]
```

Seznam definovaných proměnných lze získat příkazem "`who`". Příkaz "`whos`" zobrazí seznam proměnných i s jejich typy a dalšími informacemi. Zrušit definovanou proměnnou je možno přikáže "`clear`", např.

```
clear i
```

můžeme zrušit proměnou označenou "`i`", kterou jsem omylem předefinovali komplexní jednotku. Zadáním "`clear all`" zrušíme všechny definované proměnné.

1.5 Funkce pro tvorbu matic

Pro vytvoření matic větších rozměrů je možné použít některých funkci MATLABu. Příkaz

`Z=zeros(2,5)`

vytvoří nulovou matici příslušného typu, příkazem

`O=ones(3,4)`

vyrobíme matici že samých jedniček. Příkaz

`I=eye(5,8)`

nám dá 'jednotkovou' matici, přičemž jedničky probíhají hlavní diagonálu. Všechny výše uvedené příkazy je možné žádat i s jedním parametrem, v tom případě je výsledkem čtvercová matice příslušného řadu. Funguje taky příkaz

`o=ones(0,5)`

Matici s náhodnými prvky lze vytvořit příkazem

`R1=rand(3,5)`,

kdy prvky budou mít hodnoty mezi 0 a 1, nebo příkazem

`R2=randn(6)`

pro matici s prvky majícími standardizovane normální rozdělení. Vektory jsou považovány za matice s jedním řádkem nebo jedním sloupcem. Vektor, jehož prvky tvoří aritmetickou posloupnost, je možné vytvořit následujícím způsobem:

`x=1:10`

vytvoří vektor postupně obsahující čísla 1 až 10. Pro jiný rozdíl mezi jednotlivými prvky než 1 lze použít např. příkaz

`y=0:2:12`

pro y se sudými čísly od 0 do 12 nebo

`z=0:0.01:2`

pro z s čísly od 0 do 2 s krokem 0.01. Je možno použít i záporný krok:

`x1=10:-1:1`

1.6 Další operace s maticemi

Rozměr matice zjistíme příkazem "`size`". Na jednotlivé prvky matice je možné se odkázat pomoci kulatých závorek - tj. "`A(3,2)`" je prvek matice "`A`" ve 3. řádku a 2. sloupci. Toto vyjádření ovšem lze použít obecněji, kdy první parametr je vektor obsahující indexy vybraných řádku a druhý parametr je vektor sloupcových indexu. Pak

`A([1 3],[5 2])`

je vybrána submatice z "`A`" tvořena 1. a 3. řádkem a 5. a 2. sloupcem. Je možno taky provádět přiřazení do takto vybraných prvků při zachování sprvnych rozměrů:

```
A([1 3],[5 2])=eye(2)
```

Pokud chceme vybrat celý řádek nebo sloupec použijeme symbol ':', např. "B(3,:)" je 3. řádek matice "B". Příkazem "B(3,:)=[]" vypustíme z matice B tento řádek. Zajímavé je, že nefunguje přiřazení

```
o=[];
B(3,:)=o
```

Matic stejných rozměrů lze sčítat a odčítat $(+,-)$, matice vhodných rozměrů se dají násobit (*). Je také možné násobit konstantou nebo přičíst konstantu k matici, pak se přičte ke všem prvkům. Dělení jsou v MATLABu dvě – pravé a levé: '\', '/', přičemž výraz

```
X=B\C
```

dá řešení maticové rovnice `B*X=C` a výraz

```
X=B/C
```

je řešením maticové rovnice `X*C=B`.

Symbolem " ' " dostaneme hermitovskou transpozici matice, tj. matici transponovanou a komplexně sdruženou. Operátor ˆ slouží k umocňování čtvercových matic. Uvedené operátory mají také tzv. tečkové varianty, kdy se operace provádějí na jednotlivé prvky matice, např.

```
A=B.*C
```

vynásobí odpovídající prvky matic "B" a "C", příkazem

```
A.^2
```

se umocní všechny prvky matice "A". Operátor " .' " se používá pro transpozici matic. Na matice lze aplikovat taky všechny běžné funkce ("sin", "cos",…), které se provádějí člen po členu.

1.7 Příklady

Vyzkoušejte následující příkazy:

```
x=0:0.01:2*pi;
y=sin(x.^2).*cos(3*x);
plot(x,y)
```

1.8 Tipy a triky

Pokud máme matici "A" typu $m \times n$, pak pokus o určení hodnoty "A(k,l)", kde k> m nebo l> n, skončí chybou. Ovšem příkaz "A(k,l)=1" přiřazení provede, přičemž chybějící členy jsou doplněny nulami.

Náhodnou matici s celočíselnými členy v daném rozmezí lze vytvořit vhodnou kombinací funkce "rand" a některé zaokrouhlovací funkce. Například náhodnou matici řádu 4 s prvky mezi -5 a 5 získáme příkazem

```
A=round(10*rand(4))-5
```

Kapitola 2

Příkazy MATLABu

2.1 Obecná pravidla

Na jednu řádku můžeme zadat několik příkazu oddělených čárkou nebo středníkem, který potlačuje výstup na obrazovku:

`příkaz1, příkaz2, příkaz3,` atd.,

přičemž za posledním příkazem na rádce čárka být nemusí. Pokud je příkaz příliš dlouhý, můžeme ukončit řádku třemi tečkami "..." a příkaz pak pokračuje na další rádce.

Jednotlivé příkazy mohou být jednoduché příkazy MATLABu ("`who`", "`clear`", "`dir`"), příkazy definující proměnné ("`A=[1 2;3 4];`"), volání MATLABovskych programů – tzv. skriptů, nebo volání funkci MATLABu. Toto volání má v obecném případě tvar:

```
[v1,v2,...,vm]=jmeno_funkce(p1,p2,...,pn)
```

"`v1, ..., vm`" jsou výstupní parametry, "`p1,..., pn`" jsou parametry vstupní. Pokud je výstupní parametr jen jeden, nemusí být uzavřen v hranatých závorkách. Funkce také nemusí mít žádné vstupní nebo výstupní parametry, také je možné zadávat různý počet vstupních nebo výstupních parametrů pro tutéž funkci. Vyzkoušejte například:

```
A=rand(3);
s=size(A)
[x,y]=size(A)
```

Vykřičník na začátku příkazu způsobí vykonání příkazu operačního systému, např. příkazem "`!netscape`" spustíme známý internetový prohlížeč.

Symbol '%' na příkazové rádce způsobí ignorování zbytku řádky (komentář).

2.2 Některé speciální výrazy a funkce

Zvláštní postavení mezi proměnnými má proměnná "`ans`" (z answer), do které se automaticky přiřazují hodnoty, jež nebyly přiřazeny explicitně – stačí zadat např. "`pi`" . Dále "`i`" a "`j`" jsou imaginární jednotky pro práci s komplexními čísly. Další speciální hodnotou je "`eps`" – je to rozdíl mezi 1 a nejbližším vyšším zobrazitelným číslem. Příkazem "`realmin`" a "`realmax`" zjistíme hodnotu nejmenšího, resp. největšího zobrazitelného čísla podle normy IEEE. Hodnoty "`Inf`" a "−`Inf`" vznikají např. při dělení nulou a symbolizují nekonečné hodnoty (+ nebo −). Hodnotu "`NaN`" (Not a Number) dostaneme jako výsledek neurčitého výrazu – např. 0/0.

Způsob výpisu hodnot můžeme ovlivnit příkazem "`formát`":

`formát long`	výpis na plný počet desetinych míst
`formát short`	výpis na omezený počet desetinych míst (implicitní)
`formát hex`	hexadecimální výpis
`formát rat`	čísla jsou aproximovaná zlomky
`formát compact`	potlačí se vynechávání řádku při výpisu
`formát loose`	zapne se vynechávání řádku při výpisu

2.3 Manipulace s maticemi

Základní příkazy pro práci s maticemi byly uvedeny v první kapitole. Pro vytváření vektoru, jeho prvky tvoří konečnou aritmetickou posloupnost je možné kromě znaku ':' použít také příkaz "`linspace`":

`x=linspace(a,b,n)`,

kde "`a`", "`b`" jsou první a poslední prvek vektoru "`x`" a "`n`" je počet prvků. Třetí parametr je nepovinný a pokud není uveden, bere se roven 100. Podobně je taky možné definovat vektor s desitkovou logaritmickou škálou:

`x=logspace(a,b,n)`,

který je ekvivalentní příkazu

`x=10.^linspace(a,b,n)`.

Zde je ovšem implicitně n=50.

Příkazem "`reshape`" je možné přeskládat matici do jiného typu. Zadáním

`A=rand(10); B=reshape(A,5,20);`

vytvoříme z prvků matice "`A`" matici "`B`" o 5 řádcích a 20 sloupcích, přičemž prvky se při přeskládání berou po sloupcích. Pokud bychom chtěli matici "`A`" přeskládat po řádcích, museli bychom nejdřív matici "`A`" transponovat a pak opět transponovat výsledek:

`B=reshape(A',5,20)'`

Překlopit matici, tj. obrátit pořadí řádku nebo sloupců je možné provést příkazem "`flipud`" resp. "`fliplr`". Také je možné matici „*otočit*“ o 90 stupňů příkazem "`rot90`", přičemž další nepovinný parametr udává, kolikrát se má rotace provést.

2.4 Další funkce pro práci s maticemi

Všechny operátory pro práci s maticemi uvedené v první kapitole mají své funkci ekvivalenty:

funkce	význam	operátor
`plus`	plus	+
`uplus`	unární plus	+
`minus`	minus	−
`uminus`	unární minus	−
`mtimes`	maticové násobení	*
`times`	násobení po prvcích	.*
`mpower`	maticová mocnina	ˆ
`power`	mocnina po prvcích	.ˆ
`mldivide`	levé maticové dělení	\
`mrdivide`	právě maticové dělení	/
`ldivide`	levé dělení po prvcích	.\
`rdivide`	právě dělení po prvcích	./

Příkaz "`a:b`" případně "`a:d:b`" lze nahradit příkazem "`colon(a,b)`" nebo "`colon(a,d,b)`".

Dvojí úlohu má funkce "`diag`". Jeho aplikaci na vektor získáme diagonální matici s argumentem na hlavní diagonále. Pokud ji použijeme na matici, funkce "`diag`" vybere z matice hlavní diagonálu a umístí ji do vektoru. Pokud chceme pracovat s jinou diagonálou než s hlavní, můžeme použít jako druhý parametr funkce "`diag`" číslo diagonály, přičemž kladná čísla se použijí nad hlavní diagonálou a záporná pod ní. Příklad:

```
A=rand(5)
x=diag(A)
B=diag(x,2)
C=diag(pi,-4)
d=diag(diag(A))
```

Pomoci funkci "`tril`" a "`triu`" vyrobíme z daně matice dolní nebo horní trojúhelníkovou matici, přičemž je možné podobně jako u "`diag`" použít další nepovinný parametr.

Příkaz "`max`" najde maximální prvek ve vektoru, přičemž pokud na výstupu uvedeme i druhý výstupní parametr, uloží se do něj index tohoto prvku. Při použití příkazu "`max`" na matici se hledají maximální prvky v jednotlivých sloupcích. Podobně funguje příkaz "`min`":

```
x=rand(1,6)
M=max(x)
[m,k]=min(x)
A=rand(4)
[m,k]=min(A)
```

U komplexních matic se maximum či minimum hledá mezi absolutními hodnotami prvků.

K setřídění vektoru (vzestupné) se používá příkaz "`sort`", u komplexních hodnot se opět provádí třídění podle absolutních hodnot. Do druhého nepovinného výstupního parametrů je umístěna třídicí permutace indexu. Tj. pokud zadáme "`[y,ind]=sort(x)`" pak platí `y=x(ind)`. Při použití příkazu "`sort`" na matice se třídí jednotlivé sloupce.

Příkazy "`sum`" a "`prod`" sečtou nebo vynásobí všechny prvky vektoru. Aplikovaný na matici provedou příslušnou operaci po jednotlivých sloupcích. Např. faktoriál čísla n zjistíme snadno pomoci příkazu "`f=prod(1:n)`", přičemž příkaz funguje správné i pro "`n`" rovno 0, protože součin přes prázdnou matici je roven 1.

Pokud bychom chtěli zjistit součet všech prvků matice, musíme příkaz "`sum`" použít dvakrát:

```
sum(sum(A))
```

Příkazem "`size`" zjistíme rozměry matice, přičemž je možné jej použít ve formě "`s=size(a)`" nebo "`[m,n]=size(a)`". Příkaz "`length`" dává délku vektoru, přičemž pokud jej aplikujeme na matici, dostaneme maximální z rozměrů, tj. "`length(A)`" dává stejný výsledek jako "`max(size(A))`".

2.5 Základní funkce lineární algebry

V MATLABu je implementováno velké množství funkci a algoritmu lineární algebry. Podrobný výpis lze získat pomoci nápovědy "`help matfun`".

Zde vyjmenujeme jen některé základní:

`det`	determinant matice
`rank`	hodnost matice
`norm`	maticová nebo vektorová norma
`trace`	stopa matice (součet diagonálních prvků)
`inv`	inverzní matice
`pinv`	pseudoinverzni matice
`lu`	LU rozklad
`qr`	QR rozklad
`svd`	singulární rozklad
`eig`	vlastní hodnoty a vektory matice
`póly`	charakteristický polynom matice

Syntaxi jednotlivých funkci zjistíme nejlépe pomoci nápovědy.

2.6 Tipy a triky

Pomocí funkce "`sort`" lze náhodně zpermutovat danou konečnou posloupnost čísel. Chtějme například vytvořit náhodnou permutaci čísel od 1 do 10. Vytvořme nejprve pomocný náhodný vektor daného rozměru:

`p=rand(1,10)`, pak vektor "`p`" setřídíme:

```
[p,I]=sort(p);
```

a v proměnné "`I`" máme hledanou náhodnou permutaci. V případě, že bychom chtěli zpermutovat prvky nějakého vektoru "`x`", stačí zadat

`x=x(I)`,
kde "`I`" je třídící permutace pro indexy náhodného vektoru stejného rozměru jako "`x`".

Kapitola 3

Práce se soubory

3.1 Záznam práce

Prováděné příkazy i s výsledky je možné zaznamenávat do tzv. žurnálu. Toho dosáhneme pomoci příkazu "`diary`". Příkaz "`diary on`" zapne ukládání a všechny další příkazy i jejich výsledky budou ukládány v pracovním adresáři do souboru s názvem "`diary`" v prcovním adresáři. Příkaz "`diary off`" toto ukládání přeruší a uzavře soubor, přičemž nové použití "`diary on`" nesmaže původní obsah souboru. Použití samotného příkazu "`diary`" bez parametrů přepne režim ukládání, tj. pokud bylo ukládání zapnuto, pak ho vypne a naopak.

Ukládání do souboru s jiným jménem dosáhneme pomoci příkazu "`diary jmeno_souboru`" nebo "`diary('jmeno_souboru')`", kde "`jmeno_souboru`" je libovolně jméno souboru. Další použití příkazu "`diary`" se pak bude vztahovat k uvedenému souboru.

3.2 Ukládání a načítání proměnných

Pro uložení proměnných do souboru slouží příkaz "`save`". Jeho použití bez parametrů způsobí uložení všech definovaných proměnných do souboru s názvem "`matlab.mat`". Pro uložení jen vybraných proměnných do námi zvoleného souboru žádáme jako první parametr jméno souboru a jako další parametry názvy proměnných, které chceme uložit. Tedy příkaz

```
save data
```

uloží všechny definované proměnné do souboru data a příkaz

```
save data1 a b c
```

uloží do souboru "`data1.mat`" proměnné s názvy "`a`", "`b`" a "`c`".

Stejný výsledek dá příkaz

```
save('data1','a','b','c').
```

Uložené soubory mají automaticky příponu `mat`. Je možné zadat i příponu jinou.

Načíst data z uloženého souboru můžeme načíst příkazem "`load`". Použijeme-li ho bez parametrů, načtou se všechny uložené proměnné že souboru "`matlab.mat`". Jinak je možné použít příkaz

`load`

podobně jako příkaz "`save`", tedy např.

`load data1`

nebo `load('data1')`

načte všechny proměnné uložené v souboru "`data1.mat`". Pokud soubor s uloženými daty má jinou příponu než "`mat`", musí se při načítání jeho obsahu použít parametr "`-MAT`":

`load data1.dat -MAT` (lze použít i malá písmena: "`-mat`").

Data lze uložit do souboru i v tzv. ASCII tvaru, který je běžné čitelný v textovém editoru. Dosáhneme toho užitím volby "`-ASCII`":

`save data2.dát X -ASCII`

V tomto souboru ovšem není uloženo jméno proměnné "`X`", pouze její obsah. Při načítání souboru, který má jinou koncovku než "`mat`", se automaticky předpokládá, že jsou v ASCII tvaru. Z takového souboru je možné ovšem načíst pouze jednu proměnnou, která má navíc stejné jméno, jako původní soubor (bez přípony). Takové soubory je možné vytvářet i ručně, případně jako výsledek práce jiných programu. Je ovšem nutno mít na paměti, že data v nich obsažena musí mít tvar matice, tj. každý řádek musí mít stejný počet sloupců.

3.3 Soubory v systému MATLAB

Matlab většinu příkazu, které provádí, hledá v souborech, které tyto příkazy obsahují jako funkce. Přípona těchto souboru je "`m`", proto se taky nazývají m-soubory (m-file). Tyto soubory obsahují jednak zápis algoritmu, pomoci něhož se provádí daný výpočet či dané operace a taky nápovědu, která se vypisuje příkazem "`help`".

Některé funkce jsou tzv. vnitřní, ty jsou uloženy v předkompilovane podobě v knihovně funkci a příslušný m-soubor obsahuje jen nápovědu. Pomoci příkazu "`which`" zjistíme, kde se soubor s daným programem či funkci nachází, nebo zda se jedna o vnitřní funkci MATLABu:

```
which poly
which eig
```

Další funkce mohou být uloženy v tzv. mex-souborech (s příponou "`mex`"). Ty jsou vytvořeny v některém jiném programovacím jazyce (C, FORTRAN) a jsou speciálně zpracovány, aby je bylo možné používat v MATLABu.

Datové soubory, jak už bylo řečeno, mají standardně příponu "`mat`".

3.4 Cesta k souborům

Programy a funkce spouštěné při práci v MATLABU se hledají v adresářích, do nichž je nastavená cesta – tzv. "`matlabpath`". Její obsah zjistíme příkazem "`path`" nebo "`matlabpath`". Pokud chceme do této cesty přidat nějaký další adresář, můžeme to udělat pomoci příkazu "`path`":

```
path(path,'dalsi_adresar')
```

nebo "`addpath`:

```
addpath dalsi_adresar
```

Proměnná "`dalsi_adresar`" musí obsahovat řetězec s daným adresářem. Konkrétní tvar závisí na použitém operačním systému, např. v MS Windows bychom mohli zadat

```
path(path,'d:\user\matlab')
```

či

```
addpath d:\user\matlab
```

a v UNIXu

```
addpath /home/user/matlab
```

Pokud na konci příkazu "`addpath`" přidáme přepínač "`-BEGIN`", uloží se nový adresář na začátek seznamu a bude pak prohledáván jako první. Uložení na konci je možné zadat přepínačem "`-END`". Ke smazání adresáře že seznamu slouží příkaz "`rmpath`".

3.5 Další příkazy pro práci se soubory

Výpis obsahu pracovního adresáře zjistíme příkazem "`dir`" nebo "`ls`". Můžeme také použít hvězdičkovou konvenci – příkazem "`dir *.mat`" zjistíme všechny datové soubory v daném adresáři.

Pracovní adresář zjistíme příkazem "`pwd`", který vrací jako hodnotu řetězec s témto adresářem. Je tedy možné jej přidat do cesty pro prohledávání:

```
d=pwd;
addpath(pwd)
```

Změnit pracovní adresář můžeme příkazem "`cd`", např.:

```
cd matlab/data
```

Pro výpis obsahu zvoleného souboru můžeme použít příkaz "`type`", přičemž výpis dlouhého souboru se zastavuje po stránkách, pokud předtím žádáme příkaz "`more on`". Vypnutí stránkování provedeme příkazem "`moře off`".

Kopírovat libovolně soubory v rámci MATLABu lze pomoci příkazu "`copyfile`", mazat je můžeme příkazem "`delete`". Příkaz "`lookfor`" prohledává nápovědu pro zadaný výraz:

```
lookfor poynomial
```

Editor pro psaní souboru lze vyvolat příkazem "`edit`", jeho nastavení závisí na operačním systému. V některých verzích MATLABu lze tento editor vyvolat z menu.

Kapitola 4

Logické operace

4.1 Relační operátory

Relační operátory slouží k porovnávání proměnných. Jedna se o následující operátory:

”==”	test rovnosti
”~=”	test nerovnosti
”<”	menší
”<=”	menší nebo rovno
”>=”	větší nebo rovno
”>”	větší

Porovnávat je možné jen matice stejných rozměrů nebo matici se skalárem. Při porovnávání se vytvoří matice příslušných rozměrů obsahující jedničky a nuly podle toho, zda je příslušná relace splněna nebo není. Vyzkoušejte třeba:

```
A=[1 2 3; 4 5 6];
B=[1 1 1; 8 7 6];
c=5;
X=(A==B)
Y=(A>=B)
Z=(A<c)
```

Při porovnávání komplexních proměnných se porovná pouze reálná část kromě testu na rovnost nebo nerovnost.

4.2 Logické operátory

Mezi logické operátory patří:

”&”	logické ’a současně’
”|”	logické ’nebo’
”~”	logicky zápor
”xor”	vylučovací ’nebo’ (buď a nebo)

Jednotlivé operátory se aplikují na matice, případně na matici a skalár, podobně jako relační operátory. Produktem operace je matice odpovídajícího typu s jedničkami a nulami podle výsledků daně operace. Jednotlivé operace se provádějí po prvcích. Přitom nula je brána jako nepravdivá hodnota a nenulové číslo (včetně "Inf" a "NaN") se bere jako pravdivá hodnota. Př.:

```
A=-2:2;
B=zeros(1,5);
C=ones(1,5);
X1=A|B
X2=A|C
X3=~A
X4=xor(A,pi)
```

Místo "`A&B`" můžeme psát "`and(A,B)`", místo "`A|B`" "`or(A,B)`" a místo "`~A`" "`not(A)`".

4.3 Logické funkce

Logické funkce vrací jako hodnoty matice či skaláry obsahující nuly s jedničky podle typu daně funkce:

`isinf(A)`
: vrací matici stejného typu jako "`A`" s jedničkami na místech, kde je hodnota "`Inf`" nebo "`-Inf`", na zbylých místech jsou nuly

`isnan(A)`
: vrací matici stejného typu jako "`A`" s jedničkami na místech, kde je hodnota "`NaN`", na zbylých místech jsou nuly

`isfinite(A)`
: vrací matici stejného typu jako "`A`" s jedničkami na místech, kde nejsou hodnoty "`Inf`", "`-Inf`" nebo "`NaN`", na zbylých místech jsou nuly

`all(A)`
: je-li "`A`" vektor, funkce vrací jedničku, pokud jsou všechny hodnoty nenulové, jinak vrací nulu. Je-li "`A`" matice, aplikuje se funkce "`all`" na jednotlivé sloupce matice "`A`" a výsledkem je řádkový vektor obsahující výsledky pro jednotlivé sloupce.

`any(A)`
: funguje podobně jako "`all`", vrací jedničku pokud je alespoň jeden prvek nenulový.

`isempty(A)`
: vrací 1, pokud "`A`" je prázdná matice

`isreal(A)`
: - vrací 1, pokud "`A`" je reálná matice (ne komplexní)

`islogical(A)`
: - vrací 1, pokud "`A`" je logická matice. Příznak logické matice mají všechny výsledky relačních a logických operátorů a logických funkci.

`isstr(A)`
: - vrací 1, pokud je "`A`" textová matice

`isnumeric(A)`
: - vrací 1, pokud je "`A`" číselná matice (ne textová)

4.4 Funkce `find` a `exist`

Další užitečnou funkci je funkce "`find`", která vrací indexy nenulových prvků vektoru nebo matice. Použití je následující: Pro vektor "`x`" příkaz

`I=find(x)`

umístí do proměnné "`I`" indexy nenulových prvků vektoru "`x`". Pro matici "`A`" příkaz

`I=find(A)`

najde indexy nenulových prvků bráno po sloupcích. Příkaz

`[I,J]=find(A)`

umístí do "`I`" řádkové indexy a do "`J`" sloupcové indexy nenulových prvků matice "`A`". Příkaz

`[I,J,H]=find(A)`

umístí navíc do proměnné "`H`" příslušné nenulové prvky. Pokud chceme zjistit hodnoty nenulových prvků i pro vektor, musíme příkaz použít stejně jako pro matici, tedy

`[I,J,H]=find(X)`

Typické použití příkazu `find` je například

`[I,J]=find(A>0)`

kterým najdeme indexy kladných prvků matice "`A`".

Funkcí "`exist`" testujeme existenci a typy objektů MATLABu. Výsledkem příkazu "`ex=exist('A')`" jsou následující možnosti proměnné "`ex`":

0	objekt se jménem "`A`" neexistuje
1	"`A`" je proměnná
2	"`A`" je m-soubor v běžném adresáři nebo v MATLABPATH
3	"`A`" je MEX-soubor v běžném adresáři nebo v MATLABPATH
4	"`A`" je kompilovana funkce v systému SIMULINK
5	"`A`" je interní (předkompilovaná) funkce MATLABu
6	"`A`" je předkompilovaná m-funkce s názvem "`A.p`". Soubor "`A.p`" lze vytvořit pomoci příkazu pcode (viz. "`help pcode`")
7	"`A`" je adresář

Při kolizi jmen (existují různé objekty téhož jména) platí následující priorita:

1. proměnná
2. interní funkce
3. funkce MEX (napřed pracovní adresář, pak MATLABPATH od začátku)
4. příkaz jako p-soubor (napřed pracovní adresář, pak MATLABPATH od začátku)
5. příkaz jako m-soubor (napřed pracovní adresář, pak MATLABPATH od začátku)

Funkci "`exist`" je možné také použít se dvěma parametry:

`exist('A','var')`	testuje jen, zda "`A`" je proměnná
`exist('A','builtin')`	testuje jen, zda "`A`" je interní funkce
`exist('A','file')`	testuje jen, zda "`A`" je soubor
`exist('A','dir')`	testuje jen, zda "`A`" je adresář

Vrácena hodnota je pak 1 (existuje) nebo 0 (neexistuje).

4.5 Smíšené operace

Jednotlivé operace (logické i aritmetické) je možné vzájemné efektivně kombinovat. Priorita operaci je následující (od nejvyšší po nejnižší):

1. umocňování: `^ .^`
2. násobení, dělení: `* .* \ .\ / ./`
3. sečítání, odčítání: `+ -`
4. relační operátory: `== ~= < <= > >=`
5. negace: `~`
6. logické spojky 'a' a 'nebo': `& |`

4.6 Příklady

1. Utvořme matici "`A`":

```
A=round(20*rand(5)-10)
```

tj. "`A`" je náhodná matice řádu 5 s celočíselnými prvky mezi -10 a 10. Naším úkolem je vytvořit matici "`B`" mající hodnotu 10 na místech, kde má "`A`" kladné prvky, na ostatních pozicích má matice "`B`" stejné prvky jako "`A`".
Příkaz "`(A>0)`" umístí na hledaná místa 1 jinam 0. To je možné využít následovně:

```
B=A.*(A<=0)+10*(A>0)
```

2. Další možnosti je využít příkaz "`find`". Zadáním

```
I=find(A>0)
```

najdeme příslušné indexy. Chybou by teď bylo

```
A(I)=10
```

poněvadž tím by se matice "`A`" předefinovala. Správě řešení je

`B=A; B(I)=10;`
Celé je to ovšem možné zkrátit:
`B=A; B(find(A>0))=10`

4.7 Tipy a triky

```
Počet nulových prvků libovolně matice "X" můžeme zjistit např. jako:
sum(sum( X))
Počet nenulových prvků matice "X" bychom mohli získat příkazem
prod(size(X))-sum(sum( X))
ale efektivněji vypadá
length(find(X))
```