Příloha A

Základy programování v Matlabu

MATLAB je program firmy MathWorks určený pro numerické výpočty. Jedná se vlastně o velmi jednoduchý interaktivní programovací jazyk, který umožňuje přirozený zápis matematických výrazů, včetně maticových operací. Jednotlivé příkazy lze zadávat buď přímo v interaktivním módu na příkazové řádce nebo vytvářet programy a funkce, které lze později opakovaně spouštět. MATLAB je vybaven rozsáhlými knihovnami funkcí, které jsou psány přímo v MATLABovém jazyce, takže je možné se podívat, jak tu kterou úlohu řešili analytici firmy MathWorks.

MATLAB pracuje v podstatě jen s jedním typem dat a tím je matice (reálná nebo komplexní). To znamená, že všechny proměnné jsou matice. Číslo je vlastně jen matice typu 1×1, vektor pak matice o jednom řádku nebo jednom sloupci. Obsah těchto matic může být buď zadán ručně, vygenerován funkcí nebo načten z externího datového souboru. Rozměr matice není potřeba explicitně určovat – MATLAB jejich rozměry upravuje automaticky sám.

A.1 Základní operace s maticemi

Zadávání matic

Proměnnou (tedy místo v paměti počítače, kde je uloženo číslo, vektor nebo matice) vytvoříme prostě tak, že jí přiřadíme nějakou hodnotu. Přiřazení hodnoty se děje pomcí operátoru =. Na levé straně přiřazovacího příkazu stojí jméno proměnné, které přiřazujeme hodnotu, na pravé pak nějaký výraz. *Výraz* je matematický vzorec, který MATLAB nejprve vyhodnotí (tj. spočítá jeho hodnotu) a pak přiřadí tuto hodnotu dané proměnné. Stejným způsobem můžeme také přiřadit novou hodnotu již existující proměnné.

Pozor! MATLAB rozlišuje malá a velká písmena. Proměnné `ahoj`, `Ahoj` a `AHOJ` jsou tedy tři různé proměnné. Mějte také na paměti, že jedna proměnná může obsahovat vždy jen jednu hodnotu (jedno číslo, jeden vektor nebo jednu matici); pokud tedy nějaké proměnné přiřadíte novou hodnotu, původní hodnota je nenávratně ztracena.

Nejjednodušší způsob, jak zadat hodnoty matice, je zadat MATLABu přímo hodnoty této matice. Hodnoty matice se uzavírají do hranatých závorek. Jednotlivé prvky matice v řádku se oddělují buď mezerou nebo čárkou, sloupce se oddělují středníkem nebo koncem řádku.

Například příkaz

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

vytvoří matici o třech řádcích a třech sloupcích a na obrazovce vypíše

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

Tím je vytvořena matice rozměrů 3×3, kterou si MATLAB uloží pro další zpracování.

Pokud chcete zabránit vypisování matic na obrazovku, zapište za konec přiřazovacího příkazu středník. To platí obecně – středník na konci příkazu potlačuje jeho výstup na obrazovku ve znakovém režimu.

Stejného výsledku jako v předchozím případě dosáhneme, když zapíšeme

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

Pokud chcete zadat vektor, pak stačí zadat jednořádkovou nebo jednosloupcovou matici. Číslo lze vytvořit stejně; pro jednoduchost je dokonce možné vynechat hranaté závorky, tedy např.

```
5 a=5;
```

Některé speciální matice je možné vytvořit pomocí funkcí `eye`, `ones`, `rand`, `randn` a `zeros`. Tyto funkce mají buď jeden nebo dva parametry, které určují počet řádků a sloupců vytvořené matice (jeden parametr znamená čtvercovou matici), a vytvářejí postupně jednotkovou matici (na diagonále jedničky, jinak nuly), matici jedniček (samé jedničky a nic než jedničky), matici náhodných čísel s rovnoměrným rozdělením v intervalu $\langle 0, 1\rangle$, matici náhodných čísel s normálním normovaným rozložením a nulovou matici. Například

```
6 B=rand(5,3)
```

vytvoří matici náhodných čísel `B` velikosti 5×3 (tedy 5 řádků a 3 sloupce) a vypíše výsledek na obrazovku.

Jak už bylo řečeno, při zadávání matic je možné použít i libovolné výrazy. Například

```
7 C=[-1.3 sqrt(3) (1+2+3)*4/5 0.5*a]
```

vytvoří matici a vypíše na obrazovku

$$\boldsymbol{C} = \begin{pmatrix} -1.3000 & 1.7321 & 4.8000 & 2.5000 \end{pmatrix}.$$

Je také možno vytvořit vektor rozsahem. Tento rozsah se zapisuje jako dvě nebo tři čísla oddělená dvojtečkou. První číslo znamená počáteční hodnoutu rozsahu, poslední koncovou. V tomto případě se vytvoří vektor od prvního do posledního čísla rozsahu s krokem 1. V případě tří čísel je prostřední číslo krok. Například

```
8 D=50:57
```

vypíše

$$\boldsymbol{D} = \begin{pmatrix} 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 \end{pmatrix}.$$

Rozsah „s krokem“ může být například zapsán takto:

```
9 F=0.3:0.2:1
```

Jeho výsledkem je vektor

$$\boldsymbol{F} = \begin{pmatrix} 0.3 & 0.5 & 0.7 & 0.9 \end{pmatrix}.$$

Pokud napíšeme jen výraz bez toho, aby se přiřadil nějaké proměnné (tj. bez jména proměnné a bez symbolu =), potom se výraz vyhodnotí a přiřadí do implicitní proměnné `Ans`. Například pokud napíšeme

```
10 \verb|1900/91|
```

vyhodnotí MATLAB výraz, uloží ho do proměnné `Ans` a zobrazí na obrazovce.

Práce s prvky matice

Někdy můžeme chtít pracovat pouze s částí matice nebo přímo s vybranými prvky matice. Pokud chceme pracovat jen s jedním prvkem matice, můžeme ho indexovat podobně jako pole v klasických programovacích jazycích (nebo matici v matematice). Za jméno matice napíšeme do kulatých závorek dvě čísla – první určuje zvolený řádek, druhé zvolený sloupec. Například napíšeme:

```
11 A=[1 2 3;4 5 6;7 8 9];
12 a=A(3,1)
```

MATLAB odpoví $a = 7$. (Všimněte si, že na první řádek MATLAB nic nevypsal na obrazovku, protože přiřazení je následováno středníkem.)

V případě, že chceme pracovat s podmnožinou matice, postupujeme v podstatě stejně, pouze místo jednoduchých čísel zadáváme rozsah vybraných řádků a sloupců. Tento rozsah je vlastně vektor a může tedy být zadán stejným způsobem jako jakýkoli jiný řádkový vektor. Při stejné matici `A` můžeme tedy napsat třeba

```
13 v=A(1:2,[1 3])
```

MATLAB přiřadí výsledek do vektoru `v` a vypíše výsledek:

$$\boldsymbol{v} = \begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}.$$

Je to proto, že první rozsah, řádky je zadán jako `1:2` – tedy vektor rozsahu od prvního do druhého řádku – jsou vybrány první a druhý řádek; druhý rozsah je zadán přímo: je vybrán první a třetí sloupec. Výsledkem je průnik těchto dvou rozsahů (řádků a sloupců).

V případě vektorů (tedy matic o jednom řádku nebo jednom sloupci) stačí zadat jen jedno číslo nebo rozsah – ten se bere vždy v tom směru, ve kterém to má smysl.

Výběr z matice může být použit v libovolném výrazu. Navíc ho můžeme napsat i na straně přiřazované proměnné. Je tedy možné zapsat například

```
14 A(1,1)=A(3,3
```

Povede to k tomu, že se hodnota třetího řádku třetího sloupce matice `A` překopíruje i do prvního řádku prvního sloupce.

Je také možné napsat `A(:,3)` – znamená to, že chceme pracovat jen se třetím sloupcem (všechny řádky, třetí sloupec). Znak dvojtečka zde znamená „celý dostupný rozsah“ – tedy buď všechny řádky nebo všechny sloupce.

Maticové operace

Operace uvedené v tomto oddíle se vztahují adekvátním způsobem jak na čísla, tak na vektory i na matice (připomínáme, že číslo je vlastně matice typu 1×1) – nicméně výsledky mohou být různé: je rozdíl v násobení čísel a matic a rozdíl v násobení matic a násobení matic po prvcích.

Transpozice matic. Transpozice se provádí pravým apostrofem (') (na anglické klávesnicije umístěn na levém malíčku; existuje ještě levý apostrof, ten však není v MATLABu využit). Předpokládejme následující matici a vektor:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}.$$

Jejich transpozice se provede příkazy `B=A'`, respektive `y=x'`. Výsledkem je matice `B` a vektor `y`

$$\boldsymbol{B} = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Je zde také vidět, jak je možné snadno vytvářet sloupcové vektory transpozicí řádkových.

Sčítání a odečítání. Sčítání a odečítání se provádí znamínky plus (+) a minus (−). Je ovšem potřeba si uvědomit, jak se (z matematického hlediska) sčítají matice, vektory a čísla navzájem – MATLAB to dělá právě tak. Především je potřeba dbát na to, aby matice a vektory měly správný počet řádků a sloupců. Příklad:

```
15 D=A+B-C
```

Pro zopakování: Matice lze z matematického hlediska sečíst, pokud mají stejný počet řádků a sloupců. Vektory lze sečíst, pokud jsou stejně dlouhé a stejného typu (tj. oba řádkové a oba sloupcové). Čísla lze sčítat a odečítat bez omezení. V každém z těchto případů je výsledkem číslo, vektor nebo matice, kde každý prvek je součtem odpovídajících prvků sčítanců.

Pokud sečítáme vektor a číslo nebo matici a číslo, mohou být jejich rozměry libovolné; číslo se přičte ke všem prvkům vektrou (matice). Pokud sečítáme vektor a matici, musí délka vektoru stejná jako je délka odpovídající strany matice (tj. řádkový vektor musí mít stejný počet prvků, jako mají řádkové vektory matice (tedy jako má matice sloupců)). Potom se vektor přičte ke všem odpovídajícím vektorům matice.

Násobení matic. Násobení matic se provádí hvězdičkou (∗). I zde je potřeba dbát na zachování správného počtu řádků a sloupců. Při násobení dvou matic (libovolná může být vektor) `C=A*B` musí mít matice $\boldsymbol{A}_{m\times n}$ m řádků a n sloupců a matice $\boldsymbol{B}_{n\times r}$ n řádků a r sloupců. Jinak řečeno, první matice musí mít právě tolik sloupců, co druhá řádků. Výsledná matice $\boldsymbol{C}_{m\times r}$ má pak m řádků a r sloupců. To tedy také mimo jiné znamená, že při násobení matic záleží na pořadí při násobení. Jakoukoli matici lze násobit číslem.

Dělení matic. MATLAB má dva znaky pro dělení: normální lomítko (/) a obrácené lomítko (\). Protože je výsledek získán přímo bez výpočtu inverze, je možné dělit i nečtvercové matice. Platí, že `B=A\C` je řešením `C=A*B`, zatímco `B=A/C` je řešením `C=B*A`. I zde je třeba dbát, aby matice měly správné rozměry.

Umocňování. Umocňování se provádí znakem stříšky (^). Například

```
16 C=A^3
```

Pozor, jde ovšem o maticové umocnění, které je možné jinak zapsat jako `C=A*A*A`, proto i zde záleží na velikosti matic; obecně lze umocňovat pouze *čtvercové* matice.

Operace po prvcích. Předchozí operace byly zřetelně maticového charakteru. Někdy je však potřeba provádět jednotlivé operace „po prvcích“. „Po prvcích“ zde znamená, že se s jednotlivými maticemi nepracuje jako s celky, ale pouze s jejich jednotlivými prvky. Například násobení matic po prvcích znamená, že se vzájemně vynásobí odpovídající prvky obou matic a výsledky uloží do odpovídajících prvků matice výsledku. V tomto případě musejí mít obě matice *stejné* rozměry. Podobně umocňování matice po prvcích znamená, že je umocněn každý prvek matice zvlášť. Matice teď už nemusí být čtvercová.

Mějme tyto matice

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}.$$

Matice $\boldsymbol{X}$ je maticovým násobkem $\boldsymbol{A}$ a $\boldsymbol{B}$ a matice $\boldsymbol{Y}$ jejich součinem po prvcích:

$$\boldsymbol{X} = \begin{pmatrix} 14 & 32 & 50 \\ 32 & 77 & 122 \\ 50 & 122 & 194 \end{pmatrix}, \boldsymbol{Y} = \begin{pmatrix} 1 & 8 & 21 \\ 8 & 25 & 48 \\ 21 & 48 & 81 \end{pmatrix}.$$

Nyní je snad zřejmé, o co jde.

Operace po prvcích se zapisují tak, že se před běžný operátor napíše tečka. Místo symbolů `*`, `/`, `\` a `^` máme tedy nyní `.*`, `./`, `.\` a `.^`.

Práce s proměnnými

K tomu, abychom zjistili, jaké proměnné jsme si zavedli, slouží dva příkazy: `who` a `whos`. První vypíše prostý seznam definovaných proměnných, druhý vypíše také jejich rozměry, počet jejich prvků a velikost paměti, kterou zabírají.

K odstranění proměnných z paměti slouží příkaz `clear`, za který napíšeme jméno proměnné nebo proměnných, které chceme odstranit: `clear A B C`. Pokud chceme odstranit z paměti počítače všechny definované proměnné, použijeme příkaz `clear all` (proto není moudré pojmenovat si nějakou proměnnou `all`).

Abychom zjistili velikost matice, použijeme příkaz `size(promenna)`. Ten nám vrátí vektor o dvou složkách – první složka určuje počet řádků matice a druhý počet sloupců. Pro vektory je navíc možné použít příkaz `length(promenna)`, který vrací délku vektoru (ať už řádkového nebo sloupcového). Pokud použijeme `length` na matici, vrátí nám větší číslo z počtu řádků a počtu sloupců.

Když máme vytvořenou nějakou matici, můžeme ji uložit na disk do souboru typu „`.mat`“ pomocí příkazu `save`. Zpátky do paměti je možné takovou matici načíst příkazem `load`.

Speciální proměnné

Jak už bylo řečeno, svoji proměnnou si můžete v MATLABu pojmenovat, jak chcete. Výjimku tvoří *klíčová slova* MATLABu, příkazy a speciální proměnné. Nikdy byste se neměli snažit pojmenovat si vlastní proměnnou stejně, jako se jmenuje speciální proměnná.

Speciální proměnné obsahují některé důležité konstanty (Ludolfovo číslo π, Eulerovo číslo e, imaginární část komplexních čísel i) a speciální hodnoty, jako je např. hodnota pro obecné výsledky `Ans`. Mezi nejdůležitější konstanty patří `pi`, `e`, `i`, `I`, `j`, `J`, `Ans` a `ans`.

A.2 Matematické a statistické funkce

Programovací jazyk \mat u zahrnuje velké množství funcí. Každá funkce může mít několik argumentů (vstupů) a vracet několik hodnot (výstupů). Vstupy i výstupy mohou být čísla, vektory i matice. V případě vektorů a matic se může funkce uplatňovat buď na každý prvek zvlášť (např. funkce `sin`), na celou matici dohromady (např. funkce `regress`) nebo postupně na jednotlivé prvky s přihlédnutím k předchozí hodnotě (např. funkce `cumsum`). Zde až na několik výjimek uvádíme jen názvy funkcí – podrobný popis získáte pomocí příkazu `help`, např.:

```
17 help sin
```

Mezi funkce, které se aplikují na každý prvek matice patří především:

příkaz	**význam**
`abs`	absolutní hodnota
`acos`	arkus kosinus
`asin`	arkus sinus
`atan`	arkus tangens
`cos`	kosinus
`exp`	exponenciální funkce e^x
`fix`	zaokrouhlení na celé číslo bližší k nule
`gcd`	největší společný dělitel
`lcm`	nejmenší společný násobek
`log`	přirozený logaritmus
`log10`	dekadický logaritmus
`rem`	zbytek po celočíselném dělení
`round`	zaokrouhlení k nejbližšímu celému číslu
`sin`	sinus
`sqrt`	druhá odmocnina
`tan`	tangens

Další funkce se aplikují na matici jako celek:

příkaz	význam
`cumprod`	kumulativní součin prvků
`cumsum`	kumulativní součet prvků
`diag`	diagonála matice
`max`	největší prvek vektoru
`mean`	průměr vektoru
`median`	medián vektoru
`min`	nejmenší prvek vektoru
`prod`	součin prvků vektoru
`sort`	setřídění prvků vektoru
`std`	směrodatná odchylka vektoru
`sum`	součet prvků vektoru

K dalším významným funkcím patří statistické funkce pro výpočet kovariance a korelace.

příkaz	význam
`corrcoef`	matice korelačních koeficientů
`cov`	kovarianční matice

Je nutno poznamenat, že funkce `corrcoef` vrací matici korelačních koeficentů dvou vektorů $\boldsymbol{x}$ a $\boldsymbol{y}$ ve tvaru

$$\begin{pmatrix} 1 & \rho_{y,x} \\ \rho_{x,y} & 1 \end{pmatrix},$$

kde $\rho_{x,y} = \rho_{y,x}$ je korelační koeficient obou vektorů, zatímco číslo 1 je korelační koeficient vektoru $\boldsymbol{x}$ sám se sebou a vektoru $\boldsymbol{y}$ také se sebou samým.

Další funkcí, která je důležitá v rámci tohoto předmětu je funkce `regress`. Tato funkce odhaduje koeficienty lineárního regresního modelu. Použití této funkce je následující:

```
18 b=regress(y,X)
```

nebo

```
19 [b,bint,e,eint,Stat]=regress(y,X,alpha)
```

Argumenty funkce jsou vektor vysvětlované časové řady `y`, matice plánu `X` a nepovinná hladina významnosti `alpha` (pokud není zadána, uvažuje MATLAB implicitně $\alpha = 0.05$). Funkce vrací vektor odhadu parametrů `b`, interval spolehlivosti parametrů `bint` na hladině významnosti α, vektor reziduí `e`, interval spolehlivosti reziduí `eint` na hladině α a vektor statistik `Stat`. První hodnota vektoru `Stat` je koeficient determinace R^2.

A.3 Scripty a funkce

Zatím jsme uvažovali, že všechny příkazy píšeme na klávesnici. Většinou je však výhodnější psát jednotlivé příkazy „jako program“ a znovu a znovuje spouštět. To umožňuje postupně program odladit, tedy zbavit chyb (není nutné psát při každé chybě vše znova od začátku, stačí opravit chybný řádek a vše spustit znovu). Programům se v MATLABu říká *skripty*. Je také možné vytvořit si svou vlastní funkci.

Skripty

Script je normální textový soubor napsaný v běžném textovém editoru (například notepadu). Jméno tohoto souboru musí končit koncovkou „`.m`“. Obsahem tohoto souboru jsou pak stejné příkazy, jaké jsme dosud psali přímo na klávesnici. Výhodou scriptů je, že je můžeme spouštět opětovně.

Skripty také mohou zavádět nové globální proměnné – je tedy možné si do scriptu zapsat matice, které budeme častěji používat pro svoje výpočty (např. časové řady makroekonomických ukazatelů) a vždy při spuštění MATLABu tato data načíst spuštěním tohoto skriptu.

Skripty se spouští tak, že napíšete jméno skriptu bez koncovky. Předpokládejme, že máme skript `umoruj.m`, potom ho spustíme tak, že napíšeme `umoruj`.

Funkce

MATLAB nám umožňuje vytvářet své vlastní funkce. Jejich vstupními parametry i výstupními hodnotami jsou opět matice. Fyzicky je nová funkce (stejně jako skript) textový soubor na disku počítače, který končí koncovkou „`.m`“. Jméno tohoto souboru musí být stejné jako jméno funkce. Také obsah tohoto souboru je normalizován. V nejjednodušším případě vypadá například takto:

```
20 function s=sedminasobek(x)
21 %Tato funkce vraci sedminasobek zadane hodnoty
22 %Syntaxe sedminasobek(x)
23 s=7*x;
24 end
```

Prvním řádkem MATLABu říkáme několik věcí: zaprvé, že se jedná o funkci (klíčové slovo `function`), která se jmenuje `sedminasobek`; dále že tato funkce má jeden vstupní parametr `x` a vrací jednu výstupní matici `s`.

Další dva řádky jsou „zakomentářované“ – MATLAB normálně řádky začínající znakem procento (%) ignoruje .Na tomto místě však má tento komentář zvláštní význam – je použit pro nápovědu. Pokud napíšete `help sedminasobek`, MATLAB odpoví:

```
Tato funkce vraci sedminasobek zadane hodnoty

Syntaxe sedminasobek(x)
```

Funkce končí příkazem `end`. To, co je mezi prvním řádkem a jemu odpovídajícím `end` je „tělo“ funkce – příkazy, které se vyhodnotí při každém spuštění funkce.

Vlastní funkce se spouští jako každá jiná (ve skutečnosti není žádný rozdíl mezi funkcemi, které píšete vy a funkcemi, které jsou součástní MATLABových knihovem): napíšete její jméno do výrazu, např.

```
25 A=sedminasobek(65);
```

MATLAB funkci vyhodnotí a výsledek (455) uloží do proměnné `A`.

Pokud v rámci funkce nadefinujete nějaké proměnné, potom jsou tyto proměnné chápány jako *lokální*, tj. neexistují mimo tuto funkci. To umožňuje rekurzivní volání funkce, kdy funkce volá sama sebe. Například:

```
26 function f=fibfun(n)
27 %Funkce fibfun pocita tzv. Fibonacciho cislo
```

```
28 if (n>2)
29   f=fibfun(n-1)+fibfun(n-2);
30 else
31   f=1;
32 end
```

Funkce může mít pochopitelně více vstupních parametrů a více výstupních hodnot. Syntaxe je potom následující:

```
33 function [A,B,C]=funkce(a,b,c,d)
```

To definuje funkci `funkce`, která má čtyři vstupní parametry a tři výstupní hodnoty. (Každé z těchto výstupních hodnot musí být přiřazena hodnota!) Funkce se potom použije takto:

```
34 [X,Y,Z]=funkce(x,y,z,57);
```

Je možné napsat na výstupu méně výsupních proměnných, než předpokládá funkce (např. jen `[X,Y]`) – přebývající hodnoty budou „zahozeny“.

Řídící struktury

MATLAB obsahuje řídící struktury, které jsou obvyklé ve většině programovacích jazyků: je to podmínka a cykly.

Cyklus FOR. Cyklus `for` se používá v případě, že potřebujeme nějaké operace několikrát opakovat, přičemž známe žádoucí počet těchto opakování.

Například pokud chceme vytvořit vektor, který ná na každé čtvrté pozici jedničku a jinak samé nuly, můžeme to udělat právě pomocí cyklu `for`:

```
35 for k=1:4:50
36   A(k)=1;
37 end
```

Na řádku 35 říkáme, že se má něco opakovat tak dlouho, dokud hodnota čítače `k`, která je na počátku 1, nedosáne hodnoty větší nebo rovné 50, přičemž se při každém opakování hodnota `k` zvýší o 4. Cyklus začíná příkazem `for` a končí příkazem `end`. Opakují se tedy ty příkazy, které leží mezi tím (v našem případě pouze řádek 36).

Pokud je *krok* cyklu (tj. hodnota, o kterou se v každém průběhu zvýší čítač `k`, v našem případě číslo 4) rovné 1, potom může být vynecháno. Krok cyklu může být i záporný. Pozor na cykly, které nikdy nekončí (*deadlock*).

Cyklus WHILE. Cyklus WHILE se používá v případě, že něco potřebujeme opakovat „neznámý počet krát“. Cyklus se opakuje tak dlouho, dokud je splněná jeho podmínka. Například:

```
38 n=50;
39 while (n>5)
40   n=n-1
41 end
```

Cyklus se bude opakovat tak dlouho, dokud bude hodnota `n` větší než 5.

V podmínce je možné používat tyto relační operátory:

operátor	význam
<	menší než
<=	menší nebo rovno než
==	rovno
>=	větší nebo rovno než
>	větší než
~=	nerovno

Dále je možné používat tyto logické operátory:

operátor	význam
&	logický součin („a zároveň“)
|	logický součet („nebo“)
~	negace („ne“)

Potom můžeme vytvářet i realtivně složité podmínkyvytvářet libovolným kombinováním jednotlivých výrazů a relačních a logických operátorů. Například:

```
42 while ((n<50) & (m<100)) | (j=1000)
```

Tato podmínka je splněná, pokud se `j` rovná 1000 nebo `n` je menší než 50 a zároveň je `m` menší než 100.

Podmínky. Podmínky se používají k „větvení“ programu, když je potřeba, aby se část výpočtu provedla pouze při splnění určité podmínky. Tato podmínka se formuluje stejně jako u cyklu `while`. Například:

```
43 if (n<0)
44   n=-n;
45 end
```

Tělo (tady pouze řádek 44) se provede pouze, když je splněná podmínka.

Kromě možnosti přeskočit část programu, pokud není podmínka splněná, umožňují podmínky program také „větvit“, tj. mít dvě různé varianty kódu podle toho, zda je nebo není podmínka splněna.

```
46 if (n<0)
47   n=-n; %Prikaz 1
48 else
49   n=0; %Prikaz 2
50 end
```

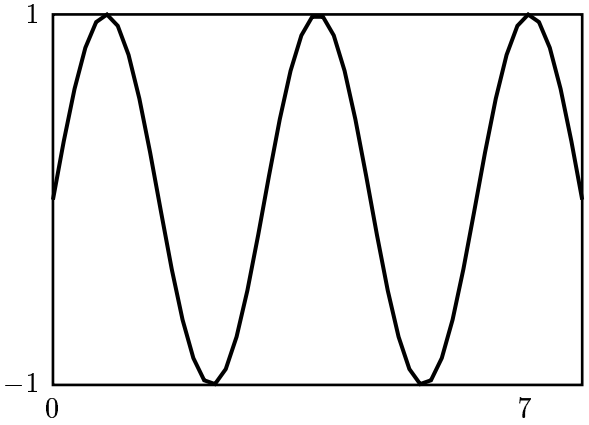
Při splnění podmínky se provede příkaz 1 (tady řádek 47), v opačném případě příkaz 2 (v našem případě řádek 49).

Podmínky je možné i vkládat do sebe.

A.4 Grafy

MATLAB umožňuje kreslit grafy. Grafy je možné malovat do více oken. Nejprve je třeba přepnout se do okna, do kterého budeme malovat (implicitně je to okno číslo 1). To se dělá příkazem `figure(`*číslo okna*`)`. Pokud už grafické okno něco obsahuje, je možné ho smazat příkazem `clf`.

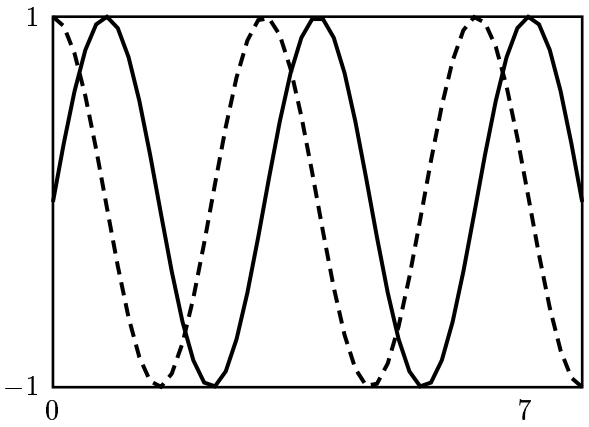
Vlastní kreslení se provádí příkazem `plot`, jehož parametry jsou dva vektory – popis x-ové a y-ové osy. Příkladem nám může být obrázek A.1, nakreslený následujícím programem:



Obrázek A.1: Základní příkazy pro kreslení

```
51 x=0:0.1:5*pi;
52 s=sin(x);
53 figure(1)
54 plot(x,s)
```

V případě, že chceme naráz vykreslit do jednoho okna více grafů, můžeme dvojice vektorů `x` a `y` v příkazu `plot` libovolně opakovat. Obrázek A.2 vznikl pomocí následujícího



Obrázek A.2: Více čar pomocí jednoho příkazu `plot`

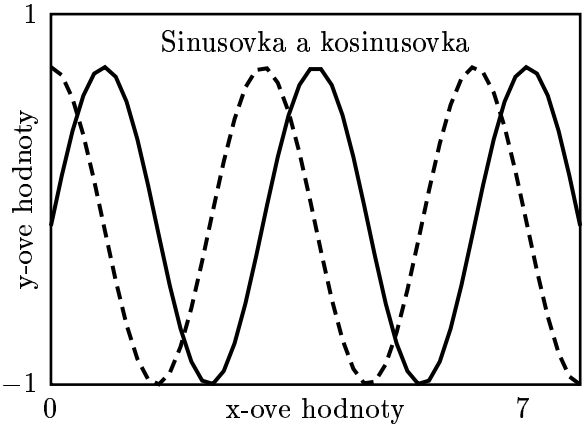
programu (na obrazovce počítače jsou normálně čáry odlišeny barevně; zde použijeme odlišení pomocí různých typů čar).

```
55 c=cos(x);
56 plot(x,s,x,c)
```

Každý další příkaz `plot` maže obsah okna – tomu je možno zabránit tak, že zadáme příkaz `hold on` (do původního stavu zapneme opět `hold off`). V tomto případě můžeme kreslit více křivek přes sebe pomocí jednotlivých příkazů `plot`.

Okno je dále možno popsat: příkaz `title` vytváří titulek, příkaz `xlabel` popis x-ové osy a příkaz `ylabel` popis y-ové osy. Tyto popisy je ovšem možné dělat až po vykreslení

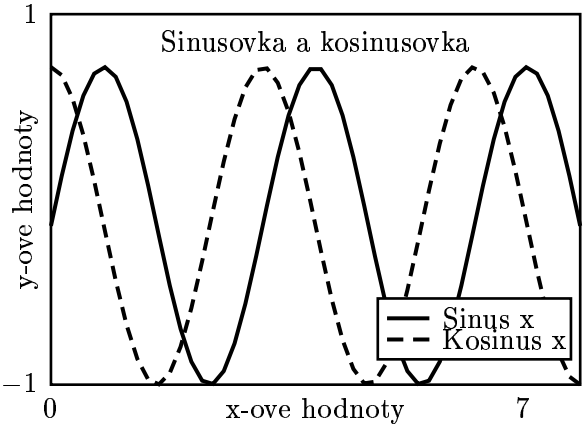
grafu, jinak je příkaz `plot` okamžitě smaže. Argumenty všech tří příkazů (vlastně funkcí) jsou textové řetězce uzavřené v pravých apostrofech. Obrázek A.3 byl nakreslen tímto programem:



Obrázek A.3: Popis obrázku

```
57 plot(x,s,x,c)
58 title('Sinusovka a kosinusovka')
59 xlabel('x-ove hodnoty')
60 ylabel('y-ove hodnoty')
```

Dále je možné příkazem `legend` vytvořit *legendu*, tedy popis jednotlivých čar. Do apostrofů sem napíšeme popisy jednotlivých čar, oddělené čárkou, v tom pořadí, v jakém jsme je pomocí příkazu plot kreslili. Opět tedy platí, že příkaz `legend` lze použít až po vykreslení všech čar, tj. po posledním použití příkazu `plot`. Použití příkazu legend ilustruje obrázekA.4..



Obrázek A.4: Popis jednotlivých čar (příkaz `legend`)

```
61 legend('Sinus x','Kosinus x')
```

V případě, že nejsme spokojeni automatickým nastavení os a výřezu grafu, můžeme je nastavit explicitně příkazem `axis`. Parametrem toho příkazu je řádkový vektor, který obsahuje čtyři komponenty: minimum na ose x, maximum na ose x, minimum na ose ya maximum na ose y. Tento příkaz je potřeba použít před příkazem plot. V našem případě se může hodit nastavit osy takto:

```
62 axis([0 5*pi -1.5 1.5])
```

MATLAB umožňuje také měnit styl čar, kreslených pomocí příkazu `plot`: v tom případě je nutné ke dvojici vektorů `x` a `y` přidat třetí složku, určující styl. Ta je textová – je tedy uzavřena v apostrofech a obsahuje jeden nebo dva z následujících znaků, které určují barvy a styl čáry:

symbol	**barva**	**symbol**	**styl čáry**
y	žlutá	.	jednotlivé body
m	fialová	o	jednotlivé kroužky
c	tyrkysová	x	jednotlivé značky x
r	červená	+	jednotlivá znamínka plus
g	zelená	*	jednotlivé hvězdičky
b	modrá	-	plná čára
w	bílí	:	tečkovaná čára
k	černá	-.	čerchovaná čára
		–	čárkovaná čára

Existuje také příkaz `subplot`, který umožňuje umístit více grafů do jednoho okna.

A.5 Další užitečné příkazy

Mezi další užitečné příkazy patří příkazy pro práci s diskem: příkaz `cd` nastavuje aktuální adresář, příkaz `pwd` vypíše cestu k aktuálnímu adresáři a příkaz `dir` obsah aktuálního adresáře. Tyto příkazy se používají stejně, jako jejich DOSovské nebo UNIXové ekvivalenty.

Pokud chcete použít přímo nějaký DOSový program, stačí když jeho jméno s patřičnou cestou uvedete vykřičníkem:

```
63 !notepad bum.m
```

Pokud chceme zajistit, aby se náš program na některém místě zastavil, můžeme použít příkaz `pause`.

Pro výpis textů na obrazovku se používá příkaz `disp`; pro přesměrování výstupu na obrazovku do souboru příkaz `diary`.

Ukončení MATLABu se provádí buď příkazem `quit` nebo `exit` – fungují oba.

Velice užitečný je také příkaz `help`, který poskytuje velice kvalitní nápovědu. Používá se `help` *jméno příkazu*, např.

```
64 help help
```

A to je vše, přátelé, tádydádydádydá!