OCTAVE

Rozšírený úvod

Ján Buša

Táto publikácia vznikla s prispením grantovej agentúry SR KEGA v tematickej oblasti „Nové technológie vo výučbe“– projekt: 3/2158/04 – „Využitie OPENSOURCE softvéru vo výučbe na vysokých školách“.

Recenzovali: Michal Kaukič
Miloš Šrámek

ISBN 80-8073-596-4

Sadzba programom pdfTEX

Obsah

Úvod 8

1 Prvé kroky 11
1.1 Inštalácia Octave 11
1.1.1 Inštalácia Octave v operačnom systéme Windows . . 11
1.1.2 Inštalácia Octave v Linuxe 14
1.2 Skúška funkčnosti Octave 14

2 Dátové typy 24
2.1 Čísla a ich zobrazovanie 24

3 Matice a lineárna algebra 28
3.1 Matice 28
3.1.1 Komplexné čísla 32
3.2 Maticové operácie a funkcie 33
3.2.1 Súčet, rozdiel, súčin a transponovanie matíc 33
3.2.2 Zložkové operácie 36
3.2.3 Základné maticové funkcie 37
3.3 Riešenie sústav lineárnych algebrických rovníc 41
3.3.1 Frobeniusova veta, inverzná a pseudoinverzná matica 42
3.3.2 Porovnanie presnosti MATLABu a Octave v prípade zle podmienenej SLAR 48
3.3.3 Riedke matice 50

3.4 Vlastné čísla, vlastné vektory a singulárny rozklad matíc . . 51
3.4.1 Vlastné čísla a vlastné vektory matíc 51
3.4.2 Singulárny rozklad matíc 54
3.4.3 Použitie singulárneho rozkladu pri riešení SLAR . . 57

4 Reťazce **63**
4.1 Zadávanie reťazcov 63
4.2 Funkcie orientované na reťazce 66
4.2.1 Vytváranie reťazcov 66
4.2.2 Vyhľadávanie a výmena podreťazcov 69
4.2.3 Konverzie reťazcov 72
4.2.4 Znakové funkcie 72

5 Dátové štruktúry, zoznamy a bunkové polia **73**
5.1 Štruktúry 73
5.2 Zoznamy 77
5.3 Bunkové polia 80

6 Výrazy a operátory **82**
6.1 Výrazy 82
6.1.1 Aritmetické výrazy 82
6.1.2 Logické výrazy a funkcie 84
6.2 Operátory 88
6.2.1 Logický operátor `if` 89
6.2.2 Operátor vetvenia `switch` 91

6.2.3 Operátor cyklu `for` 91
6.2.4 Operátor cyklu `while` 93
6.2.5 Operátor cyklu `do-until` 94
6.2.6 Príkazy `break` a `continue` 94

7 Scenáre a funkcie **96**
7.1 Scenáre 96
7.2 Funkcie 99
7.2.1 Systémové funkcie 100
7.2.2 Funkcie definované užívateľom 100
7.2.3 Dynamicky pripájané funkcie 107
7.3 Rozmiestnenie funkcií a scenárov v distribúcii Octave 107

8 Vstup a výstup údajov **110**
8.1 Vstup údajov 110
8.1.1 Príkazy `input` a `menu` 110
8.1.2 Načítanie údajov zo súboru príkazom `load` 111
8.1.3 Načítanie formátovaných údajov zo súboru 113
8.2 Zápis údajov do súborov 114
8.2.1 Zápis údajov do súboru príkazom `save` 114
8.2.2 Zápis formátovaných údajov do súboru a na obrazovku 115

9 Grafický výstup Octave **118**
9.1 Príkazy `plot` a `gplot` 118
9.2 Použitie polárnych súradníc 122

9.3 Tvorba histogramov príkazom `hist` . . . 124
9.3.1 Príkazy `bar`, `stairs`, `loglog`, `semilogx` a `semilogy` . . 125
9.4 Zobrazenie kriviek v trojrozmernom priestore . . . 125
9.5 Zobrazovanie grafov funkcií dvoch premenných . . . 127
9.5.1 Vstevnicové grafy . . . 128
9.5.2 Grafy funkcií dvoch premenných v trojrozmernom priestore . . . 129
9.6 Uloženie viacerých obrázkov vedľa seba . . . 131
9.7 Záver . . . 133

10 Vybrané aplikácie **134**
10.1 Štatistické funkcie . . . 134
10.1.1 Základné štatistické funkcie . . . 135
10.1.2 Štatistické funkcie rôznych rozdelení pravdepodobnosti . . . 137
10.1.3 Testovanie štatistických hypotéz . . . 139
10.2 Práca s polynómami . . . 142
10.2.1 Riešenie algebrických rovníc . . . 143
10.2.2 Rozklad racionálnej funkcie na súčet parciálnych zlomkov . . . 144
10.2.3 Aproximácia polynómami v zmysle najmenších štvorcov . . . 145
10.2.4 Splajny a funkcie po častiach kubické . . . 146
10.3 Riešenie sústav nelineárnych rovníc . . . 148
10.4 Záver . . . 151

P Prílohy **152**
P.1 Zoznam funkcií 152
P.2 Zoznam systémových premenných 155
P.3 História príkazov 163

Použitá literatúra **165**

Úvod

Táto skromná učebnica oboznamuje čitateľa s programom Octave, jeho inštaláciou a použitím. Žiaľ, hoci príručka nie je taká stručná, ako som pôvodne zamýšľal, viaceré témy ostali nedopovedané, mnohé funkcie neboli spomenuté. Podrobný popis práce programu je možné nájsť tiež v aktuálnej verzii manuálu Octave (EATON, 1997), ktorého online verzia sa nachádza na stránke http://www.network-theory.co.uk/docs/octave/.

Program Octave bol pôvodne navrhnutý ako softvér k stredoškolskej učebnici J. B. Rowlingsa a J. G. Ekerdta, neskôr sa z neho vyvinul systém na jednoduché riešenie reálnych úloh. Autor programu J. W. Eaton nazval program podľa svojho učiteľa. Počas viac ako desiatich rokov prispel k zlepšeniu programu celý rad ľudí, uvedených v Eatonovej knihe (1997).

V roku 1984 firma The MathWorks, Inc. vydala prvú verziu programu MATLAB. Za 22 rokov sa z MATLABu a z programu SIMULINK vyvinul rozsiahly systém, v súčasnosti je aktuálna verzia MATLAB 2006a. MATLAB doplnený o množstvo aplikačných balíkov umožňuje jednoduché a rýchle riešenie mnohých úloh, aj keď v interaktívnom režime sa nehodí na riešenie veľkých úloh. Je možné ho odporučiť aj pri výučbe viacerých predmetov, napríklad lineárnej algebry, matematickej analýzy, numerických metód, matematickej štatistiky, operačnej analýzy, teórie riadenia, atď. Nevýhodou komerčného MATLABu je jeho vysoká cena (odpovedajúca však jeho kvalite), dokonca aj využívanie oficiálnych multilicencií s ohraničeným počtom inštalácií na školách sa deje na hranici zákonnosti.

Máloktorý absolvent univerzity teda bude mať záujem o zakúpenie tohoto softvéru, či už počas štúdia alebo po opustení školy. Program Octave je jednou z možností, ako zvládnuť základy MATLABu a zároveň si udržať finančne veľmi prístupnú možnosť legálnej práce s podobným nástrojom, a to rovnako pre samotné vysoké alebo stredné školy, ako aj pre študentov doma na vlastnom počítači.

Program Octave patrí medzi tzv. programy s otvoreným zdrojovým kódom (budeme ich označovať OPENSOURCE).[1] Je možné nielen legálne (v tomto prípade zadarmo) získať jeho skompilované verzie pre rôzne operačné systémy, ale aj si vytvoriť vlastnú kompiláciu programu z dostupných zdrojových súborov. Jeho syntax je prakticky zhodná so syntaxou MATLABu. Pracuje sa v interaktívnom režime, ale je tiež možné spúšťať vykonávateľné súbory – scenáre. Na internete je možné nájsť množstvo takýchto „programov", ktoré pre MATLAB aj Octave vytvorili mnohí autori a poskytli ich verejnosti. Rovnako ako MATLAB je možné použiť Octave pri výučbe viacerých predmetov, vrátane základov programovania. Spolu s ďalším OPENSOURCE softvérom je možné naučiť sa riešiť široký okruh úloh a najmä s ním legálne pracovať aj po skončení štúdia.[2]

Podobné vlastnosti ako Octave má ďalší OPENSOURCE systém Pylab, ktorý využíva programovací jazyk Python a je do veľkej miery kompatibilný so syntaxou MATLABu. Svojou objektovou orientáciou však predčí

[1] Viac sa dozviete na http://skosi.org.

[2] Elektronické verzie tejto a ďalších príručiek vytvorených v rámci projektu KEGA sú prístupné v priečinkoch na adrese http://people.tuke.sk/jan.busa/kega.

Octave aj MATLAB. Jeho základný popis nájdete v učebnici (KAUKIČ, 2006).

Na tomto mieste by som sa rád poďakoval recenzentom Michalovi Kaukičovi (pri štúdiu jeho knihy (KAUKIČ, 1998) som sa prvý raz dozvedel o existencii programu Octave) a Milošovi Šrámkovi za pozorné prečítanie príručky, ale aj za ostatnú činnosť, ktorou prispievajú k propagácii otvoreného softvéru a jeho využívania na vysokých i stredných školách. Poďakovanie patrí aj Ladislavovi Ševčovičovi za mnohé cenné rady a všestrannú pomoc.

Košice 30. júna 2006 Ján Buša

1. Prvé kroky

Pre čitateľa bude najefektívnejšie overovať a osvojovať si popísanú činnosť Octave a jeho vlastnosti priamo pri práci s programom. Preto na úvod stručne popíšeme postup získania programu, jeho inštalácie a vyskúšame jeho funkčnosť na jednoduchom príklade. V ďalších kapitolách popíšeme jednotlivé súčasti systému Octave. V prílohách uvádzame zoznamy funkcií, operácií a viacerých systémových premenných. Ich presnejší popis nájdete v podrobnom manuále (EATON, 1997), ktorý je dostupný na stránke http://www.gnu.org/software/octave/docs.html. Jeho český variant (JUST, 2006) nájdete na stránke http://octave.wz.cz/index.html.

1.1. Inštalácia Octave

1.1.1. Inštalácia Octave v operačnom systéme Windows

V tomto oddiele popíšeme inštaláciu programu Octave v OS Windows (pozri tiež (JUST, 2006)), inštaláciu v Linuxe stručne popíšeme v ďalšom oddiele. Program je možné stiahnuť z viacerých stránok. My sme použili stránku http://www.math.sfu.ca/~cbm/ta/octave.html, kde okrem samotného inštalačného súboru `octave-2.1.50a-inst.exe`, nájdete aj krátky popis inštalácie.[3]

[3] Novšiu verziu Octave nájdete na stránke
http://prdownloads.sourceforge.net/octave/octave-2.1.73-1-inst.exe?download

Po spustení inštalačného súboru najprv zvolíme jazyk inštalácie (podrobnejší popis nájdete v (JUST, 2006)) English a potvrdíme súhlas s licenčnými podmienkami voľbou I Agree. Po odkliknutí Next, voľbe adresára programu nasledovanej kliknutím na Install nám inštalačný program ponúkne v samostatnom okne zadanie editora

```
STEP 3

Set Editor used by the "edit" command
The default is currently "notepad"
If you wish to use a different editor, enter the path and
filename here

Editor (currently notepad)>
```

V OS Windows odporúčame ponechať editor `notepad`, príkaz `edit` radšej nepoužívať a editovať `m`-súbory vo svojom obľúbenom editore. Cestu a názov editora je potrebné zadať v úvodzovkách, s použitím lomiek /.

Následne inštalátor požiada o potvrdenie alebo zadanie cesty k programu `ghostscript`, napríklad:

```
STEP 4
```

alebo na stránke http://www.octave.org. Je o niečo kvalitnejšia ako popisovaná verzia a navyše jej inštalácia prebieha jednoduchšie.

```
In order to use the epstk graphics functions, you
need to have a postscript interpreter. This would
typically be gswin or ghostscript

The program associated with ps files is C:\gs\gsview32.exe
To accept this program type return, otherwise,
enter the full path of the desired ps viewer program
New PS viewer>
```

alebo v prípade chýbajúceho programu `ghostscript`[4]

```
STEP 4

In order to use the epstk graphics functions, you
need to have a postscript interpreter. This would
typically be gswin or ghostscript

cmd: not found
No program is currently associated with ps files
If you have such a program installed, enter the
full path to the executable, otherwise type return
```

[4] Ak nemáte nainštalované programy `ghostscript` a `ghostview`, odporúčame Vám stiahnuť si ich. Možnosť pracovať s PostScriptom sa Vám iste v budúcnosti zíde. Ak ste už nainštalovali Octave, odinštalujte ho, nainštalujte `ghostscript` a `ghostview`, potom znova Octave.

```
New PS viewer>
```

1.1.2. Inštalácia Octave v Linuxe

Väčšina distribúcií Linuxu obsahuje inštalačné balíky Octave ako svoju štandardnú súčasť. Napríklad v distribúciách založených na Debiane (Debian, Ubuntu, Kubuntu) stačí ako správca systému pustiť príkaz:

```
apt-get install octave
```

alebo jeho ekvivalent v niektorom grafickom inštalačnom programe. Pre distrubúciu Mandriva treba použiť príkaz `urpmi octave` a pre Fedoru/RedHat `yum install octave`. Ďalšie možnosti pre inštaláciu sú uvedené na stránke http://www.gnu.org/software/octave/download.html.

1.2. Skúška funkčnosti Octave

Po úspešnej inštalácii môžete spustiť Octave kliknutím na ikonku programu na pracovnej ploche alebo príkazom `octave` v Linuxe.

Vo Windows sa po spustení na obrazovke objaví okno programu Octave:

Pracovným adresárom programu Octave je adresár `/octave_files`. Pracovný adresár nastavíme editovaním súboru `start_octave.sh`, ktorý sa nachádza v adresári `/bin`.[5] Tu môžeme nastaviť aj editor, prípadne implicitné správanie programu. Riadok

```
export HOME="/octave_files"
```

nahradíme, napríklad, riadkom

```
export HOME="d:/Users/Busa/Octave"
```

[5] Vo verzii Octave 2.1.73 je to súbor `octave.sh` v koreňovom adresári.

a môžete začať pracovať. Po ukončení každého príkazu sa objaví „výzva“ (prompt) v tvare >>. Zadaním príkazu

```
>> PS1="skuska:\\#> "
```

zmeníme výzvu na `skuska:n>` , kde `n` označuje poradové číslo príkazu aktuálneho spustenia programu Octave.

V Linuxe sa Octave spúšťa v okne terminálu príkazom `octave`, ktorý sa priamo ohlási výzvou

```
octave:1>
```

Základné konfiguračné súbory `octave` sú `/etc/octaveXX.conf` (systémový) a `/.octaverc` (používateľský – spočiatku neexistuje, v prípade potreby ho treba vytvoriť). Žiadne nastavenia však nie sú na úvod potrebné.

Vyskúšajte zadať nasledujúcu postupnosť príkazov a všímajte si, čo sa po ich potvrdení mení na obrazovke a v pracovnom adresári.[6]

[6] Pri zobrazovaní výpisov Octave vynechávame niektoré prázdne riadky a výpisy, presahujúce šírku riadku, zobrazujeme v dvoch riadkoch.

```
>> PS1="skuska:\\#> "
PS1 = skuska:\#>
skuska:2> x=0:0.01:2*pi;
skuska:3> plot(x,sin(x),x,cos(x));
skuska:4> axis([0 2*pi -1.2 1.2]);
skuska:5> replot
skuska:6> gset terminal postscript
skuska:7> gset output "sincos.ps"
skuska:8> replot
skuska:9>
```

Ak tieto príkazy zadáme v Linuxe, po (prvom) zadaní príkazu gset sa objavia varovania:

```
octave:5> gset terminal postscript
warning: gset is deprecated and will be removed from a future
warning: version of Octave.
warning: You should use the higher-level plot functions
warning: ("plot", "mesh", "semilogx", etc.) instead
warning: of the low-level plotting commands.
warning: If you absolutely must use this function, use the
warning: internal version __gnuplot_set__ instead.
```

Namiesto uvedených príkazov v riadkoch 6 a 7 teda v Linuxe[7] zadáme:

[7] Ďalej už budeme popisovať prácu programu vo Windows.

```
octave:5> __gnuplot_set__ terminal postscript
octave:6> __gnuplot_set__ output "sincos.ps"
```

Vidíme, že v priebehu niekoľkých minút sme schopní vykresliť jednoduché grafy funkcií a uložiť grafy do súboru na ďalšie použitie. Vysvetlíme zmysel jednotlivých príkazov, podrobnejšie sa im budeme venovať nižšie v ďalších kapitolách.

V prvom riadku je už vyššie uvedený príkaz na zmenu výzvy programu Octave. Keďže sme príkaz neukončili bodkočiarkou, v druhom riadku Octave vypísal výsledok priradenia. Všimnite si, že namiesto dvoch spätných lomiek, sa v reťazci premennej `PS1` objavila len jedna spätná lomka \. To je vlastnosť práce programu Octave s reťazcami, opíšeme ju nižšie.

`skuska:2> x=0:0.01:2*pi;` — premennej `x` je priradený vektor hodnôt od 0 do 2π s krokom 0,01. Posledná hodnota vektora `x` bude 6,28 – ďalšia hodnota 6,29 by už presiahla hornú hranicu 2π.

`skuska:3> plot(x,sin(x),x,cos(x));` — vykreslia sa grafy funkcií $\sin x$ a $\cos x$ na intervale $\langle 0, 6.28 \rangle$.[8] V tomto momente sa na obrazovke otvorilo ďalšie okno s nadpisom `gnuplot graph`, ktoré obsahuje samotný graf, vytvorený programom GNUPLOT. Vo Windows sa otvorí aj

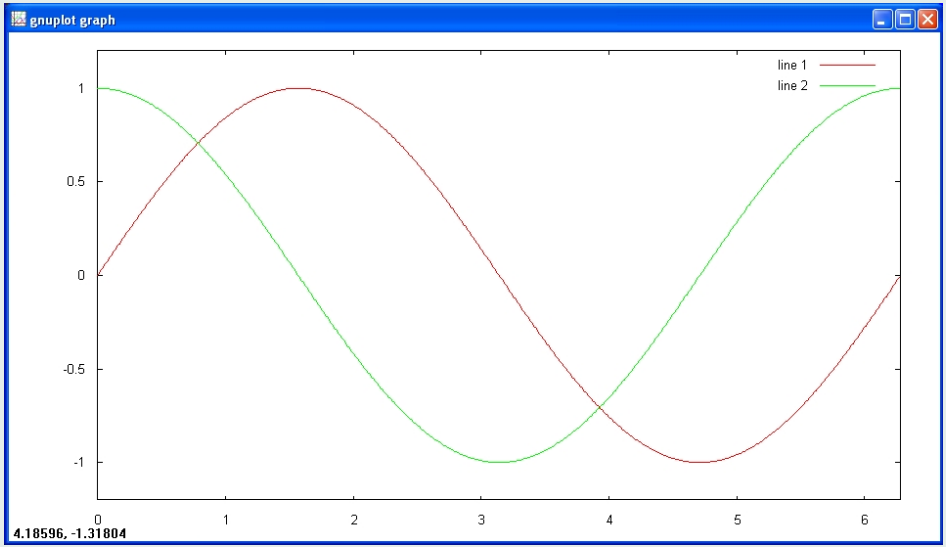
[8] Odteraz budeme namiesto desatinnej čiarky používať „desatinnú bodku“, čo je v súlade s formátom výpisu Octave aj MATLABu.

druhé okno s nadpisom `gnuplot`. Je to okno samotného programu GNUPLOT[9].

`skuska:4> axis([0 2*pi -1.2 1.2]);` — zmení sa rozsah hodnôt na osiach x a y. Takto dosiahneme, že sa grafy nebudú dotýkať hornej a dolnej hranice rámčeku.

`skuska:5> replot` — graf sa prekreslí v súlade s príkazom `axis` (pozrite obrázok nižšie).

[9]Program GNUPLOT umožňuje ďalšie manipulácie s vytvoreným grafom. V tejto príručke sa nebudeme programom GNUPLOT podrobne zaoberať, stručne je popísaný v príručke (DOBOŠ, 2006), podrobnejší popis nájdete na stránke http://learn.tsinghua.edu.cn:8080/2001315450/gnuplot-doc/TOC.html. Na stránke http://www.linuxsoft.cz/ sa popri množstve zaujímavých informácií nachádza aj český seriál o GNUPLOTe, ktorý napísal Vladimír Jarý – stránka http://www.linuxsoft.cz/article_list.php?id_kategory=214. Ďalšie užitočné informácie o GNUPLOTe sa nachádzajú na stránke http://gnuplot.info/.



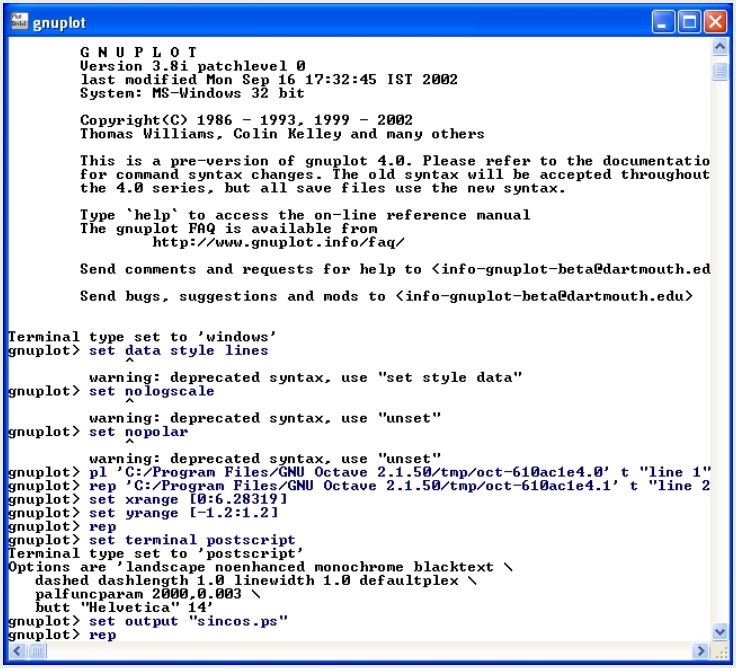
Výsledok sa prejaví v okne `gnuplot graph`, ako to vidíme na obrázku. Každá funkcia je zobrazená svojou farbou, čo je popísané aj v legende. Táto vlastnosť je užitočná, ak pracujeme s farebnými textovými dokumentami.

`skuska:6> gset terminal postscript` — namiesto výstupu na obrazovke sa nasledujúci graf „zobrazí“ do postscriptového súboru.[10]

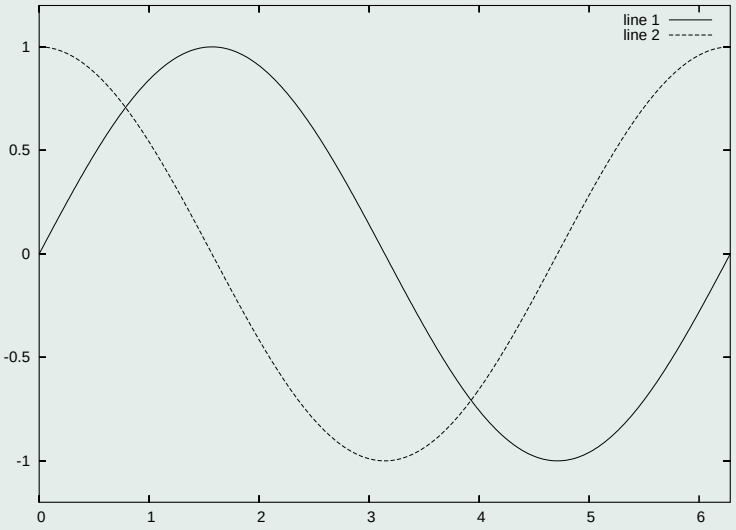
[10] Ešte lepšie by bolo použiť `gset terminal postscript eps`, na výstup priamo do Encapsulated PostScriptu.

`skuska:7> gset output "sincos.ps"` — zadanie názvu súboru, do ktorého sa uloží obrázok vo formáte postscript.

`skuska:8> replot` — „prekreslenie“ grafu do požadovaného súboru. V okne `gnuplot` sa vo Windows zapisujú príkazy, ktoré sme zadávali v Octave, avšak vo formáte GNUPLOTu (pozri nasledujúci obrázok).



Obrázok vo formáte PostScript je možné použiť ďalej.



Napríklad v tomto prípade sme ho prekonvertovali na formát Encapsulated PostScript (eps) a ďalej na formát Portable Document Format (pdf), ktorý dokážeme vložiť do textu. Výsledok je zobrazený na obrázku vyššie. Všimnite si, že program GNUPLOT pri zápise/konverzii do postscriptového súboru zmenil farebné čiary na čiernobiele, pričom druhý graf je znázornený prerušovanou čiarou, odpovedajúcim spôsobom sa zmenila aj legenda! Práve toto je užitočné pri čiernobielych textoch!

Vyššie uvedený príklad svedčí o tom, že po zvládnutí základov práce s programom Octave (a popritom s programom GNUPLOT :), získate efektívny nástroj na jednoduché riešenie náročných úloh.

Príkazy `gset terminal` respektíve `__gnuplot_set__ terminal` (v Linuxe) prepnú výstupné zariadenie. Ak chceme obrázok znova vidieť na obrazovke, musíme výstup prepnúť príkazmi `gset terminal windows` respektíve `__gnuplot_set__ terminal x11`.

Zoznam funkcií a premenných v prílohe P Vám môže pomôcť pri získavaní informácií, či už priamo alebo prostredníctvom príkazu `help niečo`[11]. Napríklad sa môžeme uistiť, že `pi` je premenná s hodnotou Ludolfovho čísla π:

```
skuska:9> help pi
pi is a built-in constant

- Built-in Variable: pi
  The ratio of the circumference of a circle to its diameter.
  Internally, 'pi' is computed as '4.0 * atan (1.0)'.
...
skuska:10>
```

Činnosť programu ukončíme príkazom `quit`.

Prejdime k popisu základných vlastností programu Octave.

[11] Pri dlhších výpisoch sa výpis v Linuxe po stránke zastaví a zobrazí sa dvojbodka. Stlačením klávesu ENTER výpis pokračuje. Výpis ukončíme stlačením klávesu `q`.

2. Dátové typy

Octave má zabudované rôzne dátové typy. Sú to predovšetkým:

- Reálne a komplexné čísla a matice.
- Znaky a reťazce.
- Dátové štruktúry.

Užívateľ si môže definovať vlastné špecializované dátové typy, na to musí napísať krátky kód v jazyku C++. Tejto problematike sa nebudeme venovať.

2.1. Čísla a ich zobrazovanie

Všetky číselné objekty sa uchovávajú s *dvojnásobnou presnosťou* (anglicky *double precision*). V systémoch používajúcich formát s plávajúcou desatinnou bodkou sa dajú uložiť kladné čísla zhruba od $2{,}2251 \times 10^{-308}$ po $1{,}7977 \times 10^{308}$ s relatívnou presnosťou $2{,}2204 \times 10^{-16}$. Presné hodnoty sú priradené premenným `realmin`, `realmax` a `eps`:

```
>> format long
>> realmin
realmin =  2.22507385850720e-308
```

```
>> realmax
realmax =  1.79769313486232e+308
>> eps
eps =  2.22044604925031e-16
```

Zobrazenie čísel na obrazovke (ako echo alebo príkazom `disp`) a pri výstupe do súboru je možné ovplyvniť príkazom `format`[12]. Ako príklad zobrazme riadkový vektor, ktorého hodnoty sú π, π^{10} a π^{-10}.

Pri zapnutí Octave sa nastaví implicitný formát zobrazovania čísel. K tomuto formátu sa môžeme vrátiť z iného nastavenia zadaním príkazu `format` bez parametra. Vidíme, že tento formát „vynuluje“ relatívne malé hodnoty. Najväčšie číslo sa zobrazuje s jednou platnou cifrou pred desatinnou bodkou, dopredu sa vynáša odpovedajúca mocnina desiatky:

```
>> format
>> [pi pi^10 pi^(-10)]
ans =
  1.0e+04  *
  0.00031  9.36480  0.00000
```

Ešte kratší výstup (so zobrazením len troch platných cifier) dosiahneme príkazom `format short`, ktorý v tomto prípade zobrazí len to najväčšie číslo:

[12] Niektoré formáty sa nehodia na zobrazovanie matíc, ktorých prvky majú príliš rôzne veľkosti.

```
>> format short
>> [pi pi^10 pi^(-10)]
ans =
  1.0e+04  *
  0.000  9.365  0.000
```

V tomto prípade je prijateľné nastavenie `format long`:

```
>> format long
>> [pi pi^10 pi^(-10)]
ans =
  1.0e+04  *
  0.000314159265359  9.364804747608298  0.000000001067828
```

Nasledujúci formát je prijateľný aj v *krátkej* verzii:

```
>> format short E
>> [pi pi^10 pi^(-10)]
ans =
   1.0e+04 *
   3.14E-04   9.36E+00   1.07E-09
```

Dlhá[13] verzia (namiesto veľkého `E` je možné použiť aj malé `e`) je v tomto prípade asi najvhodnejšia:

[13] Posledné dve cifry sme v tomto príklade vymazali.

```
>> format long e
>> [pi pi^10 pi^(-10)]
ans =
  1.0e+04  *
  3.141592653589e-04  9.364804747608e+00  1.067827922686e-09
```

Vhodné môže byť aj zobrazenie vo formáte `free` alebo `none`:

```
>> format free
>> [pi pi^10 pi^(-10)]
ans =
 3.1415926535898 93648.047476083 1.0678279226862e-05
>> format none
>> [pi pi^10 pi^(-10)]
ans =
 3.1415926535898 93648.047476083 1.0678279226862e-05
```

Zaujímavá je voľba `+`. Ak je aktívny tento formát, zobrazujú sa len znamienka jednotlivých prvkov matíc a na mieste núl sú medzery:

```
>> format +; a=[1 0 2;-1 1 0;0 0 -7]
a =
  + +
  -+
    -
```

Vyskúšajte si ešte použitie ďalších volieb `bank`, `hex` a `bit`, ich popis nájdete napríklad v príručke Octave (EATON, 1997).

3. Matice a lineárna algebra

Veľmi užitočnou vlastnosťou Octave je to, že pracuje s *komplexnými* maticami ako so základnými objektami. Rovnako ako MATLAB – MATRIX LABORATORY – má Octave zabudované základné maticové operácie. Navyše poskytuje viaceré maticové funkcie, umožňujúce riešiť také úlohy lineárnej algebry, ako napríklad určenie hodnosti, či determinantu matice, riešenie sústav lineárnych algebrických rovníc a rôzne rozklady matíc.

3.1. Matice

Maticu môžeme zadať vymenovaním jej prvkov, ktoré zadávame po riadkoch, oddelených *bodkočiarkou*. Prvky jednotlivých riadkov odpovedajúce jednotlivým stĺpcom sa oddeľujú *čiarkou* alebo *medzerou*. Ak príkaz *nie je ukončený bodkočiarkou*, výsledok priradenia sa znázorní na obrazovke:

```
Matice:1> a=[1 2]
a =
        1        2
Matice:2> A=[1;2]
A =
        1
        2
```

Príkazy v riadkoch 1 a 2 sa líšia jednak označením premenných (Octave *rozlišuje* malé a veľké písmená!) a tiež oddeľovačom jednotlivých prvkov.

Keďže `a` aj `A` sú súčasne matice aj vektory, môžeme príkazom `length` zistiť ich rozmer (výsledky uložíme do riadku):[14]

```
Matice:3> [length(a),length(A)]
ans =
       2       2
```

Matice je možné vytvárať aj po blokoch z už definovaných podmatíc. Pritom je potrebné dbať na to, aby sa bloky dali postaviť vedľa seba alebo pod seba (teda aby sa odpovedajúce rozmery rovnali):

```
Matice:4> d=[a;3]
error: number of columns must match (1 != 2)
error: evaluating assignment expression near line 4, column 2
```

V tomto prípade má 1. riadok (matica `a`) dva stĺpce a 2. riadok (číslo 3) len jeden stĺpec – Octave upozorní na chybu a miesto jej výskytu.

```
Matice:4> b=[a 3],c=[a;[3,4]]
b =
       1       2       3
c =
       1       2
       3       4
```

Prázdna matica sa zadá príkazom `d=[]`. Sú situácie, keď je to nevyhnutné.

[14] Ak sa v príkaze objaví výraz bez priradenia nejakej premennej, hodnotu výrazu priradí Octave zabudovanej premennej `ans` — skratke slova „answer“ – odpoveď.

Príkazom `whos` alebo jeho stručnejšou verziou `who` zistíme, ktoré premenné sú aktívne:

```
Matice:5> whos
...
*** local user variables:
prot  type                        rows   cols  name
====  ====                        ====   ====  ====
 rwd  matrix                         2      1  A
 rwd  matrix                         1      2  a
 rwd  matrix                         1      3  b
 rwd  matrix                         2      2  c
```

Všeobecnejší príkaz ako príkaz `length` je príkaz `size`, ktorý sa dá použiť pre ľubovoľné matice, nielen pre vektory. Výsledkom použitia tohoto príkazu je riadkový vektor, ktorého prvá zložka sa rovná *počtu riadkov* a druhá zložka sa rovná *počtu stĺpcov* vstupnej matice.[15] Riadok 7 ukazuje, ako získame jednotlivé hodnoty výsledného vektora.

```
Matice:6> [size(a),size(b),size(c)]
ans =
       1       2       1       3       2       2
Matice:7> [size(b,2),size(c,1)]
ans =
       3       2
```

[15] Na rozdiel od MATLABu má Octave tiež funkcie `rows` a `columns`.

Premennú odstránime z pamäte príkazom `clear`, napríklad `clear a`. Všetky premenné odstránime z pamäte príkazom `clear all`. Samozrejme tým stratíme možnosť s hodnotami týchto premenných ďalej pracovať.

Octave ponúka jednoduchý spôsob zadávania *podmatíc* (*submatíc*) danej matice. V nižšie uvedenom príklade matica `b` vznikne z matice `a` tak, že sa uvažujú jej prvky na priesečníkoch riadkov 1 a 3 a stĺpcov 2 a 3. Namiesto vymenovania všetkých riadkov alebo všetkých stĺpcov je možné použiť znak `:`, čo sme využili pri definovaní matice `c`, obsahujúcej druhý riadok (a všetky stĺpce) matice `a`:

```
Matice:8> clear all
Matice:9> a=[1 2 3;4 5 6;7 8 9],b=a([1 3],[2 3]),c=a(2,:)
a =
       1       2       3
       4       5       6
       7       8       9
b =
       2       3
       8       9
c =
       4       5       6
```

3.1.1. Komplexné čísla

Pri zadávaní *komplexných* čísel *nesmie byť* medzi imaginárnou časťou a symbolom imaginárnej jednotky (je možné použiť `i`, `I`, `j` alebo `J`) *medzera*. Na iné medzery nie je Octave citlivý.

```
Matice:10> x=3 i+2
parse error:
>>> x=3 i+2
       ^
Matice:10> x=3i +2
x = 2 + 3i
Matice:11> x=2+ 3*j
x = 2 + 3i
Matice:12> x=I*3 + 2
x = 2 + 3i
Matice:13> x=3J+2
x = 2 + 3i
```

Komplexne združené číslo získame použitím funkcie `conj`, veľkosť komplexného čísla vypočítame funkciou `abs`:[16]

[16]Symbol ^ má v Octave význam mocniny a na rozdiel od MATLABu má rovnaký význam aj **, tak ako v jazyku FORTRAN.

```
Matice:14> y=conj(x)
y = 2 - 3i
Matice:15> x*y
ans = 13
Matice:16> abs(x)^2
ans = 13.000
```

3.2. Maticové operácie a funkcie

3.2.1. Súčet, rozdiel, súčin a transponovanie matíc

Uvažujme nasledujúce matice a, b a c:

```
Matice:17> a=[1 -2 3;-4 5 -6]
a =
         1        -2         3
        -4         5        -6
Matice:18> b=ones(2,3)
b =
        1        1        1
        1        1        1
Matice:19> c=[1 2; 0 -3]
c =
         1         2
         0        -3
```

Súčet a *rozdiel* matíc rovnakého typu získame jednoduchým „maticovým“ zápisom:

```
Matice:20> a+b
ans =
         2        -1         4
        -3         6        -5
```

```
Matice:21> a-b
ans =
         0        -3         2
        -5         4        -7
```

Pokus sčítať matice rôznych rozmerov dopadne podľa očakávania:

```
Matice:22> a+c
error: operator +: nonconformant arguments
       (op1 is 2x3, op2 is 2x2)
error: evaluating binary operator ‘+’ near line 22, column 2
```

Výnimkou je prípad, ak ku matici ľubovoľného rozmeru pridáme číslo (maticu rozmeru 1×1). Výsledkom je matica, ktorej všetky prvky sú prvky pôvodnej matice zväčšené o dané číslo:

```
Matice:22> a+2
ans =
         3         0         5
        -2         7        -4
```

Maticu môžeme vynásobiť skalárom:

```
Matice:23> a*2
ans =
         2        -4         6
        -8        10       -12
```

Znakom *transponovania* matice je *apostrof* ’:

```
Matice:24> a’
ans =
         1        -4
        -2         5
         3        -6
```

Podobne ako súčet matíc, aj *súčin* matíc vykoná Octave len v prípade odpovedajúcich rozmerov matíc. Na vykonanie súčinu postačuje znova „maticový“ zápis:

```
Matice:25> a*c
error: operator *: nonconformant arguments
       (op1 is 2x3, op2 is 2x2)
error: evaluating binary operator ‘*’ near line 25, column 2
Matice:25> c*a
ans =
        -7         8        -9
        12       -15        18
```

3.2.2. Zložkové operácie

Občas sa vyskytujú situácie (napríklad pri spracovaní grafiky alebo v štatistike), keď je vhodné vykonať rovnakú operáciu so všetkými zložkami danej matice, prípadne pre matice rovnakých rozmerov vykonať nejakú operáciu po zložkách. Takéto operácie sa vykonávajú s použitím obyčajných znamienok operácií, pred ktoré sa napíše *bodka* „.“. Napríklad:[17]

```
Matice:26> a^2
error: for A^b, A must be square
error: evaluating binary operator ‘^’ near line 26, column 2
Matice:26> a.^2
ans =
        1        4        9
       16       25       36
Matice:27> a.*a
ans =
        1        4        9
       16       25       36
Matice:28> b./a
ans =
   1.00000  -0.50000   0.33333
  -0.25000   0.20000  -0.16667
```

[17] `a.+b` je samozrejme to isté, ako `a+b`.

3.2.3. Základné maticové funkcie

Nižšie uvádzame niektoré často používané maticové funkcie. Doplňujúcu informáciu môžete získať príkazom `help funkcia`. Ďalšie funkcie, ako napríklad `inv`, `pinv`, `eig` a `svd`, uvedieme v nasledujúcich oddieloch 3.3.1 a 3.4.

`det` – determinant matice:

```
Matice:29> a=[1 2;3 4],d=det(a)
a =
        1        2
        3        4
d = -2
```

`rank` – hodnosť matice – počet nezávislých riadkov, resp. stĺpcov:

```
Matice:30> b=[1,2,3;2,4,6;3,6,9],d=det(b),h=rank(b)
b =
        1        2        3
        2        4        6
        3        6        9
d = 0
h = 1
```

`zeros` – nulová matica zadaného rozmeru:

```
Matice:31> z=zeros(2,4)
z =
       0       0       0       0
       0       0       0       0
```

`ones` – matica zadaného rozmeru, ktorej všetky prvky sú jednotky:

```
Matice:32> tri=3*ones(2,3)
tri =
       3       3       3
       3       3       3
```

`eye` – matica zadaného rozmeru, obsahujúca na hlavnej diagonále jednotky, mimo hlavnej diagonály nuly. Ak je táto matica štvorcová, jedná sa o *jednotkovú* maticu. V tomto prípade stačí na jej zadanie jeden rozmer.

```
Matice:33> c=eye(2,3),e=eye(3)
c =
       1       0       0
       0       1       0
e =
       1       0       0
       0       1       0
       0       0       1
```

`norm` – rôzne *normy* (miery „veľkosti“) matice. Implicitne sa volá *spektrálna* alebo 2-norma, `norm(a,1)` je *stĺpcová norma*, `norm(a,Inf)` alebo

`norm(a,inf)` je *riadková* alebo tiež ∞-norma:

```
Matice:34> a
a =
        1        2
        3        4
Matice:35> [norm(a),norm(a,2),norm(a,1),norm(a,Inf)]
ans =
  5.46499  5.46499  6.00000  7.00000
```

`linspace` – vytvára vektor čísel aritmetickej postupnosti (s rovnakým *krokom*, resp. *odstupom*) od `xmin` po `xmax` tak, že celkovo vznikne zadaný počet čísel.[18] Niekedy je praktickejšie zadať tzv. *interval* pomocou príkazu `xmin:krok:xmax`. Ak je v príkaze len jedna dvojbodka, považuje sa to za vynechaný `krok` a tomuto sa automaticky priradí hodnota 1:

```
Matice:36> xmin=1;xmax=5;x=linspace(xmin,xmax,6)
x =
  1.00000  1.80000  2.60000  3.40000  4.20000  5.00000
Matice:37> x=xmin:(xmax-xmin)/5:xmax
x =
  1.00000  1.80000  2.60000  3.40000  4.20000  5.00000
```

`rand, randn` – generátory náhodných čísel, ktoré sa najčastejšie využívajú na simuláciu. `rand` vytvorí maticu zadaného rozmeru, ktorej prvky

[18] Podobný príkaz je `logspace`.

sú náhodné čísla s *rovnomerným rozdelením pravdepodobnosti* na intervale $\langle 0, 1)$, funkcia `randn` vytvorí maticu, ktorej prvky majú *normované normálne rozdelenie pravdepodobnosti*:

```
Matice:38> [rand(2,2),randn(2,2),rand(2,2)]
ans =
   0.88878   0.60670   0.71614  -0.40162   0.25113   0.98916
   0.81418   0.00915  -1.68275   0.13231   0.14057   0.00317
```

Nasledujúce štyri funkcie `max`, `min`, `sum` a `prod` sa vykonávajú po stĺpcoch v prípade, ak má matica viac ako jeden riadok. Ak chceme vykonať operáciu po riadkoch, musíme maticu aj výsledok transponovať.

`max` – maximálne prvky v jednotlivých stĺpcoch, resp. maximálny prvok riadkového vektora:

```
Matice:39> a
a =
        1        2
        3        4
Matice:40> max(a),max(max(a))
ans =
        3        4
ans = 4
```

`min` – minimálne prvky v jednotlivých stĺpcoch, resp. minimálny prvok riadkového vektora:

```
Matice:41> min(a')',min(min(a')')
ans =
        1
        3
ans = 1
```

`sum` – súčet prvkov v jednotlivých stĺpcoch, resp. súčet prvkov riadkového vektora:

```
Matice:42> sum(a),sum(sum(a))
ans =
        4        6
ans = 10
```

`prod` – súčin prvkov v jednotlivých stĺpcoch, resp. súčin prvkov riadkového vektora:

```
Matice:43> prod(a')',prod(prod(a))
ans =
        2
       12
ans = 24
```

3.3. Riešenie sústav lineárnych algebrických rovníc

Riešenie sústav lineárnych algebrických rovníc (SLAR) je jednou zo základných úloh numerickej matematiky. SLAR vznikajú, napríklad, aj pri

riešení nelineárnych sústav Newtonovou metódou, pri výpočte koeficientov splajnov, pri riešení úloh aproximácie lineárnou aj nelineárnou metódou najmenších štvorcov.

V praxi často sa riešia obrovské sústavy lineárnych rovníc, kde počet neznámych dosahuje niekoľko miliónov. Matice sústavy sú často *riedke*, počet ich nenulových prvkov je oveľa nižší, ako počet ich nulových prvkov. Také sústavy sa dajú efektívnejšie uložiť v pamäti, na druhej strane ale vyžadujú špeciálne metódy riešenia.

3.3.1. Frobeniusova veta, inverzná a pseudoinverzná matica

Uvažujme sústavu lineárnych rovníc

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{x} \in \mathbb{R}^n, \quad \boldsymbol{b} \in \mathbb{R}^m,$$

alebo po zložkách

$$\begin{aligned} a_{1,1}\, x_1 + a_{1,2}\, x_2 + \cdots + a_{1,n}\, x_n &= b_1, \\ a_{2,1}\, x_1 + a_{2,2}\, x_2 + \cdots + a_{2,n}\, x_n &= b_2, \\ &\cdots \\ a_{m,1}\, x_1 + a_{m,2}\, x_2 + \cdots + a_{m,n}\, x_n &= b_m. \end{aligned} \tag{1}$$

Uvažujme najprv sústavu (1):

```
Matice:44> A=[1,2,3;4,5,6;7,8,8],b=[6;12;24]
A =
        1        2        3
        4        5        6
        7        8        8
b =
        6
       12
       24
```

Sústava (1) má podľa Frobeniusovej vety riešenie práve vtedy, ak sa hodnosť matice $\boldsymbol{A}$ rovná hodnosti rozšírenej matice $\boldsymbol{A}' = [\boldsymbol{A}|\boldsymbol{b}]$, čo v Octave overíme jednoducho:

```
Matice:45> h1=rank(A), h2=rank([A b]),
h1 = 3
h2 = 3
```

Keďže hodnosť matice sústavy sa rovná hodnosti rozšírenej matice a obe sa rovnajú počtu neznámych 3, sústava má jediné riešenie. Porovnajme tri spôsoby riešenia, z ktorých prvé dve získame pomocou vynásobenia sústavy (1) zľava inverznou maticou `inv(A)*b`, resp. použitím operátora \ (spätná lomka) príkazom `A\b`[19]:

[19] Túto operáciu si môžeme zapamätať ako „delenie zľava“.

```
Matice:46> [inv(A)*b A\b pinv(A)*b]
ans =
  1.0e+01  *
  -0.80000  -0.80000  -0.80000
   1.60000   1.60000   1.60000
  -0.60000  -0.60000  -0.60000
```

Ako vidíme, dostali sme rovnaké riešenia (prvý a druhý stĺpec), $\boldsymbol{x} = (-8, 16, -6)^T$. Vhodnejší je druhý spôsob, ktorý je vo všeobecnosti rýchlejší aj presnejší (pri riešení sústav lineárnych rovníc sa počítajú upresnenia v dvojnásobnej presnosti). Tretiu možnosť vysvetlíme nižšie.

Vymeňme teraz jeden prvok matice $\boldsymbol{A}$ a overme existenciu riešenia:

```
Matice:47> A(3,3)=9
A =
        1        2        3
        4        5        6
        7        8        9
Matice:48> h1=rank(A), h2=rank([A b]),
h1 = 2
h2 = 3
```

Môžeme teda skonštatovať, že pre danú maticu a pravú stranu sústava nemá riešenie. Ak sa pokúsime ju riešiť pomocou inverznej matice, dostávame:

```
Matice:49> x=inv(A)*b
x =
  1.8913e+16
  -3.7827e+16
  1.8913e+16
```

Hoci inverzná matica *neexistuje*, Octave *ju našiel*[20]! V dôsledku výpočtových chýb sa totiž z matice, ktorej determinant je rovný nule stala matica s veľmi malým determinantom, ktorá už inverznú má. Preto je dôležité najprv sa venovať hodnosti matice sústavy a rozšírenej matice.

Hoci sústava nemá riešenie, netreba sa hneď vzdávať. Môžeme sa zaujímať o vektor, ktorý *minimalizuje veľkosť odchýlky* $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2$. V tomto prípade sa dá ukázať, že existuje nekonečné množstvo takýchto vektorov, ktoré sa nazývajú *pseudoriešenia*. Jedno z nich, tzv. *normálne pseudoriešenie*, t. j. pseudoriešenie s najmenšou normou, získame, ak sústavu (1) vynásobíme tzv. *pseudoinverznou* maticou, ktorú uznačujeme $\boldsymbol{A}^+$:

```
Matice:50> xn=pinv(A)*b
xn =
  1.50000
  1.00000
  0.50000
```

Z množstva *rozumných* vektorov je teda v istom zmysle najvhodnejší vektor $\boldsymbol{x}_n = (1.5, 1, 0.)^T$. Pseudoinverznú maticu `pinv` sme použili aj vyššie,

[20] Confucius: „Je ťažké nájsť v tmavej miestnosti čiernu mačku. Najmä ak tam nie je!“

v riadku 46. Samozrejme sme dostali tiež správne riešenie. Je známe, že ak inverzná matica existuje, pseudoinverzná matica sa rovná inverznej. Preto by sme mohli na inverznú maticu zabudnúť a používať vždy len pseudoinverznú, ktorá sa dá navyše použiť aj v prípade sústav s obdĺžnikovou maticou sústavy. Problém je v tom, že algoritmus určenia pseudoinverznej matice je zložitejší, ako algoritmus určenia inverznej matice.

Pozrime sa ešte, čo dostaneme, ak sa pokúsime *vyriešiť neriešiteľnú sústavu* pomocou operátora \:

```
Matice:51> x=A\b
warning: matrix singular to machine precision, rcond=2.2030e-18
warning: attempting to find minimum norm solution
x =
  1.50000
  1.00000
  0.50000
```

Octave nás upozornil na problém a sám navrhol, aby sme našli *normálne pseudoriešenie*. Znova sa preukázala výhoda použitia operátora „spätná lomka“ voči použitiu inverznej matice.

Zmeňme teraz jeden prvok pravej strany:

```
Matice:52> b(3)=18; b'
ans =
        6       12       18
```

a overme podmienku Frobeniusovej vety:

```
Matice:53> h1=rank(A), h2=rank([A b]),
h1 = 2
h2 = 2
```

V tomto prípade má sústava riešenie, ale keďže hodnosť matice sústavy 2 je menšia ako počet premenných 3, sústava má nekonečný počet riešení. Jedno z nich, znova *normálne pseudoriešenie*, získame použitím *pseudoinverznej* matice.

```
Matice:54> xn=pinv(A)*b
xn =
  -0.33333
   0.66667
   1.66667
```

Nesklame nás ani operátor \:

```
Matice:55> x=A\b
warning: matrix singular to machine precision, rcond=2.2030e-18
warning: attempting to find minimum norm solution
x =
  -0.33333
   0.66667
   1.66667
```

Rozumné riešenie je asi v tomto prípade $\boldsymbol{x}_n = (-1/3, 2/3, 5/3)$. Ako nájsť všetky riešenia sústavy? Na to odpovieme nižšie, v oddiele 3.4.

3.3.2. Porovnanie presnosti MATLABu a Octave v prípade zle podmienenej SLAR

Je známe, že presnosť určenia riešenia SLAR súvisí s *podmienenosťou* sústavy, ktorú charakterizuje *číslo podmienenosti* (po anglicky *condition number*) matice sústavy. Určíme ho volaním funkcie `cond`.

Medzi najznámejšie zle podmienené matice patria takzvané Hilbertove matice, ktorých prvky v i-tom riadku a j-tom stĺpci sú definované vzťahom $H(i,j) = 1/(i+j-1)$. Pre Hilbertovu maticu rádu 10 porovnajme *kvalitu* inverznej matice, ktorej vzorec je známy a získa sa volaním funkcie `invhilb`, získanej rôznymi spôsobmi v Octave a MATLABe:[21]

```
Matice:56>  H=hilb(10); % Hilbertova matica radu 10
Matice:57>  Hi=invhilb(10); % Presna inverzna
Matice:58>  Hin=inv(H); % Inverzna volanim funkcie inv
Matice:59>  Hip=pinv(H); % Inverzna pomocou funkcie pinv
Matice:60>  His=H\eye(10); % Ries. s jednotkovou pravou str.
Matice:61>  E10=eye(10); cond(H); % Cislo podmienenosti
ans =  1.6025e+13
```

Násobením matice $\boldsymbol{H}$ a jej inverznej $\boldsymbol{H}^{-1}$ by sme mali dostať jednotkovú maticu `eye(10)`. Vidíme, že pri zmene poradia násobenia pri numerickom výpočte v Octave aj v MATLABe dostávame napriek *teoretickej komutatívnosti násobenia* rôzne výsledky. Číslo podmienenosti je rádovo 10^{13}, čo je

[21] V Octave, rovnako ako v MATLABe a v TEXu znak % označuje začiatok komentára

v porovnaní s *najlepšie* podmienenou jednotkovou maticou, pre ktorú je $\mathrm{cond}(E) = 1$ veľmi veľké číslo, matica H_{10} je teda veľmi zle podmienená. Odchýlky od jednotkovej matice sú:

```
Matice:62> max(max(abs(H*Hi-E10))), max(max(abs(Hi*H-E10)))
ans =   0.0000677034258842   (1.525878906250000e-005)
ans =   0.0000677034258842   (1.525878906250000e-005)
Matice:63> max(max(abs(H*Hin-E10))),max(max(abs(Hin*H-E10)))
ans =   0.0016671568155289   (9.460449218750000e-004)
ans =   0.0000425744801760   (6.866455078125000e-005)
Matice:64> max(max(abs(H*Hip-E10))),max(max(abs(Hip*H-E10)))
ans =   0.0000567361712456   (1.754760742187500e-004)
ans =   0.0001230314373970   (8.392333984375000e-005)
Matice:65> max(max(abs(H*His-E10))),max(max(abs(His*H-E10)))
ans =   0.0000228881835938   (8.010864257812500e-005)
ans =   0.0003849023487419   (0.00127410888672)
```

V zátvorke sú uvádzané výsledky získané programom MATLAB. Vidíme, že ani presná inverzná matica `Hi` pri aritmetike `double` nedáva veľmi dobrý výsledok. Niektoré výpočty sú presnejšie v MATLABe, iné zasa v Octave. Zaujímavé je tiež, že numerická „presná“ inverzná matica nie je najlepšia vo všetkých prípadoch.

Teraz vyskúšajme, akú presnosť dosiahneme pri riešení SLAR. Zvolíme vektor $\boldsymbol{x}$ a vypočítame odpovedajúcu pravú stranu $\boldsymbol{b}$. Potom sústavu vyriešime rôznymi spôsobmi. V zátvorke sú znova odchýlky, dosiahnuté výpočtom v MATLABe:

```
Matice:66> x=(1:10)';
Matice:67> b=H*x;
Matice:68> y1=Hi*b; max(abs(x-y1))
ans =   0.0007376670837402   (0.00195312500000)
Matice:69> y2=Hin*b; max(abs(x-y2))
ans =   0.0007755756378174   (0.00219726562500)
Matice:70> y3=Hip*b; max(abs(x-y3))
ans =  0.0041679143905640    (0.00244140625000)
Matice:71> y4=His*b; max(abs(x-y4))
ans = 0.0114405155181885     (0.03906250000000)
Matice:72> y5=H\b; max(abs(x-y5))
ans =   0.0003192367215856   (0.00157805385351)
```

Najlepšiu presnosť sme získali v Octave riešením sústavy s použitím operátora \ (pri riešení sa používajú upresnenia s vyššou presnosťou ako je `double`).

3.3.3. Riedke matice

Octave 2.1.50 poskytuje niekoľko funkcií na prácu s riedkymi maticami. Tieto sa nachádzajú v podpriečinkoch

```
libexec/octave/2.1.50/site/oct/i686-pc-cygwin/octave-forge
```

a

```
share/octave/2.1.50/site/m/octave-forge/sparse
```

hlavného adresára, napríklad,

```
C:\Program Files\GNU Octave 2.1.50\opt\octave\
```

Hoci v porovnaní s MATLABom je týchto funkcií oveľa menej, je možné získať aspoň základnú predstavu o definovaní riedkych matíc, o práci s nimi a o riešení SLAR s riedkymi maticami (na riešenie odporúčame použiť operátor \: `x=A\b` namiesto funkcie `spinv`).

Keďže sa jedná o špeciálne funkcie, uvádzame len ich zoznam v poradí podľa ich dôležitosti:[22]

Funkcie na prácu s riedkymi maticami: `sparse, spy, trisolve, splu, spinv, sprand, spabs, spsum, spstats, issparse, is_sparse, is_real_sparse, is_complex_sparse, sphcat, spvcat, sp_test, fem_test, spdiags.`

3.4. Vlastné čísla, vlastné vektory a singulárny rozklad matíc

3.4.1. Vlastné čísla a vlastné vektory matíc

Vlastné čísla matíc nám pomáhajú pochopiť vlastnosti rôznych funkcií či riešení rôznych úloh. Napríklad v teórii riadenia lineárnych systémov nás zaujímajú reálne zložky vlastných čísel matíc (anglicky *eigenvalues*). V Octave získame vlastné čísla volaním maticovej funkcie `eig`:

[22] Na podrobnejšie oboznámenie sa s týmito funkciami použite „pomoc“ `help funkcia`.

```
Matice:73> A=[1 -2; 3 4]
A =
         1        -2
         3         4
Matice:74> eig(A)
ans =
  2.50000 + 1.93649i
  2.50000 - 1.93649i
```

Ak potrebujeme okrem vlastných čísel matíc aj vlastné vektory, použijeme volanie funkcie `eig` s dvomi výstupnými maticami:

```
Matice:75> [V,lambda]=eig(A)
V =
  -0.38730 + 0.50000i  -0.38730 - 0.50000i
   0.77460 + 0.00000i   0.77460 - 0.00000i
lambda =
  2.50000 + 1.93649i  0.00000 + 0.00000i
  0.00000 + 0.00000i  2.50000 - 1.93649i
```

Overme, že prvý stĺpec matice V je vlastný vektor matice A odpovedajúci vlastnému číslu λ_1:

```
Matice:76> v1=V(:,1)
v1 =
  -0.38730 + 0.50000i
   0.77460 + 0.00000i
```

```
Matice:77> [A*v1 lambda(1)*v1]
ans =
  -1.93649 + 0.50000i  -1.93649 + 0.50000i
   1.93649 + 1.50000i   1.93649 + 1.50000i
```

Na záver na príklade overme, že vlastné čísla symetrických matíc sú reálne a vlastné vektory odpovedajúce rôznym vlastným číslam sú navzájom kolmé:

```
Matice:78> A(2,1)=A(1,2)
A =
         1        -2
        -2         4
```

```
Matice:79> [V,lambda]=eig(A)
V =
  -0.89443  -0.44721
  -0.44721   0.89443
lambda =
        0         0
        0         5
```

```
Matice:80> V(:,1)'*V(:,2)
ans =  -1.2604e-17
```

3.4.2. Singulárny rozklad matíc

Singulárny rozklad (anglicky *singular value decomposition*) matíc (SVD) je jedna z najužitočnejších maticových funkcií. Hoci v teórii operátorov bol známy už začiatkom 20. storočia, skutočný rozvoj zaznamenalo jeho praktické použitie až koncom 60-tych rokov, keď G. H. Golub[23] navrhol algoritmus SVD matíc (GOLUB, 1968). Dnes sa SVD používa na riešenie snáď desiatok rôznych úloh. Podrobnejšie sa o singulárnom rozklade matíc dočítate v knihách (GOLUB A VAN LOAN, 1989; FORSYTHE, MALCOLM A MOLER, 1977; KAHANER, MOLER A NASH, 1989; KAUKIČ, 1998).

Teraz len stručne napíšme, že ľubovoľná matica $\boldsymbol{A}$ typu m/n sa dá rozložiť na súčin

$$\boldsymbol{A} = \boldsymbol{U} \cdot \boldsymbol{S} \cdot \boldsymbol{V}^T, \tag{2}$$

kde matice $\boldsymbol{U}$ a $\boldsymbol{V}$ sú ortogonálne matice, ktorých stĺpce tvoria *singulárne vektory* a matica $\boldsymbol{S}$ je (obdĺžniková) diagonálna matica, ktorá má na diagonále nezáporné *singulárne čísla* σ_k matice $\boldsymbol{A}$. Rozklad matice $\boldsymbol{A}$ sa dá napísať aj v tvare súčtu

$$\boldsymbol{A} = \sum_{k=1}^{\min(m,n)} \sigma_k \cdot \boldsymbol{u}_k \cdot \boldsymbol{v}_k^T, \tag{3}$$

kde $\boldsymbol{u}_k$ a $\boldsymbol{v}_k$ sú stĺpce matíc $\boldsymbol{U}$ a $\boldsymbol{V}$.

[23] Môže nás tešiť, že to bolo publikované v „našom“ časopise Aplikace Matematiky.

```
Matice:81> svd(A)
ans =
  5.00000
  0.00000
```

Nulové singulárne číslo svedčí o tom, že matica A má nulový determinant. Úplný rozklad získame použitím nasledujúceho volania funkcie `svd`:

```
Matice:82> [U,S,V]=svd(A)
U =
  -0.44721   0.89443
   0.89443   0.44721
S =
  5.00000  0.00000
  0.00000  0.00000
V =
  -0.44721   0.89443
   0.89443   0.44721
```

Keď porovnáme výsledky použitia funkcií `eig` a `svd` vidíme, že sa prakticky nelíšia. Je to preto, lebo singulárny rozklad je v podstate zovšeobecnením rozkladu symetrických matíc pomocou vlastných hodnôt a vektorov na prípad nesymetrických matíc. Preto pre symetrické matice sú výsledky prakticky zhodné (rozdiel spočíva v tom, že singulárne čísla sú nezáporné, hoci vlastné čísla môžu byť aj záporné – v tom prípade sa singulárne a vlastné čísla symetrických matíc môžu líšiť len v znamienku).

Pozrime sa ešte na prípad nesymetrickej matice

```
Matice:83> A(2,1)=3
A =
         1        -2
         3         4
```

```
Matice:84> svd(A)
ans =
  5.11667
  1.95440
```

```
Matice:85> [U,S,V]=svd(A)
U =
   0.22975  -0.97325
  -0.97325  -0.22975
S =
  5.11667  0.00000
  0.00000  1.95440
V =
  -0.52573  -0.85065
  -0.85065   0.52573
```

V tomto prípade nesymetrickej matice sa už vlastné a singulárne čísla (ani vektory) nerovnajú (pozrite riadky 73–75).

3.4.3. Použitie singulárneho rozkladu pri riešení SLAR

Vypočítajme singulárny rozklad matice A z riadku 47 na strane 44:

```
Matice:86> [U,S,V]=svd(A)
U =
  -0.21484   0.88723   0.40825
  -0.52059   0.24964  -0.81650
  -0.82634  -0.38794   0.40825
S =
  1.0e+01  *
  1.68481  0.00000  0.00000
  0.00000  0.10684  0.00000
  0.00000  0.00000  0.00000
V =
  -0.47967  -0.77669  -0.40825
  -0.57237  -0.07569   0.81650
  -0.66506   0.62532  -0.40825
```

Podľa matice S vidíme, že matica sústavy má jedno nulové singulárne číslo (a teda jej determinant bude nulový). Tomuto singulárnemu číslu odpovedá singulárny vektor – tretí stĺpec matice V, teda `V(:,3)`. Množina pseudoriešení (a v prípade rovnosti hodností matice sústavy a rozšírenej matice sústavy množina riešení) bude $x = x_n + t \cdot v_3$, kde $t \in \mathbb{R}$ je ľubovoľné reálne číslo (parameter) a x_n je normálne pseudoriešenie.

Porovnajme pravé strany pre rôzne hodnoty parametra t:

```
Matice:87> v3=V(:,3)
v3 =
  -0.40825
   0.81650
  -0.40825
```

```
Matice:88> [A*(xn+0*v3) A*(xn+11*v3) A*(xn-37*v3)]
ans =
  1.0e+01  *
  0.60000  0.60000  0.60000
  1.20000  1.20000  1.20000
  1.80000  1.80000  1.80000
```

Podobne pre maticu A:

```
Matice:89> A=[1 2 3;2 4 6;3 6 9]
A =
       1       2       3
       2       4       6
       3       6       9
```

získame singulárny rozklad

```
Matice:90> [U,S,V]=svd(A)
U =
  -0.26726   0.95351  -0.13922
  -0.53452  -0.26690  -0.80190
  -0.80178  -0.13990   0.58101
S =
  1.0e+01  *
  1.40000  0.00000  0.00000
  0.00000  0.00000  0.00000
  0.00000  0.00000  0.00000
V =
  -0.26726   0.76530   0.58557
  -0.53452  -0.62336   0.57071
  -0.80178   0.16047  -0.57567
```

z ktorého podľa matice S vidíme, že matica A má dve nulové singulárne čísla.

Zvoľme nejaký vektor x a vypočítajme vektor $b = A\,x$. Sústava (1) bude mať v tomto prípade nekonečne veľa riešení:

```
Matice:91> x=[1,-1,3]'
x =
         1
        -1
         3
```

```
Matice:92> b=A*x
b =
        8
       16
       24
```

Ak označíme v_2 a v_3 druhý a tretí stĺpec matice V, tak všeobecné riešenie uvažovanej sústavy bude mať tvar $x = x_n + t_1 \cdot v_2 + t_2 \cdot v_3$, kde $t_1, t_2 \in \mathbb{R}$ sú ľubovoľné reálne parametre.

```
Matice:93> xn=pinv(A)*b
xn =
  0.57143
  1.14286
  1.71429
Matice:94> v2=V(:,2),v3=V(:,3)
v2 =
   0.76530
  -0.62336
   0.16047
v3 =
   0.58557
   0.57071
  -0.57567
```

```
Matice:95> [A*(xn+v2-3*v3) A*(xn-11*v2+7*v3) A*(xn+6*v2-13*v3)]
ans =
  1.0e+01  *
  0.80000  0.80000  0.80000
  1.60000  1.60000  1.60000
  2.40000  2.40000  2.40000
```

Vidíme, že pre tri rôzne dvojice parametrov (t_1, t_2) dostávame rovnaké pravé strany. Určme ešte hodnoty parametrov, ktoré odpovedajú zvolenému vektoru x:

```
Matice:96> t=pinv([v2,v3])*(x-xn)
t =
   1.87007
  -1.71214
```

```
Matice:97> xn+t(1)*v2+t(2)*v3
ans =
   1.00000
  -1.00000
   3.00000
```

Na záver ešte uveďme vzorec pseudoinverznej matice na základe jej singulárneho rozkladu (2). Označme S^+ diagonálnu maticu, ktorá má rozmer $n \times m$ a na hlavnej diagonále má prevrátené nenulové singulárne hodnoty σ_k alebo nuly (v prípade nulových singulárnych čísel). Potom platí:

$$A^+ = V \cdot S^+ \cdot U^T. \tag{4}$$

Porovnajme výsledky výpočtu *pseudoinverznej matice* na základe volania funkcie `pinv` a na základe využitia vzťahu (4).[24]:

```
Matice:98> SP=diag(S); SP(1,1)=1/SP(1,1);SP=diag(SP)
SP =
   1.0e-02 *
  7.14286  0.00000  0.00000
  0.00000  0.00000  0.00000
  0.00000  0.00000  0.00000
```

```
Matice:99> [pinv(A) V*SP*U']
ans =
  1.0e-02  *
  0.51020  1.02041  1.53061  0.51020  1.02041  1.53061
  1.02041  2.04082  3.06122  1.02041  2.04082  3.06122
  1.53061  3.06122  4.59184  1.53061  3.06122  4.59184
```

[24]V tomto príklade sme použili aj funkciu `diag` Táto funkcia vráti vektor s prvkami diagonály vstupnej matice, alebo naopak, vráti maticu s diagonálou obsahujúcou prvky vstupného vektora.

4. Reťazce

4.1. Zadávanie reťazcov

Reťazce tvoria *postupnosti znakov*. Tieto postupnosti sa uzatvárajú medzi dvojice znakov ’ (apostrofov) alebo " (úvodzoviek). Keďže „apostrof“ sa používa aj ako *znak transponovania*, je lepšie používať úvodzovky[25]. Reťazce môžu byť priradené premenným:

```
Retazce:1> E="Ema ma mamu"
E = Ema ma mamu
Retazce:2> whos
...
*** local user variables:

prot  type                      rows   cols  name
====  ====                      ====   ====  ====
 rwd  string                       1     11  E
```

Niektoré znaky nemôžu byť zadávané priamo, na ich zadávanie sa používa tzv. *únikový znak* \ (spätná lomka). Napríklad nemôže dobre dopadnúť pokus zadať znak " medzi dvojicou úvodzoviek. Na druhej strane môže byť „úvodzovka“ zadaná medzi apostrofmi:

[25] V MATLABe sa používajú *len* apostrofy!

```
Retazce:3> u="""
error: unterminated string constant
parse error:
>>> u="""
        ^
Retazce:3> u="\"\\"
u = "\
Retazce:4> u='"\\'
u = "\
```

Únikový znak sa používa aj na zadávanie znakov, ktoré sa nezobrazujú (nový riadok, tabulátor, a iné), podobne ako v programovacom jazyku C. Účinok však nie je vždy jasný. Uvádzame preto len niektoré:

- `\\` – zadáva znak spätná lomka \,
- `\"` – zadáva znak úvodzoviek ",
- `\'` – zadáva znak apostrof ',
- `\a` – reprezentuje zvukový signál,
- `\n` – zadáva nový riadok,
- `\t` – zadáva horizontálny tabulátor,
- `\v` – reprezentuje vertikálny tabulátor.

Reťazce môžeme spájať tak, že ich umiestnime do matice vedľa seba alebo použijeme funkciu `strcat`:

```
Retazce:5> a="Zacia";b="tok";c=[a b]
c = Zaciatok
Retazce:6> d=strcat(a,b)
d = Zaciatok
```

Uložením do matice nad sebou môžeme vytvárať viacriadkové textové výrazy. Na rozdiel od MATLABu, kde výrazy ukladané nad sebou musia mať rovnakú dĺžku, Octave doplní chýbajúce znaky podľa toho, ako je aktuálne definovaná systémová premenná `string_fill_char`:

```
Retazce:7> a="ko"; b="niec"; c=[a;b]
c =
ko
niec
Retazce:8> string_fill_char="*"
string_fill_char = *
Retazce:9> c
c =
ko
niec
```

```
Retazce:10> c=[a;b]
c =
ko**
niec
```

4.2. Funkcie orientované na reťazce

4.2.1. Vytváranie reťazcov

`blanks(n)` – vráti reťazec s `n` medzerami:

```
Retazce:11> a=blanks(4)
a =
Retazce:12> ["b" a "c"]
ans = b    c
```

`com2str(a,flag)` – vráti reťazec odpovedajúci komplexnému číslu `a`. Premenná `flag` ovplyvňuje presný tvar reťazca:

```
Retazce:13> a=3i-2
a = -2 + 3i
Retazce:14> z=com2str(a)
z = -2 + 3i
```

```
Retazce:15> whos z
*** local user variables:
prot  type                        rows   cols  name
====  ====                        ====   ====  ====
 rwd  string                         1      7  z
```

`int2str, num2str` – funkcie vracajú reťazec odpovedajúci číslu, funkcia `int2str` predtým číslo skonvertuje na celé:

```
Retazce:16> int2str(-3.14)
ans = -3
Retazce:17> mp=num2str(-3.14)
mp = -3.14
```

`isstr(a)` – vráti 1, ak je premenná a reťazec, v opačnom prípade vráti 0:

```
Retazce:18> isstr(mp)
ans = 1
Retazce:19> isstr(-3.13)
ans = 0
```

`setstr(m)` – vráti maticu ASCII znakov odpovedajúcich hodnotám prvkov matice `m`:[26]

[26] Pri znakoch s hodnotou nad 127 zobrazenie výsledku závisí od „prostredia“, v ktorom sa znaky zobrazujú.

```
Retazce:20> m=[51 52; 88 89]
m =
       51       52
       88       89
Retazce:21> n=setstr(m)
n =
34
XY
Retazce:22> whos m n
*** local user variables:
prot  type                        rows   cols  name
====  ====                        ====   ====  ====
 rwd  matrix                         2      2  m
 rwd  string                         2      2  n
```

`strcat` – ukladá reťazce vedľa seba, rovnaký výsledok získame uložením reťazcov do riadku matice:

```
Retazce:23> s=[" *";"* "]
s =
 *
*
```

```
Retazce:24> strcat(s,s,s,s)
ans =
 * * * *
* * * *
Retazce:25> [s,s,s,s]
ans =
 * * * *
* * * *
```

`str2mat` – ukladá reťazce nad sebou, rovnaký výsledok získame uložením reťazcov do stĺpca matice (obidva spôsoby sa dajú aj kombinovať):

```
Retazce:26> [str2mat(s,s) [s;s]]
ans =
 * *
* *
 * *
* *
```

4.2.2. Vyhľadávanie a výmena podreťazcov

`deblank(r)` – odstráni medzery na konci reťazca `r`

`index(s,r)`, `rindex(s,r)` – funkcie vracajú pozíciu *prvého* resp. *posledného* výskytu reťazca `r` vnútri reťazca `s`:

```
Retazce:27> E="Ema ma mamu"
E = Ema ma mamu
Retazce:28> index(E,"ma")
ans = 2
Retazce:29> rindex(E,"ma")
ans = 8
```

`findstr(s,r)` – vyhľadá všetky začiatočné pozície reťazca `r` vo vnútri reťazca `s`. Ak je funkcia volaná s tretím argumentom rovným 0, počítajú sa len neprekrývajúce sa výskyty reťazca `r`:

```
Retazce:30> B="abababababab", findstr(B,"bab"),findstr(B,"bab",0)
B = abababababab
ans =
       2       4       6       8
ans =
       2       6
```

`split(s,r)` – rozdelí reťazec `s` na tie časti, ktoré sú oddelené reťazcom `r`. Jednotlivé časti sa zapisujú do riadkov výslednej matice rovnakej dĺžky, chýbajúce znaky sú doplnené medzerami a dajú sa odstrániť pomocou funkcie `deblank`:

```
Retazce:31> Veta="Dnes je pekne pocasie"
Veta = Dnes je pekne pocasie
```

```
Retazce:32> Slova=split(Veta," ")
Slova =
Dnes
je
pekne
pocasie
Retazce:33> Slova(2,:)
ans = je
Retazce:34> length(Slova(2,:))
ans = 7
```

`strcmp(r1,r2)` – porovná reťazce `r1` a `r2`:

```
Retazce:35> a="kuk",b="kuk "
a = kuk
b = kuk
Retazce:36> strcmp(a,b)
ans = 0
Retazce:37> strcmp(a,deblank(b))
ans = 1
```

`strrep(r,s1,s2)` – v reťazci `r` nahradí všetky výskyty podreťazca `s1` podreťazcom `s2`:

```
Retazce:38> veta=strrep(Veta,"pekne","neprijemne")
veta = Dnes je neprijemne pocasie
```

`substr(r,zaciatok,pocet)` – vypíše podreťazec reťazca `r`, ktorý začína na pozícii `zaciatok` a má dĺžku `pocet`:

```
Retazce:39> slovo=substr(Veta,9,5)
slovo = pekne
```

4.2.3. Konverzie reťazcov

V tomto oddiele len čiastočne vymenujeme odpovedajúce funkcie. Podrobnejšie informácie v prípade potreby nájdete v príručke Octave (EATON, 1997) alebo ich získate použitím príkazu `help funkcia`.

Funkcie na konverziu reťazcov – `bin2dec, dec2bin, dec2hex, hex2dec, str2num, toascii, tolower, toupper.`

4.2.4. Znakové funkcie

Na tomto mieste znova len vymenujeme odpovedajúce funkcie. Podrobnejšie informácie v prípade potreby nájdete v príručke Octave (EATON, 1997) alebo ich získate použitím príkazu `help funkcia`.

Znakové funkcie – `isalnum, isalpha, isascii, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit.`

5. Dátové štruktúry, zoznamy a bunkové polia

5.1. Štruktúry

Pri rôznych príležitostiach je vhodné pracovať s rôznymi *štruktúrami* údajov. V Octave je možné vytvárať štruktúry, ktoré *zjednocujú* do skupín údaje rôznych typov, vrátane samotných štruktúr (ale aj zoznamov a bunkových polí, ktoré popíšeme ďalej).

Ak by sme chceli uchovať niektoré osobné údaje o autorovi, mohli by sme to zorganizovať napríklad nasledujúcim spôsobom:

```
1> autor.meno='Jan'; autor.priezvisko="Busa"; autor.vek=48
autor =
{
  meno = Jan
  priezvisko = Busa
  vek = 48
}
2> whos autor
*** local user variables:
prot  type                        rows   cols  name
====  ====                        ====   ====  ====
 rwd  struct                         1      1  autor
```

V prípade, ak chceme uložiť viacero údajov s *rovnakou štruktúrou*, môžeme vytvoriť *maticu* alebo *pole* štruktúr. Poradie prvku sa zadáva hneď za

názvom *štruktúry*:

```
3> rodina(1).meno='otec'; rodina(1).vek=48;
4> rodina(2).meno='syn'; rodina(2).vek=25;
5> rodina(3).meno='dcera'; rodina(3).vek=22
rodina =
{
  meno =
  (
    [1] = otec
    [2] = syn
    [3] = dcera
  )
  vek =
  (
    [1] = 48
    [2] = 25
    [3] = 22
  )
}
```

Vek druhého člena rodiny potom zistíme príkazom:

```
6> rodina(2).vek
ans = 25
```

Hoci počet vnorení v nejakej štruktúre je neohraničený, zmenou hod-

noty systémovej premennej `struct_levels_to_print` sa dá ovplyvniť výpis štruktúry. Napríklad:

```
7> a.b.c.d='e'; a.b.cc=3;
```

```
8> struct_levels_to_print=-1; a
a = <structure>
9> struct_levels_to_print=0; a
a =
{
  b: struct
}
10> struct_levels_to_print=1; a
a =
{
  b =
  {
    c: struct
   cc: scalar
  }
}
```

```
11> struct_levels_to_print=2; a
a =
{
  b =
  {
    c =
    {
      d: string
    }
    cc = 3
  }
}
12> struct_levels_to_print=3; a
a =
{
  b =
  {
    c =
    {
      d = e
    }
    cc = 3
  }
}
```

5.2. Zoznamy

Zoznam je *lineárna štruktúra*, ktorej prvky môžu byť matice čísel či reťazcov, ale aj štruktúry. Zoznamy sa vytvárajú príkazom `list`, jednotlivé prvky zoznamov sú chápané ako zoznamy. Napríklad, ak zadefinujeme štruktúru `datum`, potom môžeme vytvoriť nasledujúci zoznam:

```
13> datum.den=16; datum.mesiac=3;
14> Zoznam=list([3 4],["a";'b'],datum)
Zoznam =
(
  [1] =
          3        4
  [2] =
a
b
  [3] =
  {
    den = 16
    mesiac = 3
  }
)
15> z2=Zoznam(2); whos z2
prot  type                      rows   cols  name
 rwd  list                         -      -  z2
```

Pri pohľade na riadok 15 vidíme, že typ druhého prvku zoznamu `Zoznam` je `list` (zoznam) a nie *reťazcová* matica, ako by sa dalo očakávať. Aby sme sa dostali k druhému prvku zoznamu, musíme použiť funkciu `nth`[27]:

```
16> m=nth(Zoznam,2), whos m, m(2)
m =
a
b

*** local user variables:
prot  type                      rows   cols  name
====  ====                      ====   ====  ====
 rwd  string                       2      1  m

ans = b
```

Príkaz `reverse(z)` slúži na obrátenie poradia prvkov zoznamu `z`.

Na jednoduché spájanie zoznamov slúži funkcia `append`. V riadku 17 vidíme, že ku zoznamu `z1` sa nepridal *jeden prvok* – zoznam `z2`, ale zoznam `z2` sa *rozpadol* na jednotlivé prvky (podzoznamy) a tieto boli pridané k pôvodnému zoznamu:

[27] Slovenský preklad názvu tejto funkcie by bol asi „entý“ :).

```
17> z1=list(1,'a');z2=list(2,"b"); z=append(z1,z2)
z =
(
  [1] = 1
  [2] = a
  [3] = 2
  [4] = b
)
```

Všeobecnejšou funkciou je funkcia `splice(z,zac,dlzka,z3)`. V zozname z sa od pozície zac nahradí poc prvkov zoznamu zoznamom z3. Ak sa zac rovná dĺžke zoznamu z (`lenght(z)`) zväčšenej o 1 a poc sa rovná 0, je príkaz `splice` ekvivalentný s príkazom `append`:

```
18> z3=list([1 2;3 4],5); splice(z,2,2,z3)
ans =
(
  [1] = 1
  [2] =
          1        2
          3        4
  [3] = 5
  [4] = b
)
```

5.3. Bunkové polia

Bunkové polia môžeme považovať za dvojrozmerné zoznamy. Príkazom `cell(m,n)` zadefinujeme tvar poľa, obyčajným priradením potom *naplníme* jednotlivé bunky:

```
19> chaos=cell(1,2); chaos(1)=[1,2;3,4];
20> x.a='a'; x.b=list(-1,'c'); chaos(1,2)=x
chaos =
{
  [1,1] =

          1        2
          3        4

  [1,2] =
  {
    a = a
    b =
    (
      [1] = -1
      [2] = c
    )
  }
}
```

Pri volaní funkcie `cell` s jedným parametrom sa vytvorí štvorcové pole buniek.

K *hodnotám* jednotlivých buniek poľa, ktoré Octave chápe ako bunky, sa dostaneme použitím príkazu `nth`[28]:

```
22> nth(chaos(2),1)
ans =
{
  a = a
  b =
  (
    [1] = -1
    [2] = c
  )
}
```

Kombinovaním príkazov `nth` a bodkových operácií v štruktúrach sa môžeme dostať k *hodnotám*, ktoré potrebujeme:

```
23> nth(nth(chaos(1,2),1).b,2)
ans = c
```

[28] Správne by sme mali použiť `nth(chaos(1,2),1)`, ale Octave umožňuje pracovať s prvkami matice, ako keby boli uložené do jedného vektora (po stĺpcoch).

6. Výrazy a operátory

6.1. Výrazy

Výrazy sú základnými stavebnými prvkami príkazov Octave. Pomocou nich sa vytvárajú hodnoty, ktoré je možné priradiť nejakej premennej, uložiť, vytlačiť, prípadne zaslať ako vstupy do nejakých funkcií a pod. Aj samostatný výraz môžeme chápať ako príkaz. Z jednoduchších výrazov sa skladajú výrazy zložitejšie, ktoré môžu obsahovať konštanty, premenné, indexy, funkcie, atď.

Keďže sa Octave principiálne nelíši od štandardných jazykov programovania, nebudeme sa tu tejto problematike venovať v celej šírke a hĺbke.

6.1.1. Aritmetické výrazy

Aritmetické výrazy vytvárame pomocou číselných premenných, konštánt a funkcií. Štandardné operácie (súčet, rozdiel, súčin, podiel a umocňovanie) označujeme odpovedajúco `+`, `-`, `*`, `/` a `^` alebo `**`. Všetky tieto operácie sú maticové a preto sa dajú vykonávať len vtedy, keď sú operandy odpovedajúcich typov. Napríklad operácia `c*b`, kde `c` a `b` sú matice sa chápe ako súčin matíc a nie matica súčinov. Vzhľadom na to, že aj zložkové operácie môžu byť v praxi zaujímavé, sú v Octave implementované aj *zložkové operácie* s maticami *rovnakého typu* – tieto zložkové operácie majú vpredu pred znamienkom operácie bodku, sú to teda `.+`, `.-`, `.*`, `./` a `.^` alebo `.**` (samozrejme, pre súčet a rozdiel sú zložkové a maticové operácie identické).

Jednou z netradičných možností Octave je viacnásobné priradenie hodnoty v jednom príkaze, napríklad `x=y=z=3`.

Okrem týchto operácií pozná Octave aj operácie *zväčšovania* (*zmenšovania*) o jednotku (anglicky *increment operators*), známe z jazyka C++ (tu aj na iných miestach by sme mohli písať jednoducho C). Dôležité je vedieť, čo bude vyhodnotené ako hodnota výrazu, obsahujúceho takéto operácie:

```
Vyrazy:1> [a,(a++),a]
ans =
  -2.31000  -2.31000  -1.31000
Vyrazy:2> [a,(++a),a]
ans =
  -1.31000  -0.31000  -0.31000
Vyrazy:3> [a,(--a),a]
ans =
  -0.31000  -1.31000  -1.31000
Vyrazy:4> [a,(a--),a]
ans =
  -1.31000  -1.31000  -2.31000
```

Všimnime si, že Octave vytvára hodnoty matice zľava doprava, pričom prvky nezapĺňa hodnotami, ktoré boli pred vytváraním matice, ale *aktuálnymi*! Vidíme, že ak je ++ alebo -- zľava, najprv sa vykoná *inkrementácia* (zväčšenie alebo zmenšenie) a nová hodnota sa chápe ako hodnota celého výrazu. Ak sú znamienka sprava, najprv sa vyhodnotí hodnota výrazu a

potom sa uskutoční inkrementácia.[29]

6.1.2. Logické výrazy a funkcie

Logické výrazy sa vytvárajú kombináciou logických konštánt, premenných a funkcií, ktoré vracajú logické hodnoty. Logickú premennú môžeme vytvoriť, napríklad, priradením logického výrazu alebo priamo hodnoty logickej konštanty `true` alebo `false`:

```
Vyrazy:5> c=true
c = 1
Vyrazy:6> c=(2==3)
c = 0
Vyrazy:7> whos c
*** local user variables:
prot  type                        rows   cols  name
====  ====                        ====   ====  ====
 rwd  bool                           1      1  c
```

Logické operácie konjunkcia, disjunkcia a negácia sa označujú `&`, `|` a `~`. Sú to zložkové operácie, keď sú operandy matice.

Všimnime si, že do logických výrazov môžu vstupovať nielen logické premenné. Octave pri práci s logickým výrazom vyhodnotí ako `true` (pravdivú) každú nenulovú hodnotu (vrátane reťazca).

[29] Vyskúšajte podobný príklad, ale operátor inkremenácie vložte, napríklad, do stredu matice 3×3.

```
Vyrazy:8> a=[1 0.5;0 -3]
a =
   1.00000   0.50000
   0.00000  -3.00000
Vyrazy:9> b=[0 -2;0 3]
b =
         0        -2
         0         3
Vyrazy:10> c=a&b
c =
        0        1
        0        1
Vyrazy:11> c=a|b
c =
        1        1
        0        1
Vyrazy:12> c=~a
c =
        0        0
        1        0
Vyrazy:13> c="n"&(-2)
c = 1
Vyrazy:14> c="nie"&(-2)
c =
        1        1        1
```

Okrem zložkových logických operácií pracuje Octave aj s *maticovými*:

```
Vyrazy:15> c=a&&b
c = 0
Vyrazy:16> c=a||b
c = 0
```

Tieto operácie sú ekvivalentnými s použitím operácií `all(all(c&b))` resp. `all(all(c|b))`, kde `all` (*všetky*) je maticová logická funkcia, ktorá vracia riadkový vektor (podobne ako `max`) s hodnotami 1 (`true`), v tých stĺpcoch, pre ktoré sú prvky vo všetkých riadkoch *pravdivé* (nenulové) alebo s hodnotami 0 (`false`) v opačnom prípade. Ak je vstupným argumentom riadkový vektor, výslednou hodnotou bude 1, ak budú všetky jeho prvky nenulové. Podobne je definovaná funkcia `any` (*nejaký*), ktorá v prípade riadkového vektora vráti 1, ak je aspoň jedna jeho hodnota nenulová. Uveďme ešte funkciu `xor` (anglicky *exclusive or* – „buď jedno – alebo druhé“):

```
Vyrazy:17> xor(a,b)
ans =
        1        0
        0        0
Vyrazy:18> e=[0 -2; 3 0.4]
e =
   0.00000  -2.00000
   3.00000   0.40000
```

```
Vyrazy:19> all(e)
ans =
        0        1
Vyrazy:20> any(e)
ans =
        1        1
Vyrazy:21> [all(all(e)) all(any(e)) any(all(e)) any(any(e))]
ans =
        0        1        1        1
```

Nasledujúce riadky ukazujú ešte jeden dôležitý rozdiel medzi operátormi `&` a `&&` (dalo by sa očakávať, že v prípade skalárnych operandov bude výsledok rovnaký!) – v prípade operácie `a&&b` sa najskôr vyhodnotí výraz „`a`“. V prípade, ak je tento výraz nulový, hneď sa ukončí vyhodnotenie celého logického výrazu. V prípade operácie `a&b` sa najskôr vyhodnotia obidva výrazy `a` aj `b`!

```
Vyrazy:22> a=0;b=3;a&b++,b
ans = 0
b = 4
Vyrazy:23> a=0;b=3;a&&b++,b
ans = 0
b = 3
Vyrazy:24> a=0;b=3;b++&&a,b
ans = 0
b = 4
```

Užitočnou logickou funkciou je funkcia `find`, ktorá vyhľadáva *indexy* (pozície) *nenulových* (pravdivých) prvkov matice. Výsledkom je jednorozmerný stĺpcový vektor indexov, ktoré sú číslované po stĺpcoch zhora dole, zľava doprava:

```
Vyrazy:25> e=[1,2,0;-2,0,5]
e =
         1         2         0
        -2         0         5
Vyrazy:26> find(e)'
ans =
        1        2        3        6
```

Na záver uveďme ešte *relačné operátory* (použité nižšie), ktoré sa štandardne používajú aj v iných jazykoch: <, <=, ==, >=, >, !=, ~= a <> (všetky tri posledné operátory majú rovnaký význam *nerovná sa*).

6.2. Operátory

Operátory (alebo tiež riadiace príkazy) umožňujú ovplyvňovať priebeh programov, riadiť ich činnosť. Riadiace príkazy väčšinou začínajú odpovedajúcim *kľúčovým slovom*, ďalej obyčajne obsahujú nejaké postupnosti príkazov a končia kľúčovým slovom `end`. Na rozdiel od MATLABu umožňuje Octave použiť na ukončenie *odpovedajúce* kľúčové slovo, napríklad príkaz cyklu `while` je možné ukončiť kľúčovým slovom `endwhile`. Tento spôsob,

ktorý sa používa, napríklad, vo FORTRANe, spolu s odsadzovaním vnorených príkazov zvyšuje prehľadnosť programu. Vo FORTRANe sa zasa nemôže na ukončenie, povedzme, príkazu `if` použiť príkaz `end`. Preto by sa používateľ Octave mal rozhodnúť a používať jeden alebo druhý spôsob. Ak plánujeme písať programy použiteľné v MATLABe, mali by sme používať krátky variant `end`.

6.2.1. Logický operátor `if`

Logický operátor `if` môže mať, napríklad, nasledujúci tvar:

```
if (1. podmienka)
   operátory vykonávané pri splnení 1. podmienky
elseif (2. podmienka)
   operátory vykonávané pri nesplnení 1. a splnení 2. podm.
else
   operátory vykonávané pri nesplnení 1. a 2. podmienky
endif % koniec operatora
```

Počet podmienok môže byť aj väčší, v tom prípade sa opakuje kľúčové slovo `elseif`[30]. Na druhej strane môžeme časť `elseif`, ale aj `else` úplne vynechať. Takto dostaneme najjednoduchší príkaz `if`, ktorý sa použije len na

[30] Treba dávať pozor na rozdiel pri použití kľúčového slova `elseif` a použití dvoch kľúčových slov `else if`. V tomto druhom prípade sa začína *ďalší* príkaz `if` vo vnútri časti „`else`“.

vykonanie postupnosti príkazov v prípade splnenia podmienky 1. Všetky podmienky sú *logické výrazy*.

Operátor `if` môžeme použiť vo vnútri scenára (pozri nasledujúci oddiel) alebo funkcie, ale aj priamo v riadkovom príkazovom režime Octave – v jednom alebo vo viacerých riadkoch:

```
Vyrazy:27> x=137;
Vyrazy:28> if (mod(x,3)==0)
  disp(["Cislo " num2str(x) " je delitelne tromi!"]);
elseif (mod(x,3)==1)
  disp(["Cislo " num2str(x) " pri deleni tromi da zvysok 1!"]);
else
  disp(["Cislo " num2str(x) " pri deleni tromi da zvysok 2!"]);
end
Cislo 137 pri deleni tromi da zvysok 2!
```

6.2.2. Operátor vetvenia `switch`

```
Vyrazy:29> switch mod(x,3)
case 0
  disp(["Cislo " num2str(x) " je delitelne tromi!"]);
case 1
  disp(["Cislo " num2str(x) " pri deleni tromi da zvysok 1!"]);
otherwise
  disp(["Cislo " num2str(x) " pri deleni tromi da zvysok 2!"]);
endswitch
Cislo 137 pri deleni tromi da zvysok 2!
```

Namiesto predchádzajúceho príkazu `if` sme v tomto príklade použili príkaz `switch`. Podrobnejší komentár vynechávame, zdá sa nám, že v tomto prípade nie je nutný.

6.2.3. Operátor cyklu `for`

V nasledujúcom príklade ilustrujeme syntax príkazu `for`. V riadku 30 sa dá vcelku jasne pochopiť,[31] že premenná `k` má nadobúdať prirodzené hodnoty od 1 do 3. Podľa príkazu vo vnútri cyklu vidíme, že je potrebné vypočítať súčet prvých troch *nepárnych* kladných čísel. Príkaz môžeme zjednodušiť, ak použijeme zadanie *intervalu s krokom 2*, čím budeme automaticky prechádzať po *nepárnych* hodnotách `k`. Ak si uvedomíme, že

[31] Spomeňte si na pojem *intervalu* popísaný v oddiele 3.2.3.

interval je vlastne matica, tak vzhľadom na komutatívnosť súčtu môžeme použiť príkaz v tvare, zapísanom v riadku 32. Cyklus sa vlastne vykoná *po vymenovaných* prvkoch – tento typ cyklu nie je obvyklý v programovacích jazykoch (výnimkami sú napríklad MATLAB alebo Python):

```
Vyrazy:30> sum=0; for k=1:3, sum = sum + (2*k-1); end; sum
sum = 9
Vyrazy:31> sum=0; for k=1:2:5, sum = sum + k; end; sum
sum = 9
Vyrazy:32> sum=0; for k=[3 1 5], sum = sum + k; end; sum
sum = 9
```

Ďalším zaujímavým spôsobom použitia je cyklus *po celej štruktúre*:

```
Vyrazy:33> rod.otec="Jano";rod.pocet=5;rod.meno='Busa'
rod =
{
  otec = Jano
  pocet = 5
  meno = Busa
}
Vyrazy:34> for [val,key] = rod
val
end;
val = Jano
val = 5
val = Busa
```

Slovíčko `key` v zadaní `[val,key]` je možné vynechať.

6.2.4. Operátor cyklu `while`

Keďže cyklus typu `while` (*pokiaľ*) je dostatočne známy, uvedieme len jeden príklad:

```
Vyrazy:35> x=y=7.6;k=-1;while(y>=0),y--;k++;end; [k,floor(x)]
ans =
        7        7
```

V prvom príkaze sme využili viacnásobné priradenie, aby sme si „nepo-

kazili“ hodnotu premennej `x`. Čiarka za podmienkou nie je povinná. *Telo cyklu*, dva vnútorné príkazy, sa vykonáva, pokiaľ *je splnená podmienka* `y>=0`. Keďže sa kladná hodnota `y` v každom kroku zmenšuje, *po konečnom počte krokov* nastane prípad, keď podmienka splnená nebude – v tom momente sa cyklus ukončí. „Program“ určuje *celú časť* čísla `x`, výsledok môžeme porovnať s hodnotou funkcie `floor` (*podlaha*).

6.2.5. Operátor cyklu `do-until`

Rovnakú úlohu môžeme vyriešiť aj použitím cyklu typu `do-until`, ktorý sa výnimočne *nekončí* príkazom `end`:

```
Vyrazy:36> x=y=7.6;k=-1;do y--;k++; until(y<0);[k,floor(x)]
ans =
        7        7
```

Vidíme, že podmienka za kľúčovým slovom `until` je opačná, ako podmienka za príkazom `while`. V tomto type cyklu sa *príkazy tela cyklu* vykonávajú, *kým nie je splnená podmienka*. Navyše, test podmienky sa vykonáva až na konci, teda príkazy vo vnútri cyklu sa vykonajú aspoň raz.

6.2.6. Príkazy `break` a `continue`

Príkaz `break` slúži na ukončenie *najvnútornejšieho* cyklu, v ktorom je príkaz `break` použitý. Pozor! Zmyslovo podobný príkaz `exit` ukončí nielen cyklus, ale aj celý program Octave so všetkými dôsledkami!

Príkaz `continue` donúti Octave preskočiť *na koniec tela cyklu* a ak je to potrebné, znova sa začnú vykonávať príkazy od prvého príkazu tela cyklu.

Obidva príkazy je možné použiť napríklad vo vnútri nekonečného cyklu `while 1, ... end;`. Príkaz `break` ho umožní ukončiť, teda opustiť.

Záujemcovia o popis príkazov/operátorov `unwind_protect` a `try` si môžu naštudovať ich použitie v príručke Octave (EATON, 1997).

7. Scenáre a funkcie

7.1. Scenáre

Za *scenár* budeme ďalej považovať súbor, ktorý obsahuje postupnosť príkazov a volaní funkcií programu Octave. Používa sa tiež *skript* (z anglického *script*), nám sa zdá, že scenár lepšie odpovedá funkcii daného súboru – popísať činnosti, ktoré sa majú postupne vykonať. Scenár má obvykle príponu `m`, podobne ako v MATLABe môžeme hovoriť aj o `m`-súboroch. Pri inom rozšírení sa scenár musí vyvolať príkazom `source(názov)`, preto odporúčame dávať vždy príponu `m`. Okrem príkazov a volaní funkcií môže (a mal by) scenár obsahovať komentáre, za ktoré Octave považuje text od znaku komentára `%` do konca riadku. Činnosti, definované v scenári, sa vykonajú zadaním názvu scenára.

Scenáre použijeme obvykle v prípadoch, keď chceme zaznamenať nejakú postupnosť príkazov a volaní funkcií a použitie *histórie* nie je najvhodnejšie – napríklad, ak sa jedná o dlhú postupnosť. Scenár sa použitím podobá na (pod)program napísaný v inom jazyku, napríklad vo FORTRANe alebo v C++. Tam sa však programy kompilujú. Octave pri načítaní scenára postupne *interpretuje* jednotlivé príkazy. Aj preto sa nehodí na náročné výpočty, ktoré po skompilovaní prebiehajú mnohonásobne rýchlejšie.

Nižšie uvidíme, že aj funkcie sa píšu do súborov s príponou `m`. Rozdiel spočíva v tom, že funkcie sa začínajú deklaráciou `function` v prvom vykonávanom riadku. Napriek tomu môžu aj scenáre obsahovať definície (aj) viacerých funkcií. Komentár, ktorý sa nachádza *pred prvým vykonávateľným*

príkazom sa rovnako ako v MATLABe zobrazuje pri zadaní príkazu `help m-súbor`.

Napríklad, ak súbor `scenar.m` obsahuje text:

```
% Priklad zadania matice a vypoctu determinantu.
%    1. a=[1,3;5,7]      2. d=det(a)
a=[-1,2,-3;2,-3,4;-3,4,-5] % zadanie matice
d=det(a) % výpocet determinantu
```

tak po zadaní príkazu `help scenar` dostaneme[32]:

```
>> help scenar
scenar is the file: /cygdrive/d/Users/Busa/Octave/m/scenar.m

Priklad zadania matice a vypoctu determinantu.
   1. a=[1,3;5,7]      2. d=det(a)

Additional help for built-in functions, operators, and
variables is available in the on-line version of the manual.
Use the command 'help -i <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at http://www.octave.org and via the
help-octave@bevo.che.wisc.edu mailing list.
```

[32] Ak je `scenar.m` v pracovnom priečinku alebo v priečinku, ktorý je súčasťou `LOADPATH`!

A po *vyvolaní* scenára jeho názvom/menom v okne Octave uvidíme:

```
>> scenar
a =
        -1         2        -3
         2        -3         4
        -3         4        -5
d =  7.4051e-17
```

Ďalšie komentáre sa už (samozrejme) v okne Octave nezobrazujú. Príkaz `help` nám teda umožňuje rýchlo a jednoducho si spomenúť na obsah scenára bez toho, aby sme ho otvárali v editore.

V scenári môžeme zadávať matice aj prehľadnejšie (po riadkoch). Nový riadok v scenári sa vyhodnotí ako nový riadok matice. Nie je ani potrebné, ale je možné, písať bodkočiarku končiacu riadok matice. Tri bodky `...` označujú predĺženie zadávaného riadku. Nasledujúca časť scenára:

```
a=[-1,2,-3
    2 -3 4
   -3, 4,-5]

v = ...
cos(pi) * ...
7+11
```

sa teda vyhodnotí ako:

```
>> scenar
a =
        -1         2        -3
         2        -3         4
        -3         4        -5

v = 4
```

Premenné, ktoré používame vo volanom scenári *nie sú lokálne*, na rozdiel od premenných vo vnútri funkcií, ale sú *viditeľné* rovnako ako premenné v okne Octave. Premenné je možné deklarovať aj ako *globálne* (pred prvým použitím) príkazom

```
global x Y z;
```

V tom prípade je možné ich použiť aj vo vnútri funkcií, kde však musia byť tiež deklarované ako globálne.

7.2. Funkcie

Ako v každom programovacom jazyku, aj v Octave hrajú funkcie veľmi dôležitú úlohu. V Octave poznáme

- systémové funkcie, medzi ktoré patria, napríklad, aj bežne používané funkcie kosínus, logaritmus a pod.,
- funkcie definované užívateľom v `m`-súboroch,

- dynamicky pripájané (tým sa nebudeme venovať).

7.2.1. Systémové funkcie

Zoznam systémových funkcií uvádzame ďalej v oddiele P.1 na strane 152. Ich použitie je bežné, ako sme, napríklad, uviedli v predchádzajúcom príklade. Na tomto mieste uvedieme snáď len dve upozornenia:

1. Funkcie sú *maticové*, pričom pracujú po zložkách – ak je vstupným argumentom matica, na výstupe dostaneme maticu rovnakého typu, ktorej prvky sú funkčné hodnoty odpovedajúcich prvkov vstupnej matice.

2. Funkcia `log` je funkcia *prirodzeného* logaritmu „ln“, na zadanie *dekadického* logaritmu je potrebné použiť funkciu `log10`,[33] funkcia `log2` počíta *binárny* logaritmus pri základe 2.

7.2.2. Funkcie definované užívateľom

Užívatelia môžu sami vytvárať vlastné funkcie podľa potreby. Ak neberieme do úvahy funkcie pripájané dynamicky, možné sú v podstate 3 spôsoby definovania funkcií:[34]

1. Definícia funkcie priamo v aktívnom okne Octave.

[33] Tak je to aj v jazyku C++.
[34] V MATLABe nie je možné použiť 1. spôsob.

2. Definícia funkcie vo vnútri nejakého scenára.

3. Definícia funkcie v samostatnom súbore.

Zadefinujme, napríklad, funkciu 4. odmocniny (klávesom ENTER ukončíme jednotlivé riadky):

```
>> function f=stvrta_odmoc(x)
f=sqrt(sqrt(x));
endfunction
```

alebo jednoduchšie v jednom riadku:

```
>> function f=stvrta_odmoc(x) f=sqrt(sqrt(x)); endfunction
```

Ľahko sa presvedčíme, že Octave pozná „našu“ funkciu:

```
>> whos
*** dynamically linked functions:
prot  type                      rows   cols  name
====  ====                      ====   ====  ====
 r--  dynamically-linked function   -      -  dispatch
*** currently compiled functions:
prot  type                      rows   cols  name
====  ====                      ====   ====  ====
 rwd  user-defined function        -      -  stvrta_odmoc
```

Môžeme ju vyskúšať:

```
>> odm=stvrta_odmocnina([4 -16 1+i])
odm =
  1.41421 + 0.00000i  1.41421 + 1.41421i  1.06955 + 0.21275i
>> odm.^4
ans =
  1.0e+01  *
   0.40000 + 0.00000i  -1.60000 + 0.00000i   0.10000 + 0.10000i
>> clear all
>> o=stvrta_odmocnina([4 -16 1+i])
error: 'stvrta_odmocnina' undefined near line 49 column 3
error: evaluating assignment expression near line 49, column 2
```

Nevýhodou takéhoto postupu je, že sa definícia funkcie stratí po vypnutí Octave ale aj po použití príkazu `clear all`.

Rovnako definujeme funkcie aj v scenároch a v samostatných `m`-súboroch. Aby platila definícia funkcie definovanej v scenári, je potrebné najprv scenár „vyvolať". Táto možnosť je iste praktickejšia ako definícia v okne programu Octave, jej použitie je vhodné napríklad vtedy, ak chceme v jednom súbore zadefinovať skupinu funkcií, ktoré chceme používať. V tom prípade nesmieme zabudnúť, že máme funkcie uložiť *do scenára*, preto pred prvou deklaráciou `function` je potrebné vykonať nejaký príkaz, napríklad, stačí zadať `1;`.

Pri zadaní funkcie je potrebné dodržať nasledujúcu syntax (v prípade zadania funkcie v samostatnom `m`-súbore je možné (aj vhodné) vynechať kľúčové slovo `endfunction` – toto slovo nepozná MATLAB, ak je vyne-

chané, je možné ten istý `m`-súbor použiť aj v Octave aj v MATLABe):

```
function [vystup]=nazov(vstup)
   telo funkcie
endfunction
```

Počet výstupných ale aj vstupných argumentov môže byť väčší ako jeden (dokonca sa pripúšťa volanie tej istej funkcie s rôznym počtom argumentov)! Ak je výstupný argument jeden, nie je potrebné písať hranaté zátvorky.

Hneď po zavolaní funkcie sú dvom systémovým premenným `nargin` (počet argumentov vstupu) a `nargout` (počet argumentov na výstupe) priradené odpovedajúce hodnoty. Tieto môžeme použiť na rozvetvenie tela funkcie/programu.

Telo funkcie obsahuje postupnosť príkazov alebo volaní funkcií.

Názov funkcie (rovnako ako názov premennej) môže pozostávať z písmen, cifier a *podčiarkovníka*, nesmie sa však začínať cifrou! Volanie funkcie sa uskutočňuje v príkazovom riadku Octave alebo v scenári zadaním jej názvu a zoznamu vstupných argumentov v zátvorkách. Za jej názov sa pokladá *názov `m`-súboru* aj v prípade, že vyššie uvedený `nazov` sa s ním nezhoduje! Teda názov funkcie v `m`-súbore je v skutočnosti nepodstatný! Samozrejme, *odporúčame zadávať funkciám názvy zhodné s názvami súborov*.

V tomto príklade *sme zabudli* premenovať súbor a nechali sme starý názov `scenar.m`. Pozrime sa ako dopadne volanie funkcie `scenar` s počtom argumentov 0–3:

```
function [a,b]=komplex(z)
   if (nargout==1)
      a=abs(z);
   elseif (nargout==2)
      a=real(z);
      b=imag(z);
   else
      disp("Kolko vystupov?");
   end;
```

```
>> scenar(1+2i)
Kolko vystupov?
ans = []
>> a=scenar(1+2i)
a = 2.2361
>> [a,b]=scenar(1+2i)
a = 1
b = 2
>> [a,b,c]=scenar(1+2i)
Kolko vystupov?
a = []
b = []
error: element number 3 undefined in return list
error: evaluating assignment expression near line 5, column 8
```

Ak zmeníme definíciu funkcie s použitím premennej `varargout` typu poľa buniek, bude výsledok prijateľnejší:

```
function [varargout]=komplex(z)
   if (nargout==1)
      varargout=cell(1); varargout(1)=abs(z);
   elseif (nargout==2)
      varargout=cell(1,2);
      varargout(1)=real(z); varargout(2)=imag(z);
   else
      disp("Kolko vystupov?");
   end;
```

```
>> scenar(1+2i)
Kolko vystupov?
>> a=scenar(1+2i)
a = 2.2361
>> [a,b]=scenar(1+2i)
a = 1
b = 2
>> [a,b,c]=scenar(1+2i)
Kolko vystupov?
error: value on right hand side of assignment is undefined
error: evaluating assignment expression near line 18, col. 8
```

Podobný príklad s premenlivým počtom vstupných argumentov.

```
function [varargout]=rovnica(varargin)
   if (nargin==2)
      a=nth(varargin(1),1); b=nth(varargin(2),1);
      disp(["Riesime linearnu rovnicu " num2str(a) ...
            "x+" num2str(b) "=0"]);
      varargout=cell(1); varargout(1)=-b/a;
   elseif (nargin==3)
      a=nth(varargin(1),1); b=nth(varargin(2),1);
      c=nth(varargin(3),1); d=sqrt(b^2-4*a*c);
      disp(["Riesime kvadraticku rovnicu " num2str(a) ...
            "x^2+" num2str(b) "x+" num2str(c) "=0"]);
      varargout=cell(1,2);
      [varargout(1)=(-b-d)/(2*a);varargout(2)=(-b+d)/(2*a)];
   end;
```

```
>> [x]=rovnica(3,2)
Riesime linearnu rovnicu 3x+2=0
x = -0.66667
>> [x,y]=rovnica(3,2,1)
Riesime kvadraticku rovnicu 3x^2+2x+1=0
x = -0.33333 - 0.47140i
y = -0.33333 + 0.47140i
```

Ak Octave narazí na príkaz `return`, ukončí činnosť vo vnútri funkcie alebo scenára.

Na upozornenie na nejakú zvláštnu situáciu v priebehu vykonávania programu môžeme použiť príkaz `warning`. Podobne príkaz `error` (*chyba*) použijeme, ak zistíme, že nastala nejaká nekorektná situácia:

```
>> warning("Diskriminant je zaporny!");
warning: Diskriminant je zaporny!
>> error("Delenie nulou!");
error: Delenie nulou!
```

7.2.3. Dynamicky pripájané funkcie

Na niektorých systémoch dokáže Octave zavádzať a vykonávať funkcie napísané v jazyku C++. Je možné použiť aj funkcie napísané v iných jazykoch, ak sú volané zo zvláštnej funkcie C++. Zadaním `makeoctfile program.cc` sa vytvorí súbor `program.oct`, ktorý je možné zaradiť do štruktúry Octave a volať zadaním názvu.

Vzhľadom na to, že sa jedná o úzko špecializovaný problém, nebudeme ho tu popisovať. Podrobný návod na použitie funkcií C++ nájdete v príručke Octave (EATON, 1997).

7.3. Rozmiestnenie funkcií a scenárov v distribúcii Octave

V popisovanej distribúcii programu Octave sme našli 80 `.oct` súborov a 1241 `.m` súborov.

Preto len uvedieme názvy priečinkov, kde sa uvedené súbory nachádzajú. Veríme, že zvedaví čitatelia nahliadnu aj dovnútra týchto priečinkov. Nezabudnite, že príkazom `help súbor` zistíte mnohé zaujímavé informácie.

38 súborov s koncovkou `.oct` sa nachádza v priečinku:

```
\libexec\octave\2.1.50\site\oct\i686-pc-cygwin\octave-forge
```

zvyšných 42 sa nachádza v priečinku

```
\octave\libexec\octave\2.1.50\oct\i686-pc-cygwin
```

Užívateľské funkcie a scenáre sa nachádzajú v priečinkoch [35]:

```
Directory of C:\Program Files\GNU Octave 2.1.50\opt
\octave\share\octave\2.1.50\m
...
audio  control  deprecated  elfun  finance  general  image
io  linear-algebra  miscellaneous  plot  polynomial
quaternion  set  signal  specfun  special-matrix  startup
statistics  strings  time
```

[35] Vo verzii 2.1.73 v priečinkoch: `\GNU Octave 2.1.73\usr\share\octave\2.1.73\m` a `\GNU Octave 2.1.73\usr\share\octave\site\m\octave-forge` (navyše sú aj nové).

```
Directory of
C:\Program Files\GNU Octave 2.1.50\opt
\octave\share\octave\2.1.50\site\m\octave-forge
...
audio  civil  comm  control  FIXES  general  geometry  ident
image  integration  io  linear-algebra  miscellaneous  NaN
ode  optim  path  pdb  plot  set  signal  sparse  specfun
special-matrix  splines  statistics  strings  struct  symband
symbolic  testfun  time  tsa  vrml  Windows
```

8. Vstup a výstup údajov

Doteraz sme pracovali so vstupnými údajmi zadávanými buď z klávesnice alebo v nejakom scenári. Všetky výstupy boli robené na obrazovke (terminál). V tejto kapitole sa budeme venovať ďalším možnostiam, podrobne je táto problematika rozobraná v Eatonovej príručke (1997).

8.1. Vstup údajov

8.1.1. Príkazy `input` a `menu`

Pri spúšťaní programov potrebujeme často načítať údaje, ktoré sa menia. Jeden zo spôsobov poskytuje príkaz `input`. Po jeho zadaní sa objaví text – výzva, ktorý je jedným z parametrov tohoto príkazu. Po zadaní a potvrdení klávesom ENTER sa zadané údaje priradia premennej na ľavej strane príkazu:

```
I/O:1> a=input("Zadaj maticu a: ");
Zadaj maticu a: [1,2;-3 4]
I/O:2> a
a =
         1         2
        -3         4
```

Príkaz `menu` obsahuje okrem výzvy aj niekoľko volieb (ich počet volíme podľa potreby). Z klávesnice potom užívateľ zvolí vhodnú voľbu a jej

poradové číslo bude priradené premennej z ľavej strany príkazu:

```
I/O:3> k=menu("Diskriminant? ",'kladny','nulovy','zaporny');
Diskriminant?
  [ 1] kladny
  [ 2] nulovy
  [ 3] zaporny
pick a number, any number: 3
I/O:4> k
k = 3
```

Je zrejmé, že rovnaký výsledok ako príkazom `menu` sme schopní dosiahnuť príkazom `input`. Rozdiel je v tom, že funkcia `menu` celý postup zjednodušuje.

8.1.2. Načítanie údajov zo súboru príkazom `load`

Veľmi užitočná je možnosť načítania údajov zo súboru. Tieto môžu byť vytvorené v nejakom editore alebo môžu byť vytvorené ako výstup nejakého programu.

Na tvorbu grafov je najpraktickejšie uchovávať číselné údaje v tvare tabuliek, kde jednotlivé stĺpce odpovedajú jednotlivým premenným. Majme v súbore `b.dat` uloženú nasledujúcu tabuľku údajov:

```
% Toto je matica
1 -2.e-2 3.4
# Este riadok
-5 62.4e-1 7
```

Ako vidieť, v druhom aj vo štvrtom riadku sú tri čísla (v rôznych formátoch) oddelené medzerou. Prvý riadok a tretí riadok, ktoré sa začínajú znakmi % alebo # sú považované za komentáre a pri načítavaní sú ignorované. Teda máme zadanú tabuľku s tromi stĺpcami a dvomi riadkami čísel. Na jej načítanie do programu Octave použijeme príkaz `load`:

```
I/O:5> clear all
I/O:6> load b.dat;
I/O:7> b
b =
   1.00000  -0.02000   3.40000
  -5.00000   6.24000   7.00000
```

Príkazom v riadku 5 sme vymazali všetky premenné. Vidíme, že výsledkom použitia príkazu `load` je premenná `b`, ktorá má rovnaký názov, ako má súbor, z ktorého sme údaje načítali. Premenná `b` obsahuje v dvoch riadkoch a troch stĺpcoch zadané údaje, zapísané už v číselnom formáte Octave. Ak by sme jednotlivé údaje oddelili čiarkami[36] (s medzerami alebo bez), výsledok by bol rovnaký.

[36] Ale nie bodkočiarkami, tie však môžu byť na konci riadku!

8.1.3. Načítanie formátovaných údajov zo súboru

Ďalšou možnosťou je použitie príkazu `fscanf` na načítanie formátovaných údajov, známeho z jazyka C. Predpokladajme, že v súbore `c.dat` máme zapísané:

```
x(1)=23
x(2)=-34
x(11)=12.e-3
```

Predpokladajme ďalej, že nás zaujíma len vektor `x`. Potom jeho hodnoty získame po sérii nasledujúcich príkazov:

```
I/O:8> c=fopen("c.dat","r");  % otvorenie súboru na čítanie
I/O:9> for k=1:3,
             [s,ind,s,x(ind)]=fscanf(c,"%2c%i%2c%f\n","C");
       end;
I/O:10> fclose(c); % uzavretie suboru
I/O:11> x
x =
  1.0e+01  *
 Columns 1 through 6:
   2.30000  -3.40000   0.00000   0.00000   0.00000   0.00000
 Columns 7 through 11:
   0.00000   0.00000   0.00000   0.00000   0.00120
```

V tomto prípade sa jedná o zjednodušený príklad, keďže v každom riadku

máme na začiatku najprv dva znaky (písmená), potom celé číslo, ďalej znova dva znaky a nakoniec reálne číslo.

Situácia bola navyše zjednodušená tým, že sme poznali počet vstupných riadkov. V riadku 9 sme vo vnútri príkazu `fscanf` ako 2. parameter zadali formát údajov. Na ľavej strane sme zabezpečili, že načítaný index `ind` bol hneď použitý ako index výstupného vektora `t`. Keďže načítané reťazce nás nezaujímali, načítavali sme všetko do rovnakého reťazca `s`. Zadanie `\n` na konci znamená prechod na ďalší riadok vstupného súboru. Príkazom `frewind(c)` sa môžeme znova nastaviť na začiatok súboru, ak by sme potrebovali opätovne načítavať údaje.

Programátori so skúsenosťami s programovacím jazykom C si iste prídu na svoje pri používaní formátovaného vstupu, začiatočníkom však môžeme odporučiť používanie súborov s jednoduchými číselnými tabuľkami.

Na načítavanie údajov zo súborov je možné použiť tiež príkazy `fgetl`, `fgets`, `fread` (na načítavanie binárnych údajov rôznych typov). Príkaz `sscanf` je podobný ako príkaz `fscanf` s tým rozdielom, že sa nenačítava zo súboru, ale z reťazca.

8.2. Zápis údajov do súborov

8.2.1. Zápis údajov do súboru príkazom `save`

Podobne ako v prípade načítavania, aj v prípade zápisu je najjednoduchší spôsob uloženia číselných údajov do datového súboru v tvare tabuľky

(aby sa dali opätovne ľahko načítavať). Ak má vektor veľký počet prvkov, je lepšie ho zapisovať do stĺpca:

```
I/O:12> x=pi.^(-10:30:50)'; save -ascii x.dat x;
```

Po tomto použití príkazu `save` bol v pracovnom priečinku vytvorený súbor `x.dat`[37] obsahujúci nasledujúce riadky (prvý sme skrátili):

```
# Created by Octave 2.1.50, Fri Jun 23 17:52:40 2006 <j@BU>
# name: x
# type: matrix
# rows: 3
# columns: 1
 1.06782792268615e-05
 8769956796.08269
 7.20267194471579e+24
```

Vidíme, že zapísané údaje sú uložené v rôznych formátoch tak, ako to Octave považuje za vhodné.

8.2.2. Zápis formátovaných údajov do súboru a na obrazovku

Podobne ako pri načítavaní údajov, máme možnosť vytvárať formátovaný výstup a zapisovať ho do súboru. Napríklad po zadaní nasledujúcich príkazov:

[37] Názov súboru sa nemusí zhodovať s názvom premennej! Môže a nemusí byť zadaný vnútri úvodzoviek alebo apostrofov.

```
I/O:13> cc=fopen("x.dat","a");
I/O:14> for k=1:3
              fprintf(cc,"x(%i)=%12.4e\n",k,x(k));
        end;
I/O:15> fclose(cc);
```

sme otvorili súbor `x.dat` s voľbou `a` (z anglického *append* – pridať), ktorá zariadi, že nasledujúce údaje sa budú zapisovať na koniec súboru. V cykle sme do súboru zapísali vektor `x` vo formáte `%12.4e` (exponenciálny zápis na 12 miest so štyrmi miestami za desatinnou bodkou), po každej zložke sme zabezpečili prechod na nový riadok špecifikáciou `\n`. Po uzavretí súboru príkazom `fclose(cc);` v súbore `x.dat` nachádzame:

```
# Created by Octave 2.1.50, Fri Jun 23 17:52:40 2006 <j@BU>
# name: x
# type: matrix
# rows: 3
# columns: 1
 1.06782792268615e-05
 8769956796.08269
 7.20267194471579e+24
x(1)=  1.0678e-05
x(2)=  8.7700e+09
x(3)=  7.2027e+24
```

V exponenciálnom formáte sa pred desatinnou bodkou píše prvá platná cifra. Nevyužité cifry boli doplnené medzerami hneď za znamienkom =.

Ďalšie voľby, ktoré môžeme zadať pri otváraní súboru nájdete v príručke Octave (EATON, 1997). Spomeňme ešte voľbu `"w"` (z anglického *write* – zapíš). Pri jej použití sa otvorí súbor pripravený na *zapisovanie*, pričom jeho pôvodný obsah sa vymaže. Popis formátov (konverzií) výstupov nájdete tiež v príručke Octave v oddiele 15.2 nazvanom „*C-Style I/O Functions*“.

Na zapisovanie údajov do súborov je možné použiť tiež príkazy `fputs` a `fwrite` (na zápis binárnych údajov rôznych typov). Príkaz `sprintf` je podobný ako príkaz `fprintf` s tým rozdielom, že sa nezapisuje do súboru, ale do reťazca.

Formátovaný výstup na štandardné zariadenie (napríklad obrazovku) získame pomocou príkazu `printf` podobne ako formátovaný zápis do súboru:

```
I/O:16> for k=1:3
            printf("x(%i)=%14.6e\n",k,x(k));
        end;
x(1)=  1.067828e-05
x(2)=  8.769957e+09
x(3)=  7.202672e+24
```

9. Grafický výstup Octave

V grafických aplikáciách Octave zaostáva za komerčným MATLABom. Ako sme už písali skôr, Octave na grafický výstup používa GNUPLOT.[38] Výsledkom spolupráce týchto dvoch programov môžu byť dostatočne zložité obrázky.

Pri zadaní grafického príkazu v Octave, napríklad `plot` alebo `gplot`, sa otvorí okno programu GNUPLOT. Grafické príkazy zadávané v okne Octave sa súčasne zobrazujú v okne GNUPLOTu, grafický výstup je možné ovplyvňovať aj priamo v okne GNUPLOTu.

Pri vytváraní grafov je možné nastavovať farbu čiar, ale aj iné atribúty. Na popis kriviek je možné využiť legendu, samozrejmosťou sú popisy osí, nadpisy grafov a rôzne textové značky. Podať všetky možnosti nie je možné v rámci stručného popisu. Podrobnosti nájdete v príručke Octave (EATON, 1997).

9.1. Príkazy `plot` a `gplot`

Grafy funkcií jednej premennej môžeme v Octave vytvárať príkazom `plot` (v MATLABovskom štýle) alebo `gplot` (v štýle GNUPLOTu). Príkaz `help plot` značne rozšíri prehľad používateľa. Časťou výpisu je napríklad informácia o farbách a druhoch čiar:

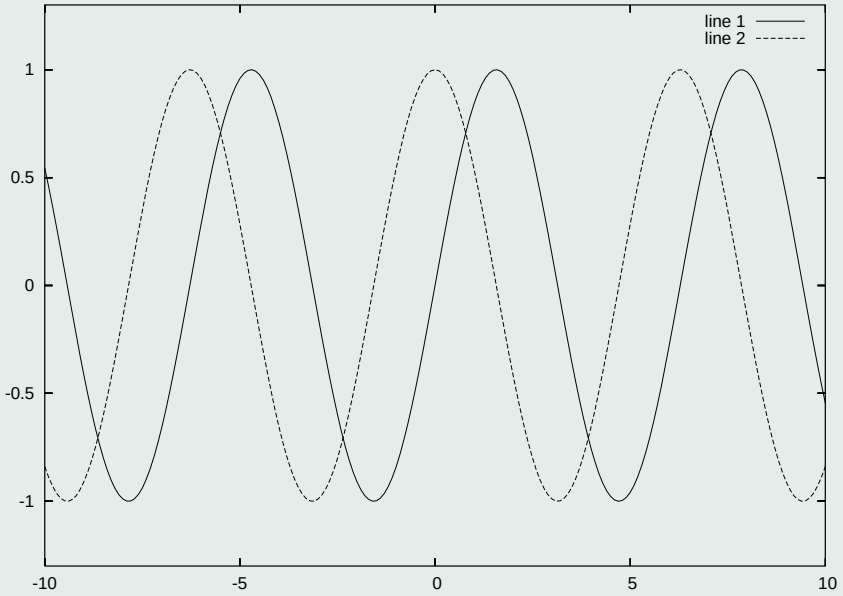
[38] Je tiež možné vytvoriť datové súbory v Octave a neskôr ich použiť v GNUPLOTe.

```
Grafy:1> help plot
plot is the user-defined function from the file
/opt/octave/share/octave/2.1.50/m/plot/plot.m
...
          Number  Gnuplot colors  (lines)points style
            1       red                   *
            2       green                 +
            3       blue                  o
            4       magenta               x
            5       cyan                house
            6       brown            there exists
...
Grafy:2>  t=0:0.1:6.3;
Grafy:3>  plot(t,cos(t),"-4;cos(t);",t,sin(t),"+3;sin(t);");
```

Príkaz v riadku 3 vykreslí graf funkcie kosínus farbou „magenta“ neprerušovanou čiarou, navyše v legende grafu bude pri 1. čiare názov $\cos(t)$. Podobne, graf funkcie sínus bude zobrazený modrými plusmi, pri 2. čiare bude názov $\sin(t)$.

V nasledujúcich riadkoch je najprv definovaný *interval* – vektor `x`, ďalej sú zobrazené dve funkcie – sínus a kosínus na danom intervale. Oblasť zobrazenia funkcií je (na „skrášlenie“) definovaná príkazom `axis([xmin xmax ymin ymax])` a graf je v GNUPLOTe znova prekreslený. Keď sme spokojní so zobrazením na obrazovke, príkazmi na riadku 6 nastavíme výstup na `postscript` a zadáme názov výstupného súboru:

```
Grafy:4> x=(-10:0.1:10)'; plot(x,sin(x),x,cos(x));
Grafy:5> axis([-10 10 -1.3 1.3]); replot;
Grafy:6> gset terminal postscript;gset output "sc.ps";replot
```



Vpravo hore vidíme zobrazenú jednoduchú legendu. Vyššie sme popísali, ako sa môže nahradiť line 1 iným textom. Ak chceme legendu zrušiť, zadáme príkazy `legend("off")`[39] a `replot`. Podrobnejšie informácie nájdete

[39] V OS Windows v okne GNUPLOTu môžeme zadať príkaz `unset key`, v Linuxe skú-

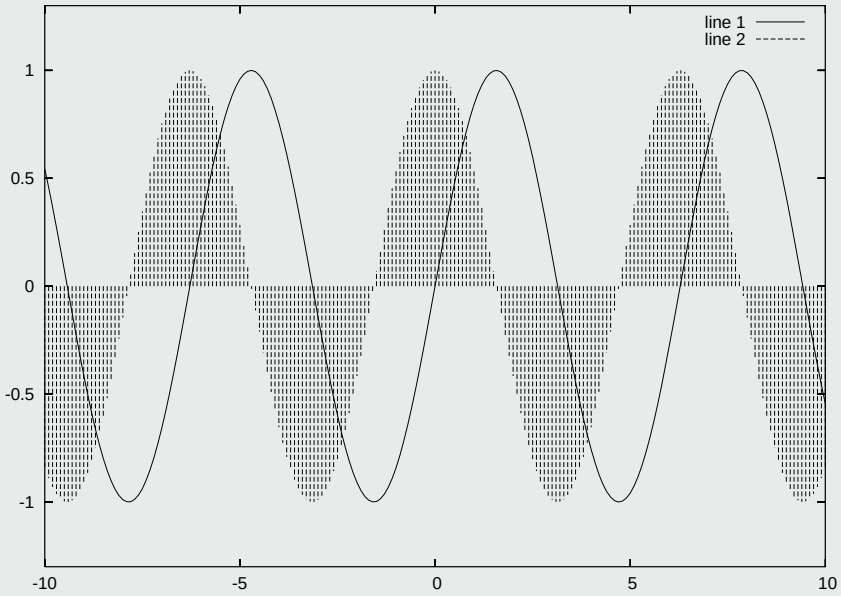
v príručke GNUPLOTu (DOBOŠ, 2006).

Aby sme mohli vytvoriť nový graf, príkazom `closepath` najprv uzavrieme GNUPLOT (riadok 7). Porovnajte syntax príkazu `plot` so syntaxou príkazu `gplot`, použitého v riadkoch 8–10:

```
Grafy:7> closeplot
Grafy:8> data=[x,sin(x),cos(x)];
Grafy:9> gplot [-10:10] [-1.3:1.3] \
          data with lines, data using 1:3 with impulses;
Grafy:10> gset terminal postscript; gset output "sc2.ps";replot
```

---

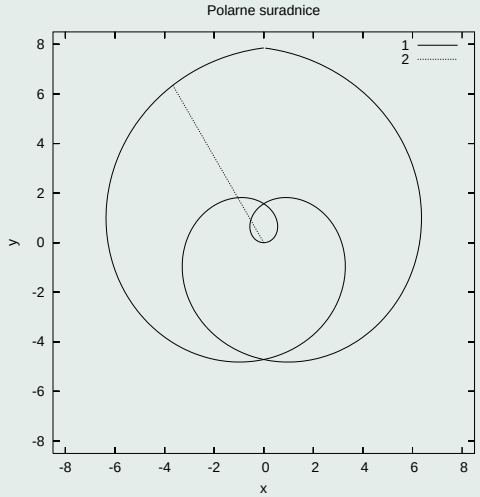
sime v okne Octave `__gnuplot_set__ unset key`.



9.2. Použitie polárnych súradníc

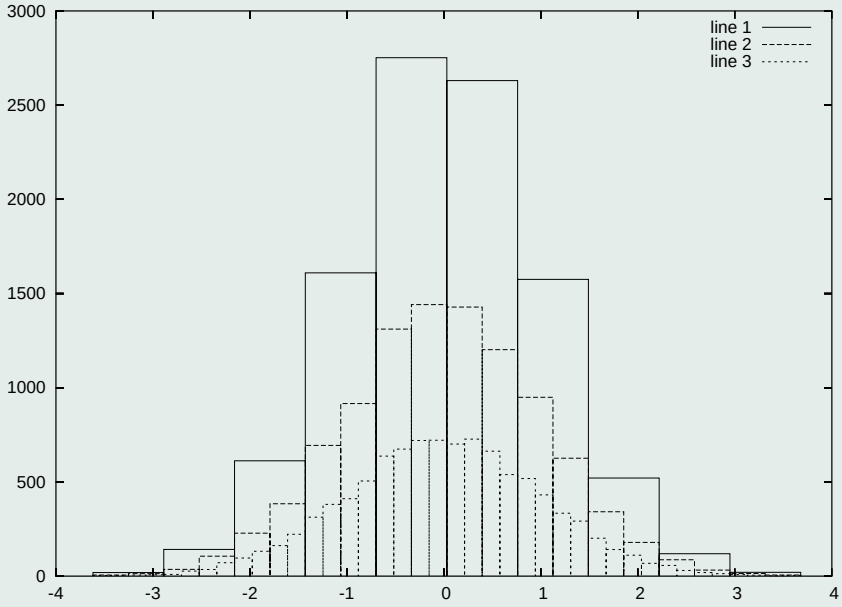
Nasledujúci obrázok bol vytvorený príkazmi z riadkov 11–16. Zamyslite sa nad nimi a predpokladaný výsledok porovnajte s obrázkom:

```
Grafy:11> phi=-5*pi/2:0.01:5*pi/2; polar(phi,rho,'r;1;');
Grafy:12> hold on; fi=-7*pi/3;
Grafy:13> plot([0 fi*cos(fi)],[0 fi*sin(fi)],'m;2;');
Grafy:14> xlabel("x"); ylabel("y"); title("Polarne suradnice");
Grafy:15> axis([-8.5 8.5 -8.5 8.5],"equal"); replot;
Grafy:16> gset terminal postscript; gset output "pr.ps"; replot
```



9.3. Tvorba histogramov príkazom `hist`

```
Grafy:17> x=randn(1,10000);  hist(x);hist(x,20);hist(x,40);
Grafy:18> gset terminal postscript eps
Grafy:19> gset output "hist.eps"
Grafy:20> replot
```



V riadku 17 sme vytvorili vektor 10000 náhodných čísel s normovaným

normálnym rozdelením a vytvorili sme histogramy pre 10 (predvolený počet), 20 a 40 tried. Histogramy svedčia o dobrej kvalite generátora pseudonáhodných čísel. V ďalších riadkoch sme nastavili výstup na „Encapsulated PostScript“, zadali názov výstupného súboru `hist.eps` a príkazom `replot` sme do neho uložili obrázok.

9.3.1. Príkazy `bar`, `stairs`, `loglog`, `semilogx` a `semilogy`

Príkaz `bar(x,y)` slúži na zobrazovanie stĺpcových diagramov. Podobne, príkazom `stairs(x,y)` zobrazíme závislosť y od x ako „schodovitú“, teda po častiach konštantnú funkciu.

Príkazy `loglog`, `semilogx` a `semilogy` slúžia na zobrazovanie grafov s logaritmickými škálami oboch alebo len jednotlivých osí.

9.4. Zobrazenie kriviek v trojrozmernom priestore

Na zobrazovanie kriviek v trojrozmernom priestore použijeme príkaz `plot3`:

```
Grafy:21> t=0:0.01:12*pi;
Grafy:22> z=t/(6*pi);
Grafy:23> x=sqrt(1-(z-1).^2).*cos(t);
Grafy:24> y=sqrt(1-(z-1).^2).*sin(t);
Grafy:25> plot3(x,y,z)
```

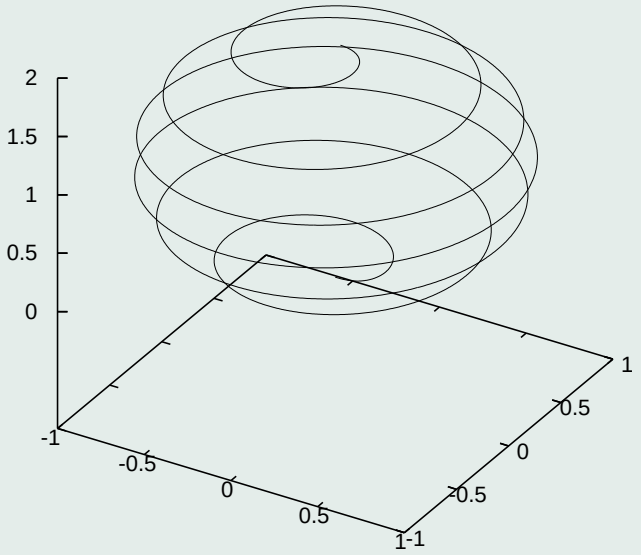
Najprv sme zadefinovali krivku pomocou parametrického zadania pre $t \in \langle 0, 12\pi \rangle$, $\boldsymbol{r}(t) = \big[x(t), y(t), z(t)\big]$. Všetky body krivky ležia na povrchu

jednotkovej gule, posunutej v smere osi z. Príkazom `plot3(x,y,z)` sme krivku zobrazili.

V programe GNUPLOT sme potom nastavili rozsah jednotlivých osí, vypli sme legendu, nastavili výstup na postscript, zadali sme názov výstupného súboru a nakoniec sme obrázok uložili do nastaveného súboru príkazom `rep` (skratka názvu `replot`):

```
gnuplot> set xrange [-1:1]
gnuplot> set yrange [-1:1]
gnuplot> set zrange [0:2]
gnuplot> set size ratio -1
gnuplot> unset key
gnuplot> set term postscript eps
Terminal type set to 'postscript'
Options are 'eps noenhanced monochrome blacktext \
   dashed dashlength 1.0 linewidth 1.0 defaultplex \
   palfuncparam 2000,0.003 \
   butt "Helvetica" 14'
gnuplot> set output "sphere.eps"
gnuplot> rep
```

Výsledok (šesť závitov „guľovej špirály“) je zobrazený na nasledujúcom obrázku:



9.5. Zobrazovanie grafov funkcií dvoch premenných

Na príklade známej Rosenbrockovej funkcie

$$z = f(x, y) = 100 \cdot (y - x^2)^2 + (1 - x)^2$$

ilustrujeme možnosti zobrazovania funkcií dvoch premenných. Najprv zadefinujeme oblasť premenných x a y, a po použití príkazu `meshgrid`, ktorý

vytvorí body dvojrozmernej sieťky, zadefinujeme samotnú funkciu v bodoch sieťky[40]:

```
Grafy:26> x=-2:0.1:2; y=0:0.1:2;
Grafy:27> [xx,yy]=meshgrid(x,y);
Grafy:28> z=100*(yy-xx.^2).^2+(1-xx).^2;
```

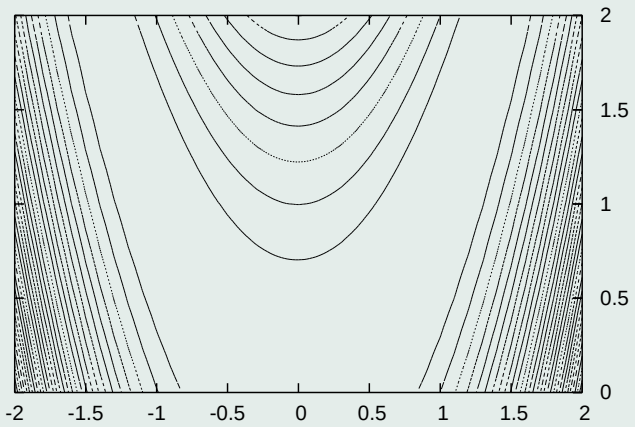
9.5.1. Vstevnicové grafy

Príkazom

```
Grafy:29> contour(x,y,z,40)
```

po vypnutí legendy v GNUPLOTe získame nasledujúci vrstevnicový graf (so 40 vrstevnicami):

[40] Na lepšie pochopenie si zobrazte pre „menšie“ vektory `x` a `y` výsledky `xx` a `yy` priradenia `[xx,yy]=meshgrid(x,y)`.

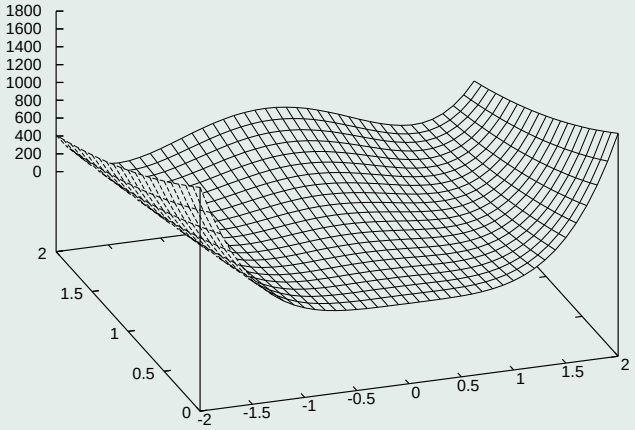


9.5.2. Grafy funkcií dvoch premenných v trojrozmernom priestore

Príkazom

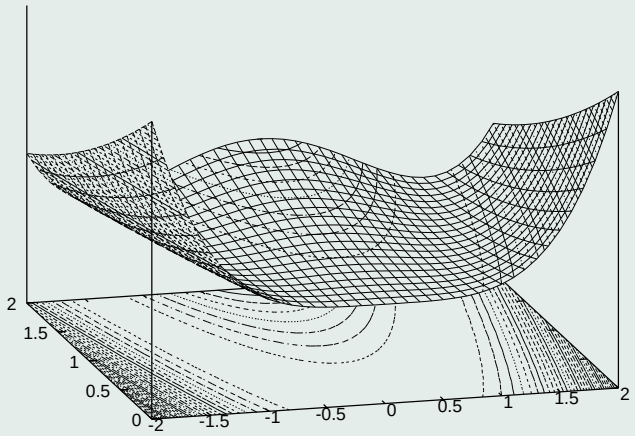
```
Grafy:30> mesh(x,y,z);
```

po vypnutí legendy získame zobrazenie plochy grafu Rosenbrockovej funkcie:



Na úrovni GNUPLOTu môžeme ešte trochu „čarovať“. Skúste porozmýšľať, čo bude výsledkom zadania nasledujúcich príkazov:

```
gnuplot> set cntrparam levels 40
gnuplot> set contour both
gnuplot> unset ztics
gnuplot> rep
```



Prvý príkaz nastaví počet úrovní vrstevnicového grafu. Ďalší príkaz pridá k zobrazeniu plochy grafu izolínie (dole aj na ploche, čo odpovedá voľbe `both`, ďalšie voľby sú `base` – izolínie len dole – a `surface` – vrstevnice na ploche). V treťom riadku sme zrušili výpis hodnôt na osi z.

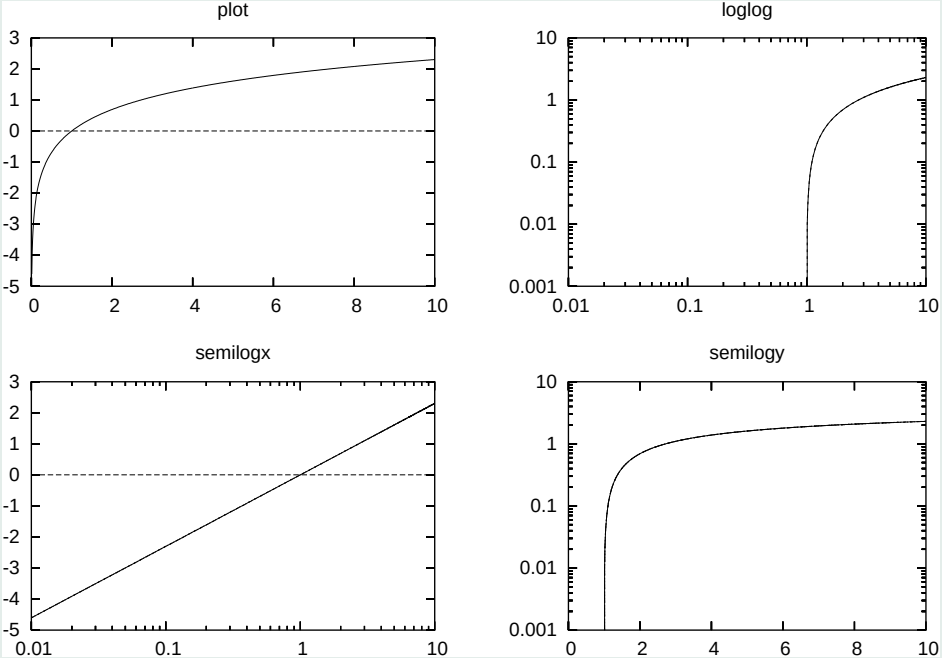
9.6. Uloženie viacerých obrázkov vedľa seba

V niektorých prípadoch je užitočné uložiť niekoľko obrázkov vedľa alebo pod seba. Octave spolu s GNUPLOTom umožňuje vytvoriť „maticu obrázkov“ použitím príkazu `multiplot`.

```
Grafy:31> x=0.01:0.01:10; multiplot(2,2);
```

Ak však chceme výsledok zapísať do súboru, musíme najprv „multiplot“ vypnúť v GNUPLOTe, potom nastaviť výstupné zariadenie a znova „multiplot“ zapnúť:

```
multiplot> unset multiplot
gnuplot> set term postscript; set output "mplot.ps"
gnuplot> unset key; set multiplot
```



Ďalej sme pokračovali znova v Octave zadaním grafov:

```
Grafy:32> subplot(2,2,1);
Grafy:33> title("plot");plot(x,log(x),[0 10],[0 0]);
Grafy:34> subplot(2,2,2); title("loglog");loglog(x,log(x));
Grafy:35> subplot(2,2,3); title("semilogx");semilogx(x,log(x));
Grafy:36> subplot(2,2,4); title("semilogy");semilogy(x,log(x));
```

V praxi je najlepšie si najprv pripraviť obrázky pri výstupe nastavenom na obrazovku. Keď sme s výsledkom spokojní, príkazom `closeplot` vypneme GNUPLOT a postup zopakujeme s výstupom nastaveným do súboru.

9.7. Záver

V stručnej príručke nie je možné podrobne vysvetliť prácu s grafikou Octave a GNUPLOTu. Ďalšie informácie získate štúdiom príručiek Octave (EATON, 1997) a GNUPLOTu (DOBOŠ, 2006), ale aj používaním príkazu `help` v obidvoch programoch.

Prajeme Vám v tejto tvorivej činnosti veľa úspechov! :)

10. Vybrané aplikácie

Aj keď Octave neponúka také rozmanité „toolboxy“ (nástroje) ako komerčný MATLAB, predsa môžeme využívať viaceré pripravené funkcie. Zoznam priečinkov obsahujúcich tieto funkcie sme uviedli v oddiele 7.3. Okrem toho je možné nájsť na internete veľký počet verejne dostupných `m`-súborov, ktoré sa dajú použiť aj v Octave.

Množstvom svojich zabudovaných funkcií poskytuje Octave viac možností, ako štandardné programovacie jazyky.

10.1. Štatistické funkcie

Octave poskytuje dobré možnosti nielen na výučbu základov štatistiky, ale aj na riešenie štandardných štatistických úloh. Pre oblasť štatistiky bol však vyvinutý špecializovaný OPENSOURCE program, ktorý sa nazýva „R“. V tomto oddiele uvedieme len stručný prehľad štatistických funkcií Octave, ich podrobnejší prehľad nájdete, napríklad, v príručke Octave (EATON, 1997).

Štatistické nástroje sú umiestnené v priečinkoch s názvom `statistics`:

Priečinok `...\opt\octave\share\octave\2.1.50\m\statistics` obsahuje podpriečinky `base`, `distributions`, `models` a `tests`.

Priečinok `...\octave\2.1.50\site\m\octave-forge\statistics` obsahuje funkcie:

```
boxplot, geomean, harmmean, mad, nanmax, nanmean, nanmedian,
```

`nanmin, nanstd, nansum, normplot, prctile, scatter, trimmean, zscore.`

O ich použití sa bližšie dozviete po naštudovaní návodu. Vyskúšajte, napríklad, zadať príkaz: `help boxplot` a ďalej zadajte postupnosť príkazov uvedenú v tomto návode.

10.1.1. Základné štatistické funkcie

Vyššie uvedený podpriečinok `base` obsahuje nasledujúce funkcie:

`center, cloglog, cor, corrcoef, cov, cut, gls, iqr, kendall, kurtosis, logit, mahalanobis, mean, meansq, median, moment, ols, ppplot, probit, qqplot, range, ranks, run_count, skewness, spearman, statistics, std, studentize, table, values, var.`

Väčšina týchto funkcií je dobre známa. Funckia `statistics(x)`, napríklad, vráti pre jednotlivé stĺpce matice (alebo vektora chápaného ako stĺpcový vektor) `x` nasledujúce hodnoty: miminum, prvú kvartilu, medián, tretiu kvartilu, maximum, strednú hodnotu, smerodajnú (štandardnú) odchýlku, šikmosť a strmosť (exces):

```
Stat:1> x=[1,3,2,2,4,3,2,1,5,4];
Stat:2> statistics(x)'
ans =
 Columns 1 through 6:
   1.00000   2.00000   2.00000   4.00000   5.00000   2.70000
 Columns 7 through 9:
   1.33749   0.24074  -1.40203
```

Funkcia `mean` vráti hodnotu aritmetického priemeru (strednú hodnotu) hodnôt stĺpcov matice (alebo vektora chápaného ako stĺpcový vektor) `x`, prípadne jej volaním s voľbami `"g"` alebo `"h"` získame geometrický alebo harmonický priemer :

```
Stat:3> [mean(x),mean(x,"a"),mean(x,'g'),mean(x,'h')]
ans =
  2.70000  2.70000  2.37707  2.05479
```

Funkcia `studentize(x)` vráti prenormované hodnoty vektora (alebo jednotlivých stĺpcov matice) `x`, $x_n = (x - \bar{x})/\sigma$, kde $\bar{x}$ je stredná hodnota a σ je smerodajná odchýlka:

```
Stat:4> xn=studentize(x)
xn =
 Columns 1 through 6:
  -1.27103   0.22430  -0.52337  -0.52337   0.97197   0.22430
 Columns 7 through 10:
  -0.52337  -1.27103   1.71963   0.97197
```

Korelačný koeficient dvoch výberov určíme nasledujúcim spôsobom:

```
Stat:5> vaha=[55,77,82,84]; vyska=[160,175,180,173];
Stat:6> corrcoef(vaha,vyska)
ans = 0.91236
```

Ďalšie podrobnosti nájdete v príručke Octave (EATON, 1997).

10.1.2. Štatistické funkcie rôznych rozdelení pravdepodobnosti

Podobne ako MATLAB, poskytuje Octave pre každé prístupné rozdelenie s názvom `nazov` 4 funkcie:

`nazov_cdf` – anglicky „cumulative density function“ – funkciu rozdelenia pravdepodobnosti (distribučnú funkciu);

`nazov_inv` – anglicky „inverse“ – inverznú funkciu k distribučnej funkcii;

`nazov_pdf` – anglicky „probability density function“ – funkciu hustoty rozdelenia pravdepodobnosti;

`nazov_rnd` – anglicky „random“ – funkciu, vytvárajúcu maticu pseudonáhodných hodnôt so zadaným rozdelením pravdepodobnosti.

Každá z uvedených funkcií potrebuje vstupné hodnoty a parametre v závislosti od typu rozdelenia, respektíve rozmere výstupnej matice.

K dispozícii sú nasledujúce názvy rozdelení:

`beta, binomial, cauchy, chisquare, discrete, empirical, exponential, f, gamma, geometric, hypergeometric, kolmogorov_smirnov`[41], `laplace, logistic, lognormal, normal, pascal, poisson, stdnormal, t, uniform, weibull, wiener`[42].

Ukážme použitie uvedených funkcií pre normálne rozdelenie. Známe „pravidlo 3σ“ tvrdí, že v intervale $\langle \mu - 3\sigma, \mu + 3\sigma \rangle$ sa nachádza viac ako 99 % vzoriek s normálnym rozdelením so strednou hodnotou μ a smerodajnou odchýlkou σ:

```
Stat:7> s=3.4; mu=2.8;
Stat:8> normal_cdf(mu+3*s,mu,s^2)-normal_cdf(mu-3*s,mu,s^2)
ans = 0.99730
Stat:9> normal_inv(0.995,0,1)
ans =   2.57583
```

Teda hodnote 99 % (zvyšuje po 0,5 % zľava i sprava) odpovedá v skutočnosti približne $2.58\,\sigma$. Vidíme tiež, že parametrami normálneho rozdelenia sú stredná hodnota μ a disperzia σ^2. Porovnajme ešte strednú hodnotu a smerodajnú odchýlku vektora 1000 pseudonáhodných čísel s normálnym rozdelením so strednou hodnotou $\mu = 3$ a s disperziou $\sigma^2 = 4$:

[41] Len funkcia `cdf`.
[42] Len funkcia `rnd`.

```
Stat:10> mean(normal_rnd(3,4,1000,1))
ans = 2.9530
Stat:11> std(normal_rnd(3,4,1000,1))
ans = 2.0169
```

Octave poskytuje teda štatistické funkcie pre známe a používané typy rozdelení. Ďalšie podrobnosti o ich parametroch nájdete v príručke Octave (EATON, 1997).

10.1.3. Testovanie štatistických hypotéz

Vzhľadom na to, že autor nie je odborník v oblasti matematickej štatistiky, uvedieme abecedný zoznam testov bez ich podrobného popisu, ktorý zvedavý čitateľ nájde v príručke Octave (EATON, 1997):

`anova` – test ANOVA rozdielu populácií,

`bartlett_test` – Bartlettov test homogenity rozpylu,

`chisquare_test_homogeneity` – χ^2 test na porovnanie rozdelenia dvoch vzoriek,

`chisquare_test_independence` – χ^2 test nezávislosti,

`cor_test` – testuje, či dve vzorky patria nekorelovaným populáciám,

`f_test_regression` – F test pre klasický regresný model,

`hotelling_test` – testovanie zhody strednej hodnoty vzorky s danou hodnotou,

`hotelling_test_2` – testovanie zhody strednej hodnoty dvoch vzoriek,

`kolmogorov_smirnov_test` – Kolmogorovov-Smirnovov test rozdelenia,

`kolmogorov_smirnov_test_2` – Kolmogorovov-Smirnovov test zhody rozdelenia dvoch vzoriek,

`kruskal_wallis_test` – Kruskalova-Wallisova jednofaktorová „analýza rozptylu“,

`manova` – test MANOVA (viacrozmerný test ANOVA),

`mcnemar_test` – McNemarov test symetrie,

`prop_test_2` – test rovnosti pravdepodobností dvoch vzoriek,

`run_test` – test nezávislosti údajov,

`sign_test` – znamienkový test,

`t_test` – testovanie zhody strednej hodnoty vzorky s danou hodnotou pri neznámom rozptyle,

`t_test_2` – testovanie zhody strednej hodnoty dvoch vzoriek pri neznámych rozptyloch,

`t_test_regression` – t test pre klasický regresný model,

`u_test` – Mannov-Whitneyov znamienkový U test (ekvivalentný s Wilcoxonovým testom),

`var_test` – F test zhody disperzií dvoch vzoriek,

`welch_test` – Welchov test zhody strednej hodnoty dvoch vzoriek s neznámymi a možno rôznymi disperziami,

`wilcoxon_test` – Wilcoxonov znamienkový test,

`z_test` – z test zhody strednej hodnoty vzorky so zadanou hodnotou pri známom rozptyle,

`z_test_2` – z test zhody strednej hodnoty dvoch vzoriek pri známych rozptyloch.

V nasledujúcom príklade v riadku 12 vygenerujeme 1000 pseudonáhodných hodnôt s normálnym rozdelením so strednou hodnotou $\mu = 3$ a s disperziou $\sigma^2 = 9$. V riadkoch 14 respektíve 15 otestujeme, či je stredná hodnota získanej vzorky rovná 3 respektíve 4:

```
Stat:12> x=normal_rnd(3,9,1000,1);
Stat:13> [mean(x) std(x)]
ans =
  3.02198  3.05557
Stat:14> hotelling_test(x,3);
  pval: 0.820084
Stat:15> hotelling_test(x,4);
  pval: 0
```

Odpoveď `pval` je hodnota pravdepodobnosti, že sa výberová stredná hodnota vzorky rovná zadanej hodnote.

Nie vo všetkých prípadoch testov je jasné, ako sa majú použiť. Preto môže byť efektívnejšie napísať vlastné testy s využitím štatistických funkcií rôznych rozdelení.

10.2. Práca s polynómami

Vo viacerých oblastiach sa stretávame s polynómami a s rôznymi úlohami, s nimi spojenými. Zapíšme polynóm $p(x)$ premennej x stupňa n v tvare

$$p(x) = c_1 \cdot x^n + c_2 \cdot x^{n-1} + \cdots + c_n \cdot x + c_{n+1}. \tag{5}$$

S výnimkou *nulového polynómu*, za stupeň ktorého budeme pokladať nulu, koeficient $c_1 \neq 0$. Na symbolické zobrazenie polynómu sa používa funkcia `polyout`. Program Octave nemá dôvod pri zadávaní vektora požadovať, aby bol 1. koeficient nenulový. Nadbytočné nuly je možné odstrániť:

```
Polynomy:1> c=[0 0 3 1 4 1]; polyout(c,"t");
0*t^5 + 0*t^4 + 3*t^3 + 1*t^2 + 4*t^1 + 1
Polynomy:2> c=polyreduce(c); polyout(c,"t");
3*t^3 + 1*t^2 + 4*t^1 + 1
```

10.2.1. Riešenie algebrických rovníc

Štandardnou úlohou je riešenie algebrických rovníc, teda určenie koreňov polynómu. Existuje viacero metód riešenia algebrických rovníc (pozri napríklad učebnicu (KAUKIČ, 1998)). Octave poskytuje numerické riešenie algebrických rovníc s komplexnými koeficientami nad poľom komplexných čísel:

```
Polynomy:3> roots(c)
ans =
  -0.03975 + 1.14524i
  -0.03975 - 1.14524i
  -0.25384 + 0.00000i
```

Podľa koreňov charakteristickej rovnice matice sústavy lineárnych diferenciálnych rovníc usudzujeme o stabilite jej riešení. Octave umožňuje určiť charakteristický polynóm matice $\boldsymbol{A}$ v tvare $p(r) = \det(r \cdot \boldsymbol{E} - \boldsymbol{A})$, kde $\boldsymbol{E}$ je jednotková matica odpovedajúceho rozmeru:

```
Polynomy:4> A=[1 2 3;4 5 6;7 8 9]; p=poly(A); polyout(p,"r");
1*r^3 - 15*r^2 - 18*r^1 - 2.217e-14
Polynomy:5> roots(p)
ans =
    1.6117e+01
   -1.1168e+00
   -1.2319e-15
```

Keďže sa jedná o numerické riešenie, namiesto presných nulových hodnôt (koeficientu charakteristického polynómu a vlastnej hodnoty matice A) dostávame *extrémne malé* hodnoty.

Polynómy sa dajú derivovať a integrovať (konštanta integrovania je zvolená 0):

```
Polynomy:6> polyout(polyderiv(c),"x");
9*x^2 + 2*x^1 + 4
Polynomy:7> polyout(polyinteg(c),"z");
0.75*z^4 + 0.3333*z^3 + 2*z^2 + 1*z^1 + 0
```

10.2.2. Rozklad racionálnej funkcie na súčet parciálnych zlomkov

Racionálna funkcia $R(s) = \dfrac{P(s)}{Q(s)}$, kde P a Q sú polynómy, sa dá rozložiť na súčet polynómu a parciálnych zlomkov. V Octave na to slúži funkcia `residue`:[43]

[43] Stĺpcové vektory `r`, `s` a `e` sme zapísali do riadku!

```
Polynomy:8> P=[1,-3,-1,13,-7]; Q=[1,-5,8,-4];
Polynomy:9> [r,s,k,e]=residue(P,Q)
r =
  -2.0000   7.0000   3.0000
s =
  2.00000  2.00000  1.00000
k =
  1  2
e =
  1  2  1
```

Môžete sa presvedčiť, že platí:[44]

$$\frac{s^4-3s^3-s^2+13s-7}{s^3-5s^2+8s-4} = s+2+\frac{-2}{s-2}+\frac{7}{(s-2)^2}+\frac{3}{s-1}. \qquad (6)$$

10.2.3. Aproximácia polynómami v zmysle najmenších štvorcov

Funkcia `polyfit(x,y,n)` určí polynóm, ktorý v zmysle najmenších štvorcov minimalizuje odchýlku zadaných (experimentálnych) a teoretických údajov. V prípade interpolačného polynómu bude táto odchýlka nulová. Na vstupe sú dané hodnoty nezávislej premennej a im odpovedajúce

[44] Vo vektore `e` sú uložené mocniny koreňov menovateľa, obsiahnutých vo vektore `s`, význam vektorov `r` a `k` určte sami.

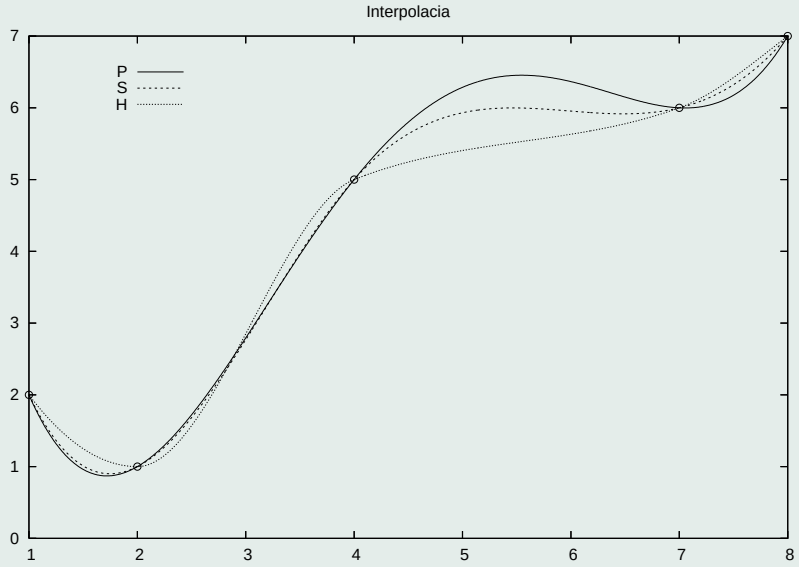
funkčné hodnoty, ako aj stupeň polynómu. Funkciu `polyval(p,t)` využijeme na výpočet funkčných hodnôt polynómu `p` v zadaných uzlových bodoch:

```
Polynomy:10> x=[1 2 4 7 8]; y=[2 1 5 6 7]; t=1:0.01:8;
Polynomy:11> [p]=polyfit(x,y,4); yt=polyval(p,t);
```

Grafické znázornenie nájdete v nasledujúcom pododdiele. Výsledky sú znázornené na grafe čiarou P.

10.2.4. Splajny a funkcie po častiach kubické

Priečinok `...\share\octave\2.1.50\site\m\octave-forge\splines` obsahuje niekoľko funkcií na výpočet koeficientov po častiach kubických funkcií aj splajnov, na výpočet funkčných hodnôt interpolačných funkcií v zadaných bodoch a na vykreslenie grafov. V článku (SCHUSTEROVÁ, 2005) nájdete informáciu o týchto metódach.



Vytvorené grafy sme získali pridaním nasledujúcej postupnosti príkazov, doplnenej zadaním legendy[45] a zmenou výstupu do súboru v GNUPLOTe príkazmi `set term postscript eps` a `set output "int.eps"`:

[45] Navyše sme v GNUPLOTe umiestnili legendu na vhodnejšie miesto príkazom `set key 2, 6.5` (možné je aj zadanie `set key outside`).

```
Polynomy:12> ys=spline(x,y,t); yc=pchip(x,y,t);
Polynomy:13> plot(t,yt,'1;P;',t,ys,'3;S;',t,yc,'4;H;',x,y,'*');
Polynomy:14> title("Interpolacia");
```

V riadku 10 sme zadali uzlové body a funkčné hodnoty v týchto bodoch. Navyše sme zadali body `t`, v ktorých sme počítali hodnoty interpolačných funkcií.

V riadku 12 sme zavolali funkciu `spline` s tromi argumentami. Hodnoty výsledného splajnu sme uložili do vektora `ys`. Ďalej sme zavolali funkciu `pchip`, ktorá realizuje tzv. Hermitovu po častiach kubickú interpoláciu. Hodnoty výslednej funkcie sme uložili do vektora `yc`. Táto funkcia zachováva „monotónnosť" údajov. Výsledky sú znázornené na grafe čiarami S respektíve H.

Ďalšie funkcie `mkpp`, `fnder` a `fnplt` umožňujú získať koeficienty interpolačných funkcií alebo ich derivácií a tiež vykresliť splajn.

Všetky tieto funkcie sú ukážkou užívateľských funkcií poskytnutých komunite používateľov programu Octave.

10.3. Riešenie sústav nelineárnych rovníc

Octave poskytuje funkciu `fsolve` na numerické riešenie sústav nelineárnych rovníc. Pomocou funkcie `fsolve_options` sa dajú nastaviť rôzne parametre metódy. Podrobnejší popis nájdete v príručke Octave (EATON, 1997) alebo ho získate príkazom `help`.

Uvažujme, napríklad sústavu nelineárnych rovníc

$$\begin{aligned} x^2 + y^4 - 5 &= 0, \\ x^4 - y^2 + \sin(x) &= 0. \end{aligned} \tag{7}$$

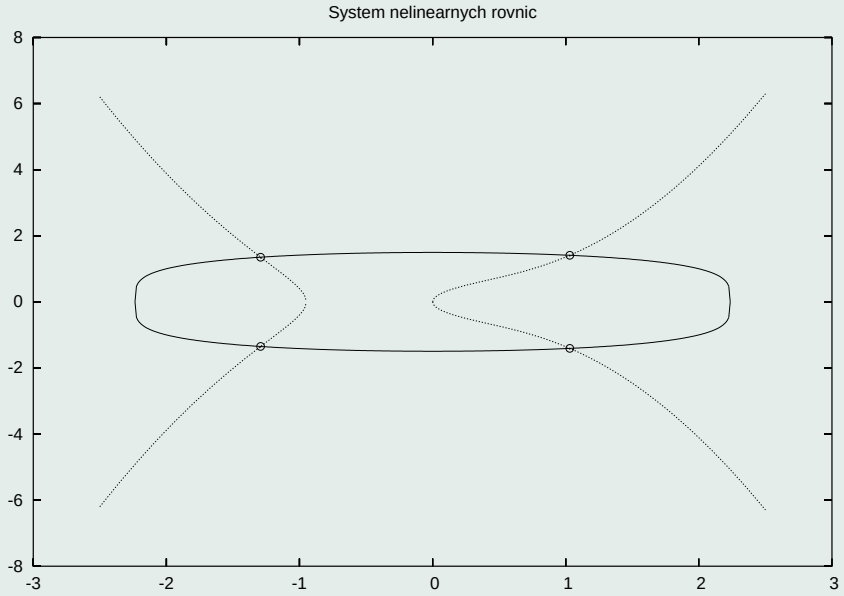
Ďalej stručne opíšeme postup, ktorý nás dovedie k riešeniu systému (7).

Najprv zadáme systém. Vytvoríme `m`-súbor, napríklad `nsystem.m`, ktorý bude obsahovať riadky:

```
function [r]=nsystem(x)
% System nelinearnych rovnic
   r(1)=x(1)^2+x(2)^4-5;
   r(2)= x(1)^4+sin(x(1))-x(2)^2;
```

Pomocou grafických prostriedkov Octave sa (v tomto prípade) môžete pokúsiť graficky znázorniť daný systém. Je dobré vedieť, či systém má nejaké riešenie a ak áno, koľko ich má? Na základe tvaru rovníc, ktoré obsahujú len párne mocniny premennej y, hneď vidíme, že ak systém má riešenie (x^*, y^*), tak riešením bude aj $(x^*, -y^*)$. Podrobnejšou analýzou zistíme, že systém má práve 4 riešenia. Keď sme už získali predstavu, približne v akej oblasti sa nachádzajú, je čas odovzdať slovo programu Octave (okrem uvedených riešení ľahko nájdeme zvyšné dve). Všetko (systém a jeho riešenia) môžeme potom graficky znázorniť:

```
Systemy:1> x1=fsolve("nsystem",[1,1])
x1 =
  1.03035
  1.40873
Systemy:2> x2=fsolve("nsystem",[-1,1])
x2 =
  -1.29200
   1.35094
```



10.4. Záver

Okrem uvedených aplikácií poskytuje Octave funkcie na:

- numerický výpočet určitých integrálov (`gquad`),
- približné riešenie začiatočných úloh pre diferenciálne rovnice a sústavy (`lsode`),
- približné riešenie diferenciálno-algebrických rovníc (`dassl`),
- riešenie úloh lineárnej metódy najmenších štvorcov (`gls` a `ols`),
- riešenie úloh z oblasti finančnej matematiky (`fv`, `fvl`, `irr`, `nper`, `npv`, `pmt`, `pv`, `pvl`, `rate` a `vol`),[46]
- riešenie úloh teórie riadenia (str. 211–258 príručky Octave (EATON, 1997)),
- spracovanie signálov, vrátane rýchlej Fourierovej transformácie (strany 259–265 príručky Octave (EATON, 1997)),
- spracovanie obrazov a audio údajov (strany 267–272 príručky Octave (EATON, 1997)).

Popri štandardných funkciách, opísaných v príručke Octave (EATON, 1997), sa mnohé užívateľské funkcie nachádzajú v priečinkoch uvedených v oddiele 7.3.

[46] Vo verzii 2.1.73 poskytuje Octave aj nástroj „Econometrics“.

P. Prílohy

P.1. Zoznam funkcií

Nižšie uvádzame zoznam najčastejšie používaných funkcií, väčšinou opísaných v príručke Octave (EATON, 1997). Ďalšie funkcie môžete nájsť v adresárovej štruktúre programu Octave, ak si dáte vyhľadávať súbory s príponou `.oct`.

a `abcddim, abs, acos, acosh, acot, acoth, acsc, acsch, all, angle, any, are, arg, asctime, asec, asech, asin, asinh, atan, atan2, atanh, atexit, axis`

b `balance, bar, besseli, besselj, besselk, bessely, beta, betai, bin2dec, bincoeff, blanks, bottom_title, bug_report`

c `c2d, cd, ceil, chdir, chol, clc, clear, clearplot, clg, clock, closeplot, colloc, colormap, columns, common_size, commutation_matrix, compan, complement, computer, completion_matches, cond, conj, contour, conv, corrcoef, cos, cosh, cot, coth, cov, cputime, create_set, cross, csc, csch, ctime, cumprod, cumsum`

d `dare, dassl, dassl_options, date, deblank, dec2bin, dec2hex, deconv, det, detrend, dgram, diag, diary, diff, dir, disp, dlqe, dlqr, dlyap, document, dup2, duplication_matrix`

e echo, edit_history, eig, endgrent, endpwent, erf, erfc, erfinv, error, etime, eval, exec, exist, exit, exp, expm, eye

f fclose, fcntl, feof, ferror, feval, fflush, fft, fft2, fftconv, fftfilt, fgetl, fgets, figure, file_in_path, filter, find, findstr, finite, fix, fliplr, flipud, floor, fnmatch, foo, fopen, fork, format, fprintf, fputs, fread, freport, freqz, frewind, fscanf, fseek, fsolve, fsolve_options, ftell, fwrite

g gamma, gammai, gammaln, gcd, getegid, getenv, geteuid, getgid, getgrent, getgrgid, getgrnam, getpgrp, getpid, getppid, getpwent, getpwnam, getpwuid, getrusage, getuid, givens, glob, gls, gmtime, gplot, gray, gray2ind, grid, gset, gshow, gsplot

h hankel, help, hess, hex2dec, hilb, hist, history, hold, home

i ifft, ifft2, imag, image, imagesc, imshow, ind2gray, ind2rgb, index, input, int2str, intersection, inv, inverse, invhilb, is_controllable, is_global, is_leap_year, is_matrix, is_observable, is_scalar, is_square, is_struct, is_symmetric, is_vector, isalnum, isalpha, isascii, iscntrl, isdigit, isempty, isgraph, ishold, isieee, isinf, islower, isnan, isprint, ispunct, isspace, isstr, isupper, isxdigit

k kbhit, keyboard, kron, kurtosis

l lcm, length, lgamma, lin2mu, linspace, load, loadaudio, loadimage, localtime, log, log10, log2, loglog, logm, logspace, lqe, lqr, ls, lsode, lsode_options, lstat, lu, lyap

m mahalanobis, max, mean, median, menu, mesh, meshdom, min, mkdir, mkfifo, mktime, more, mplot, mu2lin, multiplot

n nargchk, newtroot, nextpow2, norm, ntsc2rgb, null, num2str

o ocean, octave_config_info, ols, oneplot, ones, orth

p pause, pclose, perror, pinv, pipe, playaudio, plot, plot_border, polar, poly, polyderiv, polyfit, polyinteg, polyreduce, polyval, polyvalm, popen, popen2, pow2, printf, prod, purge_tmp_files, putenv, puts, pwd

q qr, quad, quad_options, quit, qzhess, qzval

r rand, randn, rank, readdir, real, record, rem, rename, replot, reshape, residue, rgb2ind, rgb2ntsc, rindex, rmdir, roots, rot90, round, rows, run_history

s save, saveaudio, saveimage, scanf, schur, sec, sech, semilogx, semilogy, set, setaudio, setgrent, setpwent, setstr, shg, shift, show, sign, sin, sinc, sinh, size, skewness, sleep, sort, source, sparse, split, sprintf, spy, sqrt, sqrtm, sscanf, stairs, stat, std, str2mat, str2num, strcat, strcmp, strerror, strftime,

strrep, struct_contains, struct_elements, subplot, substr, sum, subwindow, sumsq, svd, syl, sylvester_matrix, system

t tan, tanh, tic, tilde_expand, time, title, tmpnam, toc, toascii, toeplitz, tolower, top_title, toupper, trace, tril, trisolve, triu, type, tzero

u umask, undo_string_escapes, union, unlink, usage, usleep

v va_arg, va_start, vander, vec, vech, version, vr_val

w waitpid, warning, which, who, whos

x xlabel, xor

y ylabel

z zeros, zlabel

P.2. Zoznam systémových premenných

A

`all_va_args` – úplný zoznam voliteľných argumentov
`ans` – premenná obsahuje výsledok poslednej operácie ak nebol explicitne priradený nejakej premennej

`argv` – premenná obsahuje argumenty príkazového riadku Octave alebo odovzdané skriptu

`auto_unload_dot_oct_files` –

`automatic_replot` – ak je táto premenná nenulová, pri každej zmene v grafe sa prekreslí

B

`beep_on_error` – ak je táto premenná nenulová, pred každým chybovým hlásením sa Octave snaží „zazvonit'"

C

`completion_append_char` – znak, pridávaný za úspešne ukončenie príkazového riadku

`crash_dumps_octave_core` – ak je nenulová, Octave sa snaží uložit' všetky aktuálne premenné v prípade „krachu" do súboru `octave-core`

D

`default_eval_print_flag` – ak je nenulová, Octave automaticky vypíše výsledok príkazu, ak nekončí bodkočiarkou

`default_global_variable_value` – môže byt' použitá ako začiatočná hodnota globálnych premenných, ktoré neboli explicitne inicializované

`DEFAULT_LOADPATH` – zoznam adresárov oddelených dvojbodkou, v ktorých Octave hľadá funkcie

`default_return_value` – hodnota priradená neinicializovaným výstupným hodnotám

`default_save_format` – implicitný formát, ktorý použije príkaz `save`, ak nie je formát špecifikovaný

`define_all_return_values` – ak je táto premenná nenulová, hodnota `default_return_value` bude priradená nedefinovanej výstupnej hodnote

`do_fortran_indexing` – ak je táto hodnota nenulová, je možné k prvkom dvojrozmerného poľa pristupovať pomocou jediného indexu ako ku prvkom vektora vytvoreného stĺpcami matice

E

`e` – základ prirodzeného logaritmu

`echo_executing_commands` – podľa hodnoty tejto premennej sa riadi správanie príkazu `echo`

`EDITOR` – obsahuje názov editora používaného v príkaze `edit_history`

`empty_list_elements_ok` – ak je táto premenná nenulová, Octave ignoruje prázdne matice vo výrazoch

`eps` – počítačová presnosť, závisí od systému

`error_text` – obsahuje chybové hlásenie príkazov `unwind_protect`, `try` alebo `eval`

`EXEC_PATH` – zoznam adresárov oddelených dvojbodkou, v ktorých Octave hľadá pri vykonávaní podprogramov

F

`fixed_point_format` – ak je táto premenná nenulová, pri výpise matice Octave hodnoty škáluje tak, aby najväčšia z nich mala pred desatinnou bodkou jednu platnú cifru, škálovací faktor sa vypisuje na prvom riadku výstupu

G

`gnuplot_binary` – názov programu volaného príkazom `plot`
`gnuplot_has_frames` – ak je táto premenná nenulová, Octave predpokladá, že GNUPLOT má podporu násobných obrázkov
`gnuplot_has_multiplot` – ak je táto premenná nenulová, Octave predpokladá, že GNUPLOT má podporu násobných grafov

H

`history_file` – obsahuje názov súboru s uloženou históriou príkazov
`history_size` – obsahuje maximálny počet položiek histórie

I

`I` – imaginárna jednotka
`i` – imaginárna jednotka
`ignore_function_time_stamp` – podľa tejto premennej sa Octave rozhoduje, či je potrebné funkcie znovu kompilovať
`IMAGEPATH` – zoznam adresárov oddelených dvojbodkou, v ktorých Octave hľadá obrázky

`implicit_num_to_str_ok` – ak je táto hodnota nenulová, pri zmiešanom zadaní reťazcov sa používa konverzia čísel na odpovedajúce ASCII znaky
`implicit_str_to_num_ok` – ak je táto hodnota nenulová, používa sa konverzia ASCII znakov na odpovedajúce hodnoty
`inf` – nekonečno – výsledok operácií typu 1/0
`Inf` – nekonečno – výsledok operácií typu 1/0
`INFO_FILE` – ukazuje umiestnenie „info“ súboru Octave
`initialize_global_variables` – ovplyvňuje inicializáciu výstupných hodnôt

J

`J` – imaginárna jednotka
`j` – imaginárna jednotka

L

`LOADPATH` – zoznam adresárov oddelených dvojbodkou, v ktorých Octave hľadá skripty

M

`max_recursion_depth` – maximálny počet rekurzívnych volaní funkcie

N

`nan` – „not a number“ – výsledok neurčitých operácií typu 0/0
`NaN` – „not a number“ – výsledok neurčitých operácií typu 0/0

`nargin` – obsahuje počet vstupných argumentov volanej funkcie
`nargout` – obsahuje počet výstupných hodnôt, ktoré má funkcia vrátiť

O

`O_APPEND` – ovplyvňuje zápis do súborov
`O_ASYNC` – ovplyvňuje zápis do súborov
`O_NONBLOCK` – ovplyvňuje zápis do súborov
`O_RDONLY` – ovplyvňuje zápis do súborov
`O_RDWR` – ovplyvňuje zápis do súborov
`O_SYNC` – ovplyvňuje zápis do súborov
`O_WRONLY` – ovplyvňuje zápis do súborov
`ok_to_lose_imaginary_part` – ak je táto premenná nenulová, je povolená implicitná „konverzia“ komplexných čísel na reálne; ak je táto hodnota `"warn"`, konverzia je sprevádzaná pozornením
`output_max_field_width` – maximálna šírka numerického výstupu
`output_precision` – minimálny počet zobrazovaných platných cifier

P

`page_output_immediately` – ak je táto hodnota nenulová, Octave zasiela výstup na `PAGER` okomžite
`page_screen_output` – ak je táto premenná nenulová, výpisy dlhšie ako jedna „obrazovka“ sa „stránkujú“
`PAGER` – voľby `"less"`, `"more"` alebo `"cg"`
`pi` – Ludolfovo číslo π

`prefer_column_vectors` – ak je táto premenná nenulová, vektory, ktorých typ nie je jasný, sú považované za stĺpcové

`prefer_zero_one_indexing` – význam tejto premennej nie je veľmi jasný

`print_answer_id_name` – ak je táto premenná nenulová, spolu s výsledkom sa vypisujú aj názvy premenných

`print_empty_dimensions` – ak je táto premenná nenulová, spolu s prázdnou maticou sa vypisuje aj jej typ

`program_invocation_name` – obsahuje reťazec, ktorým bol program Octave spustený

`program_name` – obsahuje názov programu

`propagate_empty_matrices` – ak je táto premenná nenulová, maticové funkcie vracajú prázdnu maticu, ak majú na vstupe prázdnu maticu

`PS1` – obsahuje primárnu výzvu (prompt)

`PS2` – obsahuje sekundárnu výzvu, ktorá sa zobrazuje, keď Octave očakáva ďalší vstup na ukončenie príkazu

`PS4` – ovplyvňuje začiatok riadku „echa“

R

`realmax` – najväčšie číslo vo formáte plávajúcej desatinnej bodky

`realmin` – najmenšie kladné číslo vo formáte plávajúcej desatinnej bodky

`resize_on_range_error` – ak je táto premenná hodnota nenulová, veľkosť matice sa priradením hodnôt ďalším prvkom automaticky zväčší, nedefinovaným prvkom matice sa priradí hodnota 0

`return_last_computed_value` – ak je táto premenná nenulová a funkcia

nemá definovanú výstupnú hodnotu, vráti funkcia poslednú vypočítanú hodnotu, ináč vráti hodnotu 0

S

`save_precision` – pemenná určuje počet cifier pri uložení čísel do textového súboru

`saving_history` – ak je hodnota tejto premennej nenulová, príkazy sa ukladajú do súboru

`silent_functions` – ak je táto premenná nenulová, hodnoty výrazov vo volaných funkciách, ktoré sa nekončia bodkočiarkou, nie sú zobrazované

`split_long_rows` – pri nenulovej hodnote sa dlhé riadky delia

`stderr` – štandardný chybový výstup

`stdin` – štandardný vstup

`stdout` – štandardný výstup

`string_fill_char` – obsahuje znak, ktorý sa dopĺňa do prvkov matice reťazcov tak, aby mali všetky rovnakú dĺžku

`struct_levels_to_print` – určuje počet zobrazovaných štruktúrnych úrovní

`suppress_verbose_help_message` – ak je táto hodnota nenulová, príkaz `help` má kratší výpis

T

`treat_neg_dim_as_zero` – ak je táto premenná nenulová, záporné dimenzie sa nastavujú na nulu (vytvoria sa prázdne matice)

W

`warn_assign_as_truth_value` – ak je táto premenná nenulová, pri chybnom zápise podmienky `if (a=5) ...` namiesto `if (a==5) ...` sa vypíše varovanie a podmienke sa priradí hodnota 1

`warn_comma_in_global_decl` – čiarka pri oddelení premenných v príkaze `global` sa pri nenulovej hodnote tejto premennej nechápe ako oddeľovač príkazov

`warn_divide_by_zero` – pri delení číslom 0 sa objaví varovanie, ak je táto premenná nenulová

`warn_function_name_clash` – ak je táto hodnota nenulová, Octave upozorní na rozdielny názov `m`-súboru a funkcie, ktorá je v ňom definovaná

`warn_missing_semicolon` – pri nenulovej hodnote tejto premennej Octave upozorní na chýbajúcu bodkočiarku za príkazom vo vnútri funkcie

`warn_reload_forces_clear` – ovplyvňuje správanie Octave v prípade definícií viacerých funkcií v jednom `m`-súbore

`warn_variable_switch_label` – ak je táto premenná nenulová, Octave upozorní na prípad, keď značka príkazu `switch` nie je konštantná

`whitespace_in_literal_matrix` – táto premenná ovplyvňuje chápanie medzery pri zadávaní matice

P.3. História príkazov

Na opätovné vyvolanie použitých príkazov ukladá Octave zadané príkazy do pamäte. Pri vypnutí Octave sa posledné zadávané príkazy (ich

maximálny počet je definovaný v systémovej premennej `history_size`, napr. 1024) zapisujú do súboru, ktorého názov je definovaný v premennej `history_file`, napríklad v našej inštalácii je to súbor `.octave_hist` v pracovnom priečinku.

Najpraktickejšie je asi používanie klávesov šípiek ↑ a ↓ na pohyb medzi poslednými použitými príkazmi. Rovnaký efekt sa dosiahne zadaním dvojice Ctrl-p, respektíve Ctrl-n. Dvojice klávesov Ctrl-r a Ctrl-s slúžia na vyhľadávanie reťazcov v súbore histórie smerom „hore“ a „dole“ od aktuálneho riadku histórie. Pri zadaní prvého písmena reťazca sa hneď zobrazí prvý riadok, obsahujúci toto písmeno. Ďalším zadávaním písmen môžeme reťazec upresniť. Tým sa nastavíme na určitý riadok „histórie“, od ktorého môžeme pokračovať šípkami hore alebo dole.

Literatúra

Doboš, J. 2006. *GNUPLOT*, 52 s. 19, 121, 133

Eaton, J. W. 1997. *GNU Octave Manual*, Network Theory Limited, ISBN: 0-9541617-2-6 , 324 s. 8, 152

Eaton, J. W. 1997. *GNU Octave. A high-level interactive language for numerical computations*, http://pcmap.unizar.es/softpc/OctaveManual.pdf. 8, 11, 95, 107, 110, 117, 118, 133, 134, 137, 139, 148, 151

Eaton, J. W. 1997. *GNU Octave Manual. A high-level interactive language for numerical computations*, Edition 3 for Octave version 2.0.13, online na stránke http://www.network-theory.co.uk/docs/octave/. 27, 72

Forsythe, G. E. — Malcolm, M. A. — Moler, C. B. 1977. *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs. 54

Just, M. 2006. *Octave. Český průvodce programem*, http://www.octave.cz/. 11, 12

Golub, G. H. 1968. *Least Squares, Singular Values and Matrix Approximation*, Aplikace matematiky, **13**, N. 1, 44–51. 54

Golub, G. H. — van Loan, C. F. 1989. *Matrix Computations*, Johns Hopkins UniversityPress, Baltimore, MD, 2nd edition. 54

Kahaner, D. — Moler, C. — Nash, S. 1989. *Numerical methods and Software*, Prentice-Hall International, Inc. 54

Kaukič, M. 1998. *Numerická analýza I. Základné problémy a metódy*. Žilina, MC Energy s. r. o. 10, 54, 143

Kaukič, M. 2006. *Základy programovania v Pylabe*. 10

Schusterová, J. 2005. *Poznámky k výučbe interpolácie*. 4. konferencia Aplimat, KM SjF STU Bratislava. 146

NÁZOV: Octave. Rozšírený úvod
AUTOR: Ján Buša
VYDALA: FEI TU v Košiciach — Edícia vysokoškolských učebníc
POČET STRÁN: 167
VYDANIE: prvé
SADZBA: elektronická, programom pdfTEX.

ISBN 80-8073-596-4