Fakulta matematiky, fyziky a informatiky UK

VYBRANÉ OPEN SOURCE PROSTRIEDKY NA VÝSKUM, VÝVOJ A SPRACOVANIE DÁT

Ján Buša
Michal Kaukič
Peter Mann
Štefan Peško
Ladislav Ševčovič
Miloš Šrámek

Učebný materiál

© **Autori:** Ján Buša, Michal Kaukič, Štefan Peško, Ladislav Ševčovič
Autori: Peter Mann, Miloš Šrámek

Názov: Vybrané open source prostriedky na výskum, vývoj a spracovanie dát

Projekt č.: JPD 3 BA 2005/1 – 029
Zodpovedný riešiteľ: Damas Gruska

Vydané v rámci projektu: „Skvalitnenie podmienok pre prípravu vysokokvalifikovaných odborníkov v oblasti informačných technológií“

Projekt, ktorý vydal túto publikáciu, je spolufinancovaný Európskou Úniou.

„Európsky sociálny fond pomáha rozvíjať zamestnanosť podporovaním zamestnanosti, obchodného ducha, rovnakých príležitostí a investovaním do ľudských zdrojov.“

Do tlače pripravili:	Ladislav Ševčovič, Angelika Takáčová
Rok vydania:	2007
Miesto vydania:	Bratislava
Vydanie:	prvé
Vydavateľ:	Knižničné a edičné centrum FMFI UK
Tlač:	Paci Computer Studio, Svätoplukova 8, 972 01 Bojnice
Náklad:	40 ks **počet strán:** 176
Internet:	`http://www.fmph.uniba.sk`

Súčaťou tejto publikácie je aj CD–ROM.

ISBN: 978-80-89186-16-7

OBSAH

Milí študenti,

do rúk sa Vám dostávajú študijné materiály, ktoré boli pripravené pre doktorandský kurz „Vybrané open source prostriedky na výskum, vývoj a spracovanie dát“, organizovaný v rámci riešenia grantovej úlohy „Skvalitnenie podmienok na prípravu vysokokvalifikovaných odborníkov v oblasti informačných technológií “ na Fakulte matematiky, fyziky a informatiky Univerzity Komenského v Bratislave.

V prvom rade by som chcel v mene lektorov poďakovať riešiteľom tejto grantovej úlohy za ponuku na organizovanie kurzu o otvorených, a teda aj voľne prístupných softvérových prostriedkoch. Kurz berieme, na jednej strane, ako šancu na propagovanie niekoľkých konkrétnych programov, ktorých spoločnou črtou je, že sú, podľa nás, vhodné na použitie pri výskume, vývoji nových aplikácií či výučbe na školách s prírodovedným alebo technickým zameraním. Mohlo by sa zdať, že propagácia programov, ktoré poskytujú širokú funkcionalitu a navyše sú aj prístupné zdarma, nie je dôležitá a ani potrebná – že takéto programy sa presadia samé. Ukazuje sa však, že opak je realitou. Študenti, učitelia, vedeckí pracovníci a vlastne všetci používatelia počítačov sú dnes vystavovaní mohutnému propagačnému tlaku zo strany výrobcov komerčných softvérových produktov, či už vo veľmi príťažlivej forme bohatej ponuky študijnej literatúry, propagačných konferencií, seminárov a workshopov alebo výhodných licenčných ponúk. Keď k tomu prirátame značnú zložitosť a návykovosť softvéru, ktoré vedú k istým stereotypom ohľadom toho, ktorý program sa na akú činnosť používa, vidíme, že propagácia otvorených produktov je dôležitá. Navyše, vo svete, kde obvykle nič nie je zdarma, a ak, tak sa za tým iste skrývajú dodatočné podmienky, je nie celkom prirodzené, ak niečo zdarma predsa len je. Radi by sme preto tiež priblížili princípy, a to je druhá stránka nášho kurzu, na ktorých otvorený a slobodný softvér stojí, a ako je možné, že vysokokvalitný softvér zdarma vôbec môže byť, vo svojej plnej funkcionalite a bez dodatočných „ale“.

Samotný kurz je rozdelený na päť častí – samostatných dní. V prvej (M. Šrámek) účastníkov oboznámime s históriou a „filozofickým“ pozadím otvoreného a slobodného softvéru, niektorými jeho právnymi aspektmi, výhodami a nevýhodami, pričom budeme pokračovať nástrojmi na vývoj softvéru. Témou druhej časti (J. Buša) budú programy na počítačovú algebru a LaTeX ako nástroj vhodný na písanie vedeckých publikácií a kvalifikačných prác. Tretia časť (M. Kaukič) bude zameraná na numerické výpočty a na nástroje, ktoré sú dostupné v prostredí jazyka Python. Vo štvrtej časti (Š. Peško) predstavíme tabuľkový procesor Gnumeric v netradičnej úlohe riešenia optimalizačných problémov a v poslednej, piatej (L. Ševčovič) sa budeme venovať spracovaniu a grafickému zobrazovaniu experimentálnych výsledkov.

Súčasťou študijných materiálov kurzu je aj CD so všetkými potrebnými materiálmi a preberanými programami (P. Mann, `http://people.tuke.sk/peter.mann/ubuntu/`, `http://people.tuke.sk/jan.busa/kega/livecd/slovak_math_ubuntu.pdf`). Ide o tzv. Live CD, z ktorého možno priamo zaviesť systém po jeho vložení do CD mechaniky. CD bude používané počas kurzu, avšak je možné ho plne legálne používať aj inde, či už na škole alebo aj mimo nej. CD poskytuje aj možnosť trvalej inštalácie na pevný disk, čím sa používateľom sprístupní široké spektrum ďalších programov. Uvedomujeme si, že používanie nových softvérových prostriedkov nemusí byť vždy bezproblémové. Z tohto dôvodu sme zriadili podpornú stránku `http://sk.openacademy.eu/sk/node/106` kde je priestor na kladenie otázok a ich odpovedanie, rovnako ako aj na vyjadrovanie názorov a návrhov. Registrácia je voľná pre všetkých, ste vítaní.

Záverom by som chcel poďakovať všetkým spoluautorom za vynaloženú prácu na príprave študijných materiálov a samotných prednášok. Najväčším ocenením našej spoločnej snahy bude, ak sa prostriedky, o ktorých kurz je, budú naozaj v praxi používať, a to aj mimo samotného kurzu. Dúfame, že sa nám v tomto smere bude dariť.

Bratislava marec 2007 Miloš Šrámek, organizátor kurzu

1 SYSTÉM POČÍTAČOVEJ ALGEBRY MAXIMA

Ján BUŠA
Katedra matematiky, FEI
Technická univerzita v Košiciach

1.1 Úvod

Program MAXIMA patrí medzi tzv. systémy počítačovej algebry (SPA), ktoré umožňujú vykonávanie symbolických aj numerických výpočtov (riešenia rovníc, derivovania, integrovania, a pod.) na počítači. Jedným z prvých SPA bol program Macsyma (projekt MAC's SYmbolic MAnipulator), ktorého vývoj sa začal v roku 1968 v MIT (Massachusetts Institute of Technology), pozri (SOUZA, FATEMAN, MOSES a YAPP, 2004). Medzi ďalšie SPA patrili od začiatku systémy Reduce, CAMAL, Mathlab-68, PM a ALTRAN. Výrazný skok nastal, až keď sa objavili programy Maple (1985) a Mathematica (1988), inšpirované SPA Macsyma. Spomeňme ešte MuPAD a Derive.

Maxima je pokračovateľom SPA Macsyma. Bola navrhnutá a udržiavaná profesorom Williamom F. Schelterom z Univerzity v Texase od roku 1982, až do jeho smrti v roku 2001. V roku 1998 získal od Oddelenia energie (Department of Energy) súhlas na zverejnenie zdrojového kódu programu DOE Macsyma pod licenciou GNU Public License a v roku 2000 inicializoval na SourceForge projekt MAXIMA, na údržbu a vývoj SPA DOE Macsyma pod terajším názvom MAXIMA. MAXIMA teda patrí medzi programy s otvoreným zdrojovým kódom – OPENSOURCE softvér. Program je možné kompilovať v rôznych OS, vrátane Windows, GNU/Linuxu a MacOS X. Predkompilovaný sa dá pre GNU/Linux a Windows bezplatne získať na stránke `http://sourceforge.net/project/showfiles.php?group_id=4933` SourceForge.

Maxima je systém na manipuláciu so symbolickými a numerickými výrazmi, vrátane derivovania, integrovania, výpočtu Taylorovych polynómov, Laplaceovej transformácie, riešenia obyčajných diferenciálnych rovníc, systémov lineárnych rovníc. Pracuje s polynómami, množinami, zoznamami, vektormi, maticami a tenzormi. Umožňuje získať veľmi presné výsledky použitím presných zlomkov a celých i desatinných čísel s ľubovoľnou presnosťou. Zobrazuje grafy funkcií jednej aj dvoch premenných.

Na stránke `http://maxima.sourceforge.net/` nájdete množstvo zaujímavých informácií, týkajúcich sa nielen programu MAXIMA, ale aj ďalších OPENSOURCE SPA a OPENSOURCE matematického softvéru. Rozsiahla dokumentácia, uvedená aj v zozname použitej literatúry, sa nachádza na stránke `http://maxima.sourceforge.net/docs.shtml`.

Z vyššie uvedeného vyplýva, že SPA MAXIMA sa dá využiť v rôznych oblastiach výskumu a výučby. Každý môže tento systém využívať za rovnakých podmienok (bezplatne) aj po ukončení štúdia. Táto kapitola je výberom z učebice (BUŠA, 2006), ktorej cieľom je oboznámiť čitateľa s inštaláciou programu a so základnými úlohami, ktoré sa s jeho pomocou dajú efektívne riešiť. Podrobný popis poskytuje rozsiahly (MAXIMA MANUAL, 2005). Je zrejmé, že mnohé príkazy v stručnom popise nie sú ani spomenuté, nielen opísané. Dôležité je urobiť prvý krok. Veríme, že táto kapitola Vám tento prvý (ale rozhodný) krok uľahčí.

1.2 Prvé kroky

1.2.1 Inštalácia programu MAXIMA

V tomto oddiele popíšeme inštaláciu programu MAXIMA v operačných systémoch Windows a GNU/Linux. Na stránke

`http://sourceforge.net/project/showfiles.php?group_id=4933`

je možné nájsť a stiahnuť si inštalačné `rpm` súbory pre OS GNU/Linux a súbor `maxima-5.11.0b.exe`[1] pre OS Windows.

Proces inštalácie v OS Windows odštartujeme spustením inštalačného súboru. Potvrdíme voľbu grafického prostredia WXMAXIMA, ktoré zjednodušuje prácu s programom MAXIMA.

Pri inštalácii programu WXMAXIMA v OS GNU/Linux postavených na Debiane je problém, že program MAXIMA skompilovaný s GNU/Lispom nebude fungovať s programom WXMAXIMA. Tento nedostatok je možné riešiť skompilovaním programu MAXIMA buď s balíkom `clisp` alebo s balíkom `cmucl`. Uvedieme postup inštalácie s balíkom `cmucl`:

1. odinštalujeme starú verziu programu MAXIMA aj balík `gcl`:
   `sudo apt-get remove --purge gcl maxima`
2. nainštalujeme balík `cmucl`:
   `sudo apt-get install cmucl`
3. stiahneme najnovšie zdrojáky programu MAXIMA:
   `http://prdownloads.sourceforge.net/maxima/maxima-5.11.0.tar.gz?download` (momentálne je najnovšia verzia 5.11)
4. skompilujeme program MAXIMA s balíkom `cmucl` namiesto `gcl`:
   ```
   tar xzf maxima-5.11.0.tar.gz
   cd maxima-5.11.0
   ./configure --enable-cmucl
   make install
   ```
5. nainštalujeme program WXMAXIMA (akceptujte všetky závislosti!):
   `apt-get install wxmaxima`

Týmto je inštalácia ukončená, teraz uz môžeme spustiť program MAXIMA s grafickou nadstavbou `wx`, napr. z príkazového riadku zapísaním príkazu `wxmaxima` a odoslaním klávesom ENTER.

Ďalej budeme predpokladať, že ste si program nainštalovali a máte možnosť jednotlivé ukážky overiť samostatne. Nie je to síce nutné, ale takéto štúdium bude určite efektívnejšie.

1.2.2 Prostredia programu MAXIMA

V knihe The Maxima Book autorov (SOUZA, FATEMAN, MOSES a YAPP, 2004) sú popísané viaceré prostredia programu MAXIMA. Okrem vyššie spomenutých grafických prostredí WXMAXIMA a XMAXIMA je možné použiť prostredia:

- terminál – pôvodné prostredie, práca v príkazovom riadku,

[1] Alebo novší.

- editor Emacs – prepracované negrafické prostredie, umožňuje kombinovať vstupy a výstupy programu MAXIMA s textom a po ich kompilácii programom TEX/LATEX vznikne typograficky kvalitný dokument,

- TEXmacs – vedecký WYSIWYG editor, spolupracujúci s rôznymi SPA. Dá sa využívať možnosť výstupu programu MAXIMA vo formáte TEX,

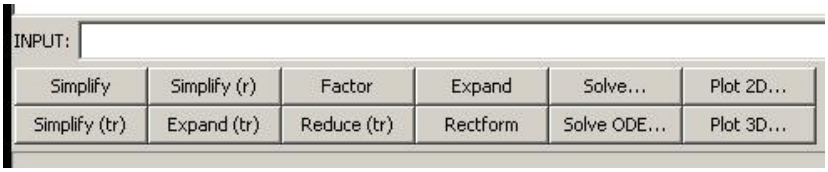
- ďalšie prostredia nie je odporúčané používať, nie sú kvalitne udržiavané.

1.2.3 Spustenie programu MAXIMA

V Linuxe spustíme program MAXIMA v príkazovom riadku terminálu príkazom `maxima` alebo `xmaxima`. Po potvrdení stlačením klávesu ENTER sa otvorí okno programu MAXIMA alebo grafického prostredia `xMaxima`. V OS Windows klikneme na ikonku WXMAXIMA (ďalej budeme popisovať túto možnosť) alebo XMAXIMA, čím zvolíme grafické prostredie programu MAXIMA. Príkazy môžeme zadávať v príkazovom riadku programu alebo použijeme položky menu, ku ktorým sa dostaneme použitím tlačidiel na hornej lište.

Po kliknutí na ikonku program WXMAXIMA sa na obrazovke objaví okno, v hornej časti ktorého sa nachádza lišta so základnou ponukou (obrázok 1). Položky na lište prostredia WXMAXIMA preskúmame neskôr. Nič Vám však nebráni pootvárať ich a presvedčiť sa o bohatých možnostiach SPA MAXIMA.

Obrázok 1: Položky menu programu MAXIMA



Obrázok 2: Príkazový riadok

V dolnej časti sa nachádza *vstupný riadok* (obrázok 2), v ktorom sa zadávajú príkazy programu MAXIMA a pod ním sú umiestnené tlačidlá vybraných funkcií programu MAXIMA.

Môžeme ho vyskúšať a postupne zadať, napríklad, príkazy uvedené v nasledujúcej ukážke v riadkoch, začínajúcich sa znakmi `(%i*)`, kde `*` je poradové číslo vstupného riadku. Zadanie každého riadku potvrdíme klávesom ENTER. V prvom riadku zadávame príkaz na riešenie kvadratickej rovnice $x^2 + 2x + 3 = 0$, ktorá má dva komplexné korene $x_{1,2} = -1 \pm \mathrm{i}\sqrt{2}$. Výpis je v symbolickom tvare, a preto ďalej vypisujeme 2. riešenie v desatinnom tvare s nastavenými rôznymi presnosťami.

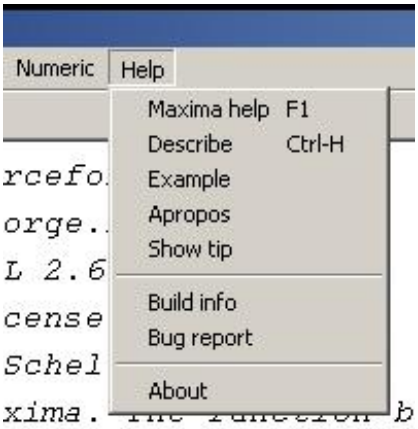
```
(%i1)  solve(x^2+2*x+3);
(%o1)  [ x = -√2 %i - 1 , x = √2 %i - 1 ]
(%i2)  %o1[2],float;
(%o2)  x = 1.414213562373095 %i - 1
(%i3)  fpprec : 4;
(%o3)  4
(%i4)  %o1[2],float;
(%o4)  x = 1.414213562373095 %i - 1
(%i5)  %o1[2],bfloat;
(%o5)  x = 1.414b0 %i - 1.0b0
(%i6)  fpprec : 50;
(%o6)  50
(%i7)  %o1[2],bfloat;
(%o7)  x = 1.4142135623730950488016887242096980785696718753777b0 %i - 1.0b0
```

Vidíme, že z dvoch príkazov – `float` a `bfloat` – na výpis čísla s plávajúcou desatinnou bodkou (anglicky floating point) len druhý reaguje na zmenu nastavenia počtu desatinných miest príkazom `fpprec` (floating point precision).

1.2.4 Help programu MAXIMA

Grafické prostredie WXMAXIMA poskytuje pomocnú informáciu užívateľovi prostredníctvom položky `Help` (obrázok 3).

Online `Maxima help` v prvom riadku otvoríme aj stlačením klávesu `F1`. Príkazy v ďalších riadkoch – `describe`, `example` a `apropos` – je možné zadávať aj v príkazovom riadku, teda sa dajú využívať aj v terminálovom režime programu MAXIMA. Vyskúšajme ich:[2]



Obrázok 3: Možnosti položky `Help`

[2]Pri dlhých výpisoch pokračuje MAXIMA v ďalšom riadku. Náš výpis môže byť rozdelený v inom mieste.

```
(%i8) apropos('pl);
(%o8) [pl,playback,plog,plot,plot2d,plot2dopen,plot2d_ps,
plot3d,plotheight,plotmode,plotting,plot_format,plot_options,
plot_realpart,plus]
```

Všimnite si apostrof ' za otváracou zátvorkou príkazu `apropos`. Netradične je len jeden, bez párového kamaráta na konci výrazu `pl`. Iste ste už pochopili, že funkcia `apropos` nám pomáha spomenúť si na presný názov príkazu. Túto funkciu v niektorých systémoch plní kláves TAB, umožňujúci výpis všetkých funkcií, ktorých názvy majú na začiatku daný výraz.

```
(%i9) describe(plot2d);
0: plot2d :(maxima.info)Definitions for Plotting.
1: plot2d_ps :Definitions for Plotting.
Enter space-separated numbers, 'all' or 'none':
Still waiting: 0;
-- Function: plot2d (<expr>, <range>, ..., <options>, ...)
-- Function: plot2d (<parametric_expr>)
-- Function: plot2d (<discrete_expr>)
    ...
   Displays a plot of one or more expressions as a function
   of one variable.
   ...
   See also 'plot_options', which describes plotting options
   and has more examples.
(%o9) false
```

Funkcia `describe("názov")` vypíše popis zadaného príkazu.[3] V tejto ukážke program čaká na špecifikáciu príkazu, keďže výraz `plot2d` je na začiatku názvu dvoch funkcií. Po zadaní voľby a potvrdení klávesom ENTER nasleduje výpis.[4]

Funkcia `example(názov)` poskytne ukážky použitia zadaného príkazu. Nasledujúci príklad ukazuje výsledok rôznych substitúcií, uskutočnených príkazom `subst`.[5] K opisu jeho syntaxe sa ešte vrátime neskôr:

```
(%i10) example(subst);
(%i11) subst(a,y+x,y+(y+x)^2+x)
(%o11) y+x+a^2
(%i12) subst(-%i,%i,b*%i+a)
(%o12) a-%i*b
(%i13) subst(x,y,y+x)
(%o13) 2*x
(%i14) subst(x = 0,diff(sin(x),x))
(%o14) 1
```

[3] Názov je možné zadať aj bez úvodzoviek.
[4] Vynechali sme vyše 80 riadkov rôznych užitočných rád týkajúcich sa použitia funkcie `plot2d`.
[5] Zobrazili sme len neúplný výpis príkladov.

1.2.5 Ukončenie činnosti programu MAXIMA

V terminálovom režime ukončíme prácu programu príkazom `quit();`. Prázdne zátvorky na konci sú *povinné*:

```
(%i15) quit;
(%o15) quit
(%i16) quit();
CLIENT: Lost socket connection ...
Restart maxima with 'Maxima->Restart maxima'.
```

V grafickom režime wxMAXIMA ukončíme činnosť programu MAXIMA prostredníctvom menu – `File/Exit` alebo zadaním Ctrl Q.

1.3 Čísla, výrazy a funkcie

1.3.1 Výrazy a priradenia

Výrazy sa zadávajú pomocou obvyklých znakov operácií a okrúhlych zátvoriek. Umocnenie sa zadáva znakom ^ alebo pomocou **. Na konci príkazu sa zadáva *bodkočiarka* ; alebo, ak chceme potlačiť zobrazenie výstupu, znak dolára $. V jednom riadku môžeme zadať niekoľko príkazov.

```
(%i1) a:3$ b:4$ a+b;
(%i2)
(%i3)
(%o3) 7
(%i4) a**3;
(%o4) 27
```

Ako je vidieť, znakom *priradenia* slúži netradične *dvojbodka* : a nie =! V riadku %i1 sme premenným a a b priradili hodnoty bez ich zobrazenia, potom sme vypočítali a zobrazili hodnotu $a + b$.

1.3.2 Rôzne spôsoby zobrazenia výrazov

Zadajme výraz $\frac{x}{\sqrt{x^2+1}}$:

```
(%i5)  x/sqrt(x^2+1);
                      x
(%o5)               ------
                       2
                    sqrt(x  + 1)
(%i6)
```

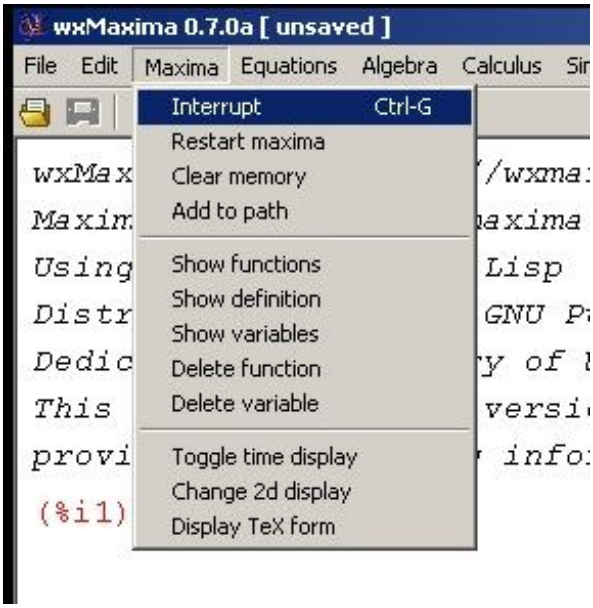
Medzi podpoložkami menu v záložke Maxima (obrázok 4) nájdeme možnosť zmeny zobrazovania výstupu kliknutím na Change 2d display. Otvorí sa ďalšie okienko, zobrazené na obrázku 5.

Vyberieme, napríklad, voľbu ascii a potvrdíme ju. Po opätovnom zadaní vstupu (%i5) sa výstup zmení na nasledujúci:

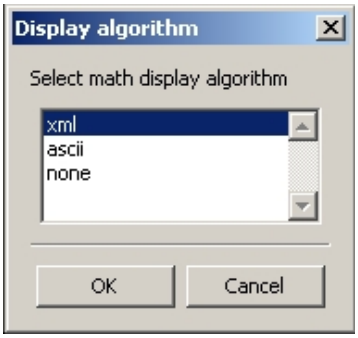
```
(%i6) set_display('ascii)$
(%i7) %i5;
                              x
(%o7)                  ------------
                             2
                       sqrt(x  + 1)
```

Pri zadaní voľby none bude zobrazovanie lineárne (jednorozmerné), keď sa výrazy zapisujú do jedného riadku:

```
(%i8) set_display('none)$
(%i9) %i5;
(%o9) x/sqrt(x^2+1)
```



Obrázok 4: Voľby programu MAXIMA



Obrázok 5: Režim zobrazovania výsledkov

K pôvodnému krajšiemu a príjemnejšiemu zobrazovaniu výsledkov sa v grafickom prostredí WXMAXIMA vrátime voľbou `xml`:

```
(%i10) set_display('xml)$
```

Na záver spomeňme ešte možnosť zapísať výstup vo formáte typografického programu TEX:

```
(%i11) tex(x/sqrt(x^2+1));
$${{x}\over{\sqrt{x^2+1}}}$$
(%o11) false
(%i12) tex(%o5);
$${{x}\over{\sqrt{x^2+1}}}\leqno{\tt (\%o5)}$$
(%o12) (%o5)
```

Na vstupe funkcie `tex` môžeme zadať buď samotný výraz alebo jeho odkaz na neho, prípadne jeho názov. Pri volaní funkcie `tex` je možné výstup zapísať do zadaného súboru.

Príkaz `kill`

Príkazom `kill` môžeme odstrániť premenné s ich všetkými priradeniami a vlastnosťami z aktuálneho pracovného prostredia programu MAXIMA:

```
(%i13) kill(a);
(%o13) done
(%i14) a;
(%o14) a
```

1.3.3 Zápis známych konštánt

V nasledujúcej tabuľke uvádzame zoznam konštánt používaných programom MAXIMA:

konštanta	MAXIMA
e – Eulerovo číslo	`%e`
i – imaginárna jednotka	`%i`
π – Ludolfovo číslo	`%pi`
∞ – reálne kladné nekonečno	`inf`
$-\infty$ – reálne záporné nekonečno	`minf`
∞ – komplexné nekonečno	`infinity`
lož – logická konštanta	`false`
pravda – logická konštanta	`true`

Aj konštanty, ktoré sú súčasťou vypočítaných výsledkov, uvádza MAXIMA znakom `%`.

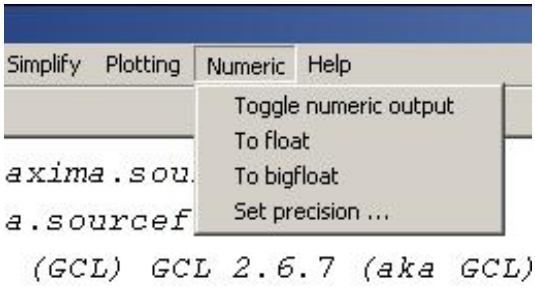
1.3.4 Voľba presnosti výpočtu a zobrazovania reálnych čísel

MAXIMA dokáže pracovať s presnými reálnymi číslami, zapísanými v symbolickom tvare. O tom, či sa čísla zapisujú v symbolickom tvare alebo v tvare desatinných čísel rozhoduje nastavenie premennej `numer`.

V grafickom prostredí WXMAXIMA môžeme formát výpisu ovplyvniť v záložke `Numeric`, kde sa dá voľbou `Toggle numeric output` prepínať medzi symbolickým a dekadickým zápisom:[6]

```
(%i15) numer:true;
(%o15) true
(%i16) sqrt(5);
(%o16) 2.23606797749979
(%i17) numer:false;
(%o17) false
```

[6] Z neznámych dôvodov sa to netýka Eulerovho čísla e! V desatinnom tvare ho vypíšeme príkazom `%e,numer` alebo `float(%e)` (sú aj ďalšie možnosti).



Obrázok 6: Položka `Numeric`

```
(%i18) %e;
(%o18) %e
(%i19) %e,numer;
(%o19) 2.718281828459045
```

Maxima zobrazuje štandardne 16 cifier desatinných čísel (medzi nimi sa ešte zobrazuje desatinná bodka). Zmenou hodnoty premennej `fpproc` môžeme dosiahnuť inú presnosť, ktorá sa však prejaví len pri použití výstupu `bfloat` (Big Float, pozri obr. 6), nie `float` – ten zobrazuje vždy rovnako. Presnosť môžeme prakticky neohraničene zvýšiť aj znížiť. Hodnotu `fpprec` je možné zmeniť aj lokálne (pozri riadok (`%i23`)) v rámci jedného príkazu[7]:

```
(%i20) fpprec:40;
(%o20) 40
(%i21) %pi,bfloat;
(%o21) 3.1415926535897932384626433832795028841970b0
(%i22) float(%pi);
(%o22) 3.141592653589793
(%i23) bfloat(%pi), fpprec=13;
(%o23) 3.14159265359b0
(%i24) %e, bfloat;
(%o24) 2.7182818284590452353602874713526624977570b0
```

1.3.5 Komplexné čísla

Komplexné čísla môžeme zadať v algebrickom tvare použitím konštanty `%i` – imaginárnej jednotky programu MAXIMA:

```
(%i25) z1:2+3*%i;
(%o25)                          3 %i + 2
(%i26) z2:4-5*%i;
(%o26)                          4 - 5 %i
```

[7] Zhodou okolností by sme rovnaký výsledok získali pri nastavení `fpprec=12`, program MAXIMA po zaokrúhlení nevypísal poslednú nulu.

Výsledok súčinu dvoch čísel by sme radšej videli priamo v zjednodušenom tvare. Toto dosiahneme až použitím príkazu `expand` alebo príkazu `rectform`:

```
(%i27) z1*z2;
(%o27)                        (4 - 5 %i) (3 %i + 2)
(%i28) expand(%);
(%o28)                              2 %i + 23
(%i29) rectform(z1*z2);
(%o29)                              2 %i + 23
```

V prípade podielu dvoch komplexných čísel však pomôže už len príkaz `rectform`[8]:

```
(%i30) z1/z2;
                                    3 %i + 2
(%o30)                              --------
                                    4 - 5 %i
(%i31) expand(%);
                                  3 %i        2
(%o31)                          -------- + --------
                                4 - 5 %i   4 - 5 %i
(%i32) rectform(z1/z2);
                                   22 %i   7
(%o32)                             ----- - --
                                    41     41
```

Samozrejme máme k dispozícii reálnu a imaginárnu zložku komplexných čísel:

```
(%i33) realpart(z1);
(%o33)                                2
(%i34) imagpart(z1/z2);
                                     22
(%o34)                               --
                                     41
```

Komplexné číslo v algebrickom tvare môžeme vypísať v exponenciálnom tvare:

```
(%i35) z3:2+2*%i;
(%o35)                              2 %i + 2
(%i36) polarform(z3);
                                         %i %pi
                                         ------
                                   3/2     4
(%o36)                            2    %e
```

[8] Vyskúšajte samostatne, ako v prípade súčinu aj podielu komplexných čísel funguje funkcia `trigrat`.

Príkazmi `abs` a `carg` získame absolútnu hodnotu (modul, veľkosť) a argument komplexného čísla:

```
(%i37) abs(z3);
                                    3/2
(%o37)                             2
(%i38) carg(z3);
                                    %pi
(%o38)                              ---
                                     4
```

Ak zadáme komplexné číslo v exponenciálnom tvare, môže byť transformované na algebrický tvar, ak sa to dá urobiť presne (bez použitia funkcií sínus a kosínus). Všimnite si, ako sa číslo z_5 (s iracionálnymi zložkami) zmenilo na približné s racionálnymi zložkami po priradení v riadku `(%i42)`. Porovnajte ešte rozdiel pri použití funkcií `numer` a `float`:

```
(%i39) z4:3*exp(2*%i*%pi/3);
                                sqrt(3) %i   1
(%o39)                     3 (---------- - -)
                                   2        2
(%i40) rectform(z4);
                                 3/2
                                3    %i   3
(%o40)                          ------- - -
                                   2      2
(%i41) z5:exp(2*%i*%pi/17);
                                  2 %i %pi
                                  --------
                                     17
(%o41)                           %e
(%i42) z5:exp(2*%i*%pi/17),numer;
(%o42)             0.36124166618715 %i + 0.93247222940436
(%i43) z5;
(%o43)             0.36124166618715 %i + 0.93247222940436
(%i44) exp(2*%i*%pi/17),float;
                           0.11764705882353 %i %pi
(%o44)                    %e
```

1.3.6 Operátory priraďovania

Operátor : používame na priradenie hodnôt alebo výrazov premenným. Týmto spôsobom však nedefinujeme funkcie:

```
(%i45) y:x^2-x+1;
                                   2
```

```
(%o45)                          x  - x + 1
(%i46) y(2);
               2
y evaluates to x  - x + 1
Improper name or value in functional position.
-- an error.  Quitting.  To debug this try debugmode(true);
```

Ako vidíme, $y = x^2 - x + 1$ nie je funkcia, je to len výraz. Ak chceme získať hodnotu výrazu napr. pre $x = 2$, musíme do výrazu y dosadiť číslo 2 namiesto x:

```
(%i47) subst(2,x,y);
(%o47)                                3
```

Treba si dávať pozor na poradie premenných vo funkcii `subst`. Porozmýšľajte, čo bude výsledkom príkazov `subst(y,x,2)` alebo `subst(x,2,y)`:

```
(%i48) subst(y,x,2);
(%o48)                                2
(%i49) subst(x,2,y);
                                  x
(%o49)                           x  - x + 1
```

Ak chceme vytvoriť funkciu argumentu x, musíme použiť priradenie s operátorom `:=`. Napríklad:

```
(%i50) f(x):=x^2-x+1;
                                      2
(%o50)                      f(x) := x  - x + 1
(%i51) f(2);
(%o51)                                3
```

1.3.7 Funkcie

MAXIMA, podobne ako Mathematica alebo Maple, má oveľa väčší počet rôznych funkcií ako štandardné programovacie jazyky. Ich zoznam je uvedený na stranách 441–451 v knihe (MAXIMA MANUAL, 2005), v ktorej nájdete aj ich popis. Nižšie uvádzame len niektoré z nich. Nezabudnite, že na získanie cenných informácií o príkazoch a funkciách môžete použiť funkcie `describe` a `example`.

Práca s reálnymi číslami

Na prácu s reálnymi číslami má MAXIMA nasledujúce:

funkcie: `bffac`, `bfloat`, `bfloatp`, `bfpsi`, `bfpsi0`, `cbffac`, `float`, `floatnump`, `?round`, `?truncate` a

premenné: `algepsilon`, `bftorat`, `bftrunc`, `float2bf`, `fpprec`, `fpprintprec`.

Popis nájdete v kapitole 10 knihy (MAXIMA MANUAL, 2005).

Trigonometrické funkcie

MAXIMA pozná nasledujúce *trigonometrické* funkcie: `acos`, `acosh`, `acot`, `acoth`, `acsc`, `acsch`, `asec`, `asech`, `asin`, `asinh`, `atan`, `atan2`, `atanh`, `cos`, `cosh`, `cot`, `coth`, `csc`, `csch`, `sec`, `sech`, `sin`, `sinh`, `tan`, `tanh`. Na manipuláciu s výrazmi obsahujúcimi trigonometrické funkcie slúžia funkcie `trigexpand`, `trigreduce`, `trigsimp`, `trigrat`. Balíky `atrig1`, `ntrig` a `spangl` obsahujú ďalšie pravidlá zjednodušovania trigonometrických funkcií. Ak sa chcete dozvedieť viac, prečítajte si kapitolu 15 knihy (MAXIMA MANUAL, 2005).

MAXIMA pozná aj niektoré hodnoty trigonometrických funkcií. Napríklad:

```
(%i52) ratsimp(sqrt((1-cos(%pi/4))/2));
                                sqrt(sqrt(2) - 1)
(%o52)                          -----------------
                                        3/4
                                       2
(%i53) tan(%pi/8);
(%o53) tan(%pi/8)
(%i54) ratsimp(%);
(%o54) tan(%pi/8)
(%i55) load(spangl);
(%o55) C:/PROGRA~1/MAXIMA~1.0/share/maxima/5.10.0/share/
         /trigonometry/spangl.mac
(%i56) tan(%pi/8);
(%o56) sqrt(2)-1
```

Niekedy by sme očakávali, že MAXIMA vydá výsledok trigonometrickej funkčnej hodnoty v algebrickom tvare (čo je síce krajšie a vhodné na ďalšie úpravy, ale možno nie jednoduchšie z hľadiska výpočtovej náročnosti). Napríklad:

```
(%i57) sin(a/2);
(%o57) sin(a/2)
(%i58) sin(%pi/8);
(%o58) sin(%pi/8)
```

Nasledujúcim príkazom sprístupníme použitie vzorcov pre polovičný uhol (treba si však uvedomiť, že vzorec je platný len pre určité uhly):

```
(%i59) halfangles:true$
(%i60) sin(a/2);
(%o60) sqrt(1-cos(a))/sqrt(2)
(%i61) sin(%pi/8);
(%o61) sin(%pi/8)
(%i62) subst(%pi/4,a,%o60);
```

```
(%o62) sqrt(1-1/sqrt(2))/sqrt(2)
(%i63) radcan(%o62);
(%o63) sqrt(sqrt(2)-1)/2^(3/4)
```

Hoci program MAXIMA už „pozná“ vzorce pre polovičné uhly, nenapadne ho, že ich má v riadku 61 použiť.[9] Poradíme si však tak, že použijeme substitúciu za a v riadku 62. Na záver sme ešte použili jeden zo základných spôsobov zjednodušovania výrazov obsahujúcich odmocniny (radikály).

Ak by ste pracovali s trigonometrickými funkciami, je potrebné naštudovať ďalšie bohaté možnosti, na začiatok môžeme odporučiť knihu (MAXIMA MANUAL, 2005).

Špeciálne funkcie

Hoci špeciálne funkcie samozrejme tvoria súčasť programu MAXIMA, nebudeme sa venovať ich popisu. Táto tématika vychádza za rámec tejto úvodnej učebnice. Ak na svoju prácu potrebujete špeciálne funkcie, otvorte kapitoly 16 a 18 knihy (MAXIMA MANUAL, 2005).

1.3.8 Prostredie `ev`

Všetky operácie programu MAXIMA sa uskutočňujú v nejakom prostredí, v ktorom systém predpokladá platnosť určitých podmienok, ktoré sa dajú meniť. Často potrebujeme zmeniť správanie systému pri nejakých výpočtoch bez toho, aby sme vykonali globálne zmeny. Na to MAXIMA poskytuje príkaz `ev`[10], ktorý je jedným z najvýkonnejších a ktorý umožňuje definovať lokálne prostredie v rámci jedného príkazu. Ak užívateľ zvládne prácu s funkciou `ev` čo najskôr, získa veľkú výhodu a úžitok. V tomto oddieli ukážeme len niektoré možnosti tohoto príkazu. Podľa príručky (RAND, 2005) má táto funkcia nasledujúcu syntax:

```
ev(a,b1,b2,...,bn)
```

Výraz a sa vyhodnotí pri platnosti podmienok b_1, b_2, ..., b_n. Týmito „podmienkami“ môžu byť aj rovnice, priradenia, slová (napríklad `numer` alebo `diff`).

Ako prvý uveďme jednoduchý príklad riešenia lineárnej algebrickej rovnice:

```
(%i64) ev(solve(a*x+b=0),x,a:3,b=12);
(%o64) [x=-4]
(%i65) a;
(%o65) a
```

Vidíme, že (rôzne) priradenia hodnôt premenným a a b vo vnútri prostredia `ev` sú len lokálne. V nasledujúcom riadku 65 nemá premenná a priradenú hodnotu.

Nasledujúci príklad je prevzatý z príručky (SOUZA, FATEMAN, MOSES a YAPP, 2004), kde nájdete aj ďalšie príklady:

[9] Pri príprave textu sa nám podarilo dosiahnuť aj stav, keď program MAXIMA samostatne upravil výraz $\sin(\pi/8)$. Nepodarilo sa nám však tento stav zopakovať znova.

[10] `ev` je asi skratka od evaluate.

```
(%i66) a:9/4;
(%o66) 9/4
(%i67) exp(a);
(%o67) %e^(9/4)
(%i68) ev(exp(a),float);
(%o68) 9.487735836358526
```

a v podobnom duchu ďalej:

```
(%i69) ev(exp(a*x));
(%o69) %e^((9*x)/4)
(%i70) ev(exp(a*x),float);
(%o70) %e^(2.25*x)
(%i71) ev(exp(a*x),x=2);
(%o71) %e^(9/2)
(%i72) ev(exp(a*x),numer,x=2);
(%o72) 90.01713130052181
```

1.3.9 Funkcie `assume` a `forget`

V niektorých situáciách je dobré predpokladať splnenie určitých podmienok. Pomocou príkazu `assume` – „predpokladajme“ – oznámime programu MAXIMA potrebnú informáciu.[11] Syntax príkazu je

```
assume(predikát_1,predikát_2,...,predikát_n)
```

Predikáty `predikát_1`, `predikát_2`, ..., `predikát_n` môžu byť len výrazy zadávané pomocou relačných operátorov `<`, `<=`, `equal`, `notequal`, `>=` a `>`.[12]

Platnosť predpokladu zadaného príkazom `assume(predikát)` zrušíme príkazom `forget(predikát)`:

```
(%i73) sqrt(a^2);
(%o73) |a|
(%i74) assume(a<0);
(%o74) [a<0]
(%i75) assume(a>=0);
(%o75) [inconsistent]
(%i76) assume(a<=0);
(%o76) [redundant]
(%i77) sqrt(a^2);
(%o77) -a
(%i78) log(a^2);
(%o78) 2*log(a)
(%i79) log(-1),numer;
(%o79) 3.141592653589793*%i
```

[11] V nápovede SPA MAXIMA sa odporúča zoznámiť sa aj s pojmami `is`, `facts`, `context` a `declare`.

[12] Podmienku $n \neq -1$ zadáme v tvare `notequal(n,-1)`, podobne sa použije operátor `equal`.

```
(%i80) float(log(1+%i));
(%o81) 0.78539816339745*%i+0.34657359027997
(%i81) forget(a<0)$
(%i82) sqrt(a^2);
(%o82) |a|
```

Riadky 75 a 76 ukazujú reakciu SPA MAXIMA na pokus predefinovať už existujúci predpoklad. Zmenu predpokladu môžeme uskutočniť až po jeho „zabudnutí“ príkazom `forget` – zabudni. Riadok 78 svedčí o tom, že MAXIMA neoveruje podmienku kladnosti vstupného argumentu logaritmu. Vysvetlením môže slúžiť riadok 79. Program MAXIMA totiž dokáže pracovať s logaritmami záporných (ale aj komplexných – pozri riadok 80) čísel.[13]

1.3.10 Príkaz `declare`

Podobnú funkciu ako príkaz `assume` plní príkaz `declare(a1,v1,a2,v2, ...)`, kde „atóm“ `ai` má vlastnosť `vi`. Tento príkaz umožňuje definovať veľký počet rôznych vlastností nielen pre premenné, pre ktoré uvedieme niekoľko príkladov. Zrušenie predpokladu (i premennej) môžeme uskutočniť príkazom `kill`.[14]

```
(%i83) (-1)^n;
(%o83) (-1)^n
(%i84) declare(n,even)$
(%i85) (-1)^n;
(%o85) 1
(%i86) declare(n,odd)$
Inconsistent Declaration: declare(n,odd)
-- an error.  Quitting.  To debug this try debugmode(true);
(%i87) kill(n)$
(%i88) declare(n,odd)$
(%i89) (-1)^n;
(%o89) -1
(%i90) kill(n)$
(%i91) limit(sin(%pi*n),n,inf);
(%o91) ind
(%i92) declare(n,integer)$
(%i93) limit(sin(%pi*n),n,inf);
(%o93) 0
(%i94) abs((-1)^n);
(%o94) |(-1)^n|
```

Riadok 86 svedčí o tom, že deklarácia sa nedá zmeniť opätovným použitím príkazu `declare`. Hoci výsledok v riadku 93 je očakávaný pre celočíselnú hodnotu n, nie je jasné, prečo MAXIMA nedokáže určiť absolútnu hodnotu v riadku 94.

[13] Bez hlbšieho štúdia programu MAXIMA je nemožné povedať, ako sa dá zúžiť oblasť vstupných argumentov logaritmu len na kladné reálne čísla, aby sme získali „klasický“ logaritmus.

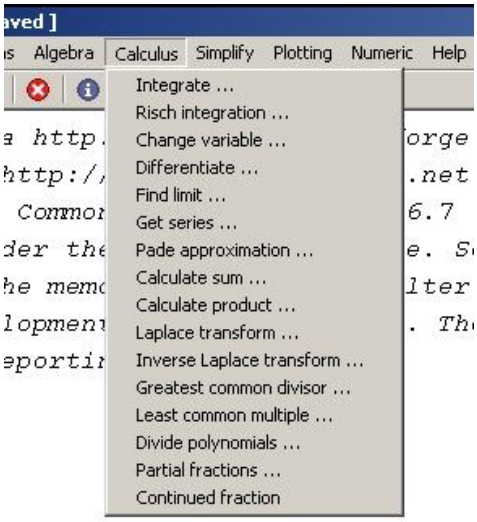
[14] Iný spôsob zrušenia platnosti deklarácie sme zatiaľ nenašli.

1.4 Riešenie niektorých úloh matematickej analýzy

Medzi úlohy matematickej analýzy patria:

- výpočet limít,
- derivovanie funkcií,
- výpočet Taylorovho polynómu funkcií,
- vyšetrovanie priebehu funkcií vrátane zobrazenia ich grafov,
- integrovanie funkcií,
- vyšetrovanie extrémov funkcií viacerých premenných,
- riešenie diferenciálnych rovníc,
- posudzovanie konvergencie číselných radov,
- Laplaceova transformácia.

Funkcie, určené na riešenie štandardných úloh, nájdeme v položke menu `Calculus` (pozri obrázok 7).



Obrázok 7: Položka `Calculus` hlavného menu

V tomto oddiele ukážeme, ako sa pomocou SPA MAXIMA riešia niektoré úlohy matematickej analýzy. Podrobnejšie je riešenie úloh matematickej analýzy popísané v učebnice (BUŠA, 2006).

1.4.1 Riešenie obyčajných diferenciálnych rovníc

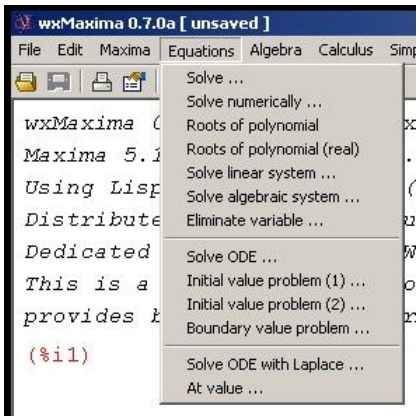
MAXIMA poskytuje niekoľko funkcií na riešenie diferenciálnych rovníc podobne ako, napríklad, Maple (pozri knihy (ĎJAKONOV, 2003; KREYSZIG a NORMINTON, 2006)). V tomto oddiele budeme využívať najmä príručku (SOUZA, FATEMAN, MOSES a YAPP, 2004).

Základ výbavy na riešenie diferenciálnych rovníc tvoria dve funkcie:

`ode2` – rieši obyčajné diferenciálne rovnice 1. a 2. rádu;

`desolve` – rieši systémy obyčajných lineárnych diferenciálnych rovníc s konštantnými koeficientmi, na riešenie MAXIMA využíva Laplaceovu transformáciu.

V prostredí WXMAXIMA sa ku funkciám na riešenie diferenciálnych rovníc dostaneme cez položky menu `Equations/Solve ODE` a pod. (pozri obrázok 8).



Obrázok 8: Ponuka menu `Equations`

1.4.2 Riešenie začiatočných úloh pre diferenciálne rovnice 1. rádu

MAXIMA dokáže riešit' nasledujúce typy diferenciálnych rovníc 1. rádu:[15]

- separovateľné,
- homogénne,
- lineárne,
- Bernoulliho,
- exaktné,
- zovšeobecnene homogénne.

Na zápis diferenciálnej rovnice, napríklad

$$x^2 y' + 3\,x\,y = \frac{\sin x}{x}, \tag{1}$$

existujú tri rôzne spôsoby, uvedené v knihe (SOUZA, FATEMAN, MOSES a YAPP, 2004). My uvedieme len dva z nich, ktoré autori odporúčajú používat' pre funkcie `ode2` a `desolve`.

Prvý z nich je vhodný na používanie vo funkcii `ode2`:

```
(%i1) depends(y,x);
(%o1) [y(x)]
(%i2) x^2*diff(y,x)+3*x*y=sin(x)/x;
```

[15] Viac príkladov nájdete v učebnici (BUŠA, 2006).

(%o2) $$x^2\left(\frac{\mathrm{d}}{\mathrm{d}\,x}\,y\right)+3\,x\,y=\frac{\sin x}{x}$$

Druhý spôsob je odporúčaný na použitie s funkciou `desolve`:

```
(%i3) x^2*diff(y(x),x)+3*x*y=sin(x)/x;
```

(%o3) $$3\,x\,y+x^2\left(\frac{\mathrm{d}}{\mathrm{d}\,x}\,y\,(x)\right)=\frac{\sin x}{x}$$

Predpokladajme, že sme na zadanie rovnice využili riadky 1–2. Môžeme pristúpiť k riešeniu pomocou funkcie `ode2`:[16]

```
(%i4) ode2(%o2,y,x);
```

(%o4) $$y=\frac{\%c-\cos x}{x^3}$$

Vidíme, že vo výsledku sa objavila **konštanta** `%c`.

MAXIMA nás môže informovať o type rovnice, ktorú sme zadali:

```
(%i5) method;
(%o5) linear
```

Zadaná rovnica (1) je teda lineárna.

Poznámka 1.1 Ak MAXIMA nedokáže vyriešiť zadanú rovnicu, ako výsledok dostaneme `false`.

Ak chceme získať jedno partikulárne riešenie, môžeme zvoliť hodnotu konštanty `%c`. Treba si však uvedomiť, že výsledok je rovnosť a ak chceme získať funkciu, musíme použiť len pavú stranu:

```
(%i6) depends(g,x)$
(%i7) g:subst(1,%c,last(%o4));
```

(%o7) $$\frac{1-\cos x}{x^3}$$

Teraz môžeme overiť správnosť riešenia:

```
(%i8) ratsimp(x^2*diff(g,x)+3*x*g);
```

(%o8) $$\frac{\sin x}{x}$$

Pomocou funkcie `ic1` dokážeme riešiť **Cauchyho začiatočnú úlohu pre diferenciálnu rovnicu 1. rádu**.

Príklad 1 .1 Určme riešenie diferenciálnej rovnice

$$y'+x\,y=x\,y^2, \tag{2}$$

vyhovujúce začiatočnej podmienke $y(1)=2$.

Riešenie. Rovnica (2) je zapísaná v tvare Bernoulliho rovnice. Pozrime sa, ako ju vidí MAXIMA:

[16] Dávajte pozor na použitie `(%oXX)` a nie `(%iXX)`! Ak však použijete ako argument funkcie `ode2` priamo vstup riadku `(%i2)`, všetko bude v poriadku.

```
(%i9) depends(y,x)$
(%i10) rov:diff(y,x)+x*y=x*y^2;
(%o10) 'diff(y,x,1)+x*y=x*y^2
(%i11) ries:ode2(rov,y,x);
(%o11) log(y-1)-log(y)=x^2/2+%c
(%i12) method;
(%o12) separable
```

Rovnica (2) je teda separovateľná (ak na pravú stranu presunieme všetky členy okrem derivácie y, dá sa x vybrať pred zátvorku). Riešenie je zapísané v implicitnom tvare.

Pokračujme v riešení začiatočnej úlohy.

```
(%i13) riesz:ic1(ries,x=1,y=2);
```

(%o13)
$$\log(y-1)-\log y=\frac{x^2-2\log 2-1}{2}$$

Trochu SPA MAXIMA pomôžeme. Oddelíme ľavú a pravú stranu implicitného zadania riešenia začiatočnej úlohy a samostatne ich exponujeme:

```
(%i14) l:first(riesz)$
(%i15) r:last(riesz)$
(%i16) l:radcan(exp(l));
(%o16) (y-1)/y
(%i17) r:radcan(exp(r));
```

(%o17)
$$\frac{e^{\frac{x^2-1}{2}}}{2}$$

Zbavili sme sa logaritmov a môžeme úlohu doriešiť:[17]

```
(%i18) f:last(solve(l=r,y)[1]);
```

(%o18)
$$-\frac{2}{e^{\frac{x^2}{2}-\frac{1}{2}}-2}$$

Na záver ešte overme správnosť riešenia:

```
(%i19) diff(f,x)+x*f-x*f^2;
```

(%o19)
$$\frac{2x\,e^{\frac{x^2}{2}-\frac{1}{2}}}{\left(e^{\frac{x^2}{2}-\frac{1}{2}}-2\right)^2}-\frac{2x}{e^{\frac{x^2}{2}-\frac{1}{2}}-2}-\frac{4x}{\left(e^{\frac{x^2}{2}-\frac{1}{2}}-2\right)^2}$$

```
(%i20) radcan(%o19);
(%o20) 0
```

[17] Všimnite si, čo sa všetko vykonalo v riadku 18!

1.4.3 Riešenie začiatočných a okrajových úloh pre diferenciálne rovnice 2. rádu

MAXIMA rieši nasledujúce typy obyčajných diferenciálnych rovníc 2. rádu:

- lineárne s konštantnými a s nekonštantnými koeficientmi,
- exaktné,
- Eulerove,
- Besselove,
- rovnice bez závislej premennej,
- rovnice bez nezávislej premennej.

Podobne, ako v prípade rovníc 1. rádu, je možné riešiť **Cauchyho začiatočnú úlohu** ale v prípade rovníc 2. rádu je navyše možné riešiť aj **okrajovú úlohu**.

Príklad 1 .2 Určme riešenie diferenciálnej rovnice

$$x^2\,y'' + x\,y' - y = 0, \tag{3}$$

vyhovujúce začiatočným podmienkam $y(1) = 2$, $y'(1) = -3$.

Riešenie. Vidíme, že túto rovnicu môžeme zaradiť medzi Eulerove rovnice bez pravej strany tvaru

$$x^2\,y'' + a\,x\,y' + b\,y = 0.$$

Postupujeme analogicky s prípadom riešenia rovnice 1. rádu. Rozdielny je typ začiatočnej podmienky. Začiatočná úloha sa rieši pomocou funkcie `ic2`:

```
(%i21) ode2(x^2*diff(y,x,2)+x*diff(y,x)-y=0,y,x);
(%o21) y=%k2*x-%k1/(2*x)
(%i22) method;
(%o22) exact
(%i23) ic2(%o21,x=1,y=2,diff(y,x)=-3);
```

(%o23) $$y = \frac{5}{2\,x} - \frac{x}{2}$$

SPA MAXIMA teda rovnicu zaradil medzi exaktné.

Príklad 1 .3 Určme riešenie diferenciálnej rovnice (3) vyhovujúce okrajovým podmienkam $y(1) = 2$ a $y(2) = 1$.

Riešenie. V tomto prípade použijeme funkciu `bc2`:

```
(%i24) bc2(%o21,x=1,y=2,x=2,y=1);
```

(%o24) $$y = \frac{2}{x}$$

Príklad 1 .4 Riešme Besselovu diferenciálnu rovnicu $x^2\,y'' + x\,y' + (x^2 - 4)y = 0$.

Poznámka 1.2 Besselove rovnice majú tvar $x^2\,y'' + x\,y' + (x^2 - n^2)y = 0$, kde n je konštanta. Príklad sme prevzali z knižky (SOUZA, FATEMAN, MOSES a YAPP, 2004).

```
(%i25) ode2(x^2*diff(y,x,2)+x*diff(y,x)+(x^2-4)*y-0,y,x);
(%o25) y=bessel_y(2,x)*%k2+bessel_j(2,x)*%k1
```

alebo

(%o25) $$y = Y_2(x)\,\%k_2 + J_2(x)\,\%k_1$$

V zápise výsledku sa vyskytujú Besselove funkcie prvého resp. druhého druhu – J_n resp. Y_n.

1.4.4 Riešenie sústav lineárnych diferenciálnych rovníc s konštantnými koeficientmi

Pri zadaní sústavy lineárnych diferenciálnych rovníc s konštantnými koeficientmi, ktorú sa chystáme riešiť pomocou funkcie `desolve`, musia byť funkcionálne vzťahy explicitne vyznačené (MAXIMA MANUAL, 2005, strana 211). Napríklad:

```
(%i26) dr1:'diff(f(x),x)='diff(g(x),x)+sin(x)$
(%i27) dr2:'diff(f(x),x)+x^2-f(x)=2*'diff(g(x),x);
(%i28) desolve([dr1,dr2],[f(x),g(x)]);
(%o28) [f(x)=sin(x)-cos(x)+(f(0)-1)*%e^(-x)+x^2-2*x+2,
         g(x)=sin(x)+(f(0)-1)*%e^(-x)+x^2-2*x+g(0)-f(0)+1]
(%i29) subst(-3,g(0),subst(2,f(0),%o28));
(%o29) [f(x)=sin(x)-cos(x)+%e^(-x)+x^2-2*x+2,
         g(x)=sin(x)+%e^(-x)+x^2-2*x-4]
```

Ak vynecháme apostrofy `'`, ktoré potláčajú vykonanie derivácie, nič zvláštneho sa nestane. Všimnite si, že vo výsledku sú nedefinované hodnoty $f(0)$ a $g(0)$. V príručkách (MAXIMA MANUAL, 2005; SOUZA, FATEMAN, MOSES a YAPP, 2004) sa uvádza, že začiatočné hodnoty je možné zadať

1. len v bode $x = 0$,
2. len pred použitím funkcie `desolve` pomocou funkcie `atvalue`.

V riadku 29 sa nám podarila substitúcia za $f(0)$ a $g(0)$. Je ťažké povedať, čo je na nej nevyhovujúce.

Vyskúšajme príklad, uvedený v knižke (SOUZA, FATEMAN, MOSES a YAPP, 2004):

```
(%i30) atvalue(f(x),x=0,1)$
(%i31) atvalue(g(x),x=0,2)$
(%i32) atvalue(diff(g(x),x),x=0,3)$
(%i33) dr3:diff(f(x),x)=diff(g(x),x)+sin(x)$
(%i34) dr4:diff(g(x),x,2)=diff(f(x),x)-cos(x)$
(%i35) sol:desolve([dr3,dr4],[f(x),g(x)]);
(%o35) [f(x)=3*%e^x-2,g(x)=cos(x)+3*%e^x-2]
```

Zdá sa, že sme dospeli k podobnému výsledku, ako v predchádzajúcom príklade, kde sme vopred nepoužili príkaz `atvalue`. Nie je však jasné, ako by sa dala dosadiť napríklad hodnota $g'(0)$ pomocou príkazu `subst`. Po ďalšom overovaní sme zistili, že pomocou príkazu `atvalue` je nutné zadať len hodnoty potrebných derivácií v bode $x = 0$, funkčné hodnoty neznámych funkcií v bode $x = 0$ je možné dodať pomocou príkazu `subst` aj po vyriešení sústavy diferenciálnych rovníc.

Ako sa dostaneme ku funkciám $f(x)$ a $g(x)$? Nasledujúce riadky ukazujú, že tieto funkcie nie sú k dispozícii hneď po riešení sústavy. Je potrebné im priradiť výrazy na pravej strane rovností (%35). Ale aj po priradení sa funkcie správajú zvláštne, ako keby boli len výrazy (pozri riadok 41) – je možné ich derivovať, ale funkčné hodnoty získame len pomocou substitúcie:

```
(%i36) f(0);
(%o36) 1
(%i37) f(1);
(%o37) f(1)
(%i38) diff(f(x),x);
(%o38) 'diff(f(x),x,1)
(%i39) g(0);
(%o39) 2
(%i40) f(x):=last(sol[1])$
(%i41) f(1);
(%o41) 3*%e^x-2
(%i42) subst(1,x,f(x));
(%o42) 3*%e-2
(%i43) diff(f(x),x);
(%o43) 3*%e^x
(%i44) g(x):=last(sol[2])$
(%i45) diff(g(x),x,3);
(%o45) sin(x)+3*%e^x
(%i46) float(subst(3,x,g(x)));
(%o46) 57.26661827296256
(%i47) float(subst(3,x,diff(g(x),x)));
(%o47) 60.11549076150314
```

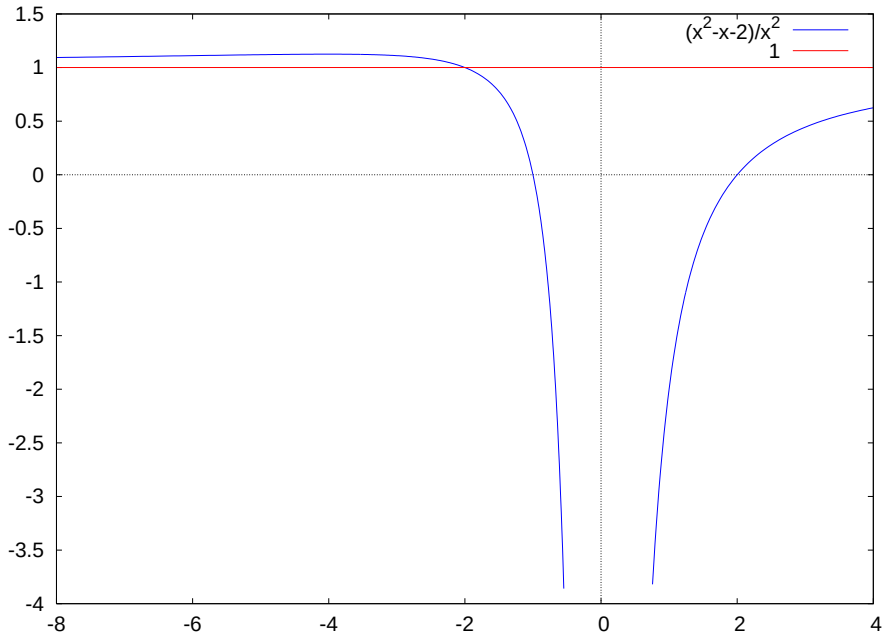
1.4.5 Zobrazenie grafu funkcie jednej premennej

Príklad 1 .5 Znázornime priebeh funkcie

$$f(x) = \frac{x^2 - x - 2}{x^2}.$$

Dá sa ukázať, že graf funkcie stačí zobraziť na intervale $\langle -8, 4\rangle$, na ktorom sa nachádzajú všetky nulové body, bod nespojitosti, lokálne maximum aj inflexný bod. Hodnoty y môžeme zvoliť, napríklad, z intervalu $\langle -4, 4\rangle$, ktorý zrejme obsahuje všetky „zaujímavé“ funkčné hodnoty. Spolu s grafom funkcie $f(x)$ zobrazíme aj asymptotu so smernicou a osi. Predtým, ako graf uložíme do súboru, môžeme si ho pozrieť na obrazovke programu Gnuplot:

```
(%i48) plot2d([f(x),p(x)], [x,-8,4], [y,-4,4],
       [gnuplot_preamble, "set zeroaxis;"],
       [gnuplot_term, ps], [gnuplot_out_file,
       "D:/Users/Busa/Kega/Maxima/TeX/fx.eps"])$
```

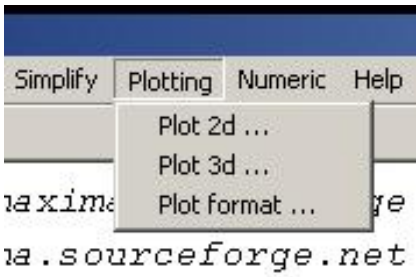


Obrázok 9: Graf funkcie $(x^3 - x - 2)/x^2$

Na obrázku 9 je znázornený graf funkcie $f(x)$.[18]

1.4.6 Zobrazenie grafu funkcie dvoch premenných

V prostredí WXMAXIMA sa nemusíme obávať vytvorenia grafu funkcie dvoch premenných. Môžeme znova využiť menu, tentoraz klikneme na položku `Plotting`, zobrazenú na obrázku 10.



Obrázok 10: Položka menu `Plotting`

Využijeme položku `Plotting/Plot 3D` a zadáme vstupné údaje. Ďalej už môžeme modifikovať údaje aj v príkazovom riadku.

Príklad 1 .6 Zobrazme funkciu $f(x,y) = \ln x - y^2 - y/x$ v okolí stacionárneho bodu $(x^*, y^*) = (1/\sqrt{2}, -1/\sqrt{2})$.

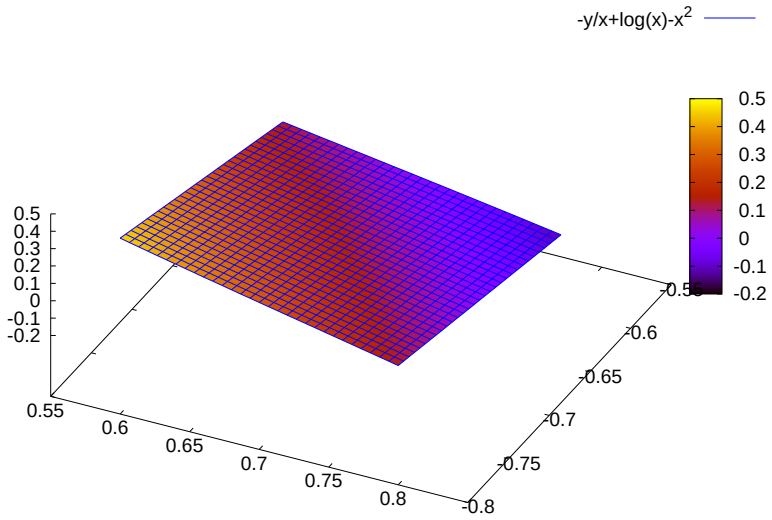
Riešenie. Po zadaní funkcie pomocou menu sme získali zobrazenie grafu. Všimli sme si, že tento graf je možné rotovať a našli sme vhodný pohľad. Ak si uvedomíme, že MAXIMA pri vytváraní grafov

[18] Z neznámeho dôvodu je maximálna hodnota na osi y rovná 1.5 a nie 4, ako sme zadali :(!

spolupracuje s programom Gnuplot, je jasné, že nastal čas nazrieť do učebnice Gnuplotu (DOBOŠ, 2006). Na strane 35 objavíme príklad použitia príkazu `view`.

```
(%i49) plot3d(log(x)-x^2-y/x, [x,0.6,0.8], [y,-0.8,-0.6],
   [gnuplot_preamble, "set view 37,26"], [gnuplot_term, ps],
[gnuplot_out_file, "D:/Users/Busa/Kega/Maxima/logxyn.eps"])$
```

Na obrázku 11 sa môžeme uistiť, že stacionárny bod $(1/\sqrt{2}, -1/\sqrt{2}) \approx (0{,}707; 0{,}707)$ nie je bodom lokálneho extrému ale môže byť len sedlovým bodom.



Obrázok 11: Graf funkcie $f(x,y) = \ln x - y^2 - y/x$

Príklad 1 .7 Znázornime funkciu $f(x,y) = x^3 + y^3$ v okolí lokálneho minima pri väzbe $x + y - 4$.

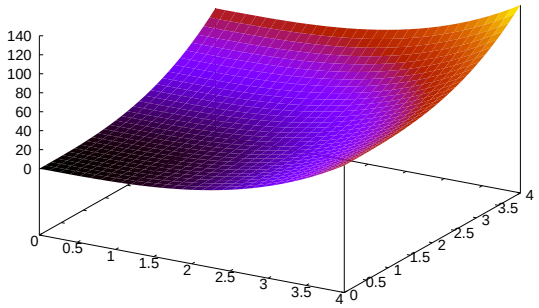
Riešenie. Postupujeme podobne ako pri riešení predchádzajúceho príkladu:

```
(%i50) plot3d(x^3+y^3, [x,0,4], [y,0,4],
[gnuplot_preamble,"set pm3d at s;unset surf;unset colorbox"],
[gnuplot_term, ps],
[gnuplot_out_file, "D:/Users/Busa/Kega/Maxima/x3py3.eps"])$
```

Väzba je rovnica priamky $x + y - 4 = 0$, ktorá v rovine $\mathcal{O}_{xy}$ spája body $(0,4)$ a $(4,0)$. Ako je vidieť na obrázku 12, nad touto priamkou nadobúda funkcia $f(x,y)$ najmenšiu hodnotu nad bodom $(2,2)$. Ak porovnáme obrázok 12 s obrázkom 11, uvidíme, že chýba farebná škála funkčných hodnôt. Toto sme dosiahli pridaním príkazu `unset colorbox`.

Ešte lepší pohľad získame, ak zobrazíme vrstevnice funkcie $f(x,y)$.

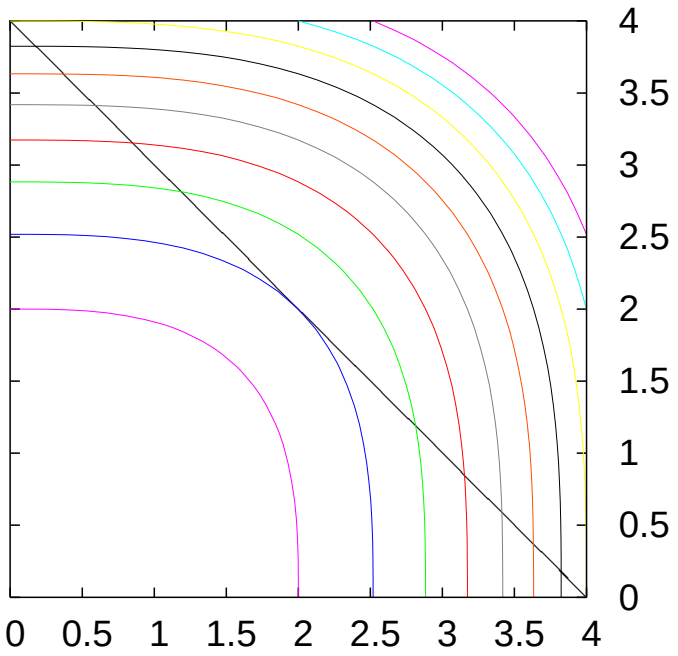
```
(%i51) plot3d(x^3+y^3, [x,0,4], [y,0,4],
       [gnuplot_preamble, "unset pm3d; set contour base;
       set cntrparam levels incremental 0,8,80; set nokey;
```



Obrázok 12: Graf funkcie $f(x,y) = x^3 + y^3$

```
set size square; unset surf; set view 0,0],
[gnuplot_term, ps], [gnuplot_out_file,
"D:/Users/Busa/Kega/Maxima/vrstevnice.eps"])$
```

V tomto príklade sme vypli farby voľbou `unset pm3d`, nastavili výstup grafu na typ izolínií voľbou `set contour base`, nastavili sme hodnoty vrstevníc, ktoré sme zobrazili – `set cntrparam levels incremental 0,8,80`, vypli sme legendu príkazom `set nokey`, nastavili sme pomer jednotiek osí x a y na 1:1 príkazom `set size square`, apod.[19] Časť priamky $x + y - 4 = 0$ (uhlopriečku štvorca) sme dokreslili samostatne. Výsledok vidíte na obrázku 13. Minimálna hodnota pri väzbe sa dosahuje tam, kde sa priamka daná väzbou dotýka druhej izolínie, ktorej odpovedá hodnota $f = 16$.



Obrázok 13: Izolínie funkcie $f(x,y) = x^3 + y^3$

[19] Popri učebnici (DOBOŠ, 2006) sme čerpali z podrobnejšej príručky (KAWANO, 2005).

Záver

Milá čitateľka, vážený čitateľ!

Nastal čas na rozlúčku s touto kapitolou. Dúfame, že Vás možnosti systému počítačovej algebry MAXIMA zaujali a stanete sa jeho používateľkou/používateľom.

Veríme, že aj tých niekoľko príkladov, ktoré sme v tejto kapitole uviedli, svedčí o veľkom potenciále programu MAXIMA. Snažili som sa ukázať, že SPA MAXIMA je vhodný nástroj na využitie vo výskume a vo výučbe (najmä) matematických predmetov, predovšetkým matematickej analýzy, lineárnej algebry i matematickej štatistiky. Dá sa použiť aj na výučbu numerických metód. Na druhej strane na riešenie úloh z vymenovaných oblastí s výnimkou matematickej analýzy je možné a vhodné použiť iné OPENSOURCE programy – Pylab, Scilab, Octave (KAUKIČ, 2006; PRIBIŠ, 2006; BUŠA, 2006), R a pod.

Vôbec sme sa nedotkli ďalších oblastí použitia programu MAXIMA, medzi nimi napríklad teórie čísel, kombinatoriky a iných. Snáď sa do toho pustí niekto ďalší, komu môžu priniesť použitie SPA MAXIMA v týchto oblastiach väčší úžitok. Pripomeňme ešte, že príkazy programu MAXIMA sú dostatočne popísané napríklad v knihe (MAXIMA MANUAL, 2005).

Použitá literatúra

BUŠA, J. 2006. *Maxima. Open source systém počítačovej algebry*, ISBN 80-8073-640-5, online na `http://people.tuke.sk/jan.busa/kega/maxima/`.

BUŠA, J. 2006. *Octave. Rozšírený úvod*, ISBN 80-8073-596-4, online na `http://people.tuke.sk/jan.busa/kega/octave/`.

DOBOŠ, J. 2006. *GNUPLOT*, ISBN 80-8073-637-5, 52 s., online na `http://people.tuke.sk/jan.busa/kega/gnuplot/`.

ĎJAKONOV, V. P. 2003. *Maple 8 v matematike, fizike i obrazovanii*, Moskva, SOLON-Press, ISBN 5-98003-038-7, 656 s.

GLÖCKNER, R. 2006. *Einführung in Maxima*, 23 s.

GRÄBE, H.-G. 2005. *Skript zum Kurs Einführung in das symbolische Rechnen*, Wintersemester 2004/2005, Institut für Informatik, Leipzig, 158 s., `http://www.informatik.uni-leipzig.de/~graebe`.

Micro introduction into Maxima, Computing Harvard Math department, online na `http://www.math.harvard.edu/computing/maxima/`.

KAWANO, T. 2005. *Gnuplot – not so Frequently Asked Questions*, `http://t16web.lanl.gov/Kawano/gnuplot/index-e.html`.

KAUKIČ, M. 2006. *Základy programovania v Pylabe*, ISBN 80-8073-635-9, `http://people.tuke.sk/jan.busa/kega/pylab/`.

KREYSZIG, E. – NORMINTON, E. J. 2006. *Maple Computer Guide, A Self-Contained Introduction*, for E. Kreyszig Advanced Engineering Mathematics, 9. vydanie, John Wiley & Sons, Inc., ISBN 0-471-72645-1, 300 s.

2006. *Maxima Manual*, 461 s.

2006. *Maxima Manual*, online na `http://maxima.sourceforge.net/docs/manual/en/maxima.html`

PRIBIŠ, J. 2006. *Scilab*, ISBN 80-8073-655-3, `http://people.tuke.sk/jan.busa/kega/scilab/`.

RAND, R. H. 2005. *Introduction to Maxima*, Cornel University.

SOUZA, P. N. DE – FATEMAN, R. J. – MOSES, J. – YAPP, C. 2004. *The Maxima Book*, 154 s.

STOPKA, M. *Hra s písmenky wxMaxima*, ABC Linuxu, 18. 5. 2006, online na `http://www.abclinuxu.cz/clanky/programovani/hra-s-pismenky-wxmaxima`.

2 NIEKOĽKO RÁD POUŽÍVATEĽOM LaTeXU

Ján BUŠA a **Ladislav ŠEVČOVIČ**
Katedry matematiky a fyziky, FEI
Technická univerzita v Košiciach

2.1 Úvod

Publikovanie výsledkov svojej práce je neoddeliteľnou súčasťou činnosti učiteľa, vedeckého pracovníka alebo výskumníka. Popri obsahovej stránke je potrebné venovať pozornosť aj estetickej a typografickej kvalite dokumentov. Už takmer 30 rokov slúži na tento účel OpenSource program TeX. Napriek jeho pokročilému veku nie je dôvod hľadať modernejšie programy.[20] Systém LaTeX už dve desaťročia uľahčuje písanie textov „obyčajným“ používateľom, ktorí nemajú chuť alebo čas zahĺbiť sa do tajov programovacieho jazyka TeX. Je len logické, že sa postupne zvyšuje počet vedeckých časopisov, ktoré prijímajú články v niektorom TeXovskom formáte.

Predpokladáme, že čitateľ je oboznámený so základmi používania LaTeXu na úrovni príručky *Nie príliš stručný úvod do systému LaTeX 2ε* (Oetiker, Partl, Hyna a Schlegl, 2002) alebo učebnice *LaTeX pro začátečníky* (Rybička, 2003). V tejto kapitole uvedieme informácie, ktoré trochu rozšíria poznatky uvedené v týchto publikáciách. Pri jej písaní sme sa okrem rôznych internetovských zdrojov opierali o knižku *LaTeX – podrobný průvodce* (Kopka a Daly, 2004) a najmä o ruský preklad vynikajúcej trilógie *The LaTeX [Web, Graphics] Companion* (Goossens, Mittelbach a Samarin, 1998; Goossens a Rahtz, 1999; Goossens, Rahtz a Mittelbach, 1999). V jednotlivých oddieloch uvedieme naše poznatky o práci s grafikou, o vytváraní zoznamu skratiek a symbolov, o citovaní použitej literatúry a o využití databázového programu BiBTeX. Stručne predstavíme použitie LaTeXovského editora KILE určeného pre OS GNU/Linux.

[20] Porovnateľné s TeXom z hľadiska typografickej kvality sú iba profesionálne typografické programy, ktorých kvalite odpovedá samozrejme aj vysoká cena.

2.2 Práca s grafickými objektmi

Grafická informácia zohráva čím ďalej tým dôležitešiu úlohu pri zdieľaní a vymieňaní vedeckých skúseností a poznatkov. Preto je dôležité, aby bola práca s grafikou čo najjednoduchšia.

2.2.1 Použitie balíčka graphicx

V príručke *Nie príliš stručný úvod do systému LaTeX 2ε* (Oetiker, Partl, Hyna a Schlegl, 2002) je oddiel venovaný zaraďovaniu grafiky vo formáte `eps` – *Encapsulated PostScript* – do LaTeXovského dokumentu. Jednou z možností je použitie balíka graphicx, ktorého autorom je D. P. Carlisle. Podrobne sú postup zaraďovania grafiky a ďalšie detaily popísané v manuáli *Using Imported Graphics in LaTeX and pdfLaTeX* (Reckdahl, 2006). Táto príručka má 124 strán a je prirodzené, že sa ani nepokúsime ju tu popísať.

Balík graphicx sa ativuje príkazom

```
\usepackage{graphicx}
```

alebo s voľbami podľa typu kompilátora:

`\usepackage[dvips]{graphicx}`, resp. `\usepackage[pdftex]{graphicx}`

pre [cs]LaTeX s následným použitím programu `dvips`, resp. pre pdf[cs]LaTeX. V prvom prípade je možné zaraďovanie „postscriptových“ obrázkov, v prípade použitia pdf[cs]LaTeXu[21] je možné zaraďovanie obrázkov v grafických formátoch `pdf`, `png`, `jpg` a `mps`.

Na vloženie obrázku `súbor.ext` do dokumentu použite príkaz

`\includegraphics}[`*kľúč=hodnota*`,...]{súbor.ext}`

Za voliteľné parametre sa berie zoznam čiarkami oddelených *kľúčov* a ich hodnôt. *Kľúče* sa môžu použiť na zmenu šírky, výšky a otáčanie vkladanej grafiky. Tabuľka 1 uvádza zoznam najdôležitejších kľúčov.

Tabuľka 1: Názvy kľučov pre balík graphicx

`width`	zmení obrázok na danú šírku
`height`	zmení obrázok na danú výšku
`angle`	otočí obrázok v smere hodinových ručičiek
`scale`	zmení mierku obrázka

Veľkosť uhla sa zadáva v stupňoch, na zadanie šírky sa dajú použiť aj relatívne dĺžkové jednotky, napríklad príkazom:

```
\includegraphics[width=0.6\textwidth]{súbor.mps}
```

nastavíme šírku obrázka vytvoreného programom METAPOST na 60 % šírky textu. Pri jej zmene sa automaticky zmení aj šírka vloženého obrázka. Neodporúča sa používať súčasne nastavenie šírky aj výšky, ak ovšem nechcete vložený obrázok deformovať.

Ak použijeme príkaz `\DeclareGraphicsExtensions{.mps,.pdf,.png,.jpg}` súčasne s voľbou `pdftex`, tak v príkaze `\includegraphics` nie je potrebné zadávať rozšírenie názvu vkladaného súboru. Ak máme pripravené množstvo obrázkov vo formáte `eps` a zároveň v ostatných uvedených formátoch, potom len zmenou deklarácie rozšírení grafických súborov môžeme prejsť od používania LaTeXu ku pdfLaTeXu a naopak.

[21] Momentálne mnou preferovaná možnosť.

2.2.2 Ďalšie možnosti balíka graphicx

Balík graphicx poskytuje aj nasledujúce tri príkazy:

```
\scalebox{h-škála}[v-škála]{obsah}
```

na zmenu veľkosti „boxu“, t. j. krabičky, pričom hodnota `h-škála/v-škála` určuje koľkokrát sa zväčší horizontálny/vertikálny rozmer krabičky.

```
\resizebox{šírka}{výška}{obsah}
\resizebox*{šírka}{celková výška}{obsah}
```

slúžia na nastavenie veľkosti obsahu krabičky na zadané rozmery. Na tomto mieste znamená celková výška súčet výšky a hĺbky boxu. Ak namiesto jednotky dĺžky zadáme `!` tak nastavíme len jeden rozmer, pričom druhý sa nastaví tak, aby bol zachovaný prirodzený pomer rozmerov obrázka. Napríklad, `\resizebox{2in}{!}{obsah}` nastaví šírku obrázka na veľkosť dvoch palcov.

```
\rotatebox[voľby]{uhol}{obsah}
```

spôsobí pootočenie obsahu boxu o uhol zadaný v stupňoch proti smeru hodinových ručičiek. Voľby umožňujú meniť polohu bodu, okolo ktorého sa rotácia uskutočňuje.

2.2.3 Prostredie `picture`

Na tomto mieste doplníme stručný popis prostredia picture.Podrobnejší popis nájdete v knižkách (Rybička, 2003; Goossens, Mittelbach a Samarin, 1998; Kopka a Daly, 2004). LaTeXovské prostredie picture umožňuje priamo v LaTeXu vytvárať jednoduché obrázky, pozostávajúce z rovných čiar, šípiek, kružníc a oválov (respektíve polkružníc, štvrťkružníc). Okrem toho vyrába Bezierove krivky. Toto prostredie je vhodné aj na umiestňovanie popisov a to aj popisov ku grafike, vytváranej mimo TeXu, napríklad k bitmapovým obrázkom. Takýmto spôsobom sa dosiahne jednota textu a popisov obrázkov. Nasledujúci obrázok 1 ilustruje uvedené možnosti. Pri použití príkazu `\bezier` je potrebný štýly `bezier.sty`.

Syntax prostredia je približne nasledujúca:

```
\begin{picture}(šírka,výška)(x_r, y_r)
\put(x,y){objekt}
\end{picture}
```

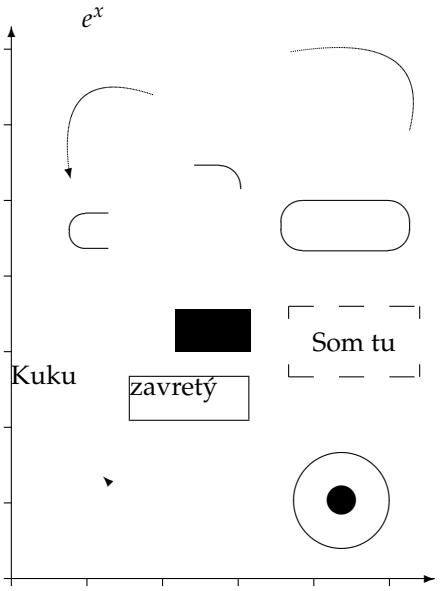
Parametre *šírka* a *výška* udávajú rozmery vytvoreného „boxu“. Zadanie $(0,0)$ spôsobí, že vytvorený objekt má nulový rozmer, teda TeX ostáva na mieste, kde bol pred zadaním prostredia `picture`. Nepovinné (x_r, y_r) sú súradnice „referenčného bodu“, teda bodu, v ktorom sa momentálne TeX nachádza, v súradnicovom systéme prostredia `picture`.

```
Uprostred slo\begin{picture}(0,0)(0,0)
\unitlength=1mm
\put(2,2.5)
    {\framebox(4,2.5)[tr]{$\bullet$}}
\end{picture}va môžem na chvíľu odísť.
```

Nasledujúci zdrojový text definuje obrázok 1:

```
\unitlength 1mm \linethickness{0.4pt}
\begin{picture}(56.00,73.00)
\put(0,0){\vector(1,0){56}} \put(0,0){\vector(0,1){73}} % osi x,y
```

```
\multiput(0,0)(10,0){6}{\line(0,-1){1}} % znacky osi x
\multiput(0,0)(0,10){8}{\line(-1,0){1}} % znacky osi y
\put(8.67,28.33){\makebox(0,0)[rt]{Kuku}}
\put(15.67,21.00){\framebox(15.67,5.67)[lt]{zavretý}}
\put(36.67,26.67){\dashbox{3.33}(17.33,9.33)[cc]{Som tu}}
\put(21.67,30.00){\rule{10.00\unitlength}{5.67\unitlength}}
\put(12.33,13.33){\vector(-1,1){0.2}}
\put(43.67,10.33){\circle{13.33}} \put(43.67,10.33){\circle*{4.00}}
\put(12.83,46.00){\oval(10.33,4.67)[l]}
\put(24.17,51.50){\oval(12.33,6.33)[rt]}
\put(44.17,46.67){\oval(17.00,6.67)[]}
\bezier{132}(52.67,59.33)(56.00,72.67)(37.00,69.67)
\bezier{116}(18.67,64.00)(5.67,68.33)(7.67,53.67)
\put(7.67,53.67){\vector(1,-4){0.2}} \put(9.33,73.00){$e^x$}
\end{picture}
```



Obrázok 1: Príklad použitia prostredia `picture`

2.2.4 Jazyk METAFONT/METAPOST

D. E. Knuth vytvoril jazyk METAFONT súčasne s TEXom v roku 1978 a zároveň napísal podrobnú učebnicu tohoto jazyka *The METAFONTbook* (Knuth, 1984). Jazyk METAFONT umožňuje popísať rôzne grafické symboly (pôvodne bol použitý na výrobu písiem). Pracuje s bodmi a krivkami, ktoré vznikajú rôznymi spôsobmi ich spojenia. Výborná je učebnica *Kreslíme METAFONTem* (Šedivý, Brož, Group, Píše a Houfek, 1998).

V čase vzniku METAFONTu ešte neexistoval jazyk PostScript. Po jeho vzniku adaptoval J. Hobby jazyk METAFONT tak, aby vytváral „postscriptový“ výstup. METAPOST je navyšed rozšírením METAFONTu najmä o prácu s farbou, umožňuje tiež umiestňovanie textu do obrázkov. Podrobný popis nájdete v príručke *A User's Manual for Metapost* (Hobby, 1992). V „rodnom“ jazyku si môžete

prečítať seriál METAPOST *a mfpic* (Krátká, 2001). V knižke *The LaTeX Graphics Companion* (Goossens, Rahtz a Mittelbach, 1999) nájdete krásne ukážky vytvorené METAPOSTom.

2.3 Bibliografické odkazy

V zozname použitej literatúry sa uvádzajú odkazy podľa normy STN ISO 690-2 (01 0197) (Informácie a dokumentácia. Bibliografické citácie. Časť 2: Elektronické dokumenty alebo ich časti, dátum vydania 1. 12. 2001, ICS: 01.140.20). Odkazy sa môžu týkať knižných, časopiseckých a iných zdrojov informácií (zborníky z konferencií, patentové dokumenty, normy, odporúčania, kvalifikačné práce, osobná korešpondencia a rukopisy, odkazy cez sprostredkujúci zdroj, elektronické publikácie), ktoré boli v diplomovej práci použité.

Forma citácií sa zabezpečuje niektorou z metód, opísaných v norme STN ISO 690, 1998, s. 21. Podrobnejšie informácie nájdete na stránke `http://www.tuke.sk/anta/` v záložke Výsledky práce/Prehľad normy pre publikovanie STN ISO 690 a STN ISO 690-2.

Existujú dva hlavné spôsoby citovania v texte.

- Citovanie podľa mena a dátumu.
- Citovanie podľa odkazového čísla.

Preferovanou metódou citovania v texte vysokoškolskej a kvalifikačnej práce je podľa normy ISO 7144 citovanie podľa mena a dátumu (Katuščák, 1998; Gonda, 2001). V tomto prípade sa zoznam použitej literatúry upraví tak, že za meno sa pridá rok vydania. Na uľahčenie vyhľadávania citácií sa zoznam vytvára v abecednom poradí autorov.

Príklad: . . . podľa (Steinerová, 2000) je táto metóda dostatočne rozpracovaná na to, aby mohla byť všeobecne používaná v . . .

Druhý spôsob uvedenia odkazu na použitú literatúru je uvedenie len čísla tohto zdroja v hranatých zátvorkách bez mena autora (autorov) najčastejšie na konci príslušnej vety alebo odstavca.

Príklad: . . . podľa [13] je táto metóda dostatočne rozpracovaná na to, aby mohla byť všeobecne používaná v . . . ako je uvedené v [14].

Citácie sú spojené s bibliografickým odkazom poradovým číslom v tvare indexu alebo čísla v hranatých zátvorkách. Odkazy v zozname na konci práce budú usporiadané podľa týchto poradových čísel. Viacero citácií toho istého diela bude mať rovnaké číslo. Odporúča sa usporiadať jednotlivé položky v poradí citovania alebo podľa abecedy.

Rôzne spôsoby odkazov je možné dosiahnuť zmenou voľby v balíku `natbib`:

```
% Citovanie podla mena autora a roku
\usepackage[]{natbib}\citestyle{chicago}
% Možnosť rôznych štýlov citácií. Príklady sú uvedené
% v preambule súboru natbib.sty.
% Napr. štýly chicago, egs, pass, anngeo, nlinproc produkujú
% odkaz v tvare (Jones, 1961; Baker, 1952). V prípade, keď
% neuvedieme štýl citácie (vynecháme \citestyle{}) v "options"
% balíka natbib zapíšeme voľbu "colon".
```

Ak zapneme voľbu `numbers`, prepneme sa do režimu citovania podľa odkazového čísla.

```
% Metoda ciselnych citacii
\usepackage[numbers]{natbib}
```

Pri zápise odkazov sa používajú nasledujúce pravidlá:

V odkaze na knižnú publikáciu (pozri príklad zoznamov na konci tejto časti):

- Uvádzame jedno, dve alebo tri prvé mená oddelené pomlčkou, ostatné vynecháme a namiesto nich napíšeme skratku et al. alebo a i.
- Podnázov sa môže zapísať vtedy, ak to uľahčí identifikáciu dokumentu. Od názvu sa oddeľuje dvojbodkou a medzerou.
- Dlhý názov sa môže skrátiť v prípade, ak sa tým nestratí podstatná informácia. Nikdy sa neskracuje začiatok názvu. Všetky vynechávky treba označiť znamienkami vypustenia „. . . “

Pri využívaní informácií z elektronických dokumentov treba dodržiavať tieto zásady:

- uprednostňujeme autorizované súbory solídnych služieb a systémov,
- zaznamenáme dostatok informácií o súbore tak, aby ho bolo opäť možné vyhľadať,
- urobíme si kópiu použitého prameňa v elektronickej alebo papierovej forme,
- za verifikovateľnosť informácií zodpovedá autor, ktorý sa na ne odvoláva.

Pre zápis elektronických dokumentov platia tie isté pravidlá, ako pre zápis „klasických“. Navyše treba uviesť tieto údaje:

- druh nosiča [online], [CD-ROM], [disketa], [magnetická páska]
- dátum citovania (len pre online dokumenty)
- dostupnosť (len pre online dokumenty)

Poradie prvkov odkazu je nasledovné: Autor. Názov. In Názov primárneho zdroja: Podnázov. [Druh nosiča]. Editor. Vydanie alebo verzia. Miesto vydania : Vydavateľ, dátum vydania. [Dátum citovania]. Poznámky. Dostupnosť. ISBN alebo ISSN.

2.4 Vytvorenie zoznamu skratiek a symbolov

Ak sú v práci skratky a symboly, vytvára sa *Zoznam skratiek a symbolov* (a ich dešifrovanie).

Zoznam symbolov a skratiek

μ	mikro, 10^{-6}
SI	Système International
V	volt, základná jednotka napätia v sústave SI

V prostredí LaTeXu sa takýto zoznam ľahko vytvorí pomocou balíka `nomencl`. Postup je nasledovný:

1. Do preambuly zdrojového súboru, napríklad `tukedip.tex`, zapíšeme nasledujúce príkazy:
   `\usepackage[slovak,noprefix]{nomencl}`
   `\makeglossary`
2. V mieste, kde má byť vložený zoznam zapíšeme príkaz
   `\printglossary`
3. V miestach, kde sa vyskytujú skratky a symboly ich definíciu zavedieme, napr. príkazmi
   `\nomenclature{$\upmu$}{mikro, $10^{-6}$}`
   `\nomenclature{V}{volt, základná jednotka napätia v sústave SI}`
   a dokument „preLaTeXujeme“, čím sa vytvorí alebo aktualizuje súbor `tukedip.glo`.
4. Z príkazového riadka spustíme program `makeindex` s prepínačmi podľa použitého operačného systému:
   pre OS Linux
   `makeindex␣tukedip.glo␣-s␣nomencl.ist␣-o␣tukedip.gls`
   pre OS Windows
   `makeindex␣-o␣tukedip.gls␣-s␣nomencl.ist␣tukedip.glo`
5. Po opätovnom „preLaTeXovaní“ dokumentu sa na požadované miesto vloží *Zoznam skratiek a symbolov*.

V pôvodnom súbore `nomencl.sty`, ktorý je súčasťou TeX distribúcií však deklarácia slovenčiny nie je podporená. Máme dve možnosti, buď si upravený balík stiahneme z URL adresy `http://www.etd.sk/tuke/dokumenty.html` (je uložený v zipovaných súboroch GNU/Linux a MS Windows) alebo vyhľadáme súbor `nomencl.sty` v štruktúre naištalovaného TeXu a deklaráciu voľby `slovak` vytvoríme pridaním týchto riadkov:

```
\DeclareOption{slovak}{%
  \def\eqdeclaration#1{, pozri rovnicu\nobreakspace(#1)}%
  \def\pagedeclaration#1{, strana\nobreakspace#1}%
  \def\nomname{Zoznam symbolov a skratiek}}
```

2.5 Spolupráca BibTeXu a LaTeXu

Na bibliografické odkazy sa v štandardnej distribúcii LaTeXu používa príkaz `\cite{}` a prostredie `thebibliography`. Existuje niekoľko balíčkov, ktoré umožňujú pridať odkazom niektoré doplnkové črty. Medzi takéto balíky patria, napríklad, cite, citesort, overcite, chicago a vyššie spomenutý balík natbib (Goossens, Mittelbach a Samarin, 1998). Namiesto príkazu `bibitem` môžu byť použité iné, napríklad v tejto práci sme použili príkazy `harvarditem`.

Súčasťou inštalácie LaTeXu je aj pomocný program BibTeX, ktorý prehľadáva jednu alebo viac databáz, obsahujúcich informácie o publikáciách a automaticky vytvára zoznam použitej literatúry, ktorý následne využije LaTeX(Kopka a Daly, 2004). Podrobný popis BibTeXu a jeho štýlov nájdete v knihe (Goossens, Mittelbach a Samarin, 1998).

Na stránke

`http://www.bibtex.org/`

nájdete množstvo odkazov a informácií súvisiacich s BibTeXom.

BibTeX je nenahraditeľná pomôcka pri práci s rozsiahlymi zoznamami publikácií, resp. pri častom citovaní prác z určitej oblasti. Medzi jeho výhody patria najmä:

- zoznam publikácií sa pripravuje len raz;
- publikácie rovnakého typu sú formátované jednotne v súlade so zvoleným štýlom;
- zmena formátu sa uskutočnuje zmenou *štýlu* – k dispozícii je množstvo štýlov;
- BibTeXovské štýly je možné modifikovať – používajú vlastný postfixový jazyk;
- mnohé časopisy majú vlastné štýly, príprava zoznamu použitej literatúry podľa požiadaviek časopisu pozostáva len v zmene štýlu v príkaze `\bibliographystyle{štýl}`.

2.5.1 Použitie BibTeXovskej databázy

Najprv na jednoduchom príklade stručne popíšeme spoluprácu LaTeXu a BibTeXu. Predpokladajme, že už máme pripravený súbor `knizky.bib`, obsahujúci informácie o publikáciách, napríklad:

```
@INCOLLECTION{kratka:metapost,
  author={Kr{\'a}tk{\'a}, Miroslava},
  year={1--3, 2001},
  title={{METAPOST} a Mfpic},
  booktitle={Zpravodaj {\v{C}eskoslovensk\'eho sdru\v{z}en\'{\i}
   u\v{z}ivatel\accent23u \TeX u}},
  publisher={CSTUG},
  pages={40--135}
}

@BOOK{knuth:meta,
  author={Knuth, Donald Erwin},
  year={1984},
  title={The {METAFONTbook}},
  publisher={American Mathematical Society}
}

@MANUAL{hobby:metapost,
```

```
  author={Hobby, John D.},
  year={1992},
  title={A User's Manual for Metapost}
}
```

Po spracovaní zdrojového súboru `pokus.tex`

```
\documentclass[12pt]{article}
\font\logo=logo10 scaled 1200
\usepackage{slovak}
\usepackage[]{natbib}
\pagestyle{empty}
\addtolength\textwidth{30mm}

\begin{document}
D.~E.~Knuth vytvoril jazyk {\logo METAFONT}\nocite{knuth:meta} súčasne
s~\TeX{}om a zároveň napísal podrobnú učebnicu tohoto jazyka. Autor
{\logo METAPOST}u, ktorý je rozšírením  {\logo METAFONT}u, zároveň
napísal príručku \emph{A User's Manual for Metapost}
\citep{hobby:metapost}. V~rodnom jazyku si môžeme prečítať seriál
\emph{{\logo METAPOST} a mfpic} \citep{kratka:metapost}.

\bibliography{knizky}
\bibliographystyle{natbib} % unsrt alpha abbrv
\end{document}
```

programom [pdf]csLaTeX sa vytvorí súbor `pokus.aux`, v ktorom LaTeX odovzdá potrebnú informáciu programu BibTeX. Letmý pohľad na LaTeXovský zdroják stačí na to, aby ste pochopili, odkiaľ sa BibTeX dozvie, že okrem súboru `pokus.aux` má použiť informácie zo súborov `knizky.bib` a `natbib.bst`[22].

Po spustení príkazu

```
bibtex pokus.aux
```

z príkazového riadku sa vytvorí súbor `pokus.bbl`, ktorý sa použije pri ďalších dvoch spusteniach LaTeXu. V tomto prípade tento súbor obsahuje nasledujúce riadky:

```
\begin{thebibliography}{}

\bibitem[Hobby(1992)Hobby]{hobby:metapost}
Hobby, J.~D. (1992).
\newblock {\em A User's Manual for Metapost\/}.

\bibitem[Knuth(1984)Knuth]{knuth:meta}
Knuth, D.~E. (1984).
\newblock {\em The {METAFONTbook}\/}.
\newblock American Mathematical Society.

\bibitem[Kr{\'a}tk{\'a}(2001)Kr{\'a}tk{\'a}]{kratka:metapost}
Kr{\'a}tk{\'a}, M. (1--3, 2001).
```

[22] Tento BibTeXovský štýl sme stiahli z internetu.

```
\newblock {METAPOST} a mfpic.
\newblock In {\em Zpravodaj {\v{C}eskoslovensk\'eho sdru\v{z}en\'{\i}
  u\v{z}ivatel\accent23u \TeX u}\/}, pages 40--135. CSTUG.

\end{thebibliography}
```

Po preTEXovaní bude výstup vyzerať približne nasledovne:[23]

D. E. Knuth vytvoril jazyk METAFONT súčasne s TEXom a zároveň napísal podrobnú učebnicu tohoto jazyka. Autor METAPOSTu, ktorý je rozšírením METAFONTu, zároveň napísal príručku *A User's Manual for Metapost* (Hobby, 1992). V rodnom jazyku si môžeme prečítať seriál METAPOST *a mfpic* (Krátká, 2001).

Literatúra

Hobby, J. D. (1992). *A User's Manual for Metapost*.

Knuth, D. E. (1984). *The METAFONTbook*. American Mathematical Society.

Krátká, M. (1–3, 2001). METAPOST a mfpic. In *Zpravodaj Československého sdružení uživatelů TEXu*, pages 40–135. CSTUG.

Ak zmeníme štýl príkazom `\bibliographystyle{alpha}`, po preLATEXovaní, BIBTEXovaní a opakovanom dvojnásobnom LATEXovaní bude súbor `pokus.bbl` vyzerať nasledovne:

```
\begin{thebibliography}{Knu84}

\bibitem[Hob92]{hobby:metapost}
John~D. Hobby.
\newblock {\em A User's Manual for Metapost}, 1992.

\bibitem[Knu84]{knuth:meta}
Donald~Erwin Knuth.
\newblock {\em The {METAFONTbook}}.
\newblock American Mathematical Society, 1984.

\bibitem[Kr{\'a}01]{kratka:metapost}
Miroslava Kr{\'a}tk{\'a}.
\newblock {METAPOST} a mfpic.
\newblock In {\em Zpravodaj {\v{C}eskoslovensk\'eho sdru\v{z}en\'{\i}
  u\v{z}ivatel\accent23u \TeX u}}, pages 40--135. CSTUG, 1--3, 2001.

\end{thebibliography}
```

a výstup bude vyzerať približne nasledovne:

[23] V súbore `pokus.bbl` sa objavilo anglické „pages“. V prípade slovenských publikácií je toto možné zmeniť buď priamou editáciou súboru `pokus.bbl` alebo vytvorením vlastného štýlu – napríklad skopírovaním použitého štýlu do súboru s iným názvom a nahradením anglických slov slovenskými.

D. E. Knuth vytvoril jazyk METAFONT súčasne s TEXom a zároveň napísal podrobnú učebnicu tohoto jazyka. Autor METAPOSTu, ktorý je rozšírením METAFONTu, zároveň napísal príručku *A User's Manual for Metapost* [Hob92]. V rodnom jazyku si môžeme prečítať seriál METAPOST *a mfpic* [Krá01].

Literatúra

[Hob92] John D. Hobby. *A User's Manual for Metapost*, 1992.

[Knu84] Donald Erwin Knuth. *The METAFONTbook*. American Mathematical Society, 1984.

[Krá01] Miroslava Krátká. METAPOST a mfpic. In *Zpravodaj Československého sdružení uživatelů TEXu*, pages 40–135. CSTUG, 1–3, 2001.

A ešte vyskúšajme štýl `\bibliographystyle{abbrv}`. Súbor `pokus.bbl` vyzerať nasledovne:

```
\begin{thebibliography}{1}

\bibitem{hobby:metapost}
J.~D. Hobby.
\newblock {\em A User's Manual for Metapost}, 1992.

\bibitem{knuth:meta}
D.~E. Knuth.
\newblock {\em The {METAFONTbook}}.
\newblock American Mathematical Society, 1984.

\bibitem{kratka:metapost}
M.~Kr{\'a}tk{\'a}.
\newblock {METAPOST} a mfpic.
\newblock In {\em Zpravodaj {\v{C}eskoslovensk\'eho sdru\v{z}en\'{\i}
  u\v{z}ivatel\accent23u \TeX u}}, pages 40--135. CSTUG, 1--3, 2001.

\end{thebibliography}
```

a výstup bude vyzerať približne nasledovne:

D. E. Knuth vytvoril jazyk METAFONT súčasne s TEXom a zároveň napísal podrobnú učebnicu tohoto jazyka. Autor METAPOSTu, ktorý je rozšírením METAFONTu, zároveň napísal príručku *A User's Manual for Metapost* [1]. V rodnom jazyku si môžeme prečítať seriál METAPOST *a mfpic* [3].

Literatúra

[1] J. D. Hobby. *A User's Manual for Metapost*, 1992.

[2] D. E. Knuth. *The METAFONTbook*. American Mathematical Society, 1984.

[3] M. Krátká. METAPOST a mfpic. In *Zpravodaj Československého sdružení uživatelů TEXu*, pages 40–135. CSTUG, 1–3, 2001.

Ako je vidieť, zmenou jediného slova – názvu štýlu – dosiahneme rôzne formátovanie použitej literatúry. Vyskúšajte si ešte štýly `plain` a `unsrt`. Zoznam ďalších štýlov nájdete, napríklad, v knihe (Goossens, Mittelbach a Samarin, 1998).

2.5.2 Príprava BIBTEXovskej databázy

Súbor `knizky.bib`, ktorého obsah sme uviedli vyššie, obsahuje 3 záznamy rôznych typov – `manual`, `book` a `incollection`. Každý typ má *povinné* (required), *nepovinné* (optional) a *ignorovateľné polia* – napríklad povinné polia typu `incollection` sú: `author`, `year`,`title`, `booktitle` a `publisher`. Nepovinné polia tohoto typu sú: `editor`, `volume` alebo `number`, `series`, `type`, `chapter`, `pages`, `address`, `edition`, `month` a `note`. Nepovinné polia nie je nutné vypĺňať. Ignorovateľné polia je vhodné používať napríklad v prípade, ak chceme v `bib`-súbore uložiť nejakú dodatočnú infromáciu o niektorej publikácii.

Najčastejšie používané typy dokumentov sú: `article`, `book`, `booklet`, `inbook`, `incollection`, `inproceedings`, `manual`, `masterthesis`, `misc`, `phdthesis`, `proceedings`, `techreport` a `unpublished`. Zoznam povinných a nepovinných polí pre jednotlivé typy dokumentov nájdete v knihe (Goossens, Mittelbach a Samarin, 1998).

Ďalej uveďme abecedný zoznam štandardných polí BIBTEXovských štýlov: `address`, `annote`, `author`, `booktitle`, `chapter`, `crossref`, `edition`, `editor`, `howpublished`, `institution`, `journal`, `key`, `month`, `note`, `number`, `organization`, `pages`, `publisher`, `school`, `series`, `title`, `type`, `volume` a `year`.

Syntax vyplnenia jednotlivých polí tu neuvádzame pre nedostatok miesta. Napríklad ako BIBTEX rozozná, keď v poli `author` nájde záznam `Ján Pavol Peter`, ktoré z mien je priezvisko a ktoré krstné meno? Jednou z možností je napísať najprv priezvisko a oddeliť ho od krstného mena čiarkou. Teda záznam `author={Peter, Ján Pavol}` v prípade použitia skratiek rodných mien bude zobrazený ako J. P. Peter alebo Peter, J. P. v závislosti od zvoleného štýlu. Príkaz `@string` sa používa na vytváranie skratiek, napríklad:

@StringTUKE =”Technická univerzita v Košiciach”

V knihe (Goossens, Mittelbach a Samarin, 1998) nájdete podrobnejší popis syntaxe. Dozviete sa tam tiež, kedy BIBTEX odlišuje veľké písmená od malých a mnoho ďalších informácií.

2.5.3 Programy uľahčujúce prácu s BIBTEXovskou databázou

Pri pohľade na množstvo typov a ešte väčšie množstvo rôznych druhov polí človek začne pochybovať o tom, či sa pustiť do práce s BIBTEXom. Našťastie existujú programy, ktoré umožňujú vytvárať a modifikovať `bib`-súbory. Spomeňme tu len dva programy: BibDB a BibEdt, ktorých binárne súbory pre OS Windows sa dajú stiahnuť zo stránky `http://dmoz.org/Computers/Software/Typesetting/TeX/Platform_Specific/DOS_and_Windows/BibTeX/`.

Obidva tieto programy umožňujú voľbu typu publikácie pri pridávaní nových členov do databázy. Podľa zvoleného typu zobrazia zoznamy povinných a nepovinných polí. Na začiatok je teda dobré vytvoriť databázu s vyplnenými povinnými položkami a neskôr prípadne doplniť informácie do nepovinných polí.

2.5.4 Balík biblatex

Na „Troch kráľov“ roku 2007 bola „uvoľnená“ verzia 0.6 LATEXovského balíka biblatex, ktorého autorom je Philipp Lehman. Jeho prvé verzie sú datované septembrom 2006, jedná sa teda o novinku. Tento balíček sa dá stiahnuť z `http://www.tex.ac.uk/tex-archive/help/Catalogue/entries/biblatex.html`, kde sa uvádza:

Balík BIBLATEX je úplná reimplementácia bibliografického príslušenstva poskytovaného LaTeXom v spojení s BibTeXom. Spôsob spolupráce LaTeXu a BibTeXu prepracovaný úplne od základu. Namiesto implementácie v BibTeXovských štýlových súboroch, je formátovanie použitej literatúry riadené výhradne TeXovskými makrami. Na vytvorenie nových bibliografických štýlov by mala byť postačujúca dobrá aktívna znalosť LaTeXu. Na to nie je nutné učiť sa BibTeXovský postfixový zásobníkový jazyk. Podobne môžu byť ľahko predefinované všetky citačné príkazy.

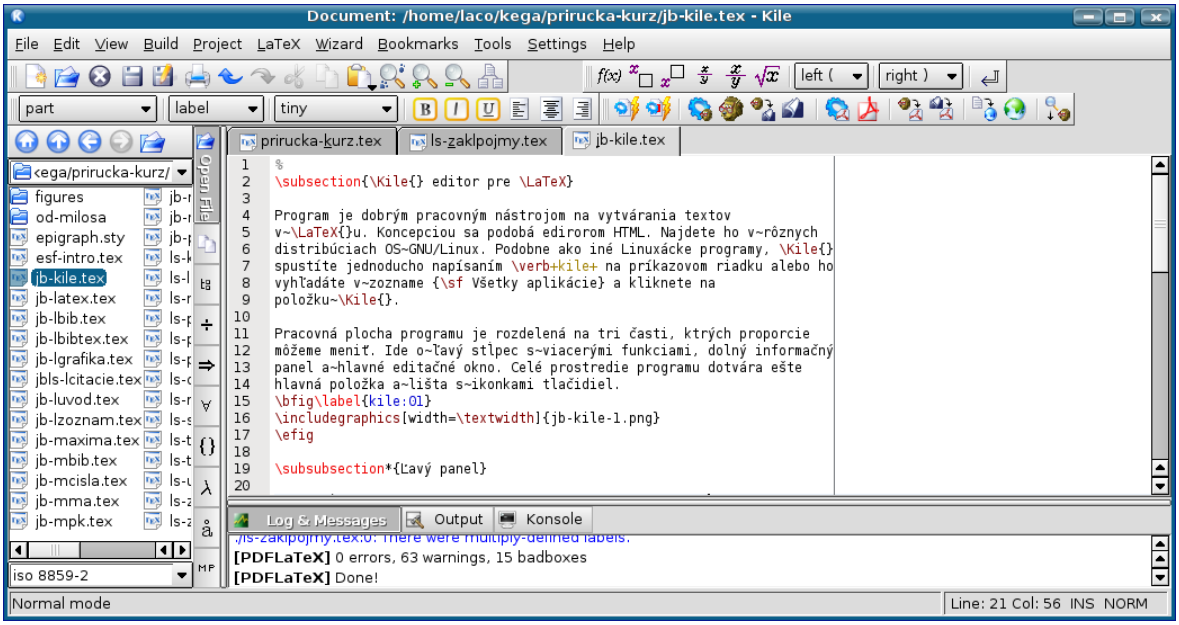
Okrem jedinečných čŕt balíka biblatex, zahŕňa tento hlavné črty nasledujúcich balíkov: babelbib, bibtopic, bibunits, chapterbib, cite, inlinebib, mlbib, multibib, splitbib. Nájdu sa aj niektoré koncepčné paralely s balikmi natbib a amsrefs. Balíček biblatex podporuje delenie citačných zoznamov, viacnásobné bibliografie v rámci jedného dokumentu, oddelené zoznamy bibliografických záznamov. Bibliografie môžu byť rozdelené na časti (podľa kapitol, oddielov a pod.) a/alebo členené podľa tém (podľa typu, kľúčových slov, atď.). Balík je úplne lokalizovaný a môže byť použitý spoločne s balíkom babel.

Príručka balíka biblatex má 117 strán a je k dispozícii na URL adrese `http://www.tex.ac.uk/tex-archive/macros/latex/exptl/biblatex/doc/biblatex.pdf`. Balík sa nedá automaticky použiť so všetkými štýlovými `bst`-súbormi, `bib`-súbory bude potrebné modifikovať len veľmi málo. Zdá sa teda, že ak ste doteraz nezačali s BibTeXom, je dobré začať prečítaním uvedeného manuálu.

2.6 KILE editor pre LaTeX

Program je dobrým pracovným nástrojom na vytváranie textov v LaTeXu. Koncepciou sa podobá edirorom HTML. Nájdete ho v rôznych distribúciach OS GNU/Linux. Podobne ako iné Linuxácke programy, KILE spustíte jednoducho napísaním `kile` na príkazovom riadku alebo ho vyhľadáte v zozname Všetky aplikácie a kliknete na položku KILE.

Pracovná plocha programu je rozdelená na tri časti, ktrých proporcie môžeme meniť. Ide o ľavý stĺpec s viacerými funkciami, dolný informačný panel a hlavné editačné okno. Celé prostredie programu dotvára ešte hlavná položka a lišta s ikonkami tlačidiel.



Obrázok 2: Pracovné okno programu KILE

Ľavý panel

Ľavá časť okna programu obsahuje stĺpec, ktorý vo svojej pravej časti má vertikálny zoznam ikon, ktoré predstavujú jeho desať častí. V dolnej časti sa nachádza ovtáracie okienko na nastavenie kódovej stránky, v ktorej sa nám otvorený súboru bude zobrazovať a v ktorej ho môžeme editovať. Kliknutím na ikony zmeníme funkciu ľavého stĺpca:

- Jednoduchý správca súborov, kliknutím na súbor typu `.tex` ho otvorí v editovacom okne.
- Manažér projektov. Do projektu môžeme pridávať súbory, odstraňovať ich, vytvoriť archív projektu a pod.
- Stromový náhľad na logickú štruktúru dokumentu. Hierarchicky sa zobrazujú nadpisy, vložené súbory a niektoré ďalšie príkazy spolu s riadkom, kde sa v zdrojovom kóde vyskytujú. Kliknutím na položku v tomto strome sa premiestnime do zdrojového textu.
- Ďalšie ikony reprezentujúce funkcie, ktorými môžeme priamo pristupovať k špeciálnym znakom a symbolom. Namiesto písania zložitých príkazov užívateľ len klikne na symbol a príkaz sa doplní automaticky do textu.

Dolný panel

Dolný panel má tri záložky: Log & Messages, Output a Konsole. Prepínaním záložiek získame podrobný prehľad o priebehu spracovania dokumentu. V prvej záložke sa zobrazujú správy, ktoré krátko informujú o (ne)úspechu pri behu programu TEX. V druhej záložke sú vypísané všetky informácie, ktoré program pri svojom behu vypisuje. Je to praktické pri hľadaní chýb, každý riadok s chybou je zvýraznený a kliknutím naň sa presunieme na miesto do zdrojového textu, kde je očakávaná príčina chyby. Tretia záložka je integrovaná konzola príkazového interpretera. Keď potrebujeme vykonať nejaký príkaz, ktorý nie je v ponuke, nie je potrebné spúšťať napr. `xterm` alebo `konsole`, ale môžeme ho previesť priamo tu.

Editovacie okno

Táto časť programu predstavuje najdôležitejší prvok aplikácie, je tu priestor, v ktorom užívateľ upravuje zdrojový text dokumentu. Práca je sprehľadnená zvýraznením syntaxe, tzn. farebným rozlíšením rôznych častí textu. Farby sú preddefinované, ale môžeme ich pochopiteľne meniť vrátane typu a rezu písma. Celé prostredie je primárne určené pre LATEX, ale môžeme v ňom upravovať aj dokumenty v iných (skriptovacích, značkovacích) jazykoch, napr. v HTML, CSS, C, Octave a pod.

Hlavný panel, panely ikon

Tieto položky sa najčastejšie nachádzajú v hornej časti okna aplikácie, ale môžeme ich presúvať. Hlavná ponuka obsahuje niekoľko položiek, ktoré nepotrebujú komentár: File, Edit, View, Bookmarks, Tools, Help. Sú to obvyklé voľby, ktoré užívateľ rýchle pochopí, niektoré ostatné stručne opíšeme:

- Položka Build obsahuje príkazy, potrebné na spracovanie zdrojového textu a vytvorenie rôznych typov koncových dokumentov (`.dvi`, `.ps`, `.pdf`, `.html`). Nájdeme tu aj spomenuté príkazy na hľadanie chýb a varovaní.
- Položka Project umožňuje vytvoriť projekt (na prácu s jedným súborom ho nemusíme používať) a pracovať so súbormi, ktoré sme do projektu vložili. Môžeme taktiež otvoriť iný projekt, súčasne však môžeme pracovať len s jedným.
- LATEX, táto položka je veľmi bohatá a rozvrstvená. Obsahuje triedené príkazy LATEXu, ktoré môžeme vybrať myšou, pričom sa samotný príkaz vloží do zdrojového textu na miesto, kde je práve textový kurzor. Táto funkcia veľmi urýchluje a uľahčuje prácu s dokumentom. Všetky ponuky sú prehľadne rozčlenené podľa zameraní vrátane podpory sadzby matematiky.
- Položka Wizard pomáha užívateľovi vytvoriť dokument podľa vlastnej predstavy (Quick Start), napr. list, tabuľku, polia a tabuľky. Vloženú tabuľku už nemôžeme rovnakým spôsobom upravovať (musíme poznať potrebné príkazy), napriek tomu nám táto ponuka veľmi uľahčuje a urýchľuje prácu s dokumentom.
- Settings je ponuka, ktorá umožňuje nastavenenie samotného programu ako aj jeho editora textovej časti. Na používanie slovenského variantu LATEXu, ktorý sa volá csLATEX je potrebné zmeniť niekoľko nastavení. „Slovenská“ verzia LATEXu sa spúšťa príkazom `cslatex`. Preto je potrebné zmeniť položku `LaTeX` v ponuke Configure Kile v sekcii Build tak, aby sa spúšťal program (formát) `cslatex`. Podobnú zmenu musíme spraviť pre položku `PDFLaTeX`, tam nastavíme spúšťanie formátu `pdfcslatex`. Ďalšie programy by už mali byť súčasťou vašej inštalácie a pravdepodobne ich nebude potrebné meniť. V ponuke `Configure Shortcuts` je

možné zmenit' implicitné nastavenia klávesových skratiek podľa vlastných želaní a zvyklostí. V editovacom okne máme pre každý otvorený dokument záložku, pomocou ktorej sa môžeme medzi dokumentmi prepínat'. Obyčajne je jeden z nich hlavný. Možnost', aby sa odštartovaním kompilácie spúšťal hlavný dokument, nastavíme jeho otvorením a aktivovaním ponuky Define Current Document as 'Master Document'. KILE umožňuje prácu s dokumentmi v rôznych kódovaniach, z veľkého množstva napr. `cp1250`, `iso 8859-2`, `utf-8`. Na používanie zvoleného kódovania zmeníme niekoľko vecí. V ponuke Configure Kile v sekcii Encoding zvolíme implicitné kódovanie, podobnú úpravu vykonáme v ponuke Configure Editor v sekcii Open/Save. Po týchto nastaveniach je KILE pripravený pracovat' s nastaveným kódovaním. Musíme to však oznámit' LaTeXu príkazmi v preambule hlavného dokumentu, napr. na prácu s kódovaním `utf-8` vložíme riadok `\usepackage[utf8]{inputenc}`.

Týmto sme samozrejme nevyčerpali všetky možnosti, ktoré aplikácia poskytuje. V každom programe je čosi krásne pri objavovaní jeho nastavení – objavujte! Žiadny program na prácu s LaTeXom a TeXom v distribúciach GNU/Linux neposkytuje také komplexné prostredie ako KILE. Jestvujú programátorské editory, ktoré môžeme prispôsobit' na prácu s LaTeXom, ale KILE je pripravené na komfortnú prácu už pri prvom štarte.

Použitá literatúra

GONDA, V. 2001. *Ako napísať a úspešne obhájiť diplomovú prácu.* Bratislava : Elita, 2001, 3. doplnené a prepracované vydanie, ISBN 80-8044-075-1, 120 s.

GOOSSENS, M. – MITTELBACH, F. – SAMARIN, A. 1998. *The LaTeX Companion* ADDISON-WESLEY, 9. vydanie, ruský preklad Moskva, Mir, 1999, ISBN 5-03-003325-4, 607 s.

GOOSSENS, M. – RAHTZ, S. 1999. *The LaTeX Web Companion* ADDISON-WESLEY, 2. vydanie, ruský preklad Moskva, Mir, 2001, ISBN 5-03-003387-4, 607 s.

GOOSSENS, M. – RAHTZ, S. – MITTELBACH, F. 1999. *The LaTeX Graphics Companion* ADDISON-WESLEY, 3. vydanie, ruský preklad Moskva, Mir, 2002, ISBN 5-03-003388-2, 622 s.

HOBBY, J. D. 1992. *A User's Manual for Metapost*, 87 s.

KATUŠČÁK, D. 1998. *Ako písať vysokoškolské a kvalifikačné práce.* Bratislava : Stimul, 1998, 2. doplnené vydanie, ISBN 80-85697-82-3, 121 s.

KOPKA, H. a DALY, P. W. 2004. *LaTeX. Podrobný průvodce.* Brno: Computer Press, 2004. Český preklad J. Gregor, ISBN 80-7226-973-9, 576 s.

KNUTH, D. E. 1984. *The METAFONTbook.* American Mathematical Society.

KRÁTKÁ, M. 2001. METAPOST a mfpic. V *Zpravodaj Československého sdružení uživatelů TeXu*, 1–3, 2001, s. 40–135.

OETIKER, T. – PARTL, H. – HYNA, I. – SCHLEGL, E. 2000. *Nie príliš stručný úvod do systému LaTeX 2ε.* Version 3.13, 2000. Slovenský preklad J. Buša ml. a st., 2002, 109 s.

RECKDAHL, K. 2006. *Using Imported Graphics in LaTeX and pdfLaTeX.* Version 3.0.1, 2006, 124 s.

RYBIČKA, J. 2003. *LaTeX pro začátečníky.* Brno: Konvoj, 3. vydání, 2003, ISBN 80-7302-049-1, 238 s.

STEINEROVÁ, J. 2000. *Základy filozofie človeka v knižničnej a informačnej vede.* In: Kimlička, č., Knižničná a informačná veda na prahu informačnej spoločnosti. Bratislava : Stimul, 2000. ISBN 80-2274-035-2, s. 327–334.

ŠEDIVÝ, P. – BROŽ, M. – GŘONDILOVÁ, J. – PÍŠE, M. – HOUFEK, K. 1978. Kreslíme METAFONTem. V *Zpravodaj Československého sdružení uživatelů TeXu*, 1, 1998, s. 1–65.

3 PROGRAMOVANIE V PYLABE A PYTHONE

Michal KAUKIČ
Katedra matematických metód, FRI
Žilinská univerzita v Žiline

3.1 Úvod

Prvoradým cieľom tejto kapitoly je poskytnúť základné informácie, potrebné k úspešnej práci s programovým systémom Pylab a s jazykom Python, na ktorom je tento systém založený. Autor tohoto textu využíva Pylab už niekoľko rokov na Fakulte riadenia a informatiky Žilinskej univerzity pri výučbe predmetov *Numerické metódy, UNIX – vývojové prostredie, Moderné približné metódy, Softvérové nástroje pre inžinierov (Open Source)*. Postupne by sme chceli tento systém zaviesť aj do výučby ďalších matematických a príbuzných predmetov a boli by sme radi, keby sa začal používať aj na iných slovenských vysokých a stredných školách. Príklady, uvedené v tomto texte, sú hlavne z oblasti numerickej analýzy, vrátane rozsiahlejších ukážok použitia Pylabu na problémy interpolácie a aproximácie dát, numerickej integrácie a riešenia nelineárnych rovníc a ich sústav.

Sme presvedčení, že zvládnutie tohto elegantného, jasného a ľahko naučiteľného programovacieho prostriedku sa čitateľovi vyplatí aj do budúcnosti. Je to mohutný systém s veľkými výpočtovými i grafickými možnosťami, určený na prácu aj s veľkými objemami dát a s ich vizualizáciou. V pozadí je kvalitný univerzálny programovací jazyk Python (VAN ROSSUM, 2006), ktorý máte v Pylabe plne k dispozícii. V prípade potreby si môžete do systému pridať ďalšie moduly (napr. na prácu s databázami SQL, lineárne programovanie, prezentáciu výpočtov a dát na webe a pod.).

Nemusíte však veľa vedieť o programovaní v Pythone, pretože Pylab je jazyk veľmi vysokej úrovne, inšpirovaný známym komerčným systémom MATLAB[24]. Pri práci v ňom dosiahnete s minimálnou „programátorskou“ námahou veľké výsledky a môžete sa viac venovať skutočne zaujímavým a podstatným veciam nielen v matematicky zameraných predmetoch, ale aj vo svojej ďalšej odbornej práci. Zo skúseností vieme a veríme, že mnohým z vás Pylab a Python pomôže napr. pri spracovaní semestrálok, bakalárskych či diplomových prác, pri práci na doktorantúre a aj neskôr, keď budete v zamestnaní.

Výhodou Pylabu je vysoký stupeň kompatibility s MATLAB-om, ktorý je veľmi rozšírený hlavne vo vzdelávacích inštitúciach, ale aj v priemysle, vďaka existencii rôznych rozšírení (toolboxov) hlavne pre oblasť inžinierskych výpočtov. To, čo MATLAB nemá, a čo považujeme za prednosť Pylabu, je univerzálny programovací jazyk (u nás Python), široko využiteľný aj samostatne. Ak však potrebujete používať už hotové programy, napísané pre MATLAB, odporúčame vám Open Source systém Octave (BUŠA, 2006).

Predpokladáme, že tento text budete čítať pozorne a nad prečítaným sa budete aj zamýšľať, najlepšie s asistenciou počítača. Bez neho je to ťažké, niečo ako lízanie medu cez sklo. Nemusíte si čítať úplne všetko a „učiť sa to“. Najlepším učiteľom ja samotný Pylab, keď ho budete používať na riešenie problémov, ktoré vás bavia. Potom pre vás bude komunikácia s týmto systémom niečo ako rozhovor s inteligentným priateľom a pomocníkom. Na takýto rozhovor si zvyknete rýchlo a čo je dôležitejšie, rýchlo sa naučíte byť pri svojej práci samostatní.

Náš názor je, že **na matematických predmetoch sa neučíme programovať**. Predpokladáme, že

[24]MATLAB je registrovaná obchodná značka softvérovej firmy The MathWorks,Inc.

máte aspoň základné návyky v algoritmizácii úloh (z oblasti aplikovanej matematiky alebo z inžinierskej praxe). Ak nie, možno vás k ďalšiemu štúdiu programovacích prostriedkov inšpirujú práve tieto materiály. Preto v tejto kapitole nájdete hlavne konkrétne príklady (kde sa dalo, aj ilustrované grafikou) a nie suchopárne programátorske pravidlá. Dúfame, že si každý z vás nájde niečo, čo ho aspoň trochu zaujme a vyprovokuje k samostatnej činnosti a experimentovaniu.

Ak narazíte na problémy, snažte sa ich najskôr vyriešiť sami (veď na čo človek príde sám, to si aj najlepšie zapamätá). Keď to už nijako nejde, hľadajte pomoc vo svojom okolí (vrátane nás, učiteľov). Ak vieme, radi poradíme. Ak nevieme, aspoň máte lepší pocit, že nie ste sami a spolu budeme hľadať niekoho, kto by to vedieť mohol (aj keby bol u protinožcov).

Tento text je mienený ako úvod do práce s Pylabom, má vám teda **pomôcť začať pracovať** v tomto systéme. Na mnohé otázky v ňom nenájdete odpoveď. Ak však budú niektoré vaše otázky (alebo aj typické chyby a problémy) veľmi časté, radi tento text upravíme, či rozšírime o odpovede na ne. Pripomienky môžete posielať na adresu autora mike@frcatel. fri.uniza.sk.

Nemohli ste si nevšimnúť, že je čoraz ťažšie zaobísť sa bez znalosti jazykov, angličtiny zvlášť. Ak máte v tejto oblasti slabiny, unikajú vám mnohé príležitosti. Zastávame názor, že učenie sa jazykov nie je nejaká separovaná činnosť (drvenie slovíčiek, gramatických poučiek a iné umelo vymyslené aktivity), ale treba ju podľa možností čo najskôr „zabudovať" do našej bežnej práce. Samozrejme, ako štartovací bod je nutná istá minimálna slovná zásoba a základná znalosť gramatiky. Hlavne na úrovni doktorandského štúdia je znalosť jazykov nevyhnutnou podmienkou úspešnej práce.

Angličtina je tiež v prípade Pylabu viac ako užitočná. Kto o tom ešte pochybuje, dúfame, že aj pri práci s Pylabom nájde ďalší podnet (a príležitosť) na zdokonalenie sa v oblasti jazykov. Pylab má vynikajúci systém na nápovedu, ale je to všetko v angličtine a v dohľadnom čase to nebude inak. Myslíme si tiež, že niektoré pojmy v oblasti informatiky strácajú prekladom na zrozumiteľnosti, takže aj k nami pridaným častiam systému sme písali nápovedu po anglicky.

Ďalšia prekážka, ktorú musíte prekonať, je zvyknúť si na iný, hoci pohodlnejší štýl práce v Pylabe, objavovať a naučiť sa využívať jeho bohaté možnosti. Nie je to PASCAL, ani C-čko, takže sa ho nesnažte podľa vzoru týchto jazykov komplikovať a zneprehľadňovať. Jasný, prehľadný a logický program je ako umelecké dielo. Nič tam nesmie byť zbytočné a všetky časti musia byť veľmi dobre vybrané a zosúladené. Potom môžete mať radosť z toho, že ste niečo také dokázali vytvoriť. Pylab ako programovací prostriedok vám bude v tomto výdatne pomáhať, ak sa ním necháte viesť a inšpirovať.

Ďakujem kolegom Janovi Bušovi a Ladislavovi Ševčovičovi za možnosť vydania tohto textu aj ako učebnice v rámci série príručiek k open source softvéru, za dôkladne testovanie Pylabu a cenné pripomienky k samotnému systému ako aj k textu tejto kapitoly. Ďakujem tiež kolegyni Alžbete Klaudiny za veľmi dôkladné prečítanie textu a početné navrhnuté opravy, ktoré výrazne pomohli zlepšiť jeho kvalitu.

3.2 Základný popis systému Pylab

3.2.1 Všeobecná charakteristika systému

Pylab, ako asi uhádnete, je skratka z Python Laboratory. Keďže inšpiráciou bol MATLAB, je to predovšetkým maticové laboratórium. Teda, interaktívny, maticovo orientovaný systém na vedecké a inžinierske výpočty a vizualizáciu dát (tzn. výsledky výpočtov môžete graficky prezentovať v rôznych typoch grafov, prípadne formou animácie).

Zdrojové programy, napísané v Pylabe (čo je vlastne jazyk Python s vhodnými knižnicami/modulmi) sa dajú bezo zmeny prenášať medzi jednotlivými implementáciami na rôznych počítačoch s rôznymi operačnými systémami. Špeciálne pre MS Windows sú vymyslené prostriedky (napr. `py2exe`, `http://www.py2exe.org/`) ako vašu aplikáciu (aj so všetkým, čo je pre jej beh potrebné, vrátane Pythonu a príslušných modulov) zbaliť do jedného vykonávateľného súboru. Takže, ak máte rozsiahlejší výpočet, môžete ísť na výkonnejší a spoľahlivejší počítač.

Asi je načase, aby ste sa posadili k počítaču a vyskúšali si zopár jednoduchých vecí. Hlavne, ako Pylab spustiť a ako z neho odísť. Pylab sa spustí príkazom `ipython -pylab`, ale to si pamätať nemusíte. V našich učebniach, kde budeme tento systém využívať, to bude jedna z položiek menu (resp. ikona na ploche) v grafickom prostredí XWindow, v ktorom sa ocitnete po prihlásení. Len tak mimochodom, budeme pracovať v OS GNU/Linux, ale to tiež nie je dôležité. Budeme využívať len Pylab a niekoľko málo iných aplikácii. Tí z vás, ktorí sa rozhodnú inštalovať a používať Pylab v MS Windows to budú mať podstatne zložitejšie.

Po spustení sa vám otvorí príkazové okno, v ktorom už budete môcť zadávať príkazy Pylabu a aj niektoré užitočné príkazy operačného systému (napr. `ls` – výpis adresára, `cd` – zmena adresára). Vaše vstupy sa značia na obrazovke ako `In [1]`, `In [2]`, ... a zodpovedajúce výstupy (odpovede) Pylabu sú `Out [1]`, `Out [2]`, ... Pre začiatok si môžete vyskúšať napr. `pylab?`, čo je prehľadná nápoveda o Pylabe. Keď sa chcete dostať znova na príkazový riadok, stlačte `q` (bude to vysvetlené ďalej).

Práca v Pylabe sa ukončí príkazmi `quit` **alebo** `exit` alebo, čo je najpohodlnejšie, kombináciou klávesov `Ctrl-D` (to je znak konca súboru, v našom prípade signál na ukončenie vstupu zo štandardného vstupného súboru – klávesnice). Keď sa systém pýta, či chceme skutočne odísť, prikývneme (klávesnicou). Všetky príkazy, ktoré ste zadávali, sa zapisujú do súboru (konkrétne, je to súbor `history` v podpriečinku `.ipython` vášho domovského priečinka – to je ten, kde sa ocitnete po prihlásení a z ktorého nie je dôvod vôbec niekam vyskakovať). Ak z Pylabu neodídete vyššie spomínaným spôsobom, ale kliknutím myšou na značku $\times$ na lište príkazového okna, príkazy sa do histórie nezapíšu, tak si na to dajte pozor!

Po príkazoch z histórie sa môžete v Pylabe pohybovať šípkami (hore, dole). Ak viete, že chcete vykonať príkaz, ktorý je v histórii a začínal napr. `y=`, tak to napíšete a stlačíte `Ctrl-P`. Príkaz sa doplní a príp. ak ich bolo viac, môžete sa opätovným `Ctrl-P` po týchto príkazoch pohybovať smerom „dozadu“, alebo stlačením `Ctrl-N` smerom „dopredu“.

Pri práci s Pylabom sa nemôžete „stratiť“, (pravda, ak trocha viete anglicky), pretože príkazom

`objekt?` alebo `?objekt`

dostanete informácie o príslušnom objekte. To je dobre vedieť hneď na začiatku a často je to rýchlejšia alternatíva ako listovanie v dokumentácii. Skúste si napr. `plot?`, `linspace?`, `roots?`. Ak je nápoveda na viac obrazoviek, zobrazí sa pomocou špeciálneho programu – stránkovača (poznáte to podľa dvoch bodiek na samostatnom poslednom riadku v okne), vtedy prezeranie ukončíte stlačením klávesu `q`.

Jediný problém je, že na začiatku asi nebudete vedieť, na čo sa pýtať. Preto na webovej stránke

`http://frcatel.fri.uniza.sk/pylab.html` nájdete názvy niektorých užitočných príkazov, spolu s ich stručným vysvetlením. Prehľad príkazov nájdete aj ako prílohu tejto príručky. Potom si môžete podrobnosti doplniť vyvolaním nápovedy.

Hoci Python, náš pracovný kôň (workhorse :-) v pozadí, pracuje s mnohými typmi objektov (celé, reálne i komplexné čísla, zoznamy, asociatívne polia, usporiadané n-tice, textové reťazce), nás budú zaujímať hlavne číselné obdĺžnikové matice a vektory (ich prvky môžu byť celočíselné, reálne, ale i komplexné). Matice a vektory sú základnym dátovým typom v Pylabe a existuje veľa funkcií, ktoré sa dajú na ne aplikovať „po prvkoch". Je teda treba mať na pamäti, že napr. **príkaz `sin(A)` neurobí len sínus jedného čísla, ale aplikuje funkciu sínus na všetky prvky matice `A`** bez toho, že by sme museli programovať cykly po prvkoch matice.

Preto programy v Pylabe vychádzajú oveľa menšie a zrozumiteľnejšie, než v tradičných programovacích jazykoch (PASCAL, FORTRAN, C/C++). Je to dané aj tým, že netreba deklarovať typy premenných; podľa toho, ako vznikli, je jasné aj akého sú typu (napr. `a=1` je celočíselná premenná, `b="facina"` je textová premenná, `c=[1,2,'tri']` je zoznam, ...). Skrátka, všetko, čo vzniká má kdesi zaarchivovaný „rodný list" a to stačí. Ďalšia príjemná vec je, že sa nemusíme starať o alokáciu a uvoľňovanie pamäte, čo iste ocenia hlavne C-čkári :-).

Keďže Pylab má vlastný programovací jazyk a je nim Python, môžeme pohodlne pridávať k už zabudovaným funkciám svoje vlastné funkcie, ba celé balíky funkcií podľa potreby. Tí skúsenejší z vás môžu využívať v Pylab-e existujúce knižnice, napísané v C/C++ a FORTRAN-e. Takisto nie je problém čítať a zapisovať textové i binárne dáta do súborov.

Okrem širokej škály funkcií pre matice a vektory (napr. maticové operácie, jednoduchá manipulácia s indexami a submaticami, riešenie sústav lineárnych rovníc, inverzia matice, determinant, vlastné čísla a vlastné vektory, Fourierova transformácia, charakteristický polynóm, ...) máme v Pylabe tiež základné numerické algoritmy:

- interpolácia a aproximácia jedno- a dvojrozmerných dát (polynomická a splajnová),
- korene polynómov, riešenie nelineárnych rovníc a ich sústav,
- numerické integrovanie (aj pre dvojné a trojné integrály),
- optimalizácia funkcií viac premenných; existuje aj možnosť využitia Pythonu ako jazyka na modelovanie a riešenie optimalizačných úloh lineárneho a celočíselného programovania (napr. modul `Pulp`),
- riešenie diferenciálnych rovníc a ich sústav,
- funkcie pre pravdepodobnosť a štatistiku,
- špeciálne funkcie (ortogonálne polynómy, eliptické integrály, gama funkcia, Besselove funkcie cylindrické i sférické, Fresnelove integrály ...)
- funkcie na spracovanie signálu a obrazu.

Pritom si môžeme výsledky okamžite znázorniť pomocou grafov v pravouhlých alebo v polárnych súradniciach, prípadne parametricky. Máme aj špeciálne grafy (histogramy, koláčové grafy). Pylab dokáže tiež zobraziť vrstevnice plochy $z = f(x, y)$, príp. farebne vyplnený priestor medzi vrstevnicami (niečo ako mapa, kde farba závisí od nadmorskej výšky). Trojrozmerná grafika (kreslenie plôch) sa plánuje v novších verziách Pylabu.

V Pylabe môžeme vykonávať tiež ľubovoľné externé programy a príkazy operačného systému (napr. tak, že pred príkaz napíšeme výkričník), čo nám umožňuje využiť hocijaké naše už skompilované programy, ale aj iné programové systémy napr. na vytvorenie užívateľského rozhrania, generovanie dátových súborov pre Pylab, rozšírenie grafických možností Pylab-u, symbolické úpravy vzorcov (derivovanie, integrovanie, úprava algebraických výrazov napr. v systéme MAXIMA[25]).

Nikdy nezabúdajte na to, že práca s počítačom je **tvorivý dialóg** medzi vami a ním. Kedykoľvek vám niečo zahlási na obrazovku, venujte tomu pozornosť (hlavne, keď je tam slovko `"Error"` a keď sa to vypisuje červeno). Počítač si nevymýšľa a nesimuluje. Pokojne si prečítajte, čo vám píše, porozmýšľajte a snažte sa jeho správanie pochopiť. Potom sa môžete (niekedy je to aj na viackrát) pokúšať odstrániť to, čo mu na vašom „výpočte“ vadí. Ignorovaním chybových hlásení a vytrvalým opakovaním tých istých chybných príkazov (alebo ešte horšie, náhodnými experimentami a zmenami programu bez toho, aby ste rozumeli, čo vlastne robíte) sa k ničomu rozumnému nedopracujete. To je „hluboké nedorozumění “ a nie dialóg.

Poznámka: Systém Pylab sa neustále vylepšuje, jeho zdrojové kódy sú voľne prístupné na Internete. Ak si ho budete chcieť inštalovať doma, je dôležité, aby ste mali najnovšie stabilné verzie modulov, z ktorych sa skladá. Inak sa vám ľahko môže stať, že to, čo nám funguje bez problémov v škole, vám doma chodiť nebude. Uvádzame najnovšie verzie (stav zo začiatku marca 2007):

Program	Verzia	URL na download	Dokumentácia
Python	2.4.4	`http://www.python.org`	(VAN ROSSUM, 2006)
IPython	0.7.3	`http://ipython.scipy.org`	(PÉREZ, 2006)
Numpy	1.0.1	`http://www.numpy.org`	(OLIFANT, 2005)
SciPy	0.5.2	`http://scipy.org`	(OLIFANT, 2004)
Matplotlib	0.90	`http://matplotlib.sf.net`	(HUNTER, 2006)

Namiesto modulu *Numpy* (OLIFANT, 2005) sa predtým používali moduly *Numeric* alebo *Numarray*, pre nové inštalácie ich však neodporúčame, lebo sa ďalej nevyvíjajú. Samozrejme, ak budete potrebovať usmernenie či radu pri inštalácii, radi pomôžeme (vyššieuvedená emailová adresa autora). No obyčajne stačí prečítať si príslušné súbory `README` či `INSTALL` a riadiť sa podľa nich. Pritom budete mať oveľa menej problémov s inštaláciou a udržiavaním pod operačnými systémami UNIX-áckeho typu (napr. Linux, FreeBSD, Open BSD), než keby ste používali MS Windows. Verte, drvivú väčšinu problémov, ktoré sa objavujú na elektronických konferenciách, hlásia užívatelia MS Windows. Pre výpočtovo náročné aplikácie sú OS UNIX-áckeho typu jednoznačne vhodnejšie a spoľahlivejšie.

Niektoré distribúcie Linuxu (napr. Debian v nestabilnej a testovacej verzii) majú už pripravené binárne balíčky pre najnovšie moduly Pylabu, takže je ich možné inštalovať bez zdĺhavej kompilácie.

3.2.2 Zadávanie vektorov, matíc a operácie s nimi

Vektory a matice sú základnými objektami v Pylabe a tvoria aj základ pre drvivú väčšinu numerických algoritmov. Hlavnou výhodou Pylabu je, že dokáže vykonávať **maticové operácie a funkcie** (napr. výber submatíc a prvkov matíc a aj modifikáciu príslušných oblastí matice, vytváranie matíc z menších „blokov“, násobenie matíc, inverznú maticu, determinant matice, vlastné čísla a vlastné vektory a veľa iných vecí), nemusíme sa teda starať o „programovanie“ na úrovni prvkov matíc a vektorov s množstvom `for` cyklov a deklarácii. Tým sa zdrojový kód značne zmenší a sprehľadní.

[25] MAXIMA je kvalitný Open Source softvér na symbolické výpočty, dá sa nájsť na adrese `http://maxima.sourceforge.net`. Užitočné je tiež užívateľsky príjemné rozhranie wxMaxima s domovskou stránkou `http://wxmaxima.sourceforge.net`.

Pokiaľ si však na taký (pohodlný, ale iný) štýl práce zvyknete, budete mať tendenciu aj jednoduché veci (z pohľadu Pylabu) robiť zložito, hlavne ak ste si predtým zvykli na „klasiku“ v PASCAL-e či C-čku. Zlaté pravidlo Pylabu hlása: **Vyhýbaj sa cyklom, kde sa to len dá.** Všetko sa snažte zapisovať aj vo vašich programoch štýlom, blízkym k obvyklému matematickému zápisu, ako ste zvyknutí napr. na matematickej analýze či algebre.

Je dobre si uvedomiť, že vektor je špeciálnym prípadom matice (matica typu $(1 \times n)$ je riadkový a typu $(n \times 1)$ stĺpcový vektor). Ale tiež môžeme mať „jednorozmerný“ vektor, kde nám na riadkovosti či stĺpcovosti nezáleží a vtedy stačí zadať len jeden rozmer – počet prvkov.

Matica môže byť zadaná viacerými spôsobmi, napr.:

- pomocou príkazov a prostriedkov Pylabu (z klávesnice – zadaním prvkov matice alebo vytvorením matice v Pylab-skom programe)
- nahratím zo súboru (vytvoreného v Pylabe, C-čku, FORTRANe, exportovaného z databázy, príp. tabuľkového procesora alebo priamo vytvoreného v nejakom textovom editore).

Neodporúčame vám používať „školský“ spôsob interaktívneho zadávania rozmerov a prvkov matice z klávesnice (`Zadaj n: Zadaj A[1,1]` a pod.). Hodí sa to pre malé úlohy, ale s reálnym softvérom to nemá nič spoločné. Predpokladáme, že chcete byť skutoční profesionáli a nie „lepiči“, preto si trénujte správne programovacie návyky :-). Aj keď používate malé vstupné dáta, programujte tak, aby ste ten istý kód mohli použiť pre realistické, veľké vstupy. Zásadne používajte vstupné a výstupné (textové, pre rozsiahle dáta aj binárne) súbory, Python má veľmi príjemné prostriedky na prácu s nimi.

Budeme sa zaoberať najskôr priamym zadávaním matíc, teda prvým prípadom. Najjednoduchším spôsobom je **zadanie vymenovaním prvkov matice**. Niekoľko príkladov (skúšajte si „naživo“):

```
A = array([[1,2.5], [-1,3]]) # realna matica 2x2
B = array([-1,3,2**80])      # celocisel. jednorozmerny
                             # vektor, 2**80 je umocnovanie
Br= array([[1,-2,3]])        # riadkovy vektor
Bs= array([[1],[-2],[3]])    # stlpcovy vektor
C = rand(4,3)        # nahodna mat., rovnomer. rozd. [0,1)
D = zeros((3,3),'d') # matica nul (realnych, nie int.)
F = 4*ones((4,2))    # matica zo samych stvoriek (int.)
E = eye(4)           # jednotkova matica 4x4 (matematicka)
G = empty((5,5))     # neinicializovana mat. 5x5, rychle
Am= mat('1,2,4;4,5,6;7,8,9')
```

Možno ste pri zadávaní príkazov zmätení, že nevidíte nič na obrazovke. Príkaz vykonáte a akoby sa nič nestalo? No, stalo sa. Do premenných `A`, `B`, ..., `G`, `Am` máte priradené, to, čo ste zadali. Výpis premennej na obrazovku zariadite jednoducho napísaním jej mena alebo cez príkaz `print`, napr. `print(A)`. Keby ste niečo vytvorili, ale nikde nepriradili, automaticky sa to vypíše na obrazovku – to je najrozumnejšie, čo môže systém pre vás urobiť (samozrejme, nepriradené objekty nemôžete použiť v ďalších výpočtoch).

Väčšina z vás už asi uhádla, že texty za znakom `#` (až do konca riadku) v horeuvedených príkladoch sú komentáre. Teda sú to vysvetľujúce poznámky pre nás, ľudí a Pylab ich ignoruje. Je dobre si robiť komentáre, sú trvácnejšie, ako napr. poznámky v zošite :-). Vaše zdrojáky a tiež históriu interaktívnych príkazov si môžete kedykoľvek pozrieť na počítači, na ktorom robíte, alebo si ich stiahnuť cez sieť.

Často, hlavne keď už robíte v Pylabe dlhší čas, stratíte prehľad o tom, čo za premenné a funkcie ste si vyrobili. Takže namiesto toho, aby ste si lámali hlavu, či sa vaša matica volá `A`, `a` alebo `Aa`, **spýtajte sa systému – Hej, kto je tu?** – príkazom `who`. Ufrflanejší príkaz `whos` vám povie aj, aké typy majú vaše objekty a vypíše aj rozumné množstvo ich prvkov (ak sú to objekty sekvenčného typu). Ľubovoľný objekt môžete zlikvidovať príkazom `del`, napr. `del(G)` alebo pre viaceré objekty `del(Ob1,Ob2,Ob3)`. Ale tie matice, čo sme vytvorili vyššie, si nechajte, budeme sa na ne ešte odvolávať. Treba si zvyknúť tiež na to, že Pylab rozlišuje malé a veľké písmená, teda `AA`, `Aa`, `aA`, `aa` sú štyri rôzne mená objektov v Pylabe.

Asi ste pochopili, že príkaz `array` vytvorí jedno alebo viacrozmerné pole, ak mu zadáte nejaký *zoznam* (to sú tie veci v hranatých zátvorkách). No a `rand`, `zeros`, `ones`, `eye` slúžia na rýchle, pohodlné vytváranie špeciálnych matíc. Pomocou funkcie `mat` a textového reťazca môžete vytvárať matice v štýle MATLAB-u, teda riadky oddelené bodkočiarkou a prvky v riadku čiarkami či medzerami. Funkcie `rand`, `eye` sa vyvolajú jednoducho, napr. `rand(2,3)` bude matica náhodných čísel typu 2×3.

Funkcie `zeros`, `ones` majú nejako viac zátvoriek. Je to preto, lebo môžeme mať nuly či jedničky celočíselné, reálne alebo komplexné. Keď zadáme len usporiadanú dvojicu, napr. `zeros((2,4))` vytvorí sa celočíselná matica (vonkajšie zátvorky sú volanie funkcie `zeros`, vnútorné zas pre usporiadanú dvojicu, určujúcu rozmer matice, v našom prípade to bude matica typu 2×4). Keď však zavoláme `zeros((2,4),'d')` tak sa vytvorí matica s nulami reálnymi. Pre počítač je to podstatný rozdiel, lebo reálne čísla ukladá do pamäte a pracuje s nimi úplne inak ako s celými číslami.

Uvedieme písmenové označenia pre niektoré často používané typy prvkov číselných polí: `'?'` – *booleovské hodnoty* (v Pythone sa značia identifikátormi `True`, `False`), `'b'` – *byte*, `'l'` – *celé čísla*, `'d'` – *reálne čísla*, `'D'` – *komplexné čísla*. Takže napr. `ones((3,5),'?')` bude matica 3×5, ktorej všetky prvky budú mať hodnotu `True`.

Z vyššieuvedeného príkladu (pre maticu `B`) vidíte, že operácia umocňovania sa značí dvomi hviezdičkami (to je prevzaté z FORTRAN-u). Ale tiež vidíte, že aj také veľké číslo ako 2^{80} sa zobrazí správne. Je to preto, že **Python pracuje s celými číslami s ľubovoľnou presnosťou**, čo neplatí napr. pre PASCAL či C/C++. Takže v Pylabe môžeme rátať napr. faktoriály veľkých čísel alebo binomické koeficienty (môžete si vyskúšať, sú to funkcie `factorial`, `comb`).

Na matici `D=4*ones((4,2))` vidíte, že môžeme robiť aritmetické operácie medzi maticami a číslami. Skúste si napr. `C+2` a dostanete maticu, kde ku každému prvku matice `C` je pripočítaná dvojka. Podobne funguje aj odčítanie a delenie nenulovým reálnym či celým číslom.

Takisto, teda ako operácie po prvkoch, fungujú aj aritmetické operácie medzi maticami (obyčajne rovnakého rozmeru). Vyskúšajte `A*A`, `C+C`, `F/4.0`. Takže, na rozdiel od MATLAB-u, hviezdičkový operátor nie je maticové násobenie, ako ho poznáme z algebry. Ak chceme násobiť matice či vektory takto „matematicky“, použijeme funkciu `dot` (z anglického názvu „dot product“), napr. `dot(C,Bs)` urobí súčin matice `C` a stĺpcového vektora `Bs` v tomto poradí. Keby sme chceli urobiť súčin `dot(Bs,C)`, systém by nám to vyčítavo odmietol, nesedia mu rozmery.

Môžeme si položiť otázku, ako zistíme rozmery viacrozmerného číselného poľa? A vôbec, aké ďalšie funkcie máme na prácu s poľami k dispozícii? Tu prichádzame do oblasti, kde je náš pracovný jazyk – *Python* pre nás veľkou posilou, pretože je objektovo orientovaný. **Všetko, čo vytvoríme, je objekt.** Podľa toho, ako vznikol náš objekt `P`, má vždy nejaký dátový typ. Ten zistíme príkazom `type(P)`. Ako to poznáte z iných objektovo orientovaných jazykov, každý objekt má tiež svoje **dáta a metódy** (keď použijeme terminológiu z jazyka C++). K nim sa dostaneme v Pylabe cez bodkové označenie.

Napr. ak `A` je číselné pole (to je u nás dátový typ `ndarray`), potom jeho metódy (funkcie) sú `A.max`,

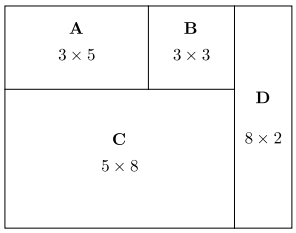
`A.sum`, `A.reshape`, `A.transpose` a veľa iných. Jeho dáta (atribúty) sú napr. `A.shape`, `A.dtypechar`, `A.ndim`. No odkiaľ sme sa to dozvedeli? To si musíme pamätať, či čo? Žiadny strach, pretože **Pylab inteligentne doplňuje názvy objektov a ich metód**. Používame na to kláves `TAB` – to je ten s dvomi diódami :-). Konkrétne, ak napíšete `A.` (tá bodka je tam dôležitá) a stlačíte `TAB`, vypíšu sa vám na obrazovku všetky dáta a metódy objektu `A`. Keď je toho veľa (akože aj je), napíšeme napr. `A.tra` a potom stlačíme `TAB`. A dostaneme len dve metódy `A.trace`, `A.transpose`. Ak nám nie je intuitívne jasné z názvu (alebo experimentovaním), o čom daná vec je, dáme si nápovedu, napr. príkaz `A.trace?`.

Teraz prezradíme, že `A.shape` je usporiadaná n-tica, určujúca rozmery poľa `A`. Konkrétne, pre maticu `A` z nášho príkladu nám Pylab vráti dvojicu `(2,2)`. Vyskúšajte, čo dostanete pre vektory `B`, `Br`, `Bs` a maticu `C`. Potom vám bude jasné, prečo nejde urobiť maticový súčin `dot(C,Br)`. Dá sa urobiť `dot(C,B)` a prečo?

Je zaujímavé, že rozmery matice môžeme meniť kedykoľvek, priradením do jej dátového atribútu `shape`, teda ak dáme `C.shape=(6,2)`, naraz sa z matice 4×3 stane matica 6×2. Rozmyslite si, ako by ste týmto štýlom z ľubovoľnej matice urobili jednorozmerné pole (malá nápoveda: `(m,n)=C.shape` priradí do premennej `m` počet riadkov a do premennej `n` počet stĺpcov matice `C`).

Je jasné, že zadávanie matíc prvkami sa hodí len pre nie príliš rozmerné matice. Numerická analýza sa však používa na riešenie reálnych problémov, kde matice rádove 100×100 sú „malé“. Našťastie, „**prvkami“** pri zadávaní matice možu byť nielen čísla, ale tiež celé **maticové bloky** či **submatice**. Matice v aplikáciach majú často blokovú štruktúru a v Pylabe máme na ich konštrukciu funkciu `bmat`.

Napríklad matica `M` typu 8×10 môže byť „pozliepaná“ z menších blokov `A`, `B`, `C`, `D`:




```
A=ones((3,5))
B=zeros((3,3))
C=3*ones((5,8))
D=-ones((8,2))
P=bmat('A,B;C'); M=bmat('P,D')
```

Najskôr vytvoríme pomocnú maticu `P` typu 8×8, ktorej prvý „riadok“ je tvorený maticami `A`, `B` a druhý „riadok“ je `C`. Potom k `P` pridáme sprava maticu `D`. Všimnite si, že argument pre funkciu `bmat` zadávame ako textový reťazec; v ňom sú prvky (bloky) v riadku oddelené čiarkami a riadky sa oddeľujú bodkočiarkou (tak je to zvykom v MATLAB-e). Vidíte tiež, že na jednom riadku môžeme zadať viac príkazov, ak ich oddelíme bodkočiarkou (platí aj v Pythone nielen v Pylabe). Na spájanie matíc (vedľa seba alebo pod sebou) máme ešte funkciu `concatenate`.

Cvičenie 3.1 *Vytvorte „šachovnicu“, teda maticu 8×8, kde na bielych poliach budú nuly a na čiernych jedničky. Myslí sa inteligentný algoritmus, s čo najmenšou námahou a nie otrocké zadávanie všetkých prvkov. A samozrejme taký aby sa dal ľahko zovšeobecniť aj na väčšie matice.*

V numerike aj v úlohách inžinierskej praxe sa stretávame s pásovými maticami. Príkaz `diag` nám umožňuje poskladať maticu zo šikmých „pásov“, rovnobežných s hlavnou diagonálou. Skúste takto vytvoriť maticu `M` rozmeru 10×10, ktorá bude mať na hlavnej diagonále trojky, nad ňou čísla -1 a pod diagonálou čísla -2; ostatné prvky matice `M` sú nulové.

Často potrebujeme vektory, ktorých prvky sú „pravidelne rozmiestnené“ medzi hodnotami `v_beg` *(počiatočná hodnota)* a `v_end` *(koncová hodnota)*. Ak poznáme počet prvkov `n` takého vektora `v`, vytvoríme ho jednoducho príkazom

$$\texttt{v} = \texttt{linspace}(\texttt{v\_beg}, \texttt{v\_end}, \texttt{n}).$$

Vektory ekvidištančných (rovnako od seba vzdialených) hodnôt, generované pomocou `linspace` využívame často na kreslenie grafov (kriviek, plôch). Napríklad príkazmi

$$\texttt{x} = \texttt{linspace}(-\texttt{pi}, 3 * \texttt{pi}, 100); \quad \texttt{y} = \texttt{sin}(\texttt{x}); \quad \texttt{plot}(\texttt{x}, \texttt{y})$$

vytvoríme 100-prvkový vektor `x` hodnôt, rovnomerne pokrývajúcich interval $\langle -\pi, 3\pi \rangle$, ďalej vypočítame hodnoty sínusu vo **všetkých** týchto bodoch `x` a vykreslíme graf funkcie $y = \sin x$ pre x v intervale $\langle -\pi, 3\pi \rangle$. Skúste si to, nech vidíte s minimálnou námahou už aj nejaký obrázok :-). Neskôr si o grafike v Pylabe povieme podrobnejšie.

Na vytvorenie vektora celočíselných ekvidištančných hodnôt je výhodnejší príkaz `arange(i_beg, i_end, step)`, ktorý vytvorí vektor ekvidištančných hodnôt s krokom `step` začínajúci hodnotou `i_beg`, pričom koncová hodnota nie je väčšia ako `i_end - 1`. Napríklad príkaz `arange(1, 10, 2)` vytvorí vektor `(1, 3, 5, 7, 9)`, ale `arange(1, 9, 2)` dá vektor `(1, 3, 5, 7)`. Ak nezadáme argument `step`, berie sa krok 1 (vektor za sebou idúcich celých čísel), napr. `arange(1, 6)` je vektor `(1, 2, 3, 4, 5)`. Ak zadáme jedinú hodnotu, napr. `arange(n)`, dostaneme vektor n hodnôt $0, 1, \ldots, n-1$. Pythonský príkaz `range` sa správa takisto, ale namiesto číselného vektora vracia zoznam.

3.2.3 Vyberanie a priraďovanie prvkov a submatíc, operátory porovnávania a ich využitie

Jednotlivé prvky vyberáme z matice pomocou indexovania hranatými zátvorkami, ako je to zvykom aj v iných programovacích jazykoch. Teda to, čo matematicky zapíšeme ako $\mathbf{A}_{ij}$, budeme v Pylabe písať ako `A[i,j]`. Je dôležité vedieť, že číslovanie riadkov a stĺpcov začína od nuly (tak, ako je to v jazyku C), takže prvok v „ľavom hornom rohu“ matice `A` je `A[0,0]`. Celé riadky vyberáme zadaním jedného indexu, napr. `A[0]` je prvý riadok matice `A`.

Namiesto číselných indexov môžu byť tzv. *slices* (po slovensky hádam „krajce“), ktoré vyzerajú ako

$$\texttt{i\_beg} : \texttt{i\_end} : \texttt{k},$$

kde `i_beg` je začiatočný index, `i_end` koncový index a `k` je krok. Táto dvojbodková syntax je ekvivalentná príkazu `arange(i_beg, i_end+1, k)`, takže platí všetko, čo sme povedali vyššie o tomto príkaze. Ibaže krajce sa dajú použiť len pri indexovaní polí. Ukážeme to na príkladoch, kde predpokladáme, že `A` je matica 5×5, generovaná napr. príkazom `A=rand(5,5)`

```
A[:2,:2]      # submatica - prve dva riadky a stlpce
A[1::2,::2]   # riadky 1,3 a stlpce 0,2,4
A[1:3,3]      # riadky 1,2 a stlpec 3
A[:,2]        # vsetky riadky, stlpec 2
A[::-1,:]     # riadky matice A v opacnom poradi
A[[3,0,2],:]  # riadky 3,0,2 a vsetky stlpce
```

Myslíme, že z uvedených príkladov je jasné, čo sa stane, keď sa vynechá začiatočný index, koncový index či krok. Osobitne zaujímavé je vytvorenie vektora s opačným poradím prvkov príkazom `v[::-1]`.

Výber prvkov pomocou „dvojbodkového označenia“ môžeme robiť na oboch stranách priraďovacieho príkazu, napr.

$$\texttt{A}[:, [1, 2, 4]] = \texttt{B}[:, 1:4]$$

nahradí stĺpce 1, 2, 4 matice `A` prvými tromi stĺpcami matice `B`. Pri podobnom priraďovaní treba dbať na rozmerovú kompatibilitu (pravá a ľavá strana musia mať ten istý rozmer). Na pravej strane môže byť tiež len jedno číslo, takže príkazom

$$\texttt{A}[:2, :3] = 0$$

vynulujeme celú príslušnú submaticu matice `A`. To je jeden zo spôsobov, ako sa vyhnúť explicitným cyklom (typu `for`, `while`) a zrýchliť beh programu.

V algebre ste často robili operácie s riadkami matice, napr. nejaký násobok prvého riadku ste pričítali k druhému riadku matice `A` a výsledok zapísali do druhého riadku. To sa urobí aj v Pylabe ľahko (nezabúdajme na indexovanie riadkov a stĺpcov od nuly):

$$\texttt{A}[1] = \texttt{A}[1] + 3.5 * \texttt{A}[0]$$

Ďalší príklad:

$$\texttt{dx} = \texttt{x}[1:] - \texttt{x}[:-1]$$

vypočíta vektor diferencií $\Delta x_i = x_{i+1} - x_i$ pre $i = 1, 2, \ldots, n-1$, ak `x` je n-prvkový vektor. Záporné indexy sú špecialitou Pythonu a počítajú sa od konca, takže `x[:-1]` je vektor `x` bez posledného prvku.

Pomocou dvojbodkoveho označenia by ste si mohli skúsiť urobiť tú šachovnicovú maticu núl a jedničiek, ktorú sme hore konštruovali z blokov. Stačilo by vyrobiť vektory prvých dvoch riadkov a tie potom „ob dva“ opakovať.

Prechádzame k operátorom porovnávania (t. j. `<`, `>`, `<=`, `>=`, `==`, `!=`). Predpokladajme, že `A` je náhodná matica, vytvorená ako `A=rand(4,4)`. Skúste si, čo urobia príkazy `A>0.7`, `A<0`, `A != A`, `A == A`. Ako vidíte, výsledok je booleovská matica s rovnakým rozmerom ako matica `A`, v ktorej hodnoty `True` sú na miestach, kde je podmienka porovnávania splnená a hodnoty `False` tam, kde nie je.

Operátory porovnávania môžeme tiež kombinovať s logickými operátormi `&`, `|`, `~`. Pre tých, čo poznajú Python – nepoužívajte na číselné polia operátory `and`, `or`, `not`, lebo tie vám vynadajú (nevedia porovnávať polia „po prvkoch“).

Najkrajšie na tom všetkom je, že výsledok porovnávania (booleovská matica) sa dá použiť na ďalšiu manipuláciu s tými prvkami matice, kde je pravdivostná hodnota `True`. Napr.

$$\texttt{A}[(\texttt{A} > 0.8) \mid (\texttt{A} < 0.1)] = 0$$

vynuluje v matici `A` všetky prvky, ktoré sú väčšie ako 0.8 alebo menšie ako 0.1. Porozmýšľajte, ako by ste vynulovali všetky prvky, ktorých absolútna hodnota je menšia ako dané kladné číslo ε, napr. $\varepsilon =$ `4.4e-16`.

Pekný príklad: zistime počet prvkov v náhodnej matici 100×100, ktoré sú väčšie ako 0.7 (keďže viete niečo z pravdepodobnosti, mali by ste tušiť, koľko ich bude):

```
A=rand(100,100)  # nahodna matica 100 x 100
A7=(A>0.7)       # bool. matica, True kde a[i,j] > 0.7
P7=A7.sum()      # sucet prvkov A7 je ten pocet
                 # (True -> 1, False -> 0 v sucte)
```

Všimnite si, ze metóda `sum` sčíta všetky prvky daného číselného poľa (aj viacrozmerného). Ak potrebujeme vektory stĺpcových alebo riadkových súčtov, treba zadať ešte „súradnicovú os“ – 0 pre stĺpcové a 1 pre riadkové súčty, teda `S_riad=A.sum(0); S_stlp=A.sum(1)`.

3.3 Základné funkcie na prácu s polynómami a maticami

Spomenieme niekoľko funkcií pre polynómy a matice, ktoré asi budete často používať.

Polynóm $P_n(x) = c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ je v Pylabe reprezentovaný vektorom jeho koeficientov, pričom začíname od najvyššej mocniny premennej x. Napr. polynóm

$$P_5(x) = x^5 - 2x^3 + 7x^2 - x - 3$$

je daný vektorom `c=[1, 0, -2, 7, -1, -3]`. Teda žiadne „ručné“ programovanie mocnín premennej a pod. Na počítanie hodnôt polynómu používame funkciu `polyval`, takže

$$\texttt{y5} = \texttt{polyval}(\texttt{c}, \texttt{rand}(100))$$

uloží do premennej `y5` hodnoty horeuvedeného polynómu, vypočítané v 100 náhodných bodoch.

Dvojica funkcií `roots`, `poly` slúži na výpočet koreňov polynómu a opačne, na rekonštrukciu koeficientov polynómu, ak sú dané jeho korene. Vyskúšajme si (príkaz `from scipy import poly` na prvom riadku slúži na načítanie funkcie `poly` z modulu `scipy` a vysvetlíme ho podrobnejšie v ďalšom odseku):

```
from scipy import poly
W=poly(arange(1.0,21.0)) # Koeficienty Wilkinsonovho
                         # polynomu s korenmi 1,2,...20
R=roots(W20)             # korene toho polynomu
                         # to, co uvidite, je zivot :-)
```

Na násobenie a delenie polynómov máme funkcie `polymul`, `polydiv`. Vynásobme polynómy $p = (x-1)^5$, $q = x^2 + 5x - 6$ a súčin vydelíme opäť polynómom q, čím by sme mali dostať polynóm p a zvyšok 0. Je to tak?

```
from scipy import polymul, polydiv
cp=poly([1.0]*5)      # polynom p s 5-nasob. korenom 1
cq=[1, 5, -6]         # polynom q
cs=polymul(cp,cq)     # vysledok nasobenia - polynom
cpd,zv=polydiv(cs,cq) # s/q = p (cpd = cp plati?)
```

Maticami sa zaoberáme dosť aj preto, aby sme vedeli riešiť sústavy lineárnych rovníc. Ak matica systému je štvorcová a regulárna, t. j. s nenulovým determinantom, môžeme použiť funkciu `solve`. Napr:

```
A=rand(1000,1000)   # nahodna matica 1000 x 1000
b=rand(1000)        # nahodna prava strana
x=solve(A,b)        # x je riesenie syst. A x = b
norm(dot(A,x)-b)    # velkost vektora A x - b
                    # musi byt velmi male cislo
```

Pravá strana b systému môže byť aj matica typu $n \times m$ ak matica systému je typu $n \times n$. Je to akoby m pravých strán naraz (stĺpce matice b). Napr.

$$\texttt{Ai} = \texttt{solve}(\texttt{A}, \texttt{eye}(1000, \texttt{dtype} = \texttt{'d'}))$$

vypočíta inverznú maticu k matici `A`. Ale nikdy to tak nerobte, je to neefektívne. A vôbec, inverznú maticu potrebujeme skutočne zriedkavo. Potrebujeme riešiť sústavy lineárnych rovníc. To vieme pomocou metód numerickej analýzy aj pre sústavy so všeobecnou obdĺžnikovou maticou (KAUKIČ, 1998; BUŠA, 2006).

3.4 Programovanie v Pylabe

3.4.1 Základy na prežitie

Pylab je interaktívny interpretačný jazyk. Vy mu zadávate rôzne príkazy (v príkazovom riadku na obrazovke), on ich vykonáva. Keď však chcete robiť komplikovanejšie veci, napíšete si váš program s pomocou nejakého textového editora (my budeme používať `nedit`, `http://www.nedit.org/`, čo je skratka z Nirvana Editor :-), uložíte na disk a potom ho spustíte v našom interaktívnom prostredí cez príkaz `run`.

Programovanie v Pylabe je vlastne programovaním v Pythone. Takže máme k dispozícii všetky dátové typy (čísla celé, reálne a komplexné, textové reťazce, zoznamy, usporiadané n-tice, asociatívne polia) a príkazy (vrátane cyklov, podmienok, ošetrovania výnimočných situácií) z Pythonu. Na produktívnu prácu v Pylabe však stačí veľmi málo a to si ukážeme na jednoduchých príkladoch.

Príklad 3.2 *Napíšme funkciu, ktorá vytvorí Hilbertovu maticu $n \times n$, t. j. maticu s prvkami*

$$h_{ij} = 1/(i+j-1),\ i,j = 1,2,\ldots,n.$$

Najskôr musíme správne nakonfigurovať editor, v ktorom budeme naše zdrojové programy písať. Je to veľmi dôležité, pretože Python používa indentáciu (odsadenie textu) na označenie blokov v programe (teda to, čo sa v PASCALe značí pomocou `begin`, `end` a v C-čku pomocou zložených zátvoriek). **Editor treba nastaviť tak, aby namiesto tabelátorov vkladal štyri medzery** (ak to vo vašom editore nejde, prejdite na iný :-). A potom **na indentáciu zásadne využívajte kláves TAB**, nie medzery. Ak ešte nemáte vybraný vhodný editor, skúste si pozrieť `http://stani.be/python/spe/blog/`, editor, či skôr IDE `SPE` je napísaný v Pythone a je veľmi dobrý.

Ak sa budete týchto rád držať, nebudete mať žiadne problémy. Ak nie, dočkáte sa mnohých hlášok typu `Indentation error` a budete nadávať na Python. Najhoršie, čo môže byť, je zdroják, v ktorom sú pomiešané tabulátory a medzery v indentácii. V každom editore to môže potom vyzerať inak. Čo je horšie, úplne sa môže zmeniť zmysel zdrojáku. Autor to môže potvrdiť z vlastnej skúsenosti, keď začínal s Pythonom veľmi krvopotne práve z uvedených malicherných dôvodov.

Takže predpokladáme, že máte editor správne nastavený. Dajte si vytvoriť nový súbor, ktorý hneď pomenujte napr. `hilbert.py`. To ukončenie súboru na `.py` pomôže editoru, aby nastavil zvýrazňovanie syntaxe, príp. dopĺňanie kľúčových slov, a pod.

V našom malom súbore vidíte definíciu funkcie, potom inicializáciu matice (alokáciu pamäte). Ďalej sú dva vnorené príkazy cyklov a nakoniec priradenie hodnôt prvkom matice (naraz dvom – pretože Hilbertova matica je symetrická, t. j. $h_{ij} = h_{ji}$). Na tomto príklade dobre vidieť použitie indentácie. Cyklus `for j in range(i,n):` je vnorený do prvého cyklu, pretože je na riadku odsadený vzhľadom na ten prvý cyklus. Príkaz priradenia `H[i,j]=H[j,i]=1.0/(i+j+1)` je súčasťou oboch cyklov, ale príkaz `return` už ani do jedného z cyklov nepatrí, pretože začína na rovnakej úrovni indentácie ako vonkajší cyklus. Keby sme pridali do súboru ďalší riadok, ktorý by začínal príkazmi na úrovni `def`, tak tieto príkazy by už nepatrili do funkcie `hilb`. Na takúto logiku sa dá pomerne rýchlo zvyknúť. No a obsah súboru bude nasledujúci:

```
from numpy import empty

def hilb(n):
    H=empty((n,n),'d')          #inicializacia matice H
    for i in range(n):          # priradenie prvkov
```

```
        for j in range(i,n):
            H[i,j]=H[j,i]=1.0/(i+j+1)
    return H
```

Keď ten súbor uložíte (pre `nedit` je to klávesová skratka `Ctrl-S`), nechajte si editor otvorený (väčšinou programy nebývajú bez chýb a treba ich ladiť, teda opakovane prepisovať v editore). Vráťte sa do interaktívneho prostredia Pylabu a príkazom `run hilbert.py` načítate do Pylabu všetko, čo je v tom súbore definované. V našom prípade je to len funkcia `hilb` a môžete sa pomocou príkazov `who`, `whos` presvedčiť, že ju máte.

Naša funkcia sa vyvolá (použije) podobne ako vstavané funkcie, teda, že zadáme meno funkcie a nejaké (konkrétne) argumenty (alebo, keby argumenty neboli, tak len prázdne zátvorky). Napr. `hilb(5)`, `hilb(12)`. No nie je blbuvzdorná, lebo napr. vykoná sa aj `hilb(5.735)`. Ale vždy začíname tak, že nám ide o správnu funkcionalitu programu a predpokladáme inteligentného užívateľa (obyčajne prvý, kto to testuje, je sám tvorca programu :-) a až neskôr sa vrátime k ošetreniu chýb vstupu a iným menej podstatným veciam.

Zapamätajme si, že **dvojbodky na konci riadkov sú súčasťou syntaxe** a píšu sa len v týchto prípadoch:

1. za definíciou funkcie, napr. `def factorial(m):`,
2. na začiatku cyklov `for`, `while`,
3. v podmienkach, za kľúčovými slovami `if`, `elif`, `else`, `finally` a tiež pri ošetrovaní výnimiek (kľúčové slová `try`, `except`).

Keď chceme, aby naša funkcia vrátila nejaký objekt, musíme to explicitne povedať príkazom `return`. Keď to neurobíme, funkcia vráti „Pythoňácke nič“, čo je špeciálny objekt `None`. Samozrejme, z funkcie môžeme vrátiť aj viac objektov naraz, napr. cez usporiadanú n-ticu ($n \geqq 2$).

V Pythone funguje viacnásobné priradenie, napr. `a=b=1`, alebo podobne `a,b=1,2`, t. j. usporiadanej dvojici `(a,b)` sa priradia príslušné hodnoty z číselnej dvojice `(1,2)`. Dva objekty zameníme jednoducho príkazom `a,b=b,a`. Všimnite si, že usporiadaná n-tica to sú objekty, oddelené čiarkou. Písať okrúhle zátvorky niekedy nie je nutné.

Často používaným trikom je „rozdistribuovanie“ výsledku funkcie do viacerých premenných. Napr. ak `MP="Alžbeta Kalininova"` je textový reťazec, tak príkazom

$$\texttt{meno,priezvisko} = \texttt{MP.split}()$$

urobíme priradenia `meno="Alžbeta"`, `priezvisko="Kalininova"`. Pozrite si, čo robí metóda `split` pre objekty typu `str`.

V Pylabe môžeme pohodlne zapisovať aj „intervalové“ nerovnosti, napr.

$$-1 \leqq x < 3 \text{ zapíšeme ako } \texttt{-1 <= x < 3}.$$

Zaujímavý je prvý riadok vyššieuvedeného súboru `hilbert.py`. V Pythone (podobne, ako je to v C-čku) je väčšina funkcionality prístupná v **moduloch**, čo je obdoba PASCAL-ovských „units“ alebo C-čkovských knižníc. Napr. funkcia `empty` je v module `numpy` a sprístupníme ju (importujeme do nášho programu) práve príkazom: `from numpy import empty`. Naraz môžeme z toho istého modulu importovať aj viac objektov, napr.:

```
from numpy import zeros, ones, eye, empty, dot.
```

Ako vieme, v ktorom module (resp. submodule) je naša funkcia? Nuž, vyskúšame moduly, tvoriace Pylab v tomto poradí `numpy`, `scipy`, `pylab`. Základné funkcie na prácu s maticami a vektormi budú v module `numpy`, tie druhé dva sú nadstavbou nad týmto fundamentálnym modulom. V interaktívnom prostredí importujeme oba výpočtové moduly, teda `import numpy, scipy`. Predpokladajme, že chceme vedieť, odkiaľ treba importovať funkciu `solve`. Skúsime `numpy.sol` a tiež `scipy.sol`, či nám to doplní `TAB`-om. Ak nie, tak tam tá funkcia nie je.

Ak niečo nie je v moduloch `numpy` ani `scipy`, treba vyskúšať submoduly v `scipy`. Najužitočnejšie pre nás budú submoduly `scipy.linalg`, `scipy.interpolate`, `scipy.integrate`, `scipy.optimize`, `scipy.stats`. V našom prípade zaberie `scipy.linalg.sol`, lebo to sa nám už `TAB`-om doplní. Teda príkaz na import funkcie `solve` bude:

```
from scipy.linalg import solve.
```

Modul na kreslenie, `pylab`, používame obyčajne tak, že z neho importujeme všetko, teda `from pylab import *`. Zvyknite si, že **všetky importy dávame na začiatok zdrojového súboru**, nikdy nie roztratené sem-tam alebo niekde v našich definovaných funkciách. Inak sa budete veľmi čudovať, prečo raz máte a raz nemáte niečo k dispozícii. Ladenie programu s roztratenými importami by bolo vrcholne nepríjemné.

Treba si ujasniť ešte jednu fundamentálnu vec: **ak je nejaký objekt dostupný v interaktívnom prostredí, nemusí byť dostupný vo vašom zdrojovom súbore**. Každý Pylabský zdrojový súbor sa chová ako „minimodul" a má svoj vlastný priestor mien. Napríklad po spustení Pylabu máte v interaktívnom prostredí k dispozícii funkcie `array`, `dot`, `solve`, `zeros`, `bmat`, `empty` a mnoho ďalších. Je to zaistené tým, že sme ich pridali do vhodných inicializačných súborov, ktoré si Pylab načíta pri štarte. Ale to váš zdroják nerobí, a preto všetky tieto funkcie musíte importovať, ak ich chcete používať. No a samozrejme, načítať vami definované funkcie vykonaním vášho zdrojáku cez príkaz `run`, napr. `run factorial.py`.

Príklad 3.3 *Napíšme funkciu na výpočet faktoriálu nezáporného celého čísla n. Uložte ju ako* `factorial.py`.

```
def factor(n):
    assert(type(n)==int and n>=0), "Need natural number"

    fct=1
    if n in [0,1]:
        return fct
    else:
        for k in range(2,n+1):
            fct = k*fct   # alebo ako v C-cku: fct *=
    return fct
```

Príklad 3.4 *Vytvoríme funkciu na výpočet n-tého člena a_n Fibonacciho postupnosti, ktorá je daná predpisom*

$$a_0 = 0, a_1 = 1; \quad a_{k+2} = a_{k+1} + a_k \ \ pre \ \ k = 0, 1, \ldots.$$

```
def fib(n):
    assert(type(n)==int and n>=0), "Need natural number."
    if n in [0,1]:
```

```
        return n
    else:
        a_akt,a_predch=1,0
        for k in range(2,n+1):
            a_akt,a_predch=a_akt+a_predch,a_akt
        return a_akt
```

Nové je v týchto funkciách ošetrenie vstupu tak, aby to neprešlo, ak vstup nie je celé číslo a ešte k tomu aj nezáporné. Funkcia `assert` zaistí, aby v nej uvedené predpoklady boli splnené. Ak nie sú, generuje sa chyba a môže sa prípadne vypísať objasňujúci text.

Podobne ako operátor `and` aj ostatné logické operátory `or`, `not` sú pomenované príslušnými anglickými slovami (ale ako sme povedali už vyššie, pre číselné polia používajte `&`, `|`, `~`).

V tom príklade vidíte tiež jednoduché vetvenie programu podľa podmienky – pre n patriace do zoznamu `[1,2]` je faktoriál rovný jednej, inak sa počíta v cykle opakovaným násobením. Rozmyslite si, že v našom príklade tú podmienku `else` ani netreba. Všeobecný príkaz na vetvenie môže obsahovať ešte niekoľko vetiev, začínajúcich kľúčovým slovom `elif`.

3.4.2 Práca s dátovými súbormi

Na uschovanie dát v textových súboroch a ich opätovné načítanie máme dvojicu funkcií `save`, `load`.

Príklad 3.5 *Vytvorme náhodný neorientovaný graf, ktorý má dvanásť vrcholov a 53 hrán. Zoznam hrán zapíšme do súboru a potom tie hrany z neho načítajme.*

Graf sa tu chápe v zmysle dopravnej siete (cestnej, železničnej), t. j. ako množina uzlov (vrcholov), pospájaných cestami (hranami). Vytvorenie grafu bude väčší problém než tie súbory. Poďme cestou najmenšieho odporu a hrubou silou. V module `numpy.random` (keďže je to submodul pre pravdepodobnosť a štatistiku, je logické, že hľadáme tam) nájdeme našim obľúbeným doplňovaním klávesom `TAB`, že existuje funkcia `numpy.random.random_integers` a tá sa nám hodí. Keďže nám nerobí problém napísať všeobecnú funkciu pre náhodný graf s `n` vrcholmi (očíslujeme ich poradovými číslami od 1 do `n`) a `m` hranami, urobíme to. Uložte si ju do súboru `randgraph.py`. V tej funkcii budeme generovať hrany (dvojice čísel 1 až `n`, pričom na poradí nezáleží, teda vždy môžeme zobrať prvý prvok menší, ako druhý), kým ich nebude presne `m`:

```
from numpy.random import random_integers as rndint
def randgraph(n,m):
    Ph=0            # pocet hran
    Zh=[]           # zoznam hran
    odpad=0         # pocet zamietnutych hran
    while True:
        u,v=rndint(1,n),rndint(1,n)
        # pripustne su len hrany pre u<v
        # a take, ktore este nemame
        if u<v and (not (u,v) in Zh):
            Zh.append((u,v)); Ph += 1
        else:
            odpad += 1
```

```
        if Ph==m:
            break
    return Zh, odpad
```

Všimnite si, že pre pohodlnosť sme si pri importe premenovali funkciu `random_integers` – dali sme jej kratšie meno. Ak sa čudujete, načo nám je premenná `odpad`, tak prakticky nanič. Ale môžeme si podľa nej urobiť predstavu, aká je naša metóda hrubej sily (ne)účinná. Viac ako polovička generovaných hrán sa nepoužije. Skúste si vyvolať našu funkciu (niekoľkokrát) tak, ako to potrebujeme, teda

```
Zh,odp=randgraph(12,53)
```

a uvidíte, že v premennej `odp` bude vždy okolo 200 odvrhnutých hrán. Potom, na „produktívne“ využitie upravte funkciu `randgraph` bez premennej `odpad`, čím sa jej zdroják dosť zjednoduší.

Zaujímavosťou funkcie `randgraph` je jej správanie sa pri volaniach ako napr. `randgraph(12,67)`. Odporúčame vyskúšať. A hlavne potom upraviť tak, aby sa podobné psie kusy neopakovali (je možné, že v panike aj počítač resetnete, ale nerobte to :-). Radšej si spomeňte na niektoré elementárne fakty z kombinatoriky.

Teraz môžeme ten zoznam hrán uložiť do textového súboru s názvom `gr12_53.txt` príkazom:

```
save('gr12_53.txt',Zh,'%i')
```

Tretí, nepovinný parameter je formátovací reťazec v štýle C-čka, teda `"%i"` znamená uloženie v celočíselnom formáte (`"%d"` by bol formát pre reálne čísla v dvojitej presnosti – typ `double` v C-čku). Súbor `gr12_53.txt` si môžete pozrieť a prípadne upravovať v textovom editore; vyskúšajte, čo to urobí, ak zavoláte funkciu `save` bez tretieho parametra.

Teraz zničíme premennú `Zh`, ako by sme to urobili s hocijakým Pylabským objektom, teda príkazom `del(Zh)`. Môžete sa presvedčiť cez príkaz `who`, že už ju nenájdete medzi živými, t. j. existujúcimi premennými. Načítame ju ale znova z nášho súboru príkazom

```
Zh_nove=load('gr12_53.txt')
```

Výsledkom bude číselné pole s 53 riadkami a dvomi stĺpcami; každý riadok je jedna hrana. Ale keď si dáte niektorý riadok vypísať, zistíte, že jeho dva prvky sú reálne a nie celé čísla. Našťastie, existuje metóda na pretypovanie celého poľa (volá sa `astype`) a pomocou nej dostaneme už celočíselné pole hrán `Zh` (to isté, ako sme mali na začiatku):

```
Zh=Zh_nove.astype(int)
```

Ak je dát veľa, oplatí sa uschovávať ich v binárnom formáte, ktorý je pre ľudí nečitateľný, ale výsledný súbor je menší a načíta sa do počítača nepomerne rýchlejšie, ako textový súbor s ekvivalentným obsahom.

Ukážeme si to na náhodnej matici 1000×1000 s celočíselnými prvkami, ktorú nagenerujeme príkazmi:

```
import scipy
# jednorozm. pole s milion prvkami v rozsahu 1-157
R=scipy.stats.randint.rvs(1,157,1e6)
R.shape=(1000,1000)  # premenime ho na maticu 1000x1000
```

Na vysvetlenie – v module `scipy.stats` je veľa spojitých i diskrétnych rozdelení náhodnej premennej. Každé rozdelenie má ešte rôzne metódy, napr. `rvs` (random variable samples), t. j. generovanie náhodných vzoriek z tohto rozdelenia, `pdf` – hustota pravdepodobnosti, `cdf` – distribučná funkcia alebo `pmf` (probability mass function), teda pravdepodobnostná funkcia diskrétneho rozdelenia, atď. My sme využili rovnomerné diskrétne rozdelenie `randint` pre celé čísla v intervale $\langle 1, 157 \rangle$, z ktorého sme urobili náhodný výberový súbor s milión prvkami.

Každé číselné pole má metódu `tofile`, pomocou ktorej ho môžeme binárne zapísať do súboru. V našom prípade urobíme:

```
# otvorime subor s nazvom Rbig.bin na zapisovanie
outfile=file("Rbig.bin","w")
R.tofile(outfile)             # zapiseme donho maticu R
outfile.close()               # zatvorime subor, schluss
```

Textový súbor, ktorý by sme dostali príkazom `save("Rbig.txt",R,"%i")` je síce trocha menší ako binárny súbor, ale zápis do binárneho súboru je viac ako osemdesiatkrát rýchlejší. Pri načítavaní dát ukážeme, ako sme to merali. Nezabudnite, že súbor, do ktorého zapisujete, treba po skončení zápisu a pred ďalšími manipuláciami s ním (obyčajne čítaním) uzavrieť príkazom `close`, ako sme to urobili povyššie. Inak môže byť neúplný (autor hovorí z vlastnej skúsenosti). Časť dát môže ešte čakať na zápis v pamäťovom bufferi – ten môžeme vyprázniť aj volaním metódy `flush`, teda v našom prípade by to bolo `outfile.flush()`.

Dáta zo vzniknutého binárneho súboru dostaneme použitím funkcie `fromfile` a priradíme ich do novej premennej `Rs`:

```
Rs=fromfile("Rbig.bin",int)
```

Druhý parameter hovorí, že typ prvkov poľa je celočíselný (iné možné typy sú napr. `float`, `complex`, `bool`, `str`). Pole, ktoré dostaneme ako výstup z funkcie `fromfile` je vždy jednorozmerné, ale vieme jeho rozmery ľahko upravovať priradením do atribútu `shape`, teda z vektora `Rs` o milión prvkoch dostaneme maticu typu 1000×1000 jednoducho príkazom

```
Rs.shape=(1000,1000)  # tvar Rs sa zmeni na 1000 x 1000
```

Sme zvedaví, či pôvodná matica `R` a načítaná matica `Rs` sú rovnaké. Presvedčime sa o tom príkazom

```
all(Rs==R) # Vsetky prvky Rs su rovne zodpovedajucim prvkom R?
```

Ak nám tento príkaz vypíše `True`, obe matice sú rovnaké. Dúfame, že je to aj u vás tak.

Ako je to s časom čítania textového a binárneho súboru? Aby sme to zistili, importujeme si z modulu `time` funkciu `time`, ktorá meria čas aj na mikrosekundy. Potom si zapíšeme čas pred začiatkom výpočtu. Po jeho skončení vypíšeme na obrazovku rozdiel medzi tým počiatočným a aktuálnym časom:

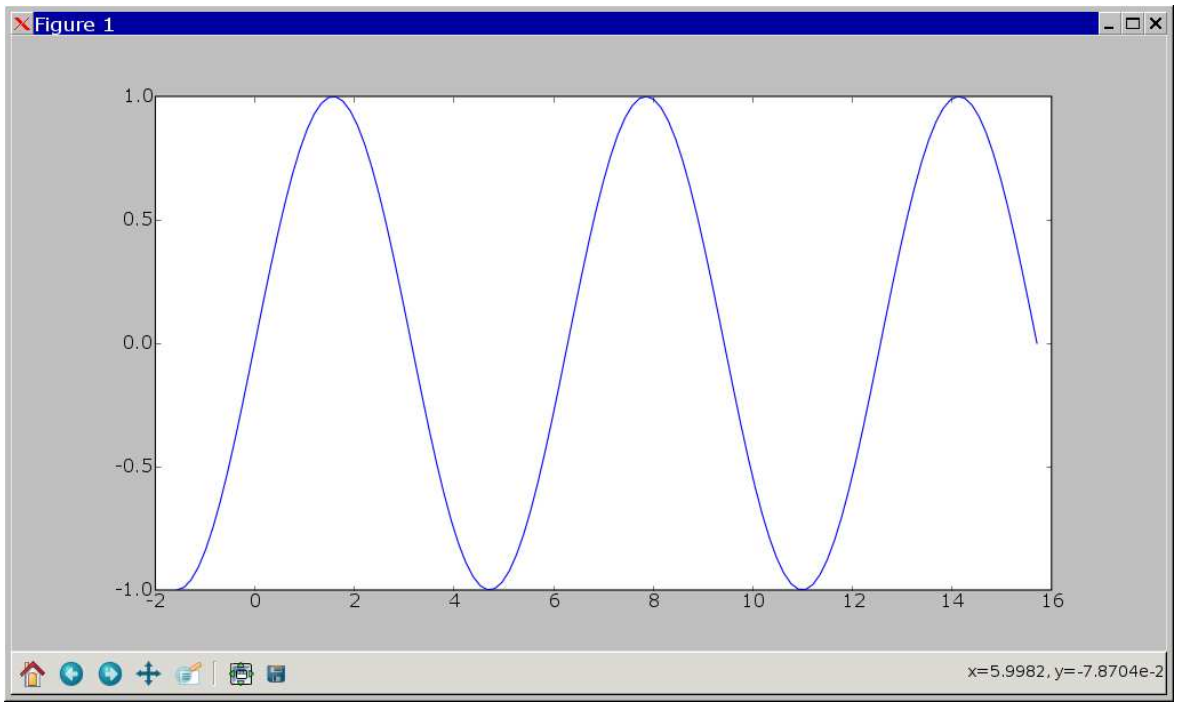
```
from time import time
tb=time(); Rs=fromfile("Rbig.bin",int);time()-tb #0.0127 sec.
tt=time(); load("Rbig.txt",Rt);        time()-tt #3.345  sec.
```

V komentároch vidíte aktuálne časy, ktoré sme dostali na našom katedrovom serveri (Athlon64 3500+, RAM 2GB). Zasa je čítanie z binárneho súboru nepomerne rýchlejšie (asi 270-krát v tomto prípade).

3.5 Grafika v Pylabe

3.5.1 Základná filozofia, grafické objekty

Grafika v Pylabe je založená na module Matplotlib (HUNTER, 2006). Používame ju dvomi spôsobmi – v dávkovom spracovaní, napr. keď generujeme obrázky pre WEB-server, ktoré my ani neuvidíme, alebo v interaktívnom režime, keď si tie naše obrázky chceme aj prezerať a upravovať.



Základným príkazom pre jednoduché grafy funkcií a kriviek je príkaz `plot`. Nakreslíme si graf funkcie $y = \sin(x)$ pre $x \in \langle -\pi/2, 5\pi \rangle$:

```
x=linspace(-pi/2,5*pi,120)      # hodnoty nezavisle premennej
y=sin(x)                        # funkcne hodnoty v bodoch x
plot(x,y)                       # nakreslenie grafu
```

Na obrazovke sa objaví nové okno, v ktorom bude nakreslený náš graf. Všimnite si v dolnej časti ikony na interaktívnu prácu s obrázkom. Ikonka domčeka vás vráti k pôvodnému pohľadu, ak ste medzitým urobili nejaký výrez (to sa dá pomocou ikony piatej zľava – tej s lupou). Ikona diskety umožňuje uloženie obrázku v niekoľkých formátoch (napr. `.png`, `.jpg`, `.eps`). Vyskúšajte si tú interaktívnu prácu najlepšie sami. Ak zvolíte zväčšenie výrezu niekoľkokrát, zistíte, že náš graf je vlastne lomená čiara. Je to 120 bodov v rovine (ich súradnice sú určené vektormi `x`, `y`), pospájaných úsečkami. Zatiaľ okno s grafikou neuzavierajte.

Vidíte, že na nakreslenie jednoduchého grafu nám stačí niekoľko príkazov. Aktuálne okno s grafikou (`current figure` v terminológii Pylabu) môžete modifikovať veľa spôsobmi, napr. skúste si:

```
grid(1)              # kresli sietku na grafe
title("Graf funkcie sin")  # titulok grafu
ylabel("Os y")             # popis osi y
```

Zasa treba zdôrazniť, že hoci v interaktívnom prostredí Pylabu máte všetky príkazy pre grafiku (napr. `plot`, `grid`, `title`, `legend`) k dispozícii, vo vašich zdrojákoch ich musíte importovať pomocou príkazu „importuj všetko z modulu pylab“, t. j.:

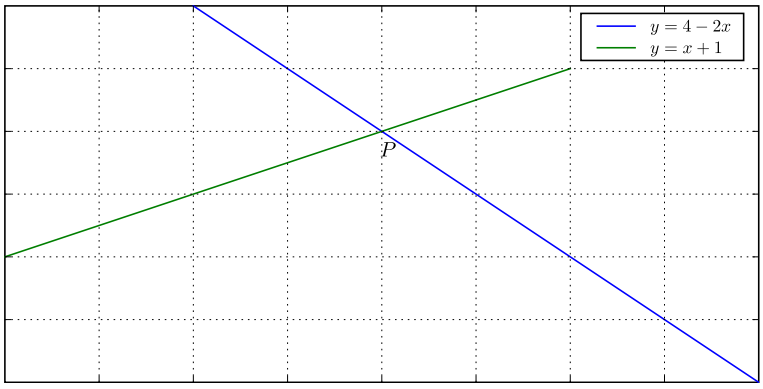
```
from pylab import *
```

Príklad 3.6 *Znázornime riešenie sústavy rovníc*

$$\begin{aligned} 2x + y &= 4, \\ -x + y &= 1. \end{aligned}$$

Riešením je bod P, ktorý je priesečníkom priamok $y_1 = 4 - 2x$, $y_2 = x + 1$. Znázornime ich časti ako úsečky, určené bodmi $A_1 = (3, -2)$, $B_1 = (0, 4)$ resp. $A_2 = (-1, 0)$, $B_2 = (2, 3)$.

```
# do plot-u osobitne zadavame
# x-ove a y-ove suradnice bodov
plot([3,0],[-2,4],[-1,2],[0,3])  # dve usecky
grid(True)
legend(('y=4-2x','y=x+1'))
t=text(1.05,1.7,'P')
title("Solution of system of linear equations")
```



Vyzerá to celkom pekne, len sa zdá, že to písmeno P by mohlo byť trochu inde (polohu sme vybrali pomocou súradníc, ktoré Pylab ukazuje, keď sa pohybujete kurzorom myši v obrázku). Nie je problém to dodatočne napraviť. Sme v Pythone a všetko je objekt. Aj náš obrázok je poskladaný z objektov.

Koncepcia grafiky v Pylabe je založená na tom, že máme obrázkové okná (`figures`) a v každom z nich môže byť niekoľko súradnicových systémov, ktoré môžeme umiestňovať v obrázku, kam sa nám zachce, aj jeden cez druhý. Grafické objekty (úsečky, body, mnohouholníky, atď.) sa pridávajú vždy do aktuálnych súradnicových osí.

Aktuálny obrázok a aktuálne súradnicové osi (ak sme nezatvorili všetky grafické okná) získame príkazmi:

```
fig=gcf()       # get current figure
axs=gca()       # get current axes
```

Keď si chceme nám už dôverne známym spôsobom (napísať `axs.` a stlačiť `TAB`) pozrieť metódy a dáta objektu `axs`, vychŕli to na nás okolo 230 možností. Namiesto písmena *P* by mohol byť ľubovoľný text, skúsime teda doplniť pomocou klávesu `TAB` text `axs.te`. Zistíme, že také objekty v aktuálnych osiach máme dva: `text`, `texts`. Ten prvý je metóda na pridávanie textu do obrázka na zadanej pozícii, ako zistíme z helpu (teda cez `axs.text?` alebo `?axs.text`). Druhý objekt, `texts`, je zoznam, lebo keď ho chceme vypísať cez `axs.texts`, dostaneme niečo ako

```
[<matplotlib.text.Text instance at 0x2aaab6c2f248>].
```

Ten zoznam má jediný prvok, inštanciu objektu `matplotlib.text.Text`. Teda náš text dostaneme ako nultý prvok zoznamu, t. j. `TP=axs.texts[0]`. Potom si zas môžeme pozrieť metódy toho objektu, ktorých je viac ako 100, ale ak sa obmedzíme na metódy, ktoré niečo nastavujú (začínajúce sa na `set` – to je užitočné aj pri iných grafických objektoch), dostaneme po doplnení `TAB`-om asi toto (niektoré riadky sme vynechali):

```
TP.set_alpha                TP.set_ma
TP.set_backgroundcolor      TP.set_name
TP.set_bbox                 TP.set_position
TP.set_clip_on              TP.set_size
TP.set_color                TP.set_style
TP.set_family               TP.set_text
TP.set_ha                   TP.set_x
TP.set_horizontalalignment  TP.set_y
```

Je jasné, že pozíciu, na ktorej sa text vypisuje, nastavíme cez `set_position`, teda pozrieme si o nej help, zistíme, že očakáva usporiadanú dvojicu súradníc a skúsime

```
TP.set_position((1.02,1.62)) # trochu dolava a dole.
```

Keď to urobíme, v obrázku sa nič nezmení. Stačí však (myšou) mierne zmeniť rozmery okna s obrázkom alebo (čo je programátorsky čistejšie) zavolať funkciu `draw()` na prekreslenie aktuálneho obrázka a zmeny sa prejavia.

To už vyzerá celkom dobre, ale chceli by sme, aby sa to písmeno *P* vypisovalo kurzívou (teda nastaviť štýl písma). Môže sa to asi robiť pomocou `set_fontstyle` alebo `set_style`. To druhé je kratšie, skúsime nápovedu a je to ono (povie nám aj to, že možné štýly sú `'normal'`, `'italic'`, `'oblique'`). Teda, stačí prikázať

```
TP.set_style('italic'); draw()
```

a zmena v štýle písma sa prejaví. Skúste si podobne zmeniť veľkosť písma, jeho typ, ale aj text, ktorý sa vypisuje (namiesto *P* dajte napr. *Solution*).

Všimnite si tiež, že obrázok, ktorý máte v tejto knižke je asi trochu iný, než máte vy na obrazovke. Je to tým, že sme spracovanie textu, hlavne matematických výrazov, zverili profesionálnemu sádzaciemu systému LaTeX. Pokiaľ ste na Linuxe a máte nainštalovaný TeX, docieli sa to maličkou zmenou v horeuvedených príkazoch (uvádzame len nové alebo odlišné riadky)

```
rc('text', usetex=True)
...
```

```
legend((r'$y=4-2x$',r'$y=x+1$'))
t=text(1.01,1.6,r'$P$')
...
```

To, že sme sa tak dlho venovali nejakému bezvýznamnému písmenku P, nebolo náhodou. Chceli sme na ňom ukázať podstatné veci z možností grafiky Pylabu. To, čo platilo o objektoch typu `Text`, bude platiť aj o súradnicových osiach a detailoch ich vykreslenia a hlavne o čiarach a bodoch, z ktorých sa naše grafy budú skladať.

3.5.2 Ukážky dvojdimenzionálnej grafiky

Príklad 3.7 *Nakreslíme graf funkcie $y = \sin x$ a jej Taylorovych polynómov piateho a desiateho stupňa.*

Pripomeňme si, že Taylorov rozvoj funkcie $y = \sin x$ je

$$\sin x = x - \frac{x^3}{3\,!} + \frac{x^5}{5\,!} + \cdots + (-1)^k \frac{x^{2k+1}}{(2k+1)\,!} + \cdots .$$

Napíšte funkciu, ktorá vráti hodnoty Taylorovho polynómu stupňa n v zadaných bodoch x, uložte si ju ako súbor `taylsin.py`:

```
from numpy import array

def taylsin(x,n):
    assert(type(n)==int and n>0)
    x=array(x,'d')   # ak bol zoznam, urobime pole
    x2=x*x
    f=1.0            # faktorial - inicializacia
    T=x.copy()       # Taylorov polynom - inicializacia
    xa=x.copy()      # aktualna mocnina x v Tayl. pol.
    for k in range(3,n,2):
        f  *= -((k-1)*k)  # - pre striedanie znamienok
        xa *= x2          # exponent x sa zvacsi o 2
        T += xa/f         # dalsi scitanec do T
    return T
```

Všimnime si niektoré pozoruhodnosti v tejto funkcii. Použitím príkazu `x=array(x,'d')` zaistíme, že ďalej vo funkcii bude premenná `x` fungovať ako číselné pole (aj z jedného čísla sa dá vyrobiť pole). Bez toho by sme nemohli používať aritmetické operácie, napr. `x*x`. Používali sme C-čkovský variant zápisu priradení, teda `T += xa/f` namiesto `T = T + xa/f`.

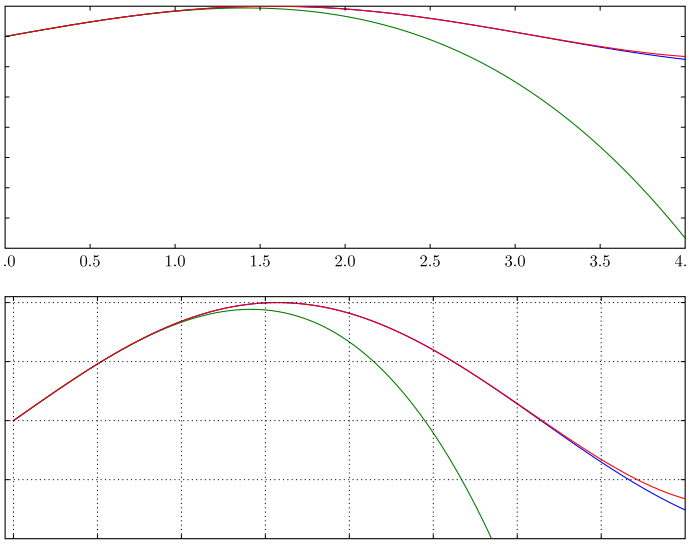
V príkazoch na inicializáciu `T`, `xa` sme zobrali **kópie** poľa `x`, pretože napr. `T=x` by spôsobilo prepisovanie poľa `x`; priradenie pre polia nevytvára novú kópiu, ale len ďalší odkaz (referenciu) na pôvodné pole. Obidve premenné budú ukazovať na to isté miesto v pamäti.

Teraz už len ostáva zvoliť vhodný interval na kreslenie (stačí od nuly, lebo aj sínus aj tie Taylorove polynómy sú stredovo symetrické podľa počiatku súradníc) a použiť príkaz `plot`

```
xx=linspace(0,4,120)
T5,T10=taylsin(xx,5),taylsin(xx,10)
lines=plot(xx,sin(xx),xx,T5,xx,T10)
```

Už predtým ste si možno všimli, že kedykoľvek ste do obrázka niečo pridali (príkazmi `plot`, `title`, `text`, `xlabel`, ...) tak sa na obrazovku niečo vypísalo, teda tie príkazy vracajú nejaké objekty. Sú to objekty toho typu, čo sme pridali, napr. `matplotlib.text.Text` pre `title`, `text`, `xlabel` alebo `matplotlib.lines.Line2D` pre `plot`. Keď si ich priradíme do nejakých premenných, budeme mať neskôr prístup k ich dátam a metódam, teda budeme ich môcť ľubovoľne modifikovať. Konkrétne, teraz máme v premennej `lines` zoznam troch čiar, ktoré sme do obrázku nakreslili.

Samotný obrázok nám ale neprináša vnútorné uspokojenie, lebo tam vyčíňa puberťácky polynóm piateho stupňa (ponáhľa sa do mínus nekonečna :-) a tým odpútava naše vnímanie od slušného správania sa dospelého polynómu deviateho stupňa.



Pylab robí za nás určenie rozsahu na súradnicových osiach tak, aby sa naše čiary zmestili do grafu. Ale kedykoľvek môžeme tieto rozsahy zmeniť príkazom `axis([xmin,xmax,ymin,ymax])`. V našom prípade dolnú hranicu na osi x dáme troška pod nulu (aby sa nezlievali čísla pri popisoch osí) a rozsah na osi y nastavíme od -1 do 1.05 (aby krivky neboli celkom na hornom okraji obrázka).

Na aktuálny obrázok pridáme aj sieťku na ľahšie odčítanie súradníc. Funkcia `grid()` bez parametrov je prepínač – ak nebola sieťka, bude a naopak, ale môžeme aj explicitne špecifikovať napr. `grid(True)`.

```
axis([-0.05, 4, -1, 1.05])
grid()
```

Môžete si tie dva obrázky porovnať. Iste uznáte, že ten druhý vyzerá lepšie. Len je škoda, že pri čiernobielom vytlačení všetky tie tri čiary vyzerajú skoro rovnako. To sa budeme teraz snažiť napraviť.

Keď máme tie naše tri čiary v zozname `lines`, roztriedime ich do troch premenných príkazom `lsin,lt5,lt9=lines`. Keď si zas známym spôsobom pozrieme metódy pre nastavenia (napr. napíšeme `lt5.set` a stlačíme kláves TAB), nájdeme metódu `set_linestyle` (alebo skrátený názov `set_ls`) a z nápovedy sa dozvieme, aké rôzne štýly čiar môžeme nastaviť. Urobíme

```
lsin.set_ls('-'); lt5.set_ls('-.'); lt9.set_ls(':')
```

a budeme mať krivku pre sínus znázornenú plnou čiarou, Taylorov polynóm piateho stupňa bodkočiarkovane a polynóm deviateho stupňa bodkovane.

V nasledujúcej tabuľke vidíme rôzne možné štýly. Prvé štyri sú pre čiary, všetky ďalšie znázorňujú zadané body (x, y) značkami.

Tabuľka 1: Štýly čiar v Pylabe

`-`	neprerušovaná čiara		`--`	čiarkovaná čiara
`-.`	bodkočiarkovaná čiara		`:`	bodkovaná čiara
`.`	body		`,`	obrazovkové pixely
`o`	kruhy		`^`	trojuholníky základňou dole
`v`	trojuholníky obrátene		$<$	trojuholníky doprava
$>$	trojuholníky doľava		`s`	štvorce
`+`	symboly plus		`x`	symboly tvaru x
`D`	kosoštvorce		`d`	tenké kosoštvorce
`1` až `4`	trojnožky rôzne otočené		`h`	šesťuholníky
`H`	otočené šesťuholníky		`p`	päťuholníky

Podobne vieme nastaviť farbu čiar a ich hrúbku. Napr. všetky tri čiary nakreslíme čiernou farbou:

```
lsin.set_c('k'); lt5.set_c('k'); lt9.set_c('k').
```

Štýly a farby čiar sa dajú nastavovať priamo v príkaze `plot`, o čom si prečítajte podrobnejšie v nápovede k tomuto príkazu. Takže, ak našu grafiku vymažeme príkazom `clf()` (clear figure) a dáme

```
plot(xx,sin(xx),'-k',xx,T5,'-.k',xx,T10,':k')
```

dostaneme tiež čierne čiary rôznych štýlov.

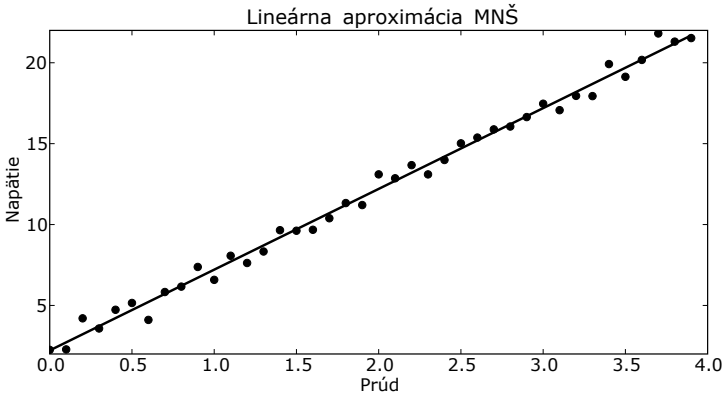
Príklad 3.8 *Majme* 40 *dátových bodov* (x_i, y_i)*, ktoré sú výsledkom meraní. Vieme, že teoreticky by mala byť medzi nimi lineárna závislosť* $y = ax + b$*. Urobme pre tieto dáta priamkovú aproximáciu pomocou metódy najmenších štvorcov (MNŠ) a nakreslime pôvodné body aj aproximačnú priamku.*

Tie dáta simulujeme pomocou malých náhodných odchýliek s normálnym rozdelením. Na aproximáciu použijeme funkciu `polyfit`, ktorá robí polynomickú aproximáciu pomocou MNŠ. Dátové body nakreslíme ako malé čierne kruhy a priamku dáme trocha hrubšiu:

```
from pylab import *

x=arange(0,4,0.1)      # 40 bodov na osi x
odch=0.5*randn(len(x)) # nahodne odchylky
y=2+5*x+odch           # linearne data s chybami
a,b=polyfit(x,y,1)     # posledny argument je stupen polyn.
plot(x,y,'ko',x, a*x+b,'-k',lw=3)
```

Ten titulok grafu a popis osí s diakritikou sme vyrobili pomocou príkazov

```
title(unicode("Lineárna aproximácia MNŠ","latin2"))
xlabel(unicode("Prúd","latin2"))
ylabel(unicode("Napätie","latin2"))
show()        # aby sa zmeny v obrazku prejavili
```



Nové v príkaze `plot` je, že sme použili štýl bodov na kreslenie dátových bodov (inak by to Pylab pospájal plnou lomenou čiarou), a že sme pri znázorňovaní priamky použili pomenovaný argument `lw` (skratka za `linewidth`) na nastavenie hrúbky čiary.

Príklad 3.9 *Nakreslíme ilustráciu k tomuto príkladu: „Zistite plochu rovinnej oblasti, ohraničenej parabolami* $y_1 = x^2, y_2 = x + 3 - x^2$.“

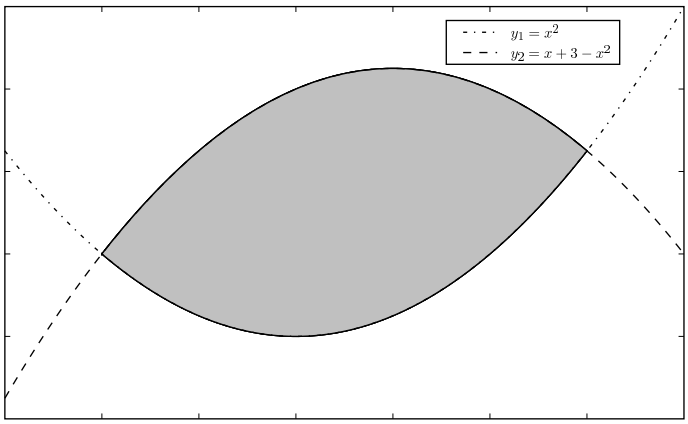
V podstate nám ide o farebné vyplnenie oblasti medzi dvomi krivkami. Pylab má funkciu `fill` na farebné vyplnenie mnohouholníka. Naše dve krivky tvoria vlastne veľký mnohouholník, ak jednu z nich budeme prechádzať v obrátenom poradí.

Priesečníky kriviek y_1, y_2 sa ľahko zistia riešením kvadratickej rovnice

$$y_1 = y_2 \Leftrightarrow 2x^2 - x - 3 = 0 \Leftrightarrow x = -1, x = 3/2.$$

```
x=linspace(-1,3/2.0,100)
yp1,yp2=x*x,x+3-x*x
xpoly=concatenate(( x,   x[::-1]))
ypoly=concatenate((yp1,yp2[::-1]))
fill(xpoly,ypoly,facecolor="#C0C0C0")
```

Ak si tieto príkazy píšete do súboru, musíte dať na jeho začiatku príkaz `from pylab import *`, inak by vám systém vypisoval, že nepozná funkcie `linspace`, `concatenate`, `fill`. Platí to aj pre nižšieuvedené ukážky grafiky, ak ich budete spúšťať zo súborov. Pripomeňme, že `x[::-1]` je vektor `x` s opačným poradím prvkov. V príkaze `fill` vidíte, že farby možno zadávať podobne, ako sa to robí v HTML-súboroch, ako reťazce RGB (Red, Green, Blue), pričom `#000000` je čierna, `#FFFFFF` biela farba a napr. `#0000AA` je tmavomodrá.

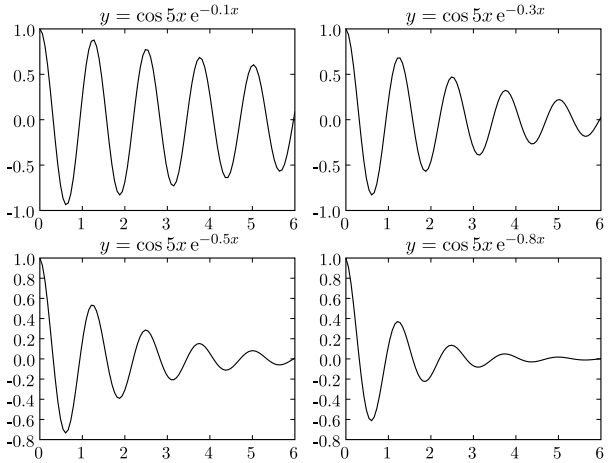


Aby to vyzeralo krajšie, potiahneme tie paraboly trochu aj mimo uvažovanej oblasti, napríklad pre $x \in \langle -1.5, 2.0 \rangle$, rozlíšime ich typom čiar, pridáme legendu a nadpis grafu.

```
xvel=linspace(-1.5,2.0,140)
yv1,yv2=xvel*xvel,xvel+3-xvel*xvel
plot(xvel,yv1,'-.k',xvel,yv2,'--k')
legend((r"$y_1=x^2$", r"$y_2=x+3-x^2$"),loc=(0.68,0.86))
title("Plocha rovinnej oblasti.")
```

Zaujímavé je tu manuálne umiestnenie legendy. Keby ste nedali parameter `loc`, legenda sa umiestni v pravom hornom rohu a prekryje jednu z parabol. Experimentovaním sme prišli k vhodným súradniciam tak, aby legenda ničomu nezavadzala. Pozor, sú to vždy „normalizované súradnice“ od 0 do 1 na oboch osiach.

Príklad 3.10 *Nakreslíme do jedného grafu štyri krivky pre tlmené kmitanie, dané rovnicou* $y = \cos(5x) \cdot \mathrm{e}^{-\alpha x}$ *s tlmiacimi činiteľmi* $\alpha = 0.1, 0.3, 0.5, 0.8$.



To sa robí pomocou príkazu `subplot`. Keď chceme vedľa seba `m` obrázkov, ktoré sú usporiadané v `n` riadkoch, dáme príkaz

`subplot(m,n,k)`.

Ten nás zároveň nastaví na k-ty obrázok (počíta sa po riadkoch smerom odhora a v riadku zľava doprava).

```
x=linspace(0,6,100)
y1=cos(5*x)*exp(-0.1*x); y2=cos(5*x)*exp(-0.3*x)
subplot(2,2,1);  plot(x,y1,'k')
title(r"$y=\cos 5x\,{\rm e}^{-0.1 x}$")
subplot(2,2,2);  plot(x,y2,'k')
title(r"$y=\cos 5x\,{\rm e}^{-0.3 x}$")
```

V tomto listingu sme vynechali príkazy pre tretí a štvrtý obrázok, lebo je to všetko na jedno kopyto.

V obrázku, ktorý vidíte, sme manuálne upravili rozmery obrázkov (pomocou `Subplot Configuration Tool` voľby v menu obrázkového okna), lebo inak by popisy osí a nadpisy grafov boli príliš natesno. Dá sa to, samozrejme, aj priamo cez atribúty aktuálneho obrázka, teda napr.:

```
cf=gcf()          # get current figure
cf.subplotpars.left=0.06; cf.subplotpars.right=0.94
```

Že sa to tak volá, zistili sme doplňovaním pomocou `TAB`-u. Súradnice sa udávajú normalizované, od 0 do 1. Všimnite si tiež matematické výrazy v textoch (sú to tie, ohraničené dolármi) – syntax, čo tam vidíte, je prevzatá z LaTeX-u.

Príklad 3.11 *Znázornite v tom istom grafickom okne funkcie*

$$f_1(x) = \frac{2}{1+x^2} - 1, \quad f_2(x) = \sin\left(x^3\right)$$

a kružnicu $x^2 + y^2 = 0.81$, *pričom body kružnice budú znázornené krúžkami po každých* 10°.

Najskôr (keďže to nemáme zadané) sa rozhodneme pre interval na osi x, na ktorom chceme funkcie znázorňovať. Napr. teraz si zvolíme $-2 \leqq x \leqq 2$.

Zostrojíme dostatočne „hustý“ vektor bodov, rovnomerne pokrývajúci daný interval (pri kreslení grafu sa jednotlivé body spoja úsečkami; na hladké zobrazenie na obrazovke obyčajne stačí rádove stovka bodov). Potom vypočítame hodnoty funkcií f_1, f_2 v týchto bodoch.

```
x=linspace(-2,2,100)
f1=2.0/(1+x*x)-1; f2=sin(x**3)
```

Tieto dve funkcie vykreslíme, pričom predpíšeme, aby druhá funkcia bola nakreslená čiarkovane a zelenou čiarou (*green*)

```
plot(x,f1,x,f2,'--g')
```

Na zostrojenie kružnice môžeme použiť parametrický tvar jej rovnice

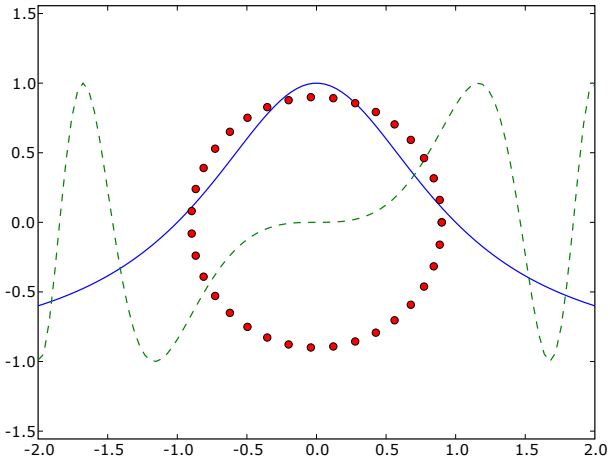
$$x = r\cos\varphi, \quad y = r\sin\varphi, \quad \varphi \in \langle 0, 2\pi\rangle.$$

Zostrojíme vektor uhlov `fi`, body `xt`, `yt` na kružnici a dokreslíme tú kružnicu červenými kolieskami:

```
fi=linspace(0,2*pi,37)         # prečo nie 36?
xt=0.9*cos(fi); yt=0.9*sin(fi)
plot(xt,yt,'or')
```

Všimnime si, že kružnica je na obrazovke akási pretiahnutá v smere osi y, t. j. je to vlastne elipsa. Keď chceme, aby sa jednotkové dĺžky na osiach zobrazovali rovnako (na obrazovke, ale aj po vytlačení obrázku, o čom sa môžete presvedčiť na vlastné oči alebo pomeraním), použijeme príkaz

```
axis("equal")
```



Ten istý príkaz v tvare `axis([xmin, xmax, ymin, ymax])` môžeme použiť na manuálne nastavenie rozsahov na súradnicových osiach, ak nám nevyhovuje to, čo Pylab vyberie automaticky.

Krivky v polárnych súradniciach (teda v tvare $\rho = \rho(\varphi)$, $\varphi \in \langle \alpha, \beta \rangle$) môžeme kresliť pomocou príkazu `polar`. Po vyčistení obrázka pomocou `clf()` môžeme tú predchádzajúcu kružnicu nakresliť príkazom

```
polar(fi,[0.9]*len(fi),'or')
```

Druhý argument `[0.9]*len(fi)` je zoznam, ktorého každý prvok je 0.9 a má dĺžku rovnakú ako vektor `fi`.

Cvičenie 3.12 *Elipsu s poloosami $a = 2$, $b = 1$ môžeme nakresliť v polárnych súradniciach, ak si uvedomíme, že jej parametrická rovnica je*

$$x = a\cos\varphi,\ y = b\sin\varphi, \quad \varphi \in \langle 0, 2\pi \rangle$$

a že polárny sprievodič $\rho(\varphi)$ sa dá určiť ako $\rho = \sqrt{x^2 + y^2}$.

V Pylabe by to mohlo vyzerať takto:

```
fi=linspace(0,2*pi,120); x=2*cos(fi); y=sin(fi)
ro=sqrt(x**2+y**2)
polar(fi,ro)
```

Ak si pozriete obrázok, iste uznáte, že naša elipsa si zaslúži prívlastok „prvoaprílová“. Vysvetlite, prečo vyzerá tak nezvykle ...

3.5.3 Znázorňovanie funkcií dvoch premenných

Príklad 3.13 *Nakreslime vrstevnicový graf funkcie dvoch premenných*

$$z = \sin(xy^2) - \cos(x^2 + y) \text{ pre } -2 \leqq x \leqq 2, \ -2.5 \leqq y \leqq 2.5.$$

Najskôr vygenerujeme ekvidištančné vektory `xx`, `yy` hodnôt na súradnicových osiach v zadaných intervaloch:

```
xx=linspace(-2,2,80);     yy=linspace(-2.5,2.5,100)
```

a tie potom použijeme v príkaze `meshgrid`, ktorý vráti matice `X`, `Y` x-ových a y-ových bodov obdĺžnikovej sieťky, na ktorej sa musia počítať funkčné hodnoty z. Napr. pre `xm=[1,2,3]`, `ym=[0,1,2,3]` vráti `meshgrid(xm,ym)` tieto matice:

```
        1   2   3              0   0   0
Xm =    1   2   3      Ym =    1   1   1
        1   2   3              2   2   2
        1   2   3              3   3   3
```

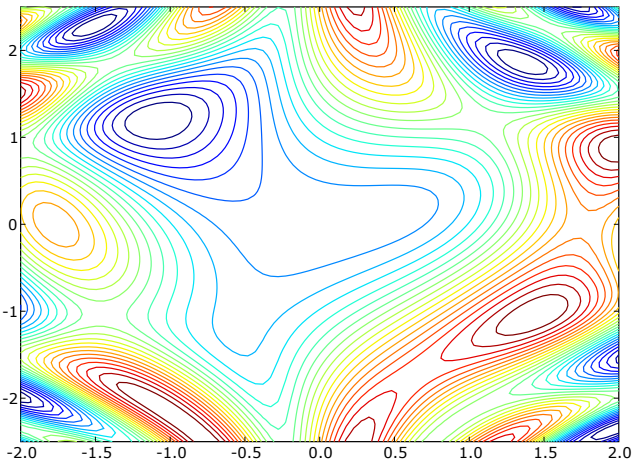
V našom prípade budú mať matice `X`, `Y` rozmery 80×100 a dostaneme ich príkazom

```
X,Y=meshgrid(xx,yy)
```

Použijeme ich na výpočet funkčných hodnôt a nakreslenie vrstevnicového grafu (contour plot) príkazom `contour`:

```
Z=sin(X*Y*Y)-cos(X*X-Y)
contour(X,Y,Z,20)
```

Dostanete nasledujúci obrázok. Ako sa dá dočítať z nápovedy k príkazu `contour`, jeho najjednoduchšie volanie je `contour(Z)`. Rozmyslite si, čo sme dosiahli ďalšími argumentami (týka sa to popisu osí a počtu vrstevníc).



Existuje príkaz `contourf`, ktorý priestor medzi vrstevnicami vyplní farebne, podobne ako to býva na mapách. Pomocou príkazu `colorbar` tiež môžeme pridať farebnú stupnicu ku grafu, takže vieme, aká farba prislúcha danej veľkosti znázorňovanej veličiny.

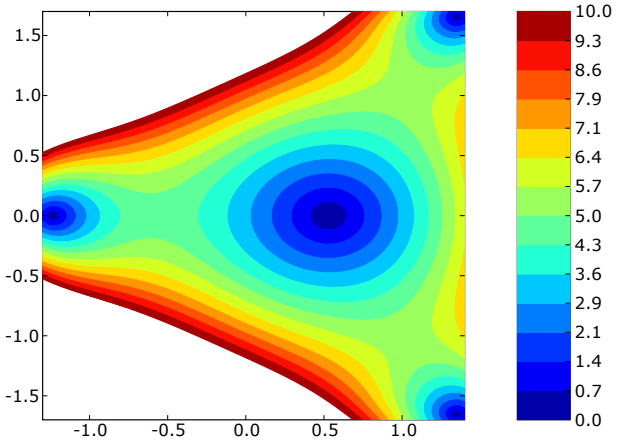
Príklad 3.14 *Pre plochu* $f(x,y) = |p_4(x+iy)|$, *kde* $z = x + iy$,

$$p_4(z) = z^4 - 2z^3 + 2z^2 + 5z - 3, \quad -1.3 \leqq x \leqq 1.4, \quad -1.7 \leqq y \leqq 1.7$$

nakreslite farebne vyplnený vrstevnicový graf. Voľte vhodné úrovne pre vrstevnice tak, aby ste získali dobrú predstavu o tej ploche a tiež aj o polohe komplexných koreňov polynómu $p_4(z)$.

Polynóm $p_4(z)$ má koreň $z_0 = x_0 + iy_0$ práve vtedy, keď $f(x_0 + iy_0) = 0$. Postupujeme podobne, ako v predchádzajúcom príklade:

```
from numpy import polyval
xx,yy=linspace(-1.3, 1.4,80),linspace(-1.7, 1.7,80)
X,Y=meshgrid(xx,yy)
j=complex(0,1)              # imaginarna jednicka, 0+1j
Z=X+j*Y                     # sietka v komplex. rovine
c4=[1,-2,2,5,-3]            # koeficienty polynomu
Fxy=abs(polyval(c4,Z))      # vypocet f(x,y) na sietke
clevels=linspace(0,10,15)   # vrstevnicove hladiny
contourf(X,Y,Fxy,clevels)   # vyplneny vrstev. graf
colorbar()                  # pridanie fareb. stupnice
```



Ako ľahko zistíte príkazom `Fxy.max()` maximálna hodnota $f(x,y)$ na zvolenej sieťke je asi 48.23. My sme znázornili 15 vrstevnicových hladín od 0 do 10, preto je časť obrázku biela. Jasne na ňom vidieť polohu koreňov, pretože okolo nich sú uzavreté, skoro eliptické vrstevnice (dva korene sú v dolnom a hornom pravom rohu a dva na reálnej osi).

3.6 Ukážky použitia Pylabu v numerike

3.6.1 Riešenie sústav nelineárnych rovníc

Keď sme prestali u vrstevnicových grafov, tak teraz tam začneme – úlohou o riešení sústavy dvoch nelineárnych rovníc o dvoch neznámych, pretože sa dá veľmi pekne vizualizovať. Majme teda sústavu rovníc

$$\begin{aligned} z_1 &= f_1(x,y) = 0, \\ z_2 &= f_2(x,y) = 0. \end{aligned}$$

Týmito rovnicami sú určené dve plochy z_1 a z_2. Riešiť zadanú sústavu znamená nájsť spoločné body (priesečníky) nulových vrstevníc týchto plôch a tie vieme znázorniť pomocou vrstevnicového grafu.

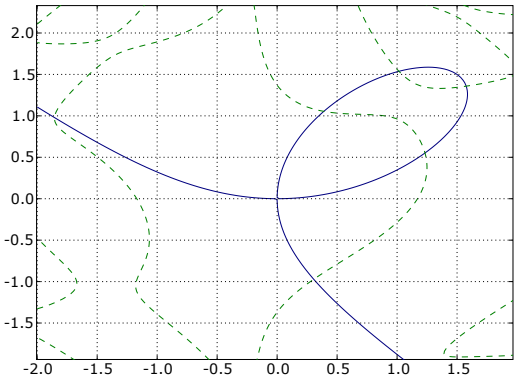
Príklad 3.15 *Nájdime tie riešenia sústavy rovníc*

$$\begin{aligned} \sin(xy^2) - \cos(x^2 - y) + 0.2 &= 0, \\ x^3 + y^3 - 3xy &= 0, \end{aligned}$$

ktoré ležia v prvom kvadrante (t. j. majú obidve súradnice kladné).

Všimnite si, že vrstevnicový graf podobnej funkcie sme robili v príklade 3.13. Budeme postupovať ako tam, len do toho istého grafu nakreslíme nulovú vrstevnicu aj pre druhú funkciu. Hladiny vrstevníc sa vždy musia zadávať ako zoznam (aj keď len jednoprvkový), inak by to Pylab chápal ako počet vrstevníc.

```
xx=linspace(-2,2,120);      yy=linspace(-2,2,120)
X,Y=meshgrid(xx,yy)
Z1=sin(X*Y*Y)-cos(X*X-Y)+0.2; Z2=X**3+Y**3-3*X*Y
contour(X,Y,Z1,[0.0]);      contour(X,Y,Z2,[0.0],colors='g')
```



Ak si cez menu obrázkového okna urobíte vhodný výrez, môže váš obrázok vyzerať podobne ako vidíte tu (čiarkovanú čiaru – pre vrstevnicu prvej funkcie nebolo až tak ľahko urobiť, ale už by ste to mali dokázať aj vy).

Z obrázku jednoznačne vidieť, že riešenia sústavy v prvom kvadrante sú štyri. Ak si zväčšíte obrázkové okno do ich tesnej blízkosti, môžete odčítať aj ich približné súradnice:

$$(0.38, 1.06),\ (1.23, 0.55),\ (1.02, 1.54),\ (1.58, 1.36).$$

Na riešenie nelineárnych rovníc a ich sústav máme v Pylabe (takisto ako v MATLAB-e) funkciu `fsolve`, je schovaná v submodule `scipy.optimize`. Prvým argumentom tejto funkcie je funkcia, ktorej keď zadáme (vektorový) argument – v našom prípade (x, y), vráti nám vektor hodnôt $(f_1(x, y), f_2(x, y))$. Napíšme ju v editore a uložme do súboru `nltrig.py`:

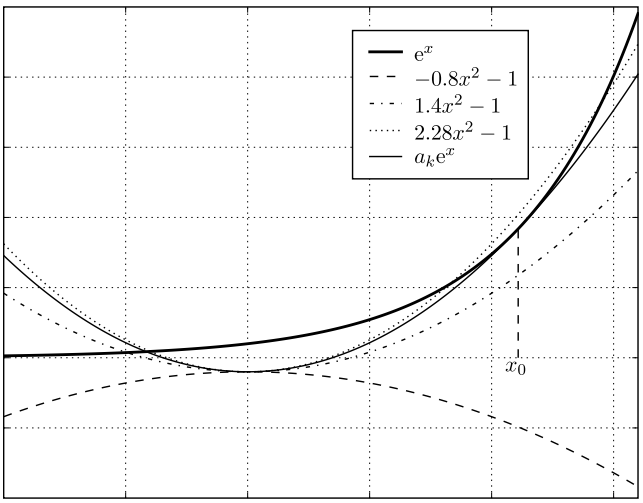
```
from numpy import sin, cos

def nlfct(xy):
    x,y=xy   # xy je dvojzlozkovy vektor, roztrhneme ho
    f1=sin(x*y*y)-cos(x*x-y)+0.2
    f2=x**3+y**3-3*x*y
    return (f1,f2)
```

Druhým argumentom pre `fsolve` je počiatočné priblíženie riešenia, teda napr. tie hodnoty, čo sme odčítali z vrstevnicového grafu. Takže načítame našu funkciu do interaktívneho prostredia cez `run nltrig.py` a skúsime

```
from scipy.optimize import fsolve
r=fsolve(nlfct,(1.23, 0.55)) # bude r=(1.2328735, 0.5521787)
```

Ak si dáte príkaz `nlfct(r)`, ktorý vypočíta hodnoty funkcií $f_1(r)$, $f_2(r)$ v riešení, uvidíte, že sú veľmi blízke nule, ako by to aj malo byť. Nám vyšli okolo `(-4.552e-15, -3.997e-14)`. Ostávajúce tri riešenia si urobte samostatne. Zaujímavé, že pre tretie a štvrté riešenie vyšli hodnoty $f_1(r)$, $f_2(r)$ horšie, rádovo okolo 10^{-12}. Dá sa to tiež názorne vysvetliť, ak si znázorníte pre obidve funkcie susedné „blízke“ vrstevnice, napr. pre hladiny -0.1, 0.1. V okolí tých dvoch posledných riešení sú vrstevnice hustejšie, preto sa funkčné hodnoty rýchlejšie menia, ak sa mierne vzdiaľujeme od riešenia. Ale neverte nám, skúste si to sami.

Príklad 3.16 *Zistite, koľko koreňov má rovnica* $\mathrm{e}^x = ax^2 - 1$ *v závislosti od reálneho parametra* a.



Z obrázka je vidieť, že pre $a \leqq 0$ neexistuje žiadny reálny koreň, pretože vtedy $ax^2 - 1 \leqq -1$, kým $\mathrm{e}^x > 0$ pre všetky reálne x.

Pre $a > 0$, ak a je dostatočne malé, rovnica má jeden záporný koreň. Ak však a bude dostatočne veľké, budeme mať tri korene (jeden záporný a dva kladné). Najdôležitejší je pre nás „medzný prípad“, kedy pre nejaké $a = a_k$ má parabola a exponenciála spoločnú dotyčnicu. Označme x-ovú súradnicu dotykového bodu ako x_0. Potom na určenie a_k, x_0 máme sústavu nelineárnych rovníc, ktorá vyplýva z rovnosti funkčných hodnôt a tiež derivácií v bode x_0

$$\begin{aligned} a_k x_0^2 - 1 &= \mathrm{e}^{x_0} \\ 2a_k x_0 &= \mathrm{e}^{x_0}. \end{aligned}$$

Z druhej rovnice ľahko vyjadríme $a_k = \mathrm{e}^{x_0}/(2x_0)$ a dosadením do prvej dostaneme pre x_0 rovnicu (index nepíšeme, je zbytočný)

$$\mathrm{e}^x(x-2) - 2 = 0.$$

Tú ľahko vyriešime v Pylabe a dopočítame príslušný parameter a_k:

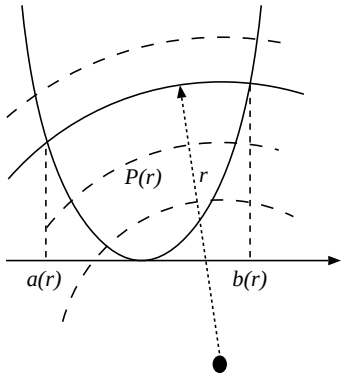
```
f=lambda x: exp(x)*(x-2)-2   # nelin. rovnica pre x0
x0=fsolve(f,1.5)             # x0=2.217715105757
ak=exp(x0)/(2*x0)            # ak=2.071122003228
```

Skrátenú definíciu funcie `f` pomocou tzv. lambda notácie vysvetlíme podrobnejšie v nasledujúcom príklade. Takže, ako zhrnutie výsledkov, môžeme povedať, že pre $0 < a < a_k$ má pôvodná rovnica jeden reálny koreň, pre $a = a_k \doteq 2.071122$ má dva korene a pre $a > a_k$ tri korene.

3.6.2 Numerické integrovanie

Príklad 3.17 (Príklad o koze). *V bode* $(1, -3)$ *(vzdialenosti budeme merať v metroch) je priviazaná koza. V parabole* $y = x^2$ *je vrbina, ktoré tá koza používa ako potravu. Mimo tej paraboly je púšť, kde sa nedá nájsť žiadna obživa. Ako dlhý má byť špagát, na ktorom je koza priviazaná, aby obhrýzla presne svoju dennú dávku* $10\ m^2$ *vŕb?*

Situáciu ukazuje obrázok:



Z rovnice hraničnej kružnice (kam až dosiahne povraz) dostávame

$$(x-1)^2 + (y+3)^2 = r^2 \;\Rightarrow y_k = \sqrt{r^2 - (x-1)^2} - 3.$$

Keď sa dohodneme na označení podľa obrázku, teda $P(r)$ bude plocha spásateľnej oblasti, potom potrebujeme určiť takú hodnotu r, aby

$$P(r) = \int_{a(r)}^{b(r)} \left(y_k - x^2\right) \mathrm{d}x = \int_{a(r)}^{b(r)} \left(\sqrt{r^2 - (x-1)^2} - 3 - x^2\right) \mathrm{d}x = 10.$$

Priesečníky $a(r), b(r)$ paraboly a kružnice sú reálne korene rovnice

$$x^2 = \sqrt{r^2 - (x-1)^2} - 3.$$

Umocnením ju ľahko upravíme na polynomickú

$$x^4 + 7x^2 - 2x + 10 - r^2 = 0 \tag{1}$$

Dôležité je uvedomiť si, že pre ľubovoľné rozumné r vieme vypočítať hodnoty funkcie $P(r)$, ktorej koreň hľadáme. Najskôr určíme reálne korene $a(r)$, $b(r)$ polynomickej rovnice (pomocou funkcie `roots`) a potom môžeme vypočítať aj príslušný určitý integrál vo vyjadrení funkcie $P(r)$. Vieme ho síce spočítať aj vzorcom, ale urobíme numerický výpočet pomocou funkcie `quad` zo submodulu `scipy.integrate`. Na riešenie rovnice $P(r) - 10 = 0$ použijeme funkciu `fsolve`, ktorú poznáme z predchádzajúceho príkladu. Výpočet v Pylabe realizujeme pomocou súboru `koza.py`:

```
from numpy import roots, sqrt
from scipy.optimize import fsolve
from scipy.integrate import quad

Pr_forint=lambda x,r: sqrt(r**2-(x-1)**2)-3-x**2
TOL=1.0e-8

def Pr10(r):
    c=[1, 0, 7, -2, 10-r*r]
    rt=roots(c)
    rr=rt[abs(rt.imag)<TOL].real
    if not len(rr):
        raise ValueError, "Rope too short, goat passed!"
    rr.sort()
    Ip,err=quad(Pr_forint,rr[0],rr[1],args=r)
    return Ip-10

print "Optimal Rope length: ", fsolve(Pr10,8)
```

Pomerne jednoduché, však? V MATLAB-e by to bolo trochu komplikovanejšie. Vysvetlíme si, čo sa v tom súbore robí. Riadok

```
Pr_forint=lambda x,r: sqrt(r**2-(x-1)**2)-3-x**2
```

definuje funkciu, ktorá sa vyskytuje v integrále. Áno, `Pr_forint` je funkcia, definovaná pomocou tzv. lambda notácie. Hodí sa to, keď máte jednoduchý výraz, ktorý funkcia vracia, napr. môžete si nadefinovať

```
pythag=lambda x,y: sqrt(x**2+y**2);   sqr=lambda(x): x*x
```

a potom tie funkcie normálne voláte – skúste si `pythag(3,4); sqr(-2); sqr(10)` a podobne. Samozrejme, zložitejšie funkcie je lepšie definovať pomocou kľúčového slova **def** ako sme to robili doteraz. Takto je definovaná aj funkcia `Pr10`, ktorá pre dané r určí hodnotu $P(r) - 10$.

V premennej `rt` sú korene polynómu (1). Tie sú buď všetky komplexné, ak je špagát príliš krátky (napr. $r = 2$; vtedy generujeme chybu s hláškou o skapíňajúcej koze) alebo dva z nich sú reálne a to sú hranice $a(r)$, $b(r)$ pre integrál. Zaujímavý je príkaz

```
rr=rt[abs(rt.imag)<1.0e-8].real
```

pomocou ktorého jedným šmahom vyberieme len tie korene, ktoré majú zanedbateľnú veľkosť imaginárnej časti (prakticky povedané, sú to reálne korene) a potom ich zreálnime – zoberieme ich reálne časti, ktoré uložíme do číselného poľa `rr`. To potom usporiadame a máme hranice integrálu. Teraz už len použijeme funkciu `fsolve` na určenie koreňa nelineárnej rovnice $P(r) - 10 = 0$. Ako počiatočné priblíženie riešenia sme zobrali $r = 8$, teda takú dĺžku špagátu, že pastva bude určite neprázdna. Pre kontrolu, nám vyšla dĺžka špagátu 6.99133891149 m.

3.6.3 Minimalizácia funkcie dvoch premenných

Príklad 3.18 (Príklad o strelcoch). *Prenasleduje nás banda štyroch strelcov, pred ktorými sa skrývame v (oplotenom a vypuklom) štvoruholníku. Každý strelec sa pohybuje po jednej zo strán štvoruholníka a všetci majú rovnaký dostrel. Predpokladáme, že sa pohybujú (pre nich) optimálne, t. j. zaujmú na svojej strane vždy takú polohu, aby nám boli najbližšie. Akú si máme vybrať polohu, aby sme mali čo najväčšiu šancu na prežitie?*

Aká poloha bude pre nás najvýhodnejšia? Určite to bude tá, kde najbližší zo strelcov (ten nás najviac ohrozuje) bude čo možno najďalej. Potrebujeme to sformulovať matematicky.

Nech sme v nejakom bode x, y štvoruholníka $\mathcal{S}$ s vrcholmi A, B, C, D a nech vzdialenosti, deliace nás od jednotlivých strelcov sú $d_1(x,y)$, $d_2(x,y)$, $d_3(x,y)$, $d_4(x,y)$. Hľadáme maximum funkcie

$$d(x,y) = \min_{(x,y)\in\mathcal{S}} \{d_1(x,y), d_2(x,y), d_3(x,y), d_4(x,y)\}. \tag{2}$$

Ak bude veľkosť $d(x,y)$ väčsia, ako je dostrel (označme ho s) každého zo strelcov, sme zachránení. Inak nás skôr-neskôr dostanú aj napriek našej (relatívne) výhodnej polohe.

Budeme potrebovať vzdialenosti ľubovoľného bodu $(x_0, y_0) \in \mathcal{S}$ od strán štvoruholníka, napr. od strany AB. Rovnica priamky, danej dvomi bodmi $A = (x_A, y_A)$, $B = (x_B, y_B)$ je

$$-kx + y + (kx_A - y_A) = 0, \quad \text{kde } k = \frac{y_B - y_A}{x_B - x_A}$$

a podľa známeho vzorca je tá vzdialenosť rovná

$$d_1 = \frac{|-kx_0 + y_0 + (kx_A - y_A)|}{\sqrt{1+k^2}}.$$

Podobne sa určia vzdialenosti d_2, d_3, d_4 od ostávajúcich troch strán.

Výpočty bude za nás robiť Pylab, v súbore `strelci.py`, ktorý si postupne budete vytvárať. Hlavička funkcie na výpočet vzdialenosti bodu P od priamky danej dvoma bodmi A, B nech je

```
def vzdial(A,B,P):
```

a na výpočet vzdialenosti $d(x,y)$ zo vzorca (2), ak na vstupe zadáme vrcholy štvoruholníka a nejakého jeho vnútorného bodu P napíšte funkciu s hlavičkou

```
def optimald(P,A,B,C,D):
```

Veríme, že po predchádzajúcich skúsenostiach vám napísanie týchto dvoch funkcií nebude robiť ťažkosti. Pretože v Pylabe máme prostriedky len na minimalizáciu, musí funkcia `optimald` vrátiť $-d(x,y)$ – minimalizácia záporne vzatej funkcie je ekvivalentná s maximalizáciou pôvodnej funkcie (tak to potrebujeme my).

V module `scipy.optimize` máme veľa funkcií na hľadanie minima funkcie jednej alebo viac premenných, ale môžeme použiť len tie, ktoré nepotrebujú výpočet gradientu (parciálnych derivácií), lebo to je pre našu funkciu zložité. Hodia sa nám napr. funkcie `fmin`, `fmin_powell`. Pre konkrétny štvoruholník s vrcholmi:

$$A = (-3,-2);\ B = (4.5,-4.5);\ C = (5,4.5);\ D = (-2,2)$$

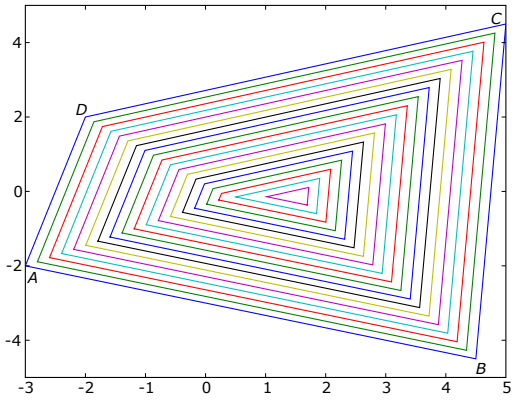
by sme postupovali v interaktívnom prostredí takto (predpokladáme, že ste príkazom `run strelci` načítali vami vytvorené funkcie `vzdial`, `optimald`:

```
from scipy.optimize import fmin, fmin_powell

A,B,C,D=(-3,-2),(4.5,-4.5),(5,4.5),(-2,2)
fmin(optimald,(1.0,0.0),args=(A,B,C,D),xtol=1.0e-8)
```

Výstup z `fmin`, ktorý sa vypíše na obrazovku, je optimálna poloha, t. j. bod $P = (1.535739, -0.1370124)$, pričom najmenšia vzdialenosť od strán štvoruholníka je 3.201712.

Cez nápovedu `fmin?` by ste zistili, že prvý argument je funkcia, ktorú máme minimalizovať, druhý je počiatočné priblíženie bodu, kde sa nadobúda minimum. Zaujímavý je pomenovaný argument `args` – hovorí, aké ďalšie argumenty (okrem prvého, čo sú premenné, v našom prípade súradnice bodu P v štvoruholníku) treba zadať do funkcie, ktorú optimalizujeme. Podobným štýlom sa dajú zadávať ďalšie parametre (argumenty) tiež pri numerickom integrovaní a pri riešení diferenciálnych rovníc, čo má veľký praktický význam.

Naznačíme si iný, geometricky názornejší spôsob riešenia príkladu o strelcoch. Uvažujme o vrstevniciach grafu funkcie $d(x,y)$ na štvoruholníku $\mathcal{S}$. Budú to menšie štvoruholníky, príp. degenerované na trojuholník alebo úsečku, ako to vidieť na obrázku. Graf funkcie $d(x,y)$ je plocha, pripomínajúca stoh slamy, vo všeobecnosti trochu asymetrický. Najvyšší bod toho stohu udáva nami hľadanú optimálnu polohu.



Vedeli by ste (samozrejme v Pylabe) tiež nakresliť podobný obrázok? Všimnite si, že vnútorné štvoruholníky, tvoriace vrstevnice, majú vrcholy na osiach uhlov pôvodného štvoruholníka (je to logické, lebo tam sú vzdialenosti od dvoch zo strán rovnaké).

Intuitívne sa zdá, že najlepšia bude taká poloha, že budeme mať od všetkých strán rovnakú vzdialenosť (a to čo najväčšiu). To by sme sa museli postaviť do stredu kružnice, opísanej štvoruholníku. Nie každý štvoruholník však takú kružnicu má (predstavme si napr. že je dlhý a tenký). Bude to asi

tak, že hľadaný bod bude stred kružnice, dotýkajúcej sa zvnútra niektorých troch strán štvoruholníka. Alebo, inak povedané, priesečník osí dvoch vhodných vnútorných uhlov štvoruholníka (v našom konkrétnom štvoruholníku je to priesečník osí uhlov pri vrcholoch B a C). Z obrázka je to skoro jasné, skúste si to zdôvodniť presnejšie a realizujte tento postup programovo v Pylabe.

3.7 Interaktívna práca s grafickým oknom

3.7.1 Ukážka interaktívneho zadávania dátových bodov

Občas sa hodí, aby sme napr. mohli zadať dátové body pomocou kurzoru myši interaktívne v grafickom okne. Teda, potrebujeme, aby Pylab reagoval podľa našich pokynov na rôzne udalosti (stlačenie a uvoľnenie tlačítka myši, písanie na klávesnici, zmena veľkosti grafického okna, atď.). V Pylabe máme tieto udalosti pomenované takto

`resize_event`	zmena veľkosti grafického okna,
`draw_event`	prekreslenie obrázku,
`key_press_event`	stlačenie klávesu,
`key_release_event`	uvoľnenie klávesu,
`button_press_event`	stlačenie tlačítka myši,
`button_release_event`	uvoľnenie tlačítka myši.

Pylab používa metódu registrácie udalostí, teda povieme mu napr., že pri stlačení tlačítka myši (registrovaná udalosť, *event*) v grafickom okne má volať nami definovanú funkciu (*callback function*). Ukážeme si to na jednoduchom príklade.

Príklad 3.19 *Urobme grafické okno, kde rozsah na osi x bude* $0 \leqq x \leqq 10$ *a nech* $-6 \leqq y \leqq 6$. *Nechajme užívateľa zadávať dátové body pravým tlačidlom myši v tomto okne, ale vždy len tak, aby nasledujúci bod mal x-ovú súradnicu väčšiu, ako predchádzajúci. Zadávanie bodov sa ukončí pravým tlačidlom myši a body sa uložia v textovom formáte do súboru s názvom* `Body.txt`.

Môžeme postupovať napr. takto (nižšieuvedené príkazy nech sú uložené v súbore `bodyxy.py`):

```
from pylab import *

# Globalne premenne
V=[]        # body -- zoznam dvojic (x,y)
vc=None     # registracia funkcie pre reakciu na mys
xmax=-1     # max. hodnota x-ovej suradnice

def Points_input():
    global vc, xmax, V
    xmax=-1; V=[]
    axis([0,1,0,1])
    ax=axes()
    ax._autoscaleon=False
    vc=connect('button_press_event',mk_point)       <-----
    show()

def mk_point(event):                                <-----
    global vc, V, xmax
    if event.button==1:
        x,y=event.xdata,event.ydata
        if x>xmax:
            xmax=x
```

```
            plot([x],[y],'o')
            V.append((x,y))
    elif event.button==3:
        disconnect(vc)                                  <-----
        save('Body.txt',V,fmt='%1.3f')
```

Interaktívne zadávanie bodov zaisťuje funkcia `Points_input()`. V nej je pomocou príkazov

```
axis([0,1,0,1])
ax=axes()
ax._autoscaleon=False
```

zaistené, že mierka na osiach bude stále taká istá a že sa nebude automaticky prispôsobovať súradniciam zadávaných bodov. Pre udalosti typu `'button_press_event'` je pomocou príkazu

```
vc=connect('button_press_event',mk_point)
```

registrovaná funkcia `mk_point`, ktorá ich bude obsluhovať. Ďalej je táto funkcia implementovaná.

Každá funkcia obsluhujúca udalosti musí mať len jediný parameter a to je (v submodule `matplotlib.backend_bases.Event`) objekt Pylabu `Event`. V obslužnej funkcii môžeme používať atribúty objektu `Event`, napr. tieto

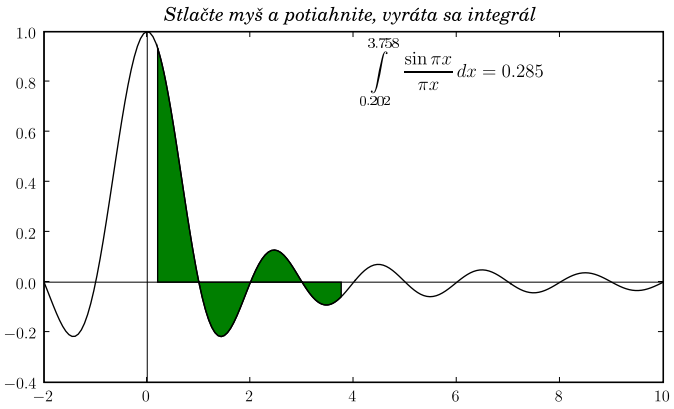
`x`	x-súradnica v pixeloch od ľavej strany obrázku
`y`	y-súradnica v pixeloch od dolnej strany obrázku
`button`	stlačené tlačidlo myši (1, 2, 3 alebo None)
`inaxes`	objekt súrad. osí, ak je myš v tých súradniciach, alebo None
`xdata`	x-súrad. v užívateľských dátových súradniciach alebo None
`ydata`	y-súrad. v užívateľských dátových súradniciach alebo None
`key`	stlačený kláves, napr. 'a', 'b', '1'

Naša funkcia `mk_point` používa atribút `button` na zistenie, či je stlačené tlačidlo myši a ak je, tak ktoré je to. Myslíme si, že podľa tohto príkladu bude čitateľ schopný vytvárať si jednoduchú interaktívnu grafiku v Pylabe.

3.7.2 Ukážka použitia grafického užívateľského rozhrania

MATLAB dáva možnosť vytvárať jednoduché užívateľské rozhrania. V Pylabe, vďaka modulu Matplotlib, máme k dispozícii tiež niekoľko jednoduchých prvkov na tvorbu užívateľských rozhraní, napr. grafické tlačidlá, posuvníky, nástroje na výber obdĺžnikových oblastí. Uvedieme malý príklad, ktorý naznačuje široké možnosti použitia Pylabu pri interaktívnej výučbe matematickej analýzy.

Príklad 3.20 *Vytvorme grafické okno, v ktorom bude mať užívateľ možnosť zadávania hraníc a, b integrálu $\int_a^b \frac{\sin \pi x}{\pi x}\,\mathrm{d}x$ pomocou myši. Po zadaní hraníc sa v grafickom okne zobrazí výsledok, napr. ako na nasledujúcom obrázku.*



Na výber hraníc integrálu použijeme grafický prvok `SpanSelector`, ktorým sa dajú v grafickom okne vyberať zvislé alebo vodorovné pásy. Nasledujúce príkazy budeme zapisovať do súboru `definteg.py`.

Najskôr ošetríme potrebné importy (grafika, numerické integrovanie). Potom pomocou lambda notácie (je vysvetlená na str. 47) zavedieme funkciu `fcn`, ktorú budeme integrovať.

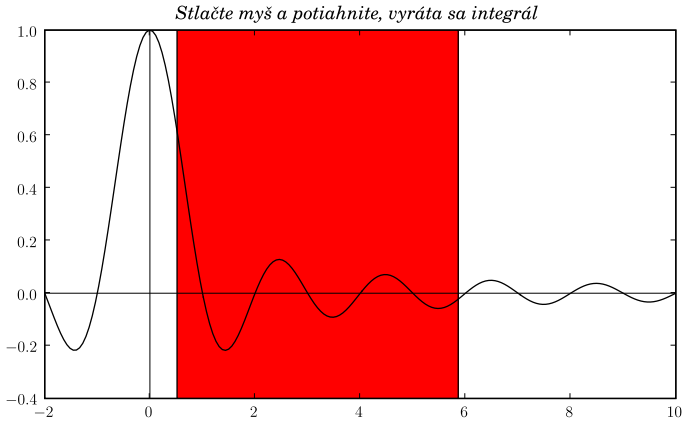
```
from pylab import *
from matplotlib.widgets import SpanSelector
from scipy.integrate import quad

fcn=lambda x: sin(pi*x)/(pi*x)
```

Ďalej nastavíme, aby sa na výpisy matematických vzorcov používal TEX, nakreslíme integrovanú funkciu a súradnicové osi pre $x \in \langle -2, 10 \rangle$, urobíme vysvetľujúci text (titulok) obrázka (diakritické znamienka sa zadávajú ako v TEX-u).

```
rc('text', usetex=True)
fig=figure()                  # nove graficke okno
ax =gca()                     # aktualne osi do premennej ax

x=linspace(-2,10,200)
y=sinc(x); xlim(-2,10)        # hranice pre x
l=plot(x,y,'k')               # graf integrovanej funkcie
axhline(0,color='k',lw=0.5)   # cierne tenke ciary
axvline(0,color='k',lw=0.5)   # pre surad. osi

t=title(r'\it Stla\v cte my\v s a potiahnite, vyr\'ata sa integr\'al')
```

Nasleduje definícia obslužnej funkcie `onselect` pre `SpanSelector`. Jej parametre sú hranice `vmin`, `vmax`, vybrané pomocou stlačenia a pohybu ľavého tlačidla myši. Po výbere hraníc vyzerá grafické

okno asi takto:



Tieto hranice sa využijú ako vstup do funkcie quad na numerickú integráciu. Oblasť v zvolených hraniciach, ohraničenú osou x a integrovanou funkciou, znázorníme pomocou farebného vyplnenia zelenou farbou a vypíšeme informácie o hodnote integrálu tak, ako je to na obrázku na predchádzajúcej strane. Takže dokončenie súboru `definteg.py` tvoria tieto príkazy:

```
Drawn=None       # ci uz bola nakreslena vyplnena plocha

def onselect(vmin, vmax):
    global Drawn
    if Drawn:      # vycisti vyplnenu plochu aj vypisy
                   # o hraniciach a hodnote integralu
        ax.patches,ax.texts=[],[]
    else:
        Drawn=True

    xx=linspace(vmin,vmax, int(200*(vmax-vmin)/12.0))
    yy=sinc(xx)
    xx=array([vmin]+list(xx)+[vmax,vmin]) # uzavrety polygon
    yy=array([0.0]+list(yy)+[0.0,0.0])    # pre vyplnenie
    fill(xx,yy,facecolor='g')
    text (6,0.7,
          r"$\displaystyle{\int\limits_{%2.3f}^{%2.3f}\
          {\frac{\sin \uppi x}{\uppi x}}\,dx=%2.3f}$" \
          %(vmin,vmax,quad(sinc,vmin,vmax)[0]),
          horizontalalignment='center',fontsize=14)
    draw()                    # aby sa aktualizoval obrazok

span = SpanSelector(ax, onselect, 'horizontal')
show()                        # prve nakreslenie obrazka
```

Okrem definície obslužnej funkcie, na konci súboru je do aktuálneho obrázka pridaný grafický prvok `SpanSelector`, ktorému je priradená táto obslužná funkcia `onselect`.

Iné príklady použitia grafických prvkov nájdete na `http://matplotlib.sourceforge.net` (domovská stránka Matplotlibu), je tam v hlavnom menu položka *Examples (zip)*, odtiaľ si ich môžete stiahnuť. Myslíme si, že prezeranie zdrojových textov týchto príkladov a ich modifikácia je najlepší spôsob, ako sa zoznámiť s Pylabom a jeho grafikou.

Záver

V tejto stručnej učebnici sme sa dotkli skutočne len niektorých oblastí, kde sa dá Pylab použiť. Takisto sme mohli spomenúť len málo z funkcií a možností, ktoré tento systém ponúka. Základom výpočtovej časti Pylabu je modul Scipy (OLIFANT, 2004) a jeho submoduly. Z tých, o ktorých sme nehovorili, spomenieme teraz aspoň `stats` - modul pre štatistiku (obsahuje veľa rozdelení pravdepodobnosti a štatistických testov[26]).

Pylab má obrovský potenciál na použitie nielen vo výučbe matematických a informatických predmetov, ale aj na riešenie reálnych, rozsiahlych úloh inžinierskej praxe. Je to vďaka univerzálne použiteľnému programovaciemu jazyku Python. Keď budete dlhšie pracovať s Pylabom, určite si aj vy vytvoríte svoje vlastné minimoduly v Pythone, alebo budete používať aj ďalšie existujúce moduly, ktoré vám veľmi uľahčia prácu v špecializovaných aplikačných oblastiach.

Python nepodporuje symbolické manipulácie (teda napr. počítanie limít, derivácií, integrálov v tvare vzorcov). Existuje však systém `SAGE`, `http://sage.scipy.org/sage/` ktorý používa Python ako svoj hlavný programovací prostriedok a je určený na podporu výskumu a výučby v algebre, geometrii, teórii čísel, kryptografii, atď. Bližšie sa o tomto systéme môžete dozvedieť z dokumentácie (JOYNER, 2006), ktorá je k dispozícii na vyššieuvedenej webovej stránke.

Na stránke `http://www.vrplumber.com/py3d.py` nájde čitateľ dobrý prehľad o aplikáciach a knižniciach, týkajúcich sa trojdimenzionálnej grafiky v Pythone. Autor zo svojej skúsenosti môže odporúčať napr. program `Mayavi`, `http://mayavi.sourceforge.net/`, čo je prezerač, umožňujúci interaktívnu manipuláciu s priestorovými objektami.

Dúfame, že aj táto učebnica podnieti ďalší záujem čitateľa o programovací jazyk Python a jeho početné rozširujúce moduly, ktoré môžu v mnohých prípadoch slúžiť ako rovnocenná náhrada komerčného softvéru, ba v mnohých ohľadoch (dostupnosť zdrojového kódu, široká a priateľská užívateľská komunita, otvorená aj pre začínajúcich programátorov, pravidelné konferencie tiež v Európe) ho aj predstihujú. Na serveri `www.py.cz` pre slovenských a českých užívateľov Pythonu nájdete veľa dokumentácie a materiálu, ktorý vám môže pomôcť v ďalšom raste a zdokonaľovaní sa v tejto oblasti.

[26] Existuje tiež modul `RPy`, ktorý umožňuje používať v Pythone, teda aj v Pylabe objekty a funkcie z programovacieho jazyka `R`, `http://www.r-project.org/`, čo je veľmi kvalitný Open Source systém na štatistické výpočty (VENABLES ET AL., 2006).

Príloha – Zoznam najpoužívanejších funkcií

Tabuľka 2: Najpoužívanejšie funkcie pre grafiku

`axis`	nastaví alebo vráti aktuálne hranice na súrad. osiach
`cla`	vyčistí aktuálny súradnicový systém
`clf`	vyčistí celé obrázkové okno, bude bez súradníc
`close`	zatvorí (aktuálne) obrázkové okno
`colorbar`	pridá farebnú stupnicu do aktuálneho obrázku
`contour`	urobí vrstevnicový graf
`contourf`	vrstevnicový graf, ale farebne vyplnený medzi vrstevnicami
`draw`	prinúti aktuálny obrázok, aby sa prekreslil
`figure`	vytvorí alebo zmení aktívny obrázok
`fill`	kreslenie (vyplnených) mnohouholníkov
`gca`	vráti aktuálny súrad. systém (na modifikáciu vlastností)
`gcf`	vráti aktuálny obrázok (na modifikáciu vlastností)
`grid`	prepína zobrazenie sieťky na grafe
`hold`	určuje, či sa grafické objekty pridávajú, alebo sa zakaždým mažú
`legend`	legenda pre aktuálne osi
`plot`	urobí normálny, čiarový graf – ASI NAJPOUŽÍVANEJŠÍ PRÍKAZ
`pcolor`	pre funkciu dvoch premenných, hodnoty znázornené farbami
`polar`	graf v polárnych súradniciach
`savefig`	uloženie aktuálneho obrázka (`.jpg`, `.png`, `.eps`)
`show`	ukázať obrázky (v neinteraktívnom režime)
`subplot`	urobí subplot (`poc_riadkov,poc_stlpcov,akt_sursys`)
`text`	pridá text na pozíciu (x, y) v aktuálnom súradnicovom systéme
`title`	pridá titulok k aktuálnemu súradnicovému systému
`xlabel`	nadpis pre x-ovú os
`ylabel`	nadpis pre y-ovú os

Tabuľka 3: Iné funkcie pre grafiku a manipuláciu s ňou

`axes`	vytvorí nový súradnicový systém
`axhline`	nakreslí vodorovnú čiaru cez celý obrázok
`axvline`	nakreslí zvislú čiaru cez celý obrázok
`axhspan`	nakreslí vodorovný (vyplnený) stĺpec cez celý obrázok
`axvspan`	nakreslí zvislý (vyplnený) stĺpec cez celý obrázok
`bar`	urobí stĺpcový graf
`barh`	vodorovný stĺpcový graf
`boxplot`	obdĺžnikový a fúzatý graf stĺpcov dátovej matice
`clabel`	označovanie vrstevníc vo vrstevnicovom grafe
`colormaps`	`colormaps?` vypíše názvy farebných paliet
`delaxes`	vymaže daný súrad. systém z aktuálneho obrázka
`figlegend`	globálna legenda pre obrázok, nie pre súr. systém
`figtext`	pridá text v obrázkových súradniciach [0,1,0,1]
`hist`	kreslí histogram
`ioff, ion`	vypína a zapína interaktívny mod (efektivita!)
`imread`	načíta obrázok do číselného poľa
`imshow`	nakreslí obrázok z číselného poľa (viď `imread`)
`loglog`	graf s obidvomi mierkami na osiach logaritmickými
`matshow`	nakreslí maticu (veľkosti prvkov sú farebne odlíšené)
`pie`	koláčový graf, obľúbená to potrava manažérov
`plot_date`	ako `plot`, ale popisy osí sú dátumy
`quiver`	obrázok smerového poľa (diferenciálnej rovnice) alebo vektorového poľa
`rgrids`	prispôsobenie radiálnej sieťky a značkovania pre polárne súradnice
`scatter`	nakreslí roztrúsené body
`semilogx`	logaritmická mierka na x-ovej osi
`semilogy`	logaritmická mierka na y-ovej osi
`spy, spy2`	zobrazuje riedke matice (tam, kde sú nenulové prvky)
`stem`	„rastlinkový“ graf – od osi x idú steblá ku bodom
`table`	pridá tabuľku do obrázka
`thetagrids`	prispôsobenie uhlovej sieťky a značkovania pre polárne súradnice
`xlim, ylim`	nastavuje respektíve vracia hranice na osiach
`xticks`	nastavuje respektíve vracia popisy a značkovanie osi x
`yticks`	nastavuje respektíve vracia popisy a značkovanie osi y

Použitá literatúra

BUŠA, J. 2006. *Octave, Rozšírený úvod*, 105 s.

VAN ROSSUM, G. 2006. *Python Documentation*, online dokumentácia na stránke `http://www.python.org/doc/`.

PÉREZ, F. 2006. *IPython. An enhanced Interactive Python*, online na stránke `http://ipython.scipy.org/doc/manual/`.

OLIPHANT, T. E. 2005. *Guide to NumPy*, 247 s.

OLIPHANT, T. E. 2004. *SciPy Tutorial*, 42 s.

KAUKIČ, M. 1998. *Numerická analýza I. Základné problémy a metódy*. Žilina, MC Energy s. r. o., 202 s.

HUNTER, J. 2006. *The Matplotlib User's Guide*, 79 s.

VENABLES, W. N. – SMITH, D. H. and the R Development Core Team. 2006. *An Introduction to R*, 99 s.

JOYNER D., STEIN W. 2006. *SAGE Tutorial*, 98 s.

4 OPTIMALIZÁCIA V TABUĽKOVOM PROCESORE GNUMERIC

Štefan PEŠKO
Katedra matematických metód, FRI
Žilinská univerzita v Žiline

4.1 Úvod

V tejto kapitole sa chceme podeliť o naše skúsenosti s pomerne málo využívanou možnosťou pohodlného riešenia niektorých optimalizačných úloh v tabuľkovom procesore *Gnumeric* bez potreby ich procedurálneho programovania. Tento prístup našiel uplatnenie vo výskume pri tvorbe základných logistických modelov, aj prekvapujúco dobrú odozvu u študentov pri precvičovaní poznatkov z predmetov *teória hramadnej obsluhy, teória hier a kvantitatívne metódy logistiky*.

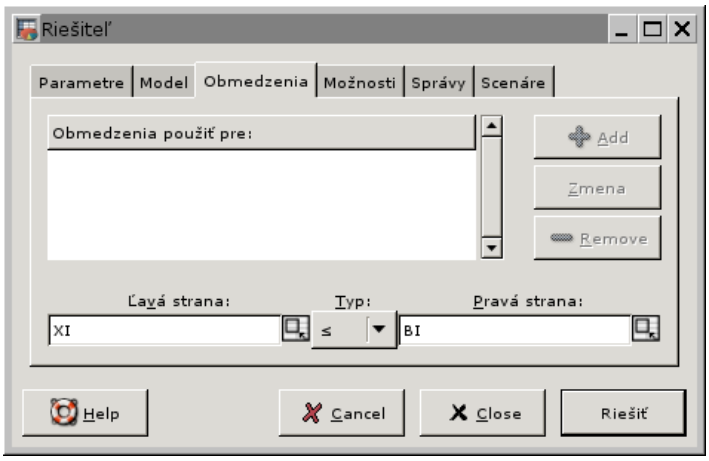
Pri riešení viacerých praktických optimalizačných úloh operačnej analýzy, teórie rozvrhov a teórie hier aplikovaných najmä v dopravnej logistike, sme zistili (PEŠKO, 2002), že i súčasný tabuľkový procesor Gnumeric (THE GNUMERIC MANUAL) pod OS GNU/Linux ponúka možnosť ich pohodlného riešenia. Z ponuky Gnumericu sa dokonca stačí obmedziť len na niektoré jeho základné funkcie, ktoré umožňujú jeho neprocedurálne programovanie a na *Riešiteľ (Solver)* na riešenie úloh lineárneho, resp. celočíselného lineárneho programovania.

4.2 Od Excelu ku Gnumericu

Aj keď máte skúsenosti s nejakým tabuľkovým procesorom, najčastejšie s Excelom od Microsoftu, nezaškodí zopakovať niekoľko pravidiel o práci s *bunkami* tabuľky. Oplatí sa hneď na začiatku sústrediť hlavne na *absolútne* ($\$A\$3$), *relatívne* ($A3$) *a zmiešané odkazy* ($\$A3$, $A\$3$) buniek. Ich zmysel sa dá rukolapne pochopiť pri kopírovaní oblastí.[27]

Ďalej sa nám osvedčilo prejsť na tabuľkové funkcie – hlavne maticové, ktoré výrazne sprehladňujú tvorbu modelov, budeme potrebovať tieto:

- $index(A; i[; j]^{28})$ vyberá z oblasti (matice) A obsah v riadku i a stĺpci j ako prvok A_{ij},
- $mmult(A; B)$ vracia maticu $A \cdot B$ rovnú maticovému súčinu matíc A a B,
- $sumproduct(A; B)$ vracia číslo $A \odot B$ rovné skalárnemu súčinu matíc $\sum_i \sum_j A_{ij} B_{ij}$.



Obrázok 1: Riešiteľ pre lineárne (celočíselné) programovanie

Kľúčovým, aj keď nie jediným nástrojom modelovania, je tu v ponuke voľba *Riešiteľ* (obrázok 1), ktorá optimalizuje za nás riešiac úlohy lineárneho (celočíselného) programovania v nasledujúcom tvare (M)LP:

$$\sum_{j=1}^{n} c_j x_j \to \min\,[\max], \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leqq [\,\geqq, =\,]\; b_i, \quad i = 1, \ldots, m, \tag{2}$$

$$x_j \geqq 0, \;\; [\text{celé}, \{0,1\}], \quad j = 1, \ldots, n. \tag{3}$$

pričom umožňuje zadávať v bunkách požadované lineárne funkcie. A tak cieľovú funkciu (1) zapíšeme v tvare formuly $= sumproduct(C; X)$, kde C, X sú príslušné vektory reprezentované súvislými oblasťami buniek. K pohodliu prispieva možnosť vektorového zápisu obmedzení (2) zhodného typu t. j. ak pre $i \in \{m_1, \ldots, m_2\}$ je $\sum_{j=1}^{n} a_{ij} x_j \leqq b_i$, potom stačí písať do obmedzení len nerovnosť $XI \leqq BI$, kde $XI = sumproduct(A_{m_1}; X) : sumproduct(A_{m_2}; X)$ je oblasť lineárnych funkcií a $BI = b_{m_1} : b_{m_2}$ je oblasť konštánt.

[27] Názvoslovie tu nie je ustálené, pre *oblasť* sa používajú i názvy *pole, tabuľka*.

[28] Nepovinné alebo alternatívne parametre budeme značiť v hranatých zátvorkách [♣].

Úlohu lineárneho programovania môžeme prirodzene zapísať aj v maticom tvare

$$\min\{cx : Ax = b,\ x \geqq 0\},$$

čo vedie k alternatívnej implementácii úlohy s použitím nielen skalárneho, ale aj maticového súčinu matíc. Pred samotným výberom implementácie sa oplatí diskutovať o výhodách a nevýhodách, a tak im poskytnúť možnosť voľby s následným porovnávaním alternatívnych prístupov.

Skôr však než tak urobíme, ukažeme si na jednoduchej hre – LIFE rekalkulačnú vlastnostnosť buniek tabuľky, ktorú budeme ďalej využivať.

4.3 Minimalizácia počtu prestupov medzi linkami

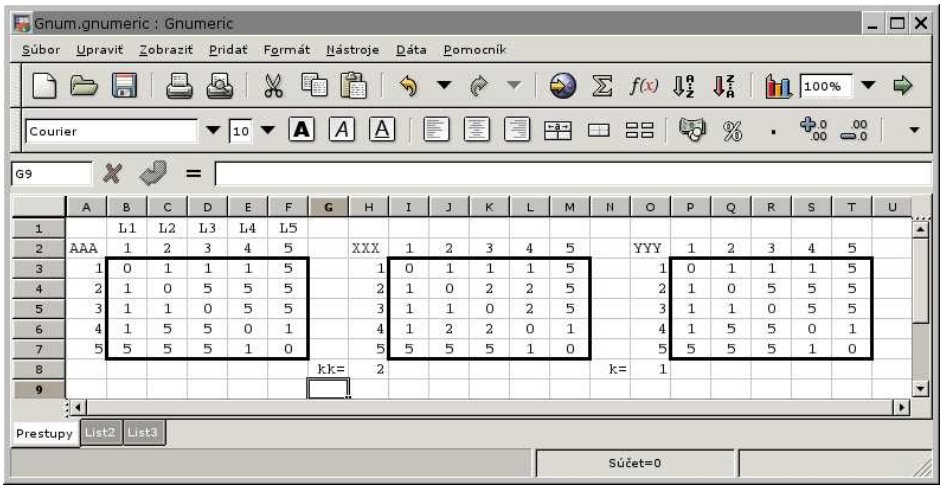
Najskôr uvedieme model, ktorého riešenie nevyžaduje použitie *Riešiteľa*, ale vystačí s citlivo „naprogramovanými“ odkazmi buniek. Pri vyhodnocovaní alternatívneho návrhu vedenia liniek MHD v Nitre vznikla potreba riešiť nasledujúcu optimalizačnú úlohu:

Nech je daná množina n liniek $\mathcal{L} = \{L_i : i \in N\}, N = \{1, 2, \ldots, n\}$ *a matica* $\mathrm{A} = (a_{ij})$*, kde prvok* $a_{ij} = 1$*, ak je možný prestup z linky* $L_i \in \mathcal{L}$ *na linku* $L_j \in \mathcal{L}$ *(t. j. linky majú aspoň jednu spoločnú zastávku) a* $a_{ij} = n$ *v opačnom prípade. Hľadá sa matica* $\mathrm{D} = (d_{ij})$*, kde prvok* d_{ij} *udáva minimálny počet prestupov z linky* L_i *na linku* L_j*.*

Takto formulovanú úlohu možno riešiť napr. známym Floydovým algoritmom na hľadanie matice minimálnych vzdialeností, ktorý by bolo možné implementovať nasledujúcou procedúrou $\mathrm{D} = FLOYD(\mathrm{A})$:

```
procedure FLOYD(A)
  D = A
  for k ∈ N do
    for i ∈ N do
      for j ∈ N do
        if d_ij > d_ik + d_kj then
          d_ij = d_ik + d_kj
return D
```

Ak však nepoznáme príslušný procedurálny jazyk tabuľkového procesora a nechceme sa ho učiť, ponúka sa nám možnosť využiť odkaz bunky na jednu bunku alebo na funkciu viacerých buniek oblastí (napr. $kk = if(k < n, k+1, n)$ realizuje počítadlo vonkajšieho cyklu indexu k algoritmu).



Obrázok 2: Floydov algoritmus pomocou odkazov po prvej iterácii

Na obrázku 2 máme v Gnumericu realizáciu Floydovho algoritmu pomocou odkazov pre úlohu s piatimi linkami. Najskôr sú v liste zošitu „Prestupy“ pomenované oblasti $XXX = Prestupy!\$I\$3 : \$M\$7$ a $YYY = Prestupy!\$O\$3 : \$T\$7$, pričom je na začiatku výpočtu oblasť $YYY = AAA$. Ak položíme bunku $I3$ rovnú

$$I3 = min(P3; index(YYY; \$H3; \$O\$8) + index(YYY; \$O\$; P\$2))$$

a rozkopírujeme do celej oblasti XXX, realizujeme k-ty iteračný krok procedúry. Vyššie uvedená formula je vlastne jadrom nášho neprocedurálneho programu. Opakovaným kopírovaním hodnôt oblasti YYY a kk do oblasti XXX a k, dostávame riešenie $XXX = YYY =$ D. Počítadlo v bunke $H8 = = if(O8 < A7; O8 + 1; A7)$ po každom kopírovaní zvýši hodnotu o $+1$, kým nedosiahne hodnotu 5.

Ak definujeme prvky a_{ij} rovné priemernej dobe prestupu medzi linkami L_i a L_j pre linky so spoločnou zastávkou a ∞ v opačnom prípade, potom naším algoritmom môžeme vypočítať tabuľku rozpätia priemernej doby trvania prestupov medzi linkami s hodnotami buniek d_{ij} medzi ľubovoľnými dvoma linkami.

Pri analýze vedenia pätnástich liniek MHD Nitra (ČERNÝ A KOL., 2005) sme zistili, že na niektorých zastávkach liniek sú potrebné až 3 prestupy a najdlhšia doba priemerného prestupu medzi linkami bola 32 minút.

4.4 Problém nakupujúceho obchodného cestujúceho

Pri tvorbe okružnej jazdy vozidla môže vzniknúť potreba riešiť nasledujúcu optimalizačnú úlohu:

Nech je daná matica dopravných nákladov medzi uzlami $N = \{0, 1, 2, \ldots, n\}$ *dopravnej siete* $\mathrm{D} = (d_{ij})$. *Množina uzlov obsahuje uzol* 0 *reprezentujúci východiskové miesto obchodného cestujúceho, ktorý hodlá nakúpiť sortiment* p *druhov tovaru,* $K = \{1, 2, \ldots, p\}$, *v množstvách* $(b_1, b_2, \ldots, b_p)$ *v niektorých predajniach, ktoré sú umiestnené v ostatných uzloch* $i \in M = N - \{0\}$, *a ponúkajú sortiment v množstvách* $(a_{i1}, a_{i2}, \ldots, a_{ip})$ *za jednotkovú cenu* $(c_{i1}, c_{i2}, \ldots, c_{ip})$.

Hľadá sa taká pochôdzka obchodného cestujúceho, ktorá začína a končí v 0 *a prechádza predajňami* i, *v ktorých nakupuje dostatočné množstvá* k*-teho druhu tovaru* y_{ik}. *Cieľom je minimalizovať celkové dopravné náklady a náklady na nákup vybraného tovaru.*

Pri hľadaní vhodného modelu sa necháme motivovať modelom Millera a kol., pre klasickú úlohu obchodného cestujúceho (bez nákupu, ale navštevujúceho všetky uzly siete), ktorý je uvedený v monografii (LAWLER ET AL., 1985) na str. 26–27, kde je cyklus (pochôdzka) obchodného cestujúceho repezentovaný $0-1$ maticou $\mathrm{X} = (x_{ij})$. Ak je $x_{ij} = 1$ potom hľadaný cyklus obsahuje úsek $i \to j$. My však nepotrebujeme navštíviť všetky predajne, čo docielime, ak položíme $d_{00} = \infty$ a $d_{ii} = 0$ v ostatných prípadoch keď $i \in M$. Potom $x_{ii} = 1$ bude znamenať, že predajňa i nebola navštívená. A tak dostávame nasledujúcu úlohu zmiešaného programovania BTSP:

$$\sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} + \sum_{i \in M} \sum_{k \in K} c_{ik} y_{ik} \to \min, \tag{4}$$

$$\sum_{j \in N} x_{ij} = 1, \qquad i \in N, \tag{5}$$

$$\sum_{i \in N} x_{ij} = 1, \qquad j \in N, \tag{6}$$

$$u_i + (n+1) x_{ij} - u_j \leqq n, \qquad i, j \in M,\ i \neq j, \tag{7}$$

$$y_{ik} + a_{ik} x_{ii} \leqq a_{ik}, \qquad i \in M,\ k \in K, \tag{8}$$

$$\sum_{i \in M} y_{ik} = b_k, \qquad k \in K, \tag{9}$$

$$y_{ik} \geqq 0, \qquad i \in M,\ k \in K, \tag{10}$$

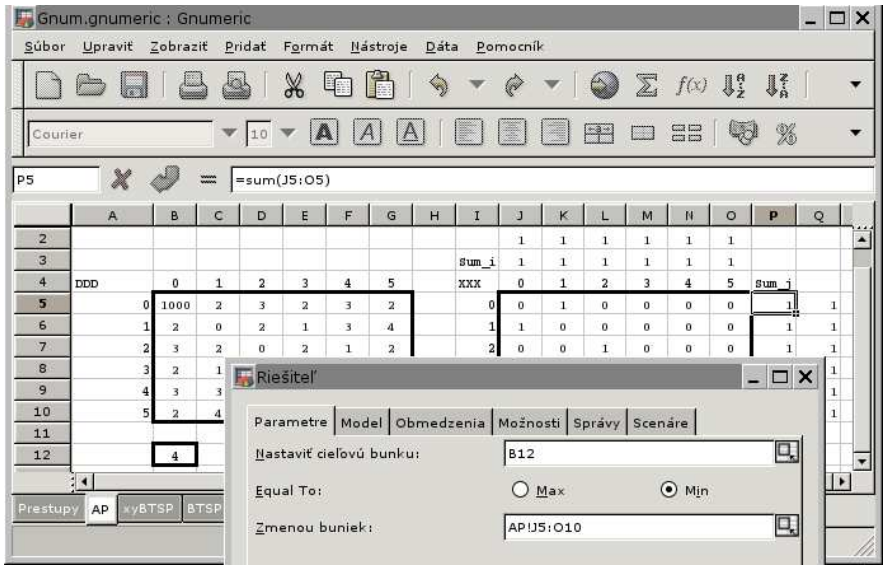
$$u_i \geqq 0, \qquad i \in M, \tag{11}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in N. \tag{12}$$

Obmedzujúce podmienky (5), (6), (12) sú podmienkami klasickej priraďovacej úlohy (assignment problem – AP). V liste AP na obrázku 3 máme riešenie príkladu s piatimi miestami. V oblastiach $DDD = AP!\$B\$5 : \$G\$10$ a $XXX = AP!\$J\$5 : \$O\$10$ máme maticu vzdialeností a maticu riešenia. Cieľovou bunkou je $B12 = sumproduct(DDD; XXX)$. Riadkové a stĺpcové súčty (5), (6) sú v oblastiach Sum_j, Sum_i v tvare súčtových formúl, napr. $P5 = sum(J5 : O5)$. Za zmienku stojí, že stačí rozkopírovať túto bunku do oblasti Sum_j a máme korektne definované všetky jej bunky. Riešením príslušnej úlohy LP sú nasledujúce cykly:

$$0 \to 3 \to 0, 1 \to 1, 2 \to 2, 4 \to 4, 5 \to 5.$$

Vidíme, že úloha nepožadovala explicitne podmienku (12) bivalentnosti premenných v oblasti XXX. Tá je zabezpečená unimodálnosťou obmedzujúcich podmienok (5), (6). Bivalentnosť premenných

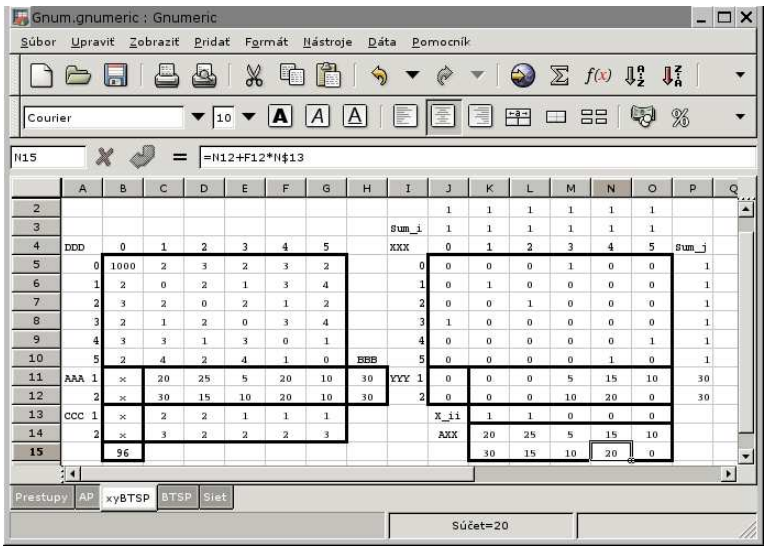


Obrázok 3: Riešenie priraďovacieho problému v BTSP

tak môže byť nahradená slabšou obligátornou podmienkou $x_{ij} \geqq 0$, čo vedie na úlohu lineárneho programovania.

Podmienka (8) v BTSP zabezpečí nulové množstvá tovaru $y_{ik} = 0$ z celého sortimentu predajne i, ak nebude navštívená, t. j. je v triviálnom cykle $i \rightarrow i$ definovanom premennou $x_{ii} = 1$. V opačnom prípade $x_{ii} = 0$ pripúšťa nákup z disponibilného množstva tovaru. Kapacitná podmienka (9) umožňuje nákup všetkého požadovaného tovaru. Podmienky (10), (11) sú obligátorné. Anticyklická podmienka (7) s obligátornou podmienkou (11) nám zaručuje, že v riešení úlohy neexistuje netriviálny cyklus neobsahujúci uzol 0.

Na obrázku 4 máme riešenie dvojsortimentovej úlohy $(p = 2)$ najskôr bez anticyklických podmienok, aby sme sa presvedčili o ich potrebe. Najskôr dodefinujeme nové oblasti vstupov a to:



Obrázok 4: Riešenie dvojsortimentovej BTSP bez anticyklických podmienok

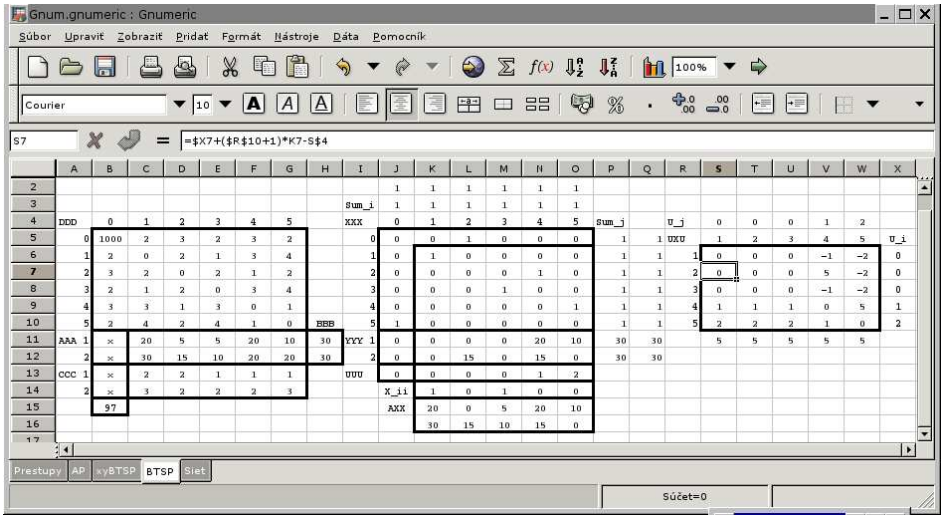
$AAA = xyBTSP!\$C\$11 : \$G\$12$,
$BBB = xyBTSP!\$H\$11 : \$H\$12$,
$CCC = xyBTSP!\$C\$13 : \$G\$14$
a oblasti výstupov $YYY = xyBTSP!\$K\$11 : \$O\$12$, $X_{ii} = xyBTSP!\$K\$6 : \$O\$6$, obsahujúce odkazy na diagonále bunky oblasti XXX a $AXX = xyBTSP!\$K\$14 : \$O\$15$, tvoriacu formuly ľavých strán obmedzení (8). Cieľová bunka $B15$ má tvar formuly

$$B15 = sumproduct(DDD; XXX) + sumproduct(CCC; YYY).$$

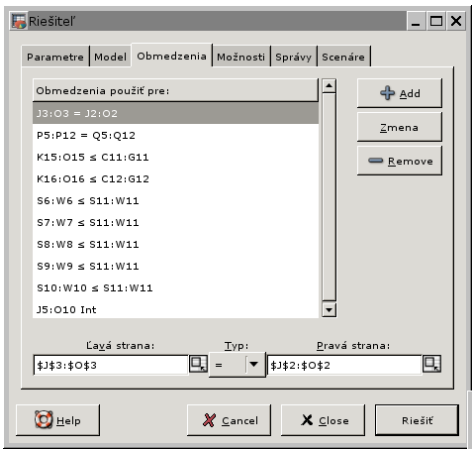
Riešenie je pri nutnej požiadavke celočíselnosti premenných oblasti XXX

$$0 \to 3 \to 0,1 \to 1,2 \to 2,5 \to 4 \to 5,$$

ale je neprípustné, nakoľko obsahuje netriviálny cyklus $5 \to 4 \to 5$.



Obrázok 5: Oblasti pre formuláciu dvojsortimentovej BTSP



Obrázok 6: Obmedzujúce podmienky na riešenie dvojsortimentovej BTSP

Doplnenie anticyklických podmienok (7) s obligatórnou podmienkou (11) na nové premenné u_i vyžaduje zaviesť ďalšiu oblasť UUU. Pretože *Riešiteľ* umožňuje pracovať len so súvislou oblasťou premenných, vložíme UUU za oblasť YYY, čím získame oblasť premenných $BTSP!J5 : O13$.

Na implemetáciu podmienok (7) potrebujeme definovať oblasť $UXU = BTSP!S5 : W10$, ktorej bunka $S7 = \$X7 + (\$R\$10 + 1) * K7 - S\$4$. Po jeho rozkopírovaní do celej oblasti UXU musíme ešte zmeniť diagonálne bunky z formúl na 0 hodnoty, čím ich vylúčime z optimalizácie.

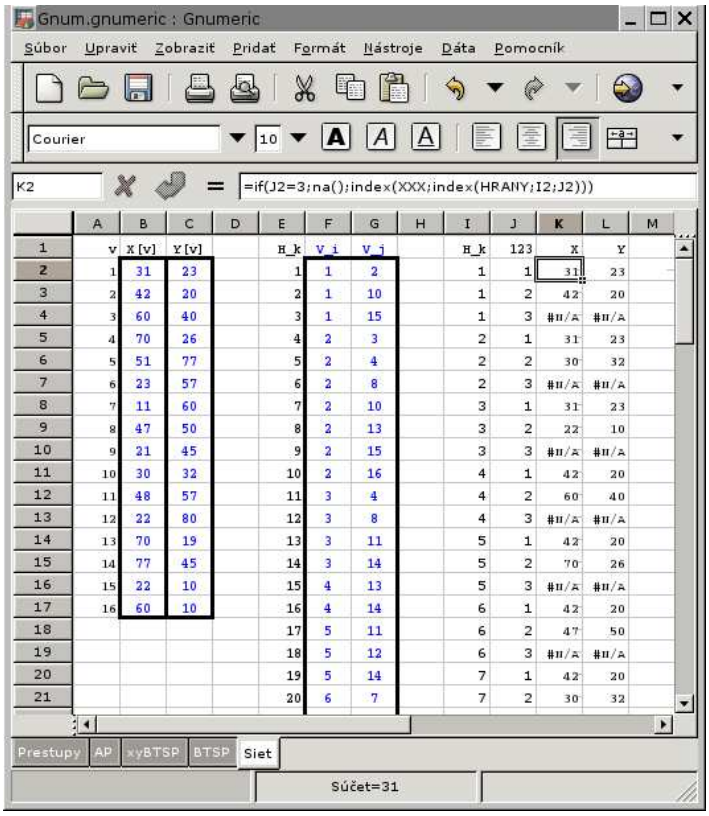
Po vložení všetkých podmienok dvojsortimentovej BTSP dostávame riešenie:

$$0 \to 2 \to 4 \to 5 \to 0,1 \to 1,3 \to 3,$$

čo znamená, že obchodný cestujúci nakupuje postupne v predajniach $2, 4, 5$. Príslušné množstvá požadovaného sortimentu nájdeme v oblasti YYY.

4.5 Graf dopravnej siete

Pre potreby grafickej reprezentácie niektorých prvkov, resp. grafových štruktúr na dopravnej sieti (napr. sledov, ciest, centier, diep, atď.), nemajú tabuľkové procesory priamu grafickú podporu. Samotný graf (presnejšie diagram) dopravnej siete môžeme pohodlne reprezentovať množinou úsečiek vďaka pomerne jednoduchému triku.



Obrázok 7: Vrcholy a hrany dopravnej siete

Na obrázku 7 máme v Gnumericu dopravnú sieť určenú množinou súradníc $X[v], Y[v]$ jej vrcholov v a množinou jej hrán $[v_i, v_j]$. V oblastiach $XXX = Siet!\$B\$2 : \$B\$17$ a $YYY = Siet!\$C\$2 : \$C\$17$ máme zoznam súradnic vrcholov a zoznam hrán je v oblasti $HRANY = Siet!\$F\$2 : \$G\$36$.

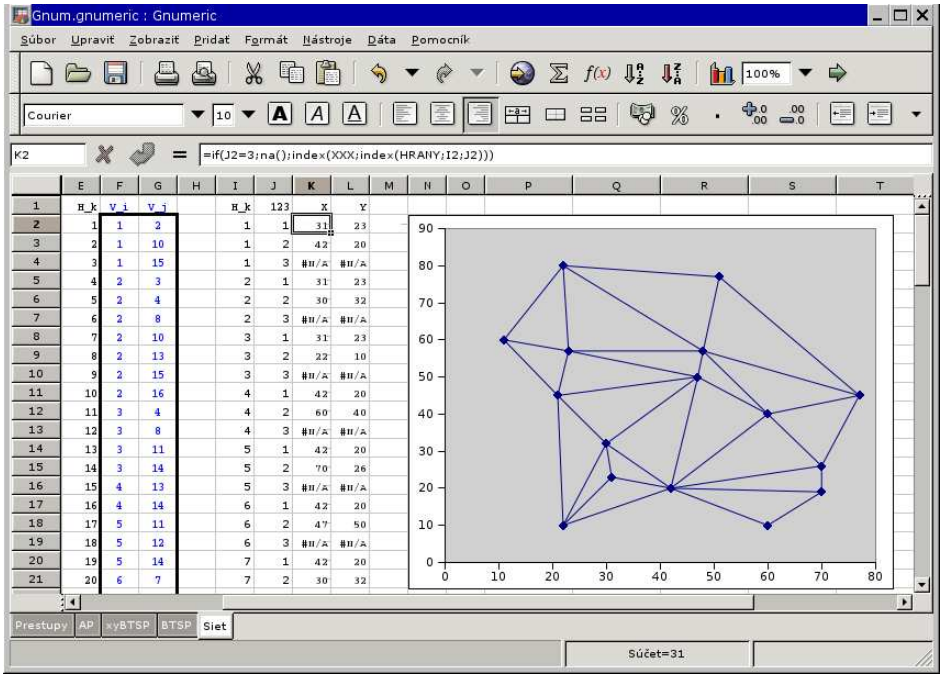
Definujeme oblasť $Siet!\$I\$2 : \$J\$4 = \{1,1;1,2;1,3\}$ a oblasť $Siet!I5 : J5 = \{F2+1, G2\}$. Formuly buniek $H2$ a $G2$ udávajú súradnice $X[v], Y[v]$ príslušného koncového vrcholu hrany $[v_i, v_j]$, alebo je to oddeľovač – nedefinovaná bunka $na()$:

$$H2 \;=\; if(G2 = 3, na(), index(XXX, index(HRANY, I2, J2))),$$
$$G2 \;=\; if(G2 = 3, na(), index(YYY, index(HRANY, I2, J2))).$$

Najskôr rozkopírujeme oblasť $I5 : J5$ do oblasti $I6 : J106$ a potom aj oblasť $K2 : L2$ do oblasti $K3 : L106$, čím získame súradnice X, Y úsečiek, ktoré reprezentujú hrany siete. Zobrazením oblasti $\$K\$2 : \$L\$106$ v XY bodovom grafe dostaneme, na obrázku 8, želaný obrázok diagramu dopravnej siete.

Analogicky sa postupuje, ak chceme v dopravnej sieti farebne vyznačiť niektoré jej hrany, napr. hrany najkratšej uv-cesty, najlacnejšej kostry, páriace hrany, atď. Jediným doteraz neodstráneným nedostat-

kom tohto prístupu je, že sa nám do grafu nepodarilo pridať popisky, napr. ako názvy vybraných uzlov.



Obrázok 8: Graf dopravnej siete

Vyššie uvedený trik teda spočíva v nakreslení sledu úsečiek ako prerušovanej lomenej čiary. Inou, podstatne pracnejšou cestou, je implementovať Tarryho prieskumom lomenú čiaru (komponent siete), ktorá každou hranou prechádza v jednom smere práve raz. Tento postup však možno odporúčať len „veľmi hravým“ študentom so znalosťou teórie grafov.

Pri analýze cestovných poriadkov MHD Nitra (ČERNÝ A KOL., 2005) bola sieť MHD tvorená pätnástimi linkami. Vrcholmi tu boli zastávky liniek a hranami úseky liniek medzi zastávkami. Získali sme tak prehľad o spoločných úsekoch liniek, čo nám pomohlo pri posudzovaní zhlukov autobusov cestovných poriadkov na zastávkach týchto úsekoch.

Záver

Uvedené praktické príklady ukazujú, že je možné aj bez znalosti procedurálneho programovania v tabuľkových procesoroch modelovať reálne optimalizačné úlohy. Limitujúcim faktorom v prípade použitia *Riešiteľa* vo výskume je jeho kvalita a najmä rozsah reálne riešiteľných úloh. No i v prípade malých inštancií pomáha pri tvorbe alternatívnych modelov.

Oboznamovanie študentov s touto formou modelovania sa začína stretávať na katedre matematických metód s dobrou odozvou, nakoľko im umožňuje pomerne ľahko overovať korektnosť vytvorených modelov. Na strane vyučujúceho zas poskytuje veľkú variabilitu pri príprave úloh na cvičenia, čo občas vedie až k obojstranne nákazlivému potešeniu z tvorivosti.

Tiež sme získali veľmi dobré skúsenosti s tabuľkovými procesormi pri rozhodovaním o voľbe modelov najmä v počiatočných fázach vedenia diplomových aj doktorantských prác. Vtedy sa totiž často stretávame s potrebou kritickej analýzy inštancií reálnych úloh a navrhovaním potrebných vstupov na tvorbu algoritmov v nízkoúrovňových programovacích jazykoch.

Domnievame sa, že dostupné tabuľkové procesory by mohli zaujať dominantné postavenie ako nástroje výučby modelovania v predmetoch operačnej analýzy. Ponechávajú totiž priestor pre tvorivosť študentov a pri nápaditom vedení umožňujú sústrediť ich pozornosť hlavne na návrh a analýzu modelov.

Použitá literatúra

LAWLER, E. L. – LENSTRA, J. K. – RINNOOY KAN, A. H. G. – SHMOYS, D. B. 1985. *The traveling salesman problem, A Guided Tour of Combinatorial Optimization*. A Wiley-Interscience Publication, ISBN 0-471-90413-9

ČERNÝ, J. – PALÚCH, S. – PEŠKO, Š. 2005. *Odborný posudok alternatívneho návrhu liniek a cestovných poriadkov MHD v meste Nitra*, štúdia

The Gnumeric Manual. `http://www.gnome.org/projects/gnumeric/doc/gnumeric.shtml`

PEŠKO, Š. 2005. *Pohodlná optimalizácia reálnych úloh v tabuľkových procesoroch*. Slovak Society for Operations Research, 7-th international seminar, APPLICATION OF QUANTITATIVE METHODS IN RESEARCH AND PRACTICE, pp. 29–35, Remata, ISBN 80-225-2079-9

PEŠKO, Š. 2002. *Vybrané modely logistiky v EXCELi*,
učebné texty k cvičeniam, `http://frcatel.fri.utc.sk/~pesko/volk.zip`

Primárnym cieľom výskumu nesmie byť viac faktorov, ale viac faktov so strategickou hodnotou.

PAUL WEISS

5 SPRACOVANIE A VIZUALIZÁCIA EXPERIMENTÁLNYCH DÁT

Ladislav ŠEVČOVIČ
Katedra fyziky, FEI
Technická univerzita v Košiciach

Úvod

Pri spracovaní výsledkov meraní a pozorovaní sa široko používajú metódy grafického zobrazenia. Číselné údaje, ako výsledky meraní a pozorovaní prezentované v tabuľkovej forme neumožňujú dostatočne názorne charakterizovať zákonitosti študovaných procesov, preto je vhodné tabuľku doplniť grafom (graf je vlastne vizuálna podoba údajov v tabuľke). Grafické znázornenie poskytuje názornejšiu predstavu o výsledkoch experimentu, umožňuje lepšie pochopiť fyzikálny zmysel študovaného procesu, zistiť (odhaliť) všeobecný charakter funkčnej závislosti premenných veličín a napokon stanoviť prítomnosť (existenciu) maxím alebo miním funkčnej závislosti.

Grafy taktiež umožňujú veľmi názorne porovnávať experimentálne hodnoty s teoretickou krivkou (závislosťou). Z precízne vyhotoveného grafu nameranej závislosti dvoch veličín sa dajú s dostatočnou presnosťou určiť napr. charakteristiky funkcie. Môžeme určiť polohu už spomínaných extrémov, inflexných bodov, pri lineárnej závislosti odčítať z grafu smernicu krivky a pod. Na okraj spomenieme, že sú známe metódy na grafické derivovanie a kvadratúru (integrovanie). Výhoda grafických metód sa uplatní predovšetkým pri meraniach s neekvidištančnými hodnotami nezávisle premennej veličiny, pretože číselné spracovanie výsledkov je pri takýchto meraniach zložitejšie (ťažšie), ako pre ekvidištančné merania, napriek tomu, grafické riešenie je vo všeobecnosti nepresnejšie. Spomínané postupy a metódy však stratili na význame v súvislosti s rozvojom výpočtovej techniky a jej aplikácií v experimentálnej praxi.

Na kreslenie grafov a ilustrácií existujú komerčné programy, ktoré sú bohato vybavené podprogramami na interpoláciu aj extrapoláciu, na fitovanie (nájdenie najlepšej aproximácie) nameranej závislosti zvolenou triedou funkcií, na optimalizáciu, obsahujú štatistické spracovanie výsledkov, vyhladenie závislostí, rôzne filtre a pod. V prostredí operačného systému GNU/Linux je bohatý výber programov na spracovanie a analýzu dát, ktoré sú na rozdiel od komerčného OS Windows šírené pod licenciou GPL (GNU General Public License)[29]*. Vymenujme niektoré matematicko-grafické programy:*

- *GNUPLOT,*

[29] *Projekt GNU bol založený na vybudovanie kompletného operačného systému, ktorého výsledky budú voľne dostupné počítačovej verejnosti. Programy dostupné v rámci GNU sú chránené tzv. GNU General Public License (GPL), ktorá na rozdiel od všetkých ostatných licencií garantuje každému právo programy slobodne používať a šíriť ďalej.*

- *Gnumeric a Calc z kancelárskeho balíka OpenOffice sú plnohodnotnou náhradou za komerčný program Excel z MS Office, ďalej sú to*
- *Veusz,*
- *LabPlot,*
- *Grace (xmgrace),*
- *Scigraphica,*
- *Octave,*
- *PyLab,*
- *QtiPlot a napokon*
- *Kpl.*

V tomto príspevku stručne opíšeme používanie posledných dvoch programov. Základom programovania v prostredí programu PyLab a jeho použitiu na podobné účely je venovaná príručka M. Kaukiča (2006) a programu Octave príručka J. Bušu (2006). Dôvody, ktoré viedli k tomuto výberu sú nasledujúce:

1. *Proces inštalácie a konfigurácie je veľmi jednoduchý a zvládne ho aj bežný používateľ výpočtovej techniky.*
2. *Oba programy majú prívetivé grafické prostredie, pod ktorým sa skrýva softvér profesionálnej kvality.*
3. *Program QtiPlot je vydareným klonom populárneho komerčného programu OriginLab^{TM}, ktorým môžete vykonať profesionálnu analýzu experimentálnych dát, nakresliť do grafu zložité funkcie. Grafický výstup je vysokej kvality vhodný na ďalšie spracovanie, napr. programom TEX.*
4. *Program Kpl je z pohľadu pomeru jednoduchosti ovládania k výkonnosti ojedinelý vo svojej kategórii. Môžeme ho dopĺňať vlastnými knižnicami na fitovanie dát a programovými skriptmi[30] na vykresľovanie všakovakých funkcií, ktoré sa napíšu a skompilujú v programovacom jazyku C. Vytvorené grafy môžeme exportovať do rôznych formátov, okrem iného do Encapsulated Postscript (EPS).*
5. *Parametre fitovacích funkcií, ktoré sme získali po spracovaní referenčných dát na testovanie matematických knižníc a algoritmov týmito programami sú v dobrej zhode s hodnotami uverejnenými na internetovej stránke Národného inštitútu štandardov a technológií Spojených štátov amerických (*NIST, *2006).*

Problematika spracovaná v príspevku je usporiadaná do piatich častí, pričom každá sa sústreďuje na jeden tématický celok. V 5.1. časti uvádzame krátky súpis hlavných pojmov z oblasti neistôt merania. Časť 5.2, ktorá je spoločná pre 5.3 a 5.4, je venovaná základným numerickým metódam na spracovanie experimentálnych dát. Čitateľ by v každom prípade mal vedieť, čo a ako počítačovým programom analyzuje a aká je podstata metódy, ktorú používa. Pri prvom čítaní príručky je možné túto kapitolu preskočiť.

Ťažiskom práce sú časti 5.3 a 5.4 , čitateľ sa z nich dozvie, aké možnosti jednotlivé programy poskytujú a ako ich rýchlo použiť na spracovanie a vizualizáciu nameraných dát, prípadne zobrazenie funkcií.

V záverečnej 5.5. časti sa kratúčko venujeme základným pravidlám na tvorbu úhľadného grafu.

Príspevok je určený všetkým, ktorí potrebujú rýchle zvládnuť prácu s programom na spoľahlivé numerické spracovanie nameraných dát a ich kvalitnú grafickú prezentáciu do publikácií, vysokoškolských kvalifikačných prác, konferenčných zborníkov, posterov a pod.

Ďakujem Jaroslavovi Skřivánkovi, Jánovi Bušovi a Igorovi Leššovi za starostlivé prečítanie rukopisu a cenné prípomienky, ktoré prispeli k spresneniu niektorých formulácií a ku skvalitneniu tejto práce.[31]

Košice 2007 *L. Ševčovič*

[30] Skripty sú ASCII (textové) súbory obsahujúce príkazy. Sú tiež známe pod názvom *zdrojové súbory* (source files) alebo *dávkové súbory* (batch files). Keď skript spustíte, príkazy sa vykonajú (interpretujú) jeden za druhým počnúc od začiatku tak, akoby ste ich písali samostatne priamo v príkazovom riadku jeden za druhým. Ide o akúsi obdobu dávkových príkazov v OS MS DOS.

[31] *Elektronická verzia tohto textu, doplnky a opravy sú prístupné na URL adrese* `http://people.tuke.sk/ladislav.sevcovic/`. *Prípomienky a návrhy, ktoré pomôžu vylepšiť ďalšie vydanie príručky, zasielajte na adresu: RNDr. Ladislav Ševčovič (*`Ladislav.Sevcovic@tuke.sk`*), Katedra fyziky, FEI, Technická univerzita v Košiciach, Park Komenského 2, 041 20 Košice.*

5.1 Základné pojmy a definície z oblasti neistôt meraní

V súčasnosti sa v metrológii, pri fyzikálnych a technických meraniach postupne prechádza na nové metódy vyjadrovania odchýlok. Doterajšie *chyby meraní* sú v súlade s medzinárodnými predpismi ISO a IEC nahradzované *neistotami meraraní*. Za hlavný dokument je možné považovať predovšetkým smernicu, ktorá bola vydaná pod názvom *Guide to Expression of the Uncertainty of Measurement (GUM)* (ISO, Switzerland 1995) medzinárodnými metrologickými orgánmi v roku 1993, korigovaná a doplnená v roku 1995. Pre prírodovedcov bude iste zaujímavé navštíviť WWW stránku Národného inštitútu štandardov a technológií Spojených štátov amerických (NIST, 2006) `http://physics.nist.gov/cuu/Uncertainty/basic.html`, ktorá prináša základné informácie o neistotách a ich vyjadrovaní.
Uvádzame zoznam niektorých významných medzinárodných organizácií, ktoré tento projekt podporujú:

- BIPM Bureau International des Poids et Mesures
- IEC International Electrotechnical Commission
- IFCC International Federation of Clinical Chemistry
- ISO International Organization for Standardization
- IUPAC International Union of Pure and Applied Chemistry
- IUPAP International Union of Pure and Applied Physics
- OIML International Organization of Legal Metrology

Aplikovanie a zavádzanie nových experimentálnych metód, prístrojov a pracovných postupov, práve tak, ako používanie starších a osvedčených metód, by malo byť podložené štúdiom ich vlastností, aby ich neskoršie používanie nenarážalo na nejasnosti pri interpretácii výsledkov získaných použitými postupmi a metódami.

Medzi základné problémy nepochybne patria otázky presnosti a správnosti, alebo skôr nepresnosti a nesprávnosti meraní. Definícia týchto dvoch pojmov je dosť ťažká, avšak ich obsah je intuitívne celkom jasný. *Správnosť* súvisí s tým, ako sa meranie (namerané hodnoty) zhodujú so skutočnou meranou hodnotou, zatiaľ čo *presnosť* súvisí s tým, ako sa opakované merania (namerané hodnoty) zhodujú medzi sebou. Môžeme teda hovoriť o *systematických chybách*, ktoré sa prejavujú ako stály rozdiel medzi nameranými hodnotami (alebo ich strednou hodnotou) a skutočnou (správnou)[32] hodnotou a o *náhodných chybách*, ktoré sa prejavujú vo variabilite nameraných hodnôt okolo ich strednej hodnoty. Uvádzajú sa ešte *hrubé chyby*, ktoré vznikajú napr. poruchou prístrojov, nepozornosťou pracovníka, krátkodobou zmenou experimentálnych podmienok a pod.

Variabilita nameraných hodnôt má dve základné príčiny, ktoré väčšinou pôsobia súčasne:

- ➤ vlastnosti vyšetrovaného javu (napr. neodstraniteľná nehomogenita vyšetrovaného materiálu, fluktuácie vyvolané fyzikálnymi procesmi a pod.)
- ➤ a technické nedostatky meracej metódy (nepresnosť meracieho zariadenia, nepresnosť pri príprave vzoriek, zmeny prostredia, v ktorom meranie prebieha, t. j. teplota, vlhkosť vzduchu a pod. a tiež vplyv osôb, ktorí sa experimentu, merania zúčastňujú).

Je potrebné teda pamätať na oba zdroje neistôt a snažiť sa o udržanie čo najstabilnejších podmienok na prevádzanie meraní. V ďalšom nás budú zaujimať otázky súvisiace s presnosťou (opakovateľnosťou), budeme pritom predpokladať, že metóda merania je správna, t. j. že sa správna hodnota rovná

[32] Niektorí autori používajú aj pomenovanie „pravá“ vo význame hodnoty získanej naprosto presným meraním (PALENČÁR A KOL., 2000).

strednej hodnote rozdelenia nameraných hodnôt. O náhodných chybách sa spravidla predpokladá, že sú rozdelené normálne, tento predpoklad, ktorý býva obvykle splnený aspoň približne budeme v ďalšom akceptovať; vo všeobecnosti však nemusí byť splnený.

Stručný slovník pojmov

Aritmetický priemer je súčet hodnôt pozorovaní (meraní, odčítaní a pod.) delený počtom hodnôt

$$x_i = \overline{X}_i = \frac{1}{n}\sum_{k=1}^{n} X_{i,k}. \tag{1}$$

Citlivosť meracieho prístroja je prakticky zmena hodnoty meranej veličiny, ktorá korešponduje s najmenším dielikom stupnice. Pre číslicové meracie prístroje je to podiel počtu číslic zmeny údaja a zmeny vstupnej veličiny, ktorá zmenu vyvolala.

Disperzia (variancia) pozri *Rozptyl*.

Chyba, máme tu ma mysli chybu meracieho prístroja, ktorá má svoj pôvod v konštrukčnom usporiadaní, v konečnom delení stupnice meraných hodnôt a pod. Základnými zdrojmi chýb sú:

- nedokonalosť meracích prístrojov,
- stárnutie a opotrebenie meracích prístrojov, čím sa môžu meniť ich charakteristiky a parametre,
- chyby experimentátora,
- nepresné metódy vyhodnocovania meraní,
- vplyv linearizácie, interpolácie a zaokrúhľovania,
- zlá kalibrácia, inštalácia alebo umiestnenie prístrojov atď.

Chyba merania (odchýlka) je rozdiel medzi nameranou X_i a skutočnou hodnotou μ určujúcej veličiny v tom istom okamihu. Pri meraní určitej veličiny sa prevádza len konečný počet meraní. Predpokladajme, že bolo prevedené meranie veličiny X a získané hodnoty $X_1, X_2, \ldots, X_n$, ktorých chyby môžeme vyjadriť vzťahom

$$\Delta X_i = X_i - \mu \tag{2}$$

a majú normálne rozdelenie. Aritmetický priemer nameraných hodnôt X_i podľa (1) dáva najpravdepodobnejšiu hodnotu meranej veličiny $\mu \equiv \overline{X}_i$.

Korelácia je kvantitatívna miera vzťahu medzi dvoma veličinami vyjadrujeme ju ako

$$r(x_i, x_k) = \frac{\sum_{i=1}^{n}(x_i - \overline{x}_i)(x_k - \overline{x}_k)}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x}_i)^2 \sum_{i=1}^{n}(x_k - \overline{x}_k)^2}} = \\ = \frac{D(x_i, x_k)}{\sqrt{D(x_i)}\sqrt{D(x_k)}}, \tag{3}$$

kde $D(x_i, x_k)$ je kovariancia, $D(x_i)$ a $D(x_k)$ sú disperzie, pričom $-1 \leqq r(x_i, x_k) \leqq 1$. Hodnota $r(x_i, x_k) = 1$ znamená, že ide o funkčnú rastúcu závislosť, hodnota $r(x_i, x_k) = -1$ znamená funkčnú klesajúcu závislosť. Obidva prípady a prívlastok funkčný zodpovedajú situácii, keď všetky body ležia na priamke. Ak sú skúmané veličiny nezávislé, bude $r(x_i, x_k) = 0$, ale naopak nulový koeficient korelácie nemusí znamenať nezávislosť. Môže ísť o (to byť) zložitejší vzťah, napr. závislosť, ktorá v jednej časti rastie a v druhej klesá. Korelácia je štatistická (pravdepodobnostná) závislosť dvoch náhodných veličín (premenných). Symbolom x_i a x_k sa tu nedáva bežný význam nezávisle a závisle premennej, pretože ani jednej z náhodných premenných neprisudzujeme charakter príčiny alebo následku.

Kovariancia (vzájomný rozptyl) je spoločná menlivosť daných dvoch vlastností a charakterizuje väzbu hodnôt výberu

$$D(x_i, x_k) = \frac{1}{n-1}\sum_{j=1}^{n}(x_{i,j} - \overline{x}_i)(x_{k,j} - \overline{x}_k). \tag{4}$$

Merací prístroj je zariadenie určené na prevod meranej veličiny na signál nesúci informáciu o jej hodnote (údaj). Charakterizujeme ho cilivosťou, schopnosťou replikovať údaje, rozptylom, presnosťou, hustotou pravdepodobnosti chýb meraní a pod. Priebeh registrácie môže byť spojitý, keď je registrácia číslicová potom má schodíkový tvar.

Náhodný výber rozsahu n je n-tica náhodných premenných, ktoré sú stochasticky nezávislé a majú rovnak rozdelenie pravdepodobnosti.

Neistota merania (skrátene neistota) je parameter, ktorý súvisí s výsledkom merania a ktorý určuje rozptyl hodnôt, ktoré môžeme ešte racionálne priradiť k meranej veličine. (*Neistota* je teda interval, v ktorom sa s určitosťou, definovanou pravdepodobnosťou bude skutočná hodnota nachádzať.) Neistoty (z jednotlivych zdrojov) môžeme vyhodnocovať dvoma základnými metódami:

- štatistickými metódami z nameraných údajov, ktoré sa nazývajú *neistoty stanovené metódou A*, skrátene ich voláme neistoty typu A a označujeme ich ako $u_{\mathrm{A}}(x_i)$,
- neistoty získané iným spôsobom ako v predošlom prípade, ktoré sa nazývajú *neistoty stanovené metódou B*, skrátene ich voláme neistoty typu B a označujeme ich ako $u_{\mathrm{B}}(x_i)$ (napr. výsledky získané pri predchádzajúcich meraniach, špecifikácie od výrobcu meracieho prístroja, údaje z certifikátov, kalibračných listov, neistoty referenčných údajov a pod.).

Vhodným zlúčením štandardných neistôt zo všetkých zdrojov získame celkovú (kombinovanú) štandardnú neistotu. Treba zdôrazniť, že nečleníme neistoty, ale metódy ich vyhodnocovania na metódu A a metódu B. Neistoty určené oboma metódami sú rovnocenné, pokiaľ boli určené korektne.

Normálné (Gaussove) rozdelenie sa používa na aproximáciu v prípadoch, keď sa často vyskytujú malé odchýlky od menovitej hodnoty, pričom s rastúcou veľkosťou odchýlok pravdepodobnosť ich výskytu klesá, napr. keď je zdrojom neistoty merací prístroj od spoľahlivého výrobcu (môžeme predpokladať, že väčšina prístrojov bude zdrojom malých chýb).

Opakovateľnosť (replikovateľnosť) je charakteristika meracieho systému a znamená, že distribučná funkcia chyby merania sa nemení pri opakovaní meraní, teda akékoľvek súbory dát získané z nezávislých opakovaných meraní hodnoty f nejakej veličiny je možné modelovať ako realizáciu náhodných výberov z toho istého rozdelenia pravdepodobnosti. Stálosť distribučnej funkcie je podmienkou replikovateľnosti meracieho systému. Merací prístroj bez driftu musí pri opakovanom meraní jednej a tej istej hodnoty vykazovať vlastnosť, ktorá sa volá *replikovateľnosť*. Matematicky to znamená, že súbory nezávisle nameraných dát sú realizáciou *náhodného výberu* z toho istého rozdelenia pravdepodobnosti (KUBÁČEK A KUBÁČKOVÁ, 2000).

Presnosť merania je miera nesúhlasu nameranej a skutočnej hodnoty určujúcej veličiny.

Reprodukovateľnosť merania je opakovateľnosť výsledkov merania prevedených za rovnakých podmienok, ale v rôznych časových okamihoch.

Rovnomerné (pravouhlé) rozdelenie pravdepodobnosti sa používa v prípadoch, keď pravdepodobnosť výskytu ktorejkoľvek odchýlky v celom intervale $\pm z_{j\max}$ je rovnaká. V praxi sa používa najčastejšie, predovšetkým preto, že väčšinou nemáme k dispozícii dostatočné poznatky o rozdelení pravdepodobnosti výskytu odchýlok a teda nemáme dôvod dávať niektorým odchýlkam prednosť tým, že použijeme iný typ rozdelenia. Spojitá náhodná veličina X sa riadi zákonom rovnomerného rozdelenia (má rovnomerné rozdelenie), keď jej možné hodnoty ležia (nachádzajú sa) v určených hraniciach, okrem toho v hraniciach tohto intervalu sú všetky hodnoty náhodnej veličiny rovnako pravdepodobné (majú rovnakú hustotu pravdepodobnosti rozdelenia). S náhodnou veličinou, ktorá má vlastnosti rovnomerného rozdelenia sa často stretávame v meracej technike (praxi) pri zaokrúhľovaní údajov z meracích prístrojov na celý dielik delenia stupnice. Chyba pri zaokrúhľovaní údaja stupnice na najbližší dielik delenia je náhodná veličina X_i, ktorá môže nadobúdať ľubovoľnú hodnotu medzi dvoma susednými dielikmi stupnice s konštantnou hustotou pravdepodobnosti. *Keď sa chyby podriaďujú zákonu rovnomerného rozdelenia počet prevedených meraní nemá vplyv na stupeň hodnovernosti výsledku merania* na

rozdiel od iných zákonov rozdelenia, napr. normálneho, kde zvyšovaním počtu meraní a ich spracovaním, môžeme podstatne zvýšiť presnosť odhadu meranej veličiny.

Rozdelenie pravdepodobnosti je funkcia vyjadrujúca pravdepodobnosť, že meranie (náhodná veličina) nadobudne určitú hodnotu alebo hodnoty z istého intervalu.

Rozptyl je stredná hodota druhej mocniny odchýlky náhodnej veličiny od jej strednej hodnoty

$$D(X_i) = s^2(X_i) = \frac{\sum_{k=1}^{n}(X_{i,k} - \overline{X}_i)^2}{n-1}. \tag{5}$$

Rozptyl registrácie je definovaný *disperziou* (druhým centrálnym momentom teoretickej distribúcie chýb merania daným prístrojom). Niekedy sa nazýva aj charkteristikou *vnútornej presnosti*. Charakteristika *vonkajšej presnosti* pri danom počte opakovaných meraní n konkrétnej hodnoty f_i je daná hodnotou veličiny

$$E\left[(\hat{f} - f_i)^2\right] = \frac{s^2}{n} + B^2, \tag{6}$$

kde $B = E(\hat{f}) - f_i$, $\hat{f}$ je uvažovaný odhad veličiny f, $E[\cdot]$ a $E(\cdot)$ vyjadrujú strednú hodnotu veličiny. Hodnota B sa nazýva vychýlenosť (bias) odhadu $\hat{f}$. Keď prístroj (súbor dát získaných prístrojom) je charakterizovaný hustotou pravdepodobnosti, pre ktorú platí $E(X_i) = 0$, potom sú charakteristiky vonkajšej a vnútornej presnosti zhodné. *Rozptyl (disperzia) meracieho prístroja* je jeho dôležitou charakteristikou, avšak *nevystihuje úplne* štatistické správanie chýb. Lepšiou charakteristikou správania chýb meraní je ich rozdelenie pravdepodobnosti. Pri použití meracieho prístroja vzniká problém, ako túto hustotu poznať aspoň približne (KUBÁČEK A KUBÁČKOVÁ, 2000).

Rozšírená neistota je veličina definujúca interval okolo výsledku merania, ktorý zahrňuje veľkú časť rozdelenia pravdepodobnosti hodnôt, ktoré je môžné priradiť k meranej veličine.

Signál je fyzikálna veličina, ktorá je nositeľkou pridanej namodulovanej informácie.

Smerodajná odchýlka je druhá odmocnina z rozptylu príslušného rozdelenia pravdepodobnosti.

Štandardná neistota merania je neistota merania vyjadrená ako smerodajná odchýlka. Pojem *štandardná neistota* (v meraní) a *smerodajná odchýlka* (odmocnina z disperzie resp. z rozptylu; charakterizuje presnosť merania) znamenajú to isté.

Trojuholníkové (Simpsonove) rozdelenie sa používa na modelovanie situácií (prípadov), ktoré sa podobajú normálnemu rozdeleniu.

Vstupný odhad je výsledok merania vypočítaný z odhadov vstupných dát pomocou funkcie modelu merania.

Výberová smerodajná odchýlka je druhá odmocnina výberového rozptylu. Náhodnú chybu v klasickej teórii chýb najčastejšie zastupuje smerodajná odchýlka výberového súboru

$$s(X_i) = \sqrt{\frac{\sum_{k=1}^{n}(X_{i,k} - \overline{X}_i)^2}{n-1}}, \tag{7}$$

zriedkavo smerodajná odchýlka aritmetického priemeru

$$s(\overline{X}_i) = \frac{s(X_i)}{\sqrt{n}} = \sqrt{\frac{\sum_{k=1}^{n}(X_{i,k} - \overline{X}_i)^2}{n(n-1)}}. \tag{8}$$

Výberový rozptyl je veličina charakterizujúca roptýlenie výsledkov série n pozorovaní (meraní, odčítaní a pod.) rovnakej meranej veličiny, získaná ako druhá mocnina *výberovej smerodajnej odchýlky*.

Výsledok merania je hodnota, ktorá prislúcha meranej veličine a bola získaná meraním. Keď použijeme pomenovanie výsledok merania, musíme uviesť, či sa vzťahuje na:

- údaj meradla (meracieho prístroja),
- nekorigovaný výsledok,
- korigovaný výsledok.

Nekorigovaný výsledok je taký výsledok merania, pri ktorom nie sú uplatnené korekcie známych systematických chýb. *Korigovaný výsledok* merania je výsledok po korekcii systematických chýb. Výsledkom merania je často hodnota získaná výpočtom z výsledku viacerých opakovaných meraní.Vzhľadom na to, že skutočnú hodnotu

meranej veličiny zisťujeme procesom merania, ktorý je zaťažený rôznymi chybami, výsledok merania je len odhadom (skutočnej) hodnoty meranej veličiny. Pri udávaní výsledku merania je preto dôležité stanoviť aj kvalitu tohto odhadu, ktorá sa definuje pomocou *neistoty*. *Úplný údaj výsledku* obsahuje okrem výslednej hodnoty meranej veličiny aj údaj o neistote merania. (WIMMER A KOL., 2001, str. 103)
Výstupná veličina je veličina, ktorá pri vyhodnotení merania predstavuje meranú veličinu.

Typy neistôt

Ako sme to už spomenuli, je pojem neistota (neistota merania) spojený s označením parametera súvisiaceho s výsledkom merania a charakterizujúceho rozsah hodnôt, ktoré môžeme racionálne priradiť meranej veličine. Zmienili sme sa aj o tom, že neistota sa skladá z niekoľkých zložiek. Na určenie ich veľkosti sú principiálne k dispozícii tieto dve metódy:

- štatistické spracovanie nameraných údajov (metóda typu A),
- iné ako štatistické spracovanie nameraných údajov (metóda typu B).

Niekedy sa neistoty získané metódou A stručne označujú ako *neistoty typu A*, podobne neistoty získané metódou B ako *neistoty typu B*. Z týchto základných typov neistôt sa potom ľahko pomocou súčtu ich štvorcov určí výsledná *kombinovaná neistota* u_C. Predpokladajme, že máme jednoduchú výstupnú modelovú funkciu niekoľkých vstupných parametrov $f = F(x_1, x_2, \ldots, x_i, \ldots, x_m)$, kde f je odhad výstupnej veličiny, x_i sú odhady vstupných veličín a F je známy funkčný vzťah. Vo všeobecnosti potom môžeme pre neistotu u_f odhadu f napísať vzťah

$$u_f = \sqrt{\sum_{i=1}^{m} A_i^2 u_{x_i}^2}, \tag{9}$$

kde u_{x_i} sú jednotlivé zložky neistôt, A_i je koeficient citlivosti (prevodu) príslušného zdroja neistoty, ktorý poznáme alebo sa určí ako parciálna derivácia funkcie f podľa príslušnej vstupmej veličiny x_i

$$A_i = \frac{\partial f}{\partial x_i} = \frac{\partial F(x_1, x_2, \ldots, x_i, \ldots, x_m)}{\partial x_i}. \tag{10}$$

Vidieť, že celá metodika určenia je dosť komplikovaná, preto v nasledujúcich častiach ukážeme len základnú metodiku a hĺbavého čitateľa odkážeme na preštudovanie príslušnej literatúry (PALENČÁR A KOL., 2000; VDOLEČEK A KOL., 2001a,b, 2002a,b; UHRIN A KOL., 2006; MELOUN A MILITKÝ, 2004).

Vyhodnotenie štandardných neistôt vstupnej veličiny metódou typu A

Metóda vyhodnotenia tohto typu neistôt je založená na štatistickej analýze opakovanej série meraní. Keď máme n nezávislých rovnako presných pozorovaní ($n > 1$), bude odhad výslednej hodnoty f reprezentovaný hodnotou výberového priemeru (aritmetického priemeru) $\overline{f}_i$. Neistota prislúchajúca odhadu f sa určí ako smerodajná odchýlka $s(\overline{f}_i)$ tejto výslednej hodnoty, teda výberového priemeru $\overline{f}_i$. Tento typ neistoty sa označí ako $u_{\mathrm{A}f}$ a môžeme ju vyjadriť v tvare

$$u_{\mathrm{A}f} = s(\overline{f}_i) = \frac{s(f_i)}{\sqrt{n}} = \sqrt{\frac{\sum_{k=1}^{n}(f_{i,k} - \overline{f}_i)^2}{n(n-1)}}. \tag{11}$$

Táto neistota je spôsobená kolísaním nameraných údajov. Keď máme k dispozícii malý počet meraní ($n < 10$) je hodnota určená podľa tohto vzťahu nespoľahlivá a mali by sme túto neistotu (spôsobenú kolísaním nameraných údajov) odhadnúť metódou typu B na základe iných informácií ako sú súčasné namerané údaje.

Vyhodnotenie štandardných neistôt vstupnej veličiny metódou typu B

Ako sme to už uviedli vyhodnotenie neistoty vstupnej veličiny metódou typu B je založené na iných ako štatistických postupoch analýzy série pozorovaní. Naskytuje sa možnosť analógie

so systematickými zložkami chýb, avšak neide o jednoznačnú súvislosť pretože metódou typu B je možné odhadnúť aj vplyv náhodnej chyby, napr. pri kalibrácii použitím predchádzajúcich meraní. Štandardná neistota typu B sa odhaduje pomocou racionálneho úsudku na základe všetkých možných a dostupných informácií. Najčastejšie sa používajú:

- údaje výrobcu meracieho prístroja,
- skúsenosti z predchádzajúcej série meraní,
- skúsenosti s vlastnosťami správania materiálov a techniky a poznatky o nich,
- údaje získané z kalibrácií a z certifikátov,
- neistoty referenčných údajov v príručkach.

Pri určovaní neistoty typu B sa vychádza z čiastkových neistôt jednotlivých zdrojov $u_{\mathrm{B}z_j}$. Keď poznáme maximálnu odchýlku j-teho zdroja neistoty $z_{j\max}$, neistota $u_{\mathrm{B}z_j}$ sa určí podľa vzťahu

$$u_{\mathrm{B}z_j} = \frac{z_{j\max}}{k}, \tag{12}$$

kde k je súčiniteľ získaný zo zákona rozdelenia pravdepodobnosti, ktorým sa riadi zdroj neistoty (napr. pre normálne rozdelenie je $k = 2$, prípadne 3, pre rovnomerné rozdelenie $k = \sqrt{3}$, pre trojuholníkové rozdelenie $k = \sqrt{6}$ atď., podrobnosti pozri odkaz *Rovnomerné rozdelenie pravdepodobnosti*).[33] V niektorých prípadoch môže byť už neistota $u_{\mathrm{B}z_j}$ známa, napr. z kalibračného certifikátu výrobcu meracieho prístroja. Výsledná neistota sa metódou B určí podobne ako v prípade závislosti vstupných funkcií od viacerých parametrov, pre p zdrojov $z_1, z_2, \ldots, z_j, \ldots, z_p$ platí

$$u_{\mathrm{B}f} = \sqrt{\sum_{j=1}^{p} A_j^2 u_{\mathrm{B}z_j}^2}, \tag{13}$$

kde $u_{\mathrm{B}z_j}$ sú neistoty jednotlivých zdrojov a A_j sú ich súčinitele citlivosti. Takýmto spôsobom sa neistota vyhodnocovaná metódou typu B prevedie do nového tvaru a vzhľadom na predchádzajúce predstavy aj tieto neistoty získavajú charakter smerodajnej odchýlky. Ako s takými, prípadne s ich druhými mocninami ako s rozptylom, sa pracuje ďalej. V samostatnej časti to ukážeme na príkladoch.

Kombinovaná a rozšírená neistota

V praxi sa zriedka vystačí len s jedným alebo druhým typom neistoty samostatne. Potom je potrebné stanoviť výsledný efekt kombinovaných neistôt meraní (alebo určení) oboch typov, A a B. Výsledná kombinovaná neistota veličiny f sa označuje $u_{\mathrm{C}f}$ a je vlastne odhadom smerodajnej odchýlky spojenej s výsledkom, ktorý je rovný druhej odmocnine kombinovaného rozptylu získaného zo všetkých rozptylov vstupných veličín; druhej odmocnine zo súčtu štvorcov oboch typov neistôt A a B podľa vzťahu

$$u_{\mathrm{C}f} = \sqrt{u_{\mathrm{A}f}^2 + u_{\mathrm{B}f}^2} \tag{14}$$

a zo všetkých prípadných kovariancií. Postup na stanovenie kombinovanej standardnej neistoty je iný pre *nekorelované* a iný pre *korelované* veličiny.

Pre veličiny *nekorelované* (vzájomne nezávislé) je kombinovaná standardná neistota $u_{\mathrm{C}f}$ stanovená ako kladná druhá odmocnina z kombinovaného rozptylu $u_{\mathrm{C}f}^2$, ktorý sa určí pomocou vzťahu

$$u_{\mathrm{C}f}^2 = \sum_{i=1}^{n} \left(\frac{\partial F}{\partial x_i} \right)^2 u^2(x_i), \tag{15}$$

kde F je funkcia vyjadrujúca závislosť výstupnej veličiny f od vstupných veličín x_i.

Pre veličiny *korelované* (vzájomne závislé) vstupujú do neistôt aj ich *kovariancie* $D(x_i, x_k)$ vzťahujúce sa na odhady x_i a x_k ako ďalší vplyv na vyjadrovanú neistotu

$$D(x_i, x_k) = u(x_i) \cdot u(x_k) \cdot r(x_i, x_k), \tag{16}$$

kde $r(x_i, x_k)$ je korelačný koeficient. Kovarianciu dvoch náhodných veličín $x_i^{(k)}$ a $x_k^{(k)}$, ktorých odhady sú získané z hodnôt opakovaných

[33] V prípadoch, keď môžeme odhadnúť len dolnú (z^-) a hornú hranicu (z^+) meranej veličiny X_i a ďalšie informácie nemáme k dispozícii, je vhodné priradiť (prisúdiť) meranej veličine *rovnomerné rozdelenie* a ako mieru neistoty $u_{\mathrm{B}}(X_i)$ použiť odhad smerodajnej odchýlky $s(X_i)$ tohto rozdelenia, teda $u_{\mathrm{B}}(X_i) = s(X_i) = \sqrt{\frac{(z^+ - z^-)^2}{12}}$.

meraní a sú vyjadrené aritmetickými priemermi $\overline{x}_i^{(k)}$ a $\overline{x}_k^{(k)}$, môžeme určiť podľa nasledujúceho vzťahu (horný index (k) vyjadruje, že máme do činenia s korelovanými veličinami)

$$D(x_i, x_k) = \frac{1}{n-1} \sum_{j=1}^{n} (x_{i,j}^{(k)} - \overline{x}_i^{(k)})(x_{k,j}^{(k)} - \overline{x}_k^{(k)}). \tag{17}$$

Kombinovaný rozptyl vzťahujúci sa na odhad funkčne závislej výstupnej veličiny od korelovaných vstupných veličín je vyjadrený vzťahom

$$u_{\mathrm{C}}^2(f) = \sum_{i=1}^{n} \left(\frac{\partial F}{\partial x_i} \right)^2 u^2(x_i) + \\ + 2 \sum_{i=1}^{n-1} \sum_{k=i+1}^{n} \frac{\partial F}{\partial x_i} \frac{\partial F}{\partial x_k} D(x_i, x_k). \tag{18}$$

Kombinovaná štandardná neistota je rovná odmocnine s takto vyjadreného kombinovaného rozptylu. Z vyjadrenia rozvoja neistôt získame príspevky jednotlivých zdrojov neistôt k celkovej neistote, preto je vhodné previesť v tejto fáze analýzu príspevkov jednotlivých zdrojov celkovej (kombinovanej štandardnej) neistoty a na základe jej výsledku prípadne previesť úpravu metodiky merania za účelom zníženia neistôt, ktoré sa na celkovej neistote najviac podieľajú.

Tam kde nevystačíme so štandardnými neistotami je potrebné použiť ich *rozšírenie* pomocou koeficientu rozšírenia k_{r}. Pôvodne stanovená smerodajná odchýlka (teda aj štandardná neistota) predstavuje napr. pri najčastejšie používanom *normálnom rozdelení* interval určený s pravdepodobnosťou asi 68 %. Podobne je to aj pri iných typoch (zákonoch) rozdelenia pravdepodobnosti. Aby sme dosiahli väčší interval pokrytia, blížiaceho sa k 100 %, je potrebné rozšíriť štandardnú neistotu koeficientom rozšírenia k_{r}, ktorého význam je v podstate zhodný s významom kvantilov pri Gaussovom (normálnom) rozdelení, kde $k_{\mathrm{r}} = 2$ pre rozšírenie na 95%-nú a $k_{\mathrm{r}} = 3$ pre rozšírenie na 99,7%-nú pravdepodobnosť a pod. Rozšírená neistota je potom vyjadrená vzťahom

$$U_f = k_{\mathrm{r}} \cdot u_{\mathrm{C}f}, \tag{19}$$

kde U_f je rozšírená neistota, k_{r} je koeficient rozšírenia a $u_{\mathrm{C}f}$ je kombinovaná neistota. *Výsledok merania* sa potom vyjadrí v tvare

$$f = \overline{f} \pm U_f \tag{20}$$

a znamená, že najlepším odhadom meranej veličiny je f a že interval od $f - U_f$ do $f + U_f$ je interval, od ktorého je možné očakávať, že obkopuje veľkú časť hodnôt, ktoré môžu byť priradené (prisúdené) výstupnej veličine f.

Zdroje neistôt

Ako zdroje neistôt možeme označiť všetky javy, ktoré nejakým spôsobom môžu ovplyvňovať neurčitosť jednoznačného stanovenia výsledku merania a tým odďaľujú (posúvajú) nameranú hodnotu od skutočnej hodnoty. Veľký vplyv na výsledok má aj tá skutočnosť akú meraciu metódu používame, priamu alebo nepriamu. Na neistoty tiež vplýva výber meracích prístrojov (analogových alebo číslicových), vzorkovačov, použitie rôznych filtrov, iných prostriedkov a zariadení v celej trase prenosu a úprave meraného signálu. K neistotám výrazne prispievajú rušivé vplyvy prostredia v najširšom slova zmysle. Na tomto mieste spomenieme najčastejšie sa vyskytujúce zdroje neistôt:

- nevhodný výber prístroja (rozlišovacia schopnosť a pod.),
- neúplná alebo nedokonalaá definícia meranej veličiny alebo jej realizácia,
- nevhodný, resp. nereprezentatívny výber vzoriek merania,
- nevhodný postup pri meraní,
- zjednodušenie alebo nesprávne zaokrúhlenie konštánt a prevzatých hodnôt,
- linearizácia, aproximácia, interpolácia alebo extrapolácia pri vyhodnocovaní,
- nekompenzované alebo neznáme vplyvy prostredia,
- nedodržanie zhodných podmienok pri opakovaných meraniach,
- subjektívne vplyvy obsluhy (experimentátora),

- nepresnosť etalónov a referenčných zdrojov alebo materiálov.

Niektoré zo zdrojov sa prejavujú významne alebo výhradne v neistotách vyhodnocovaných metódou typu A, iné zase pri požití metódy typu B. Mnohé zdroje môžu byť príčinou oboch skupín neistôt a preto vzniká nebezpečenstvo v podobe zabudnutia (vynechania) jednej zo zložiek, čo môže mať výrazne skreslujúci účinok. Situácia sa komplikuje, keď na meranie niekoľkých vstupných veličín používame rovnaký merací prístroj alebo keď sú medzi vstupnými parametrami iné kovariačné väzby. Výsledná neistota je potom podstatne väčšia a celá metodika spracovania nameraných údajov je primerane zložitejšia.

Príklady stanovenia neistoty

Čísla, ktoré vyjadrujú výsledok merania v tvare $f = \overline{f} \pm U_f$ (alebo $f = \overline{f} \pm u_{Cf}$) sú získané výpočtom a majú obyčajne toľko desatinných miest, koľko zobrazí použité výpočtové zariadenie (počítač, kalkulačka a pod.) V zobrazenom čísle sa číslice s výnimkou núl na začiatku zobrazenej hodnoty označujú ako *platné číslice*, napr. číslo 0,003 001 40 má šesť platných číslic (miest). *Výsledná neistota merania* sa zaokruhľuje najviac na *dve platné číslice*. Takýto postup vnáša do výslednej číselnej hodnoty intervalu neistoty hodnoty zaokrúhľovaciu chybu nanajvýš 0,5 %. Napríklad výsledkom výpočtu je číslo 0,004 323 4, zaokrúhlením získame hodnotu 0,004 3, alebo výsledkom výpočtu je číslo 0,004 023 4, zaokrúhlením získame hodnotu 0,004 0 (keď je druhou platnou číslicou nula, treba ju vo výsledku uviesť). Z formátu čísla vyjadrujúceho interval neistoty U_f vyplýva aj formát čísla $\overline{f}$, ktoré nemá význam uvádzať v nižšom ráde ako je rád poslednej platnej číslice neistoty. Uvedieme príklady:

1. Výpočtom sme získali tieto nezaokrúhlené čísla:

$$\begin{aligned} \overline{f} &= 38{,}395\,799\,\mathrm{cm}^2 \\ U_f &= 0{,}155\,911\,8\,\mathrm{cm}^2 \end{aligned}$$

2. Po zaokrúhlení ich zapíšeme v tvare:

$$\begin{aligned} \overline{f} &= 38{,}40\,\mathrm{cm}^2 \\ U_f &= 0{,}16\,\mathrm{cm}^2 \end{aligned}$$

3. Výsledok merania uvedieme v tvare:

$$f = (38{,}40 \pm 0{,}16)\,\mathrm{cm}^2$$

4. Pri zaokrúhlení na jedno (platné) desatinné miesto:

$$f = (38{,}4 \pm 0{,}2)\,\mathrm{cm}^2$$

5. Ďalšie možnosti zápisu výsledkov meraní:

$$\begin{aligned} I &= (8{,}37 \pm 0{,}24)\,10^{-3}\,\mathrm{A} \\ p &= (1{,}2017 \pm 0{,}0024)\,\mathrm{Pa} \\ \lambda &= 2{,}037(4)\,\mathrm{nm} \end{aligned}$$

Poslednú formu zápisu výsledku merania používajú niektoré odborné časopisy a nahradzuje klasický tvar zápisu $\lambda = (2{,}037 \pm 0{,}004)\,\mathrm{nm}$.

Číslicový merací prístroj[34]

Postup odhadu neistoty výsledku merania (typ B) je pre daný typ meracieho prístroja obyčajne stanovený výrobcom. Keď nemáme k dispozícii tento postup alebo nie je stanovený inak, je neistota výsledku principiálne určená diskrétnym charakterom číselného údaja na displeji prístroja. Najjemnejší krok delenia δ daného rozsahu meracieho prístroja X_{mr} je určený zmenou údaja na poslednom digite o hodnotu ± 1. Napríklad pre prístroj s päť digitovým displejom je $\delta = 10^{-5}\,X_{\mathrm{mr}}$ a meraná veličina sa s istotou nachádza v intervale hodnoty zobrazenej na

[34] Dovolená chyba číslicového voltmetra (ampérmetra) sa udáva súčtom relatívnej chyby v percentách δ_{mh} z meranej hodnoty a relatívnej chyby δ_{mr} v percentách, vzťahuje sa na najväčšiu hodnotu meracieho rozsahu prístroja (podrobnejšie pozri časť 5.5 *Chyby elektrických meracích prístrojov*).

displeji plus δ. Rozdelenie pravdepodobnosti výskytu meranej veličiny v tomto intervale sa obyčajne považuje za rovnomerné. Potom sa stredná hodnota meranej veličiny odhaduje hodnotou zobrazenou na displeji plus $\delta/2$ a príslušná neistota výsledku sa odhadne štandardnou odchýlkou rovnomerného rozdelenia

$$u_{\mathrm{B}} = \frac{\delta}{\sqrt{12}} = \frac{\delta}{2\sqrt{3}}. \tag{21}$$

Jedným zo zdrojov neistoty je rozlíšiteľnosť poslednej platnej číslice. Napriek tomu, že sa pri opakovanom meraní údaj na displeji nemení nie je neistota merania nulová. Pri odhade neistoty sa používa *model rovnomerného rozdelenia pravdepodobnosti* v intervale, ktorý je vymedzený rozlišovaciou schopnosťou δ daného prístroja podľa vzťahu (21).

Príklad A:

Číslicový voltmeter opakovane ukazuje na displeji napätie $U = 14{,}12$ V pri rozlíšení 10 mV (presnosť 10 mV), môžeme teda predpokladať, že $\delta^{(1)} = 0{,}01$ V a neistota je rovná

$$\begin{aligned} u_{\mathrm{B}}^{(1)} &= \frac{0{,}01}{2\sqrt{3}} = 0{,}002\,886\,75\ \mathrm{V} = \\ &= 0{,}002\,9\ \mathrm{V} \approx 0{,}003\ \mathrm{V}. \end{aligned} \tag{22}$$

Údaj v technickej dokumentácii voltmetra nás však informuje, že na rozsahu 20 V pri rozlíšení 10 mV (1 digit) má voltmeter presnosť $(0{,}3\,\%$ meranej hodnoty $+ 1$ digit$)$. Potom

$$\begin{aligned} \delta^{(2)} &= \left(\frac{14{,}12}{100}\,0{,}3\,\% + 0{,}01\right) = \\ &= (0{,}042\,36 + 0{,}01) = 0{,}052\,36\ \mathrm{V}. \end{aligned} \tag{23}$$

Príslušná zložka neistoty bude

$$\begin{aligned} u_{\mathrm{B}} &= \frac{\delta^{(2)}}{2\sqrt{3}} = \frac{0{,}052\,36}{2\sqrt{3}} = \\ &= 0{,}015\,115\,03\ \mathrm{V} \approx 0{,}015\ \mathrm{V}, \end{aligned} \tag{24}$$

čo je ale hodnota $5\times$ väčšia, ako z predchádzajúceho výpočtu.

Príklad B:

Pri meraní číslicovým ampérmetrom s piatimi digitmi na rozsahu 10 A je údaj na displeji $I = 0{,}003\,5$ A. Meraný prúd sa teda nachádza v intervale $\delta I = (0{,}003\,5 \div 0{,}003\,6)$ A. Strednú hodnotu meraného prúdu odhadneme ako $\overline{I} = 0{,}003\,55$ A. Pre neistotu merania v prípade použitia *modelu rovnomerného rozdelenia* výskytu meranej veličiny dostávame

$$\begin{aligned} u_{\mathrm{B}I} &= \frac{10^{-4}}{2\sqrt{3}} = 28{,}867\,5 \cdot 10^{-6}\ \mathrm{A} = \\ &= 0{,}000\,029\ \mathrm{A} \end{aligned} \tag{25}$$

a pre *model normálneho rozdelenia* výskytu meranej veličiny je vysledok neistoty rovný

$$\begin{aligned} u_{\mathrm{B}I} &= \frac{10^{-4}}{6} = 16{,}666\,67 \cdot 10^{-6}\ \mathrm{A} = \\ &= 0{,}000\,017\ \mathrm{A}. \end{aligned} \tag{26}$$

Výsledok merania pre model rovnomerného rozdelenia potom zapíšeme v tvare

$$I = (0{,}003\,550 \pm 0{,}000\,029)\ \mathrm{A}. \tag{27}$$

Posuvné meradlo

V prípade posuvného meradla s najjemnejším delením stupnice 0,1 mm, keď predpokladáme rovnomerné rozdelenie pravdepodobnosti, je interval neistoty výsledku merania podľa predchádzajúcej zásady (v časti *Číslicový merací prístroj*) rovný $\delta = 0{,}2$ mm a podľa vzťahu (21)

$$u_{\mathrm{B}} = \frac{0{,}1}{\sqrt{3}} = 0{,}057\,735\ \mathrm{mm} = 0{,}058\ \mathrm{mm}. \tag{28}$$

Jednoduché meranie teploty

Bežným liehovým teplomerom meriame teplotu kvapaliny v nádobe, pričom predpokladáme, že na teplomer nepôsobia iné, ako v danom prípade zanedbateľné vplyvy (sálanie, premenlivá teplota okolia, zmeny prúdenia vzduchu v miestnosti a pod.). Presnosť merania teplomerom je daná ako chyba odčítania teploty s veľkosťou jedného dielika stupnice, čiže $\pm 1\ ^{\circ}\mathrm{C}$. Výraz presnosť tu chápeme „klasicky“ prostredníctvom

chyby, teda nie ako neistotu. Meranie prevádzame opakovane v rôznych miestach nádoby tak, aby bolo možné určiť priemernú teplotu kvapaliny. Predpokladom merania je aj to, aby teplotné pole v meranom priestore bolo homogénne. Keď je táto podmienka splnená nemáme dôvod uvažovať o ďalších prídavných korekciách a môžeme použiť takýto postup.

Opakovaným meraním, pri dostatočnej dobe ustálenia údaja teplomera (vylúčime prípadnú dynamickú chybu) sa získa minimálne potrebných desať meraní. Odhadom priemernej teploty $\bar{t}$ je aritmetický priemer zo všetkých desatich hodnôt $\bar{t} = 35{,}4\,^{\circ}\mathrm{C}$. *Štandardná neistota typu A* je reprezentovaná smerodajnou odchýlkou súboru nameraných hodnôt od aritmetického priemeru

$$u_{\mathrm{A}}(t) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_i - \bar{t})^2} = 0{,}360\,^{\circ}\mathrm{C}. \quad (29)$$

Štandardná neistota typu B má pri danom usporiadaní merania jediný zdroj, ktorým je chyba odčítania s hodnotou $\pm 1\,^{\circ}\mathrm{C}$. Oprávnene môžeme predpokladať rovnomerné rozdelenie pravdepodobnosti chyby teplomera, čiže

$$u_{\mathrm{B}}(t) = \frac{1}{\sqrt{3}} = 0{,}578\,^{\circ}\mathrm{C}. \quad (30)$$

Kombinovanú štandardnú neistotu získame zlúčením oboch zložiek

$$\begin{aligned} u_{\mathrm{C}}(t) &= \sqrt{u_{\mathrm{A}}^2(t) + u_{\mathrm{B}}^2(t)} = \\ &= \sqrt{0{,}360^2 + 0{,}578^2} = 0{,}681\,^{\circ}\mathrm{C}. \end{aligned} \quad (31)$$

Výsledok merania môžeme prezentovať pomocou *rozšírenej neistoty* s koeficientom rozšírenia $k_{\mathrm{r}} = 2$ (skutočná priemerná teplota sa nachádza v intervale neistoty s asi 95%-nou pravdepodobnosťou), takže zápis po zaokrúhlení na dve platné miesta bude mať tvar

$$t = (35{,}40 \pm 1{,}36)\,^{\circ}\mathrm{C}. \quad (32)$$

V tabuľke 1 uvádzame orientačné hodnoty odhadu chýb niektorých meracích prístrojov a zariadení.

Tabuľka 1: Hodnoty odhadu chýb niektorých meracích prístrojov

Meracie zariadenie	**Delenie stupnice**	**Chyba z_{max}**
Pásové meradlo	10 dielikov na 1 cm	1 mm
Posuvné meradlo	10 dielikov na nónius	0,1 mm
Posuvné meradlo	20 dielikov na nónius	0,05 mm
Posuvné meradlo	50 dielikov na nónius	0,02 mm
Mikrometer	50 dielikov na 0,5 mm	0,01 mm
Mechanické stopky	5 dielikov na 1 s	0,2 s
Digitálne stopky	min : sek : $\frac{\mathrm{sek}}{100}$	0,01 s
Teplomer	5, 2 alebo 1 dielik na 1 °C	$(0{,}2 \div 1)\,^{\circ}\mathrm{C}$

Tvrdím len, že v každom štúdiu prírody je len toľko vlastnej vedy, koľko je v nej matematiky.

IMMANUEL KANT

5.2 Numerické metódy spracovania výsledkov meraní

Úlohy súvisiace s vyhodnotením experimentálnych dát vo fyzikálnej a technickej praxi sa vyznačujú týmito základnými vlastnosťami:

(a) rozsah a objem spracovaných dát obyčajne nie je veľký,
(b) v dátach sa nachádzajú aj vybočujúce hodnoty merania a rôzne nehomogenity,
(c) v dátach sa zvyčajne vyskytujú nelinearity, vzájomné väzby a pod., ktoré treba identifikovať a opísať,
(d) parametre modelov majú obyčajne definovaný fyzikálny význam,
(e) často narážame na istú neurčitosť (nejasnosť, nepresnosť) pri výbere modelu na opis dát.

Pri projektovaní pokusu je experimentátor vedený snahou získať z meraní čo najviac fyzikálne zaujímavých informácií.[35] Preto experiment obyčajne prebieha za rôznych (kontrolovaných) podmienok. Zmenou istých veličín sledujeme ich vplyv na iné veličiny. Vo väčšine prípadov takto získame závislosť, o ktorej predpokladáme, že je spojitá funkčná závislosť jednej veličiny od druhej veličiny. Napr. teplotnú závislosť odporu, závislosť anódového prúdu magnetrónu od indukcie magnetického poľa, závislosť intenzity jadrového žiarenia od hrúbky absorbátora atď. Nameraním závislosti veličín práca experimentátora nekončí, naopak, nasleduje najdôležitejšia úloha a to *fyzikálne interpretovať výsledky meraní*. Pod pojmom interpretácie budeme rozumieť *odôvodnenie výsledkov*. V podstate ide o *určenie príčin*, ktoré spôsobujú daný výsledok. Experimentálna práca je takto z formálneho hľadiska „obrátenou“ úlohou k teoretickému postupu, ktorý z definovaných podmienok (príčin) predpokladá závery (následky) a tento fakt treba mať na zreteli pri spracovávaní merania. V konkrétnych prípadoch sa najčastejšie stretneme s týmito situáciami:

- Fyzikálna interpretácia meranej závislosti nie je dobre prepracovaná, tzn., že v čase konania experimentu neexistuje teoretický model, ktorý by viac-menej úspešne predpovedal tvar funkčnej závislosti. Potom je možné získané závislosti interpretovať iba kvalitatívne, resp. v jednoduchých prípadoch vysloviť hypotézu (napr. o lineárnej, resp. inej závislosti).

- Teoretický model predpovedá očakávanú závislosť, napr. $y = a + bx$. Experiment[36] lineárnu závislosť potvrdí. Treba nájsť „správne“ hodnoty parametrov, napr. a, b, ktorým môžu odpovedať ďalšie dôležité informácie. Úlohami tohto druhu sa zaoberá vyrovnávací počet. V súčasnej dobe

[35] *Metóda pozorovania dáva cenné informácie o vonkajších javoch a vzťahoch (veľkosť, tvar, časová následnosť a pod.). Poznávaciu hodnotu však stráca vtedy, keď sa pýtame na charakter vzťahov alebo na príčinu javov. Hlbšie poznanie skutočnosti umožňuje experimentálna metóda, ktorej použitie znamená cieľavedomí zásah do pôvodného stavu zámernou zmenou, ktorá je exaktne sledovaná za účelom získania nových vedeckých faktov. Experimentálna metóda identifikácie zámerne vyvoláva zmeny v skúmaných objektoch a na to používa najrôznejšie techniky. Cavier tento rozdiel medzi pozorovateľom a experimentátorom vyjadril takto: „Pozorovateľ prírode načúva, experimentátor ju vypočúva.“ Dáta získané experimentom sa stanú odpoveďou na experimentátorovu otázku len po ich logickom spracovaní, ktoré najčastejšie pozostáva z matematického vyhodnotenia a poznávania, zo zovšeobecnenia zistených faktov.*

[36] Experiment môžeme rozdeliť na časti, ktoré sú do istej miery samostatné, voláme ich *pokusy*.

sa široko využíva *metóda najmenších štvorcov*.[37] Za správne hodnoty sa považujú také hodnoty parametrov, ktoré dávajú *najmenší súčet druhých mocnín odchýlok* medzi nameranými a teoreticky predpovedanými hodnotami. Uvedieme hlavné črty metódy.

Majme nameranú funkčnú závislosť $f_i = f(x_i)$ v bodoch $i = 1, 2, \ldots, n$. Teoretický model predpokladá závislosť $y = F(x, p_1, p_2, \ldots, p_k)$, kde $p_1, p_2, \ldots, p_k$ sú parametre, ktoré sa nedajú vypočítať v rámci tohto modelu (čím menej parametrov, tým je model hodnotnejší). Odchýlky modelovej F^* a experimentálej funkcie f_i, vypočítané v nameraných bodoch, označíme e_i

$$e_i = F^*(x_i, p_1^*, p_2^*, \ldots, p_k^*) - f_i. \tag{33}$$

Vzhľadom na to, že považujeme kladné odchýlky za rovnako významné ako záporné, uvažujeme druhú mocninu e_i.[38] Ďalej označíme

$$\Phi = \sum_{i=1}^{n} {e_i}^2. \tag{34}$$

Úlohou je nájsť také odhady $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_k$ parametrov $p_1, p_2, \ldots, p_k$, pre ktoré funkcia Φ (označovaná tiež ako *účelová* alebo *kriteriálna*) nadobúda minimum. Aby mala táto požiadavka zmysel, musí byť splnených niekoľko, nie práve samozrejmých predpokladov, o ktorých sa musíme pred začatím experimentu presvedčiť, pozri napr. (PETROVIČ A KOL., 1989, I., str. 74):

1. chyba nezávisle premennej x_i je zanedbateľne malá vzhľadom na chybu závisle premennej f_i,
2. chyba merania premennej f_i je náhodná veličina z normálne rozdeleného súboru, ktorý má nulovú strednú hodnotu a konštantný rozptyl v celej oblasti merania.[39]

Nutnou podmienkou pre minimum je potom splnenie rovnice

$$\frac{\partial \Phi}{\partial p_j^*} = 2 \sum_{i=1}^{n} e_i \frac{\partial e_i}{\partial p_i^*} = 2 \sum_{i=1}^{n} e_i \frac{\partial F^*\left(x_i, p_1^*, \ldots, p_k^*\right)}{\partial p_j^*} = 0\,,\ j = 1, 2, \ldots, k. \tag{35}$$

Túto sústavu je možné explicitne riešiť v niektorých špeciálnych prípadoch. Všeobecne treba používať vybrané numerické metódy (KAUKIČ, 2006; PIRČ A BUŠA, 2002). Preberieme si tie funkčné závislosti, ktoré budeme potrebovať pri vyhodnocovaní laboratórnych záznamov.

5.2.1 Lineárna závislosť $y = a + bx$ a $y = ax$

Podľa vzťahu (35) máme dva parametre $p_1 = a,\ p_2 = b$

$$e_i = a + bx_i - f_i. \tag{36}$$

[37] Metódu najmenších štvorcov, ako výpočtovú procedúru opísal *Adrien-Marie Legendre* r. 1805 v práci *Nouvelles méthodes pour la détermination des orbites des comètes*. On navrhol aj názov tejto metódy. Prvý, kto spojil metódu najmenších štvorcov s teóriou pravdepodobnosti bol *Carl Friedrich Gauss* r. 1809 v práci *Theoria motus corporum coelestium in sectionibus conicus solem ambientium auctore*, C. F. G. 1809. Poznamenal, že túto metódu použil už roku 1795.

[38] Všeobecne sa uvažuje nejaká párna funkcia, t. j. funkcia $f(x)$ taká, že $f(x) = f(-x)$. Druhej mocnine sa dáva prednosť pred absolútnou hodnotou, lebo je to hladká funkcia.

[39] Systematické chyby ovplyvňujú experiment v rovnakom zmysle, ale vo všeobecnosti všetky merania rôznou hodnotou. Majú nenulovú strednú hodnotu a prejavujú určitú mieru vzájomnej závislosti, t. j. sú korelované. Nedodržanie predpokladu náhodnosti a nezávislosti chýb i nenulovosti ich stredných hodnôt znemožňuje použitie štatistických metód vyhodnotenia.

Z rovníc (34) a (35) dostaneme sústavu dvoch lineárnych rovníc pre neznáme a a b, ktoré môžme ľahko vyriešiť. Riešenie zapíšeme v tvare výhodnom na počítačové spracovanie. Označíme

$$s_1 = \sum_{i=1}^{n} x_i, \qquad s_2 = \sum_{i=1}^{n} f_i, \qquad s_3 = \sum_{i=1}^{n} x_i^2,$$
$$s_4 = \sum_{i=1}^{n} x_i f_i, \quad s_5 = \sum_{i=1}^{n} f_i^2, \qquad \nu = n s_3 - s_1^2.$$

Potom

$$a = \frac{s_2 s_3 - s_1 s_4}{\nu}, \quad b = \frac{n s_4 - s_1 s_2}{\nu}. \tag{37}$$

Dá sa ukázať, že pre štandardné neistoty odhadnutých parametrov a a b platia tieto vzťahy:

$$\sigma_a = \sigma_f^{ab} \sqrt{\frac{s_3}{\nu}}, \quad \sigma_b = \sigma_f^{ab} \sqrt{\frac{n}{\nu}}. \tag{38}$$

V experimentoch však nie vždy poznáme hodnotu štandardnej neistoty σ_f. Jej hodnotu môžeme získať len opakovaním merania. Pri jednom experimente však máme nameraných n hodnôt f a keď sme všetky zmerali s rovnakou chybou, môžemu ju odhadnúť z rozdielov medzi nameranými bodmi a funkciou (33) vzťahom[40]

$$\sigma_f^{ab} = \sqrt{\frac{\sum_{i=1}^{n} (a + b x_i - f_i)^2}{n-2}}. \tag{39}$$

Podobne pre závislosť typu $y = ax$ platia tieto vzťahy:

$$a = \frac{s_4}{s_3}, \quad \sigma_f^a = \sqrt{\frac{\sum_{i=1}^{n} (a x_i - f_i)^2}{n-1}}, \quad \sigma_a = \frac{\sigma_f^a}{\sqrt{s_3}}. \tag{40}$$

Odvodenie týchto vzťahov však presahuje rámec tejto práce a nie je ani jej cieľom. Čitateľ sa môže o metóde najmenších štvorcov podrobnejšie dočítať napr. v prácach autorov (LYONS, 2001; KUDRACIK, 1999; SQUIRES, 2001; PETROVIČ A KOL., 1989)

5.2.2 Polynomiálna závislosť

Parametrami sú koeficienty v polynóme k-teho stupňa

$$F^* = a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x + a_0 = \sum_{l=0}^{k} a_l x^l \,,$$
$$e_i = \sum_{l=0}^{k} a_l x_i^l - f(x_i) \,,$$
$$\frac{\partial F^*(x_i, a_o, a_1, \ldots, a_k)}{\partial a_j} = \frac{\partial}{\partial a_j} \sum_{l=0}^{k} a_l x_i^l = x_i^j$$

[40] V menovateli vzťahu je hodnota $n-2$ namiesto n z toho dôvodu, aby bol odhad nevychýlený. Ak máme namerané iba dva body priamka určená metódou najmenších štvorcov prechádza presne cez ne a reziduálny súčet štvorcov je rovný nule, vzťah predstavuje výraz typu $0/0$. Musíme mať teda namerané aspoň tri body, aby sme z rozptylu bodov okolo priamky odhadli neistotu merania. Odhad je *nevychýleny*, keď stredná hodnota odhadu sa rovná jej skutočnej hodnote (nezávisle od počtu meraní).

a podľa rovnice (35) dostaneme sústavu rovníc

$$\frac{\partial \Phi}{\partial a_j} = 2 \sum_{i=1}^{n} \left[\sum_{l=0}^{k} a_l x_i^l - f(x_i) \right] x_i^j = 0.$$

Prehodením poradia sumácie dostávame sústavu $k+1$ rovníc pre $k+1$ neznámych $a_0, \ldots, a_k$

$$\sum_{l=0}^{k} B_{jl}\, a_l = y_j, \tag{41}$$

kde

$$B_{jl} = \sum_{i=1}^{n} x_i^{l+j}, \quad y_j = \sum_{i=1}^{n} f(x_i)\, x_i^j.$$

Sústavu (41) je možné riešiť napr. *Gaussovou eliminačnou metódou*. Pre $k = 1$ dostávame lineárnu závislosť, pre ktorú je riešenie zhodné s rovnicou (37).

5.2.3 Exponenciálna závislosť

$$F^* = \alpha \mathrm{e}^{\beta x}, \quad p_1 = \alpha, \quad p_2 = \beta, \quad e_i = \alpha \mathrm{e}^{\beta x_i} - f(x_i)$$

a z rovnice (35) dostaneme

$$\frac{\partial \Phi}{\partial \alpha} = 2 \sum_{i=1}^{n} \left[\alpha \mathrm{e}^{\beta x_i} - f(x_i) \right] \mathrm{e}^{\beta x_i} = 0,$$

$$\frac{\partial \Phi}{\partial \beta} = 2 \sum_{i=1}^{n} \alpha^2 x_i\, \mathrm{e}^{2\beta x_i} - 2 \sum_{i} \alpha f(x_i)\, x_i\, \mathrm{e}^{\beta x_i} = 0,$$

čo je sústava transcendentných rovníc a na ich riešenie treba zvoliť približné numerické metódy. Aby sme sa tomu vyhli, pozmeníme úlohu a namiesto extrému účelovej funkcie Φ budeme hľadať extrém funkcie $\Phi^{(L)}$, v ktorej namiesto F^* vystupuje $\ln(F^*)$. Logaritmovaním F^* dostaneme

$$\ln(F^*) = \ln(\alpha) + \beta\, x.$$

Ak označíme $a = \ln(\alpha)$, $b = \beta$, môžeme použiť výsledky rovnice (37) pre lineárnu závislosť.

Poznámka:

Ak použijeme metódu najmenších štvorcov na takto transformovanú nelineárnu funkciu, hľadané parametre nezodpovedajú minimálnemu súčtu štvorcov odchýlok $\sum_{i=1}^{n} [\ln f_i - \ln F^*(x_i)]^2$, *pretože transformácia do súradníc prirodzeného logaritmu ovplyvňuje odchýlky rozdielne v rôznych oblastiach pozdĺž krivky a tiež rozdielne ovplyvní pozitívne a negatívne chyby v tých istých bodoch krivky; preto treba problém riešiť ako sústavu nelineárnych rovníc. Podľa Brunovskej (1990, str. 50) však neexistuje všeobecné pravidlo pre nelineárne regresie, podľa ktorého by bolo možné dať prednosť jednej účelovej funkcii pred druhou. Ak rozptyl údajov nie je veľký, tento rozdiel nie je významný. Odhad cez transformáciu možno ešte zlepšiť zavedením štatistickej váhy* $w_i = \left(\sigma^2_{\ln F^*}/\sigma^2_f\right)_i^{-1}$ *do účelovej funkcie (34), ktorá potom nadobudne tvar* $\sum_{i=1}^{n} w_i\, [\ln f_i - \ln F^*(x_i)]^2 = \min$.

Záverom treba zdôrazniť, že funkcia F^* (tzv. modelová funkcia) musí byť fyzikálne opodstatnená. Ak sa predpokladá lineárna závislosť, nemá fyzikálne opodstatnenie vyrovnávať meranú závislosť kvadratickou funkciou, i keď môžeme očakávať „lepšiu“ zhodu v zmysle najmenších štvorcov. Aproximácia experimentálnych dát inými fyzikálne neodôvodnenými funkciami má význam iba z hľadiska vhodnejšieho uchovania informácie o experimentálnych dátach a z hľadiska niektorých numerických operácií, napr. interpolácie a extrapolácie.

5.2.4 χ^2 test kvality fitovania

Pri opise metódy najmenších štvorcov sme mlčky obišli otázku vplyvu chýb merania na charakter a parametre (koeficienty) určovanej analytickej závislosti. Krátko sa zmienime o tom, akú novú informáciu môžeme získať, keď vezmeme do úvahy štandardné (stredné kvadratické) chyby experimentálnych dát. Predpokladajme, že máme k dispozíci n dvojíc meraných hodnôt (x_i, f_i), pričom chyba veličiny x_i je zanedbateľne malá a chyba merania veličiny f_i je známa, rovná sa σ_{fi}.

Optimálny postup pre dáta s normálnou distribúciou šumu je taký, ktorý hľadá minimum kriteriálnej (účelovej) funkcie metódy najmenších štvorcov, t. j. váhovanej sumy štvorcov rezíduí alebo aspoň prostej reziduálnej sumy

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{e_i}{\sigma_{fi}} \right)^2 = \sum_{i=1}^{n} \left(\frac{F^*(x_i, \hat{p}_1, \ldots, \hat{p}_k) - f_i}{\sigma_{fi}} \right)^2, \tag{42}$$

kde σ_{fi} je chyba merania veličiny f v bode i. Tento výraz sa volá funkcia χ *kvadrát* (chí kvadrát, χ^2) a metóda najmenších štvorcov, v ktorej sa aproximácia dát vykonáva so započítaním chýb meraní sa volá *metóda* χ *kvadrát*. Poznamenajme, že χ^2 je opäť funkcia diskrétnej premennej a príspevok jednotlivých členov (rozdielov teoretických a meraných dát) je tým významnejší, čím presnejšie (s menšou hodnotou σ_{fi}) je zmeraná hodnota f_i. Je teda zrejmé, že keď nepoznáme chyby f_i, nemôžeme vysloviť žiaden záver ohľadne kvality fitovania.

Vo všetkých prípadoch χ^2 slúži, ako indikátor zhody medzi experimentálnymi a očakávanými hodnotami nejakej premennej. Pri dobrej zhode bude χ^2 stupňa n, pri zlej zhode bude omnoho väčšie ako n. Kritérium môžeme použiť len v tom prípade, keď poznáme očakávané hodnoty a štandardnú chybu. Pozrime sa na túto úlohu trochu podrobnejšie.

Pre jednoduchosť budeme predpokladať, že všetky merania sú zaťažené rovnakými štandardnými chybami σ_{fi}. Potom $\sigma_{fi} = \sigma_f$ (pozri vzťah 39) a v menovateľoch sumy výrazu (42) bude vystupovať pre všetky merania rovnaké σ_f. Po derivovaní vzťahu (42) za účelom hľadania koeficientov polynómu dostaneme tú istú sústavu rovníc, ako v metóde najmenších štvorcov. Keďže do sústavy rovníc vstupujú tie isté experimentálne dáta, potom prirodzene získame aj rovnaké parametre (koeficienty). Je teda na mieste otázka, akú novú informáciu nám dá použitie známych hodnôt σ_f, ktoré nie sú obsiahnuté vo vypočítaných parametroch. Vzťah (42) má „väčší“ fyzikálny význam, ako vzťah (34), χ^2 funkcia je bezrozmerná veličina, ktorá sa, ako vidíme, rovná sume štvorcov odchýlok experimentálnych bodov od teoretickej (optimálnej) krivky v násobkoch štandardnej chyby σ_{fi}.

Podmienka „fitovateľnosti“ dát je splnená, keď je počet k hľadaných parametrov rovný alebo menší ako počet nameraných bodov n. Predpokladáme však, že v mnohých prípadoch je splnený taký scenár experimentu, v ktorom $n \gg k$. Zdravý rozum nám hovorí, že keď má byť fitovanie dobré, rozdiely e_i by mali splňať rovnicu

$$e_i = |y_i - f(x_i)| \approx \sigma_{fi}. \tag{43}$$

Je to len „hrubá“ indikácia, ale vždy je lepšia, ako letmý prelet očami „pozdĺž krivky“.[41] Keď naše kritérium (43) dosadíme do rovnice (42) dostaneme

$$\chi^2 \approx n. \tag{44}$$

Čím viac parametrov bude mať modelová funkcia použitá na fitovanie, tým tesnejšie bude fitovaná krivka sledovať namerané dáta. Fitovanie budeme teda pokladať za dobré, keď $k = n$. Tento predpoklad nás vedie k tomu, aby sme aj s prihliadnutím na požiadavku vyslovenú v úvode tejto časti, že

[41] Garcia (2000) volá tento prístup *eye-balling* a Press (1992) *chi-by-eye*.

modelová funkcia $F^*(x_i, p_k)$ je tým hodnotnejšia, čím má menej parametrov, prijali praktické pravidlo pre *dobrý výsledok fitovania* v tvare

$$\chi^2 \approx n - k, \tag{45}$$

ktoré bude platiť pre jednu sériu meraní. Keby sme mohli zopakovať naše merania nekonečne mnoho ráz a po každej sérii vypočítať χ^2, potom by sa jej stredná hodnota rovnala $n - k$.

Najčastejšie môžu nastať dva prípady:

(a) keď bude $\chi^2 \gg n - k$ nemôžeme vybranú modelovú funkciu $F^*(x_i, p_k)$ použiť na fitovanie nakoľko σ_{fi} sú pre ňu „príliš malé“,

(b) keď $\chi^2 \ll n - k$ fitovanie pokladáme za *veľmi dobré*, môžeme sa domnievať, že σ_{fi} sú pre danú modelovú funkciu „dostatočne veľké“.

Vzťah (45) môžeme previesť na vhodnejší tvar, keď zavedieme *redukovanú hodnotu* χ^2 (alebo χ^2 na stupeň voľnosti), pre ktorú platí

$$\widetilde{\chi}^2 = \frac{\chi^2}{n-k}. \tag{46}$$

Keďže podľa (45) očakávame hodnotu $n - k$, má byť splnená rovnosť

$$\widetilde{\chi}^2 = 1. \tag{47}$$

Naše predchádzajúce kritériá (a) a (b) teda môžeme vysloviť v takomto znení : keď získame pre $\widetilde{\chi}^2$ hodnotu rovnú rádovo jednotke alebo menej ako jedna, potom nemáme dôvod pochybovať o našom modeli; ale keď je získaný výsledok oveľa väčší ako jedna, potom je nepravdepodobné, že náš model je správny.

Na rozdiel od predchádzajúceho prístupu môžeme použiť χ^2 štatistiku s prihliadnutím na štatistické vlastnosti dát, ktoré budeme aproximovať danou krivkou.[42] V krátkosti uvedieme základné myšlienky tohoto prístupu.

Ako sme to už spomenuli v úvode tejto časti, predpokladáme, že chyba meranej premennej f je náhodná veličina z normálneho rozdelenia súboru. Za tohto predpokladu sú aj jednotlivé e_i nezávislé s normálnym rozdelením, s nulovou strednou hodnotou a disperziou e_i^2. Potom sa suma štvorcov zapísaná v tvare (42) riadi distribučným zákonom známym pod menom χ^2 *rozdelenie* (rozdelenie chí na druhú) s m stupňom voľnosti. Pod stupňom voľnosti rozumieme počet nameraných bodov n znížený o počet parametrov k (voľných koeficientov): $m = n - k$.[43]

Integrál typu

$$P\left(\chi^2 \geqq \chi_0^2\right) = \int_{\chi_0^2}^{\infty} f_m(x)\,\mathrm{d}x, \tag{48}$$

[42] Ako príklad nám môže poslúžiť meranie elektrického výkonu na rezistore. Prúd I prechadzajúci rezistorom sa riadi normálnym rozdelením, ale výkon P sa nemôže riadiť normálnym rozdelením pretože normálne rozdelenie pripúšťa výskyt akejkoľvek reálnej (teda aj zápornej) hodnoty náhodnej premennej. Výkon elektrického prúdu musí mať také rozdelenie, v ktorom platí $f(x) = 0$ pre $x < 0$. V tomto prípade ide o rozdelenie odvodené z rozdelenia χ^2. Takéto rozdelenie má veličina, ktorá je súčtom n kvadrátov nezávislých premenných so štandardným normálnym rozdelením.

[43] Keď sa chyby meraných hodnôt f_i neriadia normálnym rozdelením, úloha sa stáva ešte zložitejšou. Na jej riešenie sa používa metóda, ktorá sa volá *Metóda najväčšej hodnovernosti*.

alebo jednoducho $P(\chi^2)$, kde $f_m(\chi^2 = x)$ je hustota rozdelenia pravdepodobnosti pre rôzne stupne voľnosti, dovoľuje vypočítať kritické χ_0^2 pre úroveň $P(\chi^2 \geqq \chi_0^2)$. Tieto hodnoty sú často vo forme tabuliek súčasťou príručiek a učebníc štatistiky, alebo sú dostupné ako súčasť štatistického softvéru, napr. program R. Na objasnenie uvedieme príklad, ako použiť tabuľku $P(\chi^2)$. Predpokladáme, že máme súbor 20 meraní. Experimentálne dáta zamýšľame interpretovať lineárnou závislosťou $y = a + bx$, pre ktorú vyčíslime parametre a a b. V tomto prípade sa počet stupňov voľnosti rovná $m = 20 - 2 = 18$. Ďalej predpokladajme, že výpočtom podľa vzťahu (42) sme získali hodnotu $\chi^2 = 9$. Z tabuľky 2 pre $P(\chi^2)$ zistíme, že pri $m = 18$ stupňoch voľnosti je pravdepodobnosť získať $\chi^2 \geqq 9$ rovná $\sim 95\,\%$. Odchýlka nameraných údajov od očakávanej lineárnej závislosti je v tomto prípade nepodstatná. Keby sme získali výsledok $\chi^2 = 28$, z tabuľky zistíme, že takúto a väčšiu hodnotu môžeme očakávať v asi 5 % prípadov. Model lineárnej závislosti nemusíme zamietnuť, ale môžeme o ňom pochybovať. Prirodzene za takýchto okolností zopakujeme experiment, aby sme získali nové dáta alebo použijeme inú modelovú funkciu. V prípade, keď je $\chi^2 \geqq 42$ (pravdepodobnosť $\approx 0{,}1\,\%$) potvrdí sa, že preverovaná hypotéza je isto nesprávna (dané body nemôžeme aproximovať lineárnou závislosťou). Na podrobnejšie oboznámenie sa s danou problematikou odporúčame čitateľovi špecializovanú literatúru, napr. (PRESS ET AL., 1992; RIEČANOVÁ A KOL., 1987; ZVÁRA A ŠTĚPÁN, 2001).

Tabuľka 2: Niektoré kritické hodnoty rozdelenia χ^2. Uvedené sú hodnoty pravdepodobnosti P pre $\chi^2 \geqq \chi_P^2$ pri m stupňoch voľnosti

	P					
m	0,99	0,98	0,95	0,9	0,05	0,001
4	0,3	0,4	0,7	1,1	9,5	18,5
5	0,6	0,8	1,1	1,6	11,1	20,5
6	0,9	1,1	1,6	2,2	12,6	22,5
7	1,2	1,6	2,2	2,8	14,1	24,3
8	1,6	2,0	2,7	3,5	15,5	26,1
9	2,1	2,5	3,3	4,2	16,9	27,9
10	2,6	3,1	3,9	4,9	18,3	29,6
11	3,1	3,6	4,6	5,6	19,7	31,3
12	3,6	4,2	5,2	6,3	21,0	32,9
13	4,1	4,8	5,9	7,0	22,4	34,5
14	4,7	5,4	6,6	7,8	23,7	36,1
15	5,2	6,0	7,3	8,5	25,0	37,7
16	5,8	6,6	8,0	9,3	26,3	39,2
17	6,4	7,3	8,7	10,1	27,6	40,8
18	7,0	7,9	9,4	10,9	28,9	42,3
19	7,6	8,6	10,1	11,6	30,1	43,8

pokračovanie tabuľky na ďalšej strane

(pokračovanie tabuľky z predošlej strany)

	P					
m	0,99	0,98	0,95	0,9	0,05	0,001
20	8,3	9,2	10,8	12,4	31,4	45,3
21	8,9	9,9	11,6	13,2	32,7	46,8
22	9,5	10,6	12,3	14,0	33,9	48,3
23	10,2	11,3	13,1	14,8	35,2	49,7
24	10,9	12,0	13,8	15,7	36,4	51,2

5.2.5 Interpolácia a extrapolácia

Interpolácia

Meraním určíme konečný počet hodnôt $x_1, x_2, \ldots, x_n$ a im prislúchajúce $f(x_1), f(x_2), \ldots, f(x_n)$. Predpokladajme, že $x_1 < x_2 < \ldots < x_n$. Často nás zaujíma hodnota veličiny f pre argument x, ktorý sa nezhoduje so žiadnou z nameraných hodnôt a leží v intervale $x_1 < x < x_n$. Hodnotu funkcie f pre argument x odhadneme interpoláciou. Z formálneho hľadiska experiment poskytuje informácie iba o hodnotách funkcie v konečnom počte bodov a o hodnote funkcie v bode x, kde sme meranie nevykonali, nemôžeme tvrdiť nič, ak nemáme nejaké ďalšie informácie. Tým, že cez body $(x_1, f_1), (x_2, f_2), \ldots, (x_n, f_n)$ „preložíme krivku“, nahradíme konečné postupnosti (spojitou) funkciou. Cez namerané body môže prechádzať veľmi veľa rôznych funkcií. Ak z teórie poznáme funkciu, ktorá má prechádzať cez namerané body, postupujeme podľa predchádzajúcej kapitoly. V opačnom prípade môžeme funkčnú závislosť medzi meranými bodmi nahradiť jednoduchými funkciami, najčastejšie lineárnou, kvadratickou, zriedkavejšie polynómom vyššieho stupňa. Hovoríme o lineárnej, kvadratickej alebo polynomiálnej interpolácii. Pri lineárnej interpolácii dostaneme „lomenú“ spojitú funkciu, ktorá však nemá derivácie práve v meraných bodoch. Pri kvadratickej interpolácii môžeme dostať „hladšiu“ krivku, pri kubickej interpolácii a po častiach kubickej interpolácii (napr. *kubické splajny*) môžeme dosiahnuť spojitosť derivácie atď. Je zrejmé, že čím hustejšie budú body namerané, tým menej sa budú líšiť hodnoty získané interpoláciou rôznymi funkciami.

Naznačíme postup pri interpolácii polynómom. Predpokladajme, že cez $k+1$ nameraných bodov prechádza polynóm k-teho stupňa, t. j., že platí

$$f(x_i) = \sum_{j=0}^{k} a_j x_i^j, \quad i = 1, 2, \ldots, k+1. \tag{49}$$

Dosadením známych hodnôt x_i a f_i dostávame $k+1$ lineárnych rovníc pre $k+1$ neznámych $a_0, a_1, \ldots$ $\ldots, a_k$. Vyriešením sústavy týchto rovníc dostaneme koeficienty $a_0, \ldots, a_k$ a môžeme vypočítať hodnotu funkcie $f(x)$ v ľubovoľnom bode, ktorý leží mimo bodov $x_1, \ldots, x_{k+1}$

$$f(x) = \sum_{j=0}^{k} a_j\, x^i, \quad x \in (x_1, x_k),$$

pre $k = 1$ dostávame lineárnu interpoláciu, pre $k = 2$ kvadratickú, atď. Samozrejme, na výpočet koeficientov $a_0, a_1, \ldots, a_k$ vyberieme tie experimentálne body, ktoré ležia v najbližšom okolí bodu x. Počet meraní n je obvykle väčší ako k.

Na prvý pohľad by sa zdalo, že zvyšovaním stupňa polynómu k sa zvyšuje aj presnosť interpolácie. V skutočnosti merané veličiny x_i a $f(x_i)$ sú zaťažené neistotami, ktoré sa zvýrazňujú pri výpočte vysokých mocnín hodnôt x_i v (49). Z tohto dôvodu sa vo väčšine prípadov uspokojíme s interpoláciou nízkeho rádu.

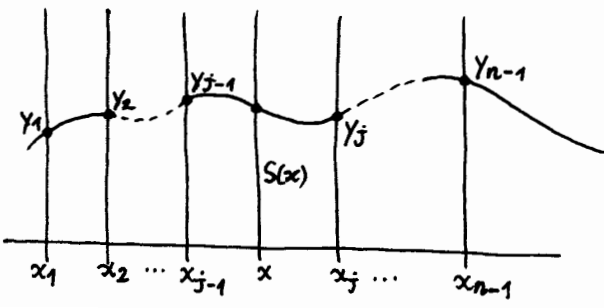
Poznámka:

Keď hodnoty $f(x_1), f(x_2), f(x_3), \ldots f(x_n)$ boli získané experimentálne a sú zaťažené určitými nezanedbateľnými chybami, nie je spravidla vhodné metódu interpolácie aplikovať. Je dokázané, že za predpokladu normálneho rozdelenia chýb meraní s nulovou strednou hodnotou v každom uzlovom bode najhodnovernejšie výsledky dosiahneme metódou najmenších štvorcov. Uvedené predpoklady sú zvyčajne pri fyzikálnych alebo technických experimentoch splnené, preto je metóda najmenších štvorcov najpoužívanejšou metódou aproximácie (vyhladenia šumu) experimentálne získaných dát.

Interpolácia nameraných dát splajn–funkciou [44]

Interpolácia patrí k bežným operáciam pri spracovaní nameraných údajov, dovolí určiť približnú hodnotu medzi dvoma susednými bodmi nameranej závislosti. Neprirodzený priebeh má takáto interpolačná funkcia v okolí nameraných bodov, v ktorých vytvára zdánlivé lokálne extrémy. Na ručné vykreslenie závislostí sa z tohto dôvodu používa *krivítko* alebo *prekladanie bodov naväzujúcimi kružnicami*.

Metóda bola nazvaná podľa pružného pravítka – *splajn* – používaného na vytváranie zaoblených (hladkých) tvarov lodí. Interpolácia splajn–funkciou, najčastejšie polynómom tretieho stupňa, má rad výhod v porovnaní s klasickou interpoláciou polynómom prechádzajúcim všetkými nameranými bodmi (najmä nižšia krivosť vzhľadom k menšiemu rádu polynómu, pri použití kubického splajnu je výsledok blízky praktickej interpolácii pomocou pružného pravítka).



Namerané hodnoty

$$f_1, f_2, \ldots, f_n \tag{50}$$

pre hodnoty nezávislej premennej

$$x_1, x_2, \ldots, x_n \tag{51}$$

chceme preložiť (opísať) optimálnou interpolačnou krivkou na každom úseku $\langle x_j, x_{j-1}\rangle$ kubickej paraboly. Predpokladajme, že $x_1 < x_2 < \ldots x_{j-1} < x_j < \ldots x_n$ a označme $h_j = x_j - x_{j-1}$, pričom hľadáme súbor polynómov tretieho stupňa, ktoré v intervale $\langle x_{j-1}, x_j\rangle$ interpolujú funkčné hodnoty a v hraničných bodoch na seba naväzujú tak, že vytvárajú hladkú krivku (čiaru). Nájdený súbor polynómov nazývame *splajn*, pri použití polynómov tretieho stupňa *kubický splajn*, ktorý má pre praktické vyhodnocovanie meraní najväčší význam. Podmienkou hladkosti krivky v hraničných bodoch je spojitosť, vrátane spojitosti derivácií až do $N-1$ rádu (N je stupeň polynómu), čiže pre kubický splajn je potrebné zaistiť spojitosť aj pre druhú deriváciu. V intervale $\langle x_{j-1}, x_j\rangle$ môžeme interpolačný polynóm zapísať v tvare

$$S_j(x) = a_0 + a_1(x - x_{j-1}) + a_2(x - x_{j-1})^2 + a_3(x - x_{j-1})^3 \tag{52}$$

[44] Problém s hodnotami vypočítanými na základe lineárnej interpolácie je, že prvá derivácia lineárne interpolovanej funkcie nie je spojitá. Pri riešení úlohy, kde záleží na tom, aby sme z interpolovaných hodnôt dostali funkciu so spojitou prvou deriváciou, musíme použiť, napr. splajnové metódy vyhladzovania.

pre $j = 2, 3, \ldots n$. Koeficienty takého polynómu v intervale (x_1, x_n)

$$a_0, a_1, \ldots, a_n \tag{53}$$

sa dajú jednoznačne určiť zo sústavy lineárnych algebraických rovníc

$$a_0 x_j^N + a_1 x_j^{N-1} + \ldots + a_{n-1} x_j + a_n = f_j \tag{54}$$

pre $j = 0, 1, 2, \ldots, n$. Determinat sústavy je Wandermondov determinant, o ktorom je známe, že je nenulový pre navzájom rôzne uzly (51). Sústava (54) má teda práve jedno riešenie (53).

Extrapolácia

Ak z meraného priebehu funkcie odhadujeme hodnotu $f(x)$ v bode x, ktorý leží mimo intervalu nameraných hodnôt, hovoríme o *extrapolácii*. Pri extrapolácii môžeme použiť numerické metódy ako pri interpolácii. Treba však mať vždy na zreteli, že pri extrapolácii musíme byť omnoho opatrnejší ako pri interpolácii, hlavne ak x je ďaleko od meraného intervalu. Pokiaľ je možné, snažíme sa nahradiť extrapoláciu interpoláciou, t. j. meraním obsiahnuť čo najväčší interval hodnôt x_i. Mimo meraného intervalu môžu mať podstatný vplyv nové fyzikálne javy, ktoré sa neprejavia v meranom intervale. Napr. pri meraní teplotnej závislosti elektrického odporu vodiča v intervale teplôt od 10 °C do 40 °C nameráme lineárnu závislosť a extrapolujeme ju do 100 °C, pričom dodatočným meraním zistíme, že vodič sa roztopil pri teplote 80 °C, takže extrapolácia nad 80 °C je neprípustná.

Numerické metódy uvedené v tejto časti sú základnými metódami, ktoré si každý experimentátor, skôr či neskôr bude nútený osvojiť a začať používať. S rozvojom výpočtovej techniky, programovacích metód a aplikačného softvéru sa rýchlo rozšírili do mnohých oblastí prírodných vied a techniky. V nasledujúcich dvoch kapitolách čitateľovi predstavíme dva programy, ktoré umožňujú ľahké a flexibilné používanie uvedených metód na numerické spracovnie experimentálnych dát a výsledky uložiť do kvalitného grafického výstupu v elektronickej alebo tlačenej podobe.

5.3 Program QtiPlot

QtiPlot je výkonný programový balík, ktorý poskytuje ako jednoduché tak aj veľmi zložité nástroje na analýzu dát a na kreslenie grafov. V tejto učebnej pomôcke sa budeme venovať opisu verzie QtiPlot 0.8.5 v prostredí OS GNU/Linux distribúcie UBUNTU 6.06. Domovská internetová stránka programu je na URL adrese `http://soft.proindependent.com/qtiplot.html` odkiaľ sa dá program stiahnuť. Na prácu v QtiPlote existujú dva druhy okien (pracovných prostredí):

- tabuľkové
- a grafické.

Tabuľkové okno zobrazuje dáta potrebné na tvorbu grafu. V *grafickom* okne je vyobrazený graf. Podľa toho, ktoré z okien je aktívne, tabuľkové alebo grafické, mení sa obsah hlavnej ponuky. V tejto kapitole opíšeme obe hlavné ponuky a bude uvedený jednoduchý príklad na tvorbu grafu. Vzhľadom na rozsah možností, ktoré poskytuje QtiPlot, budeme sa venovať len tým funkciám a ponukám QtiPlotu, ktoré sú potrebné na numerické spracovanie experimentálnych dát a ich grafickú prezentáciu.

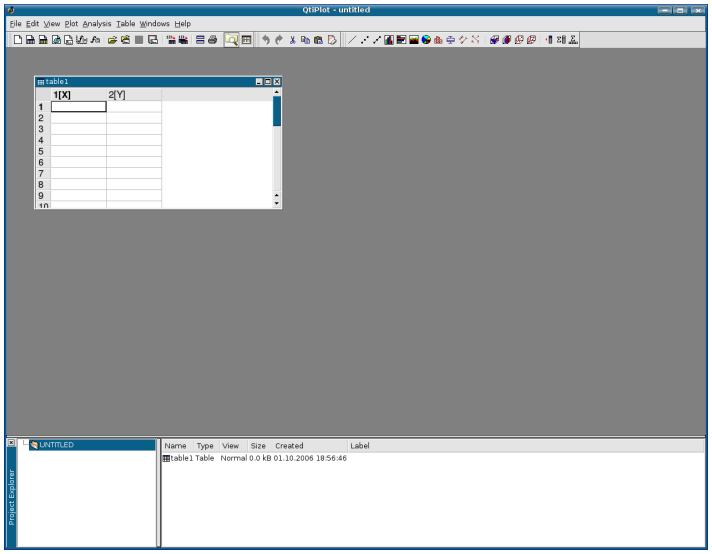
5.3.1 Ovládacie možnosti programu QtiPlot

Otvorenie QtiPlotu sa dá uskutočniť troma spôsobmi:

- kliknutie na ikonu QtiPlot na pracovnej ploche,
- Menu → Škola hrou → Mathematics → kliknutie na QtiPlot,
- z príkazového riadka X terminálu príkazom `qtiplot`.

Pri prvom čítaní tejto kapitoly môže čitateľ, ktorý sa chce rýchlo oboznámiť s používaním programu, časť 5.3.1 preskočiť a pokračovať v čítaní časťou 5.3.2 na strane 144.

Na obrazovke sa zobrazí tabuľkové okno s príslušnými ponukami a ovládacími prvkami (obrázok 1). Zatvorenie QtiPlotu sa vykoná cez záložku File a potom Quit alebo stlačením klávesov Ctrl + Q (prípadne Alt + F4).



Obrázok 1: Tabuľkové okno programu QtiPlot

Menu tabuľkového okna

Po vyvolaní QtiPlotu sa na obrazovke zobrazí okno s menom projektu UNTITLED (obrázok 1).

Ako to vidieť na obrázku, ide o tabuľkové okno. Hlavné menu obsahuje tieto položky:

File Edit View Plot Analysis Tools Window Help

V ďalšom stručne opíšeme tie položky, ktoré sú potrebné na základné zoznámenie sa s možnosťami QtiPlotu.

File

New	vytvorenie nového projektu
Open	otvorenie súboru s príponou .qti, editácia už vytvoreného projektu
Recent Projects	zoznam piatich naposledy otvorených projektov
Open image file	importovanie obrázku (jpg, bmp, gif, png a iné) do QtiPlot projektu
Import image ...	importovanie obrázkového súboru a konvertovanie intenzity obr. do maticovej tabuľky
Save Project	uloženie dokumentu pod pôvodným menom
Save Project as	uloženie dokumentu pod novým menom
Open Template	otvorenie uloženej šablóny 2D grafu, 3D grafu, tabuľky a matice
Save as Template	uloženie šablóny aktuálneho objektu
Print	vytlačenie aktívneho grafu
Print All Plots	vytlačenie všetkych grafov projektu
Export ASCII	exportovanie ASCII dátového súboru z tabuľky
Import ASCII	importovanie súboru ASCII s príponou .dat
Quit	ukončenie práce s programom QtiPlot

Edit

Undo	zruší posledný vykonaný krok
Redo	vráti posledný vykonaný krok
Cut selection	vybratie vyznačenej oblasti
Copy selection	skopírovanie vyznačenej oblasti
Paste selection	vloženie skopírovanej oblasti
Delete selection	zmazanie vyznačenej časti dát (aj celého stĺpca, ak je vyznačený)
Delete fit tables	vymazanie obsahu tabuľky s hodnotami na vykreslenie fitovanej funkcie, zmizne aj graf vytvorený z týchto dát
Clear log information	vymazanie obsahu log súboru s informáciami o výsledkoch fitovania

Plot

Line	pospájanie bodov do jednej lomenej čiary
Scatter	bodový graf
Line + Symbol	čiarový graf s vyznačenými bodmi
Special Line/Symbol	zvislé čiary alebo „schodíky“, splajnová čiara
Columns	stĺpcový graf, zvislé stĺpce
Rows	stĺpcový graf, ale stĺpce sú vodorovné
Area	graf s vyfarbenou plochou pod čiarou grafu
Pie	koláčový graf
Vectors XYXY	vytvorenie vektorového grafu, prvé dva stĺpce musia obsahovať hodnoty začiatočných súradníc vektora a posledné dva koncové súradnice vektora
Vectors XYAM	vytvorenie vektorového grafu, prvé dva stĺpce musia obsahovať hodnoty začiatočných súradníc vektora a posledné dva uhol (v radiánoch) a amplitúdu vektora
Statistical Graphs	štatistické grafy
Panel	viac grafov v jednom okne
Plot 3D	trojdimenzionálne grafy

Analysis

Statistics on Columns	štatistické vyhodnotenie dát v stĺpci
Statistics on Rows	štatistické vyhodnotenie dát v riadku
Sort Columns	usporiadanie dát v stĺpci
Sort Table	usporiadanie dát v celej tabuľke
Normalize	normalizovanie dát v stĺpci alebo vo všetkých stĺpcoch tabuľky
FFT ...	analýza dát v tabuľke rýchlou Fourierovou transformáciou
Correlate	výpočet korelácie dát dvoch vybraných stĺpcov tabuľky
Convolute	výpočet konvolúcie dát z dvoch vybraných stĺpcov tabuľky, prvý reprezentuje signál a druhý funkciu
Deconvolute	výpočet dekonvolúcie dát z dvoch vybraných stĺpcov tabuľky, prvý reprezentuje signál a druhý funkciu
Non-linear Curve Fit	nelineárna aproximácia dát vybraného stĺpca tabuľky (nesmú tvoriť priamu úmernosť)

Table

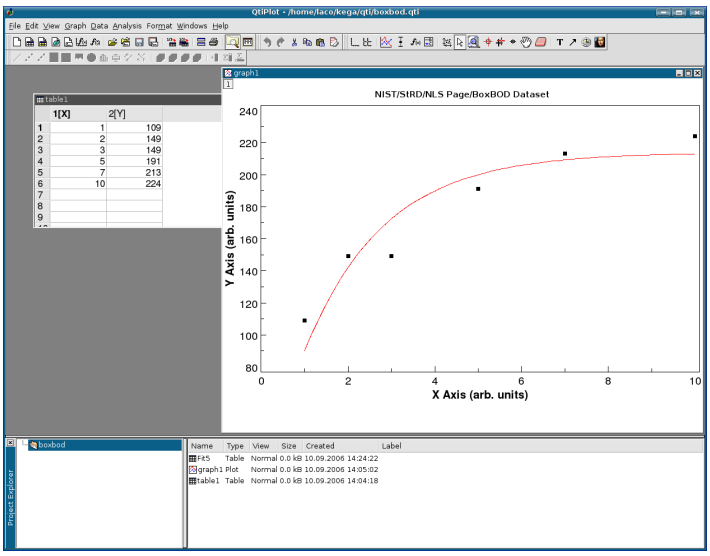
Set columns as	nastavenie dát v stĺpci (X, Y, Z, nenastavené)
Column Option ...	nastavenie vlastností stĺpca (počet riadkov,

	typ dát, formát čísiel a pod.)
Set Column Value ...	matematické operácie s dátami v stĺpci
Fill Column with	vyplnenie stĺpca tabuľky vzostupnými alebo náhodnými číslami
Add column	pridanie nového stĺpca do tabuľky
Columns ...	pridanie stĺpcov do tabuľky, nastavenie počtu stĺpcov v tabuľke
Rows ...	pridanie riadkov do tabuľky, nastavenie počtu riadkov v tabuľke
Convet to matrix	konverzia dát v tabuľke do maticového tvaru

Po konverzii tabuľky do maticového tvaru v hlavnom menu pribudnú položky 3D a Matrix. V položke Matrix nájdeme príkazy na vytvorenie transponovanej a inverznej matice, vypočet determinantu štvorcovej matice, úpravu dát vo vytvorenej maticovej tabuľke a konverziu maticovej tabuľky na XY tabuľku.

Menu grafického okna

Po vykreslení grafu z tabuľkových dát sa zmení hlavné menu a namiesto položiek Plot a Table budú Graph a Format a pribudne ešte položka Data (obrázok 2).



Obrázok 2: Grafické okno programu QtiPlot

V ďalšej časti uvedieme ponuky, ktoré budeme potrebovať na vytvorenie grafu a numerickú analýzu dát.

Graph

Add/Remove Curve ...	pridanie/odobratie krivky do/z grafu
Add Error Bars	zobrazenie chyby nameraných dát úsečkami

Add Function ...	pridanie užívateľom definovanej funkcie
New Legend	pridanie legendy (obnovenie vymazanej)
Add Text	pridanie ľubovolného textu (po kliknutí na graf sa otvorí okno na editovanie textu)
Draw Arrow/Line	pridanie šípky alebo úsečky so šípkou
Add time stamp	pridanie dátumu a času
Add Image	pridanie obrázka (jpg, bmp, gif, png a iné)
Add Layer	pridanie nového grafu do grafického okna
Remove Layer	odobratie grafu z grafického okna
Arrange Layers	úprava grafov (písmo, tituly, popis osí, ...)

Data

Disable tools	zapnutie všeobecného kurzora
Zoom	zapnutie lupy
Rescale to show all	prekreslenie grafu do celého okna
Data reader	kliknutím na bod sa otvorí okno Data display a zobrazia sa súradnice
Select data range	umožní kurzorom myši alebo šípkami klávesnice vybrať určitý rozsah dát
Screen reader	čítač súradníc, otvorí okno Data display a načíta súradnice hociktorej pozície v okne grafu
Move data Points ...	umožní premiestnenie bodov grafu, zmeny sa prejavia aj v tabuľke
Remove Bad Data Points ...	umožní odstránenie bodov z grafu, y-ové hodnoty bodov sa vymažú z tabuľky

Analysis

Translate	prekladanie dát vo vodorovnom a zvislom smere
Differentiate	výpočet prvej derivácie z dát
Integrate	numerický výpočet integrálu
Smooth	„vyhladenie“ krivky metódou FFT filtra, metódou pohyblivého priemeru a Savitzkého-Golayovou metódou
FFT filter	rôzne filtre (dolno a hornopriepustný, pásmový priepustný a blokový)
Interpolate ...	interpolácia dát (lineárna, kubická a Akimova)
FFT ...	inverzná a dopredná FFT
Fit Linear	lineárna regresia
Fit Polynomial ...	polynomická regresia do 9. stupňa
Fit Exponential Decay	regresia exponenciálnou tlmenou krivkou

Fit Exponential Growth ...	regresia exponenciálnou rastovou krivkou
Fit Boltzmann (Sigmoidal)	regresia Boltzmannovou funkciou
Fit Gaussian	regresia Gaussovou funkciou
Fit Lorentzian	regresia Lorentzovou funkciou
Fit Multi-peak	regresia na vyznačené maximá Gaussovou alebo Lorentzovou funkciou
Non-linear Curve Fit ...	nelineárna regresia Nelderovou--Meadovou simplexovou a Levenbergovou-Marquardtovou metódou, k dispozícii sú základné matematické funkcie, sedem vstavaných funkcií a je tu aj možnosť definovať vlastné funkcie

Format

Plot ...	otvorí sa okno so záložkami s možnosťami editovať rozsah, popis a formát osí, formát mriežky a všeobecné vlastnosti grafu
Curves ...	otvorí sa okno na editovanie grafických vlastností krivky
Scales ...	nastavenie rozsahu osí
Axes ...	editovanie formátu osí
Grid ...	editovanie formátu mriežky grafu
Title ...	editovanie názvu grafu

5.3.2 Príklady použitia programu

Zadávanie a import dát do tabuľky

Zadanie vlastných dát priamo do tabuľky

Vo fyzikálnych, chemických a iných laboratóriách získavame namerané hodnoty, ktoré potrebujeme vyhodnotiť napr. štatistickými metódami, vykonať regresnú analýzu rôznymi funkciami a výsledky chceme znázorniť ako čiary v grafoch. Práve na grafické zobrazenie meraní a ich vyhodnotenie s výhodou môžeme použiť QtiPlot.

Graf vo všeobecnosti chápeme ako grafické zobrazenie funkcie $y = f(x)$, pričom x je nezávisle a y závisle premenná veličina. V tabuľkovom okne (obrázok 1) vkladáme do stĺpca 1[X] nezávisle premennú a do stĺpca 2[Y] a ďalších stĺpcov 3[Y], 4[Y] atď. závisle premenné. Počet premenných definujeme podľa našich požiadaviek, v prípade potreby vytvárame ďalšie stĺpce príkazom Table → Add column alebo pravým klikom myši v hlavičke tabuľky vyberieme z kontextového menu Add column.

Importovanie dátoveho súboru `boxbod.dat`

V úvode sme spomenuli, že funkcionalitu programu sme skúšali dátami z internetovej stránky Národného inštitútu štandardov a technológií Spojených štátov amerických (NIST, 2006). Stiahli sme si dáta z kolekcie pre nelineárnu regresiu s názvom BoxBOD[45], ktoré sú zaradené do kategórie s vysokou

[45] `http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`

náročnosťou na spracovanie. Tabuľka bola uložená do dátového súboru s názvom `boxbod.dat`. Tento súbor teraz importujeme, postupným vyvolaním nasledovných ponúk File → Import ASCII → Set import option, nastavíme formát importovaných dát a potom vykonáme import dát do tabuľky, napr. z nástrojovej lišty kliknutím na ikonu

Vyhľadáme súbor `boxbod.dat`, po voľbe sa údaje prenesú do tabuľky, pozri obrázok 3.

table1

	1[X]	2[Y]
1	1	109
2	2	149
3	3	149
4	5	191
5	7	213
6	10	224

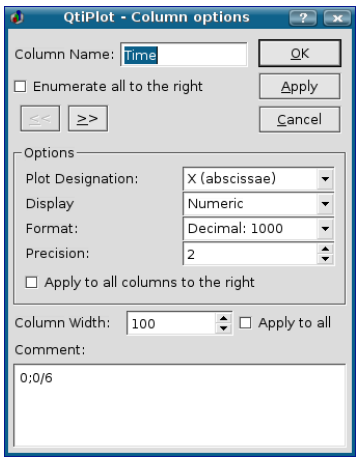
Obrázok 3: Tabuľkové okno s importovanými dátami

table1

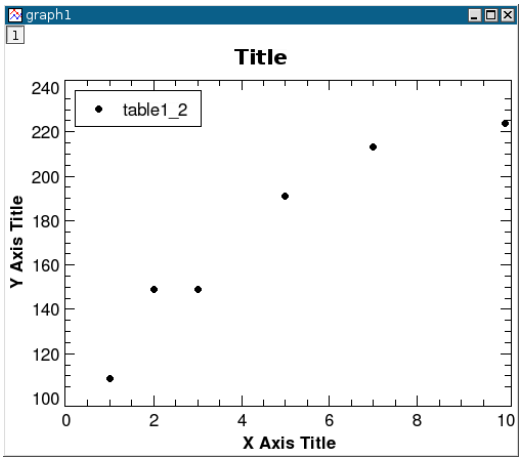
	Time[X]	2[Y]
1	1.00	109
2	2.00	149
3	3.00	149
4	5.00	191
5	7.00	213
6	10.00	224

Obrázok 4: Tabuľkové okno s importovanými dátami, premenovaným prvým stĺpcom a zmenou formátu dát

V prípade potreby môžeme premenovať nazvy stĺpcov tabuľky. Klikneme 2× na políčko 1[X], otvorí sa okno s názvom Column option, v ktorom môžeme zmeniť názov stĺpca, počet desatinných miest číselnej hodnoty premennej v stĺpci, šírku stĺpca, názov premennej a iné parametre (obrázok 5). Výsledok úpravy vidíme na obrázku 4.



Obrázok 5: Možnosti formátovania tabuľkového okna



Obrázok 6: Zobrazenie dát tabuľky z obrázku 3

Vytvorenie a úprava grafu

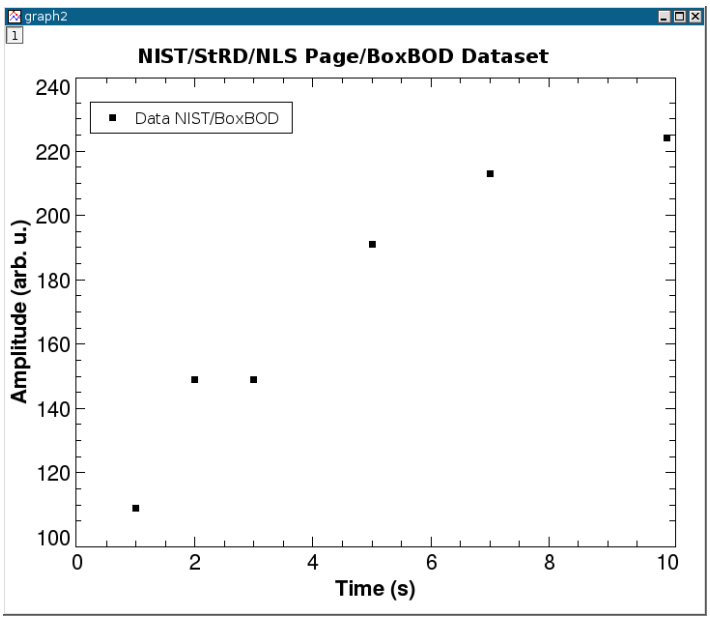
Kliknutím do hlavičky tabuľky a ťahom myši (alebo stlačením klávesu Shift a súčasným pohybom klávesových šípok) vyznačíme stĺpce závisle a nezávisle premennej a z hlavného menu zvolíme Plot → Scatter. Vykreslia sa body do grafu s názvom graph 1 (obrázok 6).

Graf by sme mohli nazvať „surovým“. V tomto grafe je zobrazená legenda a v ľavom hornom rohu grafického okna je okienko 1. Pokiaľ by sme nevyznačili v dátovej tabuľke nezávisle premennú,

dvojklikom na toto okienko sa otvorí dialógové okno Add/Remove curves, v ktorom je ponuka presunutia údajov z okienka Available data do okienka Graph contents a potom kliknutím na položku Plot Association ... si môžeme vybrať nezávisle aj závisle premennú, ktoré chceme zobraziť v grafe.

Pomenovanie osí vykonáme dvojklikom na jednotlivé osi grafu alebo z hlavného menu zvolíme Format → Plot ... → Axis a prevedieme požadované úpravy. Dvojklikom do rámčeka s legendou v ľavom hornom rohu grafu sa otvorí okno na editovanie legendy. Ak je to potrebné urobíme úpravy, zmeny sa prejavia klikom na položku Apply. Ukončenie editácie potvrdíme klikom na položku OK alebo Cancel. „Surový" graf má názov Title, jeho premenovanie môžeme urobiť dvojklikom myši na tento názov alebo z hlavného menu zvolíme Format → Title ..., otvorí sa okno na editovanie textu, napíšeme nový názov a premenovanie potvrdíme klikom na položku Apply, pričom samozrejme môžeme použiť aj písmo s diakritikou, grécke písmená, symboly a pod., pozri obrázok 11. Editáciu názvu ukončíme klikom na položku OK alebo Cancel. Podobným postupom zmeníme aj názvy osí.

Dvojklikom na ľubovolný bod grafu môžeme zmeniť tvar, farbu a veľkosť symbolov na vykreslenie dát. V prípade, že v grafe máme hustú sieť kriviek aj s dátami, odporúča sa toto editovanie vykonať z hlavného menu Format → Curves ... Opísané úpravy vidíme na obrázku 7.



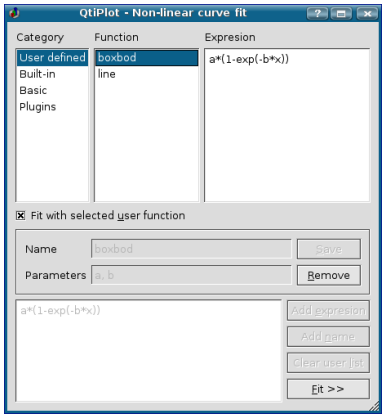
Obrázok 7: Upravený graf z obrázku 6

Nelineárna regresia pre súbor `boxbod.dat`

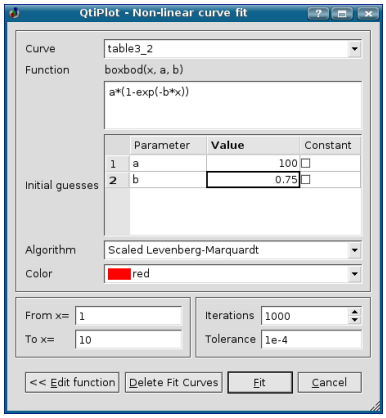
Aproximujme dáta znázornené v grafe na obrázku 7 exponenciálnou závislosťou v tvare

$$y = a[1 - \exp(-bx)],$$

ktorá je podľa (NIST, 2006) modelovou funkciou pre tieto dáta. V hlavnom menu klikneme na položku Analysis → Non-linear Curve Fit ... a vyberieme ponuku User defined. Do ľavého dolného okna zapíšeme aproximačnú rovnicu `a*(1-exp(-b*x))`, do okienka Name vpíšeme názov funkcie, napr. `boxbod`, do okienka Parameters vpíšeme symboly `a, b` fitovaných parametrov oddelených čiarkou a medzerou, potom kliknutím na položku Save vytvorenú funkciu uložíme (objaví sa v zozname

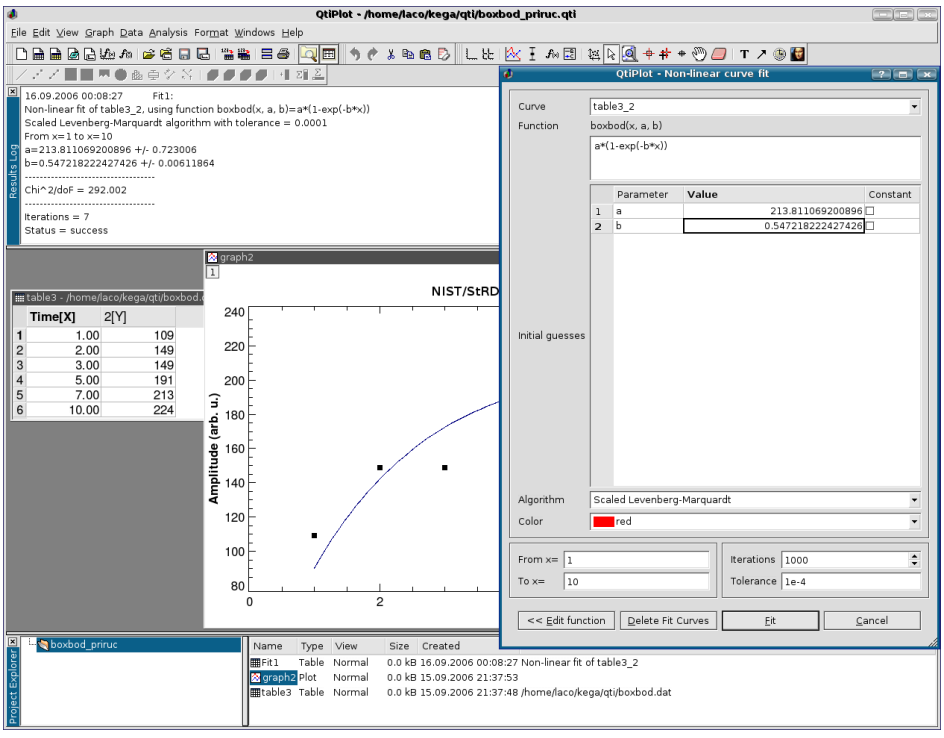


Obrázok 8: Prvé okno na definovanie požadaviek fitovania pre dáta BoxBOD



Obrázok 9: Druhé okno na definovanie požadaviek fitovania pre dáta BoxBOD

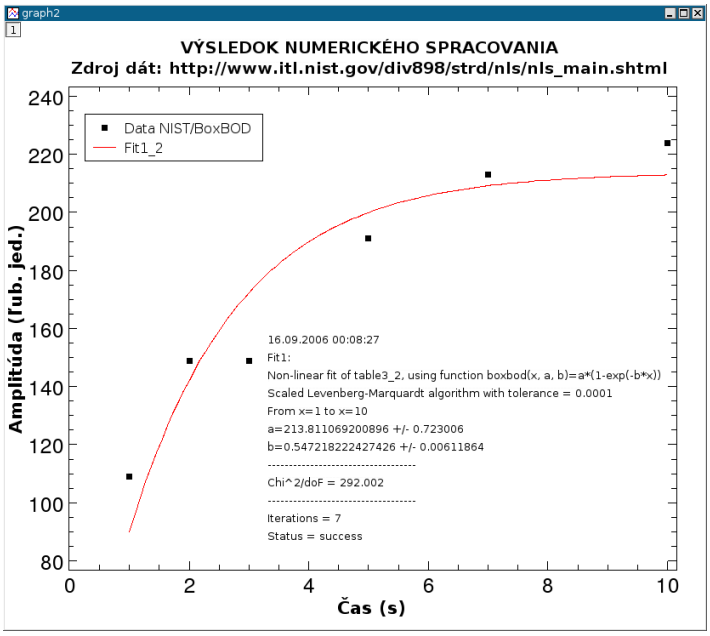
Function), pozri obrázok 8. Klikom myši na prvok zoznamu z okna Category sa zobrazí v okne Function zoznam funkcií z danej kategórie (vybrané položky sa podfarbia modrou farbou). Vyberieme si samozrejme tú našu boxbod. V príprave na fitovanie pokračujeme zaškrtnutím políčka Fit with selected user function a potom kliknutím na položku Fit >>. Otvorí sa nám okno, v ktorom nastavíme štartovacie hodnoty Initial guesses, vyberieme algoritmus fitovania, rozsah nezávisle premennej, maximálny počet iterácií a toleranciu na ukončenie procesu (obrázok 9). Kliknutím na položku Fit sa odštartuje fitovanie a po jeho ukončení sa na pracovnej ploche objaví tabuľka Result Log s výsledkami, pričom sa v grafe zobrazí regresná krivka (obrázok 10).



Obrázok 10: Hlásenie programu o ukončení a o výsledkoch fitovania

Výsledky fitovania môžeme vložiť do poľa grafu kopírovaním tabuľky cez schránku, pričom môžeme použiť postup na editovanie a vkladanie textu do plochy grafu Graph → Add text. Výsledok vidíme znázornený na obrázku 11 a uvádzame tu aj tabuľku výsledkov:

```
16.09.2006 00:08:27  Fit1:
Non-linear fit of table3_2,
using function boxbod(x, a, b)=a*(1-exp(-b*x))
Scaled Levenberg-Marquardt algorithm with tolerance = 0.0001
From x=1 to x=10
a=213.811069200896 +/- 0.723006
b=0.547218222427426 +/- 0.00611864
----------------------------------
Chi^2/doF = 292.002
----------------------------------
Iterations = 7
Status = success
```



Obrázok 11: Nelineárna regresia dát BoxBOD exponenciálnou funkciou

Výpis nás informuje o dátume a čase fitovania, že toto fitovanie údajov z tabuľky `table3_2` funkciou `boxbod` je prvé v tomto projekte, urobené nelineárnou metódou použitím Levenbergovho-Marquardtovho algoritmu s toleranciou `0.0001`. Ďalej sa uvádza rozsah nezávisle premennej `x`, fitované parametre `a` a `b` so štandardnými neistotami, hodnota $\widetilde{\chi}^2$, počet iterácií a napokon hlásenie, že proces fitovania bol ukončený úspešne.

Lineárna regresia funkciou $y = ax$

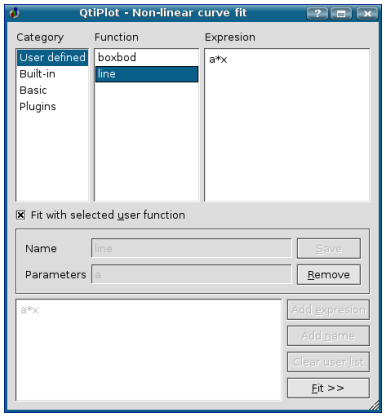
V krátkosti ešte opíšeme postup lineárnej regresie modelovou funkciou

$$y = ax,$$

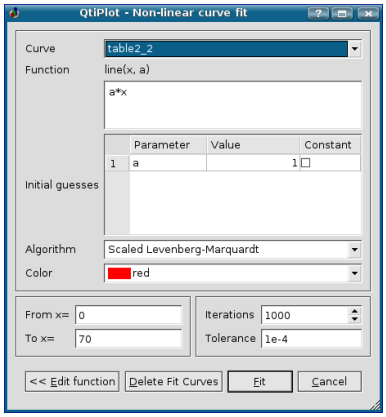
ktorú použijeme na fitovanie dát NoInt1 z kolekcie pre lineárnu regresiu už spomenutého inštitútu NIST (2006)[46]. V položke Analysis máme síce ponuku Fit Linear pre modelovou funkciou $y = a + bx$,

[46] `http://www.itl.nist.gov/div898/strd/lls/lls.shtml`

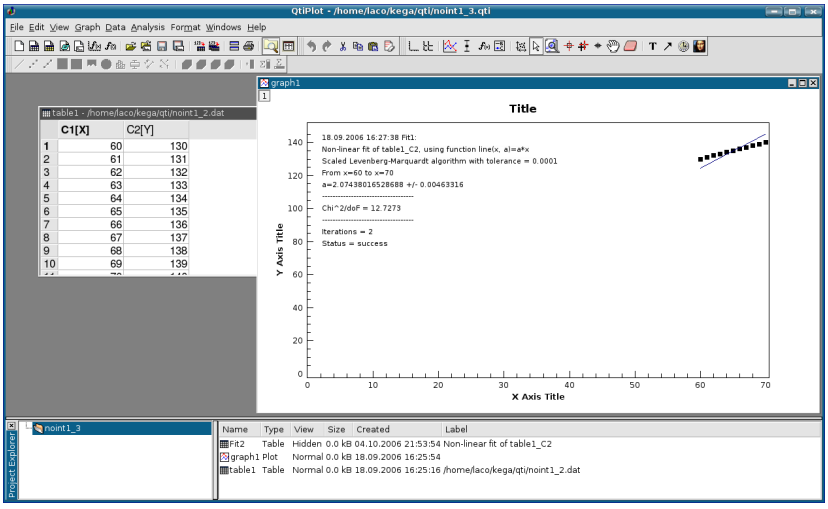
ale akosi jej chýba možnosť riešiť prípad, keď parameter $a = 0$. Pomôžeme si teda definovaním vlastnej funkcie `y = a*x`, ktorú nazveme `line`. Zopakujeme postup, ktorý sme použili v predošlom príklade nelineárnej regresie, výsledok vidíme na obrázkoch 12 a 13. Štartovaciu hodnotu parametra a nacháme implicitnú $a = 1$ a taktiež algoritmus fitovania ponecháme bez zmeny. Jediné čo zmeníme je začiatočná hodnota nezávisle premennej, ktorú nastavíme na hodnotu 0, aby sa graf zobrazil tak, ako ten, ktorý je na WWW stránke inštitútu NIST. Grafický výsledok nie je „oslňujúci“, trochu ho upravíme, aby bolo jasné, že závislosť má rozsah v oboch smeroch osí od 0. Z hlavného menu vyberieme položku Format → Axes → Scale a začneme s úpravami. Najprv zmeníme x-ový rozsah, úpravu potvrdíme klikom na Apply, postup zopakujeme pre y-ový rozsah. Očakávali sme, že čiara fitu bude predĺžená do začiatku súradnicového systému, nestalo sa tak, pozri obrázok 14.



Obrázok 12: Prvé okno na definovanie požadaviek fitovania pre dáta NoInt1



Obrázok 13: Druhé okno na definovanie požadaviek fitovania pre dáta NoInt1



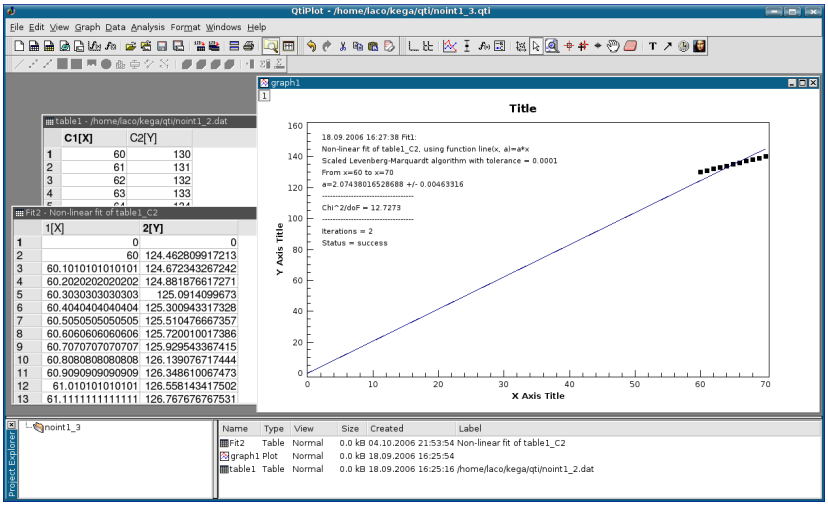
Obrázok 14: Prvé okno na definovanie požadaviek fitovania pre dáta NoInt1

Programu na jej vykreslenie pravdepodobne chýba bod $(0,0)$. Úpravu môžeme vykonať dvoma spôsobmi:

1. Tabuľku Fit2, ktorá obsahuje dáta na zobrazenie fitovanej čiary doplníme bodom $(0,0)$ tak, že

na jej začiatok vložíme nový riadok s hodnotami $x = 0$ a $y = 0$. Klikneme na značku prvého riadku tabuľky, vyfarbí sa na modro, potom pravým klikom vyberieme z kontextového menu Insert Row. Do vytvorených prázdnych políčok vložíme nuly. Touto úpravou dosiahneme to, že sa zobrazenie fitovanej čiary v grafe začne z bodu $(0,0)$. Teraz už môžeme náš obrázok porovnať s obrázkom, ktorý je na WWW stránke inštitútu NIST, pozri obrázok 15. Výpis výsledku fitovania:

```
18.09.2006 16:27:38  Fit1:
Non-linear fit of table1_C2, using function line(x, a)=a*x
Scaled Levenberg-Marquardt algorithm with tolerance = 0.0001
From x=60 to x=70
a=2.07438016528688 +/- 0.00463316
----------------------------------
Chi^2/doF = 12.7273
----------------------------------
Iterations = 2
Status = success
```



Obrázok 15: Druhé okno na definovanie požadaviek fitovania pre dáta NoInt1

2. Môžeme zopakovať postup z bodu 1 pre tabuľku table1 s dátami `noint1.dat` a urobiť nové fitovanie pre x od 0 do 70. Mohli by sme namietať, že je to „násilné konanie“ pridať do tabuľky body a tak ovplyvniť fitovanie. Priložené výpisy však ukazujú, že v oboch prípadoch sú získané parametre rovnaké. Výpis výsledku fitovania:

```
22.01.2006 01:51:44  Fit7:
Non-linear fit of table1_2, using function: user1(x, a, b)=a*x
Scaled Levenberg-Marquardt algorithm with tolerance = 0.0001
From x=0 to x=70
a=2.07438 +/- 0.00463
b=0.00000 +/- 0.00000
----------------------------------
chisq/dof = 12.7273
----------------------------------
Iterations = 2
Status = success
```

Ako vzor na „kozmetickú“ úpravu grafu nám môže poslúžiť príklad z obrázku 21. Urobíme ju pomôckami z položky Format a hotový projekt uložíme.

5.3.3 Spôsoby zobrazenia viacerých grafov

Stáva sa, že sa vyžaduje zakreslenie dvoch (a niekedy aj viacerých) fyzikálnych veličín rôznych rozsahov do jedného grafu alebo zlúčiť viac grafov do jedného obrázku. Napr. chceme zobraziť priebeh rýchlosti a zrýchlenia voľného pádu guľôčky vo viskoznom prostredí. Za predpokladu, že pohyb guľôčky sa deje iba vo zvislom smere veľmi malou rýchlosťou jej pohybová rovnica má tvar, pozri napr. (UHRIN A KOL., 2006, str. 50)

$$\frac{4}{3}\pi r^3 \rho \frac{\mathrm{d}v}{\mathrm{d}t} = \frac{4}{3}\pi r^3 g_\mathrm{n}(\rho^* - \rho) - 6\pi r \eta v, \tag{55}$$

kde r je polomer guľôčky, v rýchlosť jej pohybu vzhľadom na pokojnú tekutinu, ρ je hustota (objemová hmotnosť) guľôčky, ρ^* je hustota tekutiny, g_n je normálne tiažové zrýchlenie a η je viskozita tekutiny.

Predchádzajúca rovnica je *lineárna diferenciálna rovnica prvého rádu* s konštantnými koeficientami s pravou stranou. Rieši sa známymi štandardnými metódami a jej riešenie pre počiatočnú podmienku – rýchlosť v čase nula sa rovná nule – má tvar

$$\begin{aligned} v &= v_0 \left[1 - \exp\left(-\frac{9}{2}\,\frac{\eta\, t}{r^2\rho}\right)\right], \\ v_0 &= \frac{2\, r^2(\rho^* - \rho)\, g_\mathrm{n}}{9\,\eta}. \end{aligned} \tag{56}$$

Závislosť rýchlosti od času je teda daná rozdielom dvoch členov. Člen v_0 je časovo nezávislý, druhý člen je exponenciálne klesajúci, ktorý po určitom čase prakticky vymizne a guľôčka sa bude pohybovať rovnomerne a priamočiaro rýchlosťou v_0. Časový priebeh zrýchlenia bude rovný prvej derivácii rýchlosti v podľa času

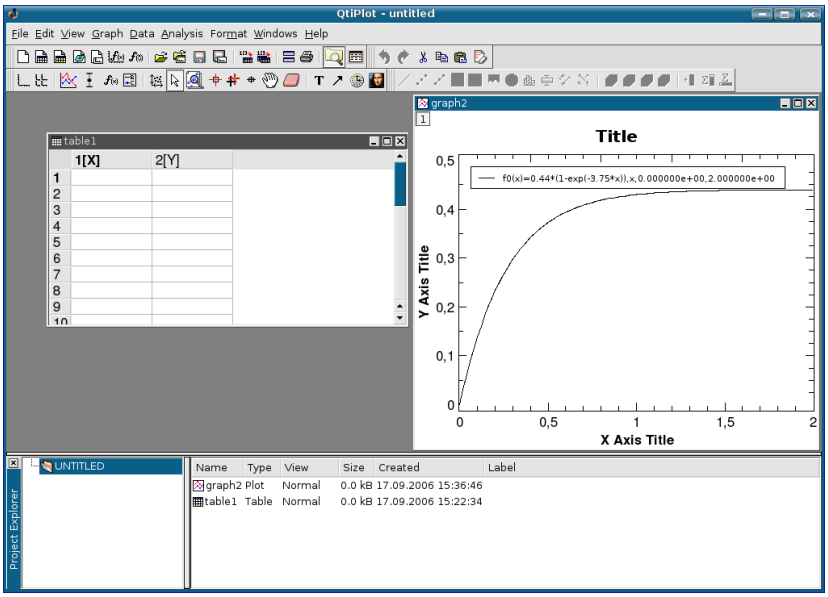
$$\begin{aligned} a &= a_0 \exp\left(-\frac{9}{2}\,\frac{\eta\, t}{r^2\rho}\right), \\ a_0 &= \frac{(\rho^* - \rho) g_\mathrm{n}}{\rho}. \end{aligned} \tag{57}$$

Urobíme numerický výpočet rýchlosti v a zrýchlenia a pre pohyb plexisklovej guľôčky o polomere $r = 1\,\mathrm{mm}$ vo vode v časovom intervale od 0 do 2 sekúnd. Pomocou možnosti vkladania funkcií do grafov, z hlavného menu grafického okna vyberieme Graph → Add Function... Číselné hodnoty v_0, a_0 a konštantného člena v exponentoch sú:

$$\begin{aligned} v_0 &= \frac{2\, r^2(\rho^* - \rho)\, g_\mathrm{n}}{9\,\eta} = \frac{2(10^{-3})^2(998 - 1\,200)\, 9{,}81}{9 \cdot 10^{-3}} = -0{,}44\ \mathrm{m\,s^{-1}} \\ a_0 &= \frac{(\rho^* - \rho) g_\mathrm{n}}{\rho} = \frac{(998 - 1\,200)\, 9{,}81}{1\,200} = -1{,}65\ \mathrm{m\,s^{-2}} \\ \frac{9}{2}\,\frac{\eta\, t}{r^2\rho} &= \frac{9 \cdot 10^{-3}}{2(10^{-3})^2 1\,200} = 3{,}75\ \mathrm{s^{-1}} \end{aligned}$$

Znamienko mínus v prvých dvoch výrazoch má fyzikálny význam. Uvedomme si, že v a a sú vlastne z-ové zložky rýchlosti a zrýchlenia v pravouhlom súradnicovom systéme a teda môžu byť kladné aj záporné.

V prvom kroku vytvoríme číselné hodnoty na zobrazenie rýchlosti v a zrýchlenia a, ktore použijeme na tvorbu kombinovaných grafických výstupov. Začneme novým projektom, potom z položky File v hlavnom menu vyberieme New Function Plot. Otvorí sa ponuka Add function curve, do okienka f(x)= vložíme `0.44*(1-exp(-3.75*x))` pre `x` od 0 do 2 a necháme vypočítať 100 hodnôt, klikneme na OK. Okno programu sa prepne do grafického módu a zobrazí sa priebeh nami zadanej funkcie `f(x)=0.44*(1-exp(-3.75*x))` vo forme spojitej čiary, obrázok 16.



Obrázok 16: Grafický priebeh vzťahu 56

Vypočítané hodnoty potrebujeme uložiť do dátového súboru. Kurzorom myši na čiare vykonáme dvojklik, otvorí sa editor parametrov čiary. Klikneme na položku Worksheet, aktivuje sa tabuľkové okno a zobrazí sa tabuľka vypočítaných hodnôt rýchlosti, ktorú uložíme postupom File → Export ASCII, vyberieme oddeľovač stĺpcov a pomenujeme ju rychlost.dat. Postup zopakujeme na vytvorenie dát zrýchlenia, do okienka f(x)= vložíme `1.65*exp(-3.75*x)` pre `x` od 0 do 2 a opäť necháme vypočítať 100 hodnôt. Dátový súbor uložíme pod menom zrychlenie.dat. Teraz už máme uložené dáta potrebné na vytvorenie ukážok.

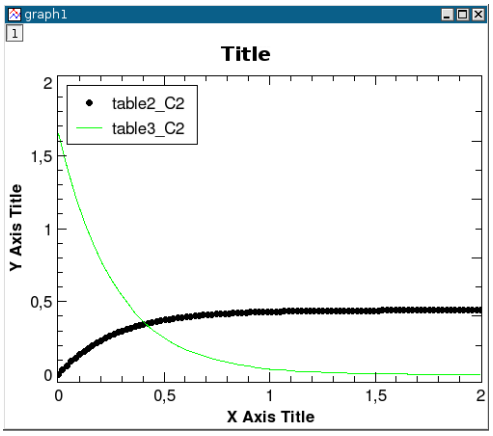
Zobrazenie dvoch priebehov v jednom grafe

Uložené dáta rychlost.dat a zrychlenie.dat použijeme na vytvorenie dvoch priebehov v jednom grafe.

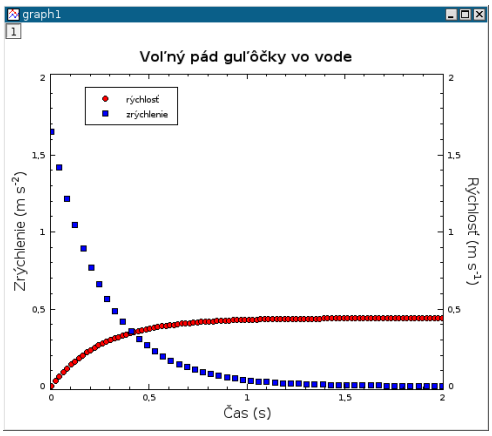
Začneme novým projektom, z ktorého vymažeme prázdnu tabuľku a budeme doňho importovať naše dáta, File → Import ASCII → Multiple files... alebo na nástrojovej lište klikneme na ikonku Import multiple data files

Vyhľadáme naše súbory, pričom súčasným stlačením klávesu `Ctrl` a klikom myši ich označíme na importovanie. V okne programu budeme mať dve tabuľky table2 s dátami rýchlosti a table3 s dátami

zrýchlenia. Označíme stĺpce tabuľky table2 a z položky Plot hlavného menu vyberieme bodové zobrazenie Scatter, otvorí sa grafické okno s priebehom rýchlosti. Do tohto grafu chceme vložiť aj priebeh zrýchlenia. Na túto operáciu použijeme postup opísaný v časti 5.3.2 na strane 149. Klikneme teda do okienka 1 a v editore Add/Remove curves presunieme dáta table3 z okna Available data do okna Graph contents. Výsledok vidíme na obrázku 17. Graf upravíme pomôckami z položky Format a projekt nezabudnime uložiť. Upravený graf vidíme na obrázku 18. Pre ďalšie použitie môžeme graf uložiť v niektorom z grafických formátov cez hlavné menu položkou File → Export Graph → Current, napr. eps.



Obrázok 17: Zobrazenie dát zo súčasne importovaných súborov rychlost.dat a zrychlenie.dat



Obrázok 18: Upravený graf z obrázku 17

Zobrazenie dvoch grafov v jednom okne

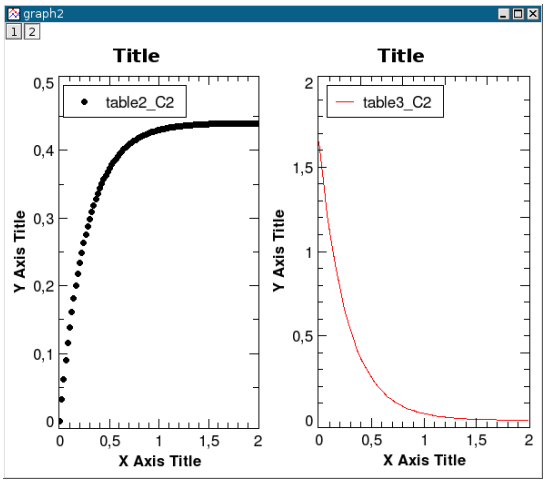
V tejto časti opíšeme postup ako zlúčiť viac grafov do jedného obrázka použije na to naše dáta rychlost.dat a zrychlenie.dat. Opäť začneme prácu novým projektom, z ktorého vymažeme prázdnu tabuľku a do tohto projektu importneme naše data postupom z predchádzajucej časti. Z tabuľkového okna sériou príkazov Plot → Scatter vykreslíme priebeh rýchlosti. Do aktívneho grafického okna, v ktorom je graf priebehu rýchlosti v, vložíme nový graf pomocou ponuky Graph → Add Layer z hlavného menu programu, potvrdíme implicitnú ponuku kliknutím na položku Guess. V ľavom hornom okne pribudne okienko 2, klikneme naň dvakrát, otvorí sa editor Add/Remove curves a môžeme presunúť tabuľku table3 do Graph contents. Výsledok nášho snaženia vidíme na obrázku 19.

Grafy upravíme pomôckami z položky Format a vytvorený projekt nezabudnime uložiť. Upravené grafy sú na obrázku 20. Na ďalšie použitie môžeme obrázok s grafmi uložiť v niektorom z grafických formátov cez hlavné menu položkou File → Export Graph → Current, napr. eps, png, jpeg, bmp, pbm, pgm, ppm, xbm, xpm. Na obrázku 21 sú oba projekty znázornené v jednom grafickom okne, na postup vyhotovenia pozorný čitateľ už iste príde aj sám.

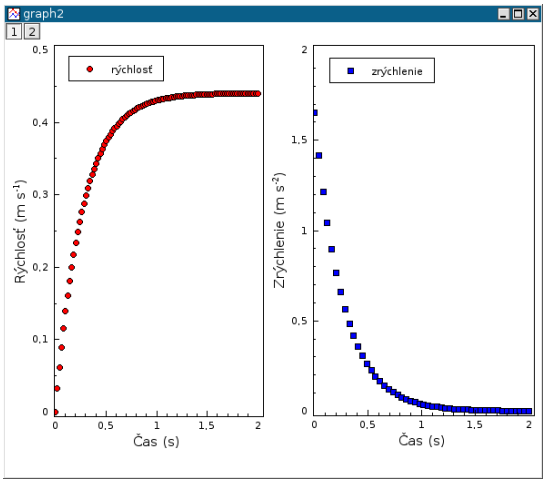
Stručne sme ukázali niektoré často používané procedúry spracovania dát a tvorby grafov. Týmito jednoduchými príkladmi sme samozrejme nevyčerpali všetky možnosti programu QtiPlot. Dobrým zdrojom ďalších informácií na prácu s programom je elektronický off-line HTML manuál prístupný na URL adrese: `http://soft.proindependent.com/manuals.html`.

V priloženej tabuľke uvádzame zoznam programom podporovaných matematických operátorov a zabudovaných funkcií, ktoré môžeme používať na vytvorenie vlastných funkcií, pri matematických

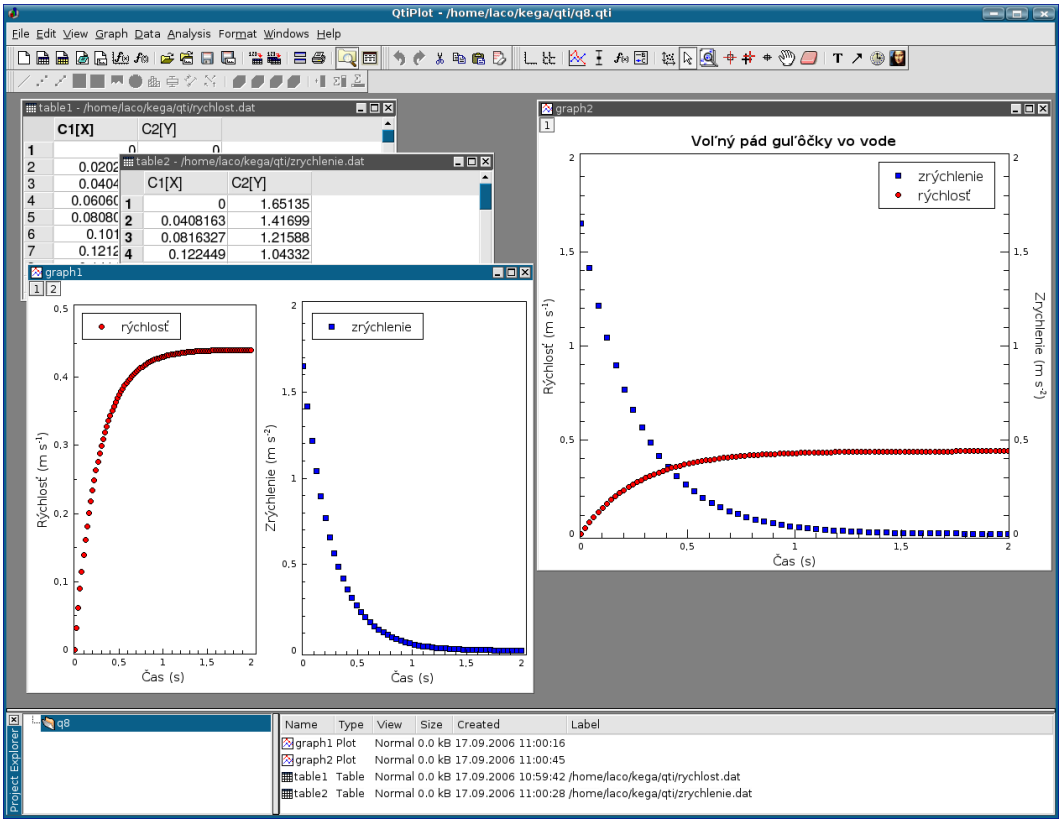
operáciach s dátami v tabuľkách a pod.



Obrázok 19: Neupravené grafy priebehov rýchlosti v a zrýchlenia a v jednom grafickom okne



Obrázok 20: Trochu upravené grafy z obrázku 19



Obrázok 21: Zobrazenie dvoch grafov v jednom okne a dvoch priebehov v jednom grafe

ZOZNAM MATEMATICKÝCH OPERÁTOROV A FUNKCIÍ PROGRAMU QtiPlot

Názov	Popis
+	súčet
−	rozdiel
*	násobenie (`a*b` = $a \cdot b$)
/	podiel, delenie
^	umocnenie (`a^b` = a^b)
`and`	logické `AND` (vracia 0 alebo 1)
`or`	logické `OR` (vracia 0 alebo 1)
`xor`	logické `Exclusive OR` (vracia 0 alebo 1)
<	menšie ako (vracia 0 alebo 1)
<=	menšie ako alebo rovná sa (vracia 0 alebo 1)
==	rovná sa (vracia 0 alebo 1)
>=	väčšie ako alebo rovná sa (vracia 0 alebo 1)
>	väčšie ako (vracia 0 alebo 1)
! =	nerovná sa (vracia 0 alebo 1)
`abs(x)`	absolútna hodnota x
`acos(x)`	arkus kosínus
`acosh(x)`	arkus hyperbolický kosínus
`asin(x)`	arkus sínus
`asinh(x)`	arkus hyperbolický sínus
`atan(x)`	arkus tangens
`atanh(x)`	arkus hyperbolický tangens
`avg(x1,x2,x3,...)`	stredná hodnota argumentov
`bessel_j0(x)`	Besselova funkcia prvého druhu $J_0(x)$ rádu 0
`bessel_j1(x)`	Besselova funkcia prvého druhu $J_2(x)$ rádu 1
`bessel_jn(x,n)`	Besselova funkcia prvého druhu $J_n(x)$ rádu n
`bessel_y0(x)`	Besselova funkcia druhého druhu $Y_0(x)$ rádu 0
`bessel_y1(x)`	Besselova funkcia druhého druhu $Y_1(x)$ rádu 1
`bessel_yn(x,n)`	Besselova funkcia druhého druhu $Y_n(x)$ indexu n
`beta(a,b)`	Beta funkcia, $B(a,b) = \Gamma(a) \cdot \Gamma(b)/\Gamma(a+b)$
`cos(x)`	kosínus x
`cosh(x)`	hyperbolický kosínus x
`erf(x)`	chýbová funkcia

pokračovanie tabuľky na ďalšej strane

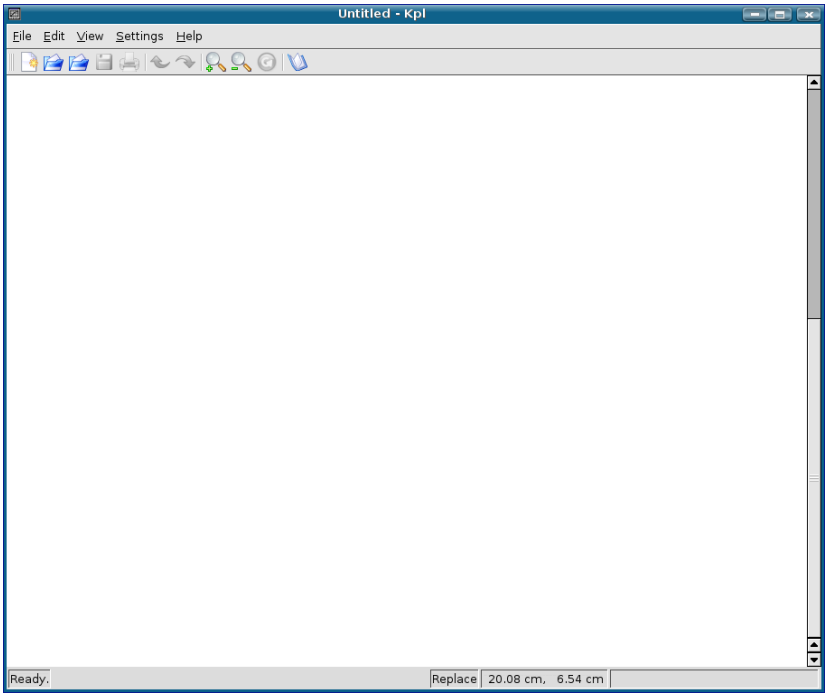
(pokračovanie tabuľky z predošlej strany)	
Názov	**Popis**
`erfc(x)`	`erfc(x)=1-erf(x)`
`erfz(x)`	hustota pravdepodobnosti normálneho rozdelenia $Z(x)$
`erfq(x)`	koncová časť normálneho rozdelenia $Q(x)$
`exp(x)`	exponenciálna funkcia so základom e
`gamma(x)`	funkcia $\Gamma(x)$
`gammaln(x)`	logaritmus funkcie $\Gamma(x)$
`hazard(x)`	`h(x)=erfz(x)/erfq(x)`[47]
`if(e1,e2,e3)`	keď je `e1` pravdivé , výpočíta sa `e2` a ešte `e3`
`ln(x)`	prirodzený logaritmus x
`log(x)`	dekadický logaritmus x
`log2(x)`	logaritmus x so základom 2
`min(x1,x2,x3,...)`	minimum zo zoznamu argumentov
`max(x1,x2,x3,...)`	maximum zo zoznamu argumentov
`rint(x)`	zaokrúhlenie na najbližšie celé číslo
`sign(x)`	funkcia znamienka x
`sin(x)`	sínus x
`sinh(x)`	hyperbolický sínus x
`sqrt(x)`	druhá odmocnina z x
`tan(x)`	tangens x
`tanh(x)`	hyperbolický tangens x

[47] V staršej literatúre sa uvádza pojem *hazard function* ako ekvivalent pojmu *hazard rate* alebo *failure rate* používaneho v teórii obnovy a poisťovníctve, v slovenčine ako intenzita poruchy (úmrtnosti). Je definovaná ako $h(x) = f(x)/(1 - F(x))$, kde $f(x)$ a $F(x)$ je hustota a distribučná funkcia doby životnosti nejakého prvku (J. Skřivánek).

5.4 Program Kpl

Program Kpl je jednoduchý z pohľadu ovládania, poskytuje rozsiahle možnosti na vyhladzovanie, optimalizáciu a numerické operácie s nameranými dátami (napr. derivovanie, integrovanie); môžeme ho dopĺňať vlastnými funkciami a knižnicami, ktoré sa napíšu a skompilujú v programovacom jazyku C. Na rozdiel od programu QtiPlot neumožňuje štatistické výpočty a charakteristiky dát. V ďalších častiach opíšeme prácu s verziou Kpl 3.3 pre grafické používateľské prostredie KDE 3.5.2.

Domovská internetová stránka programu je na URL adrese `http://frsl06.physik.uni-freiburg.de/privat/stille/kpl/`. Autor Werner Stille ponúka k programu on-line príručku prístupnú na URL adrese `http://frsl06.physik.uni-freiburg.de/privat/stille/kpl/book/index.html` . Na prácu v Kpl máme k dispozícii jedno pracovné okno, pozri obrázok 22.



Obrázok 22: Pracovné okno programu Kpl

5.4.1 Ovládacie možnosti programu Kpl

Spustenie Kpl v prostredí OS GNU/Linux sa dá uskutočniť troma spôsobmi:[48]

- kliknutie na ikonu Kpl na pracovnej ploche (keď ju máme vytvorenú),
- Menu → Kancelária → kliknutie na Kpl (keď sme program inštalovali z deb balíčka),
- z príkazového riadka X terminálu príkazom `kpl`.

Na obrazovke sa zobrazí okno programu s príslušnými ponukami a ovládacími prvkami v hlavnom menu (obrázok 22). Zatvorenie Kpl sa vykoná cez záložku File a potom Quit alebo stlačením klávesov Ctrl + Q (prípadne Alt + F4).

Opäť, pri prvom čítaní tejto kapitoly môže čitateľ, ktorý sa chce rýchlo oboznámiť s používaním programu časť 5.4.1 preskočiť a pokračovať v čítaní časťou 5.4.2 na strane 159.

[48] V prípade, keď inštaláciu vykonáte zo zdrojových súborov, musí sa cesta na spustenie z Menu nastaviť manuálne. Odporúčame spúšťať program z pracovnej plochy pomocou vytvorenej ikonky s odkazom na binárny súbor kpl.

Hlavné menu programu obsahuje tieto položky:

File Edit View Settings Help

Opíšeme tie položky, ktoré sú potrebné na základné zoznámenie sa s možnosťami Kpl. Položky označené hviezdičkou * sa aktivujú len v tom prípade, keď je otvorený grafický alebo dátový súbor.

File

New	vytvorenie nového projektu
Open Plot File ...	otvorenie súboru s príponou .kpl, editácia už vytvoreného projektu
Open Data File ...	otvorenie súboru s príponou .dat, možnosť výberu desatinnej bodky alebo čiarky
Open Recent	desať naposledy otvorených projektov
Save*	uloženie dokumentu pod pôvodným menom
Save As ...*	uloženie dokumentu pod novým menom
Close*	zatvorenie aktuálneho projektu
Print*	vytlačenie aktívneho grafu
Display Plot*	zobrazenie grafickej prezentácie alebo obnovenie zobrazeného grafu
PostScript Output	voľba orientácie grafického listu na konverziu; na výšku alebo na šírku
PostScript Preview	voľba orientácie náhľadu grafu; na výšku alebo na šírku
New Window	otvorenie nového pracovného okna programu
Close Window	zatvorenie aktuálneho pracovného okna a ukončenie programu
Quit	ukončenie práce s programom Kpl

Edit

Undo*	zruší posledný vykonaný krok
Redo*	vráti posledný vykonaný krok
Items ...	jedna z najdôležitejších položiek, umožňuje vkladať, editovať a fitovať objekty a položky v grafe (napr. funkcie, polia, splajnové krivky a pod.)

View

Zoom In (Ctrl++)	zväčšovanie krokom
Zoom Out (Ctrl+−)	zmenšovanie krokom
Zoom ...	nastavenie faktora zväčšenia/zmenšenia (%)
Redisplay* (F5)	aktuálny dátový alebo grafický súbor sa znova načíta a zobrazí, aktivuje sa funkcia Autoplot
Reload Periodically	nastavenie periodického obnovovania zobrazenia, aktivuje sa funkcia Autoplot

Settings

Hide Toolbar	skrytie/zobrazenie hlavného menu
Hide Statusbar	skrytie/zobrazenie stavovej lišty
Show Function Source	zobrazenie zdrojového súboru funkcie v dialógu voľby jej parametrov
Autoplot	automatické zobrazenie projektu po načítaní dátového alebo grafického súboru
Add Files (Insert)	pridanie nového dátového alebo grafického súboru do aktuálneho s vykreslením
Calculate PS Bounding Box	automatický výpočet hraníc postscriptového okna grafu, bez aktivácie tejto položky sa vypočítajú hranice k niektorému rozmeru strany (napr. A4 na výšku)
Print PS Output	zobrazenie dialógu tlače postscriptového súboru po jeho vytvorení
Save Absolute Paths	do grafického súboru sa uloží absolútna cesta k dátam a knižniciam
Unsaved Changes Warning	zobrazenie varovania o neuložení súboru
Save Settings At End	uloženie všetkých nastavení aktuálneho zobrazenia pri ukončení programu
Save Settings	uloženie všetkých nastavení aktuálneho zobrazenia
Configure Shortcuts ...	definovanie vlastných klávesových skratiek
Configure Toolbars ...	pridanie/odobratie ikoniek do hlavného menu
Configure Notifications ...	nastavenie hlásení a varovaní programu
Configure Kpl ...	niektoré základné implicitné nastavenia programu

Help

Kpl Handbook	otvorenie elektronického manuálu, keď je nainštalovaný
Report Bug ...	dialóg na oznamovanie chýb programu autorovi e-mailovou poštou
About Kpl	základné informácie o programe Kpl
About KDE	základné informácie o grafickom prostredí KDE

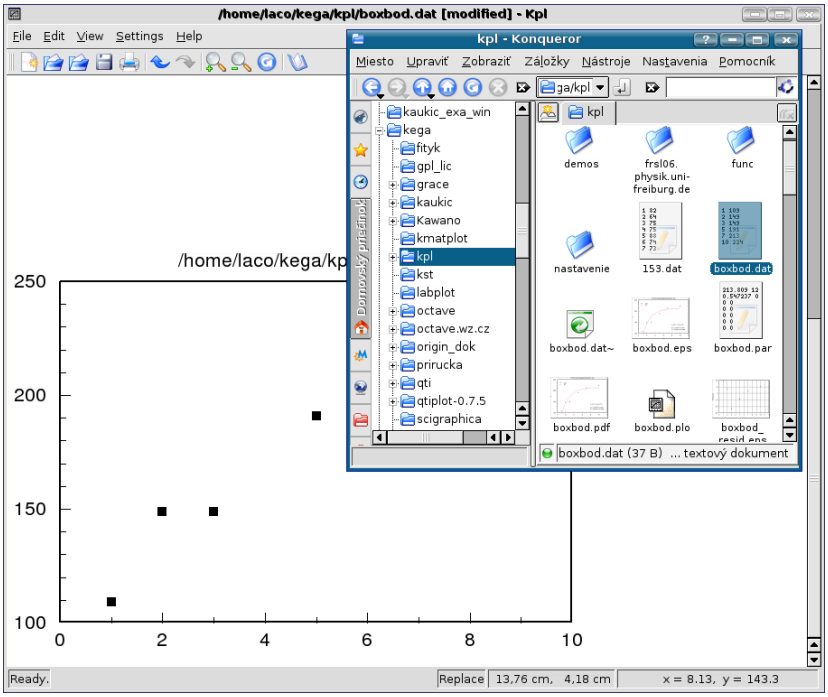
5.4.2 Príklady použitia programu

Importovanie dát, ich zobrazenie a úprava grafu

Importovanie dátoveho súboru `boxbod.dat`

Tak, ako v prípade programu QtiPlot, aj program Kpl a jeho funkcionalitu preskúšame dátami z internetovej stránky Národného inštitútu štandardov a technológií Spojených štátov amerických (NIST, 2006).

Stiahnite si dáta z kolekcie pre nelineárnu regresiu s názvom BoxBOD[49], ktoré sú zaradené do kategórie s vysokou náročnosťou na spracovanie. Tabuľku uložte do dátového súboru s názvom `boxbod.dat`.



Obrázok 23: Importovanie a zobrazenie dát metódou ťahaj a pusť

Program Kpl má síce vlastný editor tabuliek, ale s obmedzenými možnosťami formátovania, preto si na vytvorenie tabuľky vyberme radšej nejaký textový ASCII editor (napr. Kate, gedit, KSpread a pod.). Dátový súbor vytvárame a editujeme v stĺpcovom formáte, a pri jeho ukladaní do pracovného priečinka mu pridávame príponu `dat`. Symbolom desatinnej rádovej čiarky môže byť *desatinná čiarka* alebo *desatinná bodka* a ako oddeľovač (separátor) stĺpcov odporúčame použiť tabulátor (Tab) alebo medzerník (Space). Dátový súbor môžeme importovať dvoma spôsobmi:

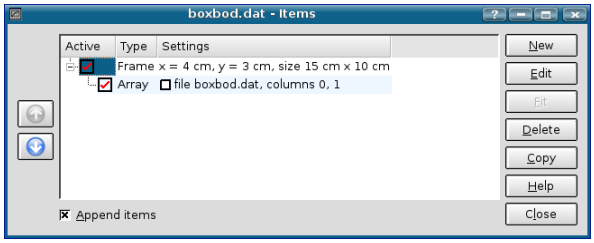
1. Vyvolaním ponúk File → Open Data File ... sa otvorí dialógové okno, v ktorom zvolíme symbol desatinnej rádovej čiarky a vyhľadáme na disku súbor na importovanie.

2. Otvoríme program Kpl a potom nejaký program na spravovanie súborov (napr. Konqueror alebo Krusader), v ktorom vyhľadáme súbor, ktorý chceme zobraziť. Označíme ho ľavým klikom myšky a ťahom ho premiestnime do okna programu Kpl, kde klik uvoľníme (metóda Drag and Drop), pozri obrázok 23.

„Surový" graf z obrázku 23 budeme upravovať vyvolaním položky Items ... Možeme ju aktivovať dvoma spôsobmi, pravým klikom myši do prázdneho poľa v okne programu (mimo poľa grafu) a z kontextového menu vyberieme žiadanú položku alebo vyvolaním ponúk Edit → Items ..., pozri obrázok 24.

Po pridaní legendy (výberom New), označení a zmene modulov osí (pozri na strane 166), zmene zafarbenia a symbolov dátových bodov (označením a výberom Edit) dostaneme takýto výsledok:

[49] `http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`



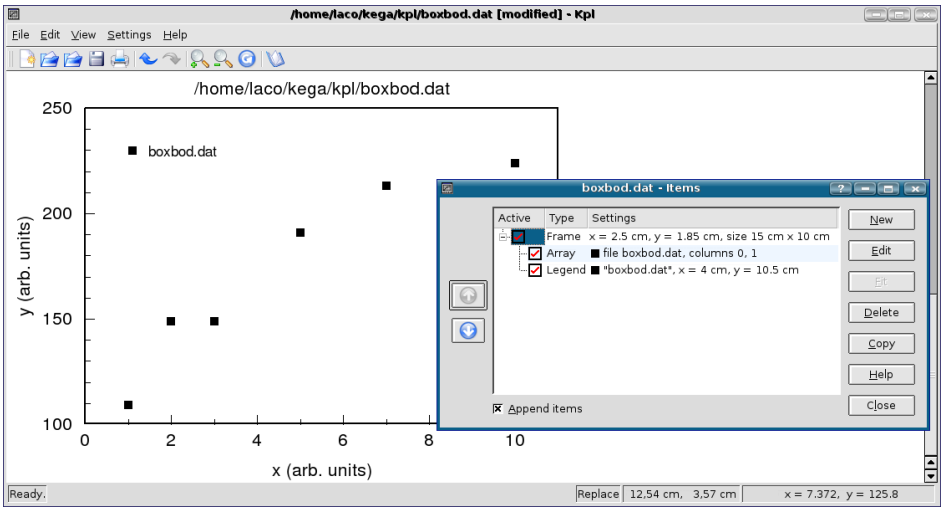
Obrázok 24: Pracovné okno položky Items ...

Nelineárna regresia pre súbor `boxbod.dat`

Tak, ako v prípade programu QtiPlot aproximujme dáta znázornené v grafe na obrázku 25 exponenciálnou závislosťou v tvare

$$y = a[1 - \exp(-bx)],$$

ktorá je podľa (NIST, 2006) modelovou funkciou pre tieto dáta. Program implicitne takúto funkciu neponúka, ale môžeme ju napísať ako skript v programovacom jazyku C, napr. `boxbod.c`, a skompilovaním vytvoriť knižnicu (modul) `boxbod.so`.



Obrázok 25: Upravený graf z obrázku 23

Postup je nasledovný: v textovom editore napíšeme napríklad takéto funkcie v jazyku C. Prvá `boxbod_1` bude na vykreslenie fitovanej čiary do grafu, druhá `boxbod_2` na iteráciu:

```
/*************************************************************/
/*      boxbod.c   2D functions for Kpl                      */
/*                                                           */
/*      Copyright (C) 2006 by Ladislav Sevcovic              */
/*      <ladislav.sevcovic@tuke.sk>                          */
/*                                                           */
/*      Released under the GPL; see file LICENSE for details.   */
```

```
/*      Use the following command to compile the C function     */
/*      and create a shared library:                            */
/*      gcc -Wall -shared -fPIC -o boxbod.so boxbod.c -lm       */
/*      Do this in a X terminal windows (shell).                */
/*      At the X terminal type:   nm boxbod.so>boxbod.def       */
/*                                                              */
/*      exponential(x, p) calculates of exponential             */
/*      Returns: p[0] * (1 - exp(-p[1] * x))                    */
/**************************************************************/
#include <math.h>
/**************************************************************/
double boxbod_1(double x, const double* p)
{
return(p[0]*(1-exp(-p[1]*x)));
}
/**************************************************************/
double boxbod_2(double x, const double* p)
{
  int i;
  double f;
  f = p[0];
  for (i = 1; i < 3; i += 1)
      f == p[i] * (1 - exp(- p[i + 1] * x));
  return f;
}
/**************************************************************/
```

Súbor uložíme do pracovného priečinka pod menom `boxbod.c` a do príkazového riadka v okne X terminálu najprv napíšeme[50]

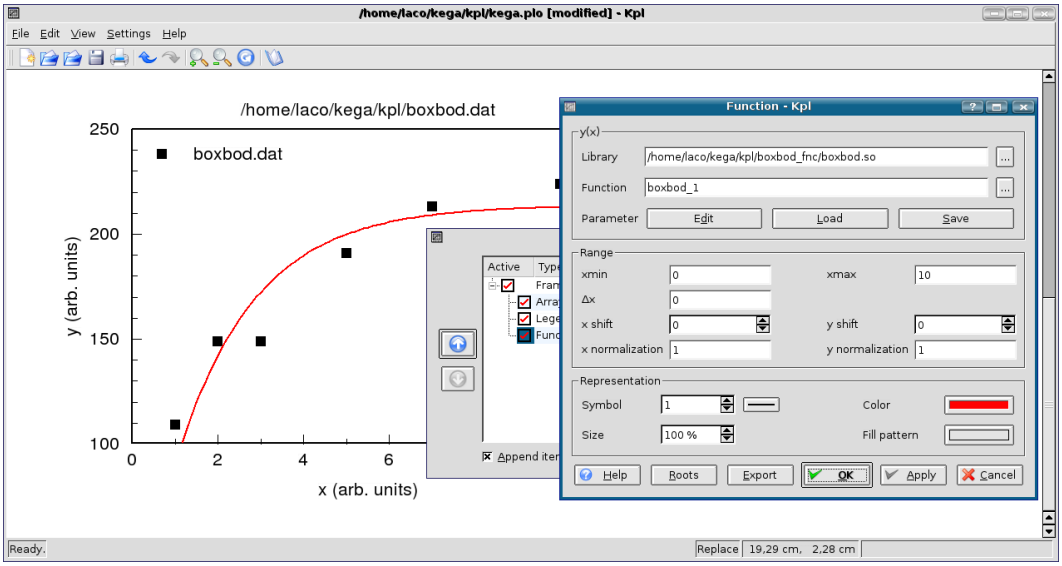
`gcc␣-Wall␣-shared␣-fPIC␣-o␣boxbod.so␣boxbod.c␣-lm`

a stlačním klávesu Enter prebehne kompilácia nášho skriptu do binárnej knižnice `boxbod.so`. Potom napíšeme

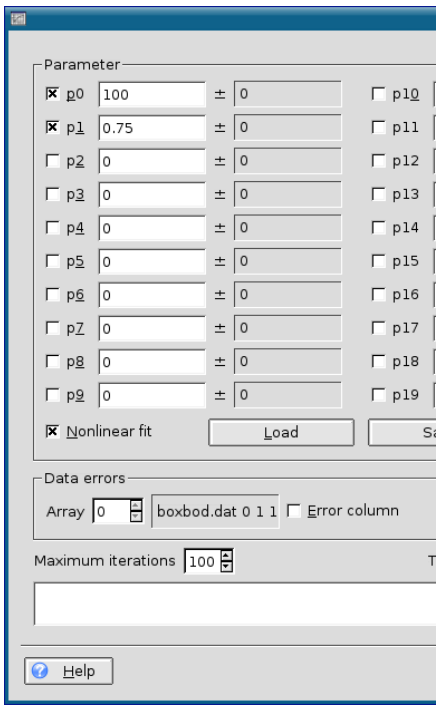
`nm␣boxbod.so>boxbod.def`

a opätovným odoslaním sa vytvorí tabuľka symbolov. Dva skompilované súbory `boxbod.so` a `boxbod.def` použijeme na fitovanie, môžeme ich teda presunúť do pracovného priečinka, v ktorom máme ostatné súbory s dátami pre Kpl alebo do osobitného podpriečinka, v ktorom budú len knižnice (aj budúce). Teraz už môžeme začať s fitovaním dát, v okne položky Items (obrázok 24) klikneme na ponuku New a potom na ponuku Function (obrázok 26). Vyhľadáme si našu knižnicu `boxbod.so` a z nej vyberieme funkciu `boxbod_1`, doplníme xmax na 10, vyberieme symbol, veľkosť a farbu fitovacej čiary v grafe a výber ukončíme potvrdením Apply a potom OK. Prejdeme opäť do okna Items, kde klikneme na novovytvorenú položku Function a potom na okienko Fit, čím sa nám otvorí okno Parameter fit, pozri obrázok 27. Zaškrtneme ľavé okienka ☒ pre parametre `p0` a `p1` a do pravých vpíšeme ich štartovacie hodnoty, pre `p0=100` a pre `p1=0.75`. Fitovanie bude nelineárne, preto zaškrtneme aj okienko ☒ Nonlinear fit. Kliknutím do položky Model sa otvorí okno Error model function, v ktorom opäť vyhľadáme knižnicu `boxbod.so` a z nej tentoraz vyberieme funkciu `boxbod_2`, ako argument zvolíme ycolumn. Kliknutím na položku Edit zadáme štartovacie parametre iteračného procesu (iteračný proces

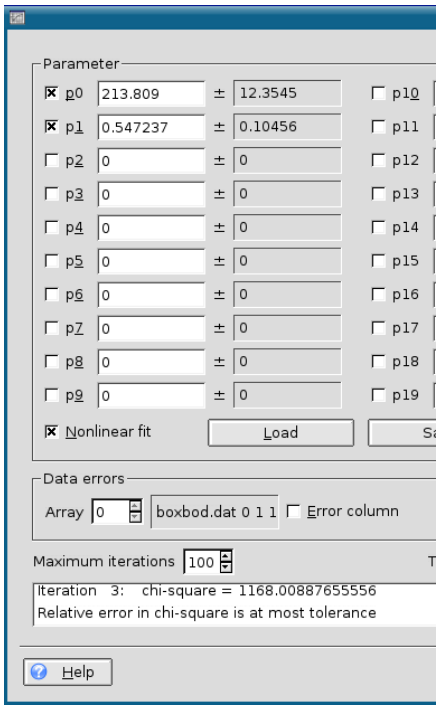
[50] Súčasťou OS GNU/Linux je aj kompilátor `gcc` jazyka C a rad ďalších programátorských nástrojov. Program `nm` vytlačí tabuľku symbolov (zoznam názvov) v abecednom poradí pre jeden alebo viac objektových súborov. Výstup obsahuje pre každý symbol meno, hodnotu, typ, veľkosť a pod.



Obrázok 26: Okno položky Function na vykreslenie fitovanej funkcie

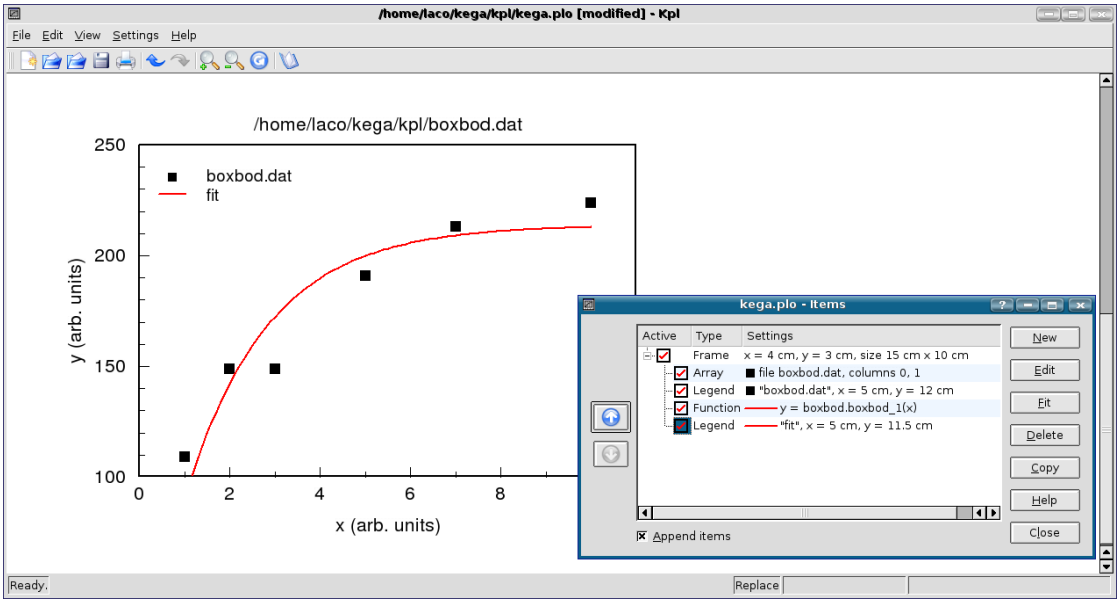


Obrázok 27: Výrez okna položky Parameter fit so štartovacími parametrami iteračného procesu fitovania



Obrázok 28: Výrez okna položky Parameter fit s výsledkami iteračného procesu fitovania

sa ukončí dobre aj so začiatočnými hodnotami `p0=1` a `p1=1`). Výsledné parametre fitovanej funkcie `y=p[0]*(1-exp(-p[1]*x))` sa vpíšu do príslušných okienok parametrov (obrázok 28). Kliknutím na položku Apply sa vysledok fitovania zobrazí v grafe (obrázok 29). Prácu s fitovaním ukončíme kliknutím na položku OK. Do grafu vložíme legendu, názvy osí a tak ďalej (pozri 5.5. časť).



Obrázok 29: Výsledný graf s fitovanou krivkou pre dáta `boxbod.dat`

Lineárna regresia funkciou $y = ax$

Podobne ako v programe QtiPlot aj na tomto mieste v krátkosti ešte opíšeme postup lineárnej regresie modelovou funkciou

$$y = ax,$$

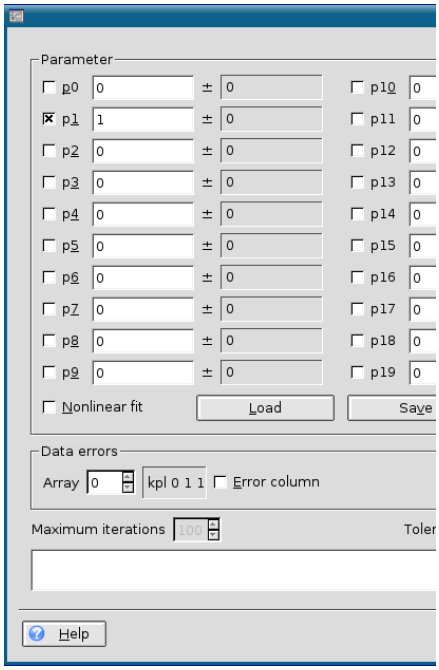
ktorú použijeme na fitovanie dát NoInt1 z kolekcie pre lineárnu regresiu už spomenutého inštitútu NIST (2006)[51], aby sme mohli výsledky oboch programov porovnať.

Aj v tomto prípade tabuľku dát môžeme doplniť bodom $(0,0)$, aby sme získali výsledné grafické zobrazenie v takom tvare, aké je na WWW stránke inštitútu NIST. Úpravu prevedieme takto: z ponuky Items označíme položku Array a klikneme na okno Edit. Otvorí sa nám nové okno, v ktorom zaškrtneme okienko ☒ Internal data a potom vyberieme ponuku Edit. Do odsadeného prvého riadka zapíšeme tri nuly s medzerami 0␣0␣0, upravu potvrdíme klikom na Apply a editor zatvoríme klikom na OK.

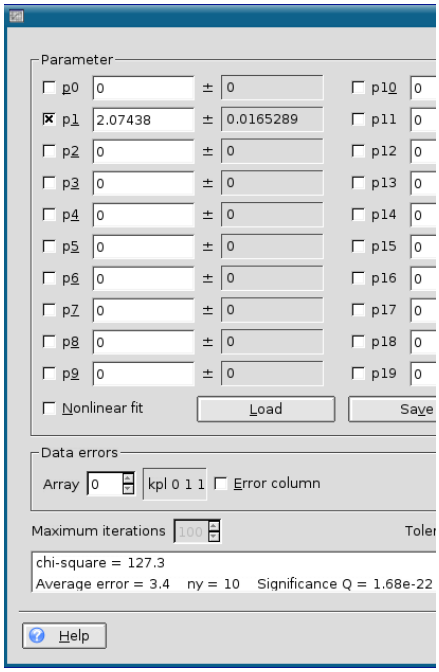
Získanie optimálneho parametra a bude jednoduchšie ako v predošlom prípade, lebo môžeme na to použiť zabudovanú funkciu programu Kpl (samozrejme, môžeme si napísať aj vlastnú). Postup bude podobný, ako pri nelineárnej regresii, s tým rozdielom, že na fitovanie použijeme knižnicu `fkt.so`, z ktorej vyberieme funkciu polynom.

Úprave hodnôt v tabuľke `noint1.dat` sa však môžeme vyhnúť. Na rozdiel od postupu v prípade postupu programom QtiPlot (pozri na strane 149) však nemusíme upravovať žiadnu tabuľku, lebo program Kpl údaje na vykreslenie krivky fitovanej funkcie neukladá do osobitnej tabuľky. Vyvolaním okna položky Items označíme v nej položku Function a klikneme na okno Fit. Otvorí sa nám okno Parameter fit, v ktorom z ľavých okienok pre parametre zaškrtneme len okienko pre parameter ☒ `p1`, do pravého okienka vpíšeme štartovaciu hodnotu `p1=1` a okienko ☐ Nonlinear fit zostane nezaškrtnuté, pozri obrázok 30. Výsledky vidíme na obrázkoch 31 a 32.

[51] `http://www.itl.nist.gov/div898/strd/lls/lls.shtml`

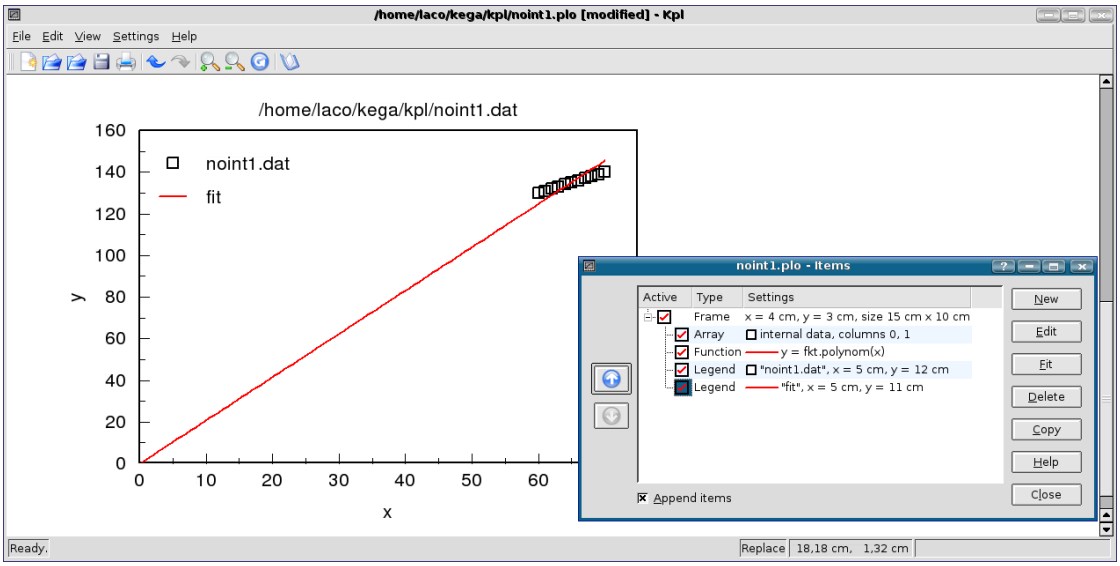


Obrázok 30: Výrez okna položky Parameter fit so štartovacím parametrom



Obrázok 31: Výrez okna položky Parameter fit s výsledkom procesu fitovania

Vytvorené grafy upravené podľa odporúčaní z časti 5.5 môžeme uložiť vo formáte `*.ps` alebo `*.eps` na ďalšie spracovanie (z hlavného menu File → PostScript Output).



Obrázok 32: Výsledný graf s fitovanou krivkou pre dáta `noint1.dat`

Jeden obrázok má hodnotu tisíc slov.

STARÉ ČÍNSKE PRÍSLOVIE

5.5 Niekoľko pravidiel na tvorbu grafov

Z precízne vyhotoveného grafu nameranej fyzikálnej závislosti dvoch veličín sa dajú s dostatočnou mierou presnosti určiť charakteristiky funkcie. Je možné napr. určiť polohu extrémov, inflexných bodov, pri lineárnej závislosti odčítať z grafu smernicu priamky atď. Graf je vždy názornejší ako tabuľka, tabuľka je však vždy presnejšia. Grafu dávame prednosť, keď chceme ukázať priebeh, tendenciu, štruktúru alebo obrazec.

Dôvod je jednoduchý a spočíva v rýchlom, pohodlnom a názornom prijímaní obrazovej informácie človekom. Dalo by sa povedať, že graf slúži na rýchlu kvalitatívnu orientáciu v nameranej závislosti a ak nás zaujímajú podrobnejšie kvantitatívne údaje, z pamäte počítača si necháme zobraziť tabuľku funkcie resp. analytický predpis, interpolačnú formulu atď. Z dôvodu názornosti je grafické zobrazenie funkcií veľmi časté i vo fyzikálnej literatúre a takmer každá nameraná závislosť je reprezentovaná grafom. Na zhotovenie grafov *nie sú jednoznačné pravidlá* a v každom odbore sú trocha odlišné zvyklosti určené napr. tradíciou, typografickými možnosťami časopisov a pod. Na zrozumiteľný a prehľadný graf budú kladené tieto požiadavky:

1. *Modul stupnice* grafu zvolíme tak, aby graf bol dostatočne veľký, t. j. interval nezávisle premennej má byť zobrazený na „vodorovnej“ osi viac ako na dvoch tretinách „vodorovného“ rozmeru grafu a analogicky interval na „zvislej“ osi. Pod pojmom *modul stupnice* rozumieme podiel intervalu nameraných (v prípade extrapolácie potrebných) hodnôt fyzikálnej veličiny k dĺžke osi, povedzme v mm, na ktorú chceme interval zobraziť. Napr. obrázok 33 znázorňuje graf, v ktorom keď zvolíme dĺžky osí 120 mm bude modul vodorovnej stupnice $\mathrm{M}_1 = (90\,\mathrm{V} - -40\,\mathrm{V})/120\,\mathrm{mm} = 0{,}416\,6\,\mathrm{V/mm}$ a zvislej stupnice $\mathrm{M}_2 = (7\,\mathrm{A} - 1\,\mathrm{A})/120\,\mathrm{mm} = 0{,}05\,\mathrm{A/mm}$.

2. Osi vyznačíme plnou úsečkou a označíme jednotkami v *okruhlej zátvorke*, v ktorých je fyzikálna veličina vynášaná. Osi nekalibrujeme hodnotami, ktoré sme namerali, ale takými hodnotami, medzi ktorými je ľahká interpolácia.

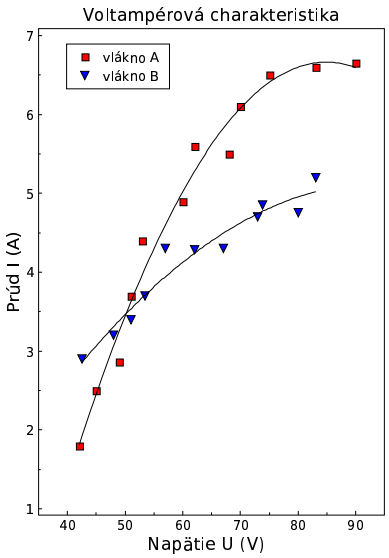
3. V každom prípade do grafu vhodnými symbolmi vyznačíme namerané hodnoty. Ak je v jednom grafe viac priebehov alebo na jednom papieri viac grafov, pre rôzne priebehy volíme rôzne symboly na označenie nameraných hodnôt (napr. plné body pre jeden graf, trojuholníky pre ďalší atď.). Od nameraných hodnôt nevedieme na osi žiadne čiary (pozri obrázok 33)[52].

4. Každý graf opíšeme stručným komentárom, aby bolo jasné, akú závislosť graf vyjadruje.

Meranie je zaťažené chybami a po vynesení nameraných hodnôt zistíme, že body sú „rozhádzané“. Treba sa rozhodnúť ako preložiť cez namerané body čiaru. Ak sme meranie vyhodnotili metódou najmenších štvorcov a určili parametre z rovníc (35), potom pretabelujeme funkciu $F^*(x, \hat{p}_1, \ldots \ldots, \hat{p}_k)$ (Kapitola 5.2) a túto funkciu vynesieme do grafu. Získame tak jednoznačne určenú (v zmysle vyrovnávajúceho počtu vyrovnávajúcu) hladkú čiaru. Napr. pri lineárnej závislosti $y = a + bx$ zistíme,

[52]Tento obrázok bol vytvorený programom QtiPlot, uložený vo formáte EPS a potom programom epstopdf konvertovaný do formátu PDF.

že táto priamka neprechádza všetkými nameranými bodmi, ale približne polovica bodov je nad a približne polovica bodov pod priamkou.



Obrázok 33: Príklad nakresleného grafu, ktorý znázorňuje voltampérovú (VA) charakteristiku dvoch kovových vlákien

V ostatných prípadoch, keď nemôžeme použiť vyrovnávajúci počet, nemáme k dispozícii ani opodstatnený návod ako preložiť čiaru cez namerané body, tu záleží veľa od skúsenosti experimentátora. Oblasti zaťažené veľkými chybami sa premerajú znova, hustejšie resp. inými metódami. Čiaru, ktorú narysujeme, sa snažíme viesť tak, aby bola vyrovnaná, t. j. nemala fyzikálne neopodstatnené skoky, zalomenia a extrémy, aby bola dostatočne hladká, aby približne rovnaký počet nameraných bodov bol nad i pod čiarou a súčet štvorcov nameraných hodnôt od čiary by mal byť čo najmenší. Majme stále na pamäti, že čiara v takomto grafe má viac-menej kvalitatívny význam.

Pri dôslednejších experimentoch sa merania v každom bode opakujú za rovnakých podmienok a každý bod v grafe je spracovaný vyššie opísanými metódami pre opakované merania. V takýchto prípadoch sa zvykne okrem najpravdepodobnejšej hodnoty (nameranej hodnoty) vyznačiť v grafoch aj štandardná neistota pre každý bod zvlášť.

Zhrnutie

- Obrys grafu nesmie byť nikdy nakreslený hrubšou čiarou, ako čiary v ploche grafu, taktiež úsečky, ktoré vyznačujú chyby meraných hodnôt, nemôžu byť výraznejšie ako vlastné krivky alebo priamky. Kóty na osiach musia udávať ľahko deliteľné hodnoty.
- Do grafu umiestňujeme čo najviac informácií, menej do legendy grafu.
- Dbáme na prehľadnosť grafu a čitateľnosť písma v grafe. Na popis osí sa častejšie používajú verzálky (veľké písmená), pre informácie vpísané do grafu mínusky (malé písmená) písma z rodiny Sans Serif.
- Osi grafu nemajú byť dlhšie, ako určuje výskyt pokusných bodov, v grafe teda *nemajú byť prázdne plochy*. Osi nemusia začínať nulovou hodnotou.
- Vhodným tvarom grafu je obvykle štvorec alebo obdĺžnik (na ležato). Uzavretie grafu do štvorca alebo obdĺžnika zjednodušuje určenie hodnôt jednotlivých bodov. Kóty môžeme na protiľahlých osiach opakovať bez doplnenia čísel.
- Poznáme šesť hlavných druhov (typov) grafov:
  - bodový garf (scattergram),
  - čiarový (priebehový) graf (line graph),
  - stĺpcový graf (bar graph),
  - histogram (vlastne stĺpcový graf so stĺpcami umiestnenými tesne vedľa seba),
  - koláčový diagram (pie graph),

  - trojrozmerný graf (three-dimensional graph).

- Dbáme na to, aby v bodovom a čiarovom grafe boli symboly a charakter jednotlivých čiar ľahko odlíšiteľný aj pri zmenšení tlače. V grafoch pripravovaných na počítači je potrebné správne zadať požadované vzdialenosti a popis kót, zvoliť len výrazné symboly, snažiť sa všetko vyjadriť jednou farbou a pod.

Záver

Keď hovoríme o príprave experimentálnych dát na prezentáciu a ďalšie vyhodnocovanie s použitím osobného počítača, potom samozrejme musíme venovať náležitú pozornosť nielen samotným programom, ale aj metódam a postupom spracovania dát.

Opis dvoch známych produktov z tejto oblasti nám v základoch objasnil ich všeobecné aj niektoré špecifické vlastnosti. Domnievame sa, že prvoradým prínosom sú základné informácie o ovládaní opísaných programov a získane poznatky, ako tvoriť grafické výstupy matematických funkcií a spracovaných experimentálnych dát na ďalšiu kvalitatívnu analýzu prípadne prezentáciu.

Tabuľka 2: Porovnanie parametrov fitovania pre referenčné dáta NIST s hodnotami získanými z programov QtiPlot a Kpl, a a b sú odhadované parametre, σ_a a σ_b sú štandardné neistoty (smerodajné odchýlky) odhadovaných parametrov, `RSD` je reziduálna štandardná odchýlka (*Residual Standard Deviation*), `SQ` je suma štvorcov odchýlok (*Sum of Squres*), `Chi^2/doF` je redukovaná hodnota χ^2 a `doF` znamená *Degrees of Freedom* čiže $n-k$

		NIST	QtiPlot	Kpl
NoInt1 $n-k=10$	a	2,074 38	2,074 38	2,074 38
	σ_a	0,016 53	0,004 63	0,016 53
		`RSD` = 3,567 53	`Chi^2/doF` = 12,727 3	`chi-square` = 127,3
		`SQ` = 127,272 72	`Chi^2` = 127,273	
BoxBOD $n-k=4$	a	213,809 41	213,809 53	213,809 00
	σ_a	12,354 52	0,722 99	12,354 50
	b	0,547 24	0,547 24	0,547 24
	σ_b	0,104 56	0,006 12	0,104 56
		`RSD` = 17,088 07	`Chi^2/doF` = 292,002	`chi-square` = 1168,008 876
		`SQ` = 1 168,088 77	`Chi^2` = 1 168,008	

Podklady k tomuto príspevku boli z veľkej časti čerpané z práce autora na projekte KEGA *Využitie* OPENSOURCE *softvéru vo výučbe na vysokých školách*. Cieľom nebolo vykonanie nejakej recenzie, na základe ktorej by sa dali oba programy rigorózne ohodnotiť. Každý z prezentovaných programov má svoje prednosti aj nedostatky (stále sa vyvíjajú a vylepšujú). Porovnanie výsledkov uvedených v tabuľke 2 má čitateľovi oboznámenému s funkčnosťou a hlavnými možnosťami programov uľahčiť rozhodovanie sa, ktorému z nich dá prednosť pri výbere. Pozorný čitateľ, ktorý vyskúšal program QtiPlot podľa nášho postupu (alebo stačí nazrieť do tabuľky 2) si isto všimne, že hodnoty štandardných neistôt, ktoré program vypočíta sú rádovo rozdielne od údajov inštitútu NIST. Je to spôsobené tým, že program QtiPlot počíta redukovanú hodnotu χ^2 (pozri vzťah 46) označenú ako `Chi^2/doF`, kde `doF` znamená *Degrees of Freedom* čiže $n-k$ a štandardná neistota parametra je určená podľa vzťahu

$$\sigma^{\mathrm{qti}} = \sqrt{\frac{(\mathrm{cov})_{ii}}{\mathrm{Chi^2/doF}}}\,. \tag{58}$$

Na WWW stránke inštitútu NIST sa však dočítame, že ich údaj štandardnej neistoty parametra sa počíta podľa vzťahu

$$\sigma^{\mathrm{nist}} = \sqrt{(\mathrm{cov})_{ii}}\,, \tag{59}$$

kde $(\mathrm{cov})_{ii}$ je v oboch prípadoch kovariančná matica parametrov regresie, pozri napr. v prácach (PRESS ET AL., 1992; KUDRACIK, 1999). Pri rovnosti kovariančných matíc, potom súvis oboch údajov môžeme vyjadriť vzťahom

$$\sigma^{\mathrm{nist}} = \sigma^{\mathrm{qti}} \sqrt{\mathrm{Chi\hat{}2/doF}}. \tag{60}$$

Keď teda potrebujeme výsledok numerického spracovania dát regresiou programom QtiPlot uviesť so štandardnou neistotou hľadaných parametrov, musíme tento „nedostatok“ výpočtu programu korigovať použitím vzťahu (60), štandardná neistota parametra regresie sa uvádza v takom tvare, ako na WWW stránke inštitútu NIST.

Mali sme možnosť pracovať aj s programom Origin 6.1[53], ktorému sa opisovaná verzia QtiPlot 0.8.5 svojími možnosťami a ponukou najviac približuje. Čo sa týka rozdielu z pohľadu bežného používateľa, QtiPlot má menší výber formátov do grafického výstupu. Nepokladáme to ale za taký veľký nedostatok. Origin 6.1 má však lepšie vypracované možnosti napr. ponuky Analysis v grafickom móde a rozšírenejšiu ponuku modulu Non-Linear Curve Fit. . . , lepšiu 3D grafiku a iné, ktoré nám však pri štandardnej práci s programom nebudú chýbať. Ako sme už spomenuli, QtiPlot je vo vývoji a neustále sa vylepšuje. V prípade uvádzania štandardných neistôt parametrov regresie program Origin ich uvádza v takej forme, ako inštitút NIST. Tento rozdiel medzi programami je snáď jediný vážny nedostatok, s ktorým sme sa počas práce s programom QtiPlot stretli.

Záverom ešte jeden postreh, zo skúsenosti odporúčame otvárať uložené projekty programu Kpl klikom[54] na súbor s príponou plo. Pri otváraní projektu cez hlavné menu File → Open Plot File . . . sa grafy v niektorých prípadoch nezobrazia presne tak, ako boli uložené (trochu sa posunú vložené texty, legendy a pod.). Tieto nedostatky sú síce formálneho charakteru, lebo graf ľahko upravíte do pôvodneho stavu (ak si ho ešte s odstupom času pamätáte :-), ale dokážu znepríjemniť pôžitok z už vykonanej práce.

Učenie a bádateľská práca je zaujímavá, často aj vzrušujúca činnosť. Keď ju vykonávame deň čo deň tvorivo s láskou aj ako záľubu, prináša nám osobnú radosť i duševné uspokojenie. Nevyhneme sa však pritom ani rutinnej a mechanickej práci, ktorú môže počítač v značnej miere uľahčiť.

[53] Komerčný program, cena aktuálnej verzie Origin 7.5 je asi 19 000,– SKK bez DPH.
[54] Alebo dvojklikom, podľa distribúcie a grafického prostredia OS GNU/Linux.

CHYBY ELEKTRICKÝCH MERACÍCH PRÍSTROJOV

Chyby analogových meracích prístrojov

Pre praktickú potrebu bola zvolená a normovaná charakteristika nazývaná *trieda presnosti* δ_{TP}. Trieda presnosti zahrňa všetky chyby samotného prístroja a definuje tak medznú (maximálnu, dovolenú) relatívnu chybu v celom meracom rozsahu prístroja

$$\delta_{\mathrm{TP}} = \frac{|\Delta_{\mathrm{max}}|}{X_{\mathrm{mr}}}\, 100\,(\%), \tag{61}$$

kde Δ_{max} je medzná (maximálna) absolútna chyba prístroja a X_{mr} je najväčšia hodnota meracieho rozsahu. *Merací rozsah* je časť stupnice meracieho prístroja, na ktorej je možné merať s predpísanou presnosťou. Najväčšia hodnota meracieho rozsahu X_{mr} je určená

- hornou hranicou meracieho rozsahu (keď je dolná hranica nula),
- súčtom oboch medzných hraníc (keď je nula uprostred stupnice),
- rozdielom hornej a dolnej hranice (keď je potlačená nula na stupnici).

Keď má prístroj určitú tiedu presnosti je tým definovaná jeho maximálna dovolená relatívna chyba vyjadrená v % najväčšej hodnoty meracieho rozsahu. Trieda presnosti je uvedená na číselníku každého analogového meracieho prístroja. *Maximálnu absolútnu chybu prístroja* možeme vyjadriť vzťahom

$$\Delta_{\mathrm{max}} = \pm\frac{X_{\mathrm{mr}}}{100}\,\delta_{\mathrm{TP}}. \tag{62}$$

Relatívna chyba meraného údaja je

$$\delta_{\mathrm{rel}} = \pm\frac{\Delta_{\mathrm{max}}}{X_{\mathrm{mh}}}\,100 = \pm\delta_{\mathrm{TP}}\,\frac{X_{\mathrm{mr}}}{X_{\mathrm{mh}}}\,(\%),$$

kde X_{mh} je nameraná hodnota. Z posledného vzťahu je vidieť, že čím menšia je meraná hodnota (čím menšia je výchylka prístroja), tým väčšia bude relatívna chyba merania. Z toho vyplýva, že pri meraní analogovými meracími prístrojmi musíme voliť taký rozsah prístroja, aby jeho výchylka bola čo najväčšia!

Príklad A

Analogovým voltmetrom s triedou presnosti $\delta_{\mathrm{TP}} = 1$ sme namerali na rozsahu $X_{\mathrm{mr}} = 60\,\mathrm{V}$ napätia 58 V a 5 V.

Absolútna chyba je pri všetkých meraniach rovnaká a je daná triedou presnosti použitého voltmetra

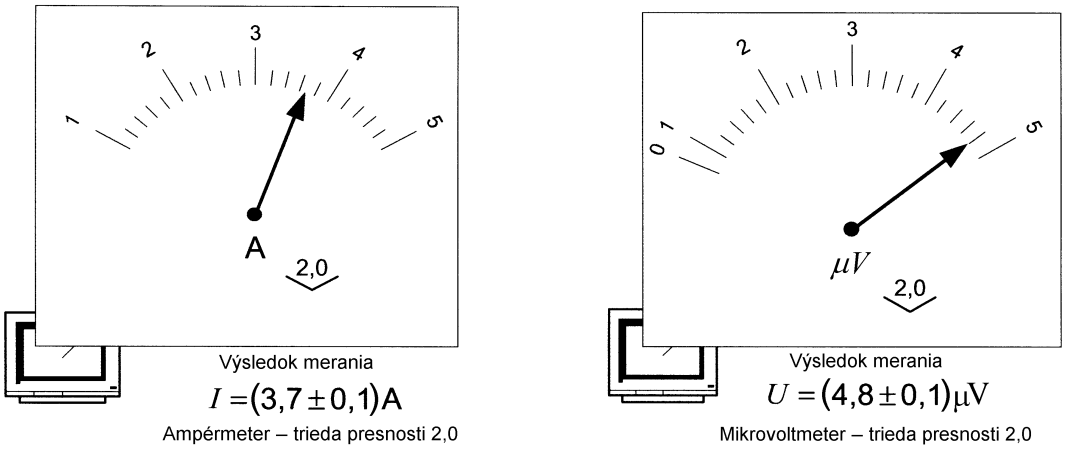
$$\Delta_{\mathrm{max}} = \pm\frac{X_{\mathrm{mr}}}{100}\,\delta_{\mathrm{TP}} = \pm\frac{60}{100}\,1 = \pm 0{,}6\,\mathrm{V}.$$

To znamená, že prístroj meria s presnosťou $\pm 0{,}6$ V na celej stupnici pri meraní napätia 58 V aj pri meraní napätia 5 V. Veľkosti relatívnych chýb údajov budú

$$\delta_{\mathrm{rel}} = \pm\delta_{\mathrm{TP}}\,\frac{X_{\mathrm{mr}}}{X_{\mathrm{mh}}} = \pm 1\,\frac{60}{58} = \pm 1{,}03\,\%,$$

$$\delta_{\text{rel}} = \pm\delta_{\text{TP}}\,\frac{X_{\text{mr}}}{X_{\text{mh}}} = \pm 1\,\frac{60}{5} = \pm 12\,\%.$$

Vidieť, že so znižovaním výchylky relatívna chyba údaja rýchle rastie, jej závislosť od výchylky je hyperbolická!



Obrázok 34: Na obrázku vidíme príklady zobrazenia hodnoty prúdu a napätia analogovými meracími prístrojmi s triedou presnosti 2,0. Maximálna absolútna chyba hodnoty merania bola vypočítaná pomocou vzťahu (62)

Chyby číslicových meracích prístrojov

Číslicové (digitálne meracie prístroje) merajú pomerne dobre len jednosmerné napätia a prúdy, ostatné veličiny s niekoľkonásobne väčšiou chybou ako presné analogové (ručičkové) meracie prístroje, pretože sa u týchto prístrojov všetky merané veličiny prevádzajú pomocou usmerňovača na jednosmerné napätie. Usmernené napätie sa ďalej digitalizuje pomocou analogovo-číslicového prevodníka (AD). AD prevodníky vnášajú do merania ďalšie chyby. Nemá teda zmysel overovať triedu presnosti analogového meracieho prístroja pomocou bežného vreckového multimetra!

Väčšina výrobcov číslicových prístrojov uvádza presnosť prístroja (tzv. *základnú chybu*) v tvare $\delta_{\text{čmp}} = \pm(\delta_{\text{mh}} + d)$, niektorí v tvare $\delta_{\text{čmp}} = \pm(\delta_{\text{mh}} + \delta_{\text{mr}})$, kde

- δ_{mh} je chyba z nameranej hodnoty, býva vyjadrená v % a je v celom meracom rozsahu konštantná, niekedy sa za ňu pripisuje značka rdg (reading–čítanie),
- δ_{mr} je chyba z meracieho rozsahu, nemôžeme ju však jednoducho sčítať s chybou z nameranej hodnoty δ_{mh}, ale ju musíme prepočítať na veľkosť nameranej hodnoty $\left(\delta_{\text{mr}}\,\frac{X_{\text{mr}}}{X_{\text{mh}}}\right)$; niekedy sa za ňu pripisuje značka FS (full scale–plný rozsah),
- d je chyba udaná z počtu jednotiek (digitov) posledného miesta displeja. Jej prepočet na chybu z meracieho rozsahu závisí od počtu zobrazovaných miest displeja. Prepočet na percentuálnu chybu z meracieho rozsahu je rovný $\delta_{\text{mr}} = \frac{d}{\text{max. počet indikovaných jednotiek}}\,100\ (\%)$.

Celková relatívna chyba číslicového meracieho prístroja je pri meraní vyjadrená vzťahom

$$\delta_{\text{rel}} = \pm\left(\delta_{\text{mh}} + \delta_{\text{mr}}\,\frac{X_{\text{mr}}}{X_{\text{mh}}}\right)\ (\%), \tag{63}$$

kde X_{mr} je hodnota meracieho rozsahu a X_{mh} je nameraná hodnota.

Súčasné číslicové meracie prístroje majú automatické prepínanie rozsahov, aby bola pri meraní vždy dosiahnutá maximálna presnosť. Podľa maximálneho počtu zobrazených miest zistíme, na ktorom rozsahu multimeter práve meria. Napr. multimeter s maximálnou hodnotou 3 999 prepína pri meraní automaticky rozsahy 400 mV – 4 V – 40 V – 400 V. Multimeter s maximálnou hodnotou 1 999 prepína pri meraní automaticky rozsahy 200 mV – 2 V – 20 V – 200 V. Prepínanie rozsahov na meranie ostatných veličín prebieha podobne. Samozrejme chyby každého multimetra pre jednotlivé rozsahy najdete v návode na používanie meracieho prístroja.

Príklad B

Číslicový voltmeter má na rozsahu 40 V základnú chybu $\pm(0{,}9\,\mathrm{rdg}+0{,}1\,\mathrm{FS})$. Máme zistiť, relatívnu chybu nameraných napätí $U_1 = 10$ V a $U_2 = 28$ V na tomto rozsahu.

$$\delta_{\mathrm{rel}}(U_1) = \pm\left(\delta_{\mathrm{mh}} + \delta_{\mathrm{mr}}\,\frac{X_{\mathrm{mr}}}{U_1}\right) = \pm\left(0{,}9 + 0{,}1\,\frac{40}{10}\right) = \pm 1{,}3\,\%,$$

$$\delta_{\mathrm{rel}}(U_2) = \pm\left(\delta_{\mathrm{mh}} + \delta_{\mathrm{mr}}\,\frac{X_{\mathrm{mr}}}{U_2}\right) = \pm\left(0{,}9 + 0{,}1\,\frac{40}{28}\right) = \pm 1{,}04\,\%.$$

Príklad C

Chyba číslicového multimetra s $3\,\frac{1}{2}$ miestnym displejom (maximálna indikovaná hodnota je 1 999) je pre meranie striedavého prúdu udaná v tvare $\delta_{\mathrm{mh}} = \pm(1{,}5\,\% + 7\,\mathrm{dibit})$[55]. Máme zistiť veľkosť relatívnej chyby multimetra, keď meriame na rozsahu 40 A prúd 6 A.

Maximálny počet indikovaných jednotiek je 2 000.

$$\delta_{\mathrm{mr}} = \frac{d}{\text{max. počet indikovaných jednotiek}}\,100 = \frac{7}{2\,000}\,100 = 0{,}35\,\%.$$

Celková chyba má tvar $\pm(1{,}5\,\% + 0{,}35\,\mathrm{FS})$.

Relatívnu chybu určíme zo vzťahu

$$\delta_{\mathrm{I}} = \pm\left(\delta_{\mathrm{mh}} + \delta_{\mathrm{mr}}\,\frac{X_{\mathrm{mr}}}{X_{\mathrm{mh}}}\right) = \pm\left(1{,}5 + 0{,}35\,\frac{40}{6}\right) = 3{,}83\,\%.$$

[55] dibit je kombinácia (skupina) dvoch binárnych čísiel (digitov) do jednej alebo štyroch kombinácií. Štyri možné stavy pre dibit sú 00, 01, 10 a 11.

Použitá literatúra

BRANDEJS, M. 2003. *Linux – Praktický průvodce*. Brno : Konvoj, 2006, 2. vydanie, ISBN 80-7302-050-5

BRUNOVSKÁ, A. 1990. *Malá optimalizácia.* Bratislava : Alfa, 1990, ISBN 80-05-00770-1

BUŠA, J. 2006. *Octave – Rozšírený úvod*. Košice, 2006, ISBN 80-8073-595-6

DÁVID, A. 1988. *Numerické metódy na osobnom počítači*. Bratislava : Alfa, 1988

GARCIA, A., L. 2000. *Numerical Methods for Physics*. New Jersey : Prentice-Hall, 2000, ISBN 013--906744-2

HOFMANN, D. 1988. *Priemyselná meracia technika*. Bratislava : ALFA. 1988, ISBN 80-05-00139-8

KAUKIČ, M. 1998. *Numerická analýza I. Základné problémy a metódy*. Žilina : MC Energy s. r. o. 1998

KAUKIČ, M. 2006. *Základy programovania v PyLabe.* Košice, 2006, ISBN 80-8073-634-0

KUBÁČEK, L. – KUBÁČKOVÁ, L. 2000. *Statistika a metrologie.* Olomouc : Univerzita Palackého v Olomouci, 2000, ISBN 80-244-0093-6

KUDRACIK, F. 1999. *Spracovanie experimentálnych dát.* Bratislava : Univerzita Komenského, 1999, ISBN 80-223-1327-0

LYONS, L. 2001. *A practical quide to data analysis for physical science students.* Cambridge : Cambridge University Press, 2001, ISBN 0-521-42463-1

MELOUN, M. – MILITKÝ, J. 2004. *Statistická analýza experimentalních dat.* Praha : Academia, 2004, ISBN 80-200-1254-0

MOLER, C. B. 2004. *Numerical Computing with MATLAB.* Philadelphia : SIAM, 2004, ISBN 0-89871--560-1

NIST 2006. *National Institute of Standards and Technology. Statistical reference Datasets*. `http://www.itl.nist.gov/div898/strd/general/dataarchive.html`

PALENČÁR, R. – VDOLEČEK, F. – HALAJ, M. 2000. *AUTOMA*, č. 7–8, 2000, str. 50–54, `http://www.automa.cz/`

PAZOUREK, J. 1992. *Simulace biologických systému*. Praha : GRADA, 1992, ISBN 80-85623-13-7

PETROVIČ, P. – NADRCHAL, J. – PETROVIČOVÁ, J. 1989. *Programovanie a spracovanie dát I., II.* Košice : Edičné stredisko UPJŠ, 1989

PIRČ, V. – BUŠA, J. 2002. *Numerické metódy*. Košice : elfa, 2002, ISBN 80-89066-25-9

PRESS, W. H. et al. 1992. *Numerical Recipes in C – The Art of Scientific Computing*. New York : Cambridge University Press, 1992, 2nd Ed. Kniha v PDF formáte je dostupná na URL adrese: `http://www.nrbook.com/b/bookcpdf.php`

RIEČANOVÁ, Z. a kol. 1987. *Numerické metódy a matematická štatistika*. Bratislava : ALFA, 1987

SQUIRES, G. L. 2001. *Practical Physics.* Cambridge : Cambridge University Press, 2001, ISBN 0-521-77940-5

ŠESTÁK, Z. 2000. *Jak psát a přednášet o věde*. Praha : Academia, 2000, ISBN 80-200-0755-5

UHRIN, J. – ŠEVČOVIČ, L. – MURÍN, J. 2006. *Fyzikálne merania*. Košice : elfa, 2006, ISBN 80-8086-032-7

VDOLEČEK, F. – PALENČÁR, R. – HALAJ, M. 2001(a). *AUTOMA*, č. 10, 2001, str. 52–56, `http://www.automa.cz/`

VDOLEČEK, F. – PALENČÁR, R. – HALAJ, M. 2001(b). *AUTOMA*, č. 12, 2001, str. 28–33, `http://www.automa.cz/`

VDOLEČEK, F. – PALENČÁR, R. – HALAJ, M. 2002(a). *AUTOMA*, č. 4, 2002, str. 41–47, `http://www.automa.cz/`

VDOLEČEK, F. – PALENČÁR, R. – HALAJ, M. 2002(b). *AUTOMA*, č. 5, 2002, str. 42–45, `http://www.automa.cz/`

ZVÁRA, K. – ŠTĚPÁN, J. 2001. *Pravděpodobnost a matematická statistika*. Bratislava : VEDA, 2001, ISBN 80-2240736-4

WIMMER, G. – PALENČÁR, R. – WITKOVSKÝ, V. 2001. *Stochastické modely merania*. Bratislava : Grafické štúdio Ing. Peter Juriga, 2001, ISBN 80-968449-2-X