# Grafické zpracování dat a měření

pomocí GNUPLOTu

Lukáš L. & Pája B.
lukasl@fykos.cz, paja@fykos.cz

26. prosince 2012 – 17:54

## 0 Jak číst tento manuál

Při prvním čtení nečtěte poznámky pod čarou, jsou zde většinou uvedeny rozšiřující informace, které nejsou potřebné pro základní pochopení textu, ale obsahují informace užitečné při dalším studiu. Vřele doporučujeme si zde uvedené příkazy ihned zkoušet v již nainstalovaném GNUPLOTu.

Pokud je nějaká část textu označena jako tento odstavec, znamená to, že popsaný způsob je funkční v Linuxu, ale nemusí být funkční v ostatních operačních systémech nebo se v nich může používat jinak. #Linux

## 1 Grafické zpracování pomocí GNUPLOTu – přehled příkazů

Gnuplot — volně šiřitelný program vyvinutý na univerzitě pro potřeby studentů a výuky (vizualizace závislostí apod.), s pestrými možnostmi a pěkným manuálem; vše lze stáhnout z `www.gnuplot.info`. Ovládá se téměř výhradně z příkazové řádky.

Program není nutné instalovat. Lze jej spustit ve většině OS; při spuštění (volání) s řídicím skriptem[1] se provedou všechny příkazy skriptu, po dokončení nebo při první jakékoliv chybě se takto spuštěný GNUPLOT ukončí. Není to jediná možnost použití, příkazy je možné zadávat přímo, tj. bez použití skriptovacího souboru[2], ale je to velmi **nedoporučovaná** možnost, protože výsledek není reprodukovatelný.

### 1.1 Základní příkazy

Klíčová slova v příkazech sice není třeba vypisovat celá, postačí jednoznačný začátek, doporučujeme je však uvádět celá. Pokud by v nové verzi byl přidán nový příkaz, tak by náš skript již nemusel být funkční.

`reset` ..... uvede GNUPLOT skoro do počátečního stavu. (Jediné, co se příkazem `reset` nenastavuje, je `terminal` a `output`, dále se také neruší inicializace proměnných a funkcí, viz dále.) *Měl by být pro jistotu uveden na začátku každého skriptu.*

`cd 'C:/fyzika/fykos'`
: změní pracovní adresář, kde jsou hledány datové soubory a ukládány výstupní soubory[3]. Při práci lze užívat též relativní cesty vztažené k tomuto pracovnímu adresáři. Je doporučeno používat jednoduché uvozovky `'`, při použití `"` je `\` řídícím znakem a je nutné jej dvojit – v Linuxu i ve Windows se používá pro oddělení adresářů v cestě `/`.

  V Linuxu je pracovním adresářem místo, odkud byl GNUPLOT spuštěn. #Linux

  Ve Windows lze též měnit pracovní adresář kliknutím na `Ch. dir.` v horní části okna.

`pwd` ..... print working directory, vypíše současný pracovní adresář, analogicky k příkazové řádce Linuxu.

`load 'skript.gp'`
: načte a vykoná příkazy uvedeného skriptu, viz pozn. pod čarou 3. V případě chyby provádění skriptu ukončí a napíše, co se mu nelíbí ⇒ používá se k ladění; po opravě je nutné skript spustit celý znovu. V linuxu je možné volání z příkazové řádky `gnuplot 'jmeno_skriptu'`, vhodné pro dávkové zpracování, např. pomocí `make`.

`save 'jmeno_souboru.plt'`
: uloží aktuální stav GNUPLOTu , tj. uživatelsky nadefinované proměnné, funkce, všechna nastavení provedená pomocí `set` a poslední příkaz `plot`, resp. `splot`.

[1] V Linuxu stačí napsat do příkazové řádky `gnuplot 'jmeno.skriptu'` a skript se provede, ale GNUPLOT ihned skončí. Jak tomu zabránit – viz sekce 1.1.

[2] Skriptovací soubor je normální textový soubor, ve kterém je na každé řádce uveden jeden příkaz pro GNUPLOT.

[3] Další lokace, kde GNUPLOT hledá skripty, je uložena v proměnné GNUPLOTu `loadpath`, která se inicializuje při spuštění ze systémové proměnné `GNUPLOT_LIB`. #Linux

`pause n "msg"`
: čeká $n$ sekund a pak pokračuje, je-li uveden nepovinný parametr `msg` (uvozovky nutné, jinak bude řetězec považovat za proměnnou, tak vypíše zprávu). Pro $n = -1$ čeká na akci uživatele (kliknutí na „OK“, popřípadě stisknutí „enter“ v příkazové řádce).

`system("command")`
: vyvolá shell a provede příkaz `command`. Všechny příkazy uvedené za `!` budou též předány shellu. #Linux

`^C` ..... ukončí právě prováděnou operaci (skript).

`exit` nebo `^D` ... ukončí GNUPLOT.

`\` *na konci řádku*
: dovoluje pokračovat v zadávání příkazu na dalším řádku (lze užít pro zpřehlednění skriptů). Pozor, ve skriptu se již nesmí za `\` nacházet žádný znak kromě nového řádku, jinak GNUPLOT nahlásí chybu.

`#` ..... uvozuje jednořádkový komentář, je vřele doporučováno složitější skripty komentovat. (Znak uvozující komentář lze nastavit příkazem `set datafile commentschar "m"`.)

*šipky nahoru a dolů*
: umožní listovat historií zadaných příkazů (pro urychlení zadávání a případné změny příkazů).

## 1.2 Definice funkcí a proměnných

Lze definovat bezparametrické funkce – proměnné příkazem[4] `a = 2`. Proměnná (funkce) nesmí mít stejný název jako nějaké klíčové slovo GNUPLOTu. Dále lze analogicky definovat funkce s libovolným počtem argumentů, např. `f(x,y,z) = x+y+z`. GNUPLOT podporuje přetěžování různým počtem argumentů, nikoli však různými typy.

Při definici funkcí lze používat základní matematické operace jako `+, -, *, /`, dále pro umocňování `**`. GNUPLOT podporuje i bitové operace nad čísly `&, |, ^`. Pro změnu priority operací se používají `( )`. GNUPLOT umí pracovat i s řetězci[5] Konstanta π je ukryta pod klíčovým slovem `pi`. Dále jsou k dispozici všechny „normální“ matematické funkce jako `sin, cos, tan, exp,...` Přirozený logaritmus se počítá pomocí funkce `log`, desítkový logaritmus se skrývá pod `log10`.

Je potřeba dávat pozor na vyhodnocování aritmetických operací. GNUPLOT se chová obdobně jako jazyk C, tj. `1/2 = 0`, `3/2 = 1`, ale `1./2 = 0.5`. (Pokud dělíme dvě „celá čísla“, tak probíhá celočíselné dělení, pokud je alespoň jeden ze činitelů číslem desetinným, tak výsledkem je opět číslo desetinné.)

## 1.3 Příkazy pro kreslení grafů funkcí

Chceme-li cokoli GNUPLOTem vykreslit, musíme použít příkaz `plot`. Dále rozebereme jednotlivé parametry příkazu `plot`.

`plot acos(x)` ... základní verze příkazu — vykreslí graf funkce, v tomto případě arkuskosinus $x$.

`plot [-1:1] exp(x), cosh(x)`
: nyní nakreslí exponenciálu a cosinus hyperbolický (jednotlivé funkce jsou odděleny pomocí čárek) pro $x \in (-1, 1)$. Standardní rozsah je $x \in (-10, 10)$. Rozsah na ose $y$ se volí tak, aby obsáhl všechny funkční hodnoty kreslených funkcí. Není-li mezi `[ ]` a `:` uvedena žádná hodnota, použije se hodnota uložená pomocí `set ?range`, kde `?` značí `x,y,z`, viz sekce 1.6[6].

`plot [-pi-0.1:pi+.1][-3:] a=1, 2*sin(x-a)`
: nastaví rozsahy hodnot pro osy $x$ a $y$. Také je zde vidět, že `a=1` provede nastavení hodnoty parametru `a`, ale nic nevykreslí.

  Nastavování parametru je vhodné, pokud chceme do grafu vykreslit nějakou ve skriptu výše definovanou funkci pro více hodnot parametru. Je potřeba nejdříve nastavit hodnoty parametrů a pak teprve zapsat příkaz pro kreslení.

(*podm*`?`*true*`:`*false*)
: je tzv. ternární operátor. Nejdříve se vyhodnotí podmínka *podm*, což může být např. `x<2 && x>-1`, tj. je splněna pro $x \in (-1, 2)$, je-li splněna, použije se výraz *true*, v opačném případě se použije výraz *false*. Tato konstrukce se hodí použít, pokud chceme např. vykreslit funkci pouze na nějakém intervalu, potom místo hodnoty *false* použijeme `0/0` neboli nedefinovaný výraz.

Pokud používáme implicitní terminál (ve Windows `win`, v Linuxu `x11`), je možno pomocí pravého tlačítka myši zoomovat graf. Pokud však byly rozsahy nastaveny pomocí `[ ]` za příkazem `plot`, tato možnost nefunguje — je nutno použít `set ?range`.

[4] Není dobré pojmenovávat proměnné `x, y, z, t, u, v`, protože tyto se používají při kreslení.

[5] Řetězce se spojují pomocí operátoru `.`, lze je porovnat pomocí `eq` a `ne` (vrací hodnoty *true* a *false*). Délka řetězce se zjišťuje pomocí funkce `strlen("str")`, podřetězec získáme pomocí `substr("str", i, j)`, vyhledávat můžeme pomocí `strstrt("str", "key")` (vrací pozici prvního výskytu klíče).

[6] Případně třetí pár `[]` nastavuje rozsah osy $z$ ve 3D režimu. V režimu `set parametric` se nastavují ještě rozsahy parametru(ů), viz dále.

## 1.4 Struktura datového souboru (souboru hodnot)

Do datového souboru ukládáme zpracovávaná data[7], tj. většinou data získaná z experimentu, popřípadě již částečně zpracovaná pomocí např. Libre–Calc nebo MS–Excel.

Místo desetinné čárky se musí používat tečka, ale je dovolený zápis `1.3e-4`. Proto je potřeba použít *Edit/Replace* např. po zkopírování hodnot z Calcu.

Doporučujeme vkládat do datových souborů veličiny v základních jednotkách, velmi to následně usnadňuje práci (hlavně pokud data dále fitujeme, dostaneme koeficienty též v základních jednotkách). Pokud chceme vykreslovat graf v jednotkách odvozených, tj. např. mV místo V, není problém veličinu vynásobit/vydělit odpovídající konstantou, tj. za klíčové slovo `using` napsat místo `1:3` o něco delší `($1*1000):3`.

Hodnoty jsou uspořádány do sloupců, které jsou oddělené pomocí pomocí „bílého znaku(ů)“, tj. mezera a tabulátor. Příklad je uveden na obrázku 1.

`#` ..... na začátku řádku znamená vykomentování řádku[8], tzn. řádek se nečte; slouží k uvedení legendy a jednotek, vyřazení neplatných bodů[9] apod.

```
# cm  I[A]     U[V]    Phi[W]     DPhi[W]    h[%]    Dh[%]
1     0.0e-3   0.010   0.000e-3   -          -       -
2     5.0e-3   1.407   0.007e-3   0.001e-3   0.097   0.009
3    10.0e-3   1.459   0.008e-3   0.002e-3   0.055   0.016
4    16.4e-3   1.507   0.016e-3   0.003e-3   0.065   0.011
```

Obrázek 1: Příklad datového souboru.

## 1.5 Příkazy pro kreslení dat

Jako v části 1.3, půjde dále také o popis možností příkazu `plot`. Nyní nebudeme vykreslovat analytické funkce, ale obsah datového souboru. Každému řádku v souboru odpovídá jeden bod na grafu.

`plot 'jmeno_DAT_souboru'`
: je základní příkaz pro kreslení. Vykreslí body o souřadnicích z prvních dvou sloupců hodnot — v prvním sloupci jsou $x$-ové hodnoty a ve druhém sloupci jsou $y$-ové hodnoty.

`plot 'data.dat' using 3:2`
: klíčové slovo `using` specifikuje sloupce použité ke kreslení — zde se $x$-ové hodnoty berou ze třetího sloupce a $y$-ové hodnoty z druhého sloupce v daném souboru (`data.dat`).

`plot 'data.dat' index 3 u 3:2`
: pokud si rozdělíme datový soubor na bloky oddělené alespoň dvěma prázdnými řádky, tak pomocí klíčového slova index můžeme vybrat pouze konkrétní blok dat. Prvnímu bloku odpovídá `index 0` na rozdíl od výběru sloupce, kde v „nultém“ sloupci se vyskytuje číslo řádku.

  Pomocí klíčového slova `index` lze vybrat i více bloků najednou. Syntaxe je pak následující: `index` *first*`:`*last*`:`*step*, kde *first* určuje první použitý blok, *last* poslední a *step* krok použitý při výběru, tj. pro *step = 1* je použije každý blok.

`plot 'data.dat' u 3:2 every ...`
: *syntaxe viz manuál* — vybírá pouze některé řádky z datového souboru.

`plot 'data.dat' using 1:6:7 with yerrorbars`
: vykreslí body včetně $y$-ových chybových úseček. První dvě čísla za `using` určují, které sloupce se mají použít jako souřadnice bodů, třetí hodnota specifikuje, ve kterém sloupci je uvedena chyba $y$-ové hodnoty, protože byla použita volba `yerrorbars`.

  Použijeme-li volbu `xerrorbars`, tak se budou vykreslovat $x$-ové chybové úsečky a GNUPLOT bude požadovat tři parametry za `using` (jinak použije sloupce 1, 2 a 3).

  Použijeme-li volbu `xyerrorbars`, tak se budou vykreslovat jak $x$-ové, tak $y$-ové chybové úsečky a GNUPLOT vyžaduje čtyři parametry[10].

[7] Na rozdíl od skriptu, kde popisujeme, jak data zpracovat a vykreslit z nich výstupní grafy.

[8] Tento parametr lze též přenastavit pomocí `set datafile commentschar "m"`.

[9] Neplatné body lze ignorovat i automaticky pomocí nastavení `set datafile missing "NaN"`. Toto nastavení je vhodné především, pokud generujeme datový soubor automaticky např. pomocí vlastního programu.

[10] Je též možnost nastavit nesymetrické chybové úsečky. Pokud vynášíme chybové úsečky pouze v jednom směru, tak je syntaxe následující: `plot␣`*příkaz pro kreslení*`␣using␣`*hod. x*`:`*hod. y*`:`*?error˙min*`:`*?error˙max*`␣with ?errorbars`. Pokud chceme nastavit minima i maxima $x$-ových i $y$-ových chybových úseček, je syntaxe stejná, ale musíme uvést celkem šest parametrů.

`plot 'data.dat' using (log($1)):2 with lines`
je možné do grafu vynášet i libovolnou funkci hodnot umístěných na řádku v datovém souboru. V tomto případě vykreslujeme na osu $x$ logaritmus prvního sloupce (právě použití `$` říká GNUPLOTu, že má použít hodnotu z prvního sloupce, nikoli konstantu `1`). Výraz je vždy nutno uzávorkovat, proto je zde použit vnější pár kulatých závorek. Samozřejmě, že je možno takto určovat i velikost chybové úsečky.

Volba `with lines`, resp. `w l` zajistí, že se nevykreslí body, ale lomená čára body spojující.

`plot 'data.dat' using 3:($4*1000)`
funkční hodnoty ve 4. sloupci se vynášejí 1000-krát větší, tzn. ne např. v V, ale v mV.

`plot 'data.dat' using 3:4 smooth csplines`
body proloží nejpěknější křivkou (splajny)[11], kterou spočte pro jejich souřadnice ve 3. a 4. sloupci. Samotné body se tímto nevykreslí, pouze ta křivka (místo `csplines` lze použít `unique`, `frequency`, `acsplines`, `csplines`, `bezier` nebo `sbezier`, některým z nich ale nestačí pouze souřadnice bodů, ale potřebují ještě další parametr, viz manuál, resp. 1.11.

`plot 'data.dat' using 1:6 smooth bezier, 'data.dat' using 1:6:($6-$7):($6+$7) with yerrorbars`
ideální příklad pro nastavení správného grafického znázornění (tentokráte s využitím Beziérovy křivky). Je zde také vidět, jak se používá nesymetrické nastavení chybových úseček (ačkoli zde jsou nastaveny symetricky) a také, jak lze konstruovat hodnoty z různých sloupců datového souboru.

`plot 'data.dat' using 1:2, a=4, a*x`
vykreslí body ze souboru a funkci $ax$ (pro zároveň daný parametr).

`plot 'data.dat' using 1:2, a=4, f(x)=a*x-6, f(x), a=5, f(x)`
vykreslí body s grafy dané funkce pro různé hodnoty parametru ($a = 4$ a $a = 5$). Pro nalezení správné hodnoty parametru využijeme fitování — viz dále, sekce 1.7.

## 1.6 Základní nastavení grafu (název grafu, popisky os apod.)

`set`..... obecně zapnutí nějaké funkce.

`unset` ..... obecně vypnutí nějakého nastavení.

`show`..... obecně výpis aktuálního nastavení na terminál.

`set title "Graf $y=\\sin(x_1^2)$"`
nastaví titulek grafu uprostřed nahoře. Zde již připravený pro výstup do LaTeXu.

`set xlabel "$U/\\jd{mV}$"`
nastaví popisek osy $x$ pro LaTeXový výstup. Předpokládá FYKOSí makra, ve kterých je obsaženo makro `\jd` pro sázení jednotek, resp. je lepší použít rovnou makro `\popi{}{}`, jehož první argument je veličina a druhý jednotka. Pro popisek osy $y$ doporučujeme volbu `norot`, která způsobí, že popisek nebude otočený.

`set xrange [-3:4]`
provede nastavení rozsahu hodnot na ose $x$. `*` změní nastavení na `auto`, prázdná volba ponechá předešlé nastavení. Hodí se volba `reverse`, která zařídí, že vlevo jsou větší hodnoty než vpravo. Analogicky pro osu $y$, parametry $t$, $u$ a $v$, ve 3D pro osu $z$ a barevný proužek $cb$.

`set key left top`
zapne legendu (umístění v grafu vlevo nahoře). Možné volby jsou `top`, `bot`, `left`, `right` a `center`.

`unset key`..... vypne zobrazování legendy. Toto nastavení nedoporučujeme, ale někdy se může hodit.

`set xtics`..... *syntaxe viz manuál* — vlastní nastavení dílků osy $x$; $y$ analogicky.

`set grid`..... *syntaxe viz manuál* — nastavení mřížky pro odečítání z grafu.

```
plot 'data.dat' using 1:2 smooth bezier title "exp. zavislost",\
     'data.dat' using 1:2 title "zmerene body",\
     f(x)=a*x-6,\
     a=4, f(x) title "1. pokus odhadu", \
     a=5, f(x) title "2. pokus odhadu"
```
Příklad nastavení popisků závislostí v legendě (`key`) a uplatnění `\` na koncích řádků ke zpřehlednění.

## 1.7 Fitování a grafické zpracování dat pomocí GNUPLOTu

Fitování je procedura, která nám umožní určit nepřímo měřené veličiny. Z teorie známe vztah, který svazuje dvě měřené veličiny, ale obsahuje nějaké neznámé (měřené) konstanty. Experimentálně proměříme tuto závislost a právě

[11] Splajny jsou po částech kubické funkce, které procházejí všemi body tak, aby měly spojité druhé derivace a lineární člen u krajích intervalů se pokládá roven nule. V mnoha případech je lepší Beziérova křivka, která nemusí procházet zadanými body.

fitování nám umožní zjistit hodnoty těchto konstant. Dále si popíšeme celý postup.

`set fit errorvariables`
: Během fitu vytvoří ke každému parametru `X`, přes který se fituje, proměnnou `X_err`, ve které je uložena absolutní chyba určení tohoto parametru. Nejde ještě o chybu určení veličiny, ta je větší – je potřeba přičíst ještě chybu přístroje, ale doposud jsem nenašel nikde postup, jak tuto chybu zohlednit.

`f(x) = A*x+B` ... nejdříve zadáme do GNUPLOTu funkční předpis s číselnými i obecnými koeficienty nějak pojmenovanými, např. `A`, `B`, `I_min`,... Dále je možno zadat počáteční hodnoty obecných koeficientů, jinak je GNUPLOT odhadne sám. Pro lineární fit není nutné počáteční hodnoty zadávat, ale pro nelineární jde téměř o nutnost. Doporučujeme si vykreslit změřené body a zkoušet různé hodnoty parametrů, až se bude prokládaná křivka alespoň řádově shodovat se změřenými daty, jinak může fit špatně zkonvergovat.

`fit f(x) 'data.dat' via A,B`
: příkazem `fit` nafitujeme data v uvedeném souboru danou funkcí přes specifikované parametry, nikoliv proměnnou x; ostatní koef. musí mít danou hodnotu. Zde fitujeme přes koeficienty `A` a `B`. Samozřejmě je dovolena a doporučována syntaxe `using` a `index`, jež vybírá vhodná data z datového souboru.

`plot 'data.dat' using 1:2, f(x)`
: vždy na grafu ověříme správnou konvergenci fitu. Je-li fit zavádějící, můžeme se pokusit odhadnout a zadat lépe nějaký koeficient, případně fitovat přes méně parametrů, tzn. ubrat parametry za `via`. Další možností neúspěchu může být nesprávně zvolená funkční závislost `f(x)`.

`set print "vysledky.txt"`
: přesměruje výstup do souboru `vysledky.txt`.

`print "A = ",A; print "A_err = ",A_err`
: vytiskne do souboru výsledky fitu včetně chyby.

`set print` ..... nastaví std. výstup (tj. výstup na konzoli) do původního stavu a uzavře výstupní soubor. Bez tohoto příkazu zůstane někdy výstupní soubor prázdný (naprosto náhodně)!!!

## 1.8 Kreslení více grafů do jednoho obrázku

`set multiplot`
: přenastaví GNUPLOT tak, aby bylo možno kreslit více grafů do jednoho obrázku.

`set size 1,0.5`
: zmenší oblast, do které bude vykreslován další graf. Šířka zůstane stejná, jako je šířka obrázku, výška je zmenšena na polovinu.

`set origin 0,0.5`
: nastaví souřadnici levého dolního rohu pro vykreslování dalšího grafu. V tomto případě dojde k vykreslení grafu do horní poloviny obrázku. (Levý dolní roh má souřadnici `0,0`.)

`plot` *příkazy pro kreslení*
: vykreslí graf o velikosti dané pomocí `set size` o počátku daném pomocí `set origin`. Pro nakreslení dalšího grafu opakujeme postup od bodu `set origin`, chceme-li ale různě veliké grafy, tak musíme též pozměnit velikost.

`unset multiplot`
: uvede GNUPLOT do původního stavu. Nyní každým vyvoláním příkazu `plot` dojde ke smazání předešlého grafu a nahrazení novým. Toto je jedno z mála nastavení, které je nutno navrátit zpět ručně, tj. nikoli pomocí `reset`.

## 1.9 Shrnutí příkazů a dalších možností programu GNUPLOT – pokročilé nastavení

`plot 'data.dat' with linespoints lt 3 lw 1 pt 2 ps 4 lc 5`
: jak již víme, vykreslí data ze souboru `data.dat`, a to body spojené lomenou čarou. Čára bude mít typ 3, tloušťku 1 a barvu 5, body budou typu 2 velikosti 4. U velikostí jsou dovolena i necelá čísla. Typ čáry a barva čáry začíná mít smysl pro barevný/černobílý grafický výstup.

  Pokud nepoužijeme žádné z výše uvedených nastavení, je velikost bodů 1, šířka čáry 1 a typy čar a bodů se zvětší vždy o 1 s každou další položkou v příkazu `plot`. Toto nastavení závisí na volbě terminálu. Některé terminály mají více různých typů čar, některé méně.

`set samples 2000`
: nastaví počet bodů, ve kterých bude určována funkční hodnota při kreslení analytické funkce. Implicitní hodnota je 100.

`set decimalsign ","`
: přenastaví desetinnou tečku na desetinnou čárku ve všech grafech.

`set arrow 1 from 2.1,3.8 to 3.3,2.6`
vykreslí šipku identifikovanou číslem `1` mezi zadanými souřadnicemi. Další nastavení viz manuál.

`unset arrow 1`
vypne zobrazování šipky definované výše.

`set label 1 "text"at 3,2.5`
umístí popisek na danou souřadnici, další nastavení a volby viz manuál.

`set encoding cp1250`
přepne vstupní kódování, nutné pro výstup do `jpg...`, aby fungovaly české znaky. UTF-8 bohužel není ještě plně implementované, ale pro výstup do LaTeXu můžeme nastavit libovolné kódování a i české znaky projdou bez problémů.

`set parametric`
se použije, pokud chceme vykreslit funkci, kterou máme zadánu paramtericky. Parametrem pro křivky je $t$, pro povrchy jsou $u$ a $v$. Syntaxe následující za příkazem `plot` či `splot` je stejná, ale pro vykreslení jedné křivky/povrchu potřebujeme dvě, resp. tři funkce definující funkční hodnoty pro jednotlivé souřadnice v závislosti na parametrech.

## 1.10 Nastavení grafického výstupu do souboru a zpět na monitor

### 1.10.1 Nastavení gnuplotu pro výstup do LaTeXu

`set terminal epslatex size 12cm,8cm`
nastavení výstupu do LaTeXu. Šířka by ideálně měla odpovídat šířce tiskového zrcadla LaTeXu, aby byl graf co největší.

`set output 'Graf1.eps'`
přesměrování výstupu. Gnuplot vytvoří ještě soubor `Graf1.tex`, který se pak pomocí `\input` vkládá do zdrojového souboru LaTeXu.

`replot` ..... znovu vykreslí posledně zobrazený graf. Je též možno zde provést kreslení pomocí `plot`. Při použití `multiplot` nefunguje příkaz `replot`, proto je nutné veškeré kreslící příkazy napsat zde.

`set output` ..... uzavře výstupní soubor. Nutný příkaz – jinak nemusí být soubory korektně uzavřeny.

`set term pop` ... uvede gnuplot do původního stavu, aby další graf byl kreslen na dříve nastavený terminál.

Další možností je použití parametru terminálu `epslatex standalone`. Jako výstup dostaneme opět dva soubory `.tex` a `.eps`, ale v tomto případě se nevkládáme soubor `.tex` přímo do našeho LaTeXového souboru, ale nejdříve jej přeložíme. Pokud jej přeložíme pomocí LaTeXu, tak dostaneme jako výstup `.eps` obsahující jak graf, tak popisky, které vložíme přímo do našeho LaTeXového souboru standardně pomocí `\includegraphics`. Pokud provedeme překlad pomocí pdfLaTeXu, tak po překladu obdržíme `.pdf` soubor, který opět standardně vložíme do našeho LaTeXového dokumentu nebo si jej již můžeme přímo prohlížet či publikovat.

### 1.10.2 Nastavení gnuplotu pro png výstup

Postup je naprosto stejný jako výše, jenom terminál nastavujeme pomocí příkazu `set term png giant size 1200,900` a nastavíme odpovídající příponu výstupního souboru.

## 1.11 Nevíte-li si rady

- Před bezhlavým pročítáním celého manuálu se zamyslete, kde by se požadovaná vlastnost mohla dát nastavit a najít – k tomu je třeba znát odpovídající anglický výraz. Mnohé příkazy by měly být intuitivní! Můžete se nechat inspirovat nejrůznějšími průvodci, úvody, tutoriály apod., a to ve volné chvíli raději než v době, kdy na něco spěcháte. Snažte se proniknout do *systému* příkazů pro gnuplot . Při nejhorším se obraťte na své známé...
- Práce s grafickým výstupem (okna s grafy): aktivujte okno s grafy kliknutím na něj, stiskněte klávesu `h` a přejděte zpět na příkazovou řádku, kde se vypsaly funkce kláves (např. `u`–návrat ze zoomu,...).
- Příkaz `test` zobrazí ukázku možných nastavení s příslušnými číselnými kódy (aneb jaká čísla zadat pro vykreslení zelených značek ve tvaru hvězdičky aj.)

`help` ..... ve Windows otevře okno nápovědy, v Linuxu vypíše odpovídající nabídku na konzoli.

`help set term`
nápověda pro konkrétnější dotaz. Zde je mj. seznam podporovaných formátů výstupu.

`help set xlabel`
potřebujete-li vědět syntaxi k nastavení popisu $x$-ové osy.

# 2 Pokročilé techniky

## 2.1 Výstup do LaTeXu – dokončení

Viz sekce 1.10.1. Pro výstup do LaTeXu je nutno umístit LaTeXové značky přímo přímo do zdrojového souboru GNUPLOTu.

- Při použití příkazu `set label` použijeme `'`, nikoli `"`[12]. Pro popisek osy používáme makro `\popi{I}{mA}`, makro je součástí FYKOSího balíčku maker.
- Je vhodné posunout $y$-ový popisek osy vlevo příkazem `set ylabel offset 3,0`, popřípadě jiné opticky pěkné posunutí.
- Pro správné popisky os je tvaru $1{,}3 \cdot 10^{-2}$ použijeme `set format $"%g"$` a FYKOSí LaTeXová makra.

## 2.2 Kreslení ve 3D

Vzhledem k tomu, že tato sekce je velmi rozsáhlá, tak pro konkrétní nastavení doporučujeme shlédnout dema dodávaná spolu s GNUPLOTem . Stažitelná jsou též z oficiálních stránek.

`splot` ..... je základním příkazem pro kreslení 3D grafů. Rozsahy na osách $x$, $y$ a $z$ je nastavují stejně jako v případě grafů funkce jedné proměnné, jen musíme použít tři páry hranatých závorek. Opět je možno použít příkazy `set ?range`. Jsou vykreslovány funkce v proměnných $x$ a $y$, popřípadě se používá nastavení `set parametric` $u$ a $v$ pro povrchy nebo $t$ pro křivky.

`set isosamples 200`
: nastavuje počet vzorků, na kterých bude kreslená funkce vyčíslována. V tomto případě to bude $200^2 = 40\,000$ vzorků.

`set pm3d at bst`
: nastavuje barevnou škálu (`cb`) grafu. `b` vykresluje funkční hodnoty v barvě na spodní podstavu grafu, $t$ na horní podstavu grafu a $s$ na povrch grafu. Jsou vykreslovány v pořadí zadání, tj. zde bude nejdříve barevná škála nakreslena na spodní podstavu, dále na povrch grafu a nakonec na horní podstavu, čímž budou povrchy správně průhledné. Pro jiné pořadí `bst` nemusí být neprůhlednost nastavena správně, pro volbu `s` se průhlednost nastavuje automaticky a většinou dobře. Není nutné použít všechna tři písmenka.

`set view 30,40`
: otáčí s grafem ve směru osy $x$ o 30° a ve směru osy $z$ o 40°.

`set view map` ... nastaví pohled na graf shora. Vhodné v kombinaci s `set pm3d at b` pro kreslení map.

`set hidden3d` ... nastaví neprůhledný povrch grafu. Shora jej nabarví červeně, zespodu zeleně.

`set contour base`
: vykreslí na spodní podstavu grafu křivky odpovídající jedné hodnotě funkce.

## 2.3 Komplexní čísla

Gnuplot též umí pracovat s komplexními čísly.

`a={2.2,3.4}` .... definuje komplexní konstantu $a$.

`real(x)` ..... vrátí reálnou část $x$.

`imag(x)` ..... vrátí imaginární část $x$.

`arg(x)` ..... vrátí argument $x$.

`abs(x)` ..... vrátí velikost $x$.

`f(x) = {1,0}*f1(real(x),imag(x)) + {0,1}*f2(real(x),imag(x))`
: dovoluje definovat vlastní funkci komplexní proměnné, kde `f1` je její reálná část a `f2` je její imaginární část. Zápis `f(x) = {f1(real(x),imag(x)),f2(real(x),imag(x))}` není možný.

Ostatní standardní funkce jsou definovány tak, aby pracovaly s komplexními čísly. Imaginární část logaritmu je nespojitá na záporné části reálné osy.

[12] V tomto případě je \ řídícím znakem a je nutno jej dvojit.