ZÁPADOČESKÁ
UNIVERZITA

Fakulta aplikovaných věd
Ústav fyzikálního inženýrství

UŽIVATELSKÉ NADSTAVBY SÁZECÍHO SYSTÉMU LaTeX 2.09

Prof. Ing. Miroslav BALDA, DrSc.

1996

M. Balda: ZČU-FAV-ÚFY
mbalda@hera.zcu.cz

Uživatelské nadstavby sázecího systému LaTeX

Text je věnován dvěma autorovým souborům maker pro sázecí systém LaTeX, které rozšiřují dosavadní možnosti tohoto systému o uživatelsky přístupné celočíselné aritmetické operace, reálnou aritmetiku v pevné řádové čárce, elementární a statistické funkce, náhodné procesy, cykly, terminálové vstupy a výstupy, práce se seznamy a zejména pak o bohaté grafické procedury v souboru `rplot.sty` a užitečná makra pro psaní učebních textů, zpráv a článků pro odborný tisk v souboru `user.sty`. Oba soubory obsahují celkem přes 200 nových maker a jsou k dispozici na `ftp`-serveru `mars.zcu.cz` v adresáři `pub\tex`.

119 stran Plzeň, 1996

Obsah

Kapitola 1

Úvod

V posledním desetiletí se počítače staly snadno dostupné každému, kdo je potřebuje. Jejich rozšíření mělo a má řadu jak pozitivních, tak i negativních důsledků. Na jedné straně počítače umožňují získat požadované výsledky rychleji, přesněji a umožňují ukládat a vybírat důležité informace a s ohledem na mezinárodní propojení počítačů v sítích, to vše nikoliv pouze z vlastních zdrojů. V neposlední řadě mají pozitivní dopad i na vzdělanost národů. Na druhé straně ovšem přinášejí i problémy spojené s počítačovou kriminalitou, s hráčskou či pracovní závislostí a porušováním autorských práv. Přes to všechno převyšují pozitivní rysy nad negativními.

Jednou z oblastí, v níž se úloha počítačů výrazně uplatňuje, je oblast sdělování informací psanou formou. Požadavky na grafickou úroveň materiálů předávaných adresátovi v poslední době mnohonásobně vzrostly. Postupně mizí ruční psaní delších textů, které mají být předávány druhému partnerovi nebo úřadu. V poslední době se však opouštějí i psací stroje a pole stále více ovládají počítače. Změnu lze však pozorovat i u jejich vyjadřovacích prostředků. Zatímco v ne příliš vzdálené minulosti byl počítač pouhou módní náhražkou psacího stroje a stačil mu pro psaní textů jednoduchý editor s omezenou sadou znaků, je trh v současné době zaplaven programovými prostředky pro kvalitní publikační práce zahrnující jak sázení textů, tak i vkládání obrázků. Mezi prvé z nich patřil i sázecí program TEX D. E. Knuthe [1] a později jeho uživatelsky přístupnější nadstavba LATEX L. Lamporta [2]. Další vývoj vede k LATEXu v. 3, jehož průběžný stav vývoje popisuje práce [3].

Pro běžného uživatele je LATEX nejsnáze použitelný, protože mu usnadňuje zápis programu pro sázení dokumentu a zmenšuje nároky na uživatele na minimum. V české odborné literatuře je k dispozici řada titulů věnovaná oběma programům. Nejvýznamnější jsou uvedeny v seznamu pod čísly [4] až [7].

1.1 Současný stav

Důvodem pro široké rozšíření a využívání obou programů je nejen dokonalost, s jakou je výsledný text vysázen, ale i skutečnost, že se autoři vzdali odměny za svá díla a oba programy a jejich další nadstavby se šíří zdarma jako tzv. programy „public domain“. To je ve výrazném kontrastu s komerčně dosažitelnými programy na trhu, které jsou velice drahé. Důvodem pro jejich prodejnost je skutečnost, že mají některé vlastnosti lepší než TEX a LATEX, zejména pokud jde o práci s grafickou informací, o výběr možných sad znaků a viditelnost vkládané informace. Nový vývoj LATEXu však tento rozdíl snižuje.

Zatímco vývoj TEXu je ukončen a jeho stav z rozhodnutí autora D. Knuthe konzervován, vývoj LATEXu pokračuje. Kromě u nás nejrozšířenější verze LATEX v. 2.09 je ve světě rozvinut mezinárodní projekt LATEX v. 3.0. Protože jeho vývoj je financován univerzitami a sponzory, potrvá ještě dlouhou dobu, než bude dokončen. Z tohoto důvodu existuje jistá dočasná verze označovaná jako LATEX2ε, která představuje okamžitý stav vývoje LATEXu v. 3 [3]. Nový LATEX by měl odstranit řadu slabých míst původního LATEXu. Z nich nejvýraznější spočívají v málo rozvinuté grafice.

Existuje řada pokusů o začlenění zlepšené grafiky do LATEXu (LATEX v.3, PicTEX [8], TeXCAD [9]) anebo o vazbu LATEXu na nové hardwarové i softwarové možnosti (PStricks [10], Scientific Word [11]) atd.). Aktuální informace o stavu vývoje lze nalézt v publikacích skupiny uživatelů TEXu – CsTUG. Výsledky nového vývoje však nemusí být pro každého uživatele LATEXu využitelné nebo výhodné pro hardwarová, softwarová či jiná omezení. Proto jsme se pokusili sestavit řadu maker, která by uživateli rozšířila možnosti využití LATEXu v. 2.09 a nebránila budoucím aplikacím u jeho vyšších verzí, možná s drobnými úpravami.

Výsledky této činnosti jsou shrnuty v následujících kapitolách, které jsou věnovány popisu balíků maker – stylů, jejichž hlavním cílem je usnadnit uživateli práci při vydávání technicky zaměřených publikací, jako jsou články, referáty pro konference, skripta a knihy.

1.2 Charakteristika nových stylů

Cílem této práce je dát běžnému uživateli LATEXu k dispozici prostředky, které mu podstatně rozšíří možnosti při vytváření grafiky v rámci dokumentu, který právě zpracovává. Při tom ale nevyžadují, aby uživatel znal základní jazyk TEX a s ním spolupracující `METAFONT`, ale pouze nové dále uvedené LATEXovské povely, po jejichž zvládnutí bude moci zrealizovat jednoduché výpočty funkcí, kreslení grafů ve tvaru hladkých a po částech lineárních čar a stát se tak alespoň zčásti nezávislým na dalších softwarových produktech.

Předkládaný hlavní balík povelů `rplot.sty` je výsledkem dlouhodobého vývoje původně dvou stylů, jednoho pro aritmetiku (`ari.sty`) a druhého pro kreslení (`fplot.sty`). Když se však ukázalo, že se obvykle potřebují makra z obou souborů, byly oba styly spojeny v jeden se současným podstatným přepracováním a doplněním na jaře 1996. Vývoj pak již probíhal společně.

Soubor `rplot.sty` obsahuje zejména makra pro celočíselnou a reálnou aritmetiku (v pevné řádové čárce), makra pro výpočet elementárních funkcí a makra pro grafické práce navazující na soubory `curves.sty` a `curvesls.sty` z lit. [12]. Nová makra rozšiřují možnosti o vynášení po částech lineárních čar, tedy i mnohoúhelníků, časových řad apod. Pro vynášení diagramů lze i vstupovat s externími daty ve formě ASCII souborů z jiných programů (např. `MATLABu`). Kromě toho může uživatel vkládat do svých programů i volání dalších podpůrných prostředků, které soubor `rplot.sty` obsahuje, jako jsou operativní vstupy a výstupy přes uživatelský terminál a makra pro přípravu diagramů.

Aby uživatel nepotřeboval žádný další referenční materiál, budou zde popsány i základy dvou existujících prostředků pro grafické práce, a to prostředí `picture` LATEXu [2], styl `curvesls.sty` [2] pro kreslení hladkých křivek a styl `ifthen.sty`, které `rplot.sty` rovněž využívá. Je zde také stručně popsán soubor maker `user.sty`, který se osvědčil při vytváření učebních textů. Obsahuje některé vybrané příkazy pro častěji se opakující struk-

tury, jimiž lze výrazně snížit pracnost zápisu a i zrychlit přípravu dokumentu. Obsahuje rovněž skupinu maker, která byla použita při přípravě tohoto textu.

1.3 Terminologie

Aby nedošlo později k omylu, je vhodné definovat hned na začátku některé pojmy, které se později užijí. Velmi důležité je si všimnout anglického pojmu `command`, který má v TEXu význam neterminálního symbolu, který se při překladu dále rozvíjí. V tomto textu bude tento pojem překládán do 3 různých výrazů a to na

- **povel** jako jméno objektu zastupujícího množinu znaků, ve které se při překladu rozvine,
- **příkaz** jako název jistého úkonu,
- **makro** jako název ucelené množiny příkazů, která může mít i parametry.

Rozdíl mezi příkazem a makrem není ostrý a často bude používán ve stejném smyslu. Všechny druhy objektů typu `command` se v TEXu zapisují formálně stejně, totiž začínají zpětným lomítkem, za nímž bezprostředně následuje vlastní název objektu.

Číselná bezrozměrná informace se ukládá do tzv. čítačů. Čítače TEXu budeme dále nazývat `Tčítači` a čítače LATEXu `Lčítači`. Toto rozlišení typu je nutné proto, že se odlišně definují a zapisují. V případech, kdy nebude záležet na typu čítače, budeme mluvit o obecných čítačích nebo prostě čítačích. Všech čítačů je k dispozici 256. Číselná informace s délkovým rozměrem se ukládá do jiné skupiny 256 délkových registrů.

Pro výpočty se dají užít obě skupiny registrů. Protože jakýkoliv registr je tvořen 32 bity, t.j. 4 byty, zdálo by se, že použití čítačů i délkových registrů bude dávat stejné výsledky i vyžadovat stejnou pracnost. Skutečnost je však jiná. Navzdory možnosti zápisu délkových informací jako reálných necelých čísel (s desetinnou tečkou), což je velice příjemná vlastnost, nebyla zde využita. Vnitřní zobrazení reálných délek je celočíselné v jednotkách nazývaných „scaled points“ `sp`. Délky se potom vyjadřují v tzv. „bodech“ – points (zkratka `pt`). To je však již v pevné řádové čárce, která leží v polovině bitového rozsahu registru, tedy za prvními dvěma bajty. Konverze mezi oběma typy zobrazení není příliš složitá, protože první dva byty představují celou část a druhé dva byty zlomkovou část délky. Uživatel však musí zajistit neustálé odtrhávání rozměru "`pt`", kde `1pt = (1/72.27)in = 0,35146mm`. Makro, které plní tuto funkci, je součástí jak souboru `rplot.sty`, tak i `usr.sty`. Zdálo by se tedy, že využití délkových registrů pro realizaci aritmetiky má až na trvalou přítomnost rozměru, který však umíme odstranit, jen dobré vlastnosti.

Vlastní pokusy s aritmetikou založenou na délkových registrech však ukázaly, že výsledky aritmetických operací jsou velmi citlivé na řádové rozdíly ve velikostech argumentů. Při nich totiž dochází k výrazné ztrátě platných míst výsledku. To potom při výpočtech hodnot funkcí, obsahujících řadu aritmetických operací, vedlo k chybám, které nebylo možno tolerovat ani pro grafické účely. Proto se autor rozhodl vybudovat aritmetiku založenou na práci s čítači i za cenu vlastního měřítkování veličin. Protože v TEXu není dosažitelná binární aritmetika, byla zvolena poloha desetinné tečky tak, že oměřítkované reálné číslo se zobrazí jako celé s rozlišitelností 10^{-4} v reálné oblasti. Tato rozlišitelnost je bohatě vyhovující pro grafiku i pro jednoduché výpočty. Toto rozhodnutí navíc i rozšířilo dynamický rozsah celých částí informace z $\pm$`16383` u délkových registrů na $\pm$`214747` u takto měřítkovaných čítačů, tedy více než o řád. I když vlastní realizace aritmetiky

v pevné řádové čárce byla pracnější, vlastní ošetření zaokrouhlování výsledků operací odstranilo nadměrné ztráty informace v mezních případech.

1.3.1 Typy veličin

Zatímco v běžném LaTeXu má uživatel k dispozici dosti omezené prostředky pro práci s číselnou informací, jsou v novém stylu vytvořeny nástroje pro práci s několika druhy veličin. Dále budeme pro ně používat následující názvy:

1. **Reálné veličiny**,
   které budeme dále označovat identifikátory, jejichž počátečním znakem bude obvykle písmeno `r`. Nesou racionální dekadická čísla, která mohou obsahovat desetinnou tečku, a mohou být z intervalu ⟨`-214748.3647, +214748.3647`⟩. Pokud číslo neobsahuje desetinnou tečku, je její implicitní poloha za poslední číslicí čísla. V rámci tohoto popisu mohou mít reálné veličiny pouze dva tvary:

   - **reálné číslo,** např.:
     ```
     {123.45}
     {3.1415927}
     {-17}
     ```
     Za reálnou veličinu lze tedy použít jak přímo reálné číslo uzavřené ve složených závorkách, tak i jméno povelu s tímto reálným číslem (bez závorek).
   - **reálný povel,** např.: `\pi,\alpha,`..., jemuž byla přiřazena reálná hodnota příkazem `\edef`, nebo `\Cmd`, anebo příkazem LaTeXu `\renewcommand`, např.:
     ```
     \edef\Pi{3.14159265} nebo  \Cmd \Pi={3.14159265}
     \edef\alpha{-.25}          \Cmd \alpha={-.25}
     \renewcommand{q}{\Pi}      \Cmd \q=\Pi
     ```

2. **Oměřítkované reálné veličiny**,

   dále označované `s`, (od **s**caled real), nesou celočíselnou informaci vzniklou z reálné předpisem

   ```
   s = round(r*10000)
   ```

   Mohou mít následující tvary:

   - **oměřítkované reálné číslo:**
     ```
     {1234500} odpovídající 123.4500
     {31416}   odpovídající   3.1416
     {-170000} odpovídající -17.0000
     ```
   - **oměřítkovaný reálný povel** definovaný uživatelem jedním z příkazů `\edef`, `\Cmd` nebo `\renewcommand`, např.:
     ```
     \edef\sa{1234500}
     \Cmd \spi={31416}
     \renewcommand{\sb}{-170000}
     ```
     odpovídající stejným reálným veličinám jako je uvedeno výše.

- **oměřítkovaný reálný čítač** definovaný povelem LaTeXu, `Lčítač`, např.:

```
\newcounter{sca}
\newcounter{scpi}
\newcounter{scb}
```
nebo povelem TeXu: (`Tčítač`):
```
\newcount\sca
\newcount\scpi
\newcount\scb
```

  kde `sca` a `\sca` jsou dva **různé** čítače (registry).

Číslo typu `s` **nesmí** obsahovat tečku! Pokud se tak omylem stane, buď se ohlásí chyba při překladu, anebo se za `s` považuje jen celá část čísla a zlomková se zašle do výstupního dokumentu.

3. **Celočíselné veličiny**,

   dále označované jmény s počátečním písmenem `i`, jsou běžná celá čísla z intervalu $\pm$`(2`^{31}`-1)`, t.j: ⟨`-2147483647, +2147483647`⟩. Mohou mít následující tvary:

   - **celé číslo**, např.: {`-17`}, {`0`}, {`453`},...
   - **celočíselný povel**, např.: `\ia, \jb,...` jímž byla přiřazena celočíselná hodnota příkazy

```
\edef\ia{-17}%                 případně:
\renewcommand {\ia}{-17}% nebo
\Cmd \ia={-17}
```

   - **Čítač** (counter, registr) definovaný povelem LaTeXu (`Lčítač`) :

```
\newcounter{ia}
\newcounter{ib}
```
a pod., nebo povelem TeXu (`Tčítač`):
```
\newcount\ia
\newcount\ib
```
a pod.

     Zde je zapotřebí upozornit, že `ia` a `\ia` jsou dva **různé** čítače.

     V předstihu zde uveďme, že všechny Lčítače (ať již oměřítkované reálné, nebo celočíselné) lze naplnit pomocí příkazů

```
\setn{jméno čítače}{celé číslo}
```
nebo
```
\setc{jméno čítače}{jméno čítače} ,
```

     které jsou součástí množiny příkazů dále uvedené celočíselné aritmetiky. Je zapotřebí však upozornit, že tyto příkazy nejsou určeny pro plnění čítačů TeXu, jejichž jména začínají znakem zpětného lomítka `\`.

Poznámky:

- Všechny číselné údaje, ať již reálné nebo celočíselné, je zapotřebí uzavírat do složených závorek. Pokud toto opomineme, přenese se jako argument jen první znak a zbytek se zašle do výsledného dokumentu, anebo vystoupí chyba při překladu.
- Využívání Tčítačů nelze běžnému uživateli LaTeXu doporučit, protože je spojeno s možnými problémy interference s již definovanými čítači v různých modulech (stylech).

4.

Konstanty - položky

Ve všech případech, kdy některý z argumentů příkazů bude označen jako konstanta nebo položka určitého typu, bude se jednat buď o

- číslo daného typu,
- povel, obsahující číslo daného typu, nebo
- jméno registru s konstantami typu `i` nebo `s`.

Nedodržení typu argumentů v povelech může vést buď k chybám při překladu, anebo k chybným výsledkům. To se týká zejména záměny reálných veličin typu `r` (necelých) a oměřítkovaných reálných veličin typu `s`.

5. **Seznamy**

Předložený soubor maker je schopen zpracovávat i velmi užitečnou strukturu dat nazývanou **seznam**. Seznam je uspořádaná množina libovolného počtu položek navzájem oddělených čárkami. Pod pojmem položka rozumíme libovolnou konstantu nebo text. Počet prvků v seznamu se neuvádí explicitně, protože se zjišťuje až při zpracování seznamu. Tak např. obecný seznam by mohl mít tvar:

```
1.23, C, abcd, 1, \command
```

Čárka může být pouze **mezi** položkami, tj. nesmí být ani na začátku ani na konci seznamu! Obvykle je seznam sestaven z uspořádaných n-tic položek. Při tom žádná z položek **nesmí** sama obsahovat znak čárky, protože ta má funkci oddělovače položek.

Poznámky:

- Některé z dále uvedených maker vyžadují, aby položky v seznamu byly homogenní v typu. Obvykle půjde o seznamy číselných položek, v nichž se mohou vyskytovat (třeba i smíšeně) pouze čísla, čítače TEXu nebo povely obsahující číselnou informaci. V případě standardních čítačů LATEXu je potom zapotřebí použít v seznamu pro danou položku konverzi jeho jména na číslo pomocí příkazů `\value{C}`, nebo `\theC`, kde symbol `C` zde zastupuje uživatelův název čítače (např. `\value{ab}` nebo `\theab`).

- Je velmi důležité dodržovat pravidlo, že číselné informace ležící mimo seznamy se **musí** uzavírat do složených závorek, jak je patrno ve výše uvedených příkladech.

Kapitola 2

Aritmetika

Běžný uživatel LaTeXu se může velice snadno dostat do problémů, chce-li umisťovat svoje objekty do textu podle své vůle, avšak automaticky (programově). Potom nutně potřebuje vypočítávat polohy a podle nich se i rozhodovat. K tomuto úkolu však LaTeX nebyl určen, a tak je úkol uživatele značně ztížen. K takřka nepřekonatelným problémům dojde běžný uživatel i při pokusu vytvořit složitější obrázek, byť i jen s jednoduchou funkcí. Pro usnadnění těchto úloh byl předkládaný balík maker vybaven prostředky pro aritmetické operace a výpočty elementárních funkcí v aritmetice s pevnou řádovou čárkou. Všechny operace se dějí nad běžnými celočíselnými registry – čítači LaTeXu a TeXu. Z tohoto důvodu je dosahovaná přesnost aritmetických operací i aproximací funkcí omezená, avšak dostatečná pro grafické práce a sázení.

Všechna makra využívaná pro výpočty, pokud by mohla být použitelná i pro konečného uživatele, jsou dále uvedena a rozdělena do několika skupin.

2.1 Pomocná makra

V případě, že jsme sami nuceni sestavit makro, které nám usnadní práci, je často výhodné pracovat i s Tčítači a příkazy TeXu, protože se tím některé operace výrazně usnadní. TeX k definici čítačů a příkazů využívá makra:

```
\newcount\jméno%       definice Tčítače \jméno
\edef\povel{obsah}%    definice TeXovského příkazu \povel
```

Tyto příkazy lze používat i v LaTeXovském programu. Tam ovšem příkaz `\povel` lze nadefinovat i LaTeXovským příkazem

```
\newcommand{\povel}[n]{obsah}%          při prvním výskytu
\renewcommand{\povel}[n]{nový obsah}%   při dalších výskytech
```

Tento zápis je dosti náročný, protože je určen pro definice i velmi složitých maker, i protože se předpokládá, že uživatel ví, zda povel již definoval, a má tedy použít druhou variantu definice, či ne. Proto byla připravena jednodušší makra pro definice obecných čítačů a povelů a jejich adresování. Patří k nim makra `\newcnt, \Cnt, \Cmd` a `\Arg`.

`\newcnt{cnt}` **Podmíněná definice obecného čítače**

Makro otestuje, zda obecný čítač `cnt` existuje. Pokud neexistuje, nadefinuje ho. Jde o velmi dobrou náhradu příkazů `\newcounter` i `\newcount`.

`\Cnt icnt={iarg}` **Definice a naplnění čítače**

Makro napřed otestuje, zda požadovaný obecný čítač `icnt` existuje. Pokud nebyl dosud definován, vytvoří ho. Nakonec do tohoto čítače dosadí hodnotu z celočíselného argumentu `iarg`. S ohledem na vestavěný test nehrozí nebezpečí vícenásobného vytvoření čítače stejného jména, i kdyby bylo makro `\Cnt` použito v cyklu (zbytečně).

Příklad:
```
\Cnt p={12345}%  p=12345
\Cnt \q=abc%     \q=hodnota z Lčítače abc
\Cnt r=\cmd%     r=hodnota z příkazu \cmd
```

`\Cmd \icmd={iarg}` **Definice a naplnění celočíselného povelu**

Makro dešifruje argument `iarg`, jímž může být celé číslo, libovolný čítač, jiný celočíselný povel, anebo konstrukce `\value{ Lčítač}` nebo `\thečítač`, a jeho hodnotu dosadí do povelu `\icmd`, který si vytvoří.

Příklad:
```
\Cmd \c={ia}%       \c=hodnota z ia
\Cmd \d={-\theia}%  \d=-hodnota z ia
\Cmd \c=\the\j%     \c=hodnota z \j
```

`\Arg{iarg}` **Dodání čísla, obsahu povelu nebo obecného čítače**

Makro funguje jako funkce. Vrací číslo nebo TEXovský název čítače nebo povelu bez ukládání, takže výsledek lze přímo použít i jako argument jiného makra. Užije se především v makrech pro dešifrování argumentů, pokud mohou být různého typu.

Příklad: `\q\Arg{#2}%  \q=hodnota 2. parametru v makru`

2.2 Celočíselná aritmetika

Jak TEX tak i LATEX poskytují uživateli jisté minimální prostředky pro základní aritmetické operace s celými čísly a s čítači. Ovšem původní příkazy pro tyto operace jsou formálně rozvleklé a bez systematické stavby. Jsou to příkazy TEXu

`\advance\i\iarg`	pro součet,
`\advance\i-\iarg`	pro rozdíl,
`\multiply\i\iarg`	pro součin a
`\divide\i\iarg`	pro podíl,

kde `\i` je Tčítač a `iarg` může být Tčítač, povel nebo celé číslo. V LATEXu jsou k dispozici pouze nejjednodušší příkazy pro práci s Lčítači `icnt`:

`\setcounter{icnt}{inum}`	pro dosazení celého čísla `inum` do čítače `icnt`
`\addtocounter{icnt}{inum}`	pro přičtení celého čísla `inum` do čítače `icnt`
`\stepcounter{icnt}`	pro přičtení jedničky do čítače `icnt`

Proto se vypracoval systém celočíselné aritmetiky, kterým se odstranily výše uvedené nedostatky a doplnily se některé další operace. Makra celočíselné aritmetiky jsou uspořádána do třech skupin, a to na makra obecná, pro práci s celými čísly a s Lčítači.

2.2.1 Obecná makra

V této skupině jsou makra pracující s obecnými předem definovanými celočíselnými veličinami označenými `iarg`, jimiž mohou být Tčítače, Lčítače, povely a celá čísla. Pokud se výsledek někam ukládá, tak výhradně do obou typů čítačů `i`. Do této skupiny zařadíme ještě makra realizující unární operace, jako jsou absolutní hodnota, znaménko a změna znaménka a navíc i víceparametrické operace minima a maxima.

`\Set{i}{iarg}` **Naplnění čítače** **i = iarg**

Po zjištění typu parametru `iarg`, kterým může být celé číslo, obecný čítač nebo povel s celočíselnou informací, dosadí makro jeho číselnou hodnotu do obecného čítače `i`.

Příklad:
```
\Set{a}{12345}%  a=12345
\Set{t}{\q}%     t=\q,   ale také
\Set{t}\q%       t=\q
\Set{\q}{s}%     \q=s,   ale také
\Set\q{s}%       \q=s
```

`\Add{i}{iarg}` **Součet** **i = i+iarg**

Celočíselná hodnota z `iarg` se přičte k hodnotě v čítači `i`, kam se i uloží výsledek.

Příklad:
```
\Add{a}{987}%  a=a+987
\Add{\q}{a}%   \q=\q+a
\Add\q{a}%     \q=\q+a
\Add{t}{\q}%   t=t+\q
\Add{t}\q%     t=t+\q
```

`\Sub{i}{iarg}` **Rozdíl** **i = i–iarg**

Celočíselná hodnota z `iarg` se odečte od hodnoty v obecném čítači `i`.

Příklad:
```
\Sub{a}{456789}%  a=a-456789
\Sub{t}\q%        t=t-\q
```
a podobně jako u `\Add`

`\Mul{i}{iarg}` **Součin** **i = i∗iarg**

Obsah obecného čítače `i` se pronásobí celočíselnou hodnotou z `iarg`.

Příklad:
```
\Mul{a}{127}%  a=a*127
```
a podobně jako u `\Add`

`\Div{i}{iarg}` **Podíl** **i = i/iarg**

Obsah čítače `i` se nahradí výsledkem celočíselného dělení jeho původního obsahu hodnotou z `iarg`. Příklady by se mohly uvést podobné jako u předešlých příkazů.

`\Abs{i}` **Absolutní hodnota** **i = |i|**

Obsah obecného čítače `i` změní znaménko, pokud je záporný. V opačném případě zůstane obsah čítače nezměněný.

Příklad:
```
\Abs {a}%  a=|a|
\Abs \q%   \q=|\q|
```

`\Neg{i}` **Změna znaménka** **i = –i**

Makro změní znaménko u obecného čítače `i`.

Příklad: `\Neg{a}%  i=-i`

`\Sgn{i}` **Znaménko** **i = - 1; 0; 1**

Makro vrátí hodnotu, aniž ji uložilo, a to:

```
-1   pro   i < 0
 0   pro   i = 0
+1   pro   i > 0
```

`\Min{i}=(seznam)` **Minimum** **i = min(seznam)**

Makro najde minimální hodnotu z celočíselného seznamu a dosadí ji do obecného čítače `i`. Seznam je tvořen celočíselnými položkami navzájem oddělenými čárkou. Jako položka může v seznamu figurovat celé číslo, celočíselný povel, Tčítač a konstrukce `\value{ic}` příp. `\theic`, kde `ic` je předem definovaný a naplněný Lčítač.

Příklad:
```
\Cnt \q={-123}%              čítač \TeXu
\Cmd \x={3456}%              povel
\Set {a}{12}%                čítač \LaTeXu
\Min a=(0,\thea,\x,257,\q)%  a=-123
```

`\Max {i}=(seznam)` **Maximum** **i = max(seznam)**

Makro najde maximální hodnotu z celočíselného seznamu a dosadí ji do obecného čítače `i`. Pro parametry makra platí totéž, co platí pro makro `\Min`. Makro `\Max` aplikované na stejný seznam jako je v příkladu u `\Min` by dalo výsledek `a=3456`.

2.2.2 Makra pro operace s celými čísly

První skupina maker je velmi obecná, protože umožňuje, aby parametry byly sice celočíselné, ale jinak obecné. Dešifrování typu argumentu pochopitelně platíme větší spotřebou času při překladu. Pokud ale víme, jaké budou typy parametrů, lze postavit makra neobsahující dešifrování typů. Mezi ně patří makra, v nichž první z operandů je Lčítač a druhý z operandů je **celé číslo**, Tčítač nebo povel s celým číslem. Na památku toho je koncovým znakem jména makra písmeno `n`:

`\setn{i}{inum}` **Naplnění Lčítače číslem** **i = inum**

Dosazení celočíselné hodnoty argumentu `inum` do Lčítače.

Příklad:
```
\setn{i}{12345}%    i=12345
\Cnt \q={345}%      \q= 345
\setn{ia}\q %       ia= \q
\Cmd \vklad{1000}%  \vklad=1000
\setn{ib}\vklad%    ib=1000
```

`\addn{i}{inum}` **Součet obsahu Lčítače s číslem** **i = i+inum**

Přičtení celočíselné hodnoty argumentu `inum` do Lčítače.

Příklad:
```
\addn{i}{12345} %    i=i+12345
\addn{ia}\q%         ia=ia+\q
\Cmd \vklad={25000}
\addn{ib}\vklad%     ib=ib+25000
```

`\subn{i}{inum}` **Rozdíl obsahu Lčítače a čísla** **i = i–inum**

Odečtení celočíselné hodnoty argumentu `inum` od hodnoty v Lčítači.

Příklad:
```
\subn{i}{12345} %    i=i-12345
\subn{ia}\q%         ia=ia-\q
\Cmd \vklad={25000}
\subn{ib}\vklad%     ib=ib-25000
```

`\muln{i}{inum}` **Součin obsahu Lčítače s číslem** **i = i∗inum**

Pronásobení obsahu Lčítače celočíselnou hodnotou argumentu `inum`.

Příklad:
```
\muln{i}{123456}%  i=i*123456
\muln{ia}\q%       ia=ia*\q
\muln{ib}\vklad%   ib=ib*\vklad
```

`\divn{i}{inum}` **Podíl obsahu Lčítače s číslem** **i = i/inum**

Obsah Lčítače `i` se podělí celým číslem `inum`. Do čítače `i` se uloží pouze celá část výsledku. Zbytek po dělení se ztrácí.

Příklad:
```
\setn{ia}{123456}
\divn{ia}{13}%      a=9496
\Set\q{-351}
\divn{a}\q%         a=-27
```

2.2.3 Makra pro operace s Lčítači

V této skupině maker jsou všechny argumenty Lčítači, které byly již dříve definovány. Na památku toho, že druhý operand je **Lčítačem**, končí názvy této skupiny maker písmenem `c`. Příklady se pro tuto skupinu neuvádějí, protože makra jsou jednoznačně definována zápisem uvedeným v hlavičce.

`\setc{ia}{ib}` **Naplnění Lčítače obsahem jiného Lčítače** **ia = ib**

Makro zašle obsah Lčítače `ib` do Lčítače `ia`.

`\addc{ia}{ib}` **Součet obsahů Lčítačů** **ia = ia+ib**

Obsah Lčítače `ib` se přičte k obsahu Lčítače `ia`.

`\subc{ia}{ib}` **Rozdíl obsahů Lčítačů** **ia = ia–ib**

Od obsahu Lčítače `ia` se odečte obsah Lčítače `ib`.

`\mulc{ia}{ib}` **Součin obsahů Lčítačů** **ia = ia*ib**

Obsah Lčítače `ia` se pronásobí obsahem Lčítače `ib`.

`\divc{ia}{ib}` **Podíl obsahů Lčítačů** **ia = ia/ib**

Obsah Lčítače `ia` se celočíselně podělí s obsahem Lčítače `ib`.

`\incc{i}` **Inkrementace obsahu Lčítače** **i = i+1**

K obsahu Lčítače `i` se přičte jednička.

`\negc{i}` **Negace obsahu Lčítače** **i = – i**

Makro změní znaménko obsahu Lčítače `i`.

`\absc{i}` **Absolutní hodnota Lčítače** **i = |i|**

Obsah Lčítače změní znaménko, byl-li záporný.

2.3 Aritmetika v pevné řádové čárce

Každý uživatel LaTeXu, který chce realizovat složitější akci, často narazí na omezení vyplývající z absence dobré aritmetiky. Ani realizace celočíselných operací tuto skutečnost příliš nezlepšuje. Proto se přistoupilo k implementaci maker umožňujících, i když v omezeném rozsahu, pracovat s reálnými čísly nebo lépe řečeno s jejich oměřítkovanými (již celočíselnými) ekvivalenty v aritmetice s pevnou řádovou čárkou (tečkou).

V souladu s terminologií uvedenou v úvodu se budou používat symboly:

`r, ra, rb, ...`	reálné konstanty, tj. reálná čísla ve složených závorkách nebo povely obsahující tato reálná čísla;
`\R, \Ra, ...`	povely předem definované uživatelem pomocí příkazu `\Cmd\R{}` pro budoucí uložení výsledků;
`s, sa, sb, ...`	oměřítkované reálné konstanty, t.j. celá čísla ve složených závorkách, čítače nebo povely, které je obsahují, vzniklé pronásobením reálných čísel 10^4 a zaokrouhlením výsledků na celá čísla;
`S, Sa, Sb, ...`	čítač pro uložení výsledku.

Výše uvedené symboly nahrazuje uživatel jím definovanými objekty (čísly, jmény čítačů, povelů).

2.3.1 Pomocné činnosti

S ohledem na různé typy veličin, se kterými se pracuje, jsou zapotřebí procedury ke konverzím mezi nimi. Sem patří nejen konverze mezi reálnými a oměřítkovanými reálnými položkami, ale i další. Navzdory pevnému měřítkovému faktoru není problém konverze zdaleka triviální záležitostí, pokud žádáme relativně volný formát vstupních informací, který je uživateli nejbližší.

`\Rset Scnt={r}` **Konverze reálného čísla na oměřítkované** **r → S**

Makro vyvolá uživatel vždy, když chce převést reálnou konstantu na oměřítkovanou do obecného čítače `Scnt`, aby s ní mohl dále pracovat. Makro je podobné celočíselnému příkazu pro definici a naplnění čítače `\Cnt`. Od něho se však odlišuje tím, že veličina `r` je reálná konstanta, tedy reálné číslo anebo příkaz s reálným číslem. Makro jeho hodnotu oměřítkuje a uloží do požadovaného čítače. Pokud ještě nebyl definován, vytvoří ho. Makro je ekvivalentní dvojicím příkazů
`\newcnt{S} \Rset S={r}` nebo `\newcnt\S \Rset\S={r}`

Příklad:
```
\Cmd\vklad{107.50}%  \vklad=107.50
\Rset tlak={12.03}%  tlak=120300
\Rset a=\vklad%      a=1075000
```

`\Rget \R={s}` **Konverze oměřítkovaného čísla na reálné** **s → \R**

Makro převádí oměřítkovanou veličinu `s` na reálné číslo, které ukládá do povelu `\R`, který vytvoří, pokud nebyl dosud definován.

Příklad:
```
\Rget \vklad={Kc}%  \vklad=zkonvertovaný obsah čítače Kc
\Rget \p=254000%    \p=25.4
\Rget \ra=\Ra%      \ra=\Ra
```

`\RloAB{s}` **Rozklad** **s → (\Ra,\Rb)**

Makro slouží k rozštěpení oměřítkovaného čísla `s` na původně celou část do `\Ra` a zlomkovou část do `\Rb`.

`\RloCD{s}` **Rozklad** **s → (\Rc,\Rd)**

Makro slouží k rozštěpení oměřítkovaného čísla z `s` na původně celou část do `\Rc` a zlomkovou část do `\Rd`.

`\RstAB{s}` **Složení** **(\Ra,\Rb) → s**

Makrem se složí registry `\Ra` a `\Rb` s oddělenými celky a zlomkovou částí do `s`.

`\RstCD{s}` **Složení** **(\Rc,\Rd) → s**

Makrem se složí registry `\Rc` a `\Rd` s oddělenými celky a zlomkovou částí do `s`.

`\dtor Srad={sdeg}` **Konverze stupňů na radiány** **Srad ← sdeg**

Makrem se konvertují hodnoty úhlů ze stupňů `sdeg` na radiány `Srad`. Parametrem `sdeg` může být libovolný úhel v oměřítkovaných stupních jako číslo, povel nebo obecný čítač. `Srad` je jméno obecného registru, který musí být uživatelem předem definován. Převod se uskutečňuje podle vztahu

$$\alpha[\,\mathrm{rad}\,] = \tfrac{\pi}{180}\,\alpha[\,\mathrm{deg}\,]$$

`\rtod Sdeg={srad}` **Konverze radiánů na stupně** **Sdeg ← srad**

Jde o inverzní funkci k `\dtor`. Tentokrát parametrem `srad` je libovolná oměřítkovaná reálná konstanta a `Sdeg` je obecný čítač, který musí být již nadefinovaný. Konverzní vztah má tvar:

$$\alpha[\,\mathrm{deg}\,] = \tfrac{180}{\pi}\,\alpha[\,\mathrm{rad}\,]$$

`\ltoc \cmd=\len` **Konverze délky do povelu** **\cmd ← \len**

Makro zajistí odtržení dimenze `[pt]` od délkové informace vyvoláním makra `\unpt`

a dosazení výsledku do žádaného povelu. To jde sice zajistit i jinak, avšak tento způsob je, jak bude patrno z příkladů, nejsnazší:

Příklad:

```
\newcounter{w}
\newlength{\wl}
\newcommand{\convlc}[2]%    Uživatelské makro pro konverzi
%          ~~~~~~~~~        length #2 in [pt] -> counter #1
{\setlength{\wl}{#2}\setn{w}{#2}%
 \setn{#1}{\wl}\divn{#1}{-878}\divn{w}{12}%
 \subc{#1}{w}\addn{#1}{\wl}\divn{#1}{6}%
}%                          Konec makra \convlc
\newlength{\dist}
\setlength{\dist}{100.0000pt}
\wri{\dist}%                *** \dist = 100.0pt
\ltoc \cmd=\dist \wric\cmd% *** \cmd = 100.0
\newcnt{d}%                 podmíněná definice čítače d
\convlc{d}{\dist} \wri{d}%  *** d = 1000000
\Cnt d=\dist \wri{d}%       *** d = 6553600
```

2.3.2 Základní aritmetické operace

V balíku jsou implementovány všechny čtyři základní operace nad oměřítkovanými reálnými položkami `sa` a `sb` t.j. v aritmetice s pevnou řádovou čárkou. Výsledek se posílá do čítače `S`, jehož skutečné jméno si předem definuje uživatel sám pomocí příkazu `\newcounter{jméno}`, nebo lépe `\newcnt{jméno}`.

`\Radd S={sa}+{sb}` **Součet** **S = sa+sb**

Součet je zrealizován prostým sečtením dvou celých čísel, která sama mohou mít znaménko.

`\Rsub S={sa}-{sb}` **Rozdíl** **S = sa–sb**

Rozdíl je proveden jako součet dvou celých čísel, přičemž u druhého se před vlastním součtem obrátí znaménko.

`\Rmul S={sa}*{sb}` **Součin** **S = sa∗sb**

Operace součinu je relativně komplikovaná. V prvé řadě se operand `sa` rozloží do vnitřních registrů – celá část do `\Ra` a zlomková část do `\Rb` a podobně operand `\sb` do registrů `\Rc` a `\Rd`. Potom se provede součin dvojic `(\Ra,\Rb)*(\Rc,\Rd)` do

$$\texttt{S = \Ra*\Rc*}10^4 \texttt{ + \Ra*\Rd + \Rb*\Rc + \Rb*\Rd/}10^{-4}.$$

Obsahy registrů `\Ra` až `\Rd` si po skončení operace ponechávají rozložený tvar operandů, aby mohly být dále opět využity bez nutnosti opakování dekompozice argumentu.

`\Rdiv S={sa}/{sb}` **Podíl** **S = sa/sb**

Jde o velmi choulostivou operaci, při níž může docházet k přeplnění. Pokud k němu dojde, je hlášena chyba přímo z překladače TEXu. Je použit následující postup:

- vydělí se celočíselně `c:=sa/sb` a tento podíl představuje celou část výsledku,
- vypočte se zbytek po dělení `sr` a dosadí se `d:=1`;
- zahájí se cykl, v němž se zbytek `sr` pronásobí nejvyšší mocninou deseti `d`, při níž ještě `|sr|*d<1073741820`;
- není-li splněna, pak pro `d<`10^4, se pokračuje v cyklu operacemi `d:=d*10` a `sb:=sb/10` až do splnění podmínky z předchozího bodu. Jinak
- výsledek `S:=`10^4`*c + sr/sb`.

Příklady:

```
\Rset c={12.3456}\Rset d={9876.005}
\Rmul F={c}*{d} \wri{F}%     *** F = 1219252073
\Rset e=\RMpi%               \RMpi = 31415.9265 = 1e4*pi
\Radd F={F}+{e} \wri{F}%     *** F = 1533411338
\Rdiv F={F}/{e} \wri{F}%     *** F = 48810
```

2.3.3 Přídavné aritmetické operace

Dosti často se potřebuje kumulovat výsledek při současném použití operace násobení stejnou hodnotou. Jde o operace typu

$$f_{i+1} = (f_i + a_i) * x$$
$$f_{i+1} = f_i * x + a_i.$$

Vyskytují se při výpočtech hodnot polynomů a tím i aproximací řady funkcí. Za tím účelem byla sestavena makra `\RmAF` pro první úlohu a `\RmFA` pro druhou. Obě makra, stejně jako makro `\Rmul`, využívají příkaz `\Rabcd` pro součin rozložených operandů, se současným předpokladem, že operand `x` je trvale rozložen v registrech `\Ra,\Rb`. Výsledek kumulace se ukládá do Tčítače vyhraženého pro výpočty funkčních hodnot `\F`. Účelem všech těchto maker je časová úspora při výpočtech.

`\RmAF{sa}` **Jeden krok kumulace** **\F←(\F+sa)∗x**

`\RmFA{sa}` **Jeden krok kumulace** **\F←\F∗x+sa**

Příklad 1:

```
\Rset sx={1.25}%                          x=1.25, sx=12500
\Rset \F={0}%                             \F=0
\RloAB{sx}%                               \Ra=1, \Rb=2500
\RmAF{40000}\RmAF{30000}\Add{\F}{20000}%  f = 4*x^2 + 3*x + 2
\wri\F%                                   \F = 120000
\Rget \f=\F \wric\f%                      \f = 12.0000
```

Příklad 2:

```
\Rset \F={4}%                          \F = 40000
\RmFA{30000}\RmFA{20000}%              f = 4*x^2 + 3*x + 2
\wri\F                                 \F = 120000
```

Tato úloha je stejná jako v prvním příkladu, avšak je řešená s lepší účinností.

`\Rabcd{sy}` **Rozklad 2. argumentu a násobení** **(\Ra,\Rb) ∗ sy**

Makro vyvolá rozklad argumentu `sy`→`(\Rc,\Rd)`, součin `(\Ra,\Rb)*(\Rc,\Rd)` a uložení výsledku do pracovního Tčítače `\w@`, který je v LaTeXu pro běžného uživatele běžně nepřístupný. Soubor `rplot.sty` však umožňuje získat výsledek vyvoláním příkazu `\Rstow{Scnt}`, kde `Scnt` je jméno uživatelova obecného čítače. Obsahy registrů `\Ra` a `\Rb` zůstanou vyvoláním makra neporušeny, zatímco obsahy registrů `\Rc` a `\Rd` se změní na nové složky vstupního argumentu `sy`.

Příklad:

```
\newcnt{x}
\Cnt sx={12000}%          sx = 1.2
\Rmul x={sx}*{sx}%        x = sx^2
\Rabcd{x} \Rstow{sx}%     sx = x*x^2
\wri{sx}%                 *** sx = 17280
```

`\Rstow{cnt}` **Uložení pracovního čítače** **\w@ → cnt**

Pracovní čítač `\w@` souboru `rplot.sty` se uloží do předepsaného čítače. Obvykle se použije po vyvolání makra `\Rabcd` pro úsporné opakované násobení, jak bylo již výše ukázáno.

Příklad:

Jako složitější příklad na použití aritmetických operací při sázení textu s obrázky uveďme makro pro jeden iterační krok hledání kořenu obecné funkce metodou regula falsi. Kořen potřebujeme znát pro jeho vynesení v grafu funkce. Iterace je dána formulí

$$x_{k+1} = \frac{x_{k-1}\, f_k - x_k\, f_{k-1}}{f_k - f_{k-1}}$$

Makro nazveme `\regfalstep`. Pro výpočet použijeme argumenty makra, které budou držet

- `#1` hodnotu výsledku iteračního kroku – lepší odhad kořene x,
- `#2` hodnotu x_{k-1},
- `#3` hodnotu x_k,
- `#4` hodnotu $f_{k-1} = f(x_{k-1})$,
- `#5` hodnotu $f_k = f(x_k)$.

Pro výpočet použijeme jako pracovní registr čítač `w`. Výpočet probíhá v aritmetice s pevnou řádovou čárkou.

```
\newcnt{w}
\newcommand{\regfalstep}[5]%
%           ~~~~~~~~~~~
{\Rmul #1={#2}*{#5}%      x(k-1)*f(k-1)
 \Rmul w={#3}*{#4}%       x(k)*f(k-1)
 \Rsub #1={#1}-{w}%       Čitatel
 \Rsub w={#5}-{#4}%       jmenovatel
 \Rdiv #1={#1}/{w}%       x(k+1)
}
\newcnt{x}
\regfalstep{x}{10000}{50000}{-1234}{3702}
\wri{x}%                  *** x = 20000
```

Pro úplný výpočet kořene by se pochopitelně muselo volání makra `\regfalstep` zabudovat do cyklu s testem přesnosti kořene a jako parametry volání by se použily konkrétní čítače, v nichž by byly uloženy hodnoty parametrů. O cyklu `\whiledo` a podmíněném příkazu `\ifthenelse` se lze více dozvědět ve 4. kapitole. Kompletní volání pro výpočet kořenu funkce $f(x) = x^2 - 2$ z intervalu $\langle 0, 3 \rangle$ by pak mohlo mít tvar:

```
\newcnt{f} \newcnt{w} \newcnt{eps}
\Cnt {xa}={0}\Cnt {fa}={-20000}%            x_{k-1}, f_{k-1}
\Cnt {xb}={30000}\Cnt {fb}={70000}%         x_{k}, f_{k}
\Cnt {eps}={10}%
\whiledo {\theeps>1}%
  {\regfalstep{x}{xa}{xb}{fa}{fb}%
   \Rmul f={x}*{x}\Rsub f={f}-{20000}%      f = x^2 - 2
   \Rmul w={f}*{fa}%
   \ifthenelse {\thew<0}%
     {\setc{eps}{xb}\setc{fb}{f}\setc{xb}{x}}
     {\setc{eps}{xa}\setc{fa}{f}\setc{xa}{x}}
   \subc{eps}{x}\absc{eps}
  }
\wri{x}%                        *** x = 14141  (=> 1.4142 ...)
```

Kapitola 3

Funkce

Výpočty hodnot funkcí patří mezi základní operace každé technické kalkulačky. Chceme-li kreslit grafy, je přirozené, že výpočet hodnot běžných funkcí musí být v arzenálu použitého prostředku. Proto soubor `rplot.sty` obsahuje několik skupin funkcí:

- elementární funkce,
- statistické funkce a
- náhodné procesy.

3.1 Elementární funkce

Pro vytváření jednoduchých grafů a grafických útvarů je účelné mít k dispozici kromě reálné aritmetiky i soupravu maker pro generování elementárních funkcí alespoň v množství, které je na běžných kalkulačkách. Pochopitelně, že vinou aritmetiky s pevnou řádovou čárkou je přesnost, jíž se vyznačují kapesní kalkulačky, nedosažitelná, ale po pravdě řečeno pro účely grafiky i zbytečná.

Obecné volání funkcí se řídí těmito **důležitými zásadami**:

- **všechny veličiny**, jak argumenty, tak funkční hodnoty, jsou **oměřítkované** reálné konstanty typu `s`
- jména funkcí začínají znaky `\R`
- argumenty funkcí jsou uzavřeny do kulatých závorek
- funkční hodnota se vrací oměřítkovaná v registrech `F` a `\F`. Ty má uživatel libovolně k dispozici i pro jiné (vlastní) použití.

Balík `rplot.sty` obsahuje následující elementární funkce:

`\Rabs(sx,sy)`	**Vzdálenost od počátku**	**F = abs(sx+i.sy)**

Makro počítá délku vektoru, modul komplexního čísla i vzdálenost bodu od počátku.

Příklad:

```
\Rabs(30000, 40000)%              F = 50000
\Rset \x = {3}\Rset \sy = {4}%    \sx = 30000, \sy = 40000
\Rabs(\x,\y)%                     F = 50000
```

`\Rpoly{sx}(seznam koeficientů)` **Hodnota polynomu** **F = p(x)**

Makro vypočte hodnotu polynomu

$$f(x) \;=\; a_n\,x^n + ... + a_1\,x + a_0$$

pro `x = sx` pomocí Hornerova schématu. Koeficienty a_i, které jsou rovněž typu `s`, jsou v seznamu uspořádány ve stejném pořadí jako ve formuli.

Příklad:
```
\Rpoly{22000}(30000,-20000,10000)%  F = 3*2.2^2 - 2*2.2 + 1
\wri{F}%                            *** F = 199200
\Rget \f={F}\wric\f%                *** \f = 19.9200
```

`\Rexp(sx)` **Exponenciála** **F = exp(sx)**

K výpočtu exponenciály se užila Padého aproximace

$$e^x \;\approx\; \frac{+x^3 + 12\,x^2 + 60\,x + 120}{-x^3 + 12\,x^2 - 60\,x + 120}$$

Ta však vyžaduje, aby $\|x\|<1$. Za tím účelem se nejdříve najde `y = x/2^n` splňující tuto podmínku, vypočte se `e^y` a tento výsledek se `n` krát umocní na druhou. Je pochopitelné, že se stále pracuje s oměřítkovanými veličinami typu `s`. Vinou toho však dochází u větších hodnot argumentů vlivem opakovaného násobení ke kumulaci zaokrouhlovacích chyb, takže výsledek se stává méně přesný. To by však pro grafické práce nemělo být na závadu s ohledem na dostatečný počet platných míst.

`\Rln(sx)` **Přirozený logaritmus** **F = ln(sx)**

Výpočet přirozeného logaritmu je založen na následujících krocích:

- V případě, že je `sx`$\leq$`0`, vydá se zpráva „Nonpositive argument of logarithm“ a opustí se vyhodnocování funkce;
- postupným půlením nebo zdvojnásobováním hodnoty argumentu `sx` za současného čítání resp. odečítání počtu půlení či násobení do `n` se přivede argument do intervalu `1` $\leq$ `sx` $\leq$ `2`;
- pro `sx = 1+x` se vypočte nejlepší aproximace (podle lit. [13], str. 72) jako:

  $$\ln(1+x) \approx \textstyle\sum_{k=4}^{4} a_k\,x^k$$

  s chybou $\varepsilon = 0,75.10^{-4}$ při koeficientech

  a_1 = 0,9974442 a_2 = -0,4712839
  a_3 = 0,2256685 a_4 = -0,0587527

  Vlastní výpočet je realizován s koeficienty 50-krát většími pro zajištění maximální přesnosti výsledku, tedy v oměřítkované formě jako:

  $$\ln x = ((((-29376\,x + 112834)\,x - 235642)\,x + 498722)\,x + n\,50\,\ln 2 \pm 25)/50,$$

  kde znaménko zaokrouhlovací korekce ± 25 je shodné se znaménkem výsledku.

`\Rlog(sx)` **Dekadický logaritmus** $\mathbf{F = \log_{10}(sx)}$

Logaritmus se základem 10 se počítá z formule:

$$\log x = 0.434294 \ln x$$

Pro udržení maximální přesnosti se výsledek získá normalizací se zaokrouhlením mezivýsledku až v posledním kroku.

`\Rpow(sx,sy)` **Reálná mocnina** $\mathbf{F = sx^{sy}}$

Počítá se za pomoci funkcí `\Rln` a `\Rexp` ze vzorce

$$x^y = \exp(y \ln x)$$

O přesnosti výsledku platí vše, co bylo řečeno u obou funkcí, totiž, že bude záviset na velikosti argumentů a možnostech uchování informace v aritmetice s pevnou řádovou čárkou. Situace je o to horší, že se zde kumulují zaokrouhlovací a aproximační chyby ze dvou funkcí. Nicméně použitelnost výsledku pro grafické práce zůstává stále velmi dobrá.

`\Rsqrt(sx)` **Druhá odmocnina** $\mathbf{F = (sx)^{0.5}}$

Výpočet druhé odmocniny je založen na Heronově iterační formuli jako zvláštním případu Newtonova vzorce:

$$y_{i+1} = \tfrac{1}{2}\left(y_i + \tfrac{x}{y_i}\right), \qquad i = 1,\ 2,\ \ldots$$

s počátečním odhadem $y_0 = 1$, což sice není optimální odhad, avšak je nejjednodušší a vyhovuje. Pokud je funkce vyvolána se záporným argumentem, vystoupí varování „Negative argument of square root“ na obrazovku a vypočte se odmocnina z jeho absolutní hodnoty.

`\Rsin(sx)` **Sinus úhlu** $\mathbf{F = \sin(sx)}$

Úhel se zadává jako oměřítkovaný, v radiánech, výsledek je oměřítkovaný v čítačích `F` a `\F`. Výpočet probíhá v následujících krocích:

- argument se transformuje do prvního a čtvrtého kvadrantu na $|y| \le 1$ jako zlomek $\pi/2$;
- vypočte aproximaci $\sin x = \sin \frac{\pi}{2} y$ ze vzorce z lit. [13], str. 101, upraveného do tvaru

$$\sin \tfrac{\pi}{2} y = (((-0.004362\, y^2 + 0.079488)\, y^2 - 0.645921)\, y^2 + 1.570795)\, y$$

S ohledem na možný výsledek z intervalu $\langle -1, 1\rangle$, realizují se výpočty s koeficienty zvětšenými stotisíckrát a převod na běžné oměřítkované veličiny typu `s` se pro zachování maximální možné přesnosti provádí až nakonec.

Příklad:

```
\def\RRsin #1(#2)%    Vlastní makro pro výpočet reálného sinu
{%  ~~~~~~~~~~~~~     #1=povel, #2=úhel ve stupních (real)
 \Rset F={#2}%            oměřítkování reálného úhlu
 \dtor F=F%               převod stupňů na radiány
 \Rsin(F)%                oměřítkovaný sinus úhlu
 \Rget #1=F%              převod na reálný povel
}%                    Konec makra \RRsin
%
\Ckbd \uhel%          Vstup reálného úhlu z klávesnice
\wric{\uhel}%         např. *** \uhel = 45
\RRsin \sinus(\uhel)% Vyvolání makra \RRsin
\wric{\sinus}%        např. *** \sinus = 0.7071
```

Pokud by se chtěl tentýž příklad počítat běžným způsobem, vypadal by např. takto:

```
\dtor F={450000}%     oměřít. stupně -> oměřít. radiány
\Rsin(F)%
\wri{F}%              *** F = 7071   (= oměřít. sinus)
```

`\Rcos(sx)` **Kosinus úhlu** **F = cos(sx)**

Úhel definovaný argumentem se před výpočtem zvětší o $\pi/2$ a pak se vyvolá makro `\Rsin`.

`\Rtan(sx)` **Tangens úhlu** **F = tan(sx)**

Makro nejdříve vypočte zvlášť sinus a kosinus úhlu udaného v radiánech. Potom otestuje zda není kosinus úhlu nulový. Pokud je, dodá jako tangens úhlu číslo 2000000000 (tj. po převodu na typ real číslo 200000.0000) se správným znaménkem. Jinak vypočte požadovaný tangens jako

$$\tan x = \sin x / \cos x$$

Protože je výsledek dán poměrem dvou aproximovaných funkcí, mohou se více projevit zaokrouhlovací chyby, které rostou s kosinem přibližujícím se nule.

Příklad:

```
\Cnt y={0}%
\Rfor x=(0:22:90)%                cykl stupňů
  {\dtor y=x \Rtan(y) \wri{F}}%
%   s výstupem na obrazovku:
% *** F = 0    správně: 0
% *** F = 4041          4040
% *** F = 9657          9657
% *** F = 22457         22460
% *** F = 286650        286363
```

V tomto příkladu a stejně tak i v dalších jsou použita makra pro opakování množin příkazů – cykly. Jde o tzv. reálný cykl `\Rfor` a v některých dalších příkladech to bude i celočíselný cykl `\Ifor`. Popisy obou maker jsou uvedeny v další kapitole.

`\Ratan(sa/sb)` **Arkustangens** **F = atan2(sa/sb)**

Makro počítá úhel, který je dán dvěma odvěsnami pravoúhlého trojúhelníka, přičemž `sa` je délka protilehlé odvěsny a `sb` je délka přilehlé odvěsny. Pokud je dán pouze tangens `tg` hledaného úhlu, dosadí se za argumenty `sa=tg, sb=10000`. Algoritmus výpočtu je následující:

- ze znamének složek `sa` a `sb` se určí korekce výsledného úhlu;
- pro z=`sa/sb` se vypočte aproximace úhlu z formule (viz lit. [13], str. 125):

  $$\arctan z \approx \sum_{k=0}^{4} a_{2k+1} x^{2k+1}$$

  dávající chybu aproximace $\varepsilon \leq 11 \,.\, 10^{-6}$ s koeficienty

  a_1 = 0.9998660 a_7 = -0.0851330
  a_3 = -0.3302995 a_9 = 0.0208351
  a_5 = 0.1801410

- Závěrem se výsledek zkoriguje na příslušný kvadrant.

Příklad:

```
\Cnt y={0}
\Rfor x=(0:22:90)%
  {\dtor y=x\Rtan(y)\Ratan(F/10000)\rtod y=\F \wri{y}}%
%
%       Výstup:      správně:
%    *** y = 0        0
%    *** y = 220016   220000
%    *** y = 439974   440000
%    *** y = 659933   660000
%    *** y = 880006   880000
```

`\Rdif(sx)` **Difrakční funkce** **F = sin(sx)/sx**

Makrem lze počítat známou funkci, která má své uplatnění ve fyzice, měření a jeho číslicovém zpracování. Pro hodnotu argumentu $x = 0$ vede na neurčitý výraz typu „0/0“. Tato singularita je v makru ošetřena.

Příklad:

```
\Rfor x=(-1:1:1){\Rdif(x)\wri{F}}
%     Výstup:
%  *** F = 8414
%  *** F = 10000
%  *** F = 8414
```

3.2 Statistické funkce

Soubor `rplot.sty` obsahuje jen nejzákladnější funkce z oblasti statistiky, totiž chybovou funkci `\Rerf(sx)`, distribuční funkci normálního rozložení `\RPn` a hustotu pravděpodobnosti `\Rgauss`. Všechny tyto funkce počítají jednu funkční hodnotu při každém vstupu.

`\Rerf(sx)` **Chybová funkce – error function** **F = erf(sx)**

Jde o funkci vyskytující se často v úlohách spojených se statistikou a jejími dalšími funkcemi. Je definována vztahem

$$\operatorname{erf} x \;=\; \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2}\, dy$$

Lze ji aproximovat formulí převzatou z lit. [14], str. 299:

$$\begin{aligned} \operatorname{erf} x &= 1 - (a_1\, t + a_2\, t^2 + a_3\, t^3)\, e^{-x^2}, \quad \text{kde} \\ t &= \frac{1}{1 + p\, x} \end{aligned}$$

Pro $\mathtt{a_1}$ = 0.3420242 p = 0.47047
$\mathtt{a_2}$ = -0.0958798
$\mathtt{a_3}$ = 0.7478556

je deklarovaná chyba tohoto postupu $|\varepsilon| \leq 25 \,.\, 10^{-6}$. Realizovaný algoritmus je přímým přepisem těchto vztahů do zobrazení v pevné řádové čárce s koeficienty zvětšenými 10-krát. V závěru výpočtu se výsledek upraví na správnou hodnotu.

Příklad:

```
\Rfor x=(-1.5:.5:1.5)%
{\Rerf(x)%
 \Rget\x={x}%
 \Rget\y={F}
 \wris{x = \x, erf(x) = \y}
}
```

Výsledek (na obrazovce):

```
*** x = -1.5000, erf(x) = -0.9661
*** x = -1.0000, erf(x) = -0.8427
*** x = -0.5000, erf(x) = -0.5204
*** x = 0.0000, erf(x) = 0.0000
*** x = 0.5000, erf(x) = 0.5204
*** x = 1.0000, erf(x) = 0.8427
*** x = 1.5000, erf(x) = 0.9661
```

`\RPn(sx,smean,sigma)` **Distribuční funkce normálního jevu**

Definuje se vztahy

$$\begin{aligned} F_n(x) &= \int_{-\infty}^{x} f_n(t)\, dt = \mathsf{P}_n(X \leq x)\ , \\ \text{kde} \qquad f_n(x) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \end{aligned}$$

je funkce hustoty pravděpodobnosti normálního rozdělení se střední hodnotou μ a směrodatnou odchylkou σ. $\mathsf{P}_n(X \leq x)$ je pravděpodobnost jevu s normálním rozdělením za podmínky dané argumentem. Parametry funkce `\RPn` jsou:

`sx` oměřítkovaná reálná nezávisle proměnná x,
`smean` oměřítkovaná reálná střední hodnota μ,
`sigma` oměřítkovaná reálná směrodatná odchylka σ.

Hodnota integrálu se může pro normální rozdělení vyjádřit i pomocí chybové funkce (tedy vyvoláním makra `\Rerf`) ve tvaru:

$$F_n(x) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

Příklad:

```
\Cnt smean={0}%                    mean  = 0
\Cnt sigma={10000}%                sigma = 1
\Cnt sx={-20000}%                  x = -2.0
\RPn(sx,smean,sigma)\wri{F}%       *** F = 227   (0.0227501)
\RPn({20000},smean,sigma)\wri{F}%  *** F = 9772  (0.9772499)
```

`\Rgauss(sx,smean,sigma)` **Hustota pravděpodobnosti normálního jevu**

Funkce je dána již výše uvedeným předpisem

$$f_n(x) \;=\; \frac{1}{\sigma\sqrt{2\pi}}\,\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

Všechny parametry jsou oměřítkované reálné položky, stejné jako u distribuční funkce. Výpočet sám je přímým vyhodnocením této formule při použití funkce `\Rexp`.

Příklad:

```
\Rgauss(0,0,10000)\wri{F}%          *** F = 3989
\Rgauss(5000,5000,20000)\wri{F}%    *** F = 1995
```

3.3 Náhodná čísla a procesy

Do kapitoly věnované funkcím je zařazen i tento odstavec. Popisují se v něm makra, jimiž lze generovat pseudonáhodná čísla a s jejich využitím i pseudonáhodné procesy. Protože se nejedná o normální funkční předpisy, liší se jejich zadávání i vyvolávání od normálních funkcí. Nemají např. žádný argument a ani jejich výstup není ukládán do čítače `F`, jak je u běžných funkcí obvyklé.

Cílem maker pochopitelně není řešení stochastických úloh v LaTeXu, ale vygenerování dat pro grafická zobrazení. Proto množina maker pro tyto účely je relativně chudá a obsahuje procedury pro iniciaci generátorů `\Rndini`, generátor pseudonáhodných čísel s rovnoměrným rozložením `\Rngu`, a dva generátory pseudonáhodných procesů – `\Rpgrc` a `\Rpgn`. Pokud by uživatel potřeboval skutečně řešit stochastické úlohy, musí použít jiný prostředek.

Makra pracují s odlišnou soupravou registrů a proto mohou být používána simultánně s ostatními funkcemi, aniž by docházelo k nežádoucím interferencím. Při každém vyvolání generátorů se vytvoří nový výsledek – pseudonáhodné číslo. To je vraceno ve

zvláštním čítači, který patří konkrétnímu generátoru. Uživatel může tyto čítače se jmény `RU, RN, RP` libovolně používat, tedy i měnit jejich obsahy.

`\Rndini{seed}{cflen}{alpha}` **Iniciace pseudonáhodných generátorů**

Jde o jediné makro pro všechny generátory. Obsahuje proto všechny parametry, které lze nastavovat, i když nejsou pro danou aplikaci zrovna nutně zapotřebí. Jeho parametry jsou:

`seed` je celočíselná nezáporná počáteční násada generátoru pseudonáhodných čísel z intervalu $\langle 0, 32767\rangle$. V případě, že se zadá prázdný argument `{}`, nedojde ke změně násady, což znamená, že zůstane buď implicitní (nulová), anebo jaká byla v okamžiku vyvolání makra. Při neprázdném argumentu `seed` se jeho hodnotou naplní čítač `RU`.

`cflen` je požadovaná délka korelace procesu vytvářeného pomocí generátoru `\Rpgn`. Není-li zapotřebí, může být argument prázdný;

`alpha` je váhový koeficient staré hodnoty procesu s exponenciální korelační funkcí. Není-li zapotřebí, může být argument prázdný.

`\Rngu` **Pseudonáhodná čísla s rovnoměrným rozdělením**

Generátor při každém vyvolání vytvoří pseudonáhodné celé číslo (vzorek bílého šumu) s rovnoměrným rozdělením z intervalu $\langle 1, 65535\rangle$ podle předpisu

$$u_{i+1} = (a\,u_i + r) \pmod m$$

Jde o tzv. smíšený kongruenční generátor, na jehož koeficienty `a` a `r` jsou kladeny zvláštní podmínky, aby délka posloupnosti čísel, kterou bude generovat, byla maximální. Předložené makro používá `a = 5061` a `r = 13849` jako osvědčené hodnoty koeficientů.

Příklad:

```
\Rndini{19327}{}{}
\Ifor i=(1:1:9){\Rngu \T{\theRU,\ }}
```

Výsledek:

```
48084, 31805, 22538,
46027, 41552, 3497,
17446, 31063, 2828,
```

I když je soubor `rplot.sty` vybaven pouze jedním generátorem pseudonáhodných čísel, může být využit simultánně pro řadu účelů. Generovaná čísla mají teoreticky nepatrnou korelaci, a tak není třeba mít obavy o „prosakování“ korelované složky do různých procesů.

`\Rpgn` **Náhodný proces s trojúhelníkovou korelační funkcí**

Makrem lze generovat body náhodného procesu s rozdělením blížícím se normálnímu. Předpis pro generování jedné hodnoty procesu je velmi jednoduchý, i když dosti náročný na programovací možnosti LaTeXu:

$$p_i = \sum_{j=i}^{i+n-1} u_j$$

- vygeneruje se posloupnost n `= cflen` pseudonáhodných čísel `RU` jako speciální seznam a čísla se sečtou a uloží do `RP` jako p_1;
- z posloupnosti čísel se vypustí první a jeho hodnota se odečte od `RP`
- vygeneruje se nové pseudonáhodné číslo `RU`, zařadí se na konec posloupnosti a přičte se do `RP`, které tak nese novou hodnotu p_i procesu a makro se opouští.
- Při dalším vstupu do makra se první bod algoritmu vynechává.

Korelační funkce procesu p vytvářeného podle horního vzorce bude mít trojúhelníkový tvar a délku korelace právě `n` bodů.

Příklad:

```
\Rndini{17453}{30}{}%
\Ifor i=(1:1:15)%
{\Rpgn \Rget\rp={RP}\T{\rp,\ }}
```

Výsledek:

```
10.7427,  16.8566,  13.2053,  15.1280,  13.8167,  16.1242,  15.8665,
15.8132,  14.7595,  10.0606,  11.9421,  11.8328,  15.7215,  15.9714,
15.8705,
```

`\Rngrc` **Náhodný proces s exponenciální korelační funkcí**

Stejně jako předcházející generátor dodá `\Rngrc` na každé zavolání jednu hodnotu náhodného procesu, tentokrát však s korelační funkcí exponenciálně doznívající. Příčinou tohoto jevu je jednoduchý číslicový filtr, který se aplikuje na dvě poslední hodnoty pseudonáhodných čísel – bílého šumu podle formule:

$$q_i \;=\; \alpha\, q_{i-1} + \beta\, u_i \ ,$$

přičemž pro stabilní filtr musí platit, že $\alpha + \beta \leq 1$. Z podmínky rovnosti plyne možnost zadávat pouze váhu „staré“ hodnoty procesu a novou vypočítat jako doplněk do jedničky. Implicitní hodnoty koeficientů jsou (oměřítkované) $\alpha = 7500$ a $\beta = 2500$. Čím větší je α, tím pomaleji se mění výstupní hodnota uložená v čítači `RP`. Filtr je tak číslicovým analogem RC-filtru v elektronice. Odtud byla vzata i poslední dvě písmena názvu makra.

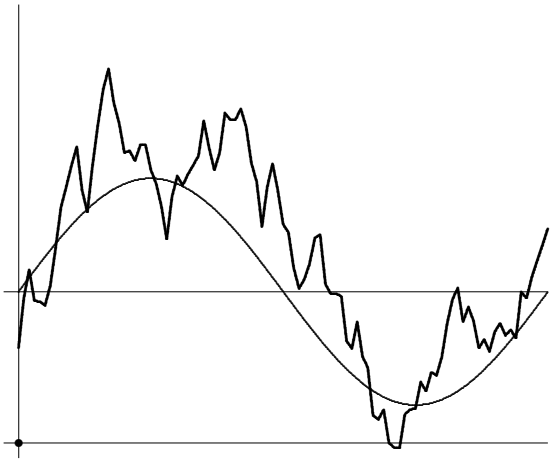
Příklad:

```
\Cmd \rp={} \Rndini{4735}{}{5000}%   *** \rp = 12138, 3411, -6097,
\Ifor i=(1:1:9)%                          -1667, -14489, -5624,
{\Rpgrc \Cmd\rp={\rp\theRP, }}\wric{\rp}%      13463, 2679, 11329,
```

Pro větší názornost bude vhodné probrat složitější příklad tvorby obecnější funkce a tu i zobrazit. O vlastním zobrazení se zde nebudeme šířit, protože je podrobně vysvětleno v odstavci 6.3.1. Úloha zní: Vypočítat a nakreslit jednu periodu zašuměného harmonického signálu se střední hodnotou. Signál má periodu $m = 100$ vzorkovacích period, amplitudu 15 a střední hodnotu $\mu = 20$, šum má korelační funkci s délkou korelace $T_c = 10$ vzorkovacích period. Zadání se realizuje v následujících krocích:

- pro každou vzorkovací periodu řešení se sečte střední hodnota μ s hodnotou $\sin 2\pi k/m$ a váženou hodnotou náhodného procesu p_k;
- výsledek se připojí k seznamu `\pt`
- po dokončení výpočtu se výsledný seznam vynese v přijatelném měřítku.

```
\Cnt {omt}={0}%      omega.t
\Cnt {domt}={628}%   oměř. 2.pi/m
\Cmd \proces={}%
\Cmd \sinus={}%
\newcnt{w}%
\Rndini{54321}{10}{}
%
\Ifor i=(0:1:100)%   Cykl period
{\setn{w}{200000}%   mu
 \Rsin(omt)\muln{F}{15}%
 \Rapps \sinus+\theF
 \addc{w}{F}%        mu+15sin(.)
 \Rpgn%              náh. proces
 \addc{w}{RP}%       mu+15sin(.)+p
 \Rapps \proces+\thew%
 \Radd {omt}={omt}+{domt}%
}
\unitlength1mm
\begin{picture}(160,80)(-30,-20)
  \put(0,0){\circle*{1}}
  \linethickness{.2pt}
  \put(-2,0){\line(1,0){72}}
  \put(0,-2){\line(0,1){80}}
  \put(-2,20){\line(1,0){72}}
  \thinlines
  \setscales(.7,0,0,1)
  \scaleput(0,20){\polylineq(\sinus)}
  \thicklines
  \scaleput(0,0){\polylineq(\proces)}
\end{picture}
```



Kapitola 4

Příkazy cyklů

Příkaz cyklu patří mezi základní programovací struktury. Využívají se všude tam, kde je zapotřebí opakovat jistou činnost pouze se změněnými parametry. Sázecí systém TEX a jeho nadstavba LATEX obsahují několik typů cyklů. Z nich však pro běžného uživatele LATEXu je k dispozici pouze jeden cyklus se jménem `\whiledo` obsažený v balíku procedur v souboru se jménem `ifthen.sty`. Pro uživatele se stane dosažitelný po uvedení jeho názvu v seznamu příkazu

```
\documentstyle[...,ifthen,...]{...}
```

Při použití stylu `rplot.sty` není zapotřebí uvádět jméno `ifthen` v seznamu, protože tuto potřebu si ošetří `rplot.sty` sám. Samotný soubor však v adresáři stylů již být musí. Pro účely stylu `rplot.sty` byly zpracovány ještě další typy cyklů, a to reálný, dva celočíselné a tři speciální. Všechny typy cyklů nyní popíšeme:

4.1 Cykl \whiledo a podmíněný příkaz \ifthenelse

Jde o cykl typu `while`, známý ze strukturovaného programování, o tvaru

```
\whiledo{TEST}{tělo cyklu}
```

Cykl typu while

`TEST` je podmínka (logický výraz), která se vyhodnotí, a v případě, že je pravdivá `(TRUE)`, vykoná se `tělo cyklu`, které tvoří množina příkazů. Potom se běh programu vrátí na začátek cyklu k novému otestování podmínky. Ta může mít některý z tvarů:

```
číslo A < číslo B
číslo A = číslo B
číslo A > číslo B
\equal{řetěz A} {řetěz B}
\(TEST \)
\not TEST
TEST \or TEST
TEST \and TEST
```

`tělo cyklu` je množina libovolných příkazů LATEXu.

Má-li cykl někdy skončit, **musí** příkazy v těle cyklu měnit hodnoty proměnných vstu-

pujících do podmínky TEST.

Příklad:
```
\whiledo{\value{i}>100}{}{\wri{i}\incc{i}}
\whiledo{\value{j}<\value{i}}{\incc{j}}%
   {\wris{KONEC po \thej cyklech}}
```

Pro úplnost dodejme, že mezi příkazy těla cyklu, ale i jinde se může vyskytnout podmíněný příkaz, který je rovněž součástí stylu `ifthen`. Má tvar

`\ifthenelse {TEST}{větev TRUE}{větev FALSE}`	**Podmíněný příkaz**

kde

`TEST`	je podmínka o stejné syntaxi jako u příkazu `\whiledo`,
`větev TRUE`	je množina libovolných příkazů, které se provedou v případě, že je podmínka splněna,
`větev FALSE`	je množina libovolných příkazů, které se provedou při nesplnění podmínky `TEST`.

Libovolná z množin příkazů může být prázdná.

Příklad: `\ifthenelse{\value{F}<0}{\setn{F}{-\value{F}}}{}% abs{F}`

Běžný uživatel obvykle zná cykly vyskytující se ve většině vyšších programovacích jazyků, které automaticky inkrementují proměnnou cyklu a testují dosažení meze intervalu povolených hodnot. Jsou to cykly typu `for` známé z `BASICu, FORTRANu, PASCALu, MATLABu` a dalších.

4.2 Reálný cykl \Rfor

Dostal jméno podle toho, že parametry cyklu jsou reálné položky. Jeho tvar je:

`\Rfor Scnt=(rbeg:rstep:rfin){tělo cyklu}`	**Reálný cykl**

Zde

`Scnt`	zde zastupuje obecný čítač, tj. oměřítkovanou reálnou proměnnou cyklu typu `s`, jejíž jméno buď již dříve uživatel nadefinoval, anebo bude příkazem `\Rfor` teprve vytvořena. Ta bude obsahovat v jednotlivých cyklech postupně hodnoty 10^4`*rbeg, `10^4`*(rbeg+rstep), `10^4`*(rbeg+2*rstep),...`
`rbeg`	je reálná položka, která se po oměřítkování dosadí jako počáteční hodnota proměnné cyklu `scnt`
`rstep`	je reálná položka, po oměřítkování krok změny proměnné cyklu `Scnt`
`rfin`	je reálná položka – hodnota konce reálného intervalu. Proměnná cyklu dosáhne (oměřítkované) hodnoty `|Scnt|`$\leq$ `|rfin|*`10^4
`tělo cyklu`	je libovolná posloupnost příkazů, která může libovolně používat proměnnou cyklu `Scnt`, ale **nesmí** ji měnit.

Poznámky:

- Parametry cyklu – reálné položky `rbeg, rstep` a `rfin` se převedou na jim odpovídající oměřítkované reálné ekvivalenty (celá čísla) a ty se pak užijí při naplňování a modifikaci obsahu proměnné cyklu – čítače `Scnt`.
- Reálnou položkou se zde rozumí
  - reálné číslo,
  - povel s reálným číslem,
  - příkaz `\theLčítač`, nebo `Tčítač` jehož obsah bude chápán jako reálné číslo.

Příklad:

```
\Rset dx=\Rpi\divn{dx}{8}% 4 body/kvadrant
\Rget \dx={dx}%            konverze na real povel
\Cmd \xy={}%
\Rfor x=(0:\dx:\Rpi)%      výpočet a tisk
{\Rsin(x)\Rget\x={x}\Rget\y={F}%
 \Cmd \xy={\xy \x & \y \cr}%
}{\bf \s\ \ Výsledek:}
\begin{tabular}[t]{rr}
  \bf x\ \ \ \s & \bf sin x\ \s\\ \\
  \xy
\end{tabular}
```

Výsledek:

x	sin x
0.0000	0.0000
0.3927	0.3827
0.7854	0.7071
1.1781	0.9239
1.5708	1.0000
1.9635	0.9239
2.3562	0.7071
2.7489	0.3827
3.1416	0.0000

4.3 Celočíselné cykly \Ifor a \Lfor

Jejich označení vyplývá z faktu, že pracují pouze s celočíselnými informacemi, tedy typu `i` nebo `s`. Přesto se od sebe výrazně odlišují, protože první z nich, `\Ifor`, je „klasického“ typu, v němž se hodnota proměnné cyklu se mění lineárně, zatímco druhý, `\Lfor`, ji mění dle zadaného seznamu.

`\Ifor cnt=(beg:step:fin){tělo cyklu}` **Celočíselný cykl**

Parametry cyklu jsou:

`cnt`	Zastupuje zde obecný čítač – proměnnou cyklu, jejíž jméno buď bylo již dříve definováno uživatelem pomocí příkazů `\newcounter, \newcount` nebo `\newcnt`, anebo si ho nadefinuje makro samo,
`beg`	je položka typu `i` nebo `s` = počáteční hodnota `cnt`,
`step`	je položka typu `i` nebo `s` = krok změny `cnt`,
`fin`	je položka typu `i` nebo `s` = konečná hodnota intervalu změn `cnt`, pro níž platí $\texttt{|fin|} \geq \texttt{|cnt|}$,
`tělo cyklu`	je libovolná posloupnost příkazů, která **nesmí** měnit proměnnou cyklu `cnt`, i když ji může libovolně používat.

Pod pojmem položka typu `i` nebo `s` se zde rozumí:

- celé číslo,
- povel s celým číslem,
- `\thečítač`, kde `čítač` je jméno obecného čítače obsahujícího celé číslo.

Příklad:
```
\newcounter{j}
\Ifor i=(0:1:25)%
   { \setc{j}{i}\mulc{j}{j}\wri{j}}%  tiskne 0, 1, 4, 9, ...
\Cmd\krok{12}
\Ifor j=(0:\krok:30){ příkazy }%      proběhne s j=0, 12, 24
```

Na rozdíl od cyklu `\Ifor`, jehož proměnná cyklu se mění mezi jednotlivými opakováními s konstantním krokem, dává cykl `\Lfor` možnost měnit proměnnou cyklu libovolně, průchod od průchodu. Onu „libovolnost“ umožňuje předpis pro přiřazování hodnot proměnné cyklu, jímž je celočíselný seznam:

`\Lfor cnt=(seznam){tělo cyklu}` **Celočíselný seznamový cykl**

Význam parametrů cyklu je následující:

`cnt`	je obecný čítač, který se eventuálně i nadefinuje,
`seznam`	je seznam celočíselných položek, tj. celých čísel (včetně oměřítkovaných reálných), výrazů typu `\value{Lčítač}`, `\theLčítač`, `\Tčítač`, `celočíselný povel` a jejich libovolná kombinace,
`tělo cyklu`	je libovolná posloupnost příkazů, která smí používat hodnotu proměnné cyklu `cnt`, ale **nesmí** ji změnit.

Příklad:

```
\Cnt n={0}
\Lfor m=(31,28,31,30,31,30,31,31,30,31,30,31)%
  {\incc{n}
   \s\qquad V~normálním roce má \then . měsíc \them\ dnů.\\}
```

s výstupem do dokumentu:

V normálním roce má 1. měsíc 31 dnů.
V normálním roce má 2. měsíc 28 dnů.
V normálním roce má 3. měsíc 31 dnů.
V normálním roce má 4. měsíc 30 dnů.
V normálním roce má 5. měsíc 31 dnů.
V normálním roce má 6. měsíc 30 dnů.
V normálním roce má 7. měsíc 31 dnů.
V normálním roce má 8. měsíc 31 dnů.
V normálním roce má 9. měsíc 30 dnů.
V normálním roce má 10. měsíc 31 dnů.
V normálním roce má 11. měsíc 30 dnů.
V normálním roce má 12. měsíc 31 dnů.

Příklad má pouze instrukční cenu, protože ukazuje, jak lze směšovat textové informace a aritmetické operace v cyklu pro dosažení cíle.

4.4 Speciální cykly pro výpočty funkcí

Při využívání souboru `rplot.sty` pro vynášení průběhů funkcí se často setkáváme s potřebou zařazení souřadnic bodů funkce do seznamu. Tuto činnost lze sice obvykle zvládnout pomocí již popsaných příkazů, ale pro zjednodušení práce uživatele je účelné mít k dispozici speciální makra.

Systém `rplot.sty` obsahuje tři makra pro cyklický výpočet funkcí. Jejich hlavním cílem je vytváření seznamů souřadnic bodů funkcí pro vynášení diagramů. Při tom jak nezávisle proměnná, tak i pořadnice funkce (funkční hodnoty) budou do seznamu zařazovány jako ztransformované s ohledem na použité stupnice grafů. Protože všechna tři makra mají velmi podobné argumenty, popíšeme je ihned:

`\xy`	Výsledný seznam souřadnic bodů, tak jak budou vynášeny na kreslicí ploše, tj. včetně všech transformací zobrazení;
`funx`	transformační předpis pro souřadnici `x` posílající výsledek transformace do Lčítače `F`;
`funy`	funkční předpis pro pořadnice bodů včetně transformace na stupnici osy `y`. Výsledek předpisu se musí uložit do Lčítače `F`;
`S`	obecný čítač – proměnná cyklu;
`(předpis)`	individuální předpis změn proměnné cyklu.

`\Rfunq \xy={funx}{funy},S=(rbeg:rstep:rfin)`

Cykl výpočtu funkcí s ekvidistantním x

Makro je příbuzné s makrem reálného cyklu, které samo ho i využívá. Tomu odpovídá i tvar posledního komplexního parametru – předpisu pro změnu nezávisle proměnné, který je zcela totožný se stejným parametrem reálného cyklu.

Příklad:

```
\Cmd \dx={0.7854}%  \dx  = reálný krok - povel
\Cmd \xm={6.2832}%  xm   = 2*pi (real) - povel
\Rfunq \xfx=%       Reálný cykl výpočtu funkce
  {\setc{F}{x}}%    funx : F = x
  {\Rsin(x)%        funy : F = sin(x)
   \wris{x = \thex, y = \theF}%
  },x=(0:\dx:\xm)%

   *** x = 0.0000,   y = 0.0000
   *** x = 0.7854,   y = 0.7071
   *** x = 1.5708,   y = 1.0000
   *** x = 2.3562,   y = 0.7071
   *** x = 3.1416,   y = 0.0000
   *** x = 3.9270,   y = -0.7071
   *** x = 4.7124,   y = -1.0000
   *** x = 5.4978,   y = -0.7071
   *** x = 6.2832,   y = 0.0000
```

`\Rfunx \xy={funx}{funy},S=(seznam_x)`

Cykl výpočtu funkcí pro x ze seznamu

Předpis pro změnu nezávisle proměnné cyklu je seznam souřadnic `x`. Není zapotřebí zadávat počet bodů, protože ten se nepotřebuje. Cykl probíhá tak dlouho s postupným výběrem souřadnic `x`, pro něž se počítají funkční hodnoty `y`, dokud se nevyčerpá celý `seznam_x`. Ten je běžným seznamem reálných položek `x` navzájem oddělených čárkami.

Příklad: Výpočet chybové funkce erf(x)

```
\odraz{3}x\hskip5ex erf(x)
\lines{1.5}
\Rfunx \fx={\Set{F}{x}}%          funx
           {\Rerf(x)%             funy
            \Rget\x={x}\Rget\y=F%
            \x \quad \y \\
           },x=(0,.5,1,1.5,2,3,5)
```

x	erf(x)
0.0000	0.0000
0.5000	0.5204
1.0000	0.8427
1.5000	0.9661
2.0000	0.9953
3.0000	1.0000
5.0000	1.0000

`\Rfunxy \xy={funx}{funy},S=(seznam_xy)`

Cykl transformace souřadnic x a y ze seznamu

Makro je určeno zejména pro případy vynášení grafů funkcí, jejichž body jsou již známy, ale je zapotřebí je ztransformovat nelineárním předpisem pro dané stupnice grafu. Předpis `seznam_xy` je běžným seznamem dvojic reálných souřadnic `x` a `y` bodů funkce. Položky jsou navzájem odděleny čárkami.

Příklad:

Přepočet bodů do logaritmických souřadnic

```
\Cmd\txy={x \ &y \ &%
  \log{x} \ & \log{y} \ \cr\cr}%
\Rfunxy \fxy=%
  {\Rlog(xy)%
   \Rget\cx={xy}\Rget\cfx={F}%
  }%
  {\Rlog(xy)%
   \Rget\cy={xy}\Rget\cfy={F}%
   \Cmd\txy={\txy \cx & \cy &%
      \cfx & \cfy \cr}
  },xy=(1,1, 2,4, 3,9, 4,16, 5,25)%
$
\begin{array}[t]{rrrr}
  \txy%
\end{array}
$
```

x	y	$\log x$	$\log y$
1.0000	1.0000	−0.0001	−0.0001
2.0000	4.0000	0.3010	0.6021
3.0000	9.0000	0.4772	0.9543
4.0000	16.0000	0.6021	1.2041
5.0000	25.0000	0.6990	1.3980

Všechna tato tři makra vytvářejí seznamy. Protože jde o činnost dosti častou, byla pro vytváření seznamů připravena dále uvedená makra, která jsou k volnému použití. Před jejich vyvoláním je zapotřebí iniciovat výsledný seznam příkazem `\Cmd\xy{}`.

`\Rappr \xy+{položka}` **Přidání položky do seznamu**

Na konec seznamu se přidá obecná položka (i nečíselná) s případným oddělovačem, takže výsledný seznam je syntakticky správný, a lze ho použít v cyklu `\Lfor`.

Příklad:

```
\Cmd \XY={} \\
\Rappr \XY+{12.3} \bs XY = \XY \\
\Rappr \XY+{4.56} \bs XY = \XY
```

\XY = 12.3
\XY = 12.3,4.56

`\Rapps \xy+{sx}` **Přidání položky po transformaci na real**

Na konec seznamu se po transformaci na reálné číslo připojí původně oměřítkovaná položka `sx` předcházená případným oddělovačem.

Příklad:

```
\Cmd \XY={} \\
\Rapps \XY+{123000} \bs XY = \XY \\
\Rapps \XY+{45600} \bs XY = \XY
```

\XY = 12.3000
\XY = 12.3000,4.5600

`\Rappxy \xy+(sx,sy)` **Přidání páru položek po převodu na real**

Na konec seznamu se připojí s příslušnými oddělovači (čárkami) dvě reálná čísla, která se získala z oměřítkovaných reálných položek `sx` a `sy`.

Příklad:

```
\Cmd \XY={} \\
\Rappxy \XY+(123,345) \bs XY=\XY\\
\Rappxy \XY+(567,789) \bs XY=\XY
```

\XY=0.0123,0.0345
\XY=0.0123,0.0345,0.0567,0.0789

`\Rapptab \txy+(řádka)` **Přidání řádky do seznamu po převodu na real pro prostředí array a tabular**

Výsledky výpočtu se někdy potřebují jak v grafické, tak i tabulkové formě. Běžný seznam s čárkami jako oddělovači není pro použití v prostředích `array` a `tabular` vhodný. Proto bylo sestaveno makro, které z dílčích obyčejných řádek tabulky vytvoří speciální seznam s oddělovači "&", který může být dodatečně vysázen v jednom z uvedených prostředí. Parametry makra jsou:

- `\txy` uživatelův povel pro tabulkový seznam, který musí být před prvním vyvoláním makra `\Rapptab` nadefinován jako prázdný;
- `řádka` je obyčejný seznam oměřítkovaných prvků, které mají být vysázeny jako reálná čísla.

Seznam `řádka` se prvek po prvku konvertuje na reálné povely, ty se prokládají znaky

`&` a za posledním prvkem se dodá znak nové řádky.

Příklad:

Jako příklad uveďme stejnou úlohu, která byla řešena za použití makra `\Rfunxy`, avšak s tím rozdílem, že seznam hodnot proměnné cyklu bude tentokrát celočíselný, tedy s prvky chápanými v pevné řádové čárce 10000-krát menšími. To má za následek, že logaritmy jsou nižší o 4.

```
\newcnt{n}\newcnt{Fx}%
\Cmd\txy={x \ &y \ &\log{x} \ &\log{y} \ \cr\cr}%
\setn{n}{0}%
\Lfor xy=(1,1, 2,4, 3,9, 4,16, 5,25)%
  {\ifthenelse{\then=0}%    x:
    {\incc{n}%
     \Rlog(xy)\setc{Fx}{F}%
     \Cmd \row={\thexy}%
    }%                      y:
    {\setn{n}{0}%
     \Rlog(xy)%
     \Cmd \row={\row,\thexy,\theFx,\theF}
     \Rapptab \txy+(\row)%
    }%
  }%
$
\begin{array}[t]{rrrr}
  \txy%
\end{array}
$
```

x	y	$\log x$	$\log y$
0.0001	0.0001	−4.0001	−4.0001
0.0002	0.0004	−3.6991	−3.3980
0.0003	0.0009	−3.5229	−3.0458
0.0004	0.0016	−3.3980	−2.7960
0.0005	0.0025	−3.3011	−2.6021

Kapitola 5

Vstupy, výstupy a seznamy

Koncepce LaTeXu takřka nepočítá s potřebou pracovat s informacemi, které by mohly operativně vstupovat z vnějšku, příp. vystupovat na obrazovku během zpracování dokumentu. Proto LaTeX poskytuje jen omezené možnosti operativních vstupů a výstupů.

Pro operativní **vstup** z klávesnice je k dispozici příkaz

```
\typein[\povel]{libovolný text}
```

Uživatelem zvolené jméno povelu se tímto makrem i nadefinuje. Při vyvolání se na obrazovku vypíše text zapsaný ve složených závorkách a čeká se na klávesnicový vstup. Po jeho ukončení znakem `Enter` lze `\povel` použít. Tak např. po vložení celého čísla lze ho následně poslat do již definovaného čítače `n` příkazem `\setn{n}{\povel}`. Povel lze i vykonat pouhým zápisem jeho jména.

Pro operativní **výstup** na obrazovku umožňuje LaTeX použít příkaz

```
\typeout{libovolná zpráva}
```

Text `libovolná zpráva` může obsahovat i povely, které se však před výstupem zprávy na obrazovku rozvinou.

Existuje řada aplikací, které si operativní vstupy a výstupy přímo vynucují. Proto byly sestaveny skupiny maker pro vstup dat z klávesnice, ze souborů a pro kontrolní výstupy informací na obrazovku.

5.1 Vstupy z klávesnice

S ohledem na potřeby vstupů různých druhů informace pro aritmetické aplikace byl původní příkaz `\typein` zabudován do skupiny zvláštních maker s rozlišnou působností:

`\Ikbd{icnt}`	**Vstup celého čísla**	**icnt = vstup**

Makro vydá na obrazovku výzvu

```
******* Integer: icnt *******
\integ=_
```

po níž čeká na vložení celého čísla z klávesnice. Vstup obsahující pouze číslice příp. se znaménkem, celočíselný povel, anebo jméno obecného čítače se zakončí znakem

`Enter`. Vstupní informace se zkonvertuje na celé číslo, které se potom uloží do obecného čítače `icnt`, který nemusel být předem definován.

Příklad:
```
\Ikbd{kusu}%   vstup znaků: - 1 2 3 Enter
\Ikbd\cyklu %  vstup znaků: 2 5 Enter
```

Po provedení těchto příkazů bude Lčítač `kusu` obsahovat číslo `-123` a Tčítač `\cyklu` číslo `25`. Při prázdném vstupu, tzn. při pouhém odklepnutí znaku `Enter`, se do čítače dosadí nula.

`\Rkbd{Scnt}` **Vstup reálného čísla** **Scnt = vstup**

Při interpretaci makra dojde k výstupu textu

```
******* Real: Scnt *******
\real=_
```

na obrazovku a k čekání na uživatelovu odezvu. Ta může obsahovat nepovinné znaménko, nepovinnou serii číslic s eventuální tečkou, nebo jméno čítače, případně reálný povel a povinný znak `Enter`. Při pouhém odklepnutí znaku `Enter` se do obecného čítače `Scnt` dosadí nula. Jinak se vložená informace považuje za reálné číslo, to se oměřítkuje a jako celé uloží do čítače `Scnt`. V případě, že nebyl dosud definován příkazy `\newcounter`, `\newcount` nebo nejlépe pomocí `\newcnt`, příkaz `\Rkbd` ho vytvoří.

Příklad:
```
\Rkbd{vklad}%                vstup znaků: 9 8 7 . 5 Enter
\Rkbd{urok}%                 vstup znaků: 8 . 2 5 Enter
\Rmul vklad={vklad}*{urok}%  nový vklad*100
\divn{vklad}{100}%           přepočet na nový vklad
\Rget\nový={vklad}%          Převod na real
Nový vklad činí \nový Kč%    text do dokumentu
```

Výstupní text bude

```
Nový vklad činí 8146.8750 Kč
```

`\Ckbd\povel` **Vstup příkazu** **\povel = vstup**

Makro způsobí výstup textu

```
******* Command: \povel *******
\cmd=_
```

a čeká na vstup z klávesnice. Ten se skládá z libovolné kombinace znaků zakončené znakem `Enter`. Tato posloupnost znaků se uloží do nově vytvořeného povelu `\povel`, jehož jméno si zvolil uživatel sám. Kdykoliv se v budoucnosti užije tento povel, v dokumentu, vystoupí vložená posloupnost znaků, pokud neobsahovala další povely (neterminální symboly). V případě, že je obsahovala, dojde k dalšímu rozkladu a výstupu konečného textu.

Příklad:

```
\Cmd \fin={no}%
\whiledo {\equal {\fin}{no}}%
{ příkazy
  \Ckbd{\fin}%       odpověď
}%                   konec \whiledo
```

Při odpovědi `no` se bude cykl opakovat, při jiné cykl končí. Tato odpověď může vypadat např. takto: `\hfill\today`, což způsobí, že na konci právě zpracovávané řádky vystoupí datum vyhotovení dokumentu.

5.2 Výstupy na obrazovku

Při práci na nových makrech i na složitějším dokumentu je pro uživatele často důležité znát okamžité hodnoty veličin, se kterými se pracuje, případně místo, které se právě zpracovává. Pro tyto účely lze sice užít již zmíněný příkaz `\typeout`, avšak pro častěji se vyskytující typy výstupů je výhodnější sestavit účelová makra. Dočasný výstup informace o ladění do cílového dokumentu není vhodný, protože právě stránka, na níž došlo k chybě nevystoupí a tedy ani nejdůležitější informace o stavu před chybou nejsou k dispozici. Ani v případech, kdy je objem výstupů na obrazovku velký, není výstupní informace ztracena, protože se ukládá do souboru `*.log`, který lze dodatečně prohlížet. Pro ladění se osvědčila následující makra, která posílají svůj argument na novou řádku obrazovky:

`\wri{reg}` **Výpis obsahu registru** **reg → screen**

Jde o velice univerzální makro. Lze ho použít pro výstup jak obecného čítače, tak i délkového registru definovaného dříve příkazem `\newlength{jméno}`, který obsahuje číselnou informaci o délce v bodech s rozměrem `pt`. Jeho vyvolání má při překladu za následek výstup textu

```
*** jméno registru = hodnota
```

na obrazovku. Podle vystupujících hodnot lze usuzovat na správnost výsledků při ladění nebo na hloubku zpracování (např. u cyklů).

Příklad:

```
\newcounter{pi}%     Čítač pro Ludolfovo číslo
\Rset pi=\Rpi%       dosazení předdefinované konstanty
\wri{pi}%            -> *** pi = 31416
\Cnt \ncyklu={20}%
\wri\ncyklu%         -> *** \ncyklu = 20
\wri\baselineskip%   -> *** \baselineskip = 14.5pt
```

Poslední výstup je roztečí mezi řádkami dokumentu při stylu `12pt`, který byl uveden v hlavičce. Odtud je patrno, že tímto způsobem lze získávat informace i o nastavení sytémových registrů.

`\wric\cmd` **Výpis hodnoty povelu** **\cmd → screen**

Makro lze užít pouze pro povely `\cmd`, které obsahují znakovou informaci, nebo i skupinu povelů uzavřenou ve složených závorkách, která je schopna se okamžitě rozvinout v řetěz znaků. Výstup má na obrazovce tvar

```
*** \jméno povelu = hodnota povelu
```

Příklad:
```
\Cmd \vzor = {*\#*\#*}%
\wric\vzor%              *** \vzor = *#*#*
\wric\Rpi%               *** \Rpi = 3.141592654
\wric{\the\hsize}%       *** \the\hsize = 455.24408pt
```

Povel `\Rpi` je v balíku `rplot.sty` předdefinovaný. Poslední příklad výpisu ukazuje použití makra `\wric` na skupinu povelů, totiž na výpis standardního délkového registru `\hsize`, který obsahuje délku řádky právě zpracovávaného textu. Je jasné, že výhodnější by v tomto případě bylo použití makra `\wri\hsize`.

`\wris{řetěz}` **Výpis řetězu** **řetěz → screen**

Toto makro je výhodné pro výstup libovolné zprávy na obrazovku během překladu. Zpráva je zde chápána jako libovolná posloupnost znaků, jejichž původ se na rozdíl od `\wri` a `\wric` neidentifikuje názvem (protože řetěz žádný název nemá). Obecný tvar výstupu je

```
*** řetěz znaků po případném rozvoji
```

Odtud je patrno, že případné povely, které mohl původní řetěz obsahovat, se nejdříve rozvinou a vystoupí až výsledná posloupnost znaků.

Příklad:
```
\wris{číslo pi = \Rpi}%       *** číslo pi = 3.141592654
\wris{*******}%               pro odělování skupin výpisů
\wris{u = \then, v = \thev}%  více výstupů na 1 řádce
```

Z příkladů je zřejmé, že zprávou nemusí být pouhý text, ale i hodnotové informace, či optické oddělovače řádek na obrazovce. Povel posledního typu je zvlášť významný, protože na rozdíl od `\wri` a `\wric`, které vydávají vždy jen **jednu** hodnotu na jednu řádku obrazovky, `\wris` umožňuje výstup několika i nehomogenních informací na řádku, čímž zpřehledňuje výstup. To je však vykoupeno víceprací uživatele při psaní povelu.

5.3 Čtení seznamu ze souboru

Části dokumentu umí TEX i LATEX zavést ze samostatných souborů pomocí příkazu `\input jméno_souboru`. Toho se běžně využívá při tvorbě rozsáhlých dokumentů. Vstupní posloupnost znaků se analyzuje, a pokud neobsahuje příkazy, posílá se do výstupního dokumentu. Příkazy ze vstupního souboru se vykonávají a opět jejich terminální výsledné sekvence se posílají do výstupního dokumentu.

Jiná situace nastává, chce-li uživatel LaTeXu přečíst data. Situace je horší proto, že data se teprve budou zpracovávat a nesmějí být zaslána do výsledného dokumentu, jako je tomu u příkazu `\input`. Proto byla sestavena makra, která umožňují data číst ze vstupních souborů a z požadovaných vytvářet seznamy, které lze potom analyzovat a výsledky nakonec použít k tvorbě dokumentu.

`\loadgf \cmd=soubor` **Čti obecný soubor** **\cmd ← soubor**

Makro otevře deklarovaný `soubor` pro čtení. Soubor musí být znakový a musí obsahovat seznam položek navzájem oddělených čárkami. Soubor se čte po textových řádkách (ke znaku nové řádky) a každá nově načtená řádka se připojí na konec uživatelem požadovaného povelu `\cmd`, který se tímto příkazem i vytvoří. Uživatel ho tedy nemusí předem deklarovat. Po přečtení všech řádek textu se vstupní soubor uzavře. Implicitní koncovkou jména souboru je `.tex`. V tom případě lze toto rozšíření jména vynechat. Je-li rozšíření jiné, musí se uvést plné jméno souboru, případně i s cestou, pokud soubor leží v jiném adresáři než v aktuálním.

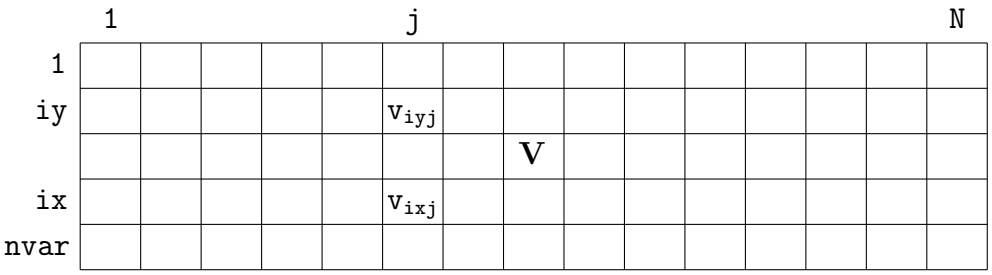
Příklad:
```
\loadgf \matrix=data.asc% Vytvoření příkazu \matrix s daty
\loadgf \xy=funkce.dat%   Vytvoření příkazu \xy
```

`\getxy \xy=\matx(ix,iy)` **Výběr vektorů x, y** **povel ← \matx(ix, iy)**

Mohlo by se zdát podivné, proč toto makro, které nekomunikuje s vnějším světem, a tedy nezavádí žádná data, se objevuje v této kapitole. Důvodem pro to je skutečnost, že je obvyklým následovníkem předcházejícího makra `\loadgf`. Vstupní soubor totiž může obsahovat souřadnice více závislostí, z nichž v daném okamžiku chce uživatel pracovat pouze s některými. Makro `\getxy` je navrženo právě pro výběr souřadnic bodů jedné závislosti `y(x)` z matice. Makro vybírá dvojice hodnot souřadnic x_j = $v_{ix,j}$ a y_j = $v_{iy,j}$ z povelu `\matx` a vytváří z nich nový povel `\xy`. Povel `\matx` obsahuje seznam položek oddělených čárkami. Struktura povelu `\matx`, který může být vytvořen příkazem `\loadgf`, je následující:

- `nvar` celé počet vektorů proměnných (procesů) v matici dat **V** (řádek)
- `N` celé počet bodů jednoho procesu (sloupců **V**); pokud není počet znám předem, stačí uvést `0`.
- `matice` real `nvar*N` hodnot proměnných uspořádaných po skupinách, v j-té skupině (sloupci **V**) o hodnotách v_{1j}, v_{2j}, ... , $v_{nvar,j}$, pro `j = 1, 2,... N`

Z uvedeného je zřejmé, že jde o matici, která se čte po sloupcích, v níž řádky jsou tvořeny hodnotami jednotlivých proměnných. Zbylé argumenty makra, `ix` a `iy`, představují indexy řádek, které jsou obsazeny hodnotami nezávisle proměnné resp. závisle proměnné. Pro oba indexy platí, že musí být z intervalu $\langle 1, \texttt{nvar} \rangle$. Nevadí, je-li `ix >iy`. Význam jednotlivých proměnných je dobře patrný i ze schematu matice :



Příklad:

```
\loadgf \meas=vibr.dat%          čtení souboru vibr.dat
\Ikbd\rpm%                       čti index řádky otáček
\Ikbd\y%                         čti index řádky vibrací
\Ikbd\nv%                        počet řádek matice
\Cmd \meas={\nv,0,\meas}%        doplň \nv a 0 za N
\getxy \rpmy=\meas(\rpm,\y)%
```

Těchto šest příkazů umožnilo z naměřeného souboru vektorů dat otáček stroje a vibrací vybrat příslušné vektory z matice, která může obsahovat ještě další vektory vibrací (z jiných měřicích míst), a vytvořit z nich povel `\rpmy` obsahující seznam souřadnic bodů pro diagram závislosti `{otáčky - vibrace}`. Za povšimnutí snad stojí, že

- vstupní soubor byl pouze „čistá“ matice dat neobsahující informaci o počtech proměnných `nvar` ani pozorování `N`. Bylo proto zapotřebí tyto dvě informace, nutné pro funkci makra `\getxy`, doplnit.
- To se stalo v 5. řádce příkladu, kde za počet proměnných byla dosazena hodnota před tím přečtená operativně z klávesnice. Za počet bodů `N`, který bez předběžné informace neznáme, byla dosazena nula. Zatímco `nvar` je krokem ve výběru prvků matice a jeho znalost je tedy podstatná, počet pozorování není nutno znát, protože se z matice vybírají prvky až do jejího vyčerpání.
- Pokud by se za počet proměnných zadala hodnota přesně `n`-násobná, než je ve skutečnosti, bude se z matice vybírat každý `n`-tý sloupec, čímž dojde k efektivnímu zředění bodů. To použijeme v případě velice hustého „vzorkování“ jevů.
- Pokud by vstupní soubor obsahoval pouze čistou matici skládající se jen z dvojic hodnot nezávisle proměnné a závisle proměnné, a to právě v tomto pořadí, není vůbec třeba vyvolávat toto makro a stačí s danou maticí dat přímo vstoupit do dalšího zpracování.

5.3.1 Práce s položkami seznamu

V řadě případů nepotřebuje uživatel pracovat s celým souborem, ale s jeho částmi. To byl již případ výběru určitých vektorů dat z matice, o kterém bylo pojednáno v předcházejícím odstavci. Někdy je ovšem zapotřebí pracovat dokonce i s jednotlivými položkami přečteného seznamu. K tomu slouží dále uvedená makra.

Pracovat se zde bude se seznamy, které budou zásadně uzavírány do kulatých závorek. Makra použijeme v případech, že budeme potřebovat jakkoliv pracovat s položkami. Jejich pořadí je dáno indexem, který roste po jedničce ve sloupcích řazených za sebou.

`\nextitem \cmd=(\seznam)` **Odtržení první položky**

Makrem lze oddělit ze seznamu první položku do proměnné `\cmd` a současně zkrátit o ni původní `\seznam`. Parametry makra jsou:

`\cmd` Jde o povel, který se makrem nadefinuje a naplní první položkou z povelu `\seznam`. Položky seznamu musí být navzájem odděleny čárkami.

`\seznam` Seznam v tomto makru **musí** mít tvar příkazu. Počáteční seznam se totiž oddělením první položky do povelu `\cmd` o ni zkrátí a takto upravený se vrátí do povelu `\seznam`. Pokud by seznam nebyl povelem, bude hlášena chyba, protože výsledný seznam nebude kam poslat.

Příklad:
```
\Cmd \xyz={3,4, 10,20,30, 40,50,60, 70,80,90, 100,110,120}%
\getxy \xy={\xyz}(1,3)
\wric\xy
```
s výstupem na obrazovku
```
*** Matrix type (3,4)
** \xy = 10,30, 40,60, 70,90, 100,120
```

`\getitem \cmd=(seznam){n}` **Okopírování položky**

Makro má následující parametry:

`\cmd` Povel, do kterého bude okopírována `n`-tá položka seznamu;

`seznam` seznam libovolných položek navzájem oddělených čárkami. Na výstupu z makra bude seznam nezměněn;

`n` index (pořadí) položky, která se okopíruje do povelu `\cmd`.

Následující příkazy se použily na stejná data, která byla uvedena v příkladu u příkazu `\nextitem`.

Příklad:
```
\Ikbd{n}%                zadej pořadí (např. 7)
\getitem \w=(\xyz){n}%   vybere např. 7. položku do \w
\wric{\w}%               -> např. *** \w = 50
```

`\putitem new=(seznam){n}` **Náhrada položky**

Makro nahradí `n`-tou položku seznamu položkou `new`. Parametry makra jsou:

`new` nová hodnota, která se dosadí na `n`-tou pozici v seznamu položek a přepíše starou položku. Veličinou `new` může být `{číslo}`, `{text}`, `\thejméno_čítače` nebo `\povel`, obsahující znaky (číslo nebo text);

`seznam` běžný seznam položek oddělených čárkami;

`n` pořadí (index) položky, která bude nahražena. Může mít formu celého čísla, jména obecného čítače obsahujícího hodnotu indexu nebo celočíselného povelu.

Příklad:
```
\Cmd \xyz={10,20,30, 40,50,60, 70,80,90}
\Cmd \cmd={2.3}
\Set{\n}{5}
\putitem {999}=(\xyz){\n} \wric\xyz
\putitem {abc}=(\xyz)\n \wric\xyz
\putitem \cmd=(\xyz)\n \wric\xyz
\putitem \the\n=(\xyz)\n \wric\xyz
```

s výstupem na obrazovku

```
*** 10,20,30, 40,999,60, 70,80,90
*** 10,20,30, 40,abc,60, 70,80,90
*** 10,20,30, 40,2.3,60, 70,80,90
*** 10,20,30, 40,5,60, 70,80,90
```

`\skipitems {low}(\seznam){high}` **Vynechávání položek**

Makro se užije vždy, když se na začátku a na konci seznamu mají jisté položky vynechat. Stává se to dosti často u měření, kdy zajímavý interval dat leží někde uprostřed seznamu, kdežto začátek a konec měření jsou nadbytečné. Parametry makra jsou:

`low` kladný index, do kterého včetně budou všechny položky vynechány;

`\seznam` běžný seznam položek oddělených čárkami, v němž je uložen jak vstupní, tak i výsledný seznam. Z tohoto důvodu **musí** být `\seznam` povelem!

`high` kladný index, od kterého včetně budou všechny položky až do konce původního seznamu vynechány. Při tom musí platit, že `low` $\leq$ `high`.

Veličiny `low` a `high` jsou celočíselnými konstantami, tedy celými čísly, celočíselnými povely nebo jmény čítačů.

Příklad:
```
\Cmd \xyz={3,4, 10,20,30, 40,50,60, 70,80,90, 100,110,120}
\skipitems{2}(\xyz){12}
\wric\xyz
```

s výsledkem na obrazovce:

```
*** \xyz = 10,20,30, 40,50,60, 70,80,90
```

`\getmn{m}{n}=(seznam)` **Zjisti typ matice**

Jde o makro, které lze aplikovat na seznamy získané přečtením souborů určených pro makro `\getxy`. Makro zjistí ze seznamu hodnotu prvních dvou položek, totiž počtu řádek `m` a počtu sloupců `n` matice, která vyplňuje zbytek seznamu. Vstupní seznam zůstane zachován beze změny. Parametry makra jsou:

`m`, `n` jména čítačů pro uložení počtu řádek rep. sloupců matice, která v seznamu následuje;

`seznam` běžný seznam položek oddělených čárkami, v němž první dvě položky udávají typ matice, která následuje;

Příklad:

```
\newcounter{m}
\newcounter{n}
\Cmd \xyz={3,4, 10,20,30, 40,50,60, 70,80,90, 100,110,120}
\getmn{m}{n}=\xyz
\wris{m=\them, n=\then}
```

s výsledkem na obrazovce:

```
*** m=3, n=4
```

5.3.2 Práce s řetězy

Zdánlivě ani tento odstavec nepatří do této kapitoly, protože dále uvedená makra také nečtou nebo nezapisují data. Přesto jsou zde, protože mohou být užita k analýze právě přečtených předem neznámých řetězů znaků. K jinému účelu patrně nebudou sloužit, protože by bylo zbytečné analýzovat známý řetěz. Zčásti se podobají makrům pro práci s položkami seznamů a proto zde uvedeme jen jejich krátký přehled:

`\nextchar \cmd=(\řetěz)` **První znak řetězu**

Makro uloží do povelu `\cmd` první znak řetězu, který zároveň o tento znak zkrátí. Řetěz **musí** být uložen v povelu, který je zde nazván `\řetěz`. Uživatel si však jeho jméno může zvolit libovolně. Postupným vyvoláváním tohoto příkazu za neustálého zkracování řetězu v povelu `\řetěz` se dostaneme k jeho libovolnému znaku.

Příklad:

```
\Cmd \abc={abcdefghijklmnopqrstuvwxyz}
\nextchar \ch=(\abc)
\wric\ch
\wric\abc
```

s výstupem na obrazovku:

```
*** \ch = a
*** \abc = bcdefghijklmnopqrstuvwxyz
```

`\getchar \cmd=(řetěz){n}` **Okopírování znaku**

Po vyvolání makra bude povel `\cmd` obsahovat `n`-tý znak řetězu. Řetěz sám zůstane vyvoláním makra nezměněn. Nemusí proto být (na rozdíl od příkazu `\nextchar`) obsahem povelu. Parametrem `n` může být libovolná celočíselná konstanta.

Příklad:

```
\Cmd \abc={abcdefghijklmnopqrstuvwxyz}
\getchar \ch=(\abc){14}
\wric \ch
```

Na obrazovku vystoupil text:

```
*** \ch = n
```

`\findchar {znak}(řetěz){čítač}` **Hledání znaku**

Makrem se prohledává `řetěz` a každý jeho znak se srovnává s daným vzorem `znak`, který je libovolným znakem ASCII. V případě, že dojde ke shodě, vloží se do proměnné `čítač` celé číslo rovné pořadí nalezeného znaku od počátku řetězu. Neobsahuje-li řetěz hledaný znak, dosadí se do `čítače` záporně vzatá délka řetězu.

Příklad:
```
\newcounter{np}
\Cmd \real={123.456}%          reálné číslo
\findchar{.}(\real){np}
\wri{np}%                      *** np = 4
\Cmd \five={5}
\findchar \five(123.456){np}
\wri{np}%                      *** np = 6
```

Z příkladu je patrné, že hledaný znak může být i vložen do uživatelova povelu (zde `\five`). Parametr `čítač` je uživatelem předem deklarovaným obecným čítačem.

Kapitola 6

Grafika

6.1 Prostředí picture

Cílem tohoto odstavce je být referenčním materiálem pro uživatele, který si pouze upřesňuje jisté informace. Proto je popis velice stručný. Jen v případech, kdy vysvětlení chování příkazu je v originálním manuálu lit. [2] příliš stručné, je zde více rozvedeno i s příklady.

Prostředí `picture` předchází příkaz definice délkové jednotky, v jejíž reálných násobcích budou uváděny všechny délky (vzdálenosti, souřadnice):

`\unitlenght`*X*`rozměr` **Definice délkové jednotky**

Zde

X je reálné číslo udávající násobitele následující délky charakterizované parametrem `rozměr`

`rozměr`

dim		[sp]	[pt]	[in]	[mm]
1 sp	=	1	1/65536		
1 pt	=	65536	1	1/72,27	0,35146
1 in	=	4736287	72,27	1	25,4
1 mm	=	186468	2,84528	0,03937	1
1 pc (pica)	=	786432	12	0,166044	4,2175

Implicitní hodnota jednotkové délky `\unitlenght` v celém prostředí `picture` je `1pt`, což je, jak ukazuje tabulka, přibližně třetina milimetru. Kromě uvedených měr lze užít ještě další, které jsou zejména při sázení textu výhodnější, protože nejsou fixní, ale typově závislé. Jsou to dimenze odvozené z velikosti písma:

`1em` = šířka písmene M právě používaného typu písma (fontu)
`1ex` = výška písmene x právě používaného typu písma (fontu)

Pro velikost písma pica (`12pt`) jsou `1em = 11.75pt, 1ex = 5.17pt`.

Vlastní prostředí `picture` je uzavřeno stejně jako i jiná prostředí standardními „závorkami“ `\begin ... \end`. Zapisuje se ve tvaru:

`\begin{picture}(xdim,ydim)(xoff,yoff)` **Začátek prostředí picture**

`\end{picture}` **Konec prostředí picture**

`(xdim,ydim)` jsou reálné položky (bez rozměru) vyjadřující v jednotkách `\unitlength` šířku a výšku obdélníka vyhrazeného pro prostředí `picture`. Bod `(0,0)` je v levém dolním rohu této plochy

`(xoff,yoff)` je nepovinná dvojice bezrozměrných reálných konstant charakterizující souřadnice uživatelova bodu, do kterého chce umístit levý dolní roh kreslicí plochy deklarované dvojicí `(xdim,ydim)`.

Prostředí `picture` lze použít jak pro grafické práce, tak i pro textové výstupy, zejména chceme-li přesně dodržet předepsanou pozici textu, jako je tomu např. u doplňování předtištěných formulářů. Uveďme příklad prostředí `picture` s uživatelovým počátkem posunutým do středu vyhrazené plochy:

Příklad:
```
\unitlength1mm
\begin{picture}(160,50)(-80,-25)
  povolené příkazy
\end{picture}
```

V prostředí `picture` jsou povoleny **pouze** dvě skupiny příkazů, a to deklarace a umisťovací příkazy.

6.1.1 Deklarace

Jedná se o příkazy bez jakéhokoliv výstupu do cílového dokumentu. Jsou to příkazy, jimiž se nastavují parametry budoucího výstupu. Do standardních deklarací prostředí `picture` patří příkazy

`\linethickness{dim}` **Tloušťka čáry**

Zde `dim` je reálná konstanta s rozměrem, udávající požadovanou tloušťku **pouze** horizontálních nebo vertikálních čar. Tak např. `\linethickness{.5mm}` změní tloušťku uvedených čar z dosud nastavené na žádanou.

`\thinlines` **Tenké čáry**

Příkaz nastaví tloušťku dále kreslených libovolných (tedy i šikmých) úseček i kružnic na implicitní hodnotu `0.4pt = 0.14mm`.

`\thicklines` **Tlusté čáry**

Vyvolání příkazu způsobí, že následně kreslené obecné úsečky a kružnice budou mít tloušťku čáry rovnou `0.8pt = 0.28mm`.

Mezi standardní deklarace se počítá i příkaz

`\savebox{\jméno}(xdim,ydim)[pos]{text}` **Deklarace a uložení boxu**

Použité argumenty mají následující význam:

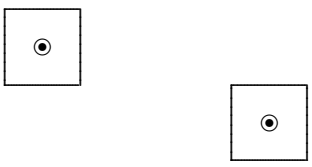
`\jméno`	Uživatelem zvolený název boxu – povel, který již dříve deklaroval příkazem `\newsavebox{\jméno}`
`(xdim,ydim)`	šířka a výška boxu jako bezrozměrné reálné konstanty – násobky `\unitlength`
`pos`	nepovinný parametr pro definici pozice `text`u, který je implicitně v boxu vycentrován
	`l = text` začíná na levém okraji boxu,
	`r = text` končí na pravém okraji boxu ,
`text`	libovolná posloupnost znaků, ve které mohou být i příkazy pro kreslení.

Při překladu se `text` rozvine (přeloží), ale na rozdíl od jiných boxů nevystoupí do dokumentu, ale uloží se přeložený do zvláštní paměti, která k tomuto účelu byla vyhražena již zmíněným příkazem `\newsavebox`. Kdykoliv potom chceme obsah boxu zařadit do dokumentu (obvykle vícekrát, na různých místech) použijeme příkaz – `\usebox{\jméno}`.

Příklad:

```
\newsavebox\ctverec
\unitlength1mm%
\begin{picture}(160,35)(-50,-5)%
 \savebox{\ctverec}(20,20)[]%
  {\unitlength1mm%
   \begin{picture}(20,20)(-10,-10)%
    \put(-5,-5){\square(10)}%
    \put(0,0){\circle{1}}%
   \end{picture}%
  }
 \put(0,0){\circle*{1}}
 \put(20,20){\circle*{1}}
 \put(20,20){\usebox\ctverec}
 \put(50,10){\circle*{1}}
 \put(50,10){\usebox\ctverec}
\end{picture}
```

Výsledek:



- počátek

V příkladu byly použity příkazy, které budou vysvětleny dále, a to:

`\put(x,y){obj}`	umístění objektu na kreslicí ploše v bodě `(x,y)`, kde `obj` byly:
`\circle{diam}`	nakreslení kružnice o průměru `diam`; u `\circle*` vyplněné;
`\square{a}`	nakreslení čtverce o straně `a` s levým dolním vrcholem v `(x,y)`.

Chceme-li se vyhnout problémům s užitím příkazu `\savebox` v prostředí `picture`, je zapotřebí:

- chápat celý box jako jedno písmeno, tedy s referenčním bodem v levém dolním rohu,
- v příkazu `\savebox` umístit opět prostředí `picture`, jehož (lokální) příkazy pro umisťování objektů zajistí požadované umístění v rámci boxu,
- v příkazu `\savebox` dávat hned za poslední znak v **každé** řádce `text`u znak %. Pokud tak neučiníme, může být následující objekt posunutý o případnou mezeru.

Z grafického výsledku příkladu je patrno, že dodržování těchto zásad vede k přesnému umisťování objektů. Kromě výše uvedených standardních deklarací lze do prostředí `picture` zařadit i nestandardní, jako jsou naplňování registrů a výpočty, případně i další příkazy, které nezpůsobí **žádný** výstup do cílového dokumentu. Za jejich použití, které nemusí být zcela bez rizika, si zodpovídá uživatel sám.

6.1.2 Příkazy pro umisťování objektů

Ve standardním LaTeXu jsou vytvořeny pouze dva příkazy pro umisťování objektů na kreslicí plochu dříve definovanou pro dané prostředí `picture`. Jsou to příkazy:

`\put(x,y){objekt}` **Jednorázové umísťování objektu**

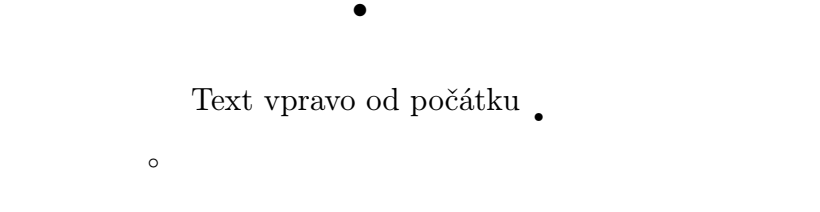
Zde parametry makra jsou

`(x,y)`	je definice bodu kreslicí plochy, v `\unitlength` do něhož bude umístěn referenční bod objektu. Symboly `x, y` zde zastupují souřadnice bodu jako reálné položky vyjádřené buď číselně, anebo jako povely s číselnou informací. Pozor! Položky bez teček jsou celá reálná čísla. Mohou tedy jimi být i neměřítkované hodnoty z čítačů ve tvaru `\value{čítač}` nebo `\thečítač`.
`objekt`	může být libovolný text, tedy i příkaz vyvolávající libovolný výstup do výsledného dokumentu. Tím může být i vyvolání nového prostředí `picture`.

Příklad:

```
\Cmd \x={21.3}\Cmd \y={12.1}%  x,y jako reálné povely
\Cnt \xc={-5}\Cnt \yc={-7}%    x,y jako Re celé Tčítače
\newcounter{x}\newcounter{y}%
\setn{x}{46}\setn{y}{-1}%      x,y jako Re celé Lčítače
\put(0,0){Text vpravo od počátku}%
\put(\x,\y){$\bullet$}%
\put(\xc,\yc){\circle{1}}%
\put(\value{x},\they){\circle*{1}}%
```

Výsledek:



`\multiput(x,y)(dx,dy){n}{objekt}` **Vícenásobné umístění objektu**

`(x,y)` dvojice reálných souřadnic v `\unitlength` pro umístění referenčního bodu prvního vynesení objektu,

`(dx,dy)` dvojice reálných přírůstků souřadnic při opakovaných vynášeních objektu v `\unitlength`. Objekty budou vynášeny se svými referenčními body o souřadnicích `(x+dx,y+dy)`, `(x+2dx,y+2dy)`, ...

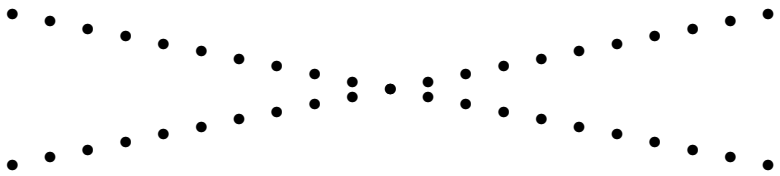
`n` počet vynášení objektů

`objekt` libovolný text

Vícenásobné vynášení objektů je analogické jednorázovému vynášení, pokud jde o zadávání souřadnic, přírůstků i objektů. I u tohoto příkazu může být objektem opět prostředí `picture`. V tomto případě je však výhodnější nadeklarovat nejdříve `\savebox` a v příkazu `\multiput` vyvolávat jako objekt příkaz `\usebox`, protože překlad boxu proběhne pouze jedenkrát a příkaz `\multiput` ho pak jen umisťuje.

Příklad:
```
\multiput(0,0)(5,1){21}{\circle*1.5}
\multiput(100,0)(-5,1){21}{\circle*1.5}
```

Výsledek:



6.1.3 Objekty v prostředí picture

Objekty, které v prostředí `picture` umisťujeme příkazy `\put` a `\multiput` na kreslicí plochu, jsou texty, boxy a čáry.

Texty a boxy

Texty samy vytvářejí obdélníkové plochy, které se nazývají boxy. Nejjednodušším boxem je samotné písmeno, složitějším slovo, skupina slov atd. Zatímco v běžném dokumentu se řádky zalamují automaticky, v prostředí `picture` tomu tak není. Vložený text vytvoří jedinou řádku, která může překračovat i šířku dokumentu. Proto musí uživatel pečlivě vážit délku textu:

`\makebox(xdim,ydim)[pos]{text}` **Vytvoření boxu**

`\framebox(xdim,ydim)[pos]{text}` **Box s rámečkem**

`\dashbox(dx,dy)(xdim,ydim)[pos]{text}` **Box s čárkovaným rámečkem**

V těchto příkazech jsou:

`(xdim,ydim)` šířka a výška boxu v násobcích `\unitlength`

`pos` nepovinná definice umístění objektu v boxu, a to: `l` na levou stranu, `r` pravou stranu boxu, `t` nahoře a `b` dole. Parametr `pos` je dán kombinací těchto písmen, přičemž implicitní umístění je ve středu boxu;

`text` libovolný text s případnými dalšími příkazy pro umístění objektů nebo dokonce i novým prostředím `picture`;

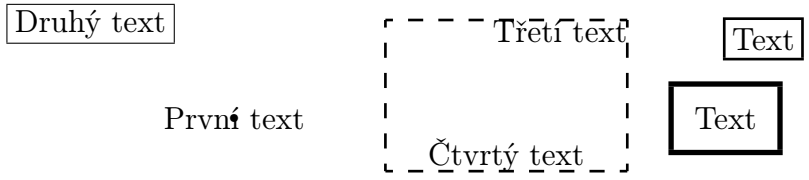
`dxy` délka čárek a mezer v horizontálním a vertikálním směru u čárkovaného rámečku. Šířka a výška rámečku `š` resp. (`v`) by měly být celým násobkem dvojnásobné délky čárek, tedy rovny `š=2m*dxy`, resp. `v=2n*dxy`, kde `m,n` jsou libovolná celá čísla – počty mezer.

Referenčním bodem boxů je stejně jako u písmen jejich levý dolní roh. Tloušťka čáry je volitelná. Velice užitečný je příkaz `\makebox(0,0){text}`, který centruje text kolem bodu určeného umisťovacím příkazem. Je třeba upozornit, že text v necentrální poloze se těsně přimyká k rámečku bez jakékoliv mezery. Text v centrální poloze je od stran odlehlý o vzdálenosti, které mu uživatel ponechal deklarací šířky a výšky. Chceme-li se tomu vyhnout, můžeme použít textový příkaz `\fbox` a nadefinovat mu odlehlost rámečku od textu nastavením hodnoty délkového registru `\fboxsep`. Rovněž tloušťku čáry tohoto rámečku lze změnit z implicitní hodnoty změnou obsahu délkového registru `\fboxrule`. Obě změny se provedou příkazem `\setlength`.

Příklad:

```
\put(0,0){\circle*{1}}
\put(0,0){\makebox(0,0){První text}}
\put(-30,10){\framebox(22,5){Druhý text}}
\linethickness{1pt}
\put(20,-7){\dashbox{2}(32,20)[tr]{Třetí text}}
\put(20,-7){\makebox(32,20)[b]{Čtvrtý text}}
\put(65,8){\framebox(10,5){Text}}
\setlength{\fboxsep}{1.5ex}
\setlength{\fboxrule}{2pt}
\put(65,0){\makebox(0,0){\fbox{Text}}}
```

Výsledek:



Poznámka: Centrování prvního textu je patrné podle silné tečky v písmeni "í", kde leží střed výšky a střed šířky textu (tedy ne těžiště) a počátek.

Mezi boxy by bylo možno zařadit ještě následující dva příkazy:

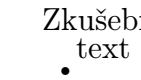
`\shortstack[pos]{sloupec}` **Sloupec textu**

Jde v podstatě o zkrácenou a mírně modifikovanou verzi prostředí `tabular`. Modifikace spočívá v absenci mezer po okrajích textu a v zmenšení rozteče řádek. Příkaz vytvoří box, jehož referenční bod je opět v jeho levém dolním rohu. Pokud tedy

chceme obsah příkazu `\shortstack` centrovat kolem bodu uvedeného v příkazu `\put`, musíme ho vložit do příkazu `\makebox(0,0)`. Příkaz pro nakreslení kroužku v následujícím příkladu je uveden pouze pro identifikaci počátku.

Příklad:
```
\put(5,5){\makebox(0,0)%
  {\shortstack[c]{Zkušební\\text}}}
\put(0,0){\circle*{1}}
```

Výsledek:

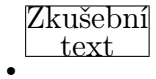


`\frame{objekt prostředí picture}` **Orámování objektu**

Příkaz nakreslí kolem objektu, který je uveden jako argument, rámeček s referenčním bodem v levém dolním rohu objektu.

Příklad:
```
\put(0,0){\circle*{1}}
\put(2,2){\frame{\shortstack{Zkušební\\text}}}
```

Výsledek:



Čáry

Sázecí systém LaTeX dává uživatelovi k dispozici relativně chudou množinu příkazů pro kreslení čar. Čáry jsou grafické objekty, které umisťujeme na kreslicí plochu pomocí umisťovacích příkazů `\put`, příp. `\multiput`. V základní verzi LaTeXu existuje navíc řada omezení jednak na délku, jednak na směr úseček.

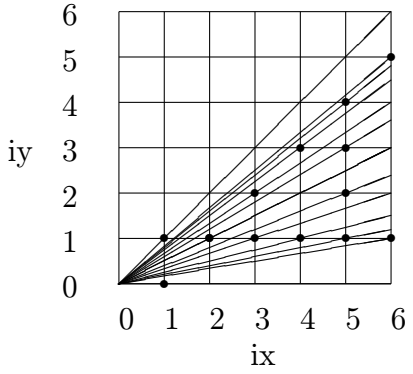
`\line(ix,iy){xydélka}` **Úsečka**

Makro nakreslí úsečku z bodu definovaného umisťovacím příkazem ve směru daném parametry `ix` a `iy`. Význam parametrů je následující:

`(ix,iy)` je dvojice celých navzájem nesoudělných čísel z intervalu $\langle -6, 6\rangle$, jejichž poměr `iy/ix` je tangentou sklonu úsečky,

`xydélka` je reálné číslo. Pro `ix`$\neq$`0` je průmětem skutečné délky do horizontální osy `x`. Pouze pro `ix=0` je `xydélka` rovna přímo délce (vertikální) úsečky.

Výše řečená omezení je zapotřebí ještě rozšířit o podmínku, že skutečná délka šikmé úsečky musí být alespoň `10pt`. Pokud je úsečka kratší, vůbec se nenakreslí! Ale úsečky ve směru os se vykreslí již od délky `1pt`. Z vedlejšího obrázku nakresleného pouze pro polovinu prvního kvadrantu vyplývá, že množina možných směrů úseček je velice řídká. Jen tučně vyznačené body vyjadřují povolené kombinace čísel `ix,iy`. Ostatní možné směry lze získat pomocí symetrie podle os a vedlejší diagonály změnami znamének čísel `ix,iy`.



Příklad:

```
\unitlength2pt
\begin{picture}(100,30)
  \put(0,0){\line(1,1){10}} \x=5%
  \put(0,0){\line(1,0){10}} \put(0,0){\line(0,1){10}}
  \Ifor \y=(1:1:4){\put(0,0){\line(\x,\y){10}}%
                   \put(0,0){\line(\y,\x){10}}}%
\end{picture}
```

`\vector(ix,iy){xydélka}` **Orientovaná úsečka**

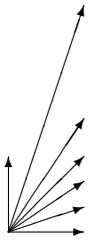
Makro kreslí orientovanou úsečku zakončenou šipkou. Je velmi podobné makru `\line`, s nímž má i stejné parametry. Jen omezení na jejich hodnoty jsou přísnější:

`(ix,iy)` je dvojice celých navzájem nesoudělných čísel z intervalu $\langle -4, 4\rangle$, jejichž poměr `iy/ix` je tangentou sklonu orientované úsečky;

`xydélka` je reálné číslo udávající délku v `\unitlength`. Pro `ix`$\neq$`0` jde o průmět skutečné délky do horizontální osy `x` . Pouze pro `ix=0` je `xydélka` rovna přímo délce vertikálně orientované úsečky.

Příklad:

```
\unitlength1mm
\begin{picture}(20,50)(-10,-5)
  \put(0,0){\vector(1,1){10}}
  \x=3%
  \Ifor \y=(0:1:2){\put(0,0){\vector(\x,\y){10}}%
                   \put(0,0){\vector(\y,\x){10}}}%
\end{picture}
```



`\circle{diam}` **Kružnice**

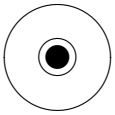
Makro lze použít pro nakreslení kružnice o průměru z intervalu $\langle 0, 40\rangle$`pt` zadaném reálným číslem `diam` v násobcích `\unitlength`.

`\circle*{diam}` **Plný kroužek**

Makrem lze kreslit plné kroužky o průměru do 15pt definovaným parametrem `diam` v násobcích `\unitlength`.

Příklady:

```
\unitlength1mm
\begin{picture}(15,30)(-7,-20)
  \put(0,0){\circle*{3}}
  \put(0,0){\circle{5}}
  \put(0,0){\circle{14}}%  největší možný
\end{picture}
```



`\oval(xdim,ydim)[část]` **Ovál**

Makrem lze kreslit ovály a jejich části. Pod pojmem ovál se zde chápe obdélníková oblast s rohy zaoblenými čtvrtkružnicemi. Poloměry zaoblení jsou automaticky upravovány tak, aby obdélník byl uzavřen polokružnicí, jejíž průměr nepřesáhne cca 14 mm. Pokud menší z obou rozměrů `(xdim,ydim)*\unitlength` by tuto mez přesáhl, dojde k již zmíněnému zaoblování rohů. Parametry makra jsou:

`(xdim,ydim)` vnější rozměry oválu (šířka a výška) v násobcích `\unitlength`.

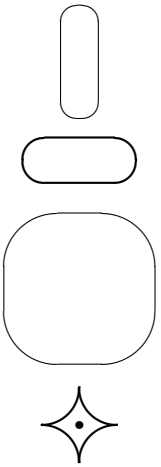
`[část]` nepovinný parametr složený z jednoho nebo dvou písmen, určujících, která z částí oválu má být nakreslena. Těmito písmeny jsou:

`l` (left) levá část oválu,
`r` (right) pravá část oválu,
`t` (top) horní část oválu,
`b` (bottom) dolní část oválu.

Díky možnosti vybrat pouze část oválu, např. pomocí kombinace `[tr]` horní pravou část, lze tímto způsobem kreslit i čtvrtkružnice. Jejich poloměr `r` se potom řídí parametry `xdim=ydim`$\leq$`14mm`.

Příklady:

```
\unitlength1mm
\begin{picture}(20,45)(-10,-10)
  \put(0,48){\oval(5,15)}
  \put(0,18){\oval(20,20)}
  \thicklines
  \put(0,35){\oval(15,6)}
  \put(0,0){\circle*{1}}
  \put(-5,-5){\oval(10,10)[tr]}
  \put(-5,5){\oval(10,10)[br]}
  \put(5,5){\oval(10,10)[bl]}
  \put(5,-5){\oval(10,10)[tl]}
\end{picture}
```



6.2 Kreslení křivek pomocí curves.sty

Z popisu prostředí `picture` je zřejmé, že jeho prostředky umožňují jen kreslení jednoduchých obrázků složených pouze z úseček, kroužků a kruhových oblouků o vrcholovém úhlu 90°. Proto se během času objevovaly různé doplňky k existující množině objektů. Z nich nejvýznamnější byl patrně soubor `bezier.sty`, který umožnil vykreslování hladkých čar množinou bodů ležících těsně vedle sebe, které se v průběhu kompilace počítaly jako body kvadratické Bézierovy křivky. Pro ni musel uživatel nadefinovat úseky křivky a pro každý z nich tři body. Střední bod byl průsečíkem tečen ke křivce procházejících prvním a posledním bodem z trojice, které tak byly body dotyku Bézierovy křivky. Dále musel definovat parametry, jimiž řídil hustotu počítaných bodů a tím i vzhled křivky.

Použití tohoto stylu kladlo dosti velké nároky na uživatele. Proto vznik nových stylů, které autor Ian Maclaine-cross nazval `curves.sty` a `curvesls.sty` (viz **??**), a které odstranily nepříjemnou práci s volnými parametry a zefektivnily výpočet, byl vřele uvítán. Autor však nezůstal pouze u vylepšení makra `\bezier`, ale na nové verzi tohoto makra založil celý balík maker, které soustředil do výše uvedených souborů. Oba se navzájem

liší především v elementárních příkazech pro vynášení úseček. Zatínco `curves.sty` vynáší části křivek klasicky, `curvesls.sty` používá zrychleného vynášení pomocí TEXovských maker `\special`. Tím významně šetří paměť i čas počítače. Pro běžného uživatele se oba styly navzájem formálně neliší. Popisujeme-li jeden z nich, popisujeme současně i druhý. Všechny vynášené spojité křivky jsou mezi dvěma body nahražovány kvadratickými parabolami. To se týká i kružnic a kruhových oblouků. Chyba náhrady ve vztahu k poloměru je podle autora `curves.sty` v setinách procenta.

Myšlenka na zefektivnění činnosti makra pro výpočet a vynášení bodů Bézierovy křivky je založena na schopnosti lidského oka vnímat body ležící blízko sebe a částečně se překrývající jako spojitou čáru. Míru překrytí počítá makro samo a optimalizuje ji s ohledem na tloušťku čáry. Tím dosahuje přijatelný výsledek v minimálním čase. Podrobný popis obou stylů je uveden v manuálu lit. [12]. Zde bude uveden jen stručný popis maker. Je zřejmé, že pro použití těchto stylů ve svém dokumentu musí uživatel deklarovat v `documentstyle` jeden z nich, např.

```
\documentstyle[...,curvesls,...]{report}
```

Při tom se předpokládá, že soubor maker je umístěn v adresáři stylů, který bude při kompilaci prohledáván. Při použití stylu `rplot` je tato deklarace zbytečná.

Soubory maker umožňují:

- používat nastavitelnou šířku čáry od 0.5pt do 15pt (0.17mm až 5mm).,
- kreslit křivky se spojitým průběhem,
- kreslit křivky procházející libovolným počtem bodů pomocí makra `\curve`,
- řídit sklon křivek na koncích vyvoláním makra `\tagcurve`,
- kreslit uzavřené hladké křivky makrem `\closecurve`,
- kreslit libovolně velké kružnice a oblouky makry `\bigcircle` a `\arc`,
- měřítkovat nezávisle na sobě osu `x` a `y` včetně vazeb mezi nimi,
- pomocí měřítkování vytvářet z kružnic elipsy, případně rotovat s objekty,
- vynášet čáry s libovolným vzorem (čárkované, čerchované, ...),
- kreslit symboly bodů bez interference s čarou.

Všechna makra souboru se používají ve standardním prostředí `picture`. Proto makra i tohoto souboru rozdělíme do stejných skupin.

6.2.1 Deklarace

Deklaracemi se nastavují hodnoty parametrů, které se uchovávají registrech a povelech. Zásadní význam pro informovanost při překladu vynášení křivek mají makra:

`\curvewarntrue` **Povolení výstupu varování**

`\curvewarnfalse` **Zákaz výstupu varování**

V obou případech jde o povely, které slouží jako přepínače stavů, v nichž se dává (nebo nedává) uživatelovi během překladu zpráva na obrazovku, že daný úsek náhrady Bézierovou křivkou není částí paraboly, ale je úsečkou.

`\renewcommand{povel}[narg]definice` **Předefinování povelu**

Jde o standardní příkaz LaTeXu, který se zde používá k dosazení nových hodnot povelů. Pro styl `curves` bude jeho varianta jednodušší, totiž

```
\renewcommand{povel}{reálné číslo}
```

Těmito povely jsou:

`\diskpitchstretch` původně nastavený na hodnotu `1`. Změnou např. na `5` se zmenší hustota bodů Bézierových křivek na pětinu a tím se zkrátí výpočetní čas také na pětinu, avšak za cenu zvětšení hrubosti čar. Naproti tomu nastavení na hodnotu 0.5 zvýší hustotu bodů na dvojnásobek, ovšem za cenu zhruba dvojnásobné doby výpočtu a vynášení čáry.

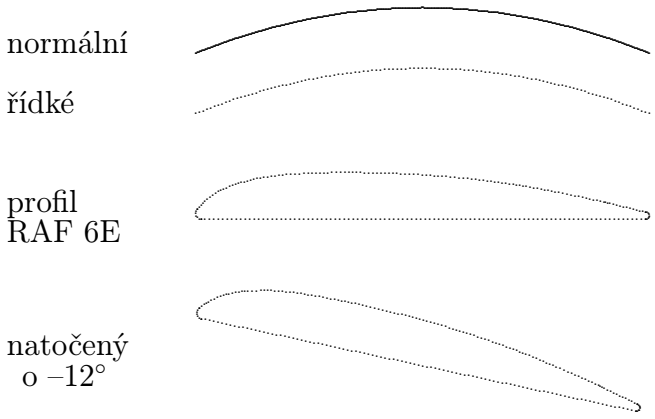
`\xscale` `\yscalex` `\xscaley` `\yscale` měřítka, reálná čísla, nastavená implicitně na hodnoty `\xscale=\yscale=1` a `\xscaley=\yscalex=0`, kterými lze měnit délkové jednotky pro jednotlivé směry os a pro vazby mezi nimi. Pokud dosadíme za měřítka hodnoty

```
\xscale=c,   \yscalex=s,
\xscaley=-s, \xscale=c,
```

kde `s` a `c` jsou reálná čísla `s=sin(alfa))` a `c=cos(alfa)`, pootočí se objekt vynášený dále uvedenými novými povely pro umisťování objektů kolem počátku o úhel `alfa`.

Příklady:

```
\renewcommand{\diskpitchstretch}{5}%  5-násobné zhrubnutí čáry
\renewcommand{xscale}{0.9781}%        rotace o -12 stupňů
\renewcommand{yscalex}{0.2079}%             včetně
\renewcommand{xscaley}{-0.2079}%            včetně
\renewcommand{yscale}{0.9781}%              včetně
```



`\setlength{\jméno}{reálné číslo s rozměrem}` **Nastavení délky**

Jde opět o standardní příkaz LaTeXu, který v rámci použití souboru `\curves.sty` slouží pro nastavování délky `\overhang`, jíž se řídí nahrazování vzoru zadaným symbolem. Blíže viz popis příkazu `\curve`.

Příklad: `\setlength{\overhang}{3mm}`

`\settowidth{\jméno}{text}` **Zjištění délky textu**

Další z příkazů LaTeXu se užívá k dosazování délky textu o šířce maximálně jedné řádky do délkového registru s uživatelem vybraným jménem `\jméno`. Při užívání souboru `curves.sty` se s ním setkáváme při zjišťování velikosti symbolů a nebo nastavování délky přerušení čáry.

Příklad:
```
\settowidth{\csdiameter}{*}
\settowidth{\csdiameter}{00}
```

`\curvesymbol{znak_nebo_symbol}` **Nastavení symbolu a \csdiameter**

Vyvoláním makra dosáhneme uložení libovolného znaku ASCII nebo grafického symbolu pro vynášení bodů diagramu a současně i zapamatování jeho velikosti v délkovém registru `\csdiameter`. Obě informace jsou později potřebné pro přerušování čáry v místech těchto bodů.

Příklady:

```
\curvesymbol{$\circ$}%       nebo jiný:
\curvesymbol{$\triangle$}%   nebo jiný:
\curvesymbol{$*$}%           anebo:
\curvesymbol{@}%             atd.
```

`\curvedashes[jedn.délka]{prázdná délka,seznam}` **Vzhled čáry**

Makro se použije vždy, když chceme kreslit přerušovanou čáru. S jeho pomocí nadefinuje uživatel vzor přerušování. Parametry makra jsou:

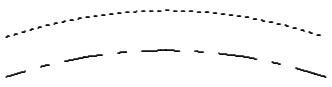
`jedn.délka` volitelná kladná reálná délka (např. `2.5mm`), jíž lze nastavit jednotku délky pro zbylé parametry nezávisle na `\unitlength`. Ta je však jeho implicitní hodnotou.

`prázdná délka` je kladné reálné číslo, vyjadřující v násobcích `jedn.délky` délku úseku křivky na začátku čáry, který nebude vykreslován.

`seznam` je posloupnost kladných reálných čísel oddělených navzájem čárkami, která mají význam délek čárek a mezer mezi nimi v jednotkách `jedn.délka`. Seznamem se definuje vzor přerušování čáry.

Příklady:

```
\curvedashes[1pt]{0, .5,2,1,2,.5}%
\curvedashes[1mm]{0, 2.5,2,1,2,2.5}%
```



Nakreslené čáry nejsou výsledkem vyvolání makra `\curvedashes`. Zde jsou uvedeny jen proto, aby ukázaly vliv makra na později vykreslenou křivku. Je zapotřebí zde upozornit, že role makra `\curvedashes` je pro kreslení křivek daleko významnější, než by se těmito jednoduchými příklady mohla jevit. Podrobněji však bude rozebrána později. Přerušování čáry a vynechávání prostoru na symboly je však zatím spojeno s menší chybou při vynášení.

6.2.2 Umisťovací příkaz

Soubor `curves.sty` přidává k příkazům `\put` a `\multiput` další příkaz, jímž lze umisťovat objekty na kreslící ploše. Je jím příkaz

`\scaleput(x,y){objekt}` **Oměřítkované umísťováníní objektu**

Význam jednotlivých parametrů je stejný, jako je u příkazu `\put` z prostředí `picture`:

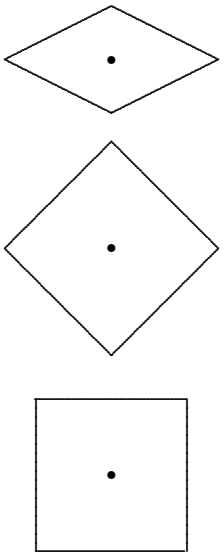
`(x,y)` je definice bodu kreslící plochy, do něhož bude umístěn referenční bod objektu. Symboly `x, y` zde zastupují souřadnice bodu jako reálné položky vyjádřené buď číselně, anebo jako povely s číselnou informací. Pozor! Položky bez teček jsou rovněž celá reálná čísla. Mohou tedy jimi být i neměřítkované hodnoty z čítačů.

`objekt` může být libovolný text, tedy i příkaz vyvolávající libovolný výstup do výsledného dokumentu. Tím může být i vyvolání nového prostředí `picture` nebo dále uvedených grafických objektů.

Na rozdíl od příkazu `\put`, může příkaz `\scaleput` oměřítkovávat objekt a tak ho přetvářet pomocí lineárních transformací. Odtud plyne, že z kružnice lze vytvořit elipsu a objekty otáčet kolem počátku. To umožňuje vytvářet i prostorové pohledy. Je velmi významné, že příkazy `\put` a `\scaleput` jsou na sobě **nezávislé** a tak je možno je kombinovat. Např. příkazem `\put` lze vybrat referenční bod na kreslicí ploše a jako jeho objekt použít příkaz `\scaleput` se skutečným grafickým objektem.

Příklad:

```
\newcommand\csquare[1]%
%           ~~~~~~~~~~~
 {\Rset w={#1}\divn{w}{-2}%
  \Rget \ah=w%
  \scaleput(\ah,\ah){\square(#1)}%
 }
 \unitlength1mm
 \begin{picture}(30,70)(-15,-15)
   \put(0,0){\circle*{1}}
   \put(0,0){\csquare{20}}%
   \scalerot{45}%          viz další popis!
   \put(0,30){\circle*{1}}
   \put(0,30){\scaleput(0,0){\csquare{20}}}
   \setscales(0.707,0.3535, -0.707,0.3535)
   \put(0,55){\circle*{1}}
   \put(0,55){\scaleput(0,0){\csquare{20}}}
 \end{picture}
```



6.2.3 Grafické objekty – vynášení křivek

Všechny grafické objekty v souboru `\curves.sty` jsou křivky. Všechny jsou vytvářeny stejným způsobem, totiž složením parabolických oblouků, a to takových, že procházejí danými body a dotýkají se tečen, které jsou rovnoběžné se sečnou spojující dva

body sousední s bodem dotyku. Tento výtvarný zákon nemusí vždy vést k uspokojujícím výsledkům, avšak jeho algoritmus je jednoduchý, a tudíž málo náročný na výpočetní čas. Celkem je k dispozici 6 maker pro vytváření křivek, které mohou být objekty jak pro všechny umisťovací příkazy, tj. pro příkazy `\put, \multiput` i `\scaleput`. Jsou to v pořadí dle potřebnosti:

`\curve[SC](seznam)` **Obecná křivka**

Makro má následující parametry:

`SC` nepovinné celé číslo, jehož `|SC|` vyjadřuje počet podúseků, na které bude každý úsek mezi dvěma body rozdělen

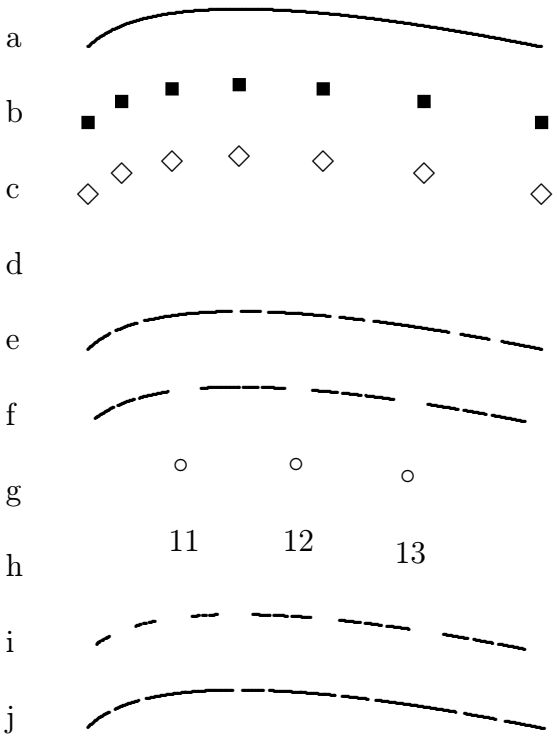
`(seznam)` posloupnost dvojic souřadnic jako reálných čísel bodů `x,y` oddělených mezi sebou čárkami. Seznamem může být i povel obsahující `seznam`. Pokud počet bodů bude roven 2, vynese se **úsečka**. V případě, že 3 body leží na jedné přímce, vynese se rovněž úsečka, a pokud nebyl vyvolán příkaz `\curvewarnfalse`, vystoupí na obrazovku varování, protože kvadratická funkce neumí vykreslovat úseky s inflexním bodem. Někdy je možná náprava ve zjemnění kroku bodů v kritickém úseku křivky.

Vynášení všech křivek vůbec je ovlivňováno řadou již popsaných parametrů. Každý příkaz pro kreslení z popisovaných stylů totiž

1. **před vyvoláním** `\curvedashes`, anebo při prázdném vzoru nebo nulové jeho délce
   - a při `SC` nulovém nebo prázdném nakreslí spojitou čáru;
   - při `SC` kladném rozdělí se každý úsek křivky mezi danými body na `SC` podúseků a ve všech uzlových bodech (i v nově vzniklých) se vynesou plné čverečky o straně rovné tloušťce čáry,
   - při `SC` záporném se v případě, že bude definován `znak_nebo_symbol` (příkazem `\curvesymbol`), rozdělí každý úsek křivky na `-SC` podúseků a ve všech uzlových bodech (i v nově vzniklých) se vykreslí `znak_nebo_symbol`,
   - a při jiných podmínká než právě uvedených se nevykreslí nic;
2. **po vyvolání** `\curvedashes`, který nadefinoval nenulovou délku vzoru, nakreslí
   - při nulovém nebo chybějícím `SC` vzor specifikovaný příkazem `\curvedashes`, zredukovaný ve své délce o `\csdiameter`, pro vynechávání mezer na symboly,
     - jinak, byl-li definován `znak_nebo_symbol` a je-li `\overgang` větší než `0pt`, vykreslí ve všech pozicích `znak_nebo_symbol`;
     - jinak při nulovém `\overgang` vykreslí místo skupin vzorů `znak_nebo_symbol` v místech jejich začátků;
   - v opačném případě vykreslí vzor, ale vynechá ve všech zadaných bodech a v `|SC|-1` bodech podúseků mezery o šířce `\csdiameter`, a to
     - přesně, není-li zadán `znak_nebo_symbol`, avšak přerušovaný mezerami,
     - a v opačném případě upravený tak, aby přesně dal celý počet vzorů mezi mezerami.

Příklady:

```
    \unitlength1mm
    \begin{picture}(160,100)
% 1.
      \linethickness{1pt}
      \put(0,90){\curve(0,0, 20,5, 60,0) \hskip-3ex a}
      \linethickness{5pt}
      \put(0,80){\curve[3](0,0, 20,5, 60,0) \hskip-3ex b}
      \curvesymbol{$\Diamond$}
      \put(0,70){\curve[-3](0,0, 20,5, 60,0) \hskip-3ex c}
      \linethickness{1pt}
      \curvesymbol{}
      \put(0,60){\curve[-3](0,0, 20,5, 60,0) \hskip-3ex d}
% 2.
      \curvedashes[1.2mm]{0,4,1,3,1,4}
      \put(0,50){\curve(0,0, 20,5, 60,0) \hskip-6ex e}
      \settowidth{\csdiameter}{00}
      \put(0,40){\curve(0,0, 20,5, 60,0) \hskip-6ex f}
      \curvesymbol{$\circ$}
      \setlength{\overhang}{1pt}
      \put(0,30){\curve(0,0, 20,5, 60,0) \hskip-6ex g}
      \curvesymbol{\addtocounter{i}{1}\thei}
      \setlength{\overhang}{0pt}
      \put(0,20){\curve(0,0, 20,5, 60,0) \hskip-6ex h}
      \put(0,10){\curve[-3](0,0, 20,5, 60,0) \hskip-6ex i}
      \curvesymbol{}
      \put(0,0){\curve[-3](0,0, 20,5, 60,0) \hskip-6ex j}
    \end{picture}
```

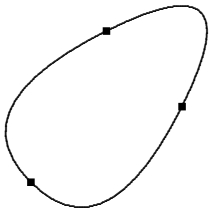


`\closecurve[SC]{seznam}` **Uzavřená křivka**

Makro je velice podobné makru `\curve`, s nímž má i stejné parametry. Jedinou, i když podstatnou, odchylkou je zabudovaná vazba mezi prvním a posledním bodem seznamu souřadnic. Jinak se chová stejně jako příkaz `\curve`.

Příklad:

```
\unitlength1mm
\begin{picture}(40,30)(0,0)
  \renewcommand\diskpitchstretch{1}
  \curvedashes{}
  \put(-10,15){\closecurve(0,0, 20,10, 10,20)}
  \curvesymbol{\rule{1mm}{1mm}}
  \put(-10,15){\closecurve[-1](0,0, 20,10, 10,20)}
\end{picture}
```

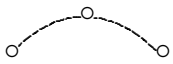


`\tagcurve[SC]{seznam}` **Křivka s externími body**

Funguje stejně jako `\curve` jen s tím rozdílem, že krajní úseky křivky se nevykreslí. To znamená, že krajní body slouží hlavně jen ke zjištění směru tečny v sousedních vnitřních bodech. Z řečeného plyne, že by měly být zadány alespoň 4 body, tj. 8 jejich souřadnic. Pokud je v seznamu pouze 6 souřadnic, vykreslí se jen poslední úsek (který by se normálně vynechával).

Příklad:

```
\unitlength1mm
\begin{picture}(140,35)(-50,0)
  \curvesymbol{$\circ$}%
  \curvedashes[1pt]{0, .5,1,3,1,.5}%
  \put(0,15){\tagcurve[-1](0,0, 10,15, 20,20, 30,15, 40,0)}%
  \curvedashes{}%
  \put(0,15){\curve[-1](0,0, 10,15, 20,20, 30,15, 40,0)}%
\end{picture}
```



`\arc[SC](x,y){úhel}` **Kruhový oblouk**

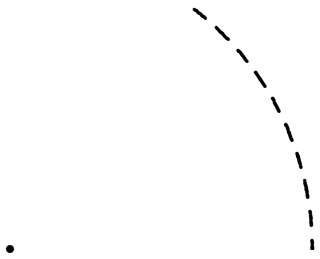
Standardní LaTeX umí nakreslit nanejvýše čtvrtkruh pokrývající jeden kvadrant, ale jen standardní tloušťkou čáry a navíc pouze do průměru 14mm. Makro `\arc` kreslí obecnou část kružnice o libovolném poloměru čarou o libovolné tloušťce. Uživatel jen musí dodat následující parametry:

`SC` nepovinný počet podúseků, na něž budou úseky křivky rozděleny;
`(x,y)` reálná čísla – souřadnice bodu, ze kterého oblouk vychází. Při tom jeho střed křivosti je v bodě, určeném umisťovacím příkazem.
`úhel` vrcholový úhel oblouku ve stupních, kladný proti směru točení hodinových ručiček.

V následujícím příkladě si povšimněme druhého volání `\arc`. I když se zdá zbytečný, protože kreslí oblouk takřka nulové délky, je důležitý, protože potlačuje chybu, spočívající v generování vertikálních mezer mezi odstavci, úměrných výšce kresleného oblouku.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(0,-5)
  \put(0,0){\circle*{1}}
  \linethickness{1pt}
  \curvedashes{0, 1,2,2,2,1}%
  \put(0,0){\arc(40,0){53.5}}
  \put(0,0){\arc(40,0){0.1}}
\end{picture}
```

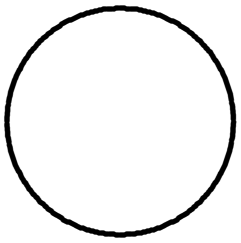


`\bigcircle[SC]{diam}` **Kružnice**

Jde o velmi užitečné makro. Standardní LaTeX, jak již bylo řečeno výše, umožňuje kreslit kružnice o maximálním průměru 14 mm, a to pouze o dvou tloušťkách čáry. Makro `\bigcircle` tato omezení nemá, a tak jím lze kreslit kružnice libovolného průměru a libovolné tloušťky čáry. Reálný parametr `diam` udává průměr kružnice v násobcích `\unitlength`. Verze plného kruhu není k dispozici.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-20)
  \linethickness{2pt}
  \put(0,0){\bigcircle{30}}
\end{picture}
```



`\bezier[SC](xa,ya)(xb,yb)(xc,yc)` **Bézierova křivka**

Jde o zrychlenou a rozšířenou variantu stejnojmenného makra ze souboru `bezier.sty`, který se při použití `curves.sty` nesmí v `documentstyle` deklarovat. Makro `\bezier` má následující parametry:

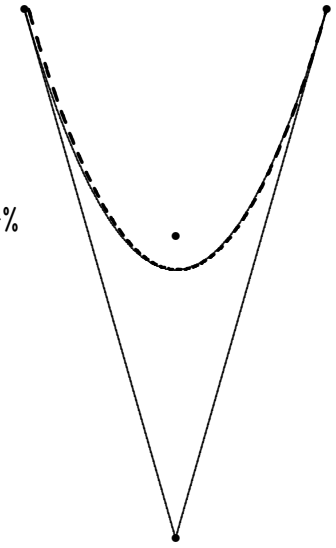
`{SC}` `|SC|` povinný počet podúseků, na něž bude úsek mezi body `a` a `c` rozdělen,
`(xa,ya)` souřadnice prvního dotykového bodu Bézierovy křivky,
`(xb,yb)` souřadnice průsečíku tečen dotýkajících se v bodech `a` a `c` Bézierovy křivky,
`(xc,yc)` souřadnice druhého dotykového bodu Bézierovy křivky.

Výhoda tohoto makra proti původnímu není pouze v rychlosti práce, ale i v možnosti

vynášet čáry s různým vzorem přerušování. Bohužel zatím je vynášení přerušované čáry zatíženo malou chybou, která se s každým přerušením kumuluje, což je dobře patrno z příkladu. Nepřerušovaná křivka tuto chybu nevykazuje. Počátek je vyznačen silnou tečkou.

Příklad:

```
\unitlength1mm
\begin{picture}(40,80)(-20,-50)
  \put(0,0){\circle*{1}}%
  \linethickness{1pt}%
  \curvedashes{0,.5,1,1,1,.5}%
  \put(0,0){\bezier{10}(-20,30)(0,-40)(20,30)}%
  \thinlines%
  \curvedashes{}%
  \put(0,0){\curve(-20,30, 0,-40)}%
  \put(0,0){\curve(20,30, 0,-40)}%
  \put(-20,30){\circle*{1}}%
  \put(20,30){\circle*{1}}%
  \put(0,-40){\circle*{1}}%
  \put(0,0){\curve(-20,30,0,-4.5,20,30)}%
\end{picture}
```



lines-1

6.3 Grafika s makry rplot.sty

Bylo již konstatováno, že prostředí `picture` dává velice omezené možnosti pro grafické práce. Soubory `curves.sty`, příp. `curvesls.sty` znamenaly výrazné zlepšení, protože daly možnost vynášet přímé i křivé čáry libovolného směru a tloušťky. Přesto se však ukázalo, že některé práce spojené s technickými aplikacemi je zapotřebí dělat složitě a že by bylo účelné zpracovat nový styl, který by uživatelovi usnadnil jeho práci. Tuto roli by měl sehrát styl `rplot.sty`. Jeho příspěvek k aritmetice byl již vyložen výše. Zbývá již jen popsat jeho možnosti v oblasti grafiky.

Makra `rplot.sty` pro grafické práce lze rozdělit v podstatě do třech skupin, a to na rozšiřující možnosti poskytované stylem `curves.sty`, podporující prostředí `picture` a na speciální.

6.3.1 Nadstavba nad curves.sty

Z popisu maker souboru `curves.sty` a z jejich používání vyplynulo, že zadávání požadavků by mohlo být zjednodušeno, anebo doplněno k rozšíření možností při aplikacích. Tak například pro zadání měřítek bylo nutno použít čtyř příkazů nebo pro vykreslení čáry s vyznačením zadaných bodů bylo zapotřebí vyvolat příkaz `\curve` několikrát. Pro usnadnění práce uživatele bylo vypracováno a do souboru `rplot.sty` zařazeno makro `\curves`, jímž lze vykreslit libovolnou čarou křivku i s body jediným voláním, protože parametry makra obsahují všechny potřebné informace:

`\curves{symbol}{vzor}(seznam)` **Obecná křivka s body**

Parametry makra jsou:

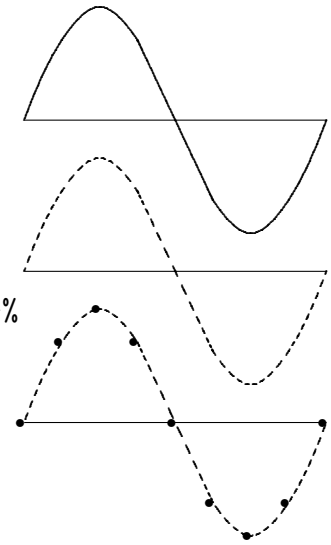
`symbol` uživatelem zvolený symbol, který bude vynášen v jenotlivých daných bodech. Může jím být libovolný znak ASCII, speciální znaky matematického režimu, nebo i symboly zkonstruované uživatelem a uložené příkazem `savebox`. Parametr může být i prázdný, ale potom se žádné symboly v uzlových bodech nevynášejí;

`vzor` pokud není prázdný, což by znamenalo, že má být vynesena plná čára, jde o seznam délek v násobcích `\unitlength`, vyjadřujících střídavě délky čárek a mezer, např.:
délka čárky,
délka mezery,
délka čárky, … atd.

`seznam` seznam dvojic souřadnic x_j , y_j zadaných uzlových bodů křivky, jimiž bude proložena soustava kvadratických Bézierových křivek.

Příklad:

```
\unitlength1mm
\begin{picture}(40,60)(0,-20)
  \Cmd \xy={0,0, 1,.707, 2,1, 3,.707,%
     4,0, 5,-.707, 6,-1, 7,-.707, 8,0}%
  \setscales(5,0,0,15)%
  \put(0,0){\line(1,0){40}}%
  \put(0,0){\curves{\circle*{1}}{.5,.5,.5}(\xy)}%
  \put(0,20){\line(1,0){40}}%
  \put(0,20){\curves{}{.5,.5,.5}(\xy)}%
  \put(0,40){\line(1,0){40}}%
  \put(0,40){\curves{}{}(\xy)}%
\end{picture}
```



Z příkladu je patrno, že vynášení křivek příkazem `\curve` ze souboru `curvesls.sty`, který byl zde použit, trpí ještě nedostatky, o nichž byla řeč výše, pokud není čára plná a pokud se mají vykreslit uzlové body. Lze však očekávat, že ve vyšších verzích `curves.sty` budou tyto nedostatky odstraněny. V příkladu se vyskytl příkaz pro nastavování měřítek u grafických objektů `\setscales`. Je jedním ze dvojice nových příkazů pro zadávání měřítek. Oba mají usnadnit uživatelovi práci, protože odstraňují malou přehlednost v jejich názvech a zdlouhavé zadávání. Jsou to příkazy:

`\setscales(txx,txy,tyx,tyy)` **Nastavení měřítek**

Makro naplní čtveřici měřítek potřebnou pro lineární transformaci grafických objektů vynášených příkazy souboru `curves.sty` i `rplot.sty`. Jde v podstatě o zadání prvků celé transformační matice $\boldsymbol{T}$ po řádkách, kde

$$\boldsymbol{T} = \begin{bmatrix} t_{xx} & t_{xy} \\ t_{yx} & t_{yy} \end{bmatrix}$$

Ty je třeba předem vypočítat, aby se dosáhlo žádoucího účinku na zobrazení objektu. Parametrem makra je seznam měřítek (v kulatých závorkách), jehož prvky jsou reálná čísla nebo povely:

- t_{xx} nastavující `\xscale` z `curves.sty`
- t_{xy} nastavující `\yscalex`
- t_{yx} nastavující `\xscaley`
- t_{yy} nastavující `\yscale`

Implicitní nastavení měřítek je jako po vyvolání makra `\setscales(1,0,0,1)`.

`\scalerot{alpha}` **Nastavení měřítek pro rotaci objektů**

Realizace reálné aritmetiky s výpočtem elementárních funkcí umožnily, aby se měřítka pro rotaci objektů dala nastavit automaticky, protože transformační matice je dána výrazem

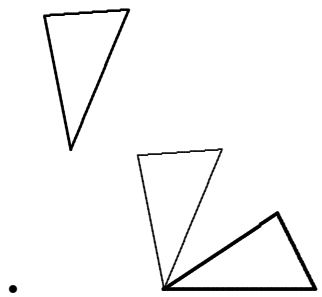
$$\boldsymbol{T} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

K otočení o úhel α měřený kladně proti směru pohybu hodinových ručiček dochází **jen** u grafických objektů nakreslených příkazy ze souboru `curves.sty` a `rplot.sty`. Například u textů dojde pouze k natočení referenčního bodu, avšak text sám zůstane v původní orientaci. Parametr `alpha` je úhel α ve stupních.

Jako příklad uveďme rotaci trojúhelníka o 67.5 stupňů. Vynesena je jak základní poloha (velmi silně), tak po rotaci kolem referenčního bodu (slabě), který je dán souřadnicemi v příkazu `\put`, tak i po rotaci kolem počátku zobrazeného silnou tečkou (silně). Aby trojúhelník nerotoval pouze kolem svého referenčního bodu, je zapotřebí ho vynést umisťovacím příkazem `\scaleput`:

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(0,-15)
  \Cmd \xy={0,0, 20,0, 15,10, 0,0}
  \put(0,0){\circle*{1}}
  \linethickness{1.5pt}
  \put(20,0){\polyline(\xy)}
  \scalerot{67.5}
  \thinlines
  \put(20,0){\polyline(\xy)}
  \thicklines
  \scaleput(20,0){\polyline(\xy)}
\end{picture}
```



Při používání `curvesls.sty` se nepříjemně pociťovala absence makra, které by vynášelo libovolně lomenou čáru složenou z úseček, a to nikoliv pomocí opakovaného volání makra `\curve` pro každý lineární úsek zvlášť, ale jediným zadáním podobným volání makra `\curve`. Tak vzniklo makro `\polyline`, které bylo v příkladu použito:

`\polyline[SC](seznam)` **Čára složená z úseček**

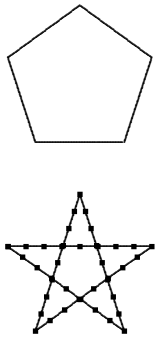
Jde o velmi užitečné makro, které se uplatnilo i v dále uvedených makrech pro kreslení jednoduchých geometrických obrazců a při vynášení diagramů průběhů měřených závislostí. Jeho parametry jsou totožné s parametry makra `\curve`:

`SC` nepovinný parametr "symbol count", který rozhoduje, pokud existuje, nebo je nenulový, o vytvoření `|SC|` podúseků z každého úseku mezi dvěma body. Podrobný popis vlivu `SC` na kreslení čáry viz u příkazu `\curve`

`seznam` uspořádaný seznam dvojic reálných souřadnic x_j a y_j koncových bodů úseček, z nichž je čára složena. Čára bude vynášena v pořadí zadaných bodů, jak je patrno i z následujícího příkladu:

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-20)
  \put(0,15){\polyline(-9.51,3.09, 0,10, 9.51,3.09,
    5.88,-8.09, -5.88,-8.09, -9.51,3.09)}
  \Cmd \xy={-9.51,3.09, 9.51,3.09, -5.88,-8.09,
    0,10, 5.88,-8.09, -9.51,3.09}
  \put(0,-5){\polyline(\xy)}%      hvězda
  \linethickness{2pt}
  \put(0,-5){\polyline[8](\xy)}%   body
  \thinlines
\end{picture}
```



`\polylineq[SC](seznam)` **Čára složená z úseček s pevným krokem v x**

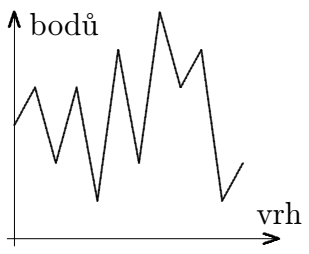
Je-li vzájemná odlehlost bodů ve směru osy `x` konstantní, je zbytečné uvádět `x`-ové souřadnice, když mohou být generovány automaticky jako vzájemně ekvidistantní. K tomu slouží právě toto makro, jehož parametry jsou:

`SC` nepovinný parametr se stejným účinkem jako u příkazů `\curve` nebo `\polyline`;

`seznam` seznam reálných pořadnic y_j oddělených navzájem čárkami. Po vyvolání si makro `\polylineq` samo vytvoří seznam dvojic souřadnic bodů `(0,`y_1`,` `1,`y_2 `, ... )`. Správné vynesení této závislosti lze zajistit parametry příkazů `\put` a měřítky.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(0,-5)
  \put(4,0){\XYaxes(-1,35){vrh}(-1,30){bodů}}
  \setscales(2.75,0,0,5)
  \put(4,0){\polylineq(3,4,2,4,1,5,2,6,4,5,1,2)}
\end{picture}
```



Jako příklad je uveden diagram výsledků náhodného házení kostkou.

Soubor `curves.sty` obsahuje příkaz `\scaleput` jako měřítkovatelnou alternativu standardního příkau LaTeXu `\put`. Nedává však analogický příkaz k `\multiput`. Soubor `rplot.sty` ho obsahuje pod názvem `\mscput` (multi-scaleput):

`\mscput(x,y)(dx,dy){n}{objekt}` **Multi-scaleput**

Makro dává stejné možnosti jako makro `\multiput`, ale navíc umožňuje zadávat souřadnice referenčního bodu, jejich přírůstky a počet opakování obecně, tedy nikoliv jen jako pouhá čísla, a pomocí měřítek lineárně transformovat objekty.

`(x,y)` reálné souřadnice referenčního bodu prvně vyneseného objektu jako reálná čísla nebo reálné povely;

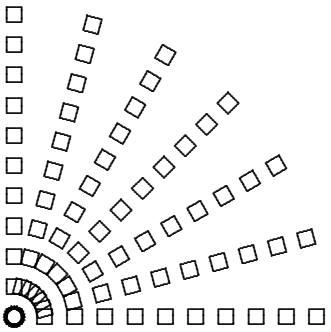
`(dx,dy)` reálné přírůstky souřadnic referenčního bodu objektu pro opakovaná vynášení

`n` celočíselný počet objektů, které budou vyneseny. Může jím být libovolná celočíselná konstanta.

`objekt` libovolný grafický objekt sestavený z příkazů souborů `curve.sty` a `rplot.sty`. Jen tak bude schopný lineární transformace.

Příklad:

```
\newcnt{f}
\unitlength1mm
\begin{picture}(40,30)(2,-11)
  \Ifor f=(0:15:90)%
     {\scalerot{f}%
      \mscput(0,0)(4,0){11}%
        {\polyline(1,1,1,-1,-1,-1,-1,1,1,1)}
     }%
\end{picture}
```



`\mscputxy(seznam){objekt}` **Souřadnicový multi-scaleput**

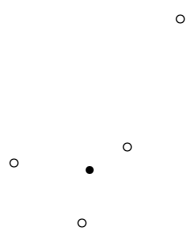
Toto makro je určeno pro vícenásobné vynášení objektů do bodů, jejichž souřadnice se nemění s konstantními přírůstky, ale které jsou předem známé.

`seznam` je seznam dvojic reálných souřadnic referenčních bodů, do nichž bude objekt opakovaně vynesen.

`objekt` libovolný grafický objekt sestavený z příkazů souborů `curve.sty` a `rplot.sty`. Jen tak bude schopný lineární transformace.

Příklad:

```
\unitlength1mm
\begin{picture}(40,30)(-20,-15)
  \put(0,0){\circle*{1}}
  \mscputxy(5,3, -10,1, 12,20, -1,-7){\circle{1}}
\end{picture}
```



`\mscputxt(seznam)` **Souřadnicový multi-scaleput textů**

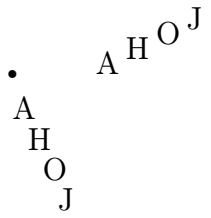
Makro bude převážně používáno pro popisování grafického výstupu, protože umožňuje operativně posílat texty na žádané souřadnice. Má jediný parametr `seznam`, který je složen z trojic informací oddělených navzájem čárkami. Každou trojici tvoří

x_j	je reálná x-ová souřadnice j-tého referenčního bodu textu (objektu)
y_j	je reálná y-ová souřadnice j-tého referenčního bodu textu (objektu)
$text_j$	j-tý text, který se umístí do bodu (x_j,y_j)

Poloha jednotlivých textů může být transformována, texty samotné však nikoliv.

Příklad:

```
\unitlength1mm
\begin{picture}(40,30)(-10,-30)
  \put(0,0){\circle*{1}}
  \Cmd \txt={6,0,A, 10,2,H, 14,0,O, 18,2,J}
  \put(5,0){\mscputxt(\txt)}
  \scalerot{-90}
  \mscputxt(\txt)
\end{picture}
```



6.3.2 Podpora prostředí picture

Zpracování aritmetiky a kreslení křivek přinesly další možnosti rozšíření palety příkazů použitelných v prostředí `picture`. Pro technické aplikace hrají dosti důležitou roli příkazy pro ovládání kreslicí plochy. Do této skupiny patří zejména makra pro kreslení os a stupnic.

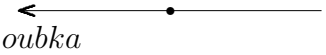
`\Xaxis(xbeg,xfin)(xtxt,ytxt){text}` **Vynesení osy x**

Makro slouží pro nakreslení osy `x` o zadané délce a popisu v požadovaném místě. Parametry makra jsou:

`(xbeg,xfin)`	reálné souřadnice začátku a konce osy `x` na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném `xfin` bude nakreslena horizontální šipka;
`(xtxt,ytxt)`	reálné souřadnice referenčního bodu textu – popisu osy `x`;
`text`	libovolný popis osy s referenčním bodem v levém dolním rohu prvního znaku textu.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-25)
  \put(0,0){\circle*{1}}
  \put(0,0){\Xaxis(20,-20)(-26,-5){$hloubka$}}
\end{picture}
```



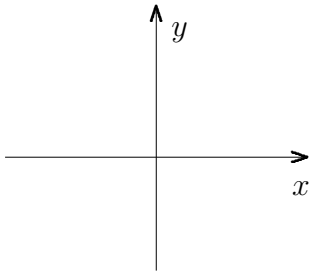
`\Yaxis(ybeg,yfin)(xtxt,ytxt){text}` **Vynesení osy y**

Makro slouží pro nakreslení osy `y` o zadané délce a popisu v požadovaném místě. Parametry makra jsou:

`(ybeg,yfin)`	reálné souřadnice začátku a konce osy `y` na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném `yfin` bude nakreslena vertikální šipka;
`(xtxt,ytxt)`	reálné souřadnice referenčního bodu textu – popisu osy `y`;
`text`	libovolný popis osy s referenčním bodem v levém dolním rohu prvního znaku textu.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-25)
  \put(0,0){\Xaxis(-20,20)(18,-5){$x$}}
  \put(0,0){\Yaxis(-15,20)(2,16){$y$}}
\end{picture}
```



Z příkladu je patrno, že dvojicí příkazů `\Xaxis` a `\Yaxis` lze vytvořit libovolnou soustavu os, s libovolným a libovolně umístěným popisem. Mnohdy je tato volnost přepychem, a proto bylo sestaveno makro `\XYaxes`, které za cenu jistých omezení nevyžaduje na uživateli tolik dat.

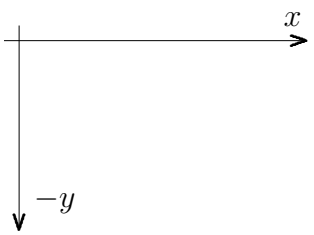
`\XYaxes(xbeg,xfin){xtext}(ybeg,yfin){ytext}` **Vynesení osového kříže**

Makro vynese a popíše osy `x` a `y`. Popisy budou umístěny u konců os a s ohledem na vyhrazené místo by měly být krátké. Parametry makra jsou:

`(xbeg,xfin)`	reálné souřadnice začátku a konce osy `x` na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném `xfin` bude nakreslena horizontální šipka;
`xtext`	text pro popis osy `x`;
`(ybeg,yfin)`	reálné souřadnice začátku a konce osy `y` na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném `yfin` bude nakreslena vertikální šipka;
`ytext`	text pro popis osy `y`;

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-5,-35)
  \XYaxes(-2,38){$x$}(2,-35){$-y$}
\end{picture}
```



Při kreslení diagramů není nutné jen nakreslit osy, ale i je označit stupnicemi. Zatímco nakreslení lineární stupnice není žádným problémem, protože máme k dispozici

příkazy `\multiput`, příp. `\mscput`, je vynášení nelineárních stupnic, případně celé osnovy, úloha podstatně složitější. Nejobvyklejší nelineární stupnicí je stupnice logaritmická. Pro ni jsou připravena následující dvě makra:

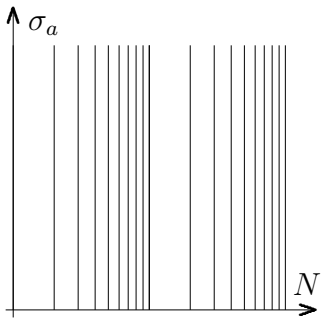
`\logx(seznam){ndekád}{délka}{tloušťka}` **Logaritmické dělení osy x**

Jeho parametry jsou:

`seznam`	je seznamem reálných kladných hodnot z jednoho intervalu (dekády), pro které bude vykreslena čárka logaritmického dělení osy `x`;
`ndekád`	je celočíselný počet dekád, které se mají vykreslit;
`délka`	je reálné číslo udávající v násobcích `\unitlength` dálku čar dělení osy `x`;
`tloušťka`	je reálná délka (s rozměrem!) udávající tloušťku čar

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)
  \XYaxes(-1,40){$N$}(-1,40){$\sigma_a$}
  \setscales(18,0,0,1)
  \scaleput(0,0)%
    {\logx(1,2,3,4,5,6,7,8,9,10){2}{35}{.2pt}}
\end{picture}
```



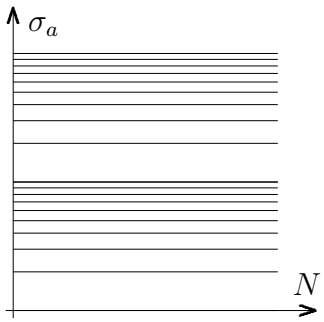
`\logy(seznam){ndekád}{délka}{tloušťka}` **Logaritmické dělení osy y**

Jeho parametry jsou:

`seznam`	je seznamem reálných kladných hodnot z jednoho intervalu (dekády), pro které bude vykreslena čárka logaritmického dělení osy `y`;
`ndekád`	je celočíselný počet dekád, které se mají vykreslit;
`délka`	je reálné číslo udávající v násobcích `\unitlength` dálku čar dělení osy `y`;
`tloušťka`	je reálná délka (s rozměrem!) udávající tloušťku čar

Příklad:

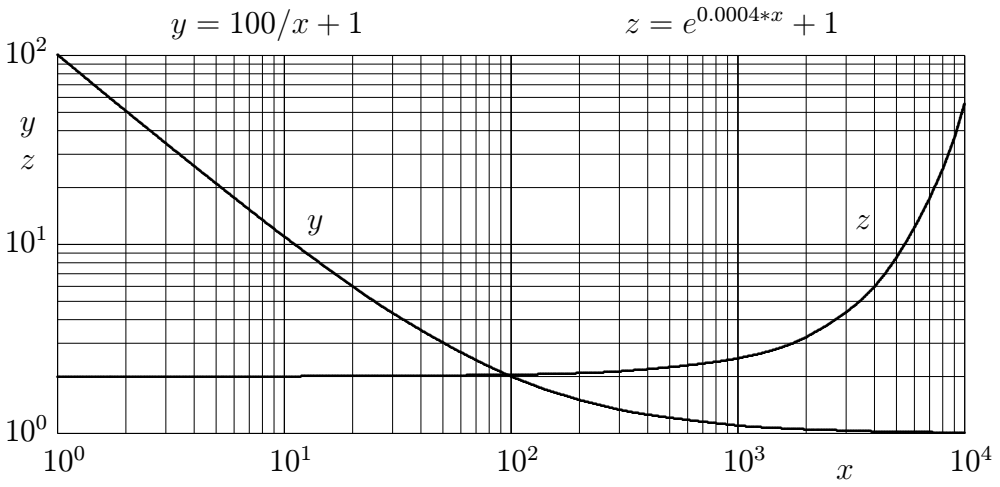
```
\unitlength1mm
\begin{picture}(40,40)(0,-6)
  \XYaxes(-1,40){$N$}(-1,40){$\sigma_a$}
  \setscales(1,0,0,17)
  \scaleput(0,0)%
    {\logy(1,2,3,4,5,6,7,8,9,10){2}{35}{.2pt}}
\end{picture}
```



Kombinací obou příkazů lze získat grafický papír „log-log“, případně jednoho z těchto příkazů a lineární stupnice papír „semilog“.

Příklad:

```
\newcnt{x}
\Cmd \xy={}
\Cmd \xz={}
\Lfor x=(1,3,10,30,100,300,1000,2000,4000,7000,10000)%
{\Rlog(x)\addn{F}{40000}\Rget\x={F}%
 \Rdiv F={100}/{x}\addn{F}{10000}\Rlog(F)\Rget\y={F}%
 \setc{F}{x}\muln{F}{4}\Rexp(F)\addn{F}{10000}\Rlog(F)\Rget\z={F}%
 \ifthenelse{\thex=1}{\edef\xy{\x,\y}\edef\xz{\x,\z}}
                     {\edef\xy{\xy,\x,\y}\edef\xz{\xz,\x,\z}}%
}
\Cmd \c={1,2,3,4,5,6,7,8,9,10}
\unitlength1mm
\begin{picture}(160,55)(-20,-5)
  \setscales(30,0,0,1)
  \scaleput(0,0){\logx(\c){4}{50}{.2pt}}
  \Ifor x=(0:1:4){\put(-2,-5){\scaleput(\thex,0){10$^\thex$}}}
  \setscales(1,0,0,25)
  \scaleput(0,0){\logy(\c){2}{120}{.2pt}}
  \Ifor x=(0:1:2){\put(-7,-1){\scaleput(0,\thex){10$^\thex$}}}
  \setscales(30,0,0,25)
  \thinlines
  \multiput(0,0)(30,0){5}{\line(0,1){50}}
  \multiput(0,0)(0,25){3}{\line(1,0){120}}
  \thicklines
  \scaleput(0,0){\curve(\xy)}
  \scaleput(0,0){\curve(\xz)}
  \setscales(1,0,0,1)
  \mscputxt(103,-6,$x$, -5,40,$y$, -5,35,$z$,
            15,53,$y=100/x+1$, 75,53,$z=e^{0.0004*x}+1$,
            105.5,27,$z$, 33,27,$y$)
\end{picture}
```



Účelem tohoto příkladu bylo ukázat nejen použití maker `\logx` a `\logy`, ale i možnosti až dosud vysvětlených maker v sestavě s aritmetikou a funkcemi.

6.3.3 Nové grafické objekty

Opět pro usnadnění práce uživatele byla vytvořena makra pro kreslení elementárních grafických útvarů. Mnohdy je jejich zápis tak jednoduchý, že vzniká otázka, zda vůbec mělo smysl je sestavovat. Nicméně zde jsou:

`\strline(xbeg,ybeg)(xfin,yfin){tloušťka}` **Rychlé kreslení úsečky**

Využívají se zde příkazy `\special` Eberharda Mattese, autora emTEXu. Parametry příkazu jsou:

`(xbeg, ybeg)`	reálné souřadnice počátečního bodu úsečky;
`(xfin, yfin)`	reálné souřadnice konečného bodu úsečky;
`tloušťka`	reálná tloušťka čáry s rozměrem!

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(5,-20)
  \put(0,0){\circle*{1}}
  \setscales(1,0,0,1)
  \scaleput(0,0){\strline(10,-5)(38,5){.3pt}}
  \scalerot{61.5}
  \scaleput(0,0){\strline(10,-5)(38,5){2pt}}
\end{picture}
```

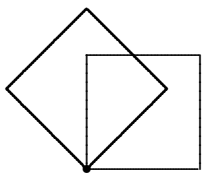
Protože soubor `curvesls.sty` stejně využívá příkazy `\special`, je možno téhož výsledku dosáhnout pomocí `\polyline(xbeg,ybeg, xfin,yfin)`.

`\square(a)` **Čtverec**

Vyvolání tohoto makra vytvoří na kreslící ploše čtverec o straně `a` s referenčním bodem vlevo dole

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-10,-15)
  \put(0,0){\circle*{1}}
  \put(0,0){\square(15)}
  \thicklines
  \scalerot{45}
  \scaleput(0,0){\square(15)}
\end{picture}
```

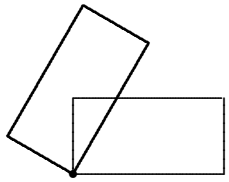


`\rectan(a,b)` **Obdélník**

Makro vygeneruje obdélník s referenčním bodem v levém dolním rohu a o straně `a` rovnoběžné s osou `x` a straně `b` rovnoběžné s osou `y`. Jeho natočení lze dosáhnout příkazem `\scaleput` předcházeným nastavením měřítek pomocí `\scalerot`.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-10,-15)
  \put(0,0){\circle*{1}}
  \put(0,0){\rectan(20,10)}
  \thicklines
  \scalerot{60}
  \scaleput(0,0){\rectan(20,10)}
\end{picture}
```

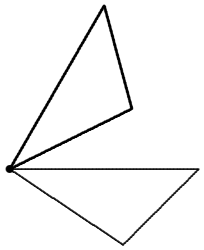


`\triangle(xb,yb,xc,yc)` **Trojúhelník**

Makro vynese obecný trojúhelník daný referenčním bodem `A` nastaveným umisťovacím povelem a zbylými dvěma body `B` a `C` danými souřadnicemi `(xb,yb)` a `(xc,yc)`

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-5,-15)
  \put(0,0){\circle*{1}}
  \thinlines
  \put(0,0){\triang(25,0, 15,-10)}
  \thicklines
  \scalerot{60}
  \scaleput(0,0){\triang(25,0, 15,-10)}
\end{picture}
```



Následující čtyři makra kreslí čtvrtkružnice. Jména těchto maker začínají znakem „q“. Jako součást jména makra jsou ještě dvě písmena, která mají v LaTeXu obvyklý význam, totiž symboly `t` nebo `b` pro horní (top) resp. dolní (bottom) a symboly `l` pro levou (left) nebo `r` pro pravou (right) část kružnice. Jejich názvy tedy jsou:

`\qtl{radius}` **Horní levý oblouk**

`\qtr{radius}` **Horní pravý oblouk**

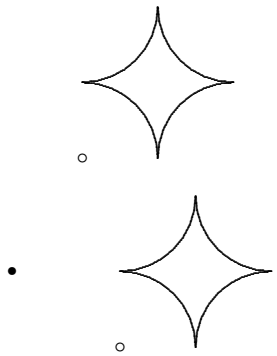
`\qbl{radius}` **Dolní levý oblouk**

`\qbr{radius}` **Dolní pravý oblouk**

Jako jediný argument je u všech maker reálný `radius`. Referenční bod je střed křivosti. V následujícím příkladu jsou použity všechny čtyři příkazy, a to v podobrázku uloženém v boxu `\cross`. Z výsledku je patrný očekávaný výsledek, totiž, že se objekt přeloží a uloží do boxu, a tím je jeho orientace fixována. Potom i při použití rotace, otočí se kolem počátku pouze poloha referenčního bodu (vyznačeného prázdným kroužkem), nikoliv však objekt sám. Ten se po přeložení a uložení do boxu chová již jen jako písmeno (s levým dolním rohem jako referenčním).

Příklad:

```
\newsavebox{\cross}
\unitlength1mm
\begin{picture}(40,40)(-5,-35)
  \put(0,0){\circle*{1}}
  \savebox{\cross}(20,20)
    {\begin{picture}(20,20)(-10,-10)
       \scaleput(10,10){\qbl(10)}
       \scaleput(-10,10){\qbr(10)}
       \scaleput(-10,-10){\qtr(10)}
       \scaleput(10,-10){\qtl(10)}
     \end{picture}
    }
  \thinlines
  \put(15,0){\usebox{\cross}}
  \scalerot{45}
  \scaleput(15,0){\usebox{\cross}}
```



Další skupina maker vynáší objekty, v nichž se vyskytují šipky. I ty se v technických aplikacích objevují dosti často. Pro ně jsou k dispozici tato makra:

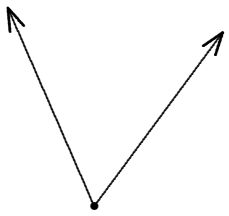
`\arrow(xpeak,ypeak){wing}` **Obecná šipka**

Makro má následující parametry:

`(xpeak,ypeak)`	reálné relativní souřadnice vrcholu šipky;
`wing`	délka vlastního křidélka šipky v `\unitlength`

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-15)
  \put(0,0){\circle*{1}}
  \scaleput(0,0){\arrow(17,23){3}}
  \thicklines
  \scalerot{60}
  \scaleput(0,0){\arrow(17,23){3}}
\end{picture}
```



Speciálním případem šipek jsou špičky horizontální a vertikální.

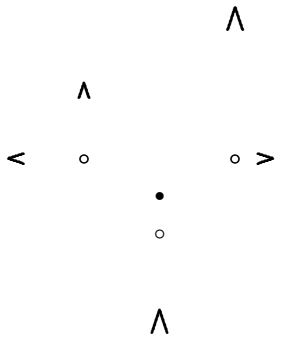
`\harrow{xdim}{wing}` **Horizontální špička**

`\varrow{ydim}{wing}` **Vertikální špička**

U těchto maker je `xdim` a `ydim` horizontální resp. vertikální odlehlost špičky od referenčního bodu daného umisťovacím příkazem a `wing` délka křidélka. Znaménko `wing` určuje orientaci špičky. Kladné znaménko odpovídá směrům „vpravo“ a „nahoru“. Všechny tyto parametry udávají skutečné rozměry v jednotkách `\unitlength`.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-25)
  \put(0,0){\circle*{1}}
  \put(10,5){\harrow{5}{2}}
  \put(10,5){\circle{1}}
  \put(-10,5){\harrow{-10}{-2}}
  \put(-10,5){\circle{1}}
  \put(10,5){\varrow{20}{3}}
  \put(10,5){\circle{1}}
  \put(-10,5){\varrow{10}{2}}
  \put(-10,5){\circle{1}}
  \put(0,-5){\varrow{-10}{3}}
  \put(0,-5){\circle{1}}
\end{picture}
```



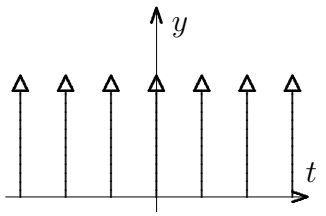
Z výsledků příkladů je patrno, že znaménko `xdim` a `ydim` rozhodují o orientaci špičky. Toho lze s výhodou použít např. při tvorbě makra pro kótování.

`\puls{ydim}` **Diracův impuls**

Jde o velmi učitečné makro pro řadu technických oblastí. Jeho jediným parametrem je `ydim` – reálná výška impulzu jako násobek `\unitlength`. V následujícím příkladu je makro vyvoláno opakovaně k vytvoření konečného úseku Diracova hřebenu, který se často vyskytuje při zpracování signálů.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-18,-5)
  \put(0,0)%
    {\XYaxes(-20,20){$\ \ t$}(-2,25){$y$}}
  \multiput(-18,0)(6,0){7}{\puls{16}}
\end{picture}
```



6.3.4 Speciální diagramy

Dosti často se vyskytuje potřeba vynášení diagramů četností jevů v jistých třídách nebo grafického vyjádření podílu jevu na celku. K tomu slouží tzv. histogramy, resp. kruhové diagramy.

Histogram

Histogramy jsou diagramy četností výskytů jevu v subintervalech nezávisle proměnné, jimž říkáme třídy. Četnost v subintervalu se vyjadřuje obdélníkem, jehož výška je úměrná počtu výskytů jevu v dané třídě. Třídy bývají rovnoměrně rozdělené na intervalu

nezávisle proměnné. Četnosti jsou celá čísla n_i. Malým zobecněním dostaneme relativní četnosti

$$r_i = n_i/N\,; \qquad N = \Sigma_i n_i$$

Oba tyto typy histogramu se stejně širokými třídami lze konstruovat pomocí makra vyvolávaného jako grafický objekt – parametr příkazu `\put` nebo `\scaleput` v prostředí `picture`.

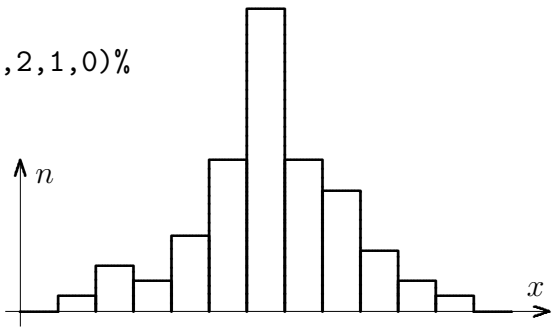
`\barq(seznam výšek obdélníků)` **Histogram**

Seznam obsahuje četnosti, příp. relativní četnosti jako reálná čísla navzájem oddělená čárkami, uspořádaná od nejnižší třídy (s indexem 1) do nejvyšší. Index třídy se generuje automaticky a s ohledem na ekvidistantnost mezí tříd ho lze použít jako nezávisle proměnnou. Skutečnou šířku třídy na vlastním diagramu pak upravíme pomocí měřítka osy x, stejně jako jeho výšku vyvoláním makra `\setscales`.
Příklad:

Má se zobrazit histogram o četnostech $n_i = 0, 1, 3, 2, 5, 10, 20, 12, 8, 4, 2, 1, 0$

```
\unitlength1mm
\begin{picture}(160,20)(-90,-10)
  \scaleput(0,0)%
   {\XYaxes(-2,60){$x$}(-2,30){$n$}%
    \setscales(5,0,0,2)%
    \linethickness{.8pt}%
    \barq(0,1,3,2,5,10,20,10,8,4,2,1,0)%
   }
\end{picture}
```

Výsledek tohoto kódu je následující:



Obecný sloupcový diagram

Nejsou-li šířky subintervalů stejně velké, nelze informaci vynést pomocí makra `\barq`. Může jít o případ histogramů s nestejně širokými třídami, ale obecně i o diagram, v němž mírou velikosti je obsah obdélníka. Takové diagramy získáme pomocí makra

`\barxy(seznam)` **Obecný sloupcový diagram**

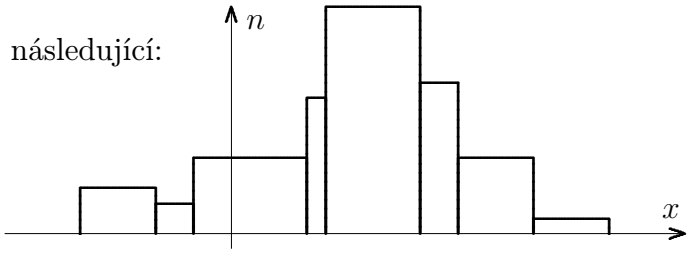
Jedná se o zobecněnou verzi předchozího histogramu s jediným parametrem:

`seznam` obsahuje souřadnice `x,y` levých horních rohů obdélníků vytvářejících sloupcový diagram. Všechny souřadnice jsou reálnými položkami, tedy reálnými čísly udávajícími polohu bodů na kreslicí ploše v jednotkách `\unitlength`. Jsou navzájem odděleny čárkami.

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-15)
 \scaleput(0,0)%
   {\XYaxes(-30,60){$x$}(-2,30){$n$}%
    \setscales(5,0,0,2)
    \linethickness{.8pt}
    \barxy(-4,0,-4,3,-2,2,-1,5,2,9,2.5,15,5,10,6,5,8,1,10,0)
   }
\end{picture}
```

Výsledek tohoto kódu je následující:



Kruhový diagram

Jiným typem diagramu, užívaným zejména pro názorné zobrazování poměrů dílčích jevů k celku, je kruhový diagram označovaný hovorově jako „koláč“. Celek tvoří kruh rozdělený na výseče s plochou úměrnou zastoupení elementárního jevu v celku. Tyto diagramy jsou velmi populární pro svoji názornost, a proto jsou často užívány k zobrazování výsledků průzkumů a pod. K tomuto účelu je k dispozici makro

`\pie{rdiam}(seznam)` **Kruhový diagram**

Parametry makra jsou:

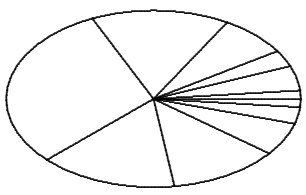
`rdiam` je reálný průměr kružnice ohraničující diagram jako násobek `\unitlength`

`seznam` je seznam reálných položek oddělených čárkami, představujících v součtu celek, který obsadí plochu celého kruhu. Poměr hodnoty položky k součtu hodnot všech položek se vyjádří stejným podílem úhlu kruhové výseče k plnému úhlu.

V příkladu jsou zobrazeny poměry odpovídající stejným datům, jaké byly užity pro ukázku histogramu:

Příklad:

```
\unitlength1mm
\begin{picture}(40,40)(-20,-15)
  \setscales(1,0,0,.6)
  \scaleput(0,0)%
    {\pie{39}(1,3,2,5,10,20,10,8,4,2,1)}
\end{picture}
```



Kapitola 7

Podpora textového módu

Pro sázení odborných textů lze s výhodou využít předem připravených postupů, které jsou zabudovány do souboru `user.sty`. Ten obsahuje řadu drobných maker, která uvolňují autora od programátorské práce ve prospěch propracování dokumentu. Toto je i důvod, proč se zde uvádí stručný popis maker zřazených do tohoto souboru. Makra z `user.sty` lze rozdělit do řady skupin:

- tisky v režimu `\tt`
- úpravy dokumentu
- prvky matematických výrazů
- boxy a prostředí
- celočíselná aritmetika (duplikace s `rplot.sty`)
- terminálové vstupy a výstupy (duplikace s `rplot.sty`)

Může vzniknout otázka, proč se poslední dvě skupiny maker, známé již z popisu souboru `rplot.sty`, objevují i zde. Důvod je jednoduchý – bylo by zbytečné zavádět tento velký soubor, pokud se vystačí s menším. Dvojímu zavedení maker je však zabráněno.

7.1 Tisky písmem typu \tt

Písmo typu pica se v LaTeXu označuje příkazem `\tt` (**t**ele**t**ype = psací stroj). Obvykle se používá pro zobrazení textů a jejich částí, jak jsou uloženy v paměti, nebo jak se vkládají do počítače. Protože pro zpracování manuálů k programovému vybavení je zapotřebí opakovaně sázet texty tohoto typu, zpracovala se pro tyto práce speciální makra.

Všechna jména maker této skupiny (až na malé výjimky) začínají symbolem `T`. Rovněž všechna makra mohou tisknout tímto typem v různé velikosti symbolů. Ta se nastavuje příkazem

```
\Cmd \fsize=\noexpand velikost,
```

kde `velikost` je příkazové vyjádření velikosti fontů. Při deklaraci velikosti písma dokumentu v `documentstyle[ ...,12pt, ...]`, je sice možno zadat všechny velikosti okamžitého fontu, avšak pouze velikosti

```
\LARGE, \Large, \large, \normalsize, \footnotesize, a \scriptsize
```

se zobrazí s fonty `\tt`, kdežto ostatní s fonty typu `\rm - roman`.

`\T{text}`	**Výstup textu písmem \tt**

`\Tc{text}` **Centrovaný výstup textu písmem \tt**

`\Tb{text}` **Orámovaný výstup textu písmem \tt**

`\Tcb{text}` **Centrovaný orámovaný výstup textu písmem \tt**

`\Tcmdh{jméno}{nadpis}` **Hlavička makra do manuálu**

`\Tx{xtxt}_{ytxt}` **Text s indexem písmem \tt**

`\Tp{xtxt}^{ytxt}` **Text s mocninou písmem \tt**

Do této skupiny maker zařaďme i makra pro snazší výstup složitějších výrazů. Patří sem různé závorky a znaky

`\bs` **Zpětné lomítko** \

`\{` **Levá složená závorka** {

`\}` **Pravá složená závorka** }

`\{text}` **Text ve složených závorkách** {text}

`\<text>` **Text v úhlových závorkách** ⟨text⟩

`\"text"` **Zápis příkazu** \text

Je pravdou, že předefinováním některých příkazů jsme se zbavili některých méně obvyklých maker, která nám však pravděpodobně nikdy nebudou chybět. Pokud ano, musel by se uživatel vrátit k novému předefinování na původní význam makra. Tato pravděpodobnost patrně není příliš vysoká.

7.2 Úpravy dokumentů

Do této skupiny jsou zařazena makra, kterými lze záměrně upravovat dokument přidávanými znaky, čarami a prokládanými horizontálními nebo vertikálními mezerami.

`\s` **Bezrozměrný textový objekt – strut**

Jde o velmi užitečný objekt, protože řada příkazů LaTeXu se potřebuje „opřít“ o nějaký text. Pokud takový text nemáme, zastoupí ho právě „strut“, který fakticky představuje čtvereček o nulové straně. Ten sice není vidět, ale úlohu „zarážky“ může plnit.

Příklad: `\s\hfill\dotfill\hfill\s` .....

`\odraz{hsp}` **Horizontální mezera**

`\pskip{hsp}` **Horizontální mezera**

Jde o tvrdou mezeru šířky `hsp`, kde `hsp` je reálné číslo bez rozměru, který je implicitně `[ex]`. Tím se odstraňují veškeré problémy s velikostí písma. Jeho použití je možné kdekoliv. Zejména je výhodné pro odsazování odstavců, pokud `\parindent = 0pt`, protože funguje i na začátku řádky.

`\lines{vsp}` **Vertikální mezera**

Jde o tvrdou vertikální mezeru výšky `vsp`, která je reálným bezrozměrným číslem (nikoliv nutně kladným), vyjadřujícím vertikální odstup základních čar textu před a po použití tohoto makra v jednotkách `\baselineskip`, tj. v roztečích řádek dokumentu. Po vyvolání začíná text na levém okraji textové plochy. Při nulovém `vsp` se nastaví počátek řádky, ze které bylo makro vyvoláno, a další text ji bude přepisovat. Zápornou mezeru lze užít ke zmenšování nadbytečných prázdných ploch.

`\xline` **Horizontální čára**

Příkaz udělá v dokumentu horizontální čáru bez odsazení od daného místa stránky (nebo ministránky v prostředí `minipage`) do konce dané řádky.

Příklad: `\xline`

`\centered{textový objekt}{vsp}` **Vystředění objektu na řádce**

Na rozdíl od prostředí `center`, které plní podobnou funkci, toto makro umožňuje uživateli velice operativně měnit vertikální odlehlost objektu od okolního textu pomocí parametru `vsp`. Ten je bezrozměrným reálným číslem udávajícím velikost vertikálních mezer jako násobek základní rozteče řádek.

Příklad: `\centered{\bf Zkušební text}{2}`

Zkušební text

7.2.1 Odsazování

V úvodu kapitoly bylo uvedeno makro pro vložení horizontální mezery `\odraz`. Jde vlastně o nejjednodušší odsazení textu od předcházejícího. S tímto prostředkem se však často nevystačí. Proto jsou dále uvedena makra pro odsazování odstavců i vnořování odsazených pododstavců do sebe. S touto úlohou jsou spojena jednak makra ze sbírek maker, které lze ve světě nalézt, jednak vlastní.

Základní charakteristikou odstavce je, že je oddělen od okolního textu prázdnou řádkou. Z toho plyne, že uvnitř textu, který chceme odsazovat **nesmí** být prázdná řádka,

protože pak by šlo o dva odstavce. Pokud chceme jedním příkazem odsazovat více odstavců, nesmíme mezi ně vložit prázdnou řádku, ale příkaz `\lines` s příslušnou vzdáleností odstupu další řádky.

`\hanghere` **Odsazování od aktuálního místa**

Jde o makro převzaté, které je jediným makrem stejnojmenného stylu, a natahuje se z adresáře stylů. Jeho funkce je jednoduchá: V okamžiku vyvolání makra se zapamatuje horizontální pozice na stránce a další řádky začínají až v této pozici. Lze ho proto použít až na základě předběžné znalosti výsledku.

Příklad:

```
Zde je příklad použití makra
\T{\bs hanghere}. Tento
\hanghere text je již od
další řádky odsazován, jak
se dalo z~popisu předpokládat.
```

Zde je příklad použití makra \hanghere. Tento text je již od další řádky odsazován, jak se dalo z popisu předpokládat.

`\begin{hanging}` **Začátek prostředí pro odsazování**

`\end{hanging}` **Konec prostředí pro odsazování**

Toto prostředí odsune text odstavce od druhé řádky o požadovanou vzdálenost doprava. Kromě toho oddělí odstavec od ostatního předcházejícího i následujícího textu o vertikální mezeru `\parskip`. Původní verze tohoto prostředí odsouvala text odstavce od druhé řádky o vzdálenost `\parindent`, anebo o `1.5em`, pokud byla hodnota v `\parindent` nulová. Oba parametry lze definovat v preambuli mezi `documentstyle` a prostředím `document` např. příkazem `\parskip=1.5ex`.

Nové prostředí `hanging` bylo zásadně přepracováno tak, aby umožňovalo odsazovat odstavce o libovolnou reálnou vzdálenost nezávisle na parametru `\parindent` bez jakékoliv další alternativy. Pro to je však zapotřebí předem nastavit velikost odsazování dále uvedeným příkazem `\sethang`. Materiál k odsazování je vložen do prostředí se standardními „závorkami“ `begin ... end`.

Příklad:

```
  Toto je text předcházejícího
  odstavce
  \sethang{3}
  \begin{hanging}
    Toto je vložený odstavec,
    který bude odsazován od
    své druhé řádky.
  \end{hanging}
Toto je následující odstavec.
Viz meziodstavcové mezery!
```

Toto je text předcházejícího odstavce.

Toto je vložený odstavec, který bude odsazován od své druhé řádky.

Toto je následující odstavec. Viz meziodstavcové mezery!

`\sethang{hang}` **Nastavení velikosti odsazení**

Parametr `hang` je bezrozměrné reálné číslo, kterému se uvnitř makra dodá rozměr `[ex]` a uloží se do délkového registru pro odsazování.

`\hangit{hang}` **Zahájení odsazování na nové řádce**

Vyvolání makra způsobí odsazování pokračujícího textu na nové řádce s odstupem rovným `hang` délek `1ex`. Pokud je `hang` kladné, bude odsazení směrem vpravo, pro záporné vlevo, ale s přídavnou vertikální mezerou, jak je ukázáno v příkladu. Obvykle tato nesymetrie vadí a je nutné ji kompenzovat příkazem `\lines`.

Příklad:

```
Text předcházejícího odstavce.
\hangit{5}
Text odsazovaného odstavce.
Text je odsazen od 1. řádky.
\hangit{-5}
Text následujícího odstavce.
```

Text předcházejícího odstavce.

Text odsazovaného odstavce. Text je odsazen od 1. řádky.

Text následujícího odstavce.

`\Hp{hang}{text}` **Odsazení odstavce od 2. řádky**

Jde o odsazování běžné u textů, u nichž má vyniknout začátek odstavce, jako je tomu například u výčtů položek. Význam parametru `hang` je stejný jako u výše uvedených maker. Vertikální odstupy odstavce `text` od okolního textu jsou standardní, tedy jako u normálních odstavců.

Příklad:

```
Toto je text pro makro \bs Hp:
\Hp{2}{První řádka. \lines1
       Druhá řádka. \lines1
       Třetí řádka.
      }%
Pokračující text%
```

Toto je text pro makro \Hp:

První řádka.
Druhá řádka.
Třetí řádka.

Pokračující text

`\Hpar{hang}{text}` **Odsazení celého odstavce**

Odstavec se odsadí již od první řádky. Parametry makra mají stejný význam, jako u předcházejících maker. Zahnízděním příkazu `\Hpar` lze dosáhnout i vnořeného odsazování.

Příklad:

```
Toto je text pro makro \bs Hpar:
\Hpar{2}{První řádka.
   \Hpar{3}{Druhá řádka.
        \Hpar{5}{Třetí řádka.}
        }  }
Pokračující text%
```

Toto je text pro makro \Hpar:

První řádka.
Druhá řádka.
Třetí řádka.

Pokračující text

7.3 Prvky matematických výrazů

Mohlo by se zdát, že vybavenost TEXu a LATEXu pro psaní matematického materiálu je tak bohatá, že je zbytečné cokoliv k tomu dodávat. Avšak při pokusu o zápis dokumentu s větším objemem matematických formulí se zjistí, že pro častěji se opakující konstrukce je výhodné vypracovat makra, jimiž se později ušetří dosti času při sázení rukopisu. Hlavní důvod spočívá v možnosti předem vyvážit požadovanou konstrukci a toto vyvážení udržovat v celém dokumentu.

Je vhodné zavést do označování veličin jistý systém. Byl učiněn pokus o jakousi systemizaci značení, která by co možná vyhovovala zavedeným zvyklostem a normám a přitom nebyla uživatelsky nepřátelská. Předložená systemizace spočívá na následujících zásadách psaní veličin:

skaláry	matematická kurzíva	$a,\ \sigma,\ z_i,\ B, ...$
náhodné veličiny	malá bezpatková písmena	x , $\mathsf{y}(t)$, ...
vektory	malá tučná matematická kurzíva	$\boldsymbol{q},\ \boldsymbol{f}(t)$, ...
matice	velká tučná matematická kurziva	$\boldsymbol{K},\ \boldsymbol{\Gamma}$, ...
operátory	velká bezpatková písmena	$\mathsf{A}\{\mathsf{x}(t)\},\ \mathsf{E}\{\mathring{\boldsymbol{x}}(t)\}$, ...
jevy	velké kaligrafické písmo	$\mathcal{A},\ \mathcal{Z},\ ...$

Makra této skupiny lze opět rozdělit do následujících podskupin :

- příznaky
- symboly a operátory
- limity, sumy a integrály
- vektory a matice
- reference

Při tvorbě těchto maker se ukázalo, že některá jména maker kolidují s již definovanými jmény. Pokud implicitní jména nevyhovovala a patřila ke konstrukcím, které se prakticky nevyskytují, byla zde předefinována.

7.3.1 Příznaky

Byl vlastně nadefinován jediný nový příznak – kroužek. Protože příkaz `\o` byl již obsazen (švédským kroužkovaným a), použila se jeho redefinice.

`\o` **Kroužek nad symbolem**

Tento příznak se používá v matematickém režimu v případech, že se chce zdůraznit cykličnost – periodicita – jevu označeného zvoleným symbolem. Ten se pak užil v dalších makrech:

`\orv` **Kroužek nad symbolem náhodné proměnné**

V posloupnosti hodnot náhodné proměnné se nevyskytuje periodicita. Při číslicovém zpracování časových řad se však rozděluje celá realizace na úseky, které se zpracovávají odděleně a které při zpracování diskrétní konečnou Fourierovou transformací vytvářejí vně základního intervalu periodické kopie. Akcent pak zdůrazňuje toto zperiodizování.

`\ot` **Kroužek nad funkcí času**

Důvod pro jeho zavedení je obdobný jako u náhodné proměnné – zpřehledňuje zápis a vlastnosti symbolů s kroužkem.

Příklady:

```
$\o{z_i}$  \\
$\orv{z}$  \\
$\ot{z}$
```

$\mathring{z}_i$
$\mathring{\mathsf{z}}$
$\mathring{z}(t)$

7.3.2 Symboly a operátory

O příkazech pro symboly a operátory platí totéž, co bylo řečeno výše, totiž, že bylo nutno některé zavedené příkazy LaTeXu předefinovat.

`\rv{znaky}` **Náhodná proměnná**

Náhodná proměnná a její závislosti na čase – náhodné procesy – se natolik odlišují od běžných veličin, že si zaslouží zvláštní typ písma. Je jím typ sans serif – bezpatkové písmo, jímž budou vysázeny znaky uvedené v argumentu makra.

Příklad: `náhodná proměnná \rv{r} ...` náhodná proměnná r ...

`\delt` **Diracův impuls**

Jde o symbol často se vyskytující v nejrůznějších oblastech vědy. Makro ho umí zobrazit pouze v základní verzi s argumenten t v matematickém režimu.

Příklad: `Diracův impuls $\delt$ ...` Diracův impuls $\delta(t)$ ...

`\dg` **Znak parciální derivace**

Zkracuje zápis jinak definovanáho povelu `\partial`. To se projeví v přehlednosti formulí, v nichž se vyskytuje mnoho těchto symbolů.

Příklad: `$\frac{\dg y}{\dg x}$ ...` $\frac{\partial y}{\partial x}$

`\re` **Reálná část komlexní veličiny**

`\im` **Imaginární část komlexní veličiny**

Oba povely jsou již definovány s úvodními písmeny velkými, pro něž však dostáváme příslušné operátory ve tvaru gotických písmen Re , Im . Ty lze použít v matematickén režimu. Protože u nás jsou obvyklejší označení Re a Im zapsaná normálním písmem, vytvořily se tyto povely aplikovatelné jak v matematickém, tak i obyčejném režimu.

Příklad: `Je-li $\im \mx{G}(\omega)$ ...` Je-li Im $\boldsymbol{G}(\omega)$ ...

`\P` **Pravděpodobnost**

`\p`

Oba povely byly použity pro stejné cílové symboly pravděpodovnosti výskytu nějakého jevu. Povel s velkým písmenem byl předefinován z celkem bezvýznamného symbolu. Symbol pravděpodobnosti vystupuje jako velké bezpatkové písmeno P, jak je obvyklé.

Příklady: `$\p{\cal{A}}\le\P{\cal{B}}$` $\mathsf{P}\{\mathcal{A}\} \le \mathsf{P}\{\mathcal{B}\}$

`\Ao{arg}` **Aplikace operátoru na argument**

Používá se např. u ergodických procesů pro vyjadřování časových průměrů. V příkladu je pro zkrácení zápisu použito makro `\limTi`, které bude popsáno v odstavci věnovaném limitám.

Příklad: `$\Ao{f(t)} = \limTi{f(t)}$` $\mathsf{A}\{f(t)\} = \lim_{T\to\infty} \frac{1}{T} \int_0^T f(t)\,dt$

`\Ei{arg}{hust}{rv}` **Střední hodnota funkce náhodné veličiny**

`\Eo{arg}` **Aplikace operátoru střední hodnoty na argument**

Operátor E se široce využívá v literatuře věnované náhodným veličinám a procesům. Parametry maker jsou:

`rv` náhodná proměnná
`arg` náhodná proměnná `rv` nebo její funkce
`hust` hustota pravděpodobnosti náhodné proměnné `rv`

Příklad: `$\Eo{x(t)} = \Ei{x(t)}{f(x,t)}{x}$` $\mathsf{E}\{x(t)\} = \int_{-\infty}^{\infty} x\,(t)\, f(x,t)\,dx$

`\D{arg}` **Aplikace operátoru rozptylu na argument**

Je-li argumentem `arg` v předchozích makrech kvadrát centrované náhodné proměnné rv, nazývá se výsledek rozptylem – disperzí (proto i symbol operátoru je D). Centrování se zajistí odečtením střední hodnoty μ_{rv} od rv.

Příklad: `$\Do{\rv{x}}=\Eo{(\rv{x}-\mu_\rv{x})^2}$` $\mathsf{D}\{\mathsf{x}\} = \mathsf{E}\{(\mathsf{x} - \mu_{\mathsf{x}})^2\}$

`\Fpope` **Operátor Fourierova páru funkcí**

`\Fpair` **Fourierův pár funkcí**

Obě makra zajišťují potřebné nástroje při zápisech Fourierovy transformace. První z maker, `\Fpope`, je symbolem vyjadřujícím ekvivalenci mezi originálem a obrazem ve smyslu Fourierovy transformace. Zastupuje vlastně rovnítko a předpis pro Fourierovu transformaci. Makro `\Fpair` využívá `\Fpope` k vytvoření symbolického zápisu Fourierovy transformace funkce.

Příklad:

```
\Fpope                          $\Fpope$
\Fpair{f(t)}{F(\omega)}         $\Fpair{f(t)}{F(\omega)}$
```

$\overset{\mathsf{F}}{\Longleftrightarrow}$

$f(t) \overset{\mathsf{F}}{\Longleftrightarrow} F(\omega)$

`\FT{funt}` **Operátor Fourierovy transformace**

`\IFT{funf}` **Operátor inverzní Fourierovy transformace**

Operátory slouží k symbolickému zápisu Fourierovy transformace, resp. její inverzní varianty.

Příklad:

```
$\FT{h(t)}  =  H(\omega)$        $\FT{h(t)} = H(\omega)$
$\IFT{H(\omega)}  =  h(t)$       $\IFT{H(\omega)} = h(t)$
```

$\mathsf{F}\{\, h(t) \,\} = H(\omega)$

$\mathsf{F}^{-1}\{\, H(\omega) \,\} = h(t)$

7.3.3 Limity, sumy a integrály

LaTeX poskytuje pouze základní makra pro všechny tři podskupiny tohoto odstavce. Pro limity dává k dispozici povel `\lim` který vypíše normálním písmem slabiku `lim`. Podobná je situace u sum, kde na povel `\sum` vystoupí symbol sumy $\sum$, a u integrálů, kde zápis `\int` způsobí výstup znaku $\int$. Vše ostatní je na uživateli, jak s danými možnostmi naloží. Proto byla pro častěji se opakující konstrukce sestavena makra:

`\limg{var}{mez}{expr}` **Obecná limita výrazu**

Makro slouží pro snadný zápis obecné limity výrazu `expr`. Parametr `var` je jméno proměnné, podle které se limituje a `mez` je hodnota, kterou má `var` dosáhnout.

Příklad:

```
$\limg{x}{0}{\frac{\sin{x}}{x}}$
```

$\limg{x}{0}{\frac{\sin{x}}{x}}$

`\limi{var}` **Limita pro var $\to +\infty$**

Příklad:

```
$\limi{x} \arctan{x}$
```

$\limi{x} \arctan{x}$

`\limT{var}` **Limita pro T $\to +\infty$**

`\limTi{var}` **Limita nevlastního integrálu s mezemi $\langle$0, T$\rangle$**

`\limTih{var}` **Limita nevlastního integrálu s mezemi $\langle -$T/2, T/2$\rangle$**

Příklady:

```
$\limT$
$\limTi{f(t)}$
$\limTih{g(t)}$
```

$\limT$

$\limTi{f(t)}$

$\limTih{g(t)}$

`\limTone{var}` **Limita pro $\mathbf{T}_1 \to +\infty$**

`\limTonei{var}` **Limita nevlastního integrálu s mezemi $\langle \mathbf{0},\, \mathbf{T}_1 \rangle$**

`\limToneih{var}` **Limita nevlastního integrálu s mezemi $\langle -\mathbf{T}_1/\mathbf{2},\, \mathbf{T}_1/\mathbf{2} \rangle$**

Tato tři makra jsou formálně stejná jako předcházející, až na proměnnou, podle které se limituje. Zatímco `T` představovala obecnou proměnnou, $\mathtt{T}_1$ je perioda první (základní) frekvenční složky ve Fourierově analýze.

Příklady:

`$\limTone$` $\lim_{T_1\to\infty}$

`$\limTonei{f(t)}$` $\lim_{T_1\to\infty} \frac{1}{T_1} \int_0^{T_1} f(t)\,dt$

`$\limToneih{g(t)}$` $\lim_{T_1\to\infty} \frac{1}{T_1} \int_{-T_1/2}^{T_1/2} g(t)\,dt$

`\sumg{bot}{top}{elem}` **Obecná suma prvků**

`\sumii{idx}{elem}` **Oboustranně nekonečná suma prvků**

První makro je zcela obecné pro libovolný index a hodnotu na jeho spodní mezi `bot` i na horní mezi `top` a sčítané prvky `elem`. Druhé z maker již vysází sumu se žádaným sčítacím indexem `idx` probíhajícím od $-\infty$ do $+\infty$ přes všechny elementy.

Příklady:

```
$$\sumg{n=0}{\infty}{a_n\,\cos{2\pi nft}}$$
```
$$\sum_{n=0}^{\infty} a_n \cos 2\pi n f t$$

```
$$\sumii{k}{b_k\,\sin{2\pi kft}}$$
```
$$\sum_{k=-\infty}^{\infty} b_k \sin 2\pi k f t$$

`\intg{bot}{top}{fun}{dvar}` **Obecný integrál**

`\intii{fun}{dvar}` **Nevlastní integrál na obou mezích**

První makro je zcela obecné pro libovolné meze `bot` , `top`, integrand `fun` i integrační proměnnou charakterizovanou jejím diferenciálem `dvar`. Druhé makro vypouští první dva pararametry – meze, protože ty jsou nevlastní.

Příklady:

```
$$\intg{0}{\infty}{g(t)\,\cos{2\pi nft}}{dt}$$
```
$$\int_0^{\infty} g(t) \cos 2\pi n f t\,dt$$

```
$$\intii{y(x)\,\sin{2\pi kfx}}{dx}$$
```
$$\int_{-\infty}^{\infty} y(x) \sin 2\pi k f x\,dx$$

7.3.4 Vektory a matice

Matice je obdélníkové schéma – tabulka – prvků uspořádaných do `m` řádek a `n` sloupců. Pak mluvíme o matici *typu* `m,n`. Pokud je počet řádek a slouců totožný jde o matici *řádu* `n`. V případě, že `m=1`, vzniká speciální typ matice – vektor-řádka *dimenze* `n`. Pokud bude `n=1` mluví se o vektoru-sloupci dimenze `m`. Protože jde o často se vyskytující objekty, zpracovala se následující makra:

`\mx{písmeno}` **Symbol matice nebo vektoru**

Makro slouží k vysázení symbolu matice nebo vektoru v matematickém i textovém módu. U matic se obvykle používají velká písmena latinské nebo řecké abecedy, u vektorů malá. V obou případech se symboly vysázejí správně v tučné matematické kurzívě.

`\dmx{písmeno}` **Symbol první derivace matice nebo vektoru**

`\ddmx{písmeno}` **Symbol druhé derivace matice nebo vektoru**

Derivace vektoru podle času se často vyskytuje v odborných textech. Tečkou nad symbolem je vyjádřena první derivace a dvěma tečkami druhá.

Příklady:

```
$$ \mx{A}\,\mx{x} = \mx{b} $$
```
$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$$

```
$$ \dmx{x}=\mx{A}\,\mx{x}+\mx{b} $$
```
$$\dot{\boldsymbol{x}} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{b}$$

```
$$ \ddmx{q}+\dmx{q}+\mx{q}=\mx{0} $$
```
$$\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}} + \boldsymbol{q} = \boldsymbol{0}$$

`\mtx{písmeno}{m}{n}` **Obecná matice rozepsaná po prvcích**

Makro je vhodné pro zápis obecné matice označené parametrem `písmeno` po prvcích. Má-li se použít i pro konkrétní číselné hodnoty, potom musí platit, že počet řádek `m` i sloupců `n` musí být větší než 3. Pokud je jménem matice velké písmeno latinské abecedy, vystoupí prvky malými písmeny. To bohužel neplatí pro řeckou abecedu, pro kterou je zapotřebí zadat symbol malý.

Příklad:

```
$$ \mx{\mathit\Gamma}=\mtx{\gamma}{m}{n} $$
```
$$\boldsymbol{\mathit{\Gamma}} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1n} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{m1} & \gamma_{m2} & \cdots & \gamma_{mn} \end{bmatrix}$$

`\vecr{písmeno}{n}` **Vektor-řádka po prvcích**

`\vecc{písmeno}{m}` **Vektor-sloupec po prvcích**

Makra jsou vhodná pro zápis vektorů po prvcích, pokud je dimenze vektoru určená druhým parametrem buď obecná, daná písmenem, anebo větší než 3.

Příklad:

```
$$ \mx{x} = \vecr{x}{n};
\quad
\mx{y} = \vecc{y}{5} $$
```

$$\boldsymbol{x} = \left[\, x_1,\ x_2,\ \cdots,\ x_n \,\right]; \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_5 \end{bmatrix}$$

`\mt{písmeno}` **Matice jako funkce času v textu**

`\ms{písmeno}{idx}{exp}` **Indexovaná matice v textu**

Obě makra mají usnadnit a zpřehlednit psaní složitějších maticových výrazů v textovém módu.

Příklad:

```
... vektor \mt{q} je ...
... mocnina matice \ms{A}{i}{k} ...
```

... vektor $\boldsymbol{q}\,(t)$ je ...
... mocnina matice $\boldsymbol{A}_i^k$ ...

`\msx{S11}{S12}{S21}{S22}` **Matice rozdělená na čtyři submatice**

`\vsx{S11}{S21}` **Sloupcový vektor rozdělený na dva subvektory**

Práce s maticemi rozdělenými na submatice je dosti častým jevem. Makra zpracovávají pouze případ, v němž dělení na formální poloviny se děje v každé dimenzi. Parametry `Sij` jsou maticové výrazy – submatice. V makru `\msx`, kde se předpokládá, že se ve výrazech nevyskytují derivace matic, lze volání makra `\mx` u každé matice vynechat. To nelze u makra `\vsx`.

Příklad:

```
$$ \vsx{\dmx{x}}{\ddmx{x}} = \msx{O}{I}{-M^{-1}K}{-M^{-1}B}\,
   \vsx{\mx{x}}{\dmx{x}} + \vsx{\mx{0}}{\mx{M}^{-1}\mx{f}(t)}$$
```

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \ddot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{O} & , & \boldsymbol{I} \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & , & -\boldsymbol{M}^{-1}\boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1}\boldsymbol{f}(t) \end{bmatrix}$$

7.3.5 Reference

Odvolávky na obrázky, tabulky, rovnice i literární prameny se zjednoduší následujícími makry, u nichž `label` je návěští použité u objektu.

`\obr{label}`	**Reference na obrázek**	**obr.**
`\tab{label}`	**Reference na tabulku**	**tab.**
`\rov{label}`	**Reference na rovnici**	**rov. ( )**
`\lit{label}`	**Reference na literaturu**	**lit. [ ]**

7.4 Boxy a prostředí

Boxy a prostředí představují užitečné prostředky pro formální úpravu textu. Lze jimi ovládat uspořádání posloupnosti znaků, které jsou do nich uzavřeny, ať již jde o textový nebo grafický režim. Dále uvedená makra rozšiřují možnosti uživatele.

7.4.1 Boxy

Následující dvě makra umožňují snadno vytvořit orámovaný obdélník s textem, který může zabírat i několik řádek, což příkaz `\framebox` neumí. Další makro vytvoří horizontálně centrovaný text požadované šířky.

`\mlframebox{width}{vpos}{hpos}{text}{sep}{thick}` **Víceřádkový framebox**

`\mlfbox{text}` **Víceřádkový fbox**

Jak je z příkladu vidět, normální příkaz `\framebox` neumožňuje psát a orámovat víceřádkové texty. Makro `\mlframebox` toto dokáže s možností plného řízení vzhledu výsledku. Význam parametrů volání je následující:

`width`	šířka rámečku v jednotkách, které jsou součástí `width`
`vpos`	vertikální poloha textu: `t, b`, (`c` implicitně)
`hpos`	horizontální poloha textu: `l, r`, (`c` implicitně)
`text`	libovolný i víceřádkový text
`sep`	vertikální odlehlost rámečku od textu (včetně rozměru!)
`thick`	tloušťka čáry rámečku s rozměrem

Jeho podstatně zjednodušená forma `\mlfbox` dokáže podobné s omezenými možnostmi řízení. Implicitními hodnotami u `\mlfbox` jsou `\fboxsep=0.5ex`, která slouží jak pro vertikální, tak i horizontální mezeru mezi textem a rámečkem, a dále `vpos=c` i `hpos=c`. Tloušťku čáry lze použít buď implicitní `\fboxrule=0.4pt`, anebo ji změnit pomocí příkazu `\setlength{\fboxrule}{tloušťka}`.

Příklad:

```
\framebox[30mm][t]{Toto je \\ víceřádkový text}%  NESPRÁVNĚ!
\mlframebox{35mm}{c}{c}{Toto je \\ víceřádkový text}{1ex}{1mm}
\mlfbox{Toto je\\text na více řádkách}
```

`\midbox{šířka}{text}` **Centrování textu dané šířky**

Úlohou makra je horizontální vystředění textu požadované šířky na stránce. Protože se nepoužilo prostředí `center`, není text vertikálně odsazen od okolí.

Příklad:

```
\midbox{80mm}%
{Parametr \T{šířka} musí mít rozměr!
 Pokud se rozměr opomene, je hlášena
 chyba z~\LaTeX u.
}
```

Parametr `šířka` musí mít rozměr! Pokud se rozměr opomene, je hlášena chyba z LaTeXu.

7.4.2 Náhrady standardních prostředí

V této podkapitole budou popsány náhrady prostředí LaTeXu příkazy, jejichž volání je jednodušší. Kromě jednoduchosti lze i požadovat speciální účinky rozdílné od vyvolání standardního prostředí.

Rovnice a jejich soustavy

jsou velmi často se opakujícím objektem v odborném textu. Proto LaTeX obsahuje prostředí `\equation` a `\eqnarray`. Pro zjednodušení zadávání nebo pro speciální účely byla sestavena následující makra, z nichž první dvě pouze převádějí prostředí na příkaz. Další dvě jsou určena pro stejné účely, avšak bez číslování rovnic.

`\eqn{rovnice}{návěští}` **Náhrada prostředí equation**

Makro funguje jako prostředí `equation`, protože ho samo vyvolává. Při vyvolání použije první parametr `rovnice` jako tělo prostředí `equation`. Vzápětí přiřadí této rovnici číslo, na které se lze později odvolávat příkazem `\ref{návěští}` nebo `\rov{návěští}` jako u normálního prostředí `equation`.

```
\eqn{a\,x^2+b\,x+c=0}{r1}
... viz \rov{r1}, kde ...
```

$$a\,x^2 + b\,x + c = 0 \quad (7.1)$$

... viz rov. (7.1), kde ...

`\eqnarr{rovnice}{návěští}` **Náhrada prostředí eqnarray**

Makro funguje jako prostředí `eqnarray`. Rozdíl je však v tom, že celé soustavě rovnic se přidělí pouze jedno číslo, a to se umístí za systém rovnic doprostřed vertikálního rozpětí. K tomu však je zapotřebí, aby i poslední rovnice byla zakončena symbolem nové řádky `\\`. Je zde ještě třeba upozornit, že obě makra, stejně jako jejich standardní

vzory z LaTeXu, si sama vytvářejí matematický mód. Pokud bychom je uzavřeli do libovolného matematického prostředí, bude hlášena chyba.

Příklady:

```
\eqnarr{a\,x + b\,y & = & e   \\
        c\,x + d\,y & = & f   \\
       }{r2}
```

$$\begin{aligned} a\,x + b\,y &= e \\ c\,x + d\,y &= f \end{aligned} \tag{7.2}$$

`\equ{rovnice}` **Nečíslovaná rovnice**

Makro vypíše parametr `rovnice` do středu nové řádky odsazené od okolí vertikální mezerou v prostředí `math = $ ··· $`, které si samo nadefinuje. Z tohoto důvodu dochází k zúženému výstupu indexů vedle symbolů sum a integrálů. Pokud tento rys vadí, lze použít dále uvedené makro `\eqa` s jedinou rovnicí, kterou lze zapsat normálně jako v makru `\equ`.

`\eqa{rovnice}` **Nečíslovaný systém rovnic**

Makro je určeno pro jednoduché zadávání systémů rovnic jako pro prostředí `eqnarray*`. Proti tomuto prostředí má jednodušší zadávání a na výstupu menší odstup od okolí. Stejně jako `eqnarray` pracuje v matematickém módu v prostředí `displaymath`, ve kterém jsou řádky zobrazovány s obvyklým odstupem indexů od symbolů.

Příklad:

```
\equ{\mu_x=\limTonei{x(t)}}
%
%
\eqa{\mu_x=\limTonei{x(t)}}
%
%
\eqa{x+y & = & 0     \\
       x & = & \sin{x}
     }
```

$$\mu_x = \lim_{T_1\to\infty} \frac{1}{T_1} \int_0^{T_1} x(t)\,dt$$

$$\mu_x = \lim_{T_1\to\infty} \frac{1}{T_1} \int_0^{T_1} x(t)\,dt$$

$$\begin{aligned} x + y &= 0 \\ x &= \sin x \end{aligned}$$

Následující makra jsou věnována jednak dosti častému požadavku na horizontální vystředění tabulky, jednak požadavku na vysázení výčtu položek v tabulce doražené v textu vpravo na stránce.

`\ctabular{pos}{cols}{tělo}` **Vystředění tabulky**

`\ctabularf{pos}{cols}{tělo}{posf}{capt}{label}` **Vystředění tabulky**

Zatímco první makro vystředí tabulku ve výstupním textu v místě, kde je uvedena, druhé makro ji sice také vystředí, avšak jako plovoucí objekt (float) `table`. To znamená, že tabulka vystoupí v tomto případě v místě stanoveném LaTeXem jako optimálním. Jednotlivé parametry mají následující význam:

`pos`	pozice referenčního bodu tabulky: `t, b, (c` implicitě)
`cols`	charakteristiky sloupců v prostředí `tabular; l, c, r, @{}, p{}`
`tělo`	vlastní obsah tabulky ve stylu `tabular`
`posf`	pozice plovoucího prostředí `table: h, t, b, p`
`capt`	popis pod tabulkou
`label`	návěští pro odvolávky

Příklad:

```
\ctabular{t}{l@{ = }l}%
  {$\mu$    & střední hodnota  \\
   $\sigma$ & směrodatná odchylka
  }
\ctabularf{t}{l@{ = }l}%
  {$\mu$    & střední hodnota  \\
   $\sigma$ & směrodatná odchylka
  }%
  {h}{První dva momenty}{T1}
```

\ctabular:
μ = střední hodnota
σ = směrodatná odchylka

\ctabularf:
μ = střední hodnota
σ = směrodatná odchylka

Tabulka 7.1: První dva momenty

`\listit{skip}{symbol}{tělo}` **Vpravo odsazený výčet**

Makro zajistí odskok výčtu o `skip ex` doprava a jeho výstup doražený k pravému okraji stránky s uživatelem zvoleným uvozovacím symbolem položek. Oproti standardnímu výčtu v prostředí `itemize` umožňuje řídit vertikální mezery mezi položkami symbolem nové řádky ve tvaru `\\[vůle]`.

Příklad:

```
\listit{3}{\char35}%
{První položka přesahující jednu
 řádku. \\[1mm]
 Druhá položka přesahující jednu
 řádku. \\[1mm]
 Třetí položka přesahující jednu
 řádku. \\
}
```

První položka přesahující jednu řádku.
Druhá položka přesahující jednu řádku.
Třetí položka přesahující jednu řádku.

Kapitola 8

Složitější příklady

Při popisu jednotlivých maker se uváděly jednoduché příklady na jejich použití. Ty sice charakterizovaly vlastnosti makra za jistých podmínek, ale nemohly ukázat jejich použití v souvislostech. Tento nedostatek by měly částečně odstranit dva složitější příklady. Jeden využívá více procedury z `rplot.sty` pro kreslení funkcí vypočtených v MATLABu a druhý pro vpisování údajů do předtištěných formulářů.

8.1 Frekvenční charakteristiky SDOF systému

Mechanický systém o jednom stupni volnosti (**S**ingle **d**egree **o**f **f**reedom) je modelem pro mnoho mechanických soustav, v jejichž frekvenčním intervalu se vyskytuje pouze jedna rezonanční frekvence. Jeho pohybová rovnice má tvar

$$m\,\ddot{y}(t) + b\,\dot{y}(t) + k\,y(t) = f(t)\ , \tag{8.1}$$

kde m je hmotnost, b je útlumový a k je tuhostní koeficient systému. Poměr Fourierových obrazů buzení $f(t)$ a jemu odpovídající odezvy $y(t)$ při nulových počátečních podmínkách se nazývá frekvenčním přenosem, který má tvar

$$G(\omega) = \frac{1}{k - \omega^2 + i\omega b} \tag{8.2}$$

Pronásobením této rovnice tuhostí k, tj. vydělením statickou tuhostí $G(0)$, přejde $G(\omega)$ na $G(\eta)$, kde

$$\frac{G(\omega)}{G(0)} = G(\eta) = \frac{1}{1 - \eta^2 + i2\eta b_p} \tag{8.3}$$

kde $\eta = \omega/\Omega$ je koeficient naladění,
$\Omega = \sqrt{k/m}$ je vlastní frekvence netlumeného systému (při $b = 0$)
$b_p = \Omega b/(2k)$ je poměrný útlum $= b/(2m\Omega)$

Závislost $G(\eta)$ na koeficientu naladění η se vypočítala v MATLABovském programu, ve kterém se vygenerovala i matice

$$\left[\ \eta,\quad \mathrm{Re}\ G(\eta),\quad \mathrm{Im}\ G(\eta),\quad |G(\eta)|,\quad \arctan\frac{\mathrm{Im}\ G(\eta)}{\mathrm{Re}\ G(\eta)}\ \right]$$

pro $\eta = 0 : 0.02 : 3$ a $b_p = 0.1$ a uložila se pomocí M-funkce `texfile.m` do souboru `sdof.dat`. Úkolem při sázení bylo přečíst tento soubor a vynést všechny frekvenční charakteristiky systému. To se zrealizovalo programem pro LaTeX:

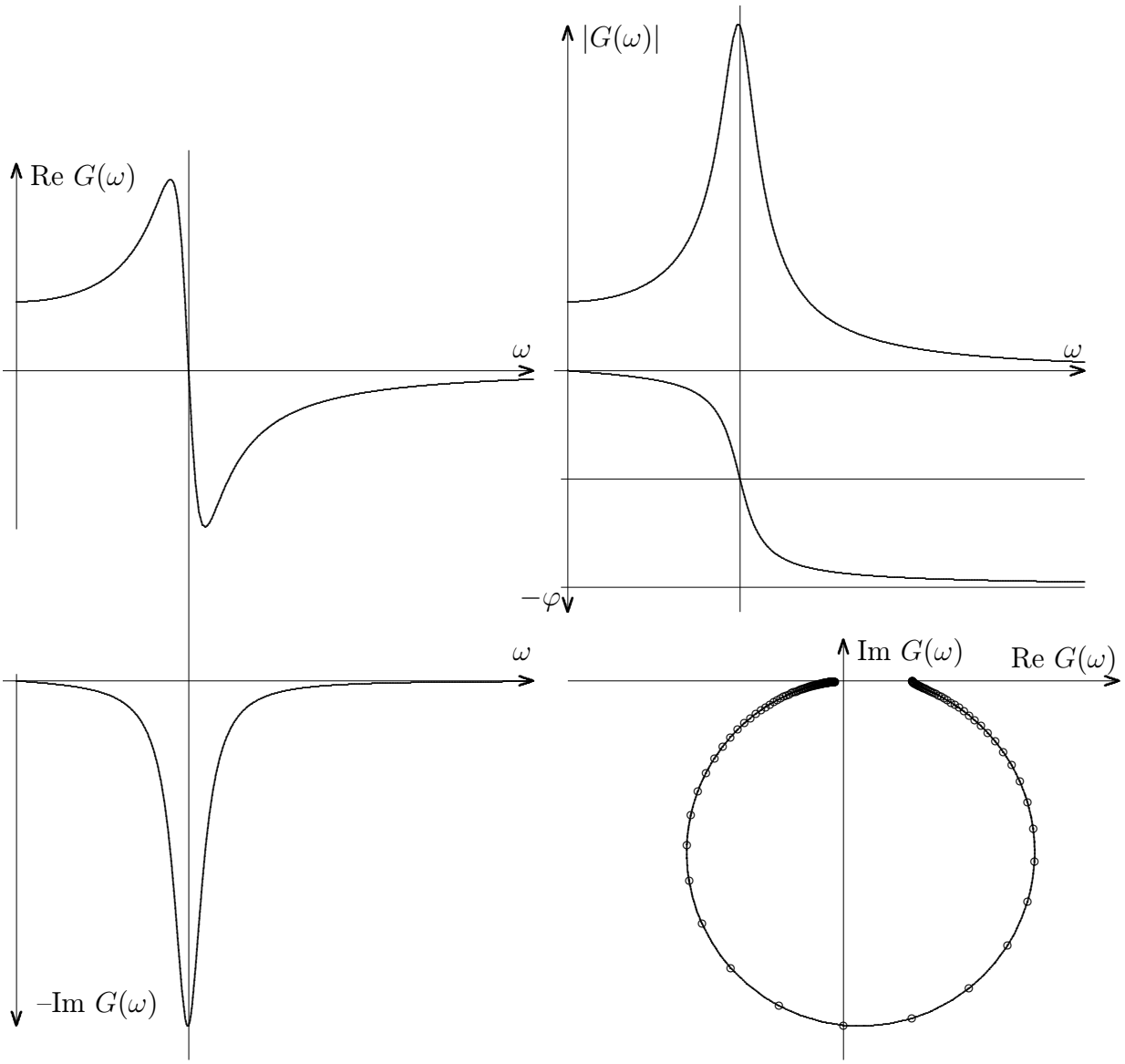
```
  \curvewarnfalse
  \newcommand{\sca}{10}

  \edef\file{sdof.d2}%      ***** FILE *****
  \loadgf \Matrix=\file%
% ~~~~~~~~~~~~~~~~~~~~~
  \unitlength1mm
  \begin{picture}(160,170)(0,-170)
    \put(0,-35)
     {\XYaxes(-2,75){$\omega$}(-23,30){Re $G(\omega)$}
      \getxy \xy=\Matrix(1,2)%      omg, Re G
      \setscales(25,0,0,\sca)%      ~~~~~~~~~
      \curve(\xy)%
     }
    \put(0,-80)
     {\Xaxis(-2,75)(72,3){$\omega$}
      \Yaxis(1,-50)(3,-48){--Im $G(\omega)$}
      \getxy \xy=\Matrix(1,3)%      omg, Im G
      \setscales(25,0,0,\sca)%      ~~~~~~~~~
      \curve(\xy)%
      \linethickness{.2pt}
      \put(25,-55){\line(0,1){132}}
      \thinlines
     }
    \put(120,-80)
     {\XYaxes(-40,40){\hskip-7ex Re $G(\omega)$}(-55,6){Im $G(\omega)$}%
      \getxy \xy=\Matrix(2,3)%     Re G, Im G
      \setscales(\sca,0,0,\sca)%   ~~~~~~~~~~
      \curves{\put(0,0){\circle{1}}}{}(\xy)%
     }
    \put(80,-35)
     {\XYaxes(-2,75){$\omega$}(0,50){$|G(\omega)|$}
      \getxy \xy=\Matrix(1,4)%     omg, abs(G)
      \setscales(25,0,0,\sca)%     ~~~~~~~~~~~
      \curve(\xy)%
      \linethickness{.2pt}
      \put(25,-35){\line(0,1){88}}
      \thinlines
     }
    \put(80,-35)
     {\Yaxis(0,-35)(-7,-34){$-\varphi$}
      \getxy \xy=\Matrix(1,5)%     omg, phi
      \setscales(25,0,0,10)%       ~~~~~~~~
      \curve(\xy)%
      \linethickness{.2pt}
      \put(-1,-15.71){\line(1,0){76}}
      \put(-1,-31.42){\line(1,0){76}}
     }
  \end{picture}
```

Na tomto místě je zapotřebí upozornit na přísnou lokálnost všech datových položek (registrů a příkazů). Tato vlastnost TEXu vede k nutnosti přečíst soubor s uloženou maticí **vně** příkazů `\put`. Pokud bychom tuto zásadu nedodrželi, byl by obsah souboru s maticí znám pouze v rámci povelů obsažených v argumentu příkazu `\put`, ale nikoliv mimo tento argument. Následkem toho musela by se matice načítat příkazem `\loadgf` v každém příkazu `\put` znovu.

Soubor `sdof.dat` měl tvar:

```
5,  151,
0.0000,1.0000,0.0000,1.0000,0.0000,
0.0200,1.0004,-0.0040,1.0004,-0.0040,
0.0400,1.0015,-0.0080,1.0016,-0.0080,
0.0600,1.0035,-0.0121,1.0035,-0.0120,
0.0800,1.0062,-0.0162,1.0063,-0.0161,
0.1000,1.0097,-0.0204,1.0099,-0.0202,
       atd.
2.9000,-0.1341,-0.0105,0.1345,-3.0635,
2.9200,-0.1321,-0.0102,0.1325,-3.0642,
2.9400,-0.1301,-0.0100,0.1304,-3.0648,
2.9600,-0.1281,-0.0098,0.1285,-3.0655,
2.9800,-0.1262,-0.0095,0.1265,-3.0661,
3.0000,-0.1243,-0.0093,0.1246,-3.0667
```

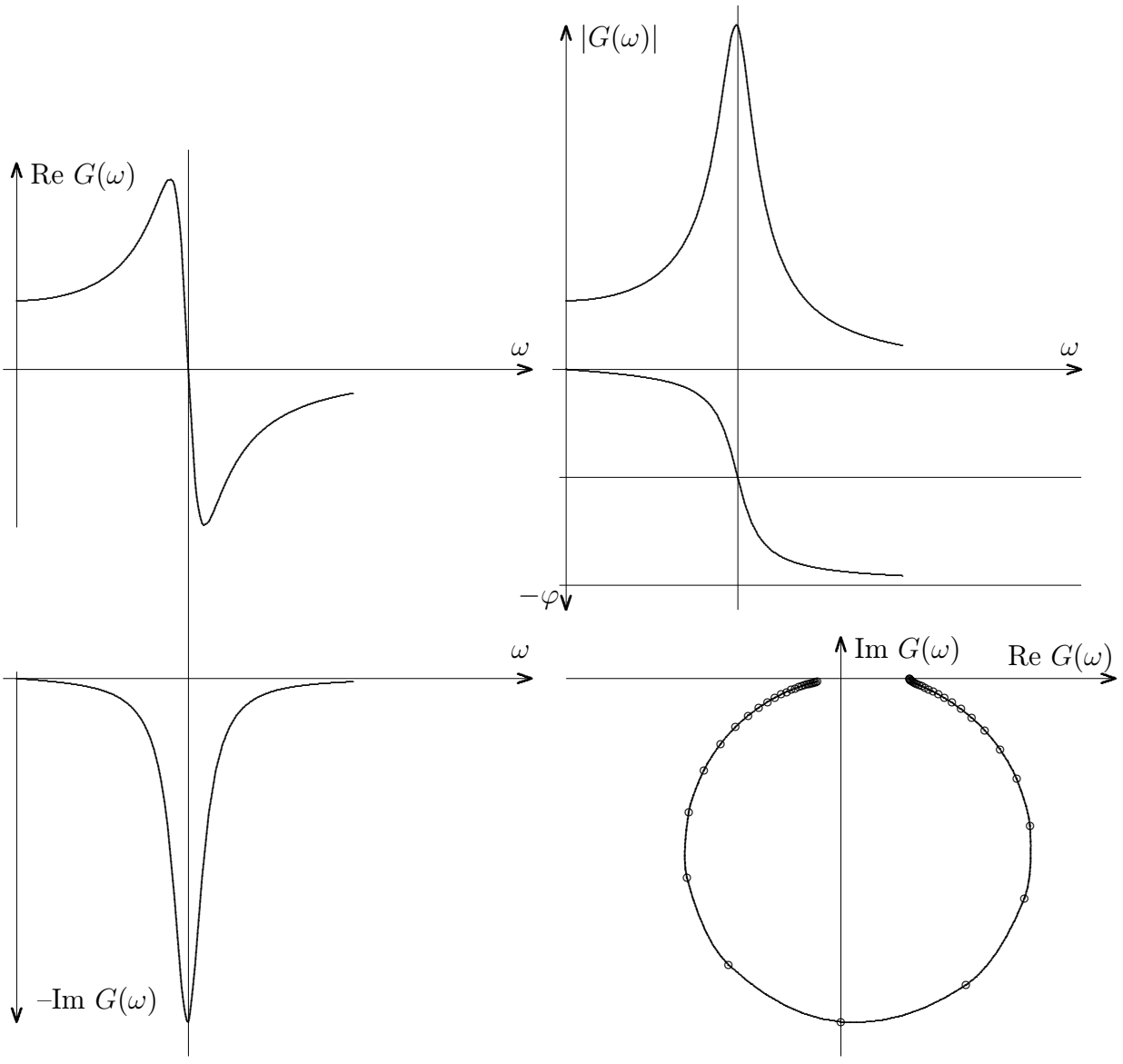


Obrázek 8.1: Frekvenční charakteristiky systému s 1 stupněm volnosti

Vynášení těchto diagramů je na pomalejších počítačích relativně zdlouhavá záležitost. Lze ji zrychlit vynecháním některých bodů. K tomuto účelu lze využít makro `\skipitems`, které vynechává jisté položky na začátku a na konci souboru, a dále čtení ob celočíselný násobek bodů. Tuto druhou možnost zajistíme změnou počtu datových vektorů v řádce. Vyneseme-li proto stejná data jen z intervalu `<0,2>` a ob jeden bod, zkrátí se čas překladu pod polovinu původního. Tuto změnu zajistí příkazy:

```
\skipitems{2}(\Matrix){503}
\Cmd \Matrix={10,0,\Matrix}
```

Ty vložíme do výše uvedeného programu za čtení matice, tedy za řádku s příkazem `\loadgf`, tj. před vlastní kreslení.



Obrázek 8.2: Frekvenční charakteristiky systému s 1 stupněm volnosti

8.2 Vyplňování formuláře

Požadavek na vyplnění předtištěného formuláře je dosti častý. Obvykle se vyplní buď ručně nebo psacím strojem. Chceme-li tuto práci přenechat počítači, začneme narážet na takřka nepřekonatelné potíže spočívající v odlišné rozteči řádek formuláře a často se měnící poloze tisku formuláře od okrajů papíru.

Jako příklad uveďme vyplnění předtištěného formuláře DB pro dílčí zprávu o čerpání nákladů přidělených na projekt Grantové agentury ČR. Při použití LaTeXu nelze úlohu dobře splnit v textovém režimu, díky inteligenci, kterou TeX dostal do vínku. Ten si totiž může sám v jistých mezích upravovat rozteč řádek i mezer, což má za následek neurčitost cílové polohy textu. Aby se tomu zabránilo, musíme vynášet i texty příkazem `\put` v prostředí `picture`, které dodržuje předepsané vzdálenosti.

Pro vyplnění formuláře byl sestaven obecný programový modul se jménem `db.tex`, do něhož se konkrétní data načítají z předem připraveného souboru. Stejně jako v předešlém příkladu se musí data přečíst v té skupině, ve které jsou příkazy `\put` pro vynášení objektů (zde textů) lokální. Program `db.tex` má tvar:

```
% DB.TEX        Dílčí zpráva - Výkaz hospodaření
% ~~~~~~        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
\documentstyle[czech,a4wide,array,zcu,12pt]{article}
  \topmargin-8mm
  \textheight260mm
  \textwidth160mm
  \parskip1ex
  \parindent0ex
  \thispagestyle{empty}
  \oddsidemargin-10mm

  \newcnt\j
  \newlength\yy
  \newlength\yo
  \newlength\lindist
  \def\h{\hfill}

\begin{document} % @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  \s
  \yo62mm%                              Pozice 1. řádky skupiny údajů
  \lindist5.5mm%                        Rozteč řádek
  \Ckbd \file%                          Zadej jméno souboru s daty
  \loadgf \data=\file%                  Přečti soubor do \data
  \wris{Čekej na sestavení tabulky! ***}
  \unitlength1mm
  \begin{picture}(180,255)(0,-255)
    \nextitem \yr=(\data)
    \put(170,-13){\Large\bf \yr}%       Současný rok
    \nextitem \yr=(\data)
    \put(114,-21){\Large\bf \yr}%       Příští rok
    \nextitem \nameA=(\data)
    \nextitem \nameB=(\data)
    \nextitem \nameC=(\data)
    \nextitem \nameD=(\data)
    \put(0,-43)%                        Řešitel & spoluřešitel
    {\begin{tabular}[t]{p{30mm}p{30mm}}
       \nameA & \nameB \\
       \nameC & \nameD
     \end{tabular}
```

```
}%----------------------------------------   VĚCNÉ NÁKLADY:
\Ifor i=(0:1:9)%
{\nextitem \da=(\data)
 \nextitem \db=(\data)
 \nextitem \dc=(\data)
 \nextitem \dd=(\data)
 \yy\thei\lindist \advance\yy\yo \yy.35146\yy
 \ltoc \ycmd=\yy
 \put(66,-\ycmd)%
 {\begin{tabular}[t]%
   {>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}
    >{\h}p{23mm}<{\h}}%
    \da & \db & \dc & \dd
  \end{tabular}
 }
 \ifthenelse{\thei=8}{\advance\yo3.5mm}{}
}%----------------------------------------  INVESTICE:
\yo127mm
\Ifor i=(0:1:3)%
 {\nextitem \nameA=(\data)
  \nextitem \da=(\data)
  \nextitem \db=(\data)
  \nextitem \dc=(\data)
  \nextitem \dd=(\data)
  \yy\thei\lindist \advance\yy\yo \yy.35146\yy
  \ltoc \ycmd=\yy
  \put(22,-\ycmd)%
  {\begin{tabular}[t]%
    {>{\h}p{42mm}<{\h}>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}
     >{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}}%
    \nameA & \da & \db & \dc & \dd
   \end{tabular}
  }
  \ifthenelse{\thei=2}{\advance\yo2.5mm}{}
}%----------------------------------------  MZDY:
\yo168mm
\Ifor i=(0:1:6)%
 {\nextitem \da=(\data)
  \nextitem \db=(\data)
  \nextitem \dc=(\data)
  \nextitem \dd=(\data)
  \yy\thei\lindist \advance\yy\yo \yy.35146\yy
  \ltoc \ycmd=\yy
  \put(66,-\ycmd)%
  {\begin{tabular}[t]%
    {>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}
     >{\h}p{23mm}<{\h}}%
     \da & \db & \dc & \dd
   \end{tabular}
  }
  \ifthenelse{\thei=5}{\advance\yo10mm}{}
}%----------------------------------------  CELKOVÉ NÁKLADY:
\lindist8mm
\yo221mm
\Ifor i=(0:1:2)%
 {\nextitem \da=(\data)
  \nextitem \db=(\data)
  \nextitem \dc=(\data)
  \nextitem \dd=(\data)
  \yy\thei\lindist \advance\yy\yo \yy.35146\yy
  \ltoc \ycmd=\yy
```

```
      \put(66,-\ycmd)%
      {\begin{tabular}[t]%
        {>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}>{\h}p{23mm}<{\h}
         >{\h}p{23mm}<{\h}}%
         \da & \db & \dc & \dd
       \end{tabular}
      }
      \ifthenelse{\thei=1}{\advance\yo3.5mm}{}
    }%-----------------------------------------  KONEC
  \end{picture}
\end{document}   % @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Program `\db.tex` využívá styl `array.sty`, který rozšiřuje možnosti LaTeXu v tabulkových prostředích `array` i `tabular`. Program nejdříve přečte datový soubor, jehož jméno zadá uživatel z klávesnice, do povelu `\file`. V našem případu měl tento soubor název `db_tabl.tex` a tvar:

```
  1996,
  1997,
  Doc. Ing. Mir.,,
  BALDA DrSc.,,
  55,83.5,60,70,
  115,44.6,86,91.3,
  300,336.6,363,383,
  25,23.8,123,61.7,
  192,192,72,123,
  61,32.5,55,55,
  ,,,,
  ,37,,,
  ,,,6,
  748,750,759,790,
  ,,,,
  ,,,,
  ,,,,
  ,,,,
  ,,,,
  13,11,48,30,
  10,10,10,10,
  39,39,78,78,
  ,,,,
  52,52,55,57,
  ,,,,
  114,112,191,175,
  862,862,950,965,
  862,862,950,965,
  ,,,,
```

Tato data se postupně odebírají z povelu `\file` a ukládají se na požadovaná místa tabulky do předtištěného formuláře. Doladění nastavení formuláře v tiskárně se bohužel musí provést zkusmo podle konkrétní polohy předtištěné předlohy. Na další stránce je uveden výsledek – doplněný formulář DB.

Výhodou tohoto postupu je skutečnost, že se pracný program formuláře zpracoval pouze jedenkrát, ale použije se ve všech létech, v nichž se podává hlášení o postupu prací. Kromě toho formuláře DB1 a DB2 mají takřka stejný tvar, takže k jejich vytvoření byly zapotřebí pouze „kosmetické“ úpravy. A tak v každém roce se sestavují pouze soubory dat `db_tabl.tex`, `db1_tabl.tex` a `db2_tabl.tex`.

Kapitola 9

Závěr

Počet publikací ve světě rok od roku výrazně narůstá. Není vůbec výjimkou, že redakce renomovaných časopisů přímo vyžadují, aby nabízený článek byl zpracován v jazyku TEX nebo LATEX. To pochopitelně klade na autora zvýšené nároky, se kterými se vyrovnává každý jinak. Nejsou to však pouze redakce časopisů, ale i nakladatelství knih, učebních textů ap., vyžadující špičkovou kvalitu tiskové předlohy, v poslední době označované jako „camera ready“. Autor této práce se potýkal s problémy spojenými s přípravou odborných textů doprovázených obrázky po dlouhou dobu. Výsledky tohoto zápolení jsou shrnuty do dvou souborů maker `rplot.sty` a `user.sty` určených pro sázecí systém LATEX v. 2.09 pro osobní počítače bez zvláštních požadavků na jejich hardware.

Soubor `rplot.sty` obsahuje více než 125 maker pro celočíselné operace i reálnou aritmetiku s pevnou řádovou čárkou a pro grafické práce v návaznosti na soubor maker `curvesls.sty` (viz lit. [12]). Rozšiřuje možnosti uživatele LATEXu o využití řady matematických funkcí, cyklů i zpracování seznamů veličin vytvořených v rámci daného dokumentu nebo snímaných z externích souborů. Jeho nespornou výhodou je skutečnost, že umožňuje současné zpracování textu i grafiky v jednom běhu. Předpokladem je zvládnutelnost úlohy předloženým aparátem a počítačem. Díky již zmíněné reálné aritmetice jsou jednoduché výpočty na počítačích PC 486 a vyšších realizovatelné v přijatelném čase. Je třeba však mít na mysli, že LATEX není určen k výpočtům, ale k sázecím pracím. Aritmetická podpora má pouze rozšířit jeho schopnosti a umožnit i činnosti, které se dosud musely zajišťovat komplikovaně i v jednoduchých případech. Rozsáhlé výpočty spojené často s počítačovou grafikou nejsou s ohledem na nepřesnost operací a malou rychlost pro tento systém vhodné. Odtud plyne i jeho omezení na jednodušší grafické činnosti. Pro složitější grafické úlohy bude nutné využití kombinovaného postupu se začleňováním výstupů z dokonalých grafických programů do výsledného dokumentu. To však má také jistá úskalí a nevýhody.

Pro podporu prací v textovém módu je zde popsán soubor maker `user.sty`. Obsahuje přibližně 85 nových maker, které jsou ještě doplněny serií 35 maker zabudovaných již v souboru `rplot.sty` věnovaných celočíselné aritmetice a vstupu a výstupu přes terminál. Je zajištěno, že při současném použití obou souborů, nedojde při jejich zavádění ke kolizi a společná makra se zavedou pouze jedenkrát. Výhodou tohoto uspořádání je úspora času překladu i operační paměti v případech, kdy uživatel potřebuje operativní vstupy a výstupy a případně i aritmetiku, ale nikoliv grafiku. Potom vystačí se souborem `user.sty`. Nová makra jsou výhradně orientována na textový režim, a to na úpravu dokumentů, snadný zápis rovnic, centrovaných objektů, odsazování, matematické symboly a konstrukce, matice a vektory, boxy a pod. Jejich využívání pomáhá jak ke zrychlení tvorby dokumentu, tak i k jeho zpřehlednění a tedy i ke snížení výskytu chyb.

Chyby při tvorbě dokumentu jsou vždy nepříjemné. Zřejmé formální chyby se odstraní snadno. Horší situace nastává, pokud se v průběhu tvorby grafické informace provádějí jisté výpočty, které skončí bez hlášení chyb, avšak výsledek je špatný. Příčinou je obvykle chybný algoritmus, anebo hrubá chyba v použitých datech. K odhalování těchto chyb jsou oba soubory vybaveny makry pro operativní výpisy na obrazovku během překladu. Ty výrazně zvyšují rychlost nalezení chyb a jejich odstraňování.

Oba soubory s více než 200 makry se nabízejí k volnému použití všem zájemcům, kterým mohou usnadnit přípravu komplikovanějších dokumentů. Předkládají se bez jakýchkoliv záruk tak, jak jsou zde popsány. Zájemce si je může překopírovat pomocí služby `ftp` voláním `ftp mars.zcu.cz` v adresáři `pub` a podadresáři `tex`, kde budou uloženy jejich aktuální verze, pomocí příkazu `get`, příp. `mget`. Autor si v žádném případě nedělá nárok na bezchybnou funkci maker za všech okolností, které mohou při jejich intenzivním využívání nastat. Přes velké úsilí věnované zpracování a kontrole maker a tohoto manuálu nelze vyloučit ani chyby či nepřesnosti v nich. Jistě se najdou makra nebo jejich úseky, která by se dala zpracovat lépe, efektivněji a s menšími nároky na čas a prostor. Protože by tato zlepšení mohla pomoci všem uživatelům, bude autor vděčen za jakékoliv připomínky, které mu budou doručeny ať ústně, písemně nebo e-mailem. Ty by se mohly promítnout do dalších verzí obou souborů, pokud se prokáže jejich užitečnost.

V Plzni, dne 29. 8. 1996

Prof. Ing. Miroslav Balda, DrSc. mbalda@hera.zcu.cz
Ústav fyzikálního inženýrství tel.: (019) – 7221178
Veleslavínova 11 fax.: (019) – 7220787
301 14 Plzeň

Příloha A

A.1 Vytváření souboru matice v jazyku MATLAB

Jak bylo řečeno v kapitole o vstupech a výstupech, makro `\getxy`, vybírá datové vektory ze seznamu, který byl před tím přečten makrem `\loadgf`. Pro činnost makra `\getxy` musí mít seznam tuto strukturu:

`nvar`	celé	počet vektorů-řádek matice dat (procesů)
`N`	celé	počet vektorů-sloupců matice dat (skupin pozorování)
`matice`	real	matice typu (nvar,N) ve tvaru seznamu, v němž jsou reálné položky odděleny čárkami.

V programech zapsaných v jazyce `MATLAB` v. 4.x lze tuto strukturu vytvořit za pomoci `M`-souboru o jménu `texfile.m`:

```
% TEXFILE.M        Prepare file for plotting within LaTeX
% ~~~~~~~~~                                  mbalda@hera.zcu.cz
function texfile(XY,file,form)
%        ~~~~~~~~~~~~~~~~~~~~~
%  XY    matrix with signals as rows of XY matrix
%  file  name of the file to be created
%  form  required format of matrix elements

[ns,N]=size(XY);%       ns = # of signals,  N = # of data per signal
if nargin<3
  f='%6.4f';
  if nargin==1, file=inp('nazev souboru','data.d'); end
end
form=[];
for n=1:ns-1, form=[form,f,','']; end
form=[form,f];
eval(['[fid,message]=fopen(''' file ''',''w'');']);
if fid<=0, message, end
fprintf(fid,'%5i,%5i,\n',ns,N);
fprintf(fid,[form,',\n'],XY(:,1:N-1));
fprintf(fid,[form,'\n'],XY(:,N));
fclose(fid);
```

V tomto `M`-souboru se užívá ještě `M`-funkce `inp.m`, a to v případě, že nebylo jméno souboru zadáno při volání. Pak musí jméno souboru zadat operativně uživatel z klávesnice. Otisk této funkce je uveden dále.

`M`-funkce `inp.m` je alternativou ke standardní `M`-funkci `input`, která je určena pro vstup dat z klávesnice a má jen jeden parametr – řetěz výzvy (nápovědy). U `inp.m` lze zadat až tři parametry. Pokud se zadá jen jeden parametr, totiž řetěz znaků pro nápovědu, má `inp.m` zcela stejnou funkci jako `input`. Druhý argument lze využít pro nabídku

doporučené hodnoty pro daný vstup. Tu uživatel může buď přijmout odklepnutím ENTER, anebo zadat jinou. Třetí argument se zadává vyjímečně, a to v případě, že odsazení výzvy na vstup o 10 znaků doprava od počátku chceme změnit na jiné. Posledním parametrem může býtřetěz s formátem, v němž požadujem výstup zadávaných dat na obrazovku, tedy '\%...'. Funkce inp.m volá někdy i funkci pro konverzi reálného čísla na řetěz f2st.m, a proto je rovněž uvedena.

```
function data = inp(prompt,deflt,nsp,form)
%        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% INP       Input data                    v 7.0   December 1996
% ~~~
%   prompt  string of characters
%   deflt   default value of input data
%   nsp     number of leading spaces before prompt
%   form    format of default output

if nargin>0
   if nargin<4, form='%8.4f';
      if nargin<3, nsp=10;
         if nargin<2, deflt=[];
end, end, end
else
   nsp=10;  prompt='input';  deflt=[];
end
str = [blanks(nsp), prompt, ' = '];
if isempty(deflt) %           in case without default value:
   while 1
      data = input(str);
      if ~isempty(data), break, end
   end
else %                        in case with default value:
   if isstr(deflt)
      data = input([str, deflt, ' => '],'s');
   else
      [md,nd]=size(deflt);
      if md>1
         dstr=[str, f2st(deflt(1,:),form)];
         blx=blanks(length(prompt)+nsp+3);
         for i=2:md
            dstr=[dstr; blx, f2st(deflt(i,:),form)];
         end
         disp(dstr)
         data=input([blanks(length(prompt)+nsp), ' => ']);
      else
         data = input([str, f2st(deflt(1,:),form), ' => ']);
      end
   end
   if isempty(data)
      data = deflt;
   end
end


function s = f2st(f,form)
%        ~~~~~~~~~~~~~~~~
% F2ST     Float to string conversion
%   f      float number or variable
%   form   required format in the form '%m.nt'

if isstr(f), s=f; return, end
if nargin<2, form='  %g'; end
[m,n]=size(f); s=[];
for i=1:m
  s=[s; sprintf(form,f(i,:))];
end
```

A.2 Klávesnicová makra pro DOS Manažer

TEXSHELL LaTeX makra		**KEY**	**prostředí** & **CTRL**	**grafika** & **ALT**	**různé** **CTRL** **ALT**	**boxy** & **CTRL** **SHIFT**	**řecká** **abcd** **SHIFT** **ALT**	**QUICK** **OPE** **CTRL** **Q**
—	**SHIFT**							
		A	array	abstract	appendix	center & arr	α	replace
		B	bibliography	bibitem	boldmath	begin & end	β	
		C	center	circle	put circle	center & pic	∂	goto EOF
		D	docstyle	date	dotfill	dashbox	δ	goto EOL
		E	equation	**EDIT**	eqnarray	fboxsep	ε	goto next err
		F	figure	**FILE**	frac	framebox	φ	search
		G	newcounter	newlength	setcounter	setlength	γ	
		H	hanging	hline	hfill	cline	χ	clear to cursor
		I	itemize	item	indent	input	ι	
		J	JUMP	minpg & pic	noindent	include	ϑ	
		K	BLOCK	thinlines	thicklines	linethickness	κ	
small	CAPITAL	**L**	SEARCH	line	put line	LaTeX	λ	
		M	minipage	multiput	multicolumn	makebox	μ	set R marg
letters	LETTERS	**N**	enumerate	newcommand	renewcmd	newsavebox	ν	
		O	INDENT	**OPTIONS**	put oval	odraz	ω	
		P	picture	put	paragraph	parbox	π	
		Q	QUICK	quote	quotation	qquad	ρ	
		R	ref	cite	rule	raisebox	ϱ	
		S	smallskip	medskip	bigskip	savebox	σ	
		T	tabular	**TEX**	tabbing	TeX	τ	
		U	unitlength	**USER**	unboldmath	usebox	υ	
		V	verbatim	vector	put vector	value	verb	
		W	WRAP	**WINDOW**	upbracefill	downbracefill	ψ	goto last err
		X	xline	**EXIT**	hskip	hspace	ξ	
		Y	DELETE		vskip	vspace	η	clear to EOL
		Z	SHELL	footnote	typein	typeout	ζ	
HLP FWD	HLP BKW	**TAB**	table	$$		center & tab	θ	
HELP	INDEX	**F1**	SYNTAX	index	tiny	chapter		
SAVE	SAVE AS	**F2**	syntax		scriptsize	section		
W OPEN	PRIMARY	**F3**	SET PRIM	W CLOSE	footnotesize	subsection		
EDIT	EDI PRIM	**F4**		ALT EDI	small	subsubsection		
TEX CAD	W ZOOM	**F5**	W CHNG	W DOS	normalsize			
COMPOSE	W NEXT	**F6**	W PREV		large			
LOG VIEW		**F7**			Large			
VIEW		**F8**			LARGE			
PRINT		**F9**	INFORM	INFORM	huge			
MENU		**F10**			Huge			
		F11						
STARTUP		**F12**	MACRO					
		[						search fwd]
		]						search bwd [
		0		W LIST				
		1-9		W 1-9				goto label
INS	PASTE	**INS**	COPY					
DEL	CUT	**DEL**	CLEAR					
BSP	BSP	**BSP**	UNDO					

Set label 0-9 : CTRL-K 0-9 (for later CTRL-Q 0-9)

Práci na dokumentu nezrychlují jen efektivní makra jazyka LaTeX, ale i možné systémové doplňky. Mezi ně patří i klávesnicová makra, jimiž se práce nesmírně urychlí. Po hledání vhodného softwarového prostředku se jako nejlepší ukázal doplňkový modul pro vytváření takových maker programu `DOS Manažer` rožnovské firmy GOLEM, která vyvinula a distribuuje manažer souborů `DOSMAN`, který je v mnoha rysech lepší než jeho vzor Norton Commander. Výše uvedené kombinace jsou patrně maximem, z něhož si uživatel stejně zapamatuje jen nejužívanější.

Práce s tímto systémem klávesnicových maker se v cca dvouletém provozu plně osvědčila. Využívá se v prostředí TEXShell zpracovaném p. Schlegelmilchem (viz lit. [15]). Prostředí je příjemné a známé, protože je založeno na produktech fy Borland. Z tohoto důvodu byla důležitým funkcím TurboVision dána přednost, takže některé kombinace kláves jsou pro LATEX nedosažitelné. Ty jsou v tabulce kombinací kláves označeny rámečkem s obsahem odpovídajícím ovládání fy Borland. Pokud to obsazení kláves umožňovalo, byly jistým kombinacím kláves přiřazeny skupiny maker:

`CTRL`	definice prostředí
`ALT`	objekty prostředí `picture` a další makra
`CTRL ALT`	`\put(,){objekt}` a různá makra
`CTRL SHIFT`	boxy a různá makra
`SHIFT ALT`	řecká abeceda

Další zásadou pro zařazování maker kombinacím kláves bylo zachování buď písmenové, nebo významové mnemotechniky. Většina maker má počáteční písmeno shodné s písmenem klávesnicového makra. Někdy při vyčerpání prostoru se makro přiřadilo vedlejší klávesnici nebo se vytvořila skupina příbuzných maker a přiřadila se nevyužitému písmenu. Jde např. o čítače a délkové registry, které jsou spojeny s písmenem `G`, nebo o vertikální mezery xxx`skip`, které obsazují písmeno `S`. Jiná logika byla použita pro horizontální a vertikální mezery, kterým byla přiřazena písmena `X` resp. `Y`.

Literatura

[1] Knuth D. E.: The TEXbook. Addison-Wesley, Reading, 1984

[2] Lamport L.: LATEX A Document Preparation System. Addison-Wesley, Reading, 1986

[3] Goosens M., Mittelbach F., Samarin A.: The LATEX Companion, Addison-Wesley, Reading, 1994

[4] Balvínová A.: Receptář uživatelů LATEXu. Ediční středisko ČVUT, Praha, 1992

[5] Balvínová A., Bílý M.: Textové informační systémy. Sázecí systém LATEX. Vydavatelství ČVUT, Praha, 1994

[6] Olšák P.: Typografický systém TEX. Praha, 1995

[7] Rybička J.: LATEX pro začátečníky. Konvoj, Brno, 1995

[8] ?: PicTEX

[9] Horn G.: TEXcad ver. 2.8, Koblenz-Moselweiss, 1990

[10] van Zandt T.: PSTricks. Postscript macros for Generic TEX. User's guide. Princeton University, 1992

[11] Hunter R., Bagby S.: Creating Documents with Scientific Word and Scientific WorkPlace. TCI Softwareresearch, 1995

[12] Ian Maclaine–cross: Curves in LATEX Pictures. A Manual for curves.sty and curvesls.sty, Stdney, 1995

[13] Ljusternik L. A., Červoněnkis O. A., Janpol'skij A. R.: Matematičeskij analiz. Fizmat, Moskva, 1963

[14] Abramowitz M., Stegun I. A.: Handbook of Mathematical functions. National Bureau of Standards, New York, 1964

[15] Schlegelmilch J.: TeXShell V2.06. TU Clausthal, 1993

Rejstřík