VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

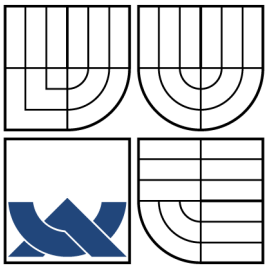
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MOŽNOSTI GRAFIKY A PREZENTACE V SYSTÉMECH TEX A LATEX

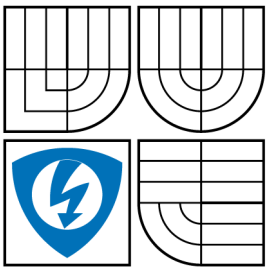
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ROMAN BUŇKA
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MOŽNOSTI GRAFIKY A PREZENTACE V SYSTÉMECH TEX A LATEX

GRAPHICS AND PRESENTATION IN THE TEX AND LATEX SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ROMAN BUŇKA
AUTHOR

VEDOUCÍ PRÁCE — MGR. PAVEL RAJMIC, PH.D.
SUPERVISOR

BRNO 2008

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Bakalářská práce se zabývá způsobem začlenění externích grafických souborů do dokumentů LaTeXu a vytvářením grafiky interními prostředky TeXu. Popisuje základní vlastnosti nástavbového balíčku PSTricks a jeho rozšíření pro tvorbu 2D a 3D grafů, speciálních matematických funkcí a elektrických obvodů. Další oblastí, kterou se práce zabývá, je způsob vytváření PDF prezentací pomocí třídy Beamer.

V rámci této práce byla navržena struktura zdrojového SVG souboru, který zobrazuje elektrické obvody nebo vývojové diagramy. Byla vytvořena příslušná konverzní šablona XSLT, která zajišťuje převod těchto schémat z formátu SVG do formátu PSTricks a umožňuje tyto grafické soubory využít v systému LaTeX.

KLÍČOVÁ SLOVA

TeX, LaTeX, PSTricks, PostScript, Beamer, XML, SVG, XSLT.

ABSTRACT

This bachelor's thesis deals with the insertion of external graphic files into LaTeX documents and the creation of graphics using internal tools of TeX. It describes the basic properties of the PSTricks additional package and its expansion enabling the creation of 2D and 3D graphs, special mathematical functions and electrical circuits. The work deals also with the method of creating PDF presentations using the Beamer class.

This work suggests a structure of a source SVG file displaying electric circuits or flow charts. It also involves the creation of an XSLT conversion template providing for the conversion of these charts from the SVG format to the PSTricks format making it possible to use these graphic files in the LaTeX system.

KEYWORDS

TeX, LaTeX, PSTricks, PostScript, Beamer, XML, SVG, XSLT.

BUŇKA, R. *Možnosti grafiky a prezentace v systémech TEX a LATEX*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 69 s. Vedoucí bakalářské práce Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Možnosti grafiky a prezentace v systémech TEX a LATEX“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Děkuji vedoucímu bakalářské práce Mgr. Pavlu Rajmicovi, Ph.D. za zadání zajímavého tématu a cenné rady při zpracování bakalářské práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Typografický systém TEX je volně šířitelný sázecí program, který již v sedmdesátých létech minulého století vytvořil D. E. Knuth [12]. Byl původně navržen především pro kvalitní sazbu matematických vztahů v technických publikacích a jejímu následnému tiskovému zpracování. Od dob svého vzniku se prakticky nezměnil a je nezávislý na operačním systému. Nevýhodou se ukázala jeho složitost pro běžné uživatele a proto postupně vznikly různá makra, které práci s tímto programovacím jazykem ulehčují. Mezi nejvýznamnější patří LATEX, vytvořený Leslie Lamportem [15], který již obsahuje různá prostředí formátující text, nadefinované styly dokumentů, stránek apod.

TEX byl vytvořen jako sázecí systém a proto není konstruován tak, aby vyhovoval současným nárokům na grafiku v dokumentech. K vytváření grafiky přímo v TEXu slouží programovací jazyk METAFONT, který je jeho nedílnou součástí, nepodporuje však práci s barvami a generuje jen výstupní bitmapu. Na základě METAFONTu vznikl METAPOST, jehož výstupem je místo bitmapy program v PostScriptu. Práci s těmito programovacími jazyky usnadňuje balík maker MFPIC. Další způsob, jak využít PostScript při tvorbě grafiky, používá balíček PSTricks. Příkazy tohoto programovacího jazyka začleňuje do výstupního souboru TEXu, kterým je DVI (DeVice Independent). Tento výstupní soubor se poté přeloží ovladačem, který tyto instrukce umí zpracovat, do formátu PS (PostScript). Také systém LATEX již obsahuje prostředky pro tvorbu primitivních vlastních kreseb v prostředí `picture`, které dále využívají různé doplňkové balíčky.

SVG je značkovací jazyk pro popis dvourozměrné grafiky v XML. Jedná se o vektorový formát, který byl navržený zejména pro využití výhod vektorové grafiky na internetových stránkách a který je podporován významnými internetovými prohlížeči a vektorovými editory. Do dokumentů LATEXu v současnosti není možné grafické SVG soubory přímo vkládat.

Tato práce se zabývá způsobem začlenění externích grafických souborů do dokumentů LATEXu a vytvářením grafiky interními prostředky TEXu. V další části popisuje základní vlastnosti nástavbového balíčku PSTricks a jeho rozšíření pro tvorbu 2D a 3D grafů, speciálních matematických funkcí a elektrických obvodů. Další oblastí, kterou se práce zabývá, je způsob vytváření PDF prezentací pomocí třídy Beamer.

V rámci této práce byla navržena struktura zdrojového SVG souboru, který zobrazuje elektrické obvody nebo vývojové diagramy. Byla vytvořena příslušná konverzní šablona XSLT, která zajišťuje převod těchto schémat z formátu SVG do formátu PSTricks a umožňuje tyto grafické soubory využít v systému LATEX.

1 ŘEŠENÍ STUDENTSKÉ PRÁCE

1.1 Úvod

V kapitole 2.1. popisuji princip importu grafických souborů v systému LaTeX a možnosti jeho konfigurace. Uvádím také podporované grafické formáty. V další části této kapitoly se zabývám příkazy balíčků `graphics` a `graphicx`, které umožňují samotný import grafiky a manipulaci s ní. Závěr této kapitoly patří balíčku `color` pro práci s barvami.

Kapitola 2.2 obsahuje popis standardních prostředků LaTeXu pro vytváření grafiky v prostředí `picture`. Je zde také uveden způsob jak vykreslit v tomto prostředí Bézierovu kvadratickou křivku. Doplňkové balíčky, které možnosti prostředí `picture` rozšiřují, popisuji v navazující části. V závěru kapitoly se zabývám externím programem `gnuplot`, který umožňuje výstup dat ve formátu LaTeXu.

Kapitola 2.3 pojednává o makru PSTricks. Je zde uveden jeho princip a základní nastavení parametrů. V převážné části této kapitoly se zabývám rozšířením PSTricks pro vykreslení: 2D grafů (`pst-plot`), 3D grafů (`pst-3dplot`), speciálních matematických funkcí (`pst-func`) a elektrických obvodů (`pst-circ`).

Kapitola 2.4 popisuje třídu Beamer, která umožňuje vytvářet kvalitní PDF prezentace. Uvádím její nejdůležitější vlastnosti, nastavení vzhledu prezentace a její hlavní části. V závěru kapitoly se zabývám efektem `overlays`, tj. postupným vykreslováním snímku prezentace a přechody mezi snímky pomocí vizuálních efektů.

Kapitola 2.5 se věnuje tvorbě stylové šablony XSLT pro konverzi grafiky z SVG formátu do PSTricks. Nejprve uvádím stručný popis technologií XML, SVG a XSLT, následuje návrh struktury zdrojového SVG souboru s vysvětlením principu konverze SVG elementů a atributů do PSTricks. V závěru shrnuji vlastnosti vytvořené stylové šablony XSLT a její praktické využití.

2 VÝSLEDKY STUDENTSKÉ PRÁCE

2.1 Vkládání externích grafických souborů do dokumentů

Systém LaTeX nabízí možnost vkládat do dokumentů grafické soubory vytvořené externími programy. Tuto grafiku rozlišujeme na vektorovou a bitmapovou (rastrovou). Vektorové grafické formáty jsou tvořeny geometrickými objekty, jako jsou úsečky, body, kružnice, křivky apod. Mezi vektorové formáty patří EPS, což je tzv. zapouzdřený PostScript, dále PS, AI, CDR, DXF a další. Bitmapové grafické formáty jsou tvořeny jednotlivými body (pixely), kde je každý bod uspořádán do mřížky (bitmapy), která určuje jeho polohu a barvu. Bitmapových formátů je celá řada, patří sem JPEG, BMP, GIF, PNG, TIFF a další.

2.1.1 Princip importu

Pro systém LaTeX je obrázek boxem dané šířky, výšky a hloubky, stejně jako všechny znaky které zpracovává, obsah souboru je tedy pro něj skrytý. Informace o velikosti se zjistí v samotném grafickém souboru z řádku `%%BoundingBox:` *llx lly urx ury*, který udává souřadnice levého dolního (*llx, lly*) a pravého horního (*urx, ury*) hraničního boxu. Příkazem `\includegraphics[bb=`*llx lly urx ury*`]{`*název-souboru*`}` lze tyto rozměry zapsat přímo nebo je třeba vytvořit soubor `název_obrázku.bb` a zkopírovat do něj výše uvedený řádek[1].

Do výstupního souboru DVI se poté zapíše název grafického souboru a informace jak má být přetvořen. Kódování závisí na zvoleném grafické ovladači. Když ovladač zpracovává soubor DVI, interpretuje tyto speciální instrukce, načte specifikovaný grafický soubor, provede transformace a výsledek umístí na pozici, kterou LaTeX udává. Pokud jsou informace o hraničním boxu správné, tak se oblast uvnitř ohraničeného boxu shoduje s boxem, který pro něj LaTeX rezervoval. V opačném případě se obrázek umístí nesprávně.

V současnosti nelze do dokumentu LaTeXu vkládat jakékoliv grafické formáty. Záleží na typu programu (LaTeX, pdfLaTeX) a použitém ovladači (viz tab.2.1). V případě jiného formátu je nutné provést externě, pomocí různých utilit nebo volně dostupných programů, konverzi grafických formátů.

[1] pro bitmapové formáty JPEG, PNG a pro PDF existuje utilita `ebb.exe` (součást instalačního balíku TeXLive2007), která vygeneruje soubor s bounding boxem

2.1.2 Nastavení importu

Výchozí ovladač je specifikován v lokálním konfiguračním souboru `graphics.cfg`. Nastavuje se příkazem `\ExecuteOptions{`*ovladač*`}`, kde *ovladač* představuje jednu z povolených voleb pro ovladače. Výchozí volbu lze explicitně předefinovat volbou pro jiný ovladač, specifikovanou přímo v dokumentu.

Příkazem `\graphicspath{`*adresáře*`}` lze nastavit vyhledávací cestu pro grafické soubory. Pro *adresáře* se používá syntaxe lokálního operačního systému, např. pro Unix, Windows `\graphicspath{`*adresář1/*`}{`*adresář2/*`}`. Pokud se cesta neuvede, hledá LaTeX grafické soubory v aktuálním adresáři a tam, kde je nastavena cesta pro všechny ostatní soubory TeXu.

Pokud mají PostScriptové grafické soubory větší velikost, nabízí se možnost je zkomprimovat pomocí utility `zip` nebo `gzip`. Soubor EPS se nahradí souborem s příponou `.zip` nebo `.eps.gz`. V balíčku `graphicx` se zkomprimovaný soubor připojí příkazem: `\includegraphics[bb=` *llx lly urx ury*`]{název_souboru.eps.gz}`, který obsahuje ručně zadané souřadnice hraničního boxu a název zkomprimovaného souboru, který je pak v dokumentu PostScriptu zahrnut v původní velikosti.

Obrázek se vloží na místo, kde je uvedený příkaz pro import. Pokud se však nevejde na aktuální stránku, umístí se na začátek následující stránky a tím v dokumentu vznikne prázdné místo. Tento problém řeší tzv. plovoucí prostředí, které umožňuje, aby obrázky (tabulky) plavaly i se svými popisy do místa, kde se sled textu nepřeruší. Prostředí se vytvoří pomocí `\begin{figure}[`*umístění*`]`, následuje příkaz pro vložení obrázku `\includegraphics{`*název-souboru*`}`, příkaz pro jeho pojmenování `\caption{`*název*`}` a prostředí se ukončí `\end{figure}`. Argument *umístění* může mít následující hodnoty: `h` (here) objekt se umístí v místě textu kde se nachází plovoucí prostředí, `t` (top) umístí se na začátek stránky, `b` (bottom) umístí se na konec stránky, `p` (page) umístí se na speciální stránku jen s obrázky. Pokud chceme povolit více možností umístění plovoucího prostředí, lze hodnoty argumentů kombinovat, pokud se neuvede, je standardní kombinací `tbp`.

Tab. 2.1: Podporované grafické formáty konkrétními ovladači

formát	ovladač		
	`dvips`	`dvipdfm`	`pdftex`
EPS	√	√	-
JPEG	-	√	√
PNG	-	√	√
PDF	-	√[2]	√

[2] nedokáže načíst některé PDF soubory, např. vytvořené programem Adobe Acrobat

2.1.3 Import grafických souborů

Ve verzi LaTeX 2_ε je k dispozici balíček `graphics` a rozšířený balíček `graphicx`. Balíčky jsou standardizované pro všechny ovladače, na rozdíl od předchozí verze LaTeX 2.09, kde byly navrženy pro specifické ovladače s odlišnou uživatelskou syntaxí.

Kód pro překlad uživatelských příkazů do instrukcí pro specifické ovladače je umístěn v souborech DEF, které se načtou volbou balíčku. K přepnutí na jiný ovladač stačí změnit volbu a text dokumentu zůstane bez změny. Volba ovladače se specifikuje pomocí příkazu `\usepackage[`*ovladač*`]{graphics}`.
Mezi další volby, které lze při zavádění balíčků přidat, patří:

`draft` – pouze umístí prázdný orámovaný box do místa, kde se má grafika objevit;
`final` – zruší platnost volby `draft`, pro případ globálního nastavení této volby;
`hidescale` – ponechá prázdné místo na místě, kde měl být umístěn text ve změněném měřítku;
`hiderotate` – ponechá prázdné místo na místě, kde měl být otočený text;
`hiresbb` – vyhledá rozměry boxu na řádku `%%HiResBoundingBox`.

Balíček graphics

Příkazem pro import je `\includegraphics[`*llx,lly*`][`*urx,ury*`]{`*název-souboru*`}`. Pokud se specifikuje pouze jeden argument, bude představovat pravý horní roh a levý spodní se nastaví na `[0,0]`. Pomocí `\includegraphics*` se obrázek ořeže tak, že se potlačí kresba mimo specifikovaný hraniční box.
Mezi příkazy tohoto balíčku patří:

`\scalebox{`*h-měřítko*`}[`*v-měřítko*`]{`*text*`}` – změní velikost argumentu *text* tak, že v horizontálním směru vynásobí jeho šířku koeficientem *h-měřítko* a ve vertikálním směru vynásobí jeho výšku koeficientem *v-měřítko*, pokud se volitelný parametr neuvede, bude mít stejnou hodnotu jako *h-měřítko*;
`\resizebox{`*šířka*`}{`*výška*`}{`*text*`}` – upraví obrázek tak, že s ním vyplní box zadané *šířky* a *výšky*, při nahrazení jednoho argumentu symbolem `!` se velikost proporciálně doplní, příkazem `\resizebox*` udává *výška* celkovou výšku včetně hloubky;
`\reflectbox{`*text*`}` – obrázek horizontálně zrcadlově převrátí, *ukázka* viz obr. 2.1b;
`\rotatebox{`*úhel*`}{`*text*`}` – otočí box kolem jeho počátku o *úhel* ve stupních, přičemž kladná hodnota je proti směru hodinových ručiček, *ukázka* viz obr. 2.1a;

Balíček graphicx

V rozhraní balíčku se používají klíče a hodnoty. Jsou dva typy klíčů: klíče které přijímají numerické hodnoty a klíče s hodnotami `true` (pravda) nebo `false` (nepravda). V případě hodnoty `true` (pravda) stačí napsat jen název klíče. Syntaxe je:

a) b)

Obr. 2.1: Ukázka manipulace s obrázkem: a) otočení kolem jeho počátku o úhel -20°, b) efekt zrcadlení

`\includegraphics[`*klíč=hodnota,...*`]{`*název-souboru*`}`.
Mezi možné klíče a jejich hodnoty patří:

`width` = *šířka* – nastaví obrázek na specifikovanou šířku, pokud není definován klíč `height`, tak se zachová poměr stran;

`height` = *výška* – nastaví obrázek na specifikovanou výšku, pokud není definován klíč `width`, tak se zachová poměr stran;

`keepaspectratio` = `true/false` – pokud jsou specifikovány oba klíče `height` i `width`, hodnota `true` zajistí zachování poměru stran při změně velikosti obrázku;

`scale` = *číslo* – specifikuje číslo, které zvětší nebo zmenší obrázek oproti jeho původní velikosti;

`angle` = *číslo* – udává počet stupňů o které se obrázek otočí, kladná hodnota je proti směru hodinových ručiček;

`origin` = *loc* – specifikujeje bod, kolem něhož se provede rotace, výchozí hodnotou je levý dolní roh `bl`, další možností je `c` označující střed, `t` označující vrch, `r` znamenající vpravo a `b` označující účaří, povolené jsou také realizovatelné kombinace těchto hodnot;

`draft` = `true/false` – použitím tohoto klíče nedojde k načtení obrázku, jen se vytiskne zarámované pole příslušné velikosti s názvem souboru;

`clip` = `true/false` – ořeže obrázek mimo hraniční box, *ukázka* viz obr. 2.2;

`bb =` *llx lly urx ury* – umožňuje přímo zadat souřadnice hraničního boxu, používá se když tyto informace chybí v grafickém souboru, pokud se nespecifikují jednotky použijí se velké body (`bp` = 0,35278mm);

`viewport =` *llx lly urx ury* – specifikuje hraniční box relativně k dolnímu levému rohu stávajícího boxu, používá se s klíčem `clip` pro výběr určité části obrázku, *ukázka* viz obr. 2.2;

`trim =` *l b r t* – podobný jako klíč `viewport`, specifikuje hodnotu o kterou se obrázek zmenší z každé strany, *l* značí levou stranu, *b* označuje účaří, *r* pravou stranu a *t* značí vrch;

`hiresbb = true/false` – v grafickém souboru načte informace o jeho hraničním boxu z řádku `%%HiResBoundingBox`.

Nastavení klíč/hodnota se od sebe odděluje čárkou. Klíče jsou volitelným argumentem a jejich pořadí není důležité s výjimkou klíče `angle`, který může změnit význam předchozího klíče `height` a `width`.

Obr. 2.2: Ukázka výběru části obrázku použitím klíče `viewport` a `clip`

2.1.4 Balíček color

Balíček `color` umožňuje v LaTeXu používat barvy. Mezi barvy definované pro všechny ovladače patří: `red` (červená), `green` (zelená), `blue` (modrá), `white` (bílá), `black` (černá), `yellow` (žlutá), `cyan` (azurová), `magenta` (purpurová).

Balíček `color` obsahuje následující příkazy:

`\color{`*barva-definovaná*`}` – nastaví barvu textu, grafických elementů apod.;

`\color[`*model*`]{`*specifikace*`}` – umožňuje použít jiné barvy než jsou definované, *model* může být RGB (red, green, blue), CMYK (cyan, magenta, yellow, black) nebo *model* `gray`, *specifikace* je číslo v rozmezí 0 až 1, které udává intenzitu příslušné komponenty, např. `[rgb]{0,1,0}` je zelená, `gray` přijímá jednu hodnotu;

`\definecolor{`*název*`}{`*model*`}{`*specifikace*`}` – umožňuje přiřadit *název* definovanému barevnému *modelu* a zvolené *specifikaci* v rozmezí hodnot 0 až 1;

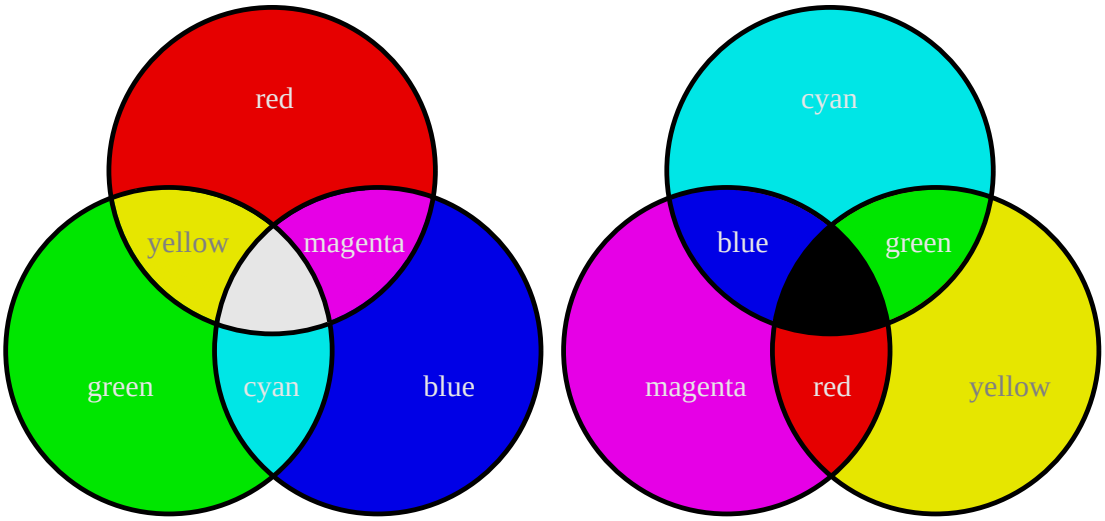
`\pagecolor[`*model*`]{`*specifikace*`}` – nastaví barvu pozadí pro aktuální a následující stránky;

`\textcolor[`*model*`]{`*specifikace*`}{`*text*`}` – nastaví *text* v dané barvě;

`\colorbox[`*model*`]{`*specifikace*`}{`*text*`}` – vypíše *text* v barevném obdelníku s danou barvou jako pozadí;

`\fcolorbox{`*specifikace-rámu*`}{`*specifikace-výplně*`}{`*text*`}` – podobný s příkazem `colorbox`, vytvoří barevný rámeček jenž se nastaví `\fboxrule`*Tloušťka-čáry*, např. `fboxrule0.5mm`;

`\normalcolor` – přepne na barvu, která byla aktivní na konci preambule a jenž mění standardní barvu pro celý dokument.



a) b)

Obr. 2.3: Barevný model: a) RGB, b) CMYK

2.2 Interní grafika LaTeXu

Prostředí `picture` obsahuje vlastní prostředky k vytváření jednoduché vektorové grafiky. Pomocí tohoto prostředí lze sice vytvářet kvalitní obrázky, ale omezeného rozsahu. Více možností nabízejí rozšiřující balíčky a externí programy.

2.2.1 Možnosti prostředí picture

Každý obrázek má definovaný kartézský systém souřadnic. Počátek systému souřadnic se nazývá referenční bod, leží v jeho levém dolním rohu, a při vkládání objektů do obrázku se k němu vztahují. Délková jednotka se nastavuje příkazem `\setlength{\unitlength}{`*délka*`}`. Její modifikací můžeme měnit měřítko výsledného obrázku. Pokud se ale neuvede, je standardně nastaveno na hodnotu 1 `pt`.

Prostředí začíná příkazem `\begin{picture}(`*šířka,délka*`)(`*x,y*`)`, následují kreslící příkazy a prostředí ukončuje `\end{picture}`. Parametr `(`*šířka,délka*`)` specifikuje rozměry obrázku ve směru osy x a y, přičemž se tyto hodnoty násobí zvolenou jednotkou délky. Pozici obrázku v dokumentu určují počáteční souřadnice `(`*x,y*`)`, které jsou vztažené k jeho referenčnímu bodu, pokud se neuvedou mají nulovou hodnotu.

V prostředí `picture` lze pro tvorbu obrázku používat následující příkazy:

`\put(`*x-souřadnice,y-souřadnice*`){`*element*`}` – příkaz pro umístění obrázku, *elementem* může být text nebo další příkazy níže uvedené;

`\multiput(`*x-souřadnice,y-souřadnice*`)(`*x-posun,y-posun*`){`*počet*`}{`*element*`}` – příkaz pro umístění obrázků tolikrát, kolik udává argument *počet*, přičemž se pokaždé posune o definovanou vzdálenost, *ukázka* viz obr. 2.4b;

`\thicklines` – nastaví režim tlustších čár;

`\thinlines` – nastaví režim tenčích čár;

`\linethickness{`*tloušťka*`}` – nastavuje *tloušťku* vodorovných a svislých čar;

Mezi elementy patří následující příkazy:

`\makebox(`*šířka,výška*`)[`*pozice*`]{`*text*`}` – umožňuje umístit *text* do boxu podle stanovených rozměrů, argument *pozice* určuje, kde se v boxu umístí a může nabývat hodnot `t` (nahoru), `b` (dolů), `l` (vlevo), `r` (vpravo) a dále možných kombinací;

`\framebox(`*šířka,výška*`)[`*pozice*`]{`*text*`}` – má stejné možnosti jako předchozí element `makebox` a navíc vytvoří rámeček boxu;

`\dashbox{`*délka-čárky*`}(`*šířka,výška*`)[`*pozice*`]{`*text*`}` – stejný jako `makebox` a navíc vytvoří orámovaný box přerušovanou čarou;

`\shortstack[`*pozice*`]{`*text*`}` – pomocí tohoto příkazu lze psát písmena a texty pod sebe tak, že se jednotlivé řádky od sebe oddělí s co nejmenším svislým rozestupem v závislosti na parametru *pozice*, tento parametr může nabývat hodnot `l` (vlevo), `r` (vpravo), `c` (centrování) je výchozí hodnota, řádky se od sebe oddělí příkazem nového řádku `\\`;

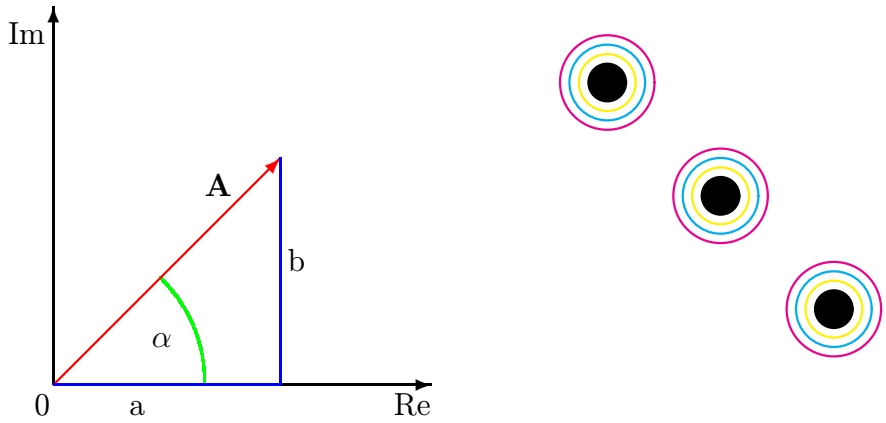
`\line(`$\Delta x, \Delta y$`){`*délka*`}` – kreslí úsečku, např. `(1,0)` nakreslí vodorovnou přímku, kreslení šikmých čar je vymezené podle následujících pravidel: hodnoty $\Delta x, \Delta y$ musí být celé v rozsahu 0 - 6, obě hodnoty nesmí mít společný dělitel a úsečka nesmí mít menší délku než přibližně 3,5 mm (10`pt`);

`\vector(`$\Delta x, \Delta y$`){`*délka*`}` – kreslí vektor, jinak stejné pravidla jako u `line`;

`\circle{`*průměr*`}` – kreslí kružnici, zadaný *průměr* je jen orientační, protože v prostředí `picture` jsou k dispozici jen určité průměry, LaTeX vybere nejbližší hodnotu k zadanému průměru, příkaz `\circle*{`*průměr*`}` nakreslí kruh;

`\oval(`*šířka,výška*`)[`*část*`]` – kreslí ovál ve významu obdelníku, který má rohové body nahrazeny čtvrkružnicemi, argument *část* umožňuje kreslení půloválů s hodnotami `t`, `b`, `l`, `r` nebo čtvrtoválů s hodnotami `tl`, `tr`, `bl`, `br`, čtvrtinu a polovinu kružnice lze kreslit do velikosti přibližně 1,5 cm;

Pro opětovné použití části obrázku můžeme uložit příkazy které jej tvoří a poté je opakovaně použít. Příkazem `\newsavebox{\název_boxu}` se vytvoří paměťový box se zvoleným názvem `název_boxu`. Požadovanou část obrázku uložíme pomocí `\savebox{\název_boxu}(`*šířka,výška*`)[`*pozice*`]{`*text*`}`. Do hlavního obrázku jej pak vložíme pomocí příkazu `\put` a elementu `\usebox{\název_boxu}`.



a) b)

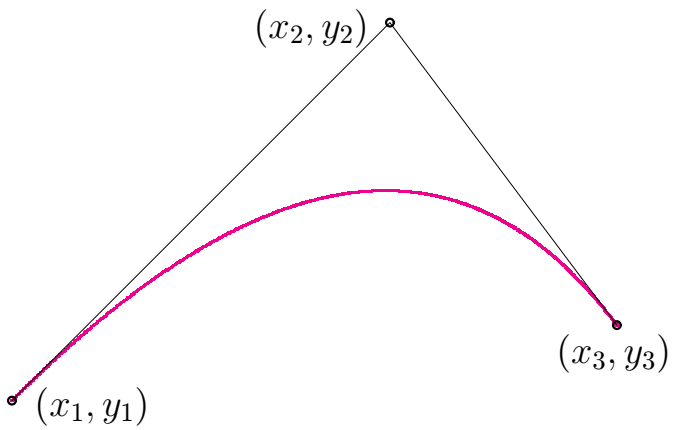
Obr. 2.4: Ukázky grafiky v prostředí `picture`: a) jednoduchý obrázek zobrazující komplexní číslo v Gaussově rovině, b) posunutí objektu o definovanou vzdálenost

2.2.2 Bézierovy křivky

Bézierovy[17] křivky mají široké uplatnění v aplikacích počítačové grafiky a designu.

V LaTeX 2$_\varepsilon$ nakreslíme příkazem `\qbezier[`*počet*`](`x_1, y_1`)(`x_2, y_2`)(`x_3, y_3`)` Bézierovu kvadratickou křivku. Počet bodů je volitelným argumentem, při jeho vypuštění se jeho hodnota vypočítá a vytvoří spojitou čáru. Výchozím bodem je souřadnice x_1, y_1, tzv. řídícím bodem křivky je x_2, y_2 a koncovým bodem x_3, y_3. Tečny sestrojené v koncových bodech procházejí kontrolním bodem, *ukázka* viz obr. 2.5.

Pomocí této křivky můžeme také nakreslit část kružnice, ale vyžaduje to již provést složitější výpočet[16].



Obr. 2.5: Ukázka Bézierovy křivky

2.2.3 Rozšiřující balíčky prostředí picture

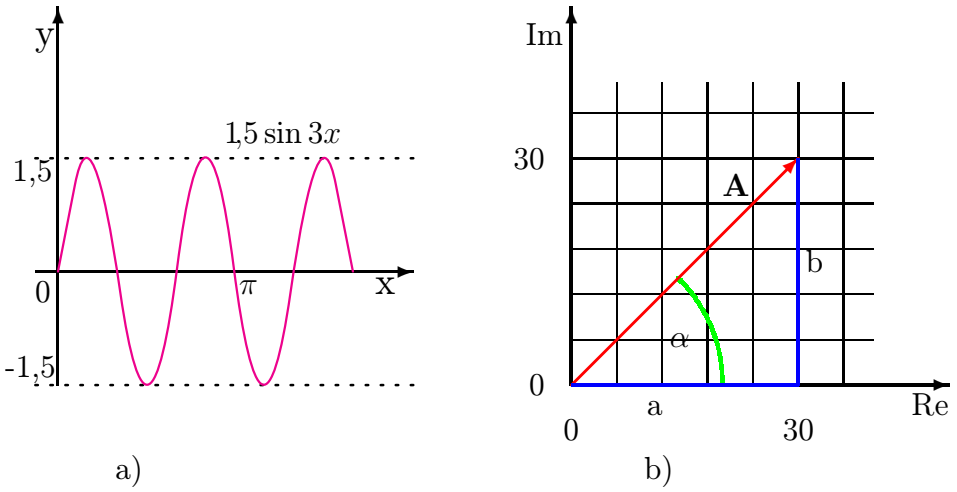
Prostředí `picture` nabízí základní a omezené možnosti pro kresbu obrázků. Z toho důvodu vznikly další balíčky, které jeho možnosti rozšiřují.

Patří mezi ně především makra EPIC a EEPIC, přičemž EEPIC dále rozšiřuje možnosti balíčku EPIC (viz obr. 2.6a). Jedná se o možnost kreslit čáry nebo vektory v jakýchkoliv úhlech, kreslit kružnice o libovolném průměru, dále kreslit tečkované a čárkované čary a umožňuje větší volbu tloušťky čar. Balíček EEPIC přináší ještě několik nových příkazů, pomocí kterých lze kreslit elipsy a jejich výplně, oblouky, kubické křivky s použitím kontrolních bodů. Všechny příkazy jsou přehledně shrnuty v dokumentaci [14], která je součástí balíčku.

Dalším makrem je PiCTeX, který nabízí další možnosti při tvorbě grafů. Např. vytvářet grafy funkcí se souřadným systémem, histogramy, diagramy apod. K tomuto balíčku není volně šířitelný manuál, který lze získat jen v tištěné podobě po zaplacení částky 30$. Volně dostupný je abecední souhrn příkazů [4].

Pro jednoduché kreslení čtverečkovaného papíru slouží balíček `graphpap`. Příkaz pro jeho vytvoření je `\graphpaper[`*počet*`](`*x,y*`)(`*lx,ly*`)`, x a y definuje levý dolní roh mřížky, lx a ly udává šířku a výšku mřížky v závislosti na zvolené jednotce, *počet* udává šířku jednoho dílku mřížky v jednotkách délky a každá pátá čára je zvýrazněna a očíslovaná, *ukázka* viz obr. 2.6b.

Balíček GasTEX[5] je určený pro kreslení automatů, grafů, diagramů a sítí v prostředí `picture`. Byl vytvořen pomocí PostScriptu a proto je pro vytvoření PDF souboru třeba použít cestu LATEX→ ovladač `dvips` → PDF soubor. Místo dokumentace nabízí jeho autor několik příkladů, na kterých ilustruje jeho vlastnosti. K tomuto balíčku bylo vytvořené formou apletu i grafické uživatelské rozhraní JasTEX[9].



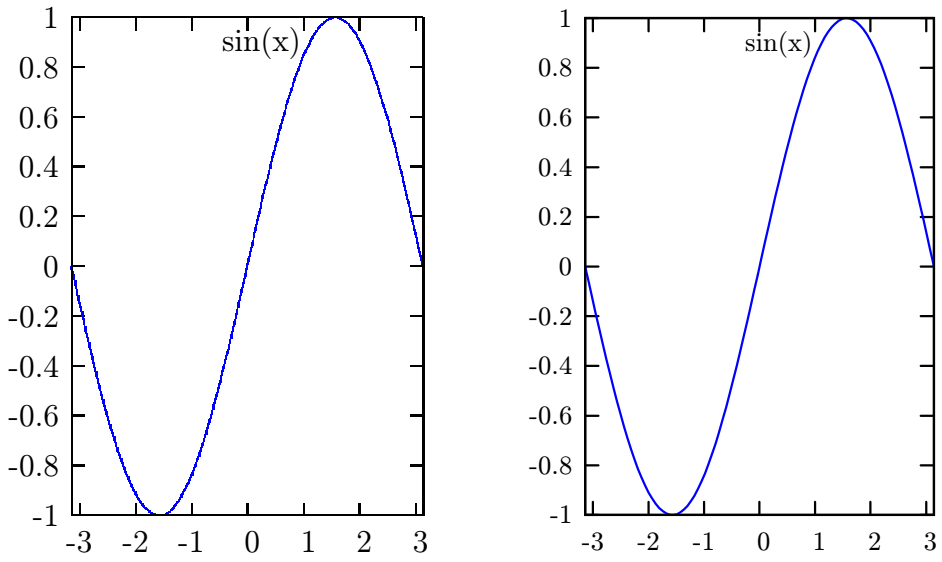
Obr. 2.6: Ukázky grafiky s použitím balíčku: a) EPIC, b) `graphpap`

2.2.4 gnuplot

`gnuplot`[3] je volně šířitelný externí grafický program pro zobrazení matematických funkcí ve 2D i 3D prostředí. Jeho výhodou je podpora různých druhů tiskáren a terminálů, mezi něž patří i LATEX.

Terminálem může být např. LATEX, EEPIC, epsLATEX, PostScript, PSTricks, mp driver pro METAPOST, PDF, PNG a další. Můžeme tedy např. zvolit výstup dat, přes zmiňovaný terminál LATEX, do výstupního souboru ve formátu LATEXu. *Ukázka jednoduchého grafu vytvořeného programem* `gnuplot` viz obr. 2.7a. Pro jeho vytvoření v programu `gnuplot` bylo třeba napsat příkazy: `plot [-3.14:3.14] sin(x)`, export dat definovat pomocí `set terminal latex` a `set output "c:\graf.tex"`. Výstupem programu je soubor `graf.tex`, který obsahuje 270 příkazů LATEXu. Soubor se pak vloží do dokumentu LATEXu pomocí `\input{graf.tex}`.

`gnuplot` tak při tvorbě grafů v prostředí `picture` umožňuje vykreslit nejrůznější křivky pomocí velkého množství příkazů. Je třeba ale počítat s tím, že vykreslené křivky nebudou hladké, protože prostředí `picture` křivku skládá z krátkých úseček podle již zmiňovaných pravidel. Lepšího výsledku tak lze docílit pomocí terminálu EEPIC (obr. 2.7b).



Obr. 2.7: Ukázka grafu vytvořeného programem `gnuplot` s terminálem: a) LaTeX, b) EEPIC

2.3 PSTricks

PSTricks je kolekce maker TEXu založených na PostScriptu, které jsou kompatibilní s většinou ostaních maker TEXu, včetně systému LATEX. Je určený k vytváření vektorové grafiky přímo v dokumentech LATEXu, ale na rozdíl od prostředí `picture` nabízí při její tvorbě neporovnatelně více možností. PSTricks navíc v současnosti nabízí další rozšíření, která se specializují na různorodé oblasti použití. Tyto makra umožňují kreslit např. 2D a 3D grafy, elektrické obvody, UML, geometrii, geografické objekty, optické systémy, chemické značky atd.[18]. Jejich počet se navíc průběžně zvyšuje a u některých rozšíření také dochází k aktualizacím. Dále je třeba zmínit, že součástí balíčku je obsáhlá a kvalitně zpracovaná dokumentace[23] založená na praktických ukázkách. V této kapitole zaměřím na základní nastavení PSTricks a poté především na jeho rozšíření, které se věnují vykreslování 2D a 3D grafů, vybraných matematických funkcí a elektrických obvodů.

2.3.1 Princip PSTricks

PostScript je programovací jazyk vyvinutý firmou Adobe Systems Incorporated, který je určený k popisu textu a grafiky na stránce dokumentu. Tento jazyk obsahuje datové typy, řídící struktury a další techniky, které umožňují detailně definovat výsledný vzhled stránky a vykreslovat grafiku.

Samotný TEX sice příkazy PostSriptu neumí zpracovat, ale umožňuje jejich zápis do výstupního souboru DVI. Používá k tomu vlastní příkaz `\special{`*příkazy*`}`, který při kompilaci uvedené *příkazy*, které mohou být napsané např. v PostScriptu, zapíše do souboru DVI. Tento DVI soubor se poté přeloží ovladačem `dvips`, který tyto PostScriptové příkazy umí zpracovat. Výstupním souborem tohoto ovladače je formát PS, který lze zobrazit na obrazovce, vytisknout na tiskárně nebo převést do PDF souboru. Program pdfLATEX, který vytvoří PDF soubor přímo z DVI, v tomto případě nelze standardním způsobem použít.

2.3.2 Základní nastavení grafických parametrů

Balíček se k dokumentu připojí příkazem `\usepackage{pstricks}`. V tomto příkazu je třeba jako další parametry připojit všechny v dokumentu použité rozšíření. Pomocí `\begin{pspicture}`$(x_0, y_0)(x_1, y_1)$<*kreslící příkazy*>`\end{pspicture}` se vytvoří prostředí pro kreslící příkazy. Vytvořený objekt se poté vloží do boxu, jehož levý spodní roh je v bodě (x_0, y_0), což je implicitně nastaveno na `(0,0)`, a pravý horní bod v bodě (x_1, y_1). Lze použít i příkazy prostředí `picture`, jejich vlastnosti se však v PSTricks nijak nemění.

PSTricks poskytuje vlastní kolekci barev a dále lze definovat vlastní barvu z modelu RGB, CMYK, HSB. Předdefinované jsou barvy ve stupnici šedi: `black` (černá), `darkgray` (tmavě šedá), `gray` (šedá), `lightgray` (světle šedá), `white` (bílá) a barvy: `red` (červená), `blue` (modrá), `cyan` (azurová), `magenta` (purpurová), `yellow` (žlutá), `green` (zelená). Pomocí příkazu `\definecolor{`*název*`}{`*model*`}{`*hodnoty*`}`, lze definovat vlastní barvu, *hodnoty* jsou v intervalu <0, 1>.

PSTricks používá pro nastavení parametrů jednotlivých objektů systém klíč – hodnota. Syntaxe je `\psset{`*parametr1=hodnota1, parametr2=hodnota2,...*`}`, přičemž toto nastavení platí uvnitř stávajícího prostředí. Druhou možností je nastavit parametry samostatně pro jednotlivé objekty, kde se umístí do hranatých závorek, např. `\psline[linecolor=yellow](5,6)` nakreslí žlutou čáru, což je ekvivaletní se zápisem `{\psset{linecolor=yellow}\psline(5,6)}`.

Nastavením rozměrové jednotky lze změnit její výchozí hodnotu, kterou je implicitně 1cm. Jednotky pro souřadnicové osy x a y lze například nastavit každou jinak: `\psset{xunit=3cm, yunit=10pt}`. Úhly pro polární souřadnice a další argumenty se zadávají ve stupních příkazem `\degrees[`*hodnota*`]`.

Mezi základní grafické parametry patří: tloušťka čáry `linewidth`=*hodnota* (implicitní *hodnota* je 0.8`pt`), barva čáry `linecolor`=*barva* (implicitně je nastavená černá), výplň plochy `fillstyle=none/solid`, pokud se zvolí `solid` je objekt vyplněn barvou definovanou parametrem `fillcolor`=*barva*, způsob pro ukončení čáry `arrows`=*styl* může být např. oboustranná šipka (`<->`), úsečka (`-`), šipka na začátku čáry (`<-`) nebo na jejím konci (`->`) apod., `showpoints`=*true/false* (implicitně je nastaveno *false*) vykreslí bodově funkční hodnoty, kontrolní body Bézierovy křivky apod.

2.3.3 pst-plot

`pst-plot` je rozšíření PSTricks určené pro 2D vykreslení matematických funkcí, které se zapisují v PostScriptovém tvaru (viz tab.2.2). Popis tohoto makra je zahrnut v dokumentaci PSTricks[23]. Základním příkazem pro vykreslení grafu matematické funkce je:

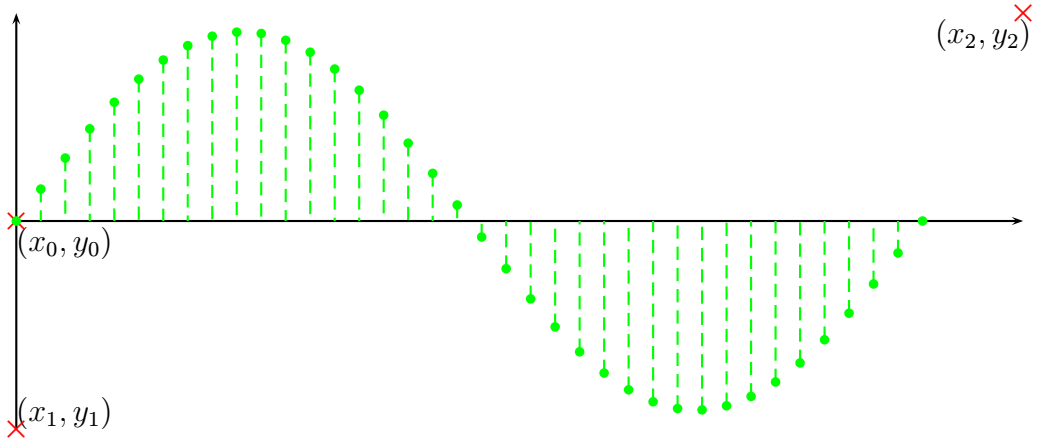
`\psplot[`*options*`]{`*xMin*`}{`*xMax*`}{`*funkce*`}`, kde *xMin* a *xMax* je rozmezí hodnot pro výpočet *funkce*, volitelným parametrem je `plotstyle`=*styl* a může jím být: `dots` (bodový), `line` (spojnicový), `polygon`, `curve` (křivkový), `ccurve` (vykreslí uzavřenou charakteristiku zadané funkce), `ecurve` (vykreslí charakteristiku od počáteční do koncové funkční hodnoty), `LineToXAxis` (spojí funkční hodnoty s osou `x`) *ukázka* viz obr. 2.8, `LineToYAxis` (spojí funkční hodnoty s osou `y`), další parametrem je `plotpoints`=*počet*, který určuje počet funkčních hodnot v zadaném intervalu.

Tab. 2.2: Přehled základních aritmetických a matematických operátorů

operátor	zápis v PostScriptu	výsledek
`add`	*číslo1 číslo2* `add`	součet *číslo1*+*číslo2*
`div`	*číslo1 číslo2* `div`	podíl *číslo1*/*číslo2*
`mul`	*číslo1 číslo2* `mul`	součin *číslo1***číslo2*
`sub`	*číslo1 číslo2* `sub`	rozdíl *číslo1*-*číslo2*
`sqrt`	*číslo* `sqrt`	druhá odmocnina *čísla*
`cos`	*úhel* `cos`	kosinus *úhlu* ve stupních
`sin`	*úhel* `sin`	sinus *úhlu* ve stupních
`atan`	*číslo1 číslo2* `atan`	úhel ve stupních $arctg(číslo1/číslo2)$
`exp`	*báze exponent* `exp`	umocní číslo *báze* číslem *exponentu*
`ln`	*číslo* `ln`	přirozený logaritmus *čísla*
`log`	*číslo* `log`	dekadický logaritmus *čísla*

Pro vykreslení grafu z externích dat je možné použít některý z následujících příkazů:`\listplot[`*seznam*`]`, `\fileplot[`*soubor*`]` nebo `\dataplot[`*příkaz*`]`. Parametrická křivka se vykreslí příkazem `\psplot[`*options*`]{`t_{min}`}{`t_{max}`}{`*funkce*`}` v zadaném intervalu s parametrem t. Ke všem uvedeným příkazům lze použít vhodné parametry PSTricks, např. již zmiňované `linewidth` a `linecolor`. Mnoho nových parametrů pro nastavení `pst-plot` nabízí také rozšiřující balíček `pstricks-add`[19].

Příkazem `\psaxes[`*options*`]{`*arrows*`}`$(x_0, y_0)(x_1, y_1)(x_2, y_2)$ se vytvoří souřadnicové osy, parametr `arrows` byl již zmiňován a způsob zadávaní hodnot do příkazu je znázorněn *v ukázce* viz obr. 2.8, (x_1, y_1) je referenčním bodem. Parametrem může být `ticks=all/x/y/none`, který vyznačí na osách krátké kolmé čáry v násobcích zvolené jednotky a parametrem `labels=all/x/y/none` se na osách nastaví číslování.



Obr. 2.8: Grafické zobrazení harmonického signálu

2.3.4 pst-3dplot

`pst-3dplot` je rozšířením PSTricks, které navazuje na makro `pst-plot` popisované v předchozí části kapitoly. Pomocí `pst-3dplot` lze vyobrazit nejrůznější objekty již v 3D souřadnicovém systému. Veškeré možnosti tohoto rozšíření jsou uvedeny v dokumentaci[24] a v další části textu se zaměřím na možnost vykreslení matematických funkcí typu $f(x,y)$. Princip makra je takový, že se funkční hodnoty vypočítavají pomocí dvou smyček `for` (vnější a vnitřní).

Samotný zápis matematické funkce se musí zadat v PostScriptu[1], který používá postfixovou notaci zápisu (Reverse Polish Notation). Např. funkce $\sqrt[2]{x^2+y^2}$ se zapíše jako `{x dup mul y dup mul add sqrt}`. Postup je takový, že se nejprve uloží na vrchol zásobníku hodnota `x`, tato hodnota se zduplikuje pomocí operátoru `dup` a ta se poté vynásobí s původní (stejnou) hodnotou pomocí operátoru `mul`, stejným postupem se získá hodnota y^2, poté se provede součet x^2+y^2 pomocí operátoru `add` a nakonec se výsledek součtu odmocní pomocí operátoru `sqrt`.

Nejdůležitějšími grafickými parametry tohoto makra jsou hodnoty úhlů α a β, které se nastavují příkazem `psset{Alpha=`*úhel*`, Beta=`*úhel*`}` ve stupních. Úhel α určuje horizontální rotaci roviny `xy`, přičemž kladná hodnota provede rotaci proti směru hodinových ručiček. Úhel β určuje vertikální rotaci roviny `yz`, pokud je hodnota úhlu α minimální, otáčí se téměř kolmo k rovině stránky. Vhodným zvolením těchto dvou parametrů tak lze nastavit z jaké perspektivy bude 3D graf vykreslen.

Syntaxe je: `\psplotThreeD[`*options*`](`*xMin,xMax*`)(`*yMin,yMax*`){<`*funkce*`>}`, přičemž *xMin,xMax* a *yMin,yMax* jsou minimální/maximální hodnoty pro výpočet funkčních hodnot.

Volitelným parametrem *options* může být:

`plotstyle` = *typ* – druh grafu: `dots` (bodový), `line` (spojnicový) je přednastaveno pokud by se tento parametr neuvedl, `polygon`, `curve` (křivkový), `ccurve` (vykreslí uzavřenou charakteristiku zadané funkce), `ecurve` (vykreslí charakteristiku od počáteční do koncové funkční hodnoty);

`showpoints` = `true/false` – zobrazí u spojnicového nebo křivkového grafu i jednotlivé body, default hodnota je `false`;

`xPlotpoints` = *počet* – počet bodů (funkčních hodnot) na ose `x`, default=25;

`yPlotpoints` = *počet* – počet bodů (funkčních hodnot) na ose `y`, default=25;

`drawStyle` = *směr* – určuje ve směru které osy jsou křivky vykresleny, *směr* může být: `xLines` (default), `yLines`, `xyLines` (nejprve jsou křivky vykresleny ve směru osy `x`), `yxLines` (nejprve jsou křivky vykresleny ve směru osy `y`);

`hiddenLine` = `true/false` – průhlednost (viz obr. 2.9) /neprůhlednost 3D grafu (viz obr. 2.10), default je nastavené na hodnotu `false`.

Nastavení souřadnicových os se provede příkazem: `\pstThreeDCoor[`*options*`]` a volitelným parametrem *options* může být:

`drawing` = `true/false` – povoluje/zakazuje zobrazení os, default=`true`;

`Alpha` = *úhel* – umožňuje horizontální rotaci kartézského souřadnicového systému ve směru roviny `xy`, default=45;

`Beta` = *úhel* – umožňuje vertikální rotaci kartézského souřadnicového systému ve směru roviny `yz`, default=30;

`xMin` = *hodnota*, `xMax` = *hodnota* – minimální a maximální hodnoty na ose `x`, default=-1 pro `xMin` a default=4 pro `xMax`;

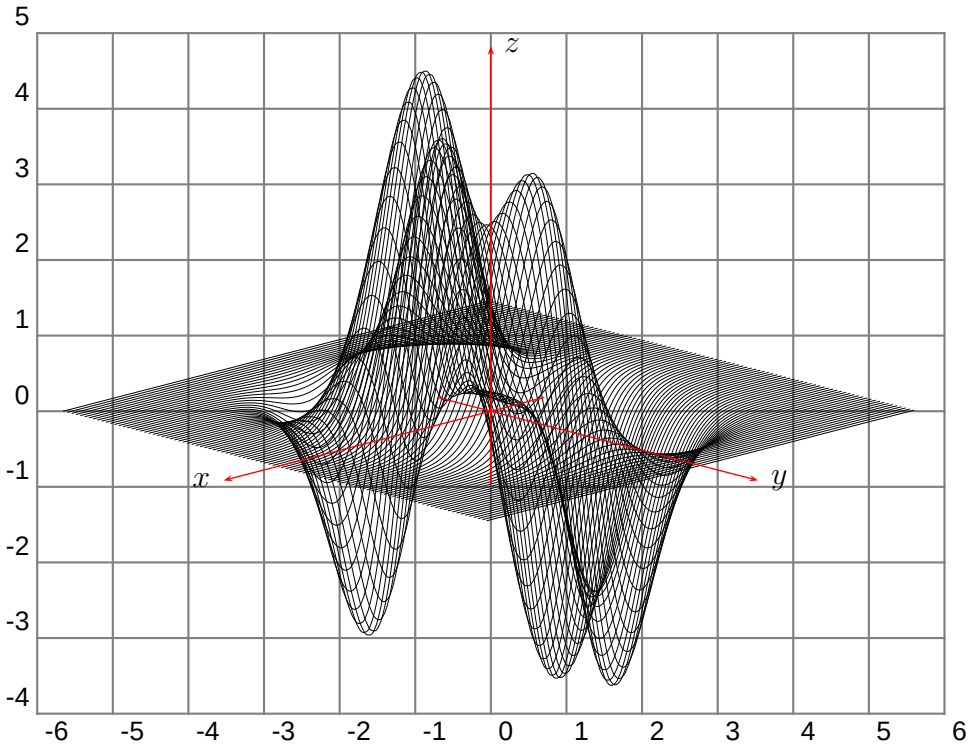
`yMin` = *hodnota*, `yMax` = *hodnota* – minimální a maximální hodnoty na ose `y`, default=-1 pro `yMin` a default=4 pro `yMax`;

`zMin` = *hodnota*, `zMax` = *hodnota* – minimální a maximální hodnoty na ose `z`, default=-1 pro `zMin` a default=4 pro `zMax`;

`nameX` = *název*, `nameY` = *název*, `nameZ` = *název* – označení jednotlivých souřadnicových os, default=`x, y, z`;

`spotX` = *úhel*, `spotY` = *úhel*, `spotZ` = *úhel* – nastavení umístění předchozího parametru okolo směrové šipky příslušné osy, default=180 pro `spotX`, default=0 pro `spotY`, default=90 pro `spotZ`;

`IIIDticks` = `true/false` – povoluje/zakazuje zobrazení hodnot na souřadnicových osách, default=`false`;



Obr. 2.9: Ukázka funkce $z = 10(x^3 + xy^5 - \frac{x}{15})e^{-(x^2+y^2)} + e^{-((x-1.225)^2+y^2)}$

`Dx` = *hodnota*, `Dy` = *hodnota*, `Dz` = *hodnota* – definuje délkovou jednotku pro vykreslení číselných hodnot na příslušných osách, default=1 pro `Dx`, `Dy`, `Dz`;

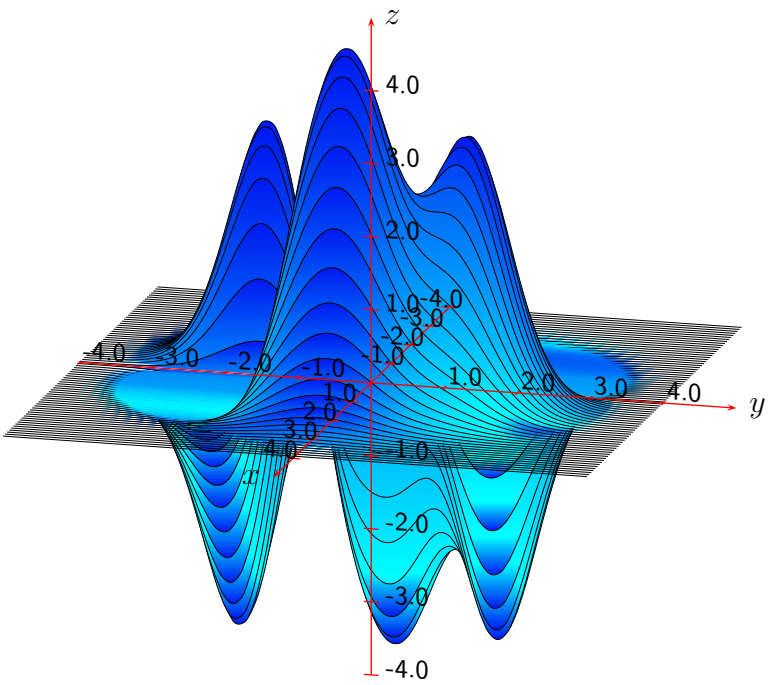
`IIIDxTicksPlane` = `xy/xz/yz` – definuje rovinu, ke které se budou popisky na ose `x` zarovnávat, default=`xy`;

`IIIDyTicksPlane` = `xy/xz/yz` – definuje rovinu, ke které se budou popisky na ose `y` zarovnávat, default=`yz`;

`IIIDzTicksPlane` = `xy/xz/yz` – definuje rovinu, ke které se budou popisky na ose `z` zarovnávat, default=`yz`;

`IIIDticksize` = *hodnota* – definuje délku čáry u jednotlivých hodnot na souřadnicových osách, default= 0.1;

`IIIDxticksep` = *hodnota*, `IIIDyticksep` = *hodnota*, `IIIDzticksep` = *hodnota* – definuje vzdálenost od příslušné osy k popiskům hodnot, default=-0.4 pro `IIIDxticksep`, default=-0.2 pro `IIIDyticksep`, default=0.2 `IIIDzticksep`.



Obr. 2.10: Ukázka funkce $z = 10(x^3 + xy^5 - \frac{x}{15})e^{-(x^2+y^2)} + e^{-((x-1.225)^2+y^2)}$ s nastavením neprůhlednosti 3D grafu

2.3.5 pst-func

`pst-func` je dalším rozšířením PSTricks, které se zaměřuje na speciální matematické funkce.

Polynomická funkce

Polynomická funkce se v makru `pst-func` vykreslí pomocí následujícího příkazu: `\psPolynomial[`*options*`]{`*xStart*`}{`*xEnd*`}`, kde *xStart* a *xEnd* označují na jakém intervalu se funkce zobrazí.

Volitelným parametrem *options* může být:

`coeff` = *hodnota* – koeficienty mnohočlenu, které musí být oddělené mezerou, default=0 0 1 dává parabolu $y = a_0 + a_1x + a_2x^2 \rightarrow y = x^2$;

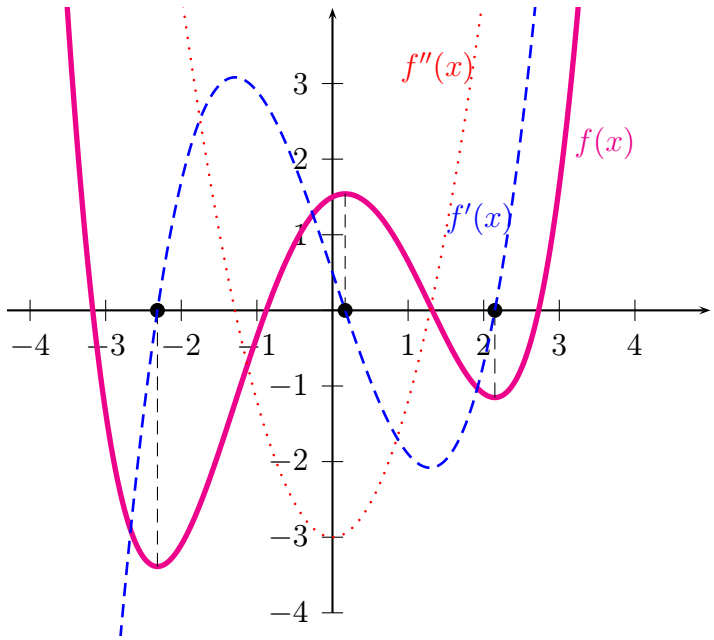
`xShift` = *hodnota* – horizontální posun funkce $(x - xShift)$, default=0;

`Derivation` = *hodnota* – derivace funkce, default=0;

`markZeros` = `true/false` – příkaz vykreslí bodově nulové funkční hodnoty, default=`false`;

`epsZero` = *hodnota* – příkazem se nastaví vzdálenost mezi dvěma nulovými funkčními hodnotami, využije se u iterační funkce k testu, jestli nulová hodnota ještě existuje, default=0.1;

`dZero` = *hodnota* – příkazem se nastaví hodnota kroku, kterým se v zadané funkci vyhledávají všechny nulové funkční hodnoty, default=0.1;



Obr. 2.11: Polynomická funkce $f(x) = 1.5 + 0.5x - 1.5x^2 + 0.15x^4$

`zeroLineTo` = `true/false` – ve stacionárních bodech[3] vykreslí kolmici k těm bodům grafu, kde mohou nastat na zvoleném intervalu lokální extrémy funkce, default=`false`;

`zeroLineStyle` = *styl* – nastaví styl čáry pro předchozí parametr dle platných stylů makra `PSTricks`, default=`dashed`;

`zeroLineColor` = *barva* – příkaz nastaví barvu čáry pro parametr `zeroLineTo`, default=`black`;

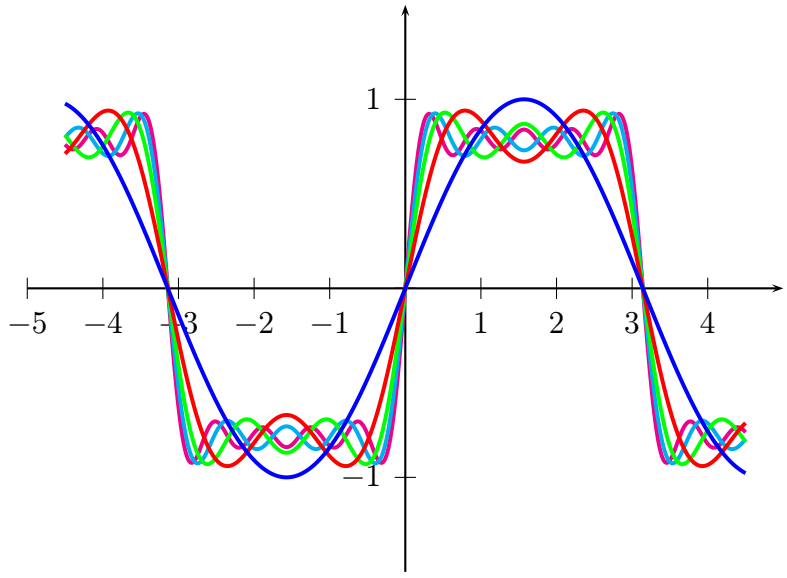
`zeroLineWidth` = *šířka* – příkaz nastaví šířku čáry pro parametr `zeroLineTo`, default=0.5`\pslinewidth`.

Fourierova řada

Fourierova řada funkce $f(x)$ má tvar:

$$f(x) = \frac{a_0}{2} + a_1 \cos \omega x + a_2 \cos 2\omega x + a_3 \cos 3\omega x + \ldots + a_n \cos n\omega x \\ + b_1 \sin \omega x + b_2 \sin 2\omega x + b_3 \sin 3\omega x + \ldots + b_m \sin m\omega x$$

Syntaxe pro vykreslení funkce a částečných součtů její Fourierovy řady je: `\psFourier[cosCoeff=`*a0 a1 a2 ...*`, sinCoeff=`*b1 b2 ...*`]{`*xStart*`}{`*xEnd*`}`, kde koeficienty *a, b* musí být oddělené mezerou a předdefinovanými hodnotami jsou `cosCoeff`=0 a `sinCoeff`=1, které dávají standardní funkci sinus.

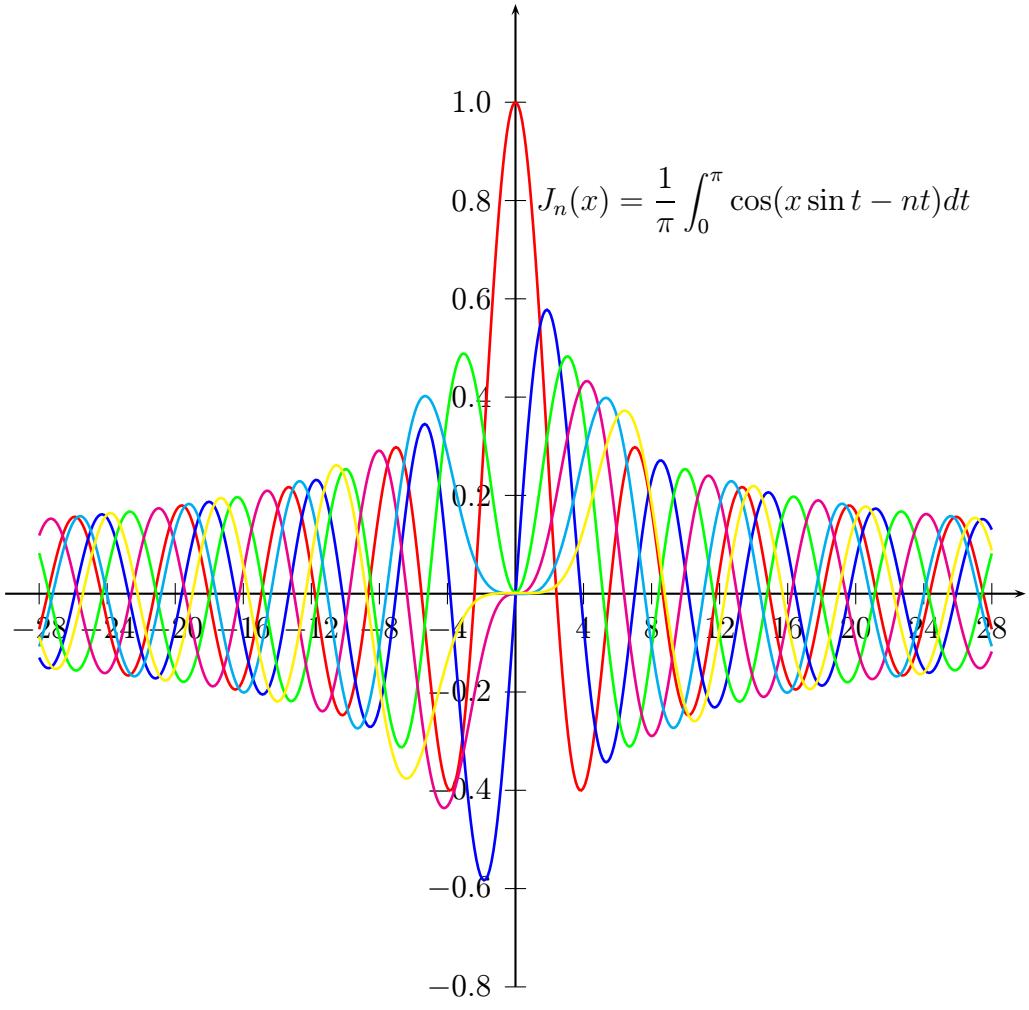


Obr. 2.12: Funkce sinus a částečné součty její Fourierovy řady

[3] stacionární body dostaneme z podmínky $f'(x) = 0$

Besselova funkce

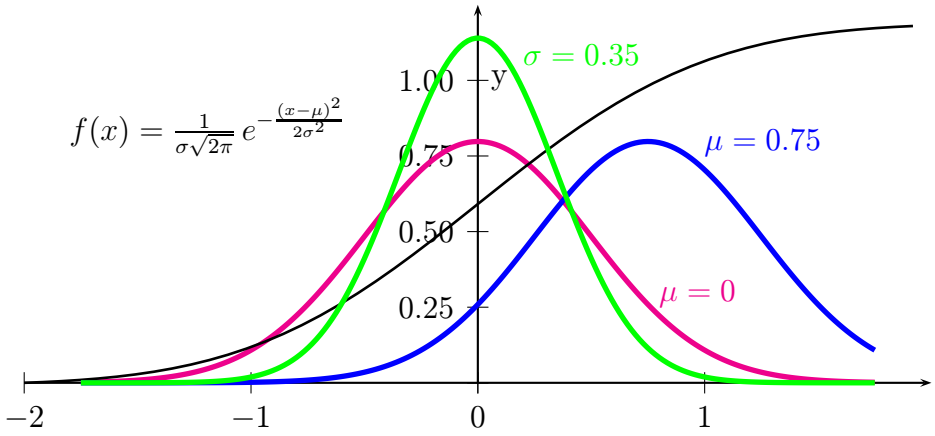
Besselova funkce je řešením Besselovi rovnice a syntaxe pro její vykreslení je: `\psBessel[`*options*`]{`*order*`}{`*xStart*`}{`*xEnd*`}`, kde *order* určuje řád funkce. Volitelnými parametry jsou `constI`=*hodnota* a `constII`= *funkce* v následujícím významu: $f(t) = constI \cdot J_n + constII$, přičemž `constII`=*funkce* se musí vyjádřit PostScriptem. Např. $f(t) = 2.3 \cdot J_n + 1.2 \cdot \sin t + 0.37$ se zapíše pomocí PostScriptu jako `[constI=2.3, constII=t k sin 1.2 mul 0.37 add]`, kde `t` je proměnná a `k` je vnitřní procedura, která konvertuje hodnotu `t` z radiánů do stupňů. Jako volitelný parametr se může uvést `plottpoints`=*hodnota*, který určuje počet funkčních hodnot na zvoleném intervalu, přednastaveno je *hodnota*=500.



Obr. 2.13: Besselova funkce $J_n(x) = \frac{1}{\pi}\int_0^{\pi} \cos(x \sin t - nt)dt$

Gaussova funkce

Gaussova funkce se vykreslí příkazem: `\psGauss[`*options*`]{`*xStart*`}{`*xEnd*`}`, kde volitelnými parametry mohou být `sigma`=*hodnota* a `mue`=*hodnota*, přednastavenými hodnotami jsou `sigma`=0.5 a `mue`=0. Parametr `sigma` mění funkci ve vertikálním směru, `mue` způsobí posun v horizontálním směru. Příkaz `\psGaussI` vykreslí integrál, který je vypočítán Simpsonovým algoritmem s volitelným parametrem `Simpson`=*hodnota*, který definuje počet kroků v zvoleném intervalu a předdefinovanou hodnotou je 5.



Obr. 2.14: Gaussova funkce $f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Makro `pst-func`[25] také umožňuje vykreslit histogram pro binomické rozdělení pravděpodobnosti, distribuční funkce pro rozdělení pravděpodobnosti: Poissonovo, Gamma, exponenciální, Student's-t, Beta, F. Dále pak Lamého křivky a elipsy, integrál sinu a cosinu, rotaci funkce kolem osy x a výpis jednotlivých hodnot matematických funkcí.

2.3.6 pst-circ

Balíček `pst-circ`[10] je kolekcí grafických elementů určených pro kreslení elekrických schémat. Patří sem základní značky elektrických komponent, mikrovlnné symboly a logické prvky.

Nejprve je třeba definovat pomocí příkazu `\node(`*x*,*y*`){`*označení*`}` souřadnice, pomocí kterých se budou v elektrickém obvodu jednotlivé prvky umísťovat. Tyto definovaná *označení* se uvedou u konkrétní komponenty a znamenají počáteční a koncový bod jejího umístění, např. `\resistor(`*označení1*`)(`*označení2*`){`*jmenovka*`}`,tyto

souřadnice lze zadat i přímo. Pokud je vzdálenost počátečního a koncového bodu větší než samotný prvek, tak jsou tyto body spojeny vodičem a značka se umístí uprostřed.

Následuje seznam nadefinovaných elektrických komponent, kde jsou pro *označení* jejího umístění voleny velká písmeny abecedy a ve složených závorkách je odpovídající *jmenovka* značky, kterou lze změnit.

`\resistor[dipolestyle=`*typ*`, variable](A)(B){$R$}` – rezistor, volitelným parametrem může být `zigzag`, který vykreslí americkou značku rezistoru nebo `varistor`, `variable` značí proměnný rezistor;

`\capacitor[dipolestyle=`*typ*`, variable](A)(B){$C$}` – kapacitor, volitelný parametr `variable` vykreslí proměnný kapacitor, typem kapacitoru může být `chemical`, `elektorchemical` nebo `elektor`;

`\coil[dipolestyle=`*typ*`, variable](A)(B){$L$}` – induktor, volitelný parametr `variable` vykreslí proměnný induktor, typem může být `rectangle`, tzn. značka ve tvaru černého obdélníku a další způsoby jeho vykreslení: `curved`, `elektor`, `electrocurved`;

`\battery[variable](A)(B){$E$}` – primární článek;

`\Ucc(A)(B){$E$}` – napěťový zdroj;

`\Icc(A)(B){$\eta$}` – proudový zdroj;

`\switch(A)(B){$K$}` – spínací kontakt;

`\diode[dipolestyle=` *typ*`](A)(B){$D$}` – dioda, volitelným parametrem může být `thyristor`, `GTO` nebo `triac`;

`\Zener(A)(B){$D$}` – Zenerova dioda;

`\LED(A)(B){$\mathcal D$}` – LED dioda;

`\lamp(A)(B){{$\mathcal L$}}` – světelný zdroj;

`\circledipole(A)(B){$\mathcal A$}` – pouzdro, prázdná kruhová značka;

`\OA[OApower=true/false,OAinvert=true/false](B)(A)(C)` – operační zesilovač, volitelný parametr `OApower` s hodnotou `true` zakreslí i napájecí nožičky, `OAinvert` s hodnotou `false` umístí neinvertující vstup nad invertující, parametr `OAperfect` s hodnotou `false` nezobrazí u výstupu symbol ∞, parametr `OAipluslabel=$i_+$` vykreslí na vodiči směrem k neinvertujícímu vstupu šipku a jmenovku směru proudu, `OAiminuslabel=$i_-$` směrem k invertujícímu vstupu, `OAioutlabel=$i_o$` směrem od výstupu zesilovače;

`\transistor[basesep=`*délka*`,transistortype=`*polarita*`](B)(A)(C)` – tranzistor, volitelný parametr `basesep` určuje délku vodiče k bázi, `transistortype` je `PNP/NPN`, dalším volitelným parametrem může být `transistorinvert` pro inverzní režim, `transistorcircle=true/false` pro zobrazení/nezobrazení pouzdra, `transistortype=FET` pro tranzistor FET, `FETchanneltype=` *typ* pro

kanál `P/N`, `transistoribaselabel=$i_B$` vykreslí na vodiči směrem k bázi šipku a jmenovku směru proudu, `transistoricollectorlabel=$i_C$` směrem ke kolektoru a `transistoriemitterlabel=$i_E$` směrem od emitoru;

`\Tswitch[tripolestyle=`*hodnota*`](A)(B)(C){$K$}` – spínací kontakt mezi dvěmi částmi obvodu, *hodnota* volitelného parametru je `left` nebo `right`;

`\potentiometer[dipolestyle=`*typ*`](A)(B)(C){$P$}` – potenciometr, volitelným parametrem může být `zigzag` pro vykreslení americké značky rezistoru;

`\transformer(A)(B)(C)(D){$\mathcal T$}` – transformátor, volitelným parametrem může být `transformeriprimarylabel=$i_1$`, který vykreslí na vodiči směrem k vstupní cívce šipku a jmenovku směru proudu, parametr `transformerisecondarylabel=$i_2$` je vykreslí směrem k výstupní cívce;

`\optoCoupler(A)(B)(C)(D){$OC$}` – optočlen;

`\multidipole(A)(B)` – umožňuje umístit mezi dva definované body více obvodových prvků[4], přičemž za posledním prvkem je třeba udělat tečku;

`\wire(A)(B)` – vodič;

`\tension(A)(B){$u$}` – napětí;

`\ground{`*úhel*`}(A)` – uzemnění, parametrem *úhel* lze prvkem otáčet.

Mezi volitelné parametry dále patří:

`labeloffset=0` – umístí jmenovku tohoto prvku dovnitř značky, např. `A` pro ampermetr, jinak se umístí nad tuto značku;

`labelangel=`*úhel* – otočí jmenovku o zvolený *úhel*, pokud se použije symbol `:U` bude umístěna rovnoběžně s obvodovým prvkem;

`labelInside=`*číslo* – umístí dovnitř značky další symboly: *číslo* 1 šipku, *číslo* 2 znaky `+-` a *číslo* 3 znak `=`;

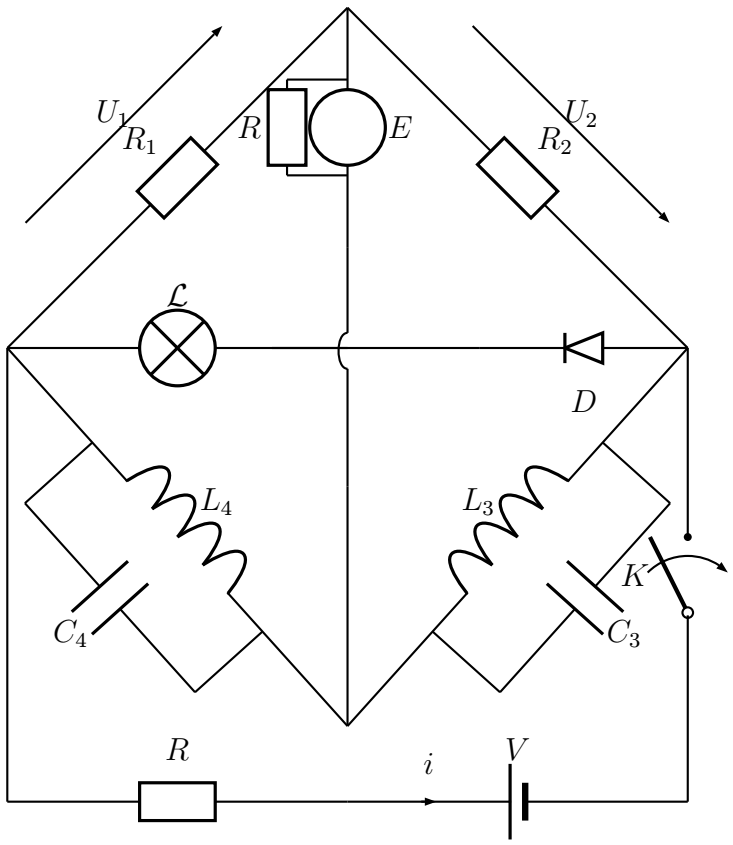
`arrows=o-o` – vykreslí na konci vodičů svorky;

`parallel` – vykreslí prvek v paralerním zapojení k tomu prvku obvodu, který má stejně definované umístění, dalším parametrem je `parallelarm`=*hodnota* určuje vzdálenost mezi dvěma prvky obvodu v paralerním zapojení, přičemž pro hodnotu `1` je vzdálenost rovna velikosti prvku, default=1.5, `parallelnode` vykreslí v uzlech body, `parallelsep`=*hodnota* určuje vzájemnou vzdálenost vodičů, přičemž hodnota `1` je ve středu prvku a `0` je 3× šířka prvku;

`intersect,intersectA=A,intersectB=B` – v místě obvodu, kde dochází ke křížení dvou vodičů, vykreslí část jednoho z těchto vodičů obloukem (viz obr. 2.15), dále se definuje dvěma parametry umístění vodiče, který má být překřížen;

[4] pokud se zvolí nevhodně vzdálenost dvou bodů, tak se kreslí prvky přes sebe

`intensity, tension` – parametr `intensity` vykreslí u zvoleného prvku obvodu na vodiči šipku ve směru proudu (po směru hodinových ručiček), parametr `directconvention=false` je pro opačný směr , `intensitylabel=$i$` navíc nad směrovou šipkou zobrazí jmenovku a `intensitylabeloffset=`*hodnota* umožňuje měnit polohu jmenovky, parametr `tension` vykreslí nad prvkem obvodu šipku ve směru napětí, parametr `dipoleconvention=generator` je pro opačný směr, než je směr proudu a hodnota `receptor` je pro souhlasný směr, `tensionlabel=$u$` zobrazí u šipky jmenovku, `tensionlabel=`*hodnota* a `tensionlabeloffset=`*hodnota* umožňují měnit polohu šipky a jmenovky;



Obr. 2.15: Ukázka zapojení můstku

Logické obvody

Syntaxe pro všechny logické členy je: `\logic[`*options*`](`*x,y*`){`*jmenovka*`}`, kde *x,y* je volitelným parametrem pro umístění obvodu a značí levý dolní roh, default=`(0,0)`.

Volitelným parametrem *options* může být:

`logicShowNode` = `true/false` – očísluje vstupy a výstup označí `Q`, default=false;

`logicNodestyle` = *příkaz* – určuje velikost písma předchozího parametru pomocí standardních příkazů LaTeXu, např. `\large`, default= `\footnotesize`;

`logicShowDot` = `true/false` – vykreslí u vstupů a výstupu bodově svorky, default=false;

`logicSymbolstyle` = *příkaz* – určuje velikost písma logického symbolu příkazem LaTeXu, default=`\large`;

`logicSymbolpos` = *hodnota* – určuje pozici předchozího parametru, default=`0.5`;

`logicType` = *typ* – určuje typ logického členu, kterým může být: `and` (&), `nand` (&), `or` (≥1), `nor` (≥1), `not` (1), *typ* pro exkluzivní OR je `exor` (=1) a pro exkluzivní NOR je `exnor` (=), *typ* pro klopné obvody: `RS`, `D`, `JK`, pokud se za typem uvedou uvozovky, např. `or"`, vykreslí vždy jen ampersand (&), default= `and`;

`logicLabelstyle` = *příkaz* – určuje velikost písma jmenovky, default=`\small`;

`logicChangeLR` = `true/false` – obrátí připojení vstupů a výstupu, default= `false` je pro vstupy umístěné vlevo, výstup je na pravé straně;

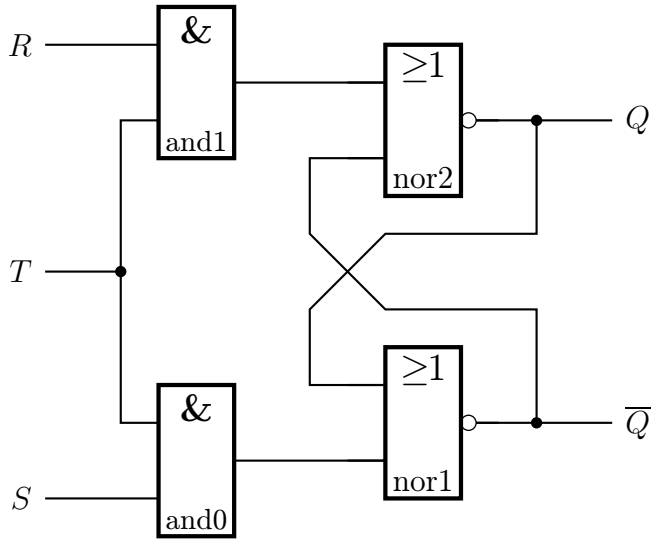
`logicWidth` = *šířka* – nastaví šířku logického členu, default=1.5;

`logicHeigth` = *výška* – nastaví výšku logického členu, default=2.5;

`logicWireLength` = *délka* – nastaví délku vodičů pro vstupy a výstup, default=0.5;

`logicNInput` = *počet* – definuje počet vstupů, default=2;

`logicJInput` = *počet*, `logicKInput` = *počet* – definuje počet vstupů pro klopné obvodu JK, default=2;



Obr. 2.16: Ukázka zapojení klopného obvodu RS s řídícím vstupem T

Pro vykreslení spojovacího vodiče z výstupu k vstupu je možno použít příkaz: `\ncbar[angleA=0,angleB=180]{`*jmenovka1*`}{`*jmenovka2*`}`, kde `angleA=0` je úhel

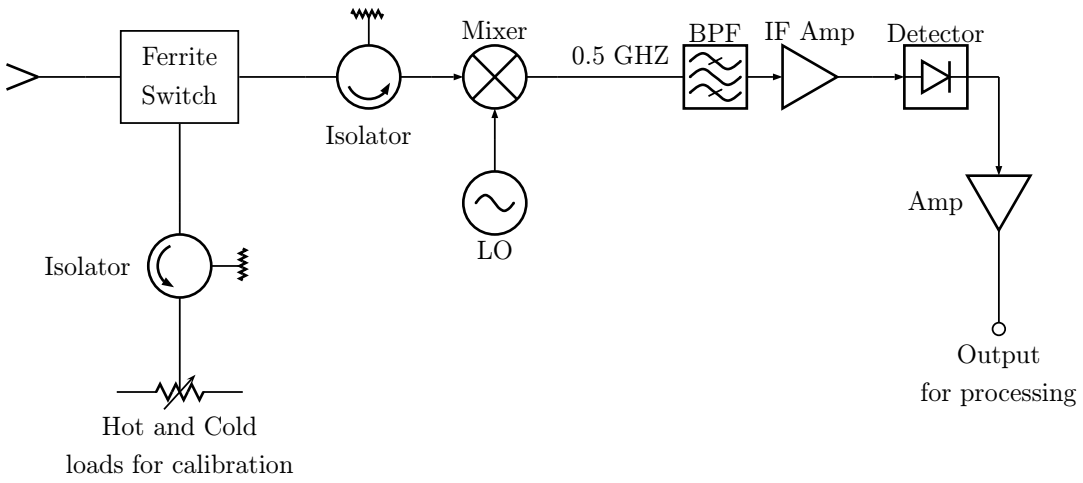
pro směr vodiče u vstupu a `angleB=180` pro směr vodiče u výstupu, přičemž default hodnoty jsou uvedeny v příkazu a znamenají horizontální pozici, dalším parametrem jsou `arrows=*-*` pro vykreslení uzlů, `dotsize=`*hodnota* je pro definování velikosti bodů v těchto uzlech a `arm=`*hodnota* definuje vzdálenost od logického členu.

Mikrovlné symboly

Nejnovější aktualizace balíčku `pst-circ` umožňuje vykreslit mikrovlnné symboly. Patří sem např. anténa, oscilátor, filtry, izolátor, frekvenční dělič/násobič, posunovač fáze, VCO (generátor, jehož kmitočet je řízen napětím), zesilovač a další. Ukázka diagramu viz. obr. 2.17

Je zde také uveden nový způsob zobrazení značky uzemnění pomocí příkazu: `\newground[groundstyle=`*styl*`]{`*úhel*`}(A)`, kde *styl* může být `triangle`, který vykreslí v spodní části značky trojúhelník, `ads` vykreslí všeobecnou značku a `old` má značku stejného tvaru jako příkaz `ground`, tzn. dvě na sebe kolmé čáry.

Veškeré nové možnosti tohoto balíčku jsou ukázány na příkladech v dokumentaci[10].



Obr. 2.17: Ukázka blokového schéma radiometru

2.4 Třída Beamer

Vedle potřeby začleňovat grafiku do dokumentů je v současnosti rozšířenou praxí vytvářet elektronické prezentace. K jejich tvorbě existují různé nástroje, např. OpenOffice.org Impress nebo komerční produkt Microsoft PowerPoint. Systém LaTeX nabízí k tvorbě prezentací vlastní třídu `slides` a různé balíčky, kterými lze vytvářet kvalitně zpracované PDF prezentace. Jedním z těchto balíčku je Beamer, jehož vlastnosti a způsob tvorby prezentace jsou popsány v této kapitole.

2.4.1 Vlastnosti třídy Beamer

Beamer je třída LaTeXu určená k vytváření prezentací. Tato prezentace se vytvoří jako každý jiný dokument LaTeXu a lze tak při její tvorbě použít všech jeho vlastností. Samotný text prezentace tak lze začlenit do jiného dokumentu LaTeXu. Mezi nejdůležitější vlastnosti této třídy patří:

- Výstupním formátem je PDF, který je možné vytvořit přímo pomocí programu pdfLaTeX nebo cestou LaTeX → ovladač `dvips` → PDF soubor.
- Obsahuje nadefinované šablony a barevné schémata.
- Automatické vytvoření navigačního panelu s odkazy sekcí a podsekcí na odpovídající snímky u vybraných šablon.
- Jednoduše lze vytvářet postupné vykreslování snímku (*overlays*) a dynamické efekty.
- Vzhled, barvy a typ písma lze změnit globálně nebo lokálně.
- Součástí jsou ukázkové i vytvořené vzorové prezentace, určené k okamžitému použití. Uživatel se tak může od první chvíle soustředit především na samotný obsah prezentace.
- Obsáhlá a kvalitně zpracovaná dokumentace [20].

2.4.2 Vzhled prezentace

Na začátku samotné tvorby prezentace je třeba zvolit její vzhled. Třída Beamer používá pět rozdílných druhů témat, kterými jej lze jednoduše měnit.

Téma prezentace Příkaz `\usetheme[`*volby*`]{`*téma*`}` definuje, jak bude vypadat každý detail prezentace. Na výběr jsou témata bez navigační lišty nebo s navigační lištou (viz obr. 2.18), které se ještě liší jejím typem a umístěním. Pomocí volitelného parametru lze navíc u některých témat měnit polohu navigační lišty, nastavovat její šířku apod.

Barevné téma Příkaz `\usecolortheme[`*volby*`]{`*téma*`}` definuje, které barvy se v prezentaci použijí. Je možné specifikovat barvu pro každý element anebo použít předdefinovaná barevné schémata. K dispozici jsou komletní schémata, které specifikují všechny barvy ve všech použitých částech prezentace nebo lze definovat barvy pro vnitřní a vnější témata.

Téma fontu Příkaz `\usefonttheme[`*volby*`]{`*téma*`}` definuje, které fonty nebo jejich atributy budou použité v prezentaci. Na výběr jsou nadefinované témata nebo je možné samostatně změnit velikost písma, jeho rodinu a další.

Vnitřní téma Příkaz `\useinnertheme[`*volby*`]{`*téma*`}` definuje vzhled elementů uvnitř jednotlivých snímků. Mezi tyto elementy patří bloky textu, výčtová prostředí (`itemize, enumerate, description`), poznámky, obrázky, tabulky a další. V prostředí `itemize` lze např. měnit tvar značky pro položky výčtu.

Vnější téma Příkaz `\useoutertheme[`*volby*`]{`*téma*`}` definuje vzhled vnějších elementů, které tvoří vzhled a strukturu prezentace. Patří sem záhlaví, zápatí, rámeček pro téma prezentace, logo, navigační prvky. Také lze definovat, které vnější elementy budou v prezentaci zastoupeny.

Veškeré možné *témy* a jejich případné *volby* jsou přehledně vyobrazeny a popsány v dokumentaci [20].

Třída Beamer má také vlastní příkaz na předefinování jakékoliv položky. Syntaxe je `\setbeamertemplate{`*element*`}{`*definice*`}`. Předdefinovanými vzory jsou čtverec `[square]`, trojúhelník `[triangle]` a kruh `[circle]`.

2.4.3 Základní části prezentace

Mezi základní části, které prezentace může obsahovat, patří úvodní stránka, osnova a snímky. Snímky může tvořit text, výčtové prostředí, tabulka, obrázek, video atd.

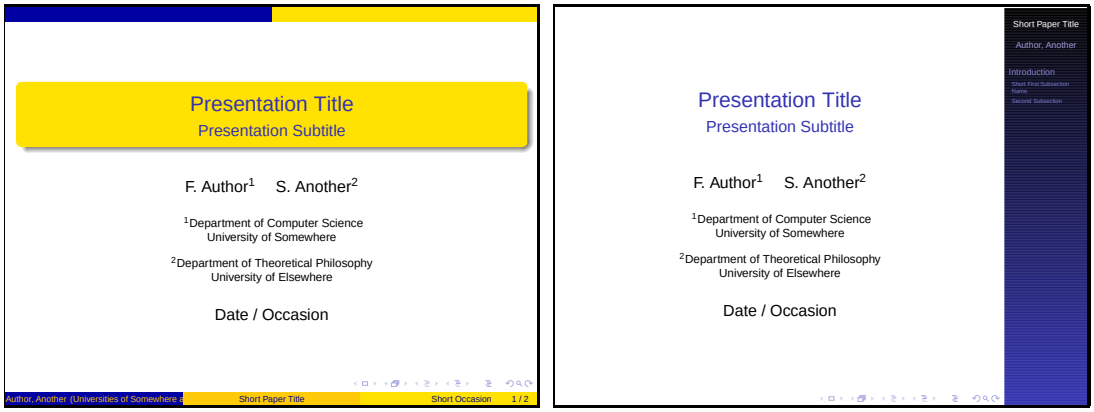
Titulní stránka Její obsah se vytvoří příkazy, které definují téma prezentace `\title{`*název*`}`, jméno autora `\author{`*jméno*`}`, datum `\date{`datum`}` a název ústavu nebo společnosti `\institute[`*zkratka*`]{`*název*`}`. Příkazy pro jméno autora a téma prezentace mohou ještě obsahovat nepovinný parametr, který jejich obsah umístí do záhlaví nebo zápatí. *Ukázka titulní strany* viz obr. 2.18. Úvodní strana se vytvoří příkazem `\titlepage`, který je uzavřený v prostředí `frame`.

Osnova prezentace Následuje za titulní stranou a vytvoří se příkazem pro vysázení nadpisu snímku `\frametitle{`*název*`}` a příkazem pro vytvoření osnovy `\tableofcontents[`*pausesections*`]`, kde nepovinný parametr zajistí postupné zobrazení jejich jednotlivých bodů. Lze ještě definovat podnadpis pomocí příkazu `\framesubtitle{`*název*`}`. Osnova je také uzavřena v prostředí `frame`.

Poté je třeba vytvořit její strukturu, již mimo prostředí `frame`, použitím příkazů `\section{`*název*`}`, `\subsection{`*název*`}` a `\subsubsection{`*název*`}`. Ty mohou také obsahovat volitelný parametr, který umístí jeho obsah do záhlaví. Sekce se pak skládají ze snímků (slajdů), které tvoří obsah celé prezentace.

Tvorba snímků Obsah snímku je vhodné umístit do některého prostředí LaTeXu. Lze použít např. prostředí pro tvorbu tabulek (`table`), obrázků (`figure`) nebo je možné použít předdefinovaná prostředí třídy Beamer. Nejčastěji se však použije některé výčtové prostředí LaTeXu (`enumerate`, `itemize`, `description`). To se umístí do prostředí `frame` a vytvoří k němu nadpis snímku příkazem `\frametitle{`*název*`}`, případně podnadpis `\framesubtitle{`*název*`}`. Jednotlivé položky nebo slova lze barevně odlišit a to buď použitím příkazů balíčku `color` nebo pomocí již zmiňovaných vnitřních témat. Samotná třída Beamer má dispozici příkaz `\alert{`*text*`}`, který změní barvu *text* na červenou. Také má vlastní prostředí `block`, které okolo svého obsahu vytvoří rámeček. K dispozici jsou i další prostředí určené pro teorémy, příklady, definice a další.

K vytvoření prezentace lze také použít vzorová řešení, která se nachází ve složce `beamer\solutions` a jsou zde na výběr tři varianty, které se liší svým rozsahem.



a) b)

Obr. 2.18: Ukázka titulní stránky: a) bez navigační lišty, b) s navigační lištou

2.4.4 Postupné vykreslování snímku

Tento efekt se vytváří tak, že se jednotlivá překrytí snímku rozdělí do samostatných stránek. K dispozici je několik způsobů jak toho docílit.

1. Pomocí příkazu `\pause` se zastaví načítání snímku a zobrazí se jen ta část, která předchází tomuto příkazu. Po stisknutí klávesy se zobrazí další položky až do následujího příkazu `\pause` nebo do konce prostředí.

2. Pokud chceme postupně zobrazovat všechny položky ve výčtovém prostředí, stačí použít inkrementační zápis `\begin{`*prostředí*`}[<+->]`. Hodnota čítače se tak u každé položky výčtu zvýší o 1. Lze také použít zápis `\item<.->`, čímž dojde u dané položky k dekrementaci, tj. ke snížení hodnoty čítače o 1.
3. Další možností je specifikovat rozsah překrytí příkazem `\item<`*rozsah*`>`, který určuje na kterém překrytí se daná položka zobrazí. Např. `\item<1-2>` se zobrazí v prvních dvou překrytí, `\item<2->` se zobrazí od druhého překrytí ve všech zbývajících, `\item<-2,4->` se zobrazí ve všech překrytích mimo třetí. Pokud je třeba zvýraznit některou položku červeně, doplní se specifikace rozsahu pomocí příkazu `alert` následovně: `\item<`*rozsah*`|alert@`*rozsah*`>`.

K výše uvedeným způsobům vytvoření `overlays` ještě existuje několik specifických příkazů:

`\onslide<`*specifikace*`>{`*text*`}` – *text* se zobrazí jen v definovaných překrytích, na ostatních překrytích bude *text* zašedlý;

`\only<`*specifikace*`>{`*text*`}` – *text* se zobrazí jen v definovaných překrytích, na ostatních překrytích tak lze na stejné místo zobrazit jiný text;

`\visible<`*specifikace*`>{`*text*`}` – *text* se zobrazí jen v definovaných překrytích, na ostatních překrytích bude *text* neviditelný;

`\alt<`*specifikace*`>{`*default-text*`}{`*alternativní-text*`}` – zobrazí *default-text*, *specifikace* udává na kterém překrytí se zobrazí *alternativní-text*.

K výše uvedeným příkazům existují ještě další, s alternativním způsobem zápisu, které lze najít v dokumentaci [20].

Pokud je třeba prezentaci vytisknout, zadá se ve specifikaci třídy nepovinný parametr `handout` a postupné vykreslování snímků bude zrušeno.

2.4.5 Přechody snímků

Přechod mezi jednotlivými snímky může být náhlý nebo realizován pomocí vizuálních efektů. V třídě Beamer jsou na výběr následující efekty:

`\transblindshorizontal<`*specifikace*`>[`*volby*`]` – snímek se vykreslí horizontálními pruhy;

`\transblindsvertical<`*specifikace*`>[`*volby*`]` – snímek se vykreslí vertikálními pruhy;

`\transboxin<`*specifikace*`>[`*volby*`]` – snímek se vykreslí ze všech stran do jeho středu;

`\transboxout<`*specifikace*`>[`*volby*`]` – snímek se vykreslí z jeho středu do všech stran;

`\transdissolve<`*specifikace*`>[`*volby*`]` – snímek se vykreslí rozplynutím kostiček;

`\transglitter<`*specifikace*`>[`*volby*`]` – stejný jako předchozí efekt + přechod zleva doprava;

`\transsplitverticalin<`*specifikace*`>[`*volby*`]` – snímek se vykreslí z levé a pravé strany do středu;

`\transsplitverticalout<`*specifikace*`>[`*volby*`]` – snímek se vykreslí ze středu do levé a pravé strany;

`\transsplithorizontalin<`*specifikace*`>[`*volby*`]` – snímek se vykreslí z vrchní a spodní strany do středu;

`\transsplithorizontalout<`*specifikace*`>[`*volby*`]` – snímek se vykreslí ze středu do vrchní a spodní strany;

`\transwipe<`*specifikace*`>[`*volby*`]` – snímek se vykreslí z vrchní na spodní stranu;

`\transduration<`*specifikace*`>{`*počet-sekund*`}` – *počet-sekund* určuje délku zobrazení snímku.

Výše uvedené příkazy se uskuteční pouze v režimu zobrazení na celou obrazovku. *Volby* mohou být dvě: délka efektu `duration`= *počet sekund* a směr přechodu `direction`= *úhel*. Pro *úhel* jsou povolené hodnoty 0, 90, 180, 270 a pro efekt `\transglitter` i 315.

Třída Beamer umožňuje použít v prezentaci i další efekty, např. lze vytvářet animace, vkládat videa, zvuky atd. Příkazy pro tvorbu těchto efektů anebo začlenění multimediálních souborů do prezentace jsou uvedeny v dokumentaci [20].

2.5 Konverze SVG grafiky do PSTricks

2.5.1 XML

XML, tj. rozšiřitelný značkovací jazyk, je standardem schváleným konsorciem W3C pro značkování dokumentů. Definuje obecnou syntaxi a ustanovuje standardní formát pro počítačové dokumenty. Tento formát je natolik flexibilní, že jej lze dále upravit pro tak různorodé oblasti, jako jsou webové stránky, elektronická výměna dat, vektorová grafika, genealogie, aplikace katastrů nemovitostí, serializace objektů, vzdálená volání procedur, či systémy hlasové pošty.

XML je meta-značkovací jazyk[5] pro textové dokumenty. Data jsou v dokumentech XML obsažena ve formě textových řetězců, jež jsou uzavřeny do textových značek, které tato data popisují. Konkrétní textový řetězec spolu s příslušnou značkou se označuje jako element. Ačkoliv je XML flexibilní v tom smyslu, že umožňuje definovat libovolné elementy, je v mnoha dalších ohledech striktní. Definuje gramatiku dokumentů XML, která určuje umístění, kde se značky mohou objevit, které názvy elementů jsou platné, jak se k elementům přiřazují atributy, a tak dále. Tato gramatika je jasně definována do té míry, aby bylo možné vyvinout analyzátory XML (tzv. parsery), které dokážou číst a porozumět libovolnému dokumentu, napsanému v XML. Dokumentům, které této gramatice vyhovují, se říká správně formulované dokumenty. Analyzátor XML odpovídá za rozdělení dokumentu na jednotlivé elementy, atributy a ostatní části. Obsah dokumentu XML pak předává aplikačnímu programu část po části. Jestliže analyzátor v libovolném místě narazí na porušení pravidel jazyka XML, nahlásí tuto chybu aplikaci a rozklad dokumentu ukončí, tzn. přestane předávat obsah jakýchkoliv dalších elementů a atributů aplikaci.

Značky, které jsou v konkrétní aplikaci XML povoleny, se dokumentují prostřednictvím definice typu dokumentu (DTD). Definice DTD jsou platnou aplikací XML a určují, kde a jak mohou být jednotlivé značky v dokumentu použity. Oproti DTD se pak porovnávají konkrétní dokumenty. Dokumentům, které konkrétnímu DTD vyhovují, se říká, že jsou platné, naopak ty, které DTD nevyhovují, jsou neplatné. Existují proto tzv. validační analyzátory XML, které porovnávají dokument s jeho definicí DTD hned při čtení, aby se mohly přesvědčit, zda dokument vyhovuje určeným omezujícím pravidlům. Porušení stanovených pravidel se označuje jako validační chyba a celý proces kontroly dokumentu oproti jeho DTD se nazývá validace. Jestliže validační analyzátor narazí na validační chybu, nahlásí tuto chybu aplikaci, pro niž provádí rozklad dokumentu. Aplikace sama se pak může rozhodnout, zda si přeje v rozkladu dokumentu pokračovat či nikoliv. Validační chyby, na rozdíl od chyb vzešlých z porušení správné formulace, nejsou nutně fatální - aplikace se může

[5] nedefinuje pevně danou množinu značek a elementů

rozhodnout, že je bude ignorovat. Použití DTD je tedy v XML nepovinné.

XML nabízí možnost tvorby skutečně na platformě nezávislých a dlouhodobých datových formátů. Dokumenty v XML jsou textové a dokáže je tedy přečíst jakýkoliv program, který umí číst textové soubory. Textově jsou uložena jak data, tak i značkování, přičemž značkování je v souboru XML uloženo v podobě značek. V mnoha ohledech je XML nejpřenositelnější a nejflexibilnější formát dokumentů, jaký byl navržen od vzniku textového ASCII-souboru.

2.5.2 SVG

Vektorový grafický formát SVG byl navržen pro oblast webové grafiky, není to však zdaleka jediná oblast, ve které je možné se s tímto formátem setkat. Právě naopak: tento formát se postupně stává průmyslovým standardem pro přenos vektorové grafiky mezi různými platformami i aplikacemi. Dochází také k integraci knihoven pro práci se SVG do grafických uživatelských rozhraní operačních systémů. Vzhledem k tomu, že se implementace celého SVG ukázala jako poměrně problematická, vznikly dvě podmnožiny SVG (SVG Basic a SVG Tiny), přičemž se předpokládá, že aplikace určené například pro mobilní zařízení s omezeným výpočetním výkonem a menší kapacitou paměti budou používat právě tyto méně náročné, ale pro mnoho služeb více než dostačující, podmnožiny celého standardu.

Z uživatelského hlediska se jedná o plnohodnotný vektorový formát podporující základní geometrické tvary, cesty (obecné křivky), pokročilou práci s textem, průhlednost apod. Jedná se o vlastnosti, které můžeme v prakticky stejné podobě najít například i v PostScriptu či PDF. SVG však umožňuje i tvorbu animací a především interaktivitu – SVG tedy nemusí sloužit pouze k zobrazování statických či animovaných obrázků, ale může se nad ním vybudovat například interaktivní mapový portál, geografický informační systém, jednodušší hry, grafické editory integrované do HTML stránek apod. To vše bez nutnosti vázat se na jednoho dodavatele technologie, jeden prohlížeč či platformu. SVG soubory je možné komprimovat pomocí GZIPu a tím dále snížit velikost celého souboru, což se příznivě projeví zejména při pomalejším síťovém připojení.

Pro SVG jsou v současnosti dostupné zásuvné moduly do webových prohlížečů a rozšiřují se webové prohlížeče, které SVG podporují nativně, tj. bez nutnosti instalace zásuvného modulu. Prakticky všechny významné vektorové editory dnes SVG podporují. Jedná se jak o komerční produkty (Adobe Illustrator, CorelDraw, Xara pro Linux, Sketsa…) i o programy šířené jako open source (Inkscape, Sodipodi, Scribus…). Taktéž existuje mnoho programů, které dokáží SVG importovat či exportovat (OpenOffice.org, FreeMind, animační programy…).

SVG je zajímavý i z programátorského hlediska. Jedná se totiž o XML aplikaci,

tj. grafické objekty i přidružené informace (například animace) jsou uloženy v XML formátu odpovídajícímu oficiálně vydané specifikaci. Každý SVG dokument je možné před vlastním zpracováním zkontrolovat na validitu, což zjednodušuje další práci. Díky XML je možné (s využitím DOM – Document Object Modelu) měnit, přidávat či ubírat jednotlivé uzly stromu, ze kterého se dokument skládá a ovlivňovat tak výslednou podobu kresby.

2.5.3 XSLT

Stylový jazyk XSL je rozdělen do dvou částí: na XSL Transformations (XSLT) a XSL Formatting Objects (XSL-FO).

XSLT, je aplikace XML, která specifikuje pravidla, na jejichž základě se provádí převody mezi dokumenty v jednom formátu XML na dokumenty v jiném formátu. Dokument v XSLT - což je v podstatě stylová šablona - obsahuje sadu vzorů. Procesor XSLT pak elementy ve vstupním dokumentu porovnává se vzory ve stylové šabloně, a když narazí na odpovídající vzor, zapíše šablonu příslušného vzoru do výstupního stromu. Po dokončení těchto kroků může výstupní strom serializovat buď do jiného dokumentu v XML, nebo do libovolného dalšího formátu, kterým může být i holý text nebo HTML. Procesor XSLT je počítačový software, který může být zabudován do webového prohlížeče, XML editoru nebo do vývojového prostředí, případně to může samostatný program, který se spouští z příkazového řádku, jako je například SAXON od Michaela Kaye[11] nebo XML projekt Xalan[21] od Apache.

Pro identifikaci odpovídajících si elementů XSLT využívá syntaxi jazyku XPath. Dokument v XML tvoří stromovou strukturu, která sestává z uzlů. Některé uzly obsahují další uzly, přičemž jeden kořenový uzel vždy obsahuje všechny ostatní uzly. XPath se na uzly odkazuje na základě jejich pozice, relativní pozice, typu, obsahu a několika dalších kritérií. Výrazy XPath se používají v XSLT pro vyhledání a výběr konkrétních elementů ve vstupním dokumentu, jež mají být zkopírovány do výstupního dokumentu nebo jinak zpracovány. Výrazy XPath mohou reprezentovat rovněž čísla, řetězce nebo logické hodnoty, stylové šablony mohou využívat jednoduchou aritmetiku například pro číslování a tvorbu seznamů obrázků, tabulek nebo rovnic. Díky manipulaci s řetězci v XPath zase XSLT může např. vykonávat takové úkony, jako je převedení nadpisů kapitol na velká písmena, zatímco v ostatním textu se pro stejné nadpisy zachová použití velkých i malých písmen.

XSLT je funkcionálním programovacím jazykem. Klíčem k transformaci dokumentů XML za použití XSLT je vytvořit vyhovující předlohy k elementům ve vstupním dokumentu. Tyto předlohy mohou obsahovat jak pevně daná výstupní data, tak i elementy XSLT, které říkají procesoru XSLT, kam má umístit jaká další data. Tato jednoduchá myšlenka představuje základ pro vše ostatní, co v XSLT děláte.

2.5.4 Stylová šablona XSLT

Cílem praktické části práce bylo navrhnout strukturu SVG souboru, který zobrazuje elektrické obvody nebo vývojové diagramy a vytvořit příslušnou konverzní šablonu XSLT pro konverzi těchto SVG souborů do PSTricks. SVG grafické soubory v současnosti nelze přímo vkládat do dokumentů LaTeXu. Bylo tedy třeba z celého SVG vybrat určitou podmnožinu, která by byla pro tvorbu schémat a diagramů dostatečná. Nejedná se tedy o konverzní šablonu celé SVG grafiky. Nicméně pokud jakýkoliv SVG soubor vyhovuje navržené struktuře, lze zkonvertovat do PSTricks i na první pohled složitější grafiku (viz obr. 2.20). Je třeba si také uvědomit, že některé SVG elementy nelze do PSTricks zkonvertovat vůbec. Jedná se např. o nastavení průhlednosti, které PostScript Level 2 nepodporuje, animace, skripty atd. V této kapitole nebude popisována celá SVG grafika, ale jen ty její části, které se týkají navržené struktury SVG souboru a mohou být konverzní šablonou zkonvertovány do PSTricks.

Struktura SVG

Základním elementem je `svg`, který představuje kořenový element a může obsahovat libovolný počet grafických elementů. `svg` element má atributy, které specifikují jeho další vlastnosti. Patří se šířka a výška kreslícího plátna, atribut `viewbox`, který umožňuje přizpůsobovat grafiku velikosti okna obrazovky a dále počátek souřadnic, který je umístěn v levé horním rohu.

V makru PSTricks se referenční bod nachází v levém spodním rohu. Konverzní šablona tuto odlišnost s SVG řeší tím, že souřadnice y odesílá do výstupního dokumentu se zápornou hodnotou a tím se posune referenční bod na levý spodní roh. Absenci alternativní příkazu `viewbox` v PSTricks řeší konverzní šablona tím, že vygeneruje na pátý řádek výstupního dokumentu příkaz PSTricks `psset`, pomocí kterého se obrázek po konverzi může zmenšovat či zvětšovat. Posun obrázku se řeší pomocí příkazu `rput`, který se nachází na osmém řádku výstupního dokumentu.

SVG umožňuje, v rámci úsporného zápisu, nastavit pomocí elementu `g` ostatním elementům společné vlastnosti, např. atribut `fill` pro barvu výplně. Všechny SVG elementy, pro které je společná vlastnost určena, se tak stanou synovskými elementy `g`, čím se změní struktura stromu. Element `g` může být navíc i několikrát zanořený. Konverzní šablona pro tento účel využívá osu `ancestor-or-self::`, pomocí které prochází zpětně všechny předchůdce kontextového uzlu i včetně kontextového uzlu samého. Nastavením vhodných podmínek a pomocí XSLT elementu `for-each` se vyhledávají jen požadované hodnoty atributů ve všech uzlech.

Cesty

Pod pojmem cesta si lze představit čáru tvořenou z úseček a křivek. V SVG grafice jsou cesty vykresleny pomocí elementu `path`. Tento element má atribut `d`, jehož obsahem je řetězec znaků. Tento řetězec je tvořen příkazy a souřadnicemi všech bodů, které cestu vytváří. Příkazy v řetězci jsou zapisovány jedním písmenem, kde velké písmeno znamená absolutní hodnotu souřadnice a malé písmeno relativní. Hodnota relativní souřadnice vyjadřuje vzdálenost od předchozí pozice absolutní souřadnice. SVG umožňuje úsporný zápis řetězce a to především z důvodu přenosu dat po internetu. Při implementaci konverzní šablony bylo třeba akceptovat všechny možné způsoby zápisu tohoto řetězce, protože `path` patří k základním a často používaným SVG elementům.

Řetězec v atributu `d` může být zapsán následujícími způsoby:

`M 100 100 L 200 200` – souřadnice x a y jsou odděleny mezerou;

`M100 100L200 200` – příkazy nejsou odděleny mezerou od hodnoty souřadnice;

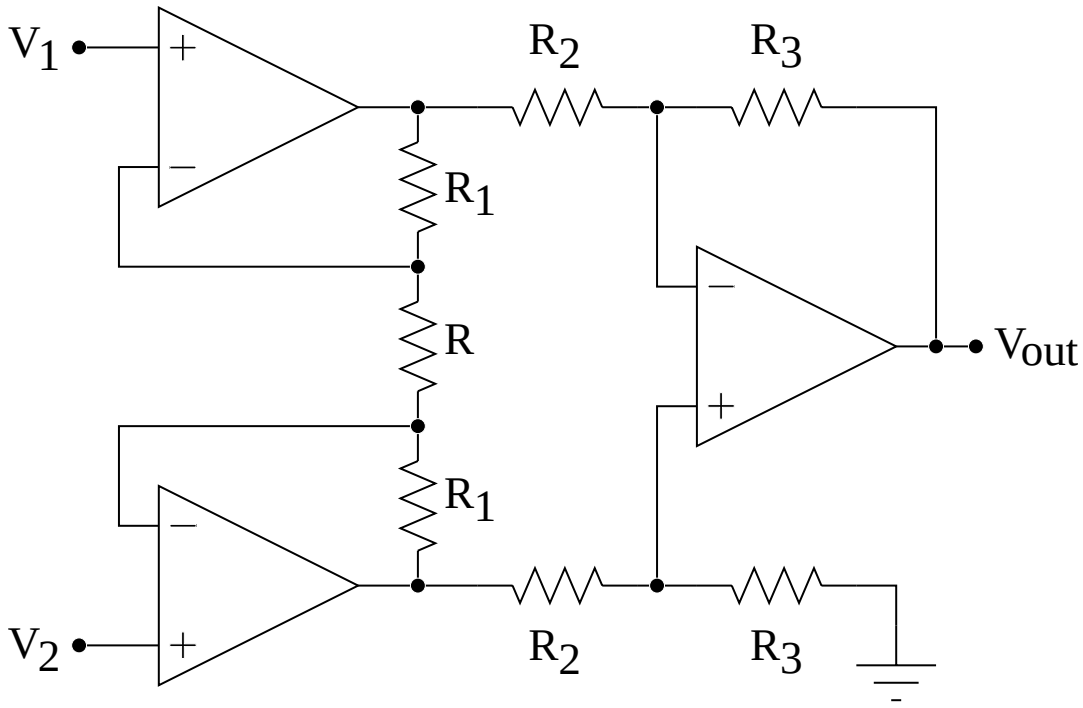
`M 100 200 L 200 100 100 200` – pokud za sebou následují stejné příkazy, nemusí se značky příkazu opakovaně zapisovat;

`M 100-200 L 200-100-100 200` – před zápornou souřadnicí lze mezeru vynechat;

`M 100,100,L 200,200` – mezery lze nahradit čárkami[6].

Všechny uvedené zápisy lze použít v jednom řetězci. V konverzní šabloně se řetězec zpracovává po jednotlivém znaku. Aby bylo možné odlišovat souřadnici x a y, byly vytvořeny tři dílčí šablony, které se při zpracování řetězce postupně volají pomocí XSLT elementu `call-template`. První šablona s názvem `rozbor` ukládá do proměnné `prikaz` znak příkazu, do proměnné `retezec1` ukládá řetězec již bez tohoto znaku a doplňuje u některých příkazů hodnotu souřadnice x. Dále se odešle do výsledného dokumentu název příkazu ve formátu PSTricks, testuje se konec řetězce a výskyt příkazu `Z` nebo `z` pro ukončení cesty. Pokud neklesne délka řetězce pod jeden znak, tak se zavolá šablona `rozbor2`. Šablona `rozbor2` zpracovává souřadnici x po jednotlivém znaku a dokud je znakem číslo, nula, tečka nebo mezera odesílá tyto znaky do výsledného dokumentu. Pokud narazí na jiný znak, následují testy na tento znak z důvodu eliminace případného výskytu dvou oddělujících znaků, tzn. mezery a čárky, nebo pro doplnění hodnoty souřadnice y u některých příkazů. Poté se zavolá šablona `rozbor3`, která na stejném principu jako šablona `rozbor2` zpracuje souřadnici `y`. Za posledním číselným znakem se opět testuje následující znak z důvodu eliminace oddělujících znaků, doplnění pomocného příkazového znaku nebo chybějícího příkazu, pokud byl použit úsporný zápis.

[6] desetinné místo se značí tečkou



Obr. 2.19: Elektrický obvod po konverzi z SVG do PSTricks

Mezi příkazy, které konverzní šablona zkonvertuje patří:

`M,m` (*x y*)+ – `moveto` nastaví aktuální pozici bez jakéhokoliv vykreslení, pro konverzi byl použit příkaz PSTricks `moveto(`*x,y*`)`;

`L,l` (*x y*)+ – `lineto` vykreslí úsečku do zadané souřadnice, pro konverzi byl použit příkaz PSTricks `lineto(`*x,y*`)`;

`H,h` *x*+ – horizontální `lineto`, je zadaná jen souřadnice *x* a znamená horizontální posun na tuto souřadnici s vykreslením úsečky, PSTricks tento příkaz nemá a v konverzní šabloně se používá příkaz `lineto(`*x,y*`)`, pro relativní variantu tohoto příkazu se doplní do řetězce za souřadnici *y* nula, za absolutní hodnotu pak předchozí souřadnice, která se uchovává v pomocných proměnných, v případě, že budou předcházet příkazu `H` relativní souřadnice, kterých může být teoreticky libovolný počet, je nutné hodnotu souřadnice *y* dodatečně opravit ve výsledném dokumentu;

`V,v` *y*+ – vertikální `lineto`, je zadaná jen souřadnice *y* a znamená vertikální posun na tuto souřadnici s vykreslením úsečky, platí zde vše, co bylo napsáno pro příkazy `H,h`, jen s prohozením souřadnic;

`Z,z` – `closepath` uzavře cestu, spojí aktuální bod úsečkou se začátkem cesty, význam obou příkazů je totožný, pro konverzi byl použit příkaz PSTricks `closepath`;

`C,c` (*x1 y1 x2 y2 x y*)+ – `curveto` vykreslí Bézierovu kubickou křivku, pro konverzi byl použit příkaz PSTricks `curveto`(*x1,y1*)(*x2,y2*)(*x3,y3*), první souřadnice je kotvící a druhá souřadnice řídící bod Bézierovy kubické křivky, třetí souřadnice udává koncový bod, počátečním bodem je konec aktuálně vytvářené cesty, konverzní šablona při zpracování těchto dvou příkazů používá pomocné příkazy `W` a `Y`, aby v případě úsporného zápisu byl odeslán do výsledného dokumentu příkaz PSTricks `curveto` za každou třetí souřadnicí, v PSTricks nelze provést úsporný zápis ve stylu SVG.

Existují ještě čtyři dvojice příkazů, které konverzní šablona nezpracuje. Jedná se o příkazy `Q,q` a `T,t` pro vykreslení kvadratické Bézierovy křivky - pro tuto křivku není v PSTricks vytvořené makro. Příkazy `A,a` pro eliptický oblouk nelze zkonvertovat do příkazu PSTricks `psellipticarc`, protože makro pracuje na odlišném principu. Příkazy `S,s` pro Bézierovu kubickou křivky bez zadané souřadnice kotvícího bodu. Tato souřadnice je zrcadlením řídícího bodu z předchozího úseku křivky a i tento příkaz nemá v PSTricks obdobu.

Základní geometrické tvary

V SVG jsou definovány elementy pro vykreslení kružnice, elipsy, obdélníku, čáry, polygonu a lomené čáry. Tyto základní geometrické tvary lze vykreslit i pomocí cest, které umožňují kratší zápis do řetězce, ale obtížnější případnou manipulaci s geometrickým tvarem.

Mezi SVG elementy pro geometrické tvary patří:

`rect` – obdélník, mezi jeho atributy patří `x`, `y` udávající pravý horní roh, `width` a `height` znamenají šířku a výšku obdelníku, pro konverzi byl použit příkaz PSTricks `psframe`(*x0,y0*)(*x1,y1*), kde první souřadnice udává levý dolní roh a druhá pravý horní roh obdélníku, v konverzní šabloně se tak druhá souřadnice musí dopočítat, tento SVG element ještě může obsahovat dva atributy `rx`, `ry` pro zakulacení rohů obdélníku a jejich hodnoty představují poloměry os elipsy, v PSTricks se zaoblují rohy obdélníku pomocí jednoho parametru, který udává poloměr kružnice, konverzní šablona tyto dva SVG atributy tedy nezpracovává, v SVG dále platí, že při nulové šířce nebo výšce se obdélník nevykreslí, tj. není vyditelná ani jeho hrana, konverzní šablona tyto hodnoty testuje a pokud jsou nulové obdelník nezkonvertuje;

`circle` – kruh, mezi jeho atributy patří souřadnice středu `cx`, `cz` a poloměr `r`, pro konverzi byl použit příkaz PSTricks `pscircle`(*x0,y0*)`{`*rádius*`}`, při záporném hodnotě poloměru se kruh nevykreslí;

`ellipse` – elipsa se středem v `cx`, `cz` a poloměry os `rx` a `ry`, pro konverzi byl použit příkaz PSTricks `psellipse(`*x0,y0*`)(`*x1,y1*`)`, při záporných hodnotách poloměrů os se elipsa nevykreslí;

`line` – čára, jejími atributy jsou `x1`, `y1` a `x2`, `y2`, tyto souřadnice udávají její počáteční a koncovou souřadnici, pro konverzi byl použit příkaz PSTricks `psline(`*x0,y0*`)(`*x1,y1*`)`;

`polyline` – lomená čára je geometrický tvar složený z libovolného počtu spojených čar, má jeden atribut `points`, který obsahuje řetězec s hodnotami souřadnic jednotlivých bodů, v kterých dochází k lomení čáry, pro konverzi byl použit příkaz PSTricks `psline(`*x0,y0*`(`*x1,y1*`)`, princip zpracování řetězce je stejný jako u elementu `path` ve zkracené podobě;

`polygon` – mnohoúhelník, tento element se liší od předchozího elementu `polyline` v tom, že je uzavřený, pro konverzi byl použit příkaz PSTricks `psline`, který je vložen mezi příkazy `newpath` a `closepath`.

Obr. 2.20: Obrázek vytvořený z Bézierových kubických křivek

Barvy

SVG grafika používá barevný model RGB, který vychází ze specifikace CCS2[7].

Barvu lze podle CCS2 definovat několika způsoby:

[7] kaskádové styly 2

`název barvy` – jedná se o přibližně 150 předdefinovaných barevných odstínů, v příslušném SVG atributu stačí pak zadat jen název barvy, v konverzní šabloně jsou tyto názvy převedeny na odpovídající hodnoty RGB;

`rgb(r,g,b)` – přímý zápis hodnot jednotlivých barevných složek, tyto hodnoty se zapisují v rozmezí 0-255 nebo mohou být vyjádřeny procentuálně, v PSTricks se rozmezí hodnot nastavuje mezi 0-1, v konverzní šabloně se tak tyto hodnoty musí dělit 255, respektive násobit 0,01 v případě procentuálního vyjádření;

`#rrggbb, #rgb` – hexadecimálním tvar hodnot RGB, který může být napsán i zkráceně, v konverzní šabloně se nejprve testuje délka řetězce, pokud je menší než pět, což nastane v případě zkráceného zápisu, hodnoty RGB se upraví na nezkrácený tvar, jednotlivé barevné složky se převádí do desítkové soustavy po dvojici znaků a hexadecimální čísla reprezentované písmeny jsou nahrazeny odpovídajícími čísly, hodnota v prvním znaku se vynásobí číslem 16 a sečte s číselnou hodnotou druhého znaku, výsledek se pak vydělí číslem 255 pro získání hodnoty v rozmezí 0-1 pro PSTricks.

Styly čar a výplní

V SVG grafice lze u základních geometrických tvarů a cest nastavovat parametry jako je tloušťka čáry, styl čáry, její zakončení, barva výplně apod.

Mezi styly čar a výplní podporovaných konverzní šablonou patří:

`stroke` – atribut, který definuje barvu elementu, pro konverzi byl použit příkaz PSTricks `newrgbcolor{colorStroke}{`$r\ g\ b$`}`, pro přepočet a odeslání hodnot $r\ g\ b$ do výsledného dokumentu se volá šablona `color`, každá specifikovaná barva, jenž je v konverzní šabloně pojmenovaná `colorStroke`, se v příkazu PSTricks `pscustom` přiřadí parametru `linecolor`;

`fill` – atribut, který definuje barvu výplní, pro konverzi byl použit příkaz PSTricks `newrgbcolor{colorFill}{`$r\ g\ b$`}`, pro přepočet a odeslání hodnot $r\ g\ b$ do výsledného dokumentu se volá šablona `color`, každá specifikovaná barva, jenž je v konverzní šabloně pojmenovaná `colorFill`, se v příkazu PSTricks `pscustom` přiřadí parametru `fillcolor`, další parametrem PSTricks je `fillstyle=solid`, přiřazená hodnota `solid` zajistí, že se v PSTricks barevná výplň vykreslí;

`stroke-width` – atribut, který definuje tloušťku objektu, pro konverzi byl použit parametr PSTricks `linewidth`, který se umístí do příkazu `pscustom`;

`stroke-linecap` – atribut, který definuje styl ukončení čáry a může nabývat tří hodnot: `butt` – čára je ukončena v místě koncových bodů, `round` – čára je ukončena kruhovým půlobloukem, tzn. přesahuje původní tvar o tuto část, `square` – čára je ukončena půlčtvercem, tzn. přesahuje původní tvar o tuto

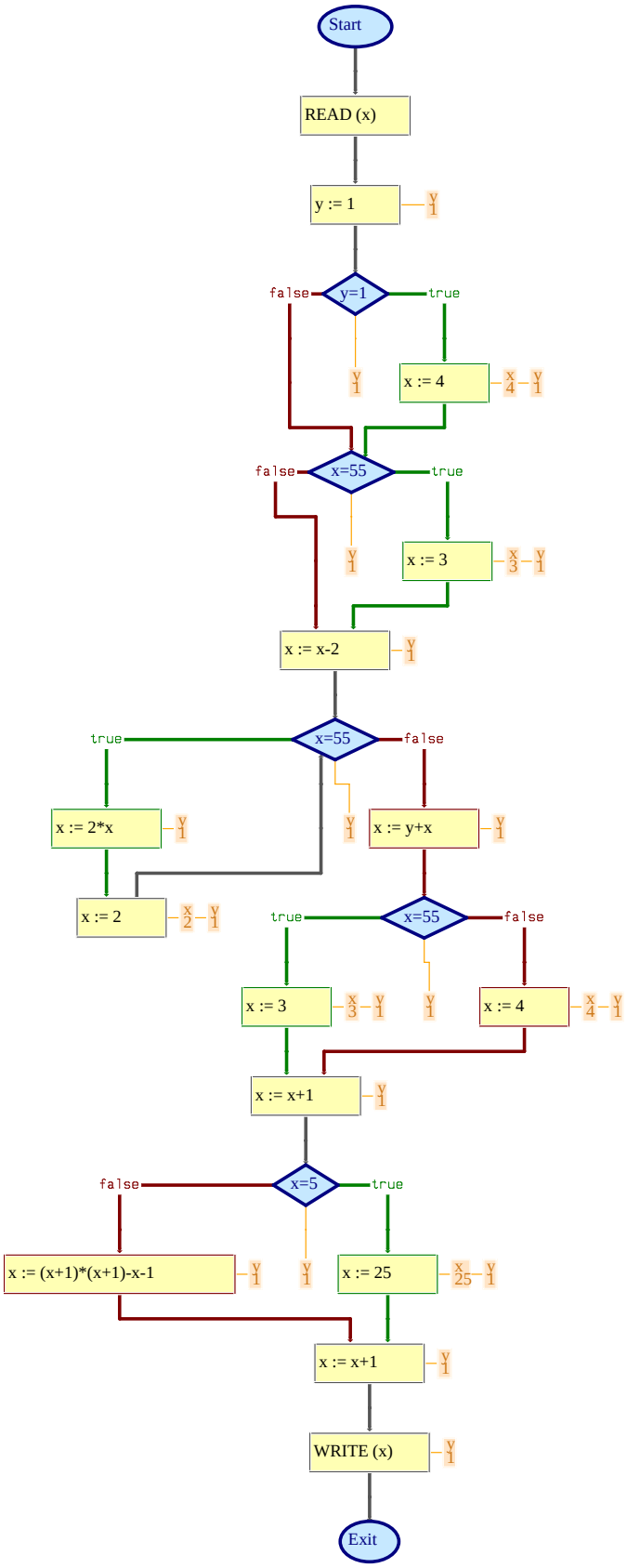
část, pro konverzi byl použit parametr PSTricks `-` pro `butt`, `cc-cc` pro `round` a `C-C` pro `square`;

`stroke-dasharray` – atribut, který vykreslí přerušovanou čáru, jeho obsahem jsou číselné hodnoty, na lichém místě se definuje délka čárky a na sudém délka mezery, pro konverzi byl použit parametr PSTricks `linestyle=dashed`, který nastaví styl čáry a parametr `dash`= *value1 value2 ...*, parametry se umístí do příkazu `pscustom`.

Text

SVG používá pro zobrazení textu v grafice fonty písma specifikované v CSS2. PSTricks používá fonty LaTeXu, případně je možné použít fonty PostScriptu. Fonty používané v SVG grafice tak nelze zkonvertovat do PSTricks. Text je v SVG grafice umístěn jako řetězec v elementu `text`.

Konverzní šablona zkonvertuje atributy `x` a `y` pro umístění textu v obrázku a dále barvu textu obsaženou v atributu `fill`. Pro umístění textu v obrázku používá příkaz PSTricks `rput` a pro barvu textu `color[rgb]{`*r,g,b*`}`. Dále šablona vygeneruje před každý textový řetězec pomocný příkaz `sf`, který umožňuje měnit po konverzi ve výsledném dokumentu globálně nastavení textu. Příkaz se nachází na šestém řádku výsledného dokumentu: `DeclareFixedFont{\sf}{T1}{ptm}{m}{n}{7pt}` a definuje kódování, rodinu, váhu, tvar a stupeň písma. V případě různé velikosti písma v obrázku, např. pro nadpis, se příkaz vymaže a parametry se nastaví dodatečně u konkrétního textu. SVG má sice pro velikost písma atribut `font-size`, ale vzhledem k rozdílnosti fontů písma by výsledek nemusel být optimální a to především u SVG obrázků vyobrazující vývojové diagramy, kde by se text nemusel vejít dovnitř značek a musel by se tak změnit pro každý text samostatně.



Obr. 2.21: Vývojový diagram po konverzi z SVG do PSTricks

3 ZÁVĚR

Bakalářská práce se zabývala možnostmi grafiky a tvorby prezentace v systémech TEX a LATEX. V první části byly popsány možnosti importu grafiky do dokumentů LATEXu, byl zde vysvětlen princip tohoto importu a upozornění na omezení ve výběru formátu obrázku. V další části se práce zabývala vytváření grafiky přímo interními prostředky LATEXu v prostředí `picture`. Jsou zde uvedeny omezení, která pro toto prostředí platí a také, jak je lze do určité míry překonat pomocí doplňkových balíčků a externího programu `gnuplot`. Dále byl popsán princip PSTricks, který využívá k tvorbě grafiky PostScript, a jeho základní nastavení. Pozornost byla věnována především jeho rozšířením pro tvorbu grafů a elektrických obvodů. Předposlední část práce se zabývala popisem třídy Beamer, kterou lze v LATEXu vytvořit kvalitní PDF prezentaci. Vzhledem k tomu, že je třída Beamer značně obsáhlá, jsou zde uvedeny především její hlavní vlastnosti a způsob tvorby prezentace.

V praktické části práce jsem navrhl strukturu zdrojového SVG souboru, který obsahuje elektrické obvody a vývojové diagramy. Implementoval jsem příslušnou konverzní šablonu XSLT, která zajišťuje převod těch souborů do PSTricks a umožňuje tyto grafické soubory využívat v systému LATEX. Konverzní šablona je plně funkční, odzkoušel jsem ji na několika vhodných SVG souborech pomocí stylového procesoru SAXON 9.0.0.5 a na výsledném dokumentu jsem nezjistil žádné nedostatky. Vytvořená stylová šablona může prakticky sloužit pro konverzi elektrických obvodů nebo vývojových diagramů do formátu PSTricks, za předpokladu že budou vyhovovat navržené struktuře zdrojové SVG souboru. Konverzní šablona by se mohla dále vyvíjet tím směrem, že bude zahrnovat větší podmnožinu SVG grafiky.

LITERATURA

[1] Adobe Systems Incorporated *PostScript Language Reference*. 1999, 3. edition. <http://www.adobe.com/products/postscript/pdfs/PLRM.pdf>

[2] CARLISLE, D., P. *Packages in the 'graphics' bundle*. 2005. <http://tug.ctan.org/tex-archive/macros/latex/required/graphics>

[3] CRAWFORD, D. *gnuplot An Interactive Plotting Program*. 2007. <http://www.gnuplot.info/documentation.html>

[4] DUGGAN, A. *PiCTEX command summary*. 1990. <http://ftp.linux.cz/pub/tex/CTAN/info/pictex/summary.zip>

[5] GasTEX *Graphs and Automata Simplified in TeX*. <http://www.lsv.ens-cachan.fr/~gastin/gastex/gastex.html>

[6] GOOSSENS, M., RAHTZ, S., MITTELBACH, F., ROEGEL, D., VOß, H. *The LaTeX Graphics Companion (2nd Edition)*. Addison-Wesley, 2007. 928 s. ISBN 0-321-50892-0.

[7] HAROLD, E., R., MEANS, W., S. *XML v kostce*. Praha: Computer Press, 2002. 439 s. ISBN 80-7226-712-4.

[8] HOLZNER, S. *XSLT příručka internetového vývojáře*. Praha: Computer Press, 2002. 515 s. ISBN 80-7226-600-4.

[9] JasTEX *Applet*. <http://www.lsv.ens-cachan.fr/~gastin/JasTeX/JastexApplet.html>

[10] JORSSEN, CH., VOß, H., BOONE, F. *pst-circ: A PSTricks package for drawing electric circuits*. 2008, version 1.45. <http://www.ctan.org/tex-archive/graphics/pstricks/contrib/pst-circ/>

[11] KAY, M, H. *SAXON Version 6.5.3*. 2003. <http://saxon.sourceforge.net/saxon6.5.3/>

[12] KNUTH, D., E. *The TeXbook*. Addison-Wesley, 1984. 496 s. ISBN 0-201-134489.

[13] KOPKA, H., DALY, P., W. *LaTeX: podrobný průvodce*. Brno: Computer Press, 2004. 576 s. ISBN 80-7226-973-9.

[14] KWOK, C. *EEPIC Extensions to EPIC and LaTeX Picture Environment Version 1.1*. 1988. <ftp://cam.ctan.org/tex-archive/macros/latex/contrib/eepic.zip>

[15] LAMPORT, L. *LaTeX: A Document Preparation System, 2/E*. Addison-Wesley, 1994. 288 s. ISBN 0-201-52983-1.

[16] LaTeX graphics *The picture Environment*.
<http://www.ursoswald.ch/LaTeXGraphics/picture/picture.html>

[17] MathWorld *Bézier Curve*.
<http://mathworld.wolfram.com/BezierCurve.html>

[18] PSTricks web site *Packages*.
<http://tug.org/PSTricks/main.cgi?file=packages>

[19] RODRIGUEZ, D., VOß, H. *pstricks-add: additionals Macros for pstricks*. 2006, version 2.82.
<http://www.ctan.org/get/graphics/pstricks/contrib/pstricks-add/>

[20] TANTAU, T. *User Guide to the Beamer Class, Version 3.07*. 2007.
<ftp://cam.ctan.org/tex-archive/macros/latex/contrib/beamer.zip>

[21] The Apache XML project *XALAN Version 2.7.1*. 2006.
<http://xalan.apache.org/index.html>

[22] TIŠNOVSKÝ, P. *Vektorový grafický formát SVG*. 2007.
<http://www.root.cz/serialy/graficke-formaty>

[23] VAN ZANDT, T. *PSTricks: user's guide*. 2003, version 97.
<http://www.dante.de/CTAN/graphics/pstricks/base/doc/pstricks-doc.pdf>

[24] VOß, H. *3D plots: PST-3dplot*. 2006, version 1.72.
<http://www.ctan.org/get/graphics/pstricks/contrib/pst-3dplot>

[25] VOß, H. *pst-func: plotting special mathematical functions*. 2008, version 0.52.
<http://www.ctan.org/get/graphics/pstricks/contrib/pst-func>

[26] W3C *Scalable Vector Graphics (SVG) 1.1 Specification*. 2003.
<http://www.w3.org/TR/2003/REC-SVG11-20030114/>

[27] W3C *XSL Transformations (XSLT) Version 1.0*. 1999.
<http://www.w3.org/TR/1999/REC-xslt-19991116>

[28] WICKS, M., A. *Dvipdfm User's Manual*. 1999, version 0.12.4b.
<http://gaspra.kettering.edu/dvipdfm/dvipdfm-0.12.4.pdf>

Aktuálnost internetových odkazů jsem zkontroloval dne 3. června 2008.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DVI	DeVice Independent
PDF	Portable Document Format
PS	PostScript
EPS	Encapsulated PostScript
AI	Adobe Illustrator Artwork
CDR	CorelDRAW
DXF	Drawing Exchange Format
JPEG	Joint Photographic Experts Group
BMP	Microsoft Windows Bitmap
GIF	Graphics Interchange Format
PNG	Portable Network Graphics
TIFF	Tagged Image File Format
XML	Extensible Markup Language
DTD	Document Type Definition
SVG	Scalable Vector Graphics
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations

SEZNAM PŘÍLOH

A UKÁZKA KONVERZE SVG GRAFIKY DO PSTRICKS

A.1 Vstupní dokument ve formátu SVG

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="744.09448" height="1052.3622"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <path d=
   "M137.14286,398.07647 L505.71429,395.21933 M145.71429,766.6479
    L140,395.21933 M500,392.36218 L505.71429,755.21933
    M145.71429,760.93361 L508.57143,758.07647 M262.85714,703.79075
    C273.6321,714.56571 262.85714,673.31456 262.85714,658.07647
    C262.85714,643.79075 262.85714,629.50504 262.85714,615.21933
    C262.85714,599.98123 262.85714,584.74313 262.85714,569.50504
    C262.85714,555.21933 262.85714,540.93361 262.85714,526.6479
    C262.85714,512.36218 262.85714,498.07647 262.85714,483.79075
    C262.85714,473.61527 253.83894,462.00368 251.42857,452.36218
    C247.03294,434.77967 234.28571,464.21442 234.28571,472.36218
    C234.28571,486.6479 262.85714,472.36218 277.14286,472.36218
    C291.42857,472.36218 305.71429,472.36218 320,472.36218
    C334.28571,472.36218 348.57143,472.36218 362.85714,472.36218
    C376.82112,472.36218 382.85714,488.28143 382.85714,500.93361
    C382.85714,515.21933 382.85714,529.50504 382.85714,543.79075
    C382.85714,557.00686 364.6435,564.05845 354.28571,566.6479
    C341.82669,569.76265 324.34703,566.6479 311.42857,566.6479
    C297.14286,566.6479 282.85714,566.6479 268.57143,566.6479
    C252.12293,566.6479 247.95227,560.31445 240,552.36218
    C226.11428,538.47647 241.13108,538.07647 257.14286,538.07647
    C265.58753,538.07647 279.09039,561.62557 282.85714,566.6479
    C291.84189,578.62756 303.5447,590.1926 314.28571,600.93361
    C323.08227,609.73017 333.88936,626.35063 345.71429,635.21933
    C357.19253,643.82801 371.21856,657.65644 377.14286,669.50504
    C380.50047,676.22027 391.6855,689.76197 397.14286,695.21933
    C401.59907,699.67553 403.20318,696.73441 408.57143,698.07647"
       fill="none" stroke="#000000" stroke-width="2"/></svg>
```

A.2 Stylová šablona XSLT pro vstupní SVG dokument

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
              xmlns:svg="http://www.w3.org/2000/svg">
<xsl:output method="text"/>
<xsl:output indent="no"/>
<xsl:template match="svg:*">
    <xsl:text>\psset{xunit=.5pt,yunit=.5pt,runit=.5pt}
    </xsl:text>
    <xsl:text>
\begin{pspicture}(</xsl:text>
    <xsl:value-of select="@width"/>
    <xsl:text>,-</xsl:text>
    <xsl:value-of select="@height"/>
    <xsl:text>)</xsl:text>
    <xsl:text>
\pscustom[linewidth=</xsl:text>
    <xsl:value-of select="svg:path/@stroke-width div 1.25"/>
    <xsl:text>]</xsl:text>
    <xsl:apply-templates select="svg:path"/>
    <xsl:text>
\end{pspicture}
    </xsl:text>
  </xsl:template>
  <xsl:template match="svg:path">
  <xsl:text>
{
\newpath</xsl:text>
    <xsl:call-template name="rozbor">
      <xsl:with-param name="retezec" select="normalize-space(@d)"/>
    </xsl:call-template>
    <xsl:text>
}</xsl:text>
  </xsl:template>
  <xsl:template name="rozbor">
```

```
    <xsl:param name="retezec"/>
    <xsl:variable name="prikaz">
      <xsl:value-of select="substring($retezec, 1, 1)"/>
    </xsl:variable>
  <!-- promenna pro retezec bez hodnoty prvniho prikazoveho znaku -->
    <xsl:variable name="retezec1">
    <!-- testuje se, jestli je 'prikaz' oddeleny mezerou od dvojice
         souradnic -->
      <xsl:choose>
        <xsl:when test="contains(substring($retezec, 2, 1), ' ')">
          <xsl:value-of select="substring-after($retezec, ' ')"/>
        </xsl:when>
      <!-- v pripade ze neni 'prikaz' oddeleny mezerou od dvojice
           souradnic -->
        <xsl:otherwise>
          <xsl:value-of select="substring-after($retezec, $prikaz)"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="$prikaz = 'M'">
        <xsl:text>
 \moveto(</xsl:text>
      </xsl:when>
      <xsl:when test="$prikaz = 'L'">
        <xsl:text>
 \lineto(</xsl:text>
      </xsl:when>
      <xsl:when test="$prikaz = 'C'">
        <xsl:text>
 \curveto(</xsl:text>
      </xsl:when>
    <!-- pomocny prikaz -->
      <xsl:when test="$prikaz = 'X'">
        <xsl:text>(</xsl:text>
      </xsl:when>
      <xsl:when test="$prikaz = 'Z' or $prikaz = 'z'">
        <xsl:text>
\closepath</xsl:text>
```

```
      </xsl:when>
    </xsl:choose>
    <xsl:choose>
    <!-- testuje se konec retezce a vola se sablona rozbor2 -->
      <xsl:when test="string-length($retezec1) > 1">
        <xsl:call-template name = "rozbor2">
          <xsl:with-param name = "retezec2"
                          select="normalize-space($retezec1)"/>
          <xsl:with-param name = "prikaz2" select="$prikaz"/>
        </xsl:call-template>
      </xsl:when>
    </xsl:choose>
  </xsl:template>
<!-- sablona pro zpracovavani hodnoty souradnice na ose 'x'  -->
  <xsl:template name = "rozbor2">
    <xsl:param name = "retezec2"/>
    <xsl:param name = "prikaz2"/>
    <xsl:choose>
      <xsl:when test="number(substring($retezec2, 1, 1)) or
                      contains(substring($retezec2, 1, 1), '0') or
                      contains(substring($retezec2, 1, 1), '.')">
        <xsl:value-of select = "substring($retezec2, 1, 1)"/>
        <xsl:call-template name="rozbor2">
          <xsl:with-param name="retezec2"
                        select="substring-after($retezec2,
                                substring($retezec2, 1, 1))"/>
          <xsl:with-param name="prikaz2" select="$prikaz2"/>
        </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text>,-</xsl:text>
        <xsl:choose>
          <xsl:when test="starts-with($retezec2, ' ,')">
            <xsl:call-template name="rozbor3">
              <xsl:with-param name="retezec3"
                            select="normalize-space(substring-after
                                    ($retezec2, ' ,'))"/>
            </xsl:call-template>
          </xsl:when>
```

```
          <xsl:otherwise>
            <xsl:call-template name="rozbor3">
              <xsl:with-param name="retezec3"
                              select="normalize-space(substring-after
                             ($retezec2, substring($retezec2, 1, 1)))"/>
              <xsl:with-param name = "prikaz3" select="$prikaz2"/>
            </xsl:call-template>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
<!-- sablona pro zpracovavani hodnoty souradnice na ose 'y'-->
  <xsl:template name = "rozbor3">
    <xsl:param name = "retezec3"/>
    <xsl:param name = "prikaz3"/>
      <xsl:choose>
        <xsl:when test="number(substring($retezec3, 1, 1)) or
                        contains(substring($retezec3, 1, 1), '0') or
                        contains(substring($retezec3, 1, 1), '.')">
          <xsl:value-of select = "substring($retezec3, 1, 1)"/>
          <xsl:call-template name="rozbor3">
            <xsl:with-param name="retezec3"
                            select="substring-after($retezec3,
                                    substring($retezec3, 1, 1))"/>
            <xsl:with-param name="prikaz3" select="$prikaz3"/>
          </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
        <!-- promenna eliminuje pripadnou oddelujici carku mezi
             dvojici souradnic  -->
          <xsl:variable name="retezec3T">
            <xsl:choose>
              <xsl:when test="starts-with(normalize-space($retezec3),
                              ',')">
                <xsl:value-of select="substring-after($retezec3, ',')
                                      "/>
              </xsl:when>
              <xsl:otherwise>
```

```
              <xsl:value-of select="$retezec3"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
      <!-- testuje se, jestli je za souradnici 'y' ciselny znak -->
        <xsl:choose>
          <xsl:when test="number(substring(normalize-space
                          ($retezec3T), 1, 1)) = false()">
            <xsl:text>)</xsl:text>
            <xsl:call-template name="rozbor">
              <xsl:with-param name="retezec"
                              select="normalize-space($retezec3T)"/>
            </xsl:call-template>
          </xsl:when>
          <xsl:otherwise>
            <xsl:text>)</xsl:text>
          <!-- doplni do retezce pomocny 'prikaz'  -->
            <xsl:choose>
              <xsl:when test="contains($prikaz3, 'C')">
                <xsl:call-template name="rozbor">
                  <xsl:with-param name="retezec"
                                  select="concat('X', $retezec3T)"/>
                </xsl:call-template>
              </xsl:when>
            <!-- doplni do retezce chybejici znak 'prikaz' -->
              <xsl:otherwise>
                <xsl:call-template name="rozbor">
                  <xsl:with-param name="retezec"
                                  select="concat($prikaz3,
                                          $retezec3T)"/>
                </xsl:call-template>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

A.3 Výstupní dokument ve formátu PSTricks

```
\psset{xunit=.5pt,yunit=.5pt,runit=.5pt}
\begin{pspicture}(744.09448,-1052.3622)
\pscustom[linewidth=1.6]
{
\newpath
 \moveto(137.14286,-398.07647)
 \lineto(505.71429,-395.21933)
 \moveto(145.71429,-766.6479)
 \lineto(140,-395.21933)
 \moveto(500,-392.36218)
 \lineto(505.71429,-755.21933)
 \moveto(145.71429,-760.93361)
 \lineto(508.57143,-758.07647)
 \moveto(262.85714,-703.79075)
 \curveto(273.6321,-714.56571)(262.85714,-673.31456)
         (262.85714,-658.07647)
 \curveto(262.85714,-643.79075)(262.85714,-629.50504)
         (262.85714,-615.21933)
 \curveto(262.85714,-599.98123)(262.85714,-584.74313)
         (262.85714,-569.50504)
 \curveto(262.85714,-555.21933)(262.85714,-540.93361)
         (262.85714,-526.6479)
 \curveto(262.85714,-512.36218)(262.85714,-498.07647)
         (262.85714,-483.79075)
 \curveto(262.85714,-473.61527)(253.83894,-462.00368)
         (251.42857,-452.36218)
 \curveto(247.03294,-434.77967)(234.28571,-464.21442)
         (234.28571,-472.36218)
 \curveto(234.28571,-486.6479)(262.85714,-472.36218)
         (277.14286,-472.36218)
 \curveto(291.42857,-472.36218)(305.71429,-472.36218)
         (320,-472.36218)
 \curveto(334.28571,-472.36218)(348.57143,-472.36218)
         (362.85714,-472.36218)
 \curveto(376.82112,-472.36218)(382.85714,-488.28143)
         (382.85714,-500.93361)
 \curveto(382.85714,-515.21933)(382.85714,-529.50504)
```

```
        (382.85714,-543.79075)
 \curveto(382.85714,-557.00686)(364.6435,-564.05845)
        (354.28571,-566.6479)
 \curveto(341.82669,-569.76265)(324.34703,-566.6479)
        (311.42857,-566.6479)
 \curveto(297.14286,-566.6479)(282.85714,-566.6479)
        (268.57143,-566.6479)
 \curveto(252.12293,-566.6479)(247.95227,-560.31445)
        (240,-552.36218)
 \curveto(226.11428,-538.47647)(241.13108,-538.07647)
        (257.14286,-538.07647)
 \curveto(265.58753,-538.07647)(279.09039,-561.62557)
        (282.85714,-566.6479)
 \curveto(291.84189,-578.62756)(303.5447,-590.1926)
        (314.28571,-600.93361)
 \curveto(323.08227,-609.73017)(333.88936,-626.35063)
        (345.71429,-635.21933)
 \curveto(357.19253,-643.82801)(371.21856,-657.65644)
        (377.14286,-669.50504)
 \curveto(380.50047,-676.22027)(391.6855,-689.76197)
        (397.14286,-695.21933)
 \curveto(401.59907,-699.67553)(403.20318,-696.73441)
        (408.57143,-698.07647)
}
\end{pspicture}
```