UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

BAKALÁŘSKÁ PRÁCE

Kreslení grafů elementárních funkcí Metapostem

Vedoucí bakalářské práce:
RNDr. Miloslav Závodný
Rok odevzdání: 2014

Vypracoval:
Romana Pítrová
ME, III. ročník

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně pod vedením pana RNDr. Miloslava Závodného s použitím uvedené literatury.

V Olomouci dne 10. ledna 2014

Poděkování

Ráda bych poděkovala panu RNDr. Miloslavovi Závodnému za odborné vedení, tvůrčí připomínky a v neposlední řadě za námět k bakalářské práci.

Obsah

Úvod

Jako téma mé bakalářské práce jsem si zvolila *Kreslení grafů elementárních funkcí Metapostem*. Dříve jsem se setkávala pouze s programy Microsoft Office, které jsem považovala za dostačující. V rámci studia matematiky jsem ale zjistila, že se denně potýkám s texty obsahujícími mnoho matematických výrazů a grafů. Proto jsem absolvovala předmět TEX, který mně přesvědčil o tom, že umí nejen vysázet složité matematické vzorce. Dozvěděla jsem se, že součástí systému TEX jsou i programy pro kreslení obrázků. Zaujal mě především Metapost – obrázek se zadává jednoduchými formulemi vektorové algebry a výsledek naprosto přesně odpovídá tomu, co bylo zadáno.

V první kapitole stručně popisuji jak vznikalo programování, považuji to nejen za zajímavé, ale také užitečné pro pochopení dalšího textu. Dále stručně představuji program TEX, jeho tvůrce Donalda Ervina Knutha a Metapost. Na to navazují ukázky práce s Metapostem. K většině z nich jsou připojeny zdrojové texty, příkazy v nich jsou vysvětlovány. V další kapitole představuji elementární funkce, jejich stručný popis (mnohé jejich vlastnosti ovlivňují způsob, jakým je ve zdrojovém souboru popsán jejich graf – sudost, lichost, periodicita) a uvádím jejich grafy tak, jak je Metapost snadno vykreslil.

1. Historie programování

Výpočetní technika je v dnešní době všude kolem nás. Dá se říct, že nám v mnohém usnadňuje práci a napomáhá k řešení různých úkolů a situací. Ne každý si ale uvědomuje, že to všechno není zásluha počítače jako takového, ale různých programů a dat, jejichž schopnosti a dovednosti nám jsou tolik užitečné. Programování je staré jako samotné počítače. První programátoři však museli vše psát v binární soustavě (ve které se používají pouze dva symboly – 0 a 1), jelikož neměli k dispozici žádné programovací jazyky. Během 2. světové války došlo k převratu jak v rozvoji počítačů, tak v jejich programování – německý technik Kondrad Zuse nabízí řadu počítačů Z 1-3. Revolučním rokem klasického programování se však stává rok 1944, ve kterém německý matematik John von Neumann přichází s počítačem, ve kterém je celý program společně s potřebnými daty uložen v operační paměti.

Počítače sloužily k řešení a zpřesnění matematických úloh a výpočtům, které se postupem času stávaly složitější, proto se i nároky na výkonnost počítačů neustále zvyšovaly. Pro programátory to představovalo vynaložení obrovského úsilí, jelikož si museli pamatovat složité číselné kódy. Důsledkem toho byl v padesátých letech vznik prvního programovacího jazyku Assembler (jazyk symbolických adres). Mezi nejznámější programy, které v něm byly napsány, patří operační systém UNIX. Psaní v Assembleru se však rovněž stávalo čím dál více složitějším, bylo potřeba vytvořit jazyk, který bude stručný, spolehlivý, výkonný a zároveň jednoduchý. Začaly se sestavovat programy především pro vědecko-technické výpočty, mezi které patří například program BASIC, Fortran nebo Algol. V roce 1972 byl vytvořen vyšší programovací jazyk C, který byl později používán pro UNIX a ovlivnil další programovací jazyky. Velký podíl na rozvoji počítačů a programování má také celosvětově známá firma Microsoft a její Windows. Poslední místo ve vývoji zatím zaujímají jazyky Perl (užívaný i TEXem) a Python.

2. TEX

2.1. Donald Knuth

Zakladatel TEXu Donald Ervin Knuth se narodil 10. ledna 1938 v Milwaukee, ve Wisconsinu. Od mládí se věnoval spíše hudbě než matematice, což se změnilo po přijetí stipendia Clevelandské univerzity, kde začal studovat fyziku. Svou publicitu získal v roce 1958, kdy se mu podařilo vytvořit program, který analyzoval výkon univerzitního basketbalového týmu. V roce 1962 začal Knuth připravovat učebnice o programovací technice a výsledkem této práce stala sedmisvazková série s názvem Umění programování. Již bylo vytištěno zhruba milion kopií, a to včetně překladů do šesti světových jazyků. Knuth však stále nebyl spokojen s kvalitou svých prací, a to především se sazbou matematických vzorců a s dodržováním typografických pravidel. V roce 1977 začal vyvíjet sázecí systém TEX, kvůli kterému také na deset let přerušil svou práci na ostatních projektech. Kromě toho se zabýval systémem Metafont. Jeho výzkumné práce byly také velice užitečné při rozvoji několika oblastí informatiky a softwarového inženýrství. Hlavním cílem jeho práce bylo nalezení vhodné rovnováhy mezi teorií a praxí.

Donald Knuth je držitelem mnoha ocenění a čestných doktorátů, je také mimo jiné členem Americké akademie umění a věd, Národní akademie věd a Národní akademie inženýrství. V současné době pracuje jako profesor na Standfordské univerzitě a jeho hlavním koníčkem je hudba.

2.2. TEX

Program TEX můžeme popsat jako typografický systém připomínající programovací jazyk. Mezi nejsilnější stránku tohoto programu patří sazba matematických výrazů, což TEX na rozdíl od ostatních systémů (jako např. MS Word) dokonale ovládá. Další výhodou je možnost oddělení tisknutelného textu od textu, ve kterém se provádějí úpravy vzhledu, jeho stabilita a také nezávislost na ope-

račním systému. To znamená, že text vytvořený v jednom operačním systému se dá zpracovat a přeložit také v jiném systému. Slabší stránkou TEXu je grafika. Ta je totiž závislá na výstupním zařízení, zatímco TEX byl vytvořen jako systém, jehož výstupy jsou na výstupních zařízení nezávislé. Program ale umožňuje vložit do textu obrázky, pro tento účel se pak výstupní soubor převádí do formátu pdf nebo postskript.

Obecně se užívají dva formáty, a to LATEX, který je vhodnější pro začátečníky, a plainTEX umožňující rychlejší formátování textu, avšak za cenu toho, že si uživatel musí sám nadefinovat některé příkazy. Do jisté míry záleží na uživateli samotném, který z formátů si zvolí. V prvním případě si nemusí dělat starosti s tím, zda má pro sazbu k dispozici vhodné příkazy. Malou nevýhodu LATEXu je to, že starší verze LATEX2.09 nerozumí novým příkazům, formát LATEX se neustále vyvíjí.

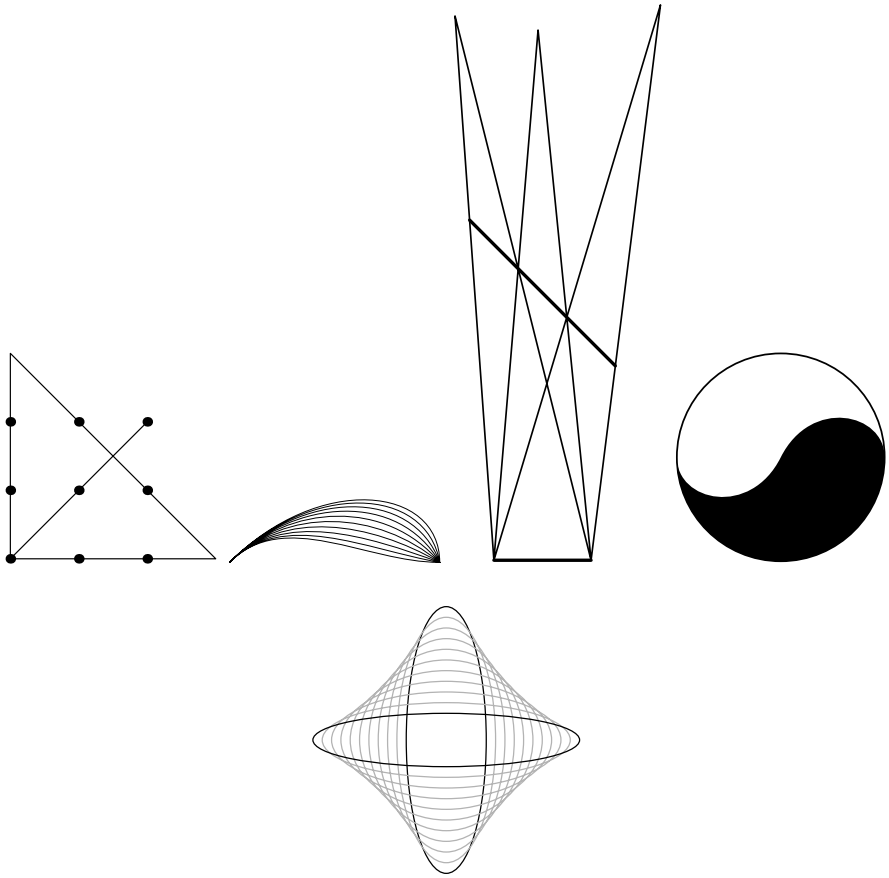
Celý systém je tvořen různými soubory (spustitelnými i nespustitelnými), které jsou nezbytné k tomu, aby TEX správně fungoval. Patří mezi ně například:

- samotný program TEXa jeho pomocné programy
  (binární programy, skripty, aj.)
- METAFONT a s ním související programy
- Metapost
- formáty a knihovny
- fonty
- dokumentace, ukázky aj.
  (uživatel se dozví, co k čemu slouží)
- soubory vytvořené uživatelem (nejlépe v texmf-local)

2.3. Metapost

Metapostem rozumíme jednak programovací jazyk, který je určený pro tvorbu obrázků, jednak vlastní program zpracovávající tento jazyk. Jeho autorem je John D. Hobby. Vychází z Knuthova METAFONTu, z něho mnohé převzal, je ale rozšířen o příkazy využívající možností postskrtiptu – pracuje s barvou, umožňuje snadný popis obrázků.

Na rozdíl od METAFONTu, jehož výstupem je „obrázkový“ font, je výstupem Metapostu sada jednotlivých postskriptových souborů – co obrázek, to jeden soubor. Je-li to potřebné, lze tyto postskriptové soubory převést programem `mftopdf` do formátu pdf. Tyto soubory – obrázky – pak můžeme např. propojit s LaTeXovským dokumentem, lze je ale používat nezávisle na TeXu.

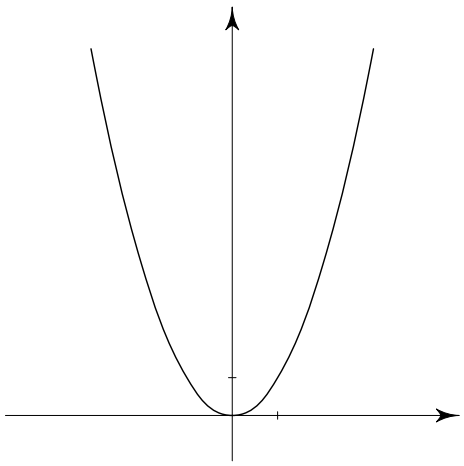


Obrázek 1: Obrázky vykreslené Metapostem

3. Umění Metapostu

Práce s Metapostem je v podstatě jednoduchá a zábavná. Je ale třeba, aby každý jeho uživatel rozuměl jednotlivým příkazům a uměl je správně použít. Metapost pak vykreslí téměř cokoliv a záleží na každém, do jaké míry jeho umění využije.

Vzhledem k tématu bakalářské práce si ukážeme, jakým způsobem se vykreslují grafy elementárních funkcí a uvedeme příkazy, které nám v některých případech mohou „usnadnit práci“. Každý metapostový vstup obsahuje příkaz `beginfig()` a `endfig`, celý zdrojový text je pak ukončen příkazem `end`. Číselný parametr příkazu `beginfig()` určuje jméno odpovídajícího výstupního souboru. Těchto souborů může být v jednom zdrojovém textu více. Na jejich začátku bývají obvykle uvedená nastavení speciální parametrů. Jako příklad uvedeme graf funkce $f(x) = x^2$ včetně jeho zadání:



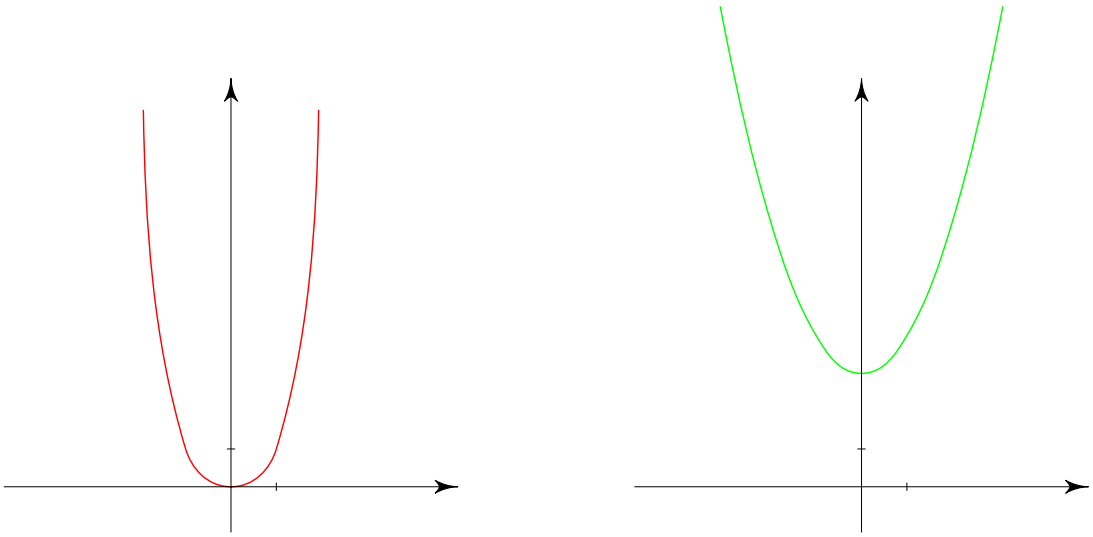
Obrázek 2: $f(x) = x^2$

```
beginfig(1); numeric w,h; w=6cm; h=6cm;
numeric xjed, yjed; xjed=6u; yjed=5u;
path s[];
z1'=(0,h/10); z2'=(w,y1'); z3'=(w/2,0); z4'=(x3',h); z5'=(x3',y1');
for k=0 upto 5:
z[k]=(k,k**2);
```

```
endfor;
s0 := z0{dir0}..z1{1,2}..z2{1,4}..z3..z4..z5;
s1 := s0 xscaled xjed yscaled yjed shifted z5';
s2 := subpath(0,3.1) of s1;
s3 := s2 reflectedabout(z3',z4');
pickup pencircle scaled d1;
draw s2; draw s3;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
hcarka((x3',y5'+yjed));
endfig;
```

Jednotlivé příkazy se v Metapostu oddělují středníkem. Základním příkazem kreslení je příkaz `draw`. Na začátku nadefinujeme délku osy x a y, vhodné je také nadefinovat si potřebné body a ty pak používat. Spojité čáry se nazývají cesty, označují se jako proměnné `path`. Tvar cesty můžeme ovlivnit pomocí řady příkazů. Pro vykreslení lomené čáry se jednotlivé body oddělují `--`, pro hladkou křivku se používají dvě tečky `..`, jak je tomu v našem případě. Jestliže je v nějakém bodě potřeba předepsat sklon křivky, používá se příkaz `dir`. Pokud bychom chtěli křivku „navést“, kterým směrem by měla směřovat, můžeme jednotlivým bodům zadat jejich souřadnice – vypočítáme derivaci dané funkce a do ní dosazujeme. Příkazy `xscaled`/`yscaled` přenásobí všechny x–ové/y–ové souřadnice danou hodnotou. Metapost umožňuje používat `for` cykly, díky nimž můžeme graf funkce nakreslit jedním cyklem. V našem konkrétním případě také využíváme příkaz `reflectedabout`. Funkce $f(x) = x^2$ je sudá, takže je její graf souměrný podle osy y. Tento příkaz je jedním z těch, které „šetří náš čas“. Stačí nadefinovat cestu pouze pro část grafu, ta se pak pomocí `reflectedabout` překlopí podle

osy y. Parametry tohoto příkazu tvoří vždy body určující osu, kolem které se má graf překlopit. Šipky na konci os zajistí příkaz `drawarrow`, pomocí `shifted` jsou posunuty na osu x a y. Příkaz `pickup pencircle` určuje použití kaligrafického pera. Změnou předpisu funkce ve `for` cyklu můžeme snadno vykreslit její možné varianty, barvu určíme příkazem `withcolor`.



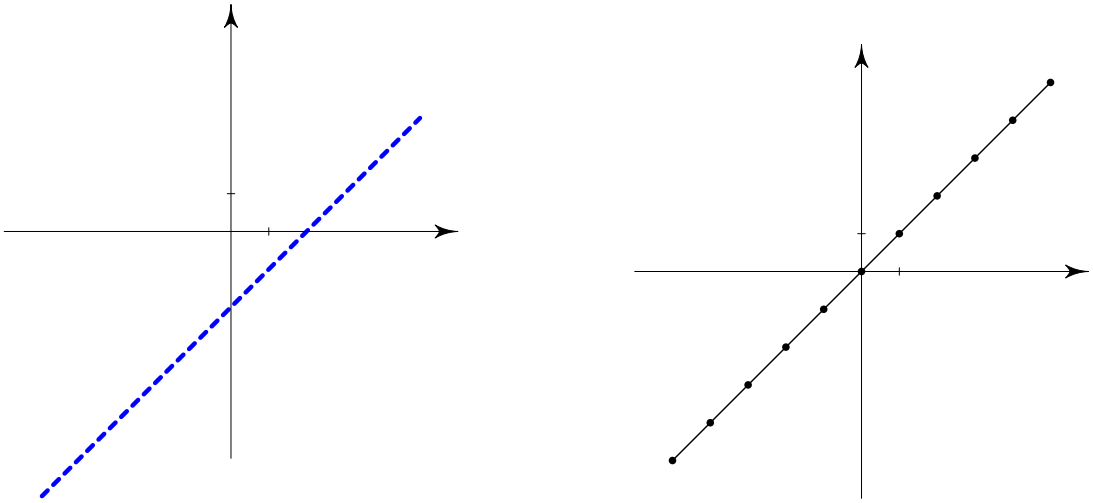
Obrázek 3: $f(x) = x^4$; $f(x) = 3 + x^2$

Zdrojový text ke grafu funkce $f(x) = 3 + x^2$:

```
beginfig(3);
numeric w,h; w=6cm; h=6cm;
numeric xjed, yjed; xjed=6u; yjed=5u;
path s[];
z1'=(0,h/10); z2'=(w,y1'); z3'=(w/2,0); z4'=(x3',h); z5'=(x3',y1');
for k=0 upto 5:
z[k]=(k,3+k**2);
endfor;
s0 := z0{dir0}..z1{1,2}..z2{1,4}..z3..z4..z5;
s1 := s0 xscaled xjed yscaled yjed shifted z5';
```

```
s2 := subpath(0,3.1) of s1;
s3 := s2 reflectedabout(z3',z4');
pickup pencircle scaled d1;
draw s2 withcolor green; draw s3 withcolor green;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
hcarka((x3',y5'+yjed));
endfig;
```

Křivky vykreslené Metapostem mohou být také přerušované. Slouží k tomu příkaz `dashed`, který má více variant – např. `dashed withdots` pro tečkovanou čáru, `dashed evenly` pro čáru čárkovanou, nebo pro uživatelem definovanou čáru čerchovanou `dashed dashpattern(on 5mm off 1mm on 1pt off 1mm)`. Použití obou příkazů vidíme na ukázce grafů lineárních funkcí $f(x) = x - 2$ a $f(x) = x$. V prvním případě je tloušťka grafu silnější, což je způsobeno změnou parametru `d` u příkazu `pickup pencircle`.

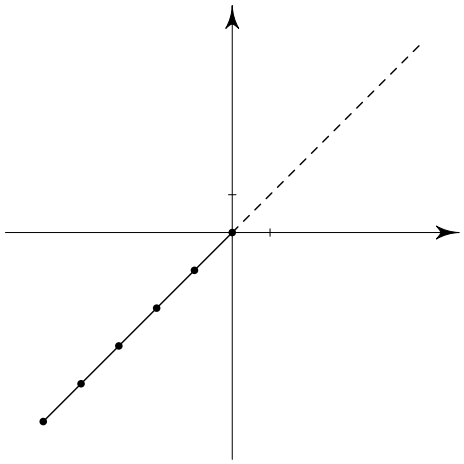


Obrázek 4: $f(x) = x - 2$; $f(x) = x$

```
beginfig(4);
numeric w,h; w=6cm; h=6cm;
numeric xjed, yjed; xjed=5u; yjed=5u;
path s[];
z1'=(0,h/2); z2'=(w,y1'); z3'=(w/2,0); z4'=(x3',h); z5'=(x3',y1');
for k=-5 upto 5:
z[k]=(k,k-2);
endfor;
s0 := z[-5]--z[-4]--z[-3]--z[-2]--z[-1]--z0{dir0}--z1--z2--z3--z4--z5;
s1 := subpath(0,10) of s0;
s3 := s1 xscaled xjed yscaled yjed shifted z5';
pickup pencircle scaled d3;
draw s3 dashed evenly withcolor blue;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
hcarka((x3',y5'+yjed));
endfig;
```

Je také možné použít pro jednu křivku oba příkazy – jak pro čárkovanou, tak pro tečkovanou čáru – současně. Stačí si nadefinovat počet cest podle toho, které části křivky chceme danými styly vykreslit:

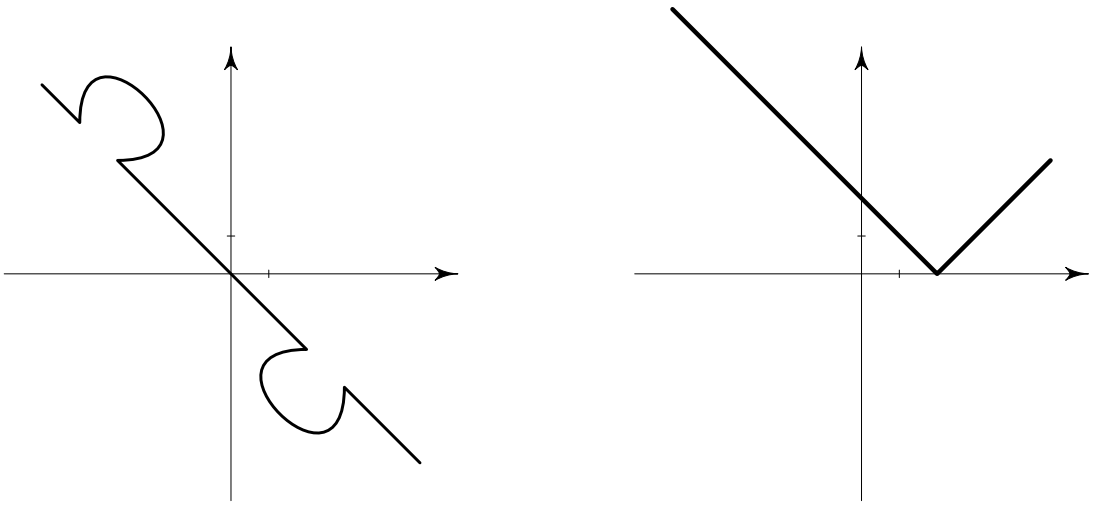
```
beginfig(29);
numeric w,h; w=6cm; h=6cm;
numeric xjed, yjed; xjed=5u; yjed=5u;
path s[];
z1'=(0,h/2); z2'=(w,y1'); z3'=(w/2,0); z4'=(x3',h); z5'=(x3',y1');
for k=-5 upto 5:
```



Obrázek 5: $f(x) = x$

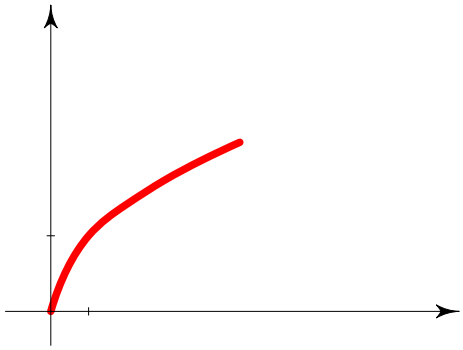
```
z[k]=(k,k);
endfor;
s0 := z[-5]--z[-4]--z[-3]--z[-2]--z[-1]--z0{dir0};
s1 := z0{dir0}--z1--z2--z3--z4--z5;
s2 := subpath(0,5) of s0;
s3 := subpath(0,5) of s1;
s4 := s2 xscaled xjed yscaled yjed shifted z5';
s5 := s3 xscaled xjed yscaled yjed shifted z5';
pickup pencircle scaled d1;
draw s4; draw s5 dashed evenly;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
hcarka((x3',y5'+yjed));
pickup pencircle scaled u;
for k=-5 upto 0:
drawdot z[k] xscaled xjed yscaled yjed shifted z5';
endfor;
```

(druhá možnost pro vykreslení tečkované čáry – `for` cyklus a příkaz `drawdot`)
`endfig;`



Obrázek 6: $f(x) = -x$; $f(x) = |x - 2|$

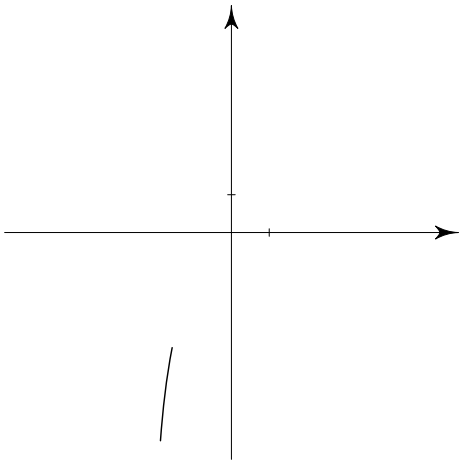
Graf funkce $f(x) = \sqrt{x}$ můžeme vykreslit pomocí příkazu pro odmocninu `sqrt`. Je ale také možné využít toho, že je to funkce inverzní k funkci $f(x) = x^2$. Stačí tedy ve `for` cyklu zadat předpis funkce $f(x) = x^2$ a použít příkaz `reflectedabout`. Za jeho parametry zvolíme souřadnice osy prvního a třetího kvadrantu, protože graf inverzní funkce k dané funkci f je osově souměrný právě podle této osy.

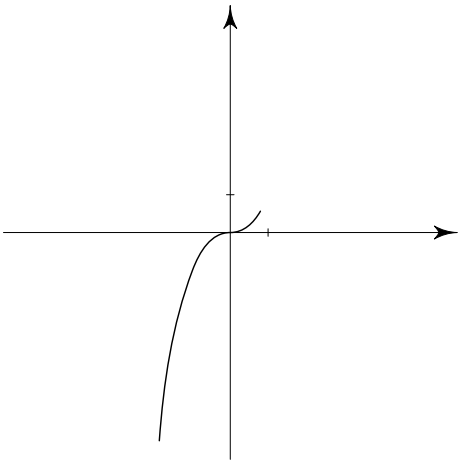


Obrázek 7: $f(x) = \sqrt{x}$

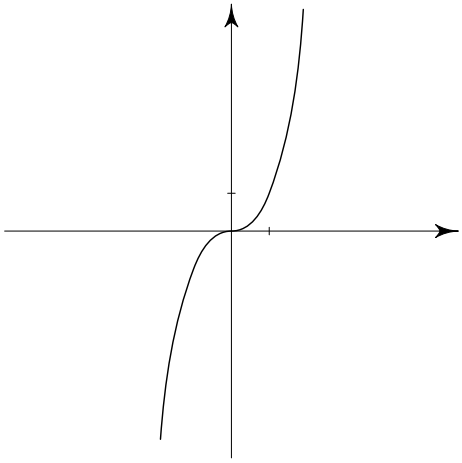
```
beginfig(10);
numeric w,h; w=6cm; h=4.5cm;
numeric xjed, yjed; xjed=5u; yjed=10u;
path s[];
z1'=(0,h/10); z2'=(w,y1'); z3'=(w/10,0); z4'=(x3',h); z5'=(x3',y1');
for k=0 upto 5:
z[k]=(k,k**2);
endfor;
s0 := z0{dir0}..z1{1,2}..z2{1,4}..z3..z4..z5;
s1 := subpath(0,2.9) of s0;
s2 := s1 reflectedabout((0,0),(1,1));
s3 := s2 xscaled xjed yscaled yjed shifted z5';
pickup pencircle scaled d4;
draw s3 withcolor red;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
hcarka((x3',y5'+yjed));
endfig;
```

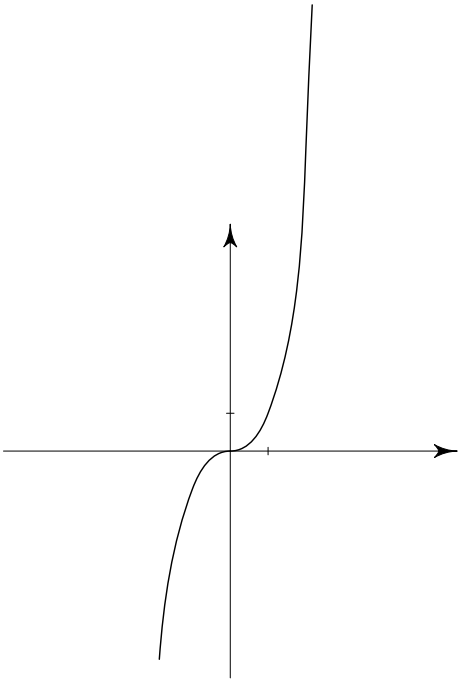
Důležité je také správně nastavit parametry u příkazu `subpath`, který určuje rozdělení cest na jednotlivé úseky. Je nutné určit poměr parametrů tak, aby křivka odpovídala skutečnému tvaru funkce. Během tvorby některých grafů jsem na tento problém narazila. Bylo zajímavé sledovat, jak i malá změna dokázala na graf zapůsobit. Na následujících obrázcích můžeme vidět, co s křivkou udělají nevhodně zvolené parametry příkazu `subpath(a,b)`:





První obrázek je vykreslen při hodnotě `subpath(3.35,3.7)`. Začátek grafu odpovídá délkám os, což znamená, že je třeba změnit – v tomto případě zvýšit – parametr `b`. Počet bodů na cestě se dá spočítat, nebo je třeba postupně zkoušet, která hodnota „sedí“ nejlépe. Na posledním obrázku vidíme, že křivka se prodlužuje nad rámec souřadnicové soustavy. Je tomu tak při hodnotě `subpath(3.35,7.2)`. Můžeme tedy srovnat parametry první a poslední ukázky – změna je velmi malá, přesto má velký vliv na délku cesty.



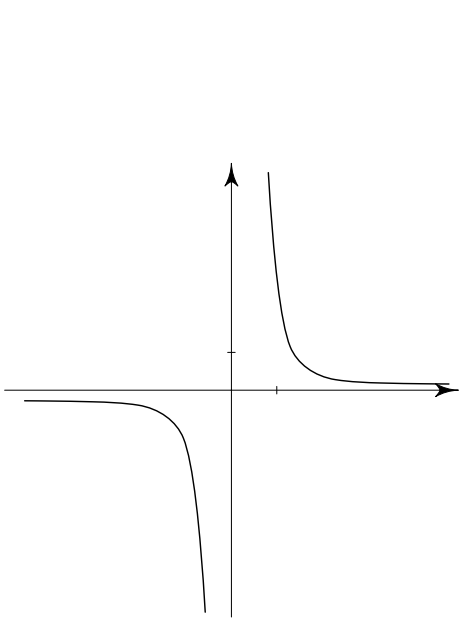


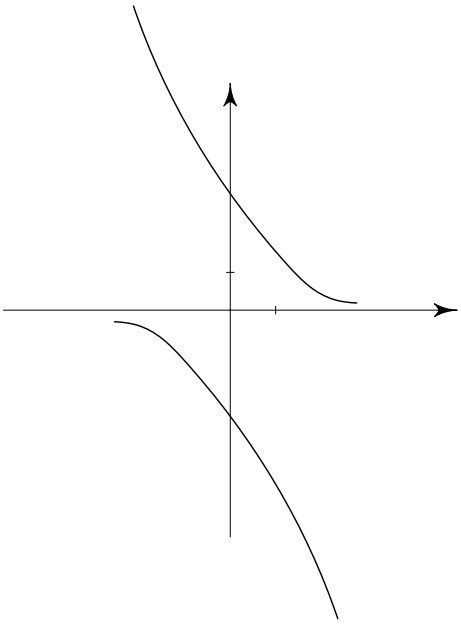
Dalším transformačním příkazem k usnadnění práce, je příkaz `rotatedaround`((a,b),α)). Díky němu můžeme vykreslenou křivku otáčet proti směru hodinových ručiček, a to okolo bodu (a,b) o úhel α zadaný ve stupních. Ve své práci jsem ho využila při kreslení grafu funkce $f(x) = x^{-3}$ – vykreslila jsem křivku pouze v prvním kvadrantu a do třetího ji pomocí `rotatedaround` otočila. Pro srovnání vidíme, co způsobí odstranění souřadnic bodů a změna parametrů příkazu `subpath`:
pro první graf:

```
s0 := z0{1,-250}..z1{1,-3}..z2{1,-3/16}..z3..z4..z5;
s1 := s0 xscaled xjed yscaled yjed shifted z5';
s2 := subpath(0.7,4.5) of s1;
s3 := s2 rotatedaround((xjed+x5'-6u,y5'),180);
```

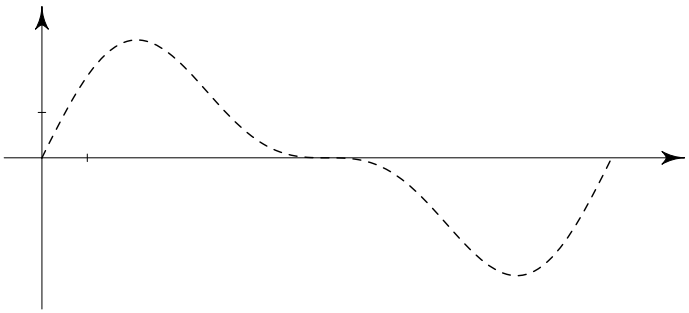
pro druhý graf:

```
s0 := z0..z1..z2..z3..z4..z5;
s1 := s0 xscaled xjed yscaled yjed shifted z5';
s2 := subpath(0.7,2.5) of s1;
s3 := s2 rotatedaround((xjed+x5'-6u,y5'),180);
```





Metapost si umí poradit také s goniometrickými funkcemi. Obdobně vykreslíme osy a měřítka a zvolíme cestu `path`. Pomocí `for` cyklu nadefinujeme příslušnou funkci. V případě $f(x) = 2\sin x + \sin 2x$ se omezíme na interval od 0 do 2π.

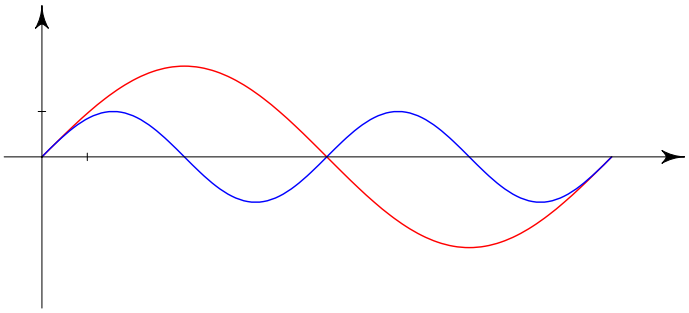


Obrázek 8: $f(x) = 2\sin x + \sin 2x$

```
beginfig(14);
numeric w,h; w=9cm; h=4cm;
numeric xjed, yjed; xjed=6u; yjed=6u;
path s[];
z1'=(0,h/2); z2'=(w,y1'); z3'=(5u,0); z4'=(x3',h); z5'=(x3',y1');
for k=0 step 10 until 360:
z[k]=(k,((2sind k)+(sind 2k)));
endfor;
s0 := for k=0 step 10 until 360:z[k]...endfor z[360];
s2 := s0;
s3 := s2 yscaled yjed xscaled ((xjed/45)*1.57) shifted z5';
pickup pencircle scaled d1;
draw s3 dashed evenly;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
```

```
hcarka((x3',y5'+yjed));
endfig;
```

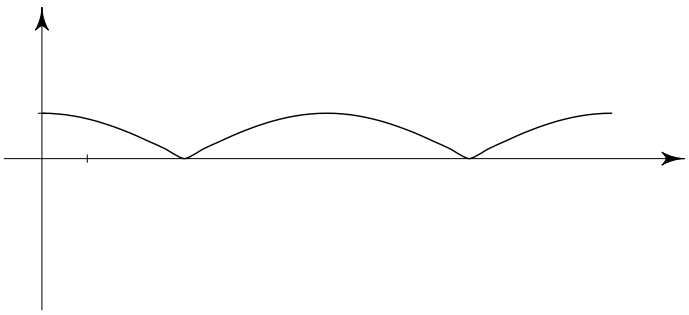
Je také možné vykreslit dvě křivky současně do jednoho grafu, například pokud chceme srovnat jejich tvar měnící se se změnou jejich funkčního předpisu. Chceme-li vykreslit graf funkce $\sin 2x$ a $2\sin x$, jednoduše nadefinujeme každou funkci zvlášť a následně také jejich cesty. Příkaz `draw` poté použijeme samostatně pro každou cestu. Pro zřetelnost můžeme každý graf vykreslit jinou barvou.



Obrázek 9: Srovnání grafů funkcí $f(x) = 2\sin x$ a $f(x) = \sin 2x$

```
beginfig(13);
numeric w,h; w=9cm; h=4cm;
numeric xjed, yjed; xjed=6u; yjed=6u;
path s[];
z1'=(0,h/2); z2'=(w,y1'); z3'=(5u,0); z4'=(x3',h); z5'=(x3',y1');
for k=0 step 10 until 360:
z[k]=(k,2sind k);
z'[k]=(k, sind 2k);
endfor;
s0 := for k=0 step 10 until 360:z[k]...endfor z[360];
s10 := for k=0 step 10 until 360:z'[k]...endfor z'[360];
s2 := s0;
s3 := s2 yscaled yjed xscaled ((xjed/45)*1.57) shifted z5';
s11 := s10 yscaled yjed xscaled ((xjed/45)*1.57) shifted z5';
```

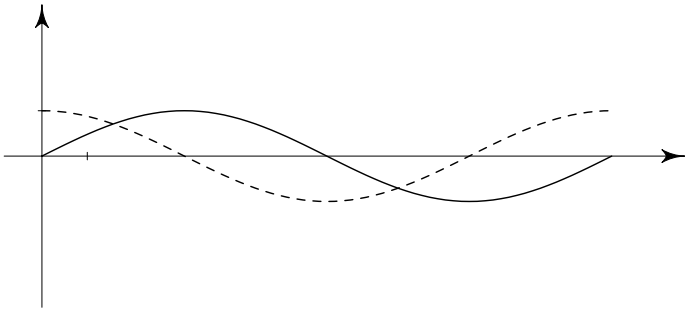
```
pickup pencircle scaled d1;
draw s3 withcolor red; draw s11 withcolor blue;
pickup pencircle scaled d0;
draw z1'--z2'; draw z3'--z4';
filldraw arrow(3u,90) shifted z4';
filldraw arrow(3u,0) shifted z2';
vcarka((x5'+xjed,y1'));
hcarka((x3',y5'+yjed));
endfig;
```



Obrázek 10: $f(x) = |\cos x|$

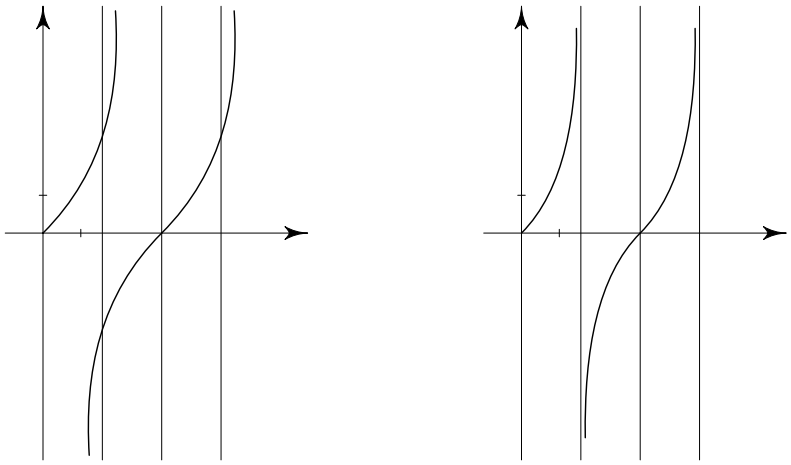
Chceme-li vykreslit křivku funkce kosinus, postupujeme stejným způsobem jako u funkce sinus. Jediná změna je v zadání funkce, tedy místo příkazu `sind` použijeme `cosd`. Také vidíme, jak si Metapost dokáže poradit s absolutní hodnotou; tu zadáváme pomocí příkazu `abs`. Následující zdrojový text je uveden pouze v rámci odlišného zápisu ve `for` cyklu:

```
...
for k=0 step 10 until 360:
z[k]=(k,abs(cosd k));
endfor;
...
```



Obrázek 11: $f(x) = \sin x$; $f(x) = \cos x$

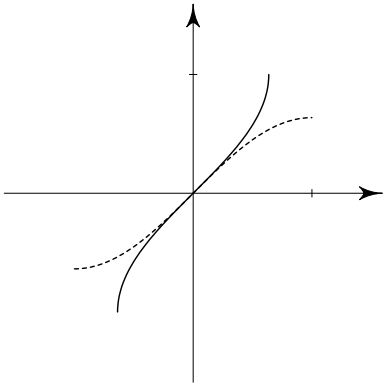
Někdy se může stát, že Metapost sice správný tvar i délku grafu vykreslí, ale přesto křivka zasahuje i mimo oblast, ve které ji chceme mít. Za tímto účelem se nabízí možnost, kdy jednotlivé body neoddělíme dvěma tečkami, jak tomu bylo v předchozích případech, ale třemi. Jako bychom chtěli Metapostu říct, aby se opravdu soustředil na co nejpřesnější vykreslení a v námi požadovaném úseku. Stejná strategie platí i u lomených čar. Na ukázce grafu funkce $f(x) = \operatorname{tg} x$ vidíme, jak změna oddělovačů křivku ovlivnila:



Obrázek 12: $f(x) = \operatorname{tg} x$

Na závěr můžeme pro zajímavost nastínit postup při kreslení grafů cyklometrických funkcí. Vzhledem k tomu, že známe postup vykreslení grafů funkcí goniometrických, máme v podstatě vyhráno. Jedná se o funkce navzájem inverzní,

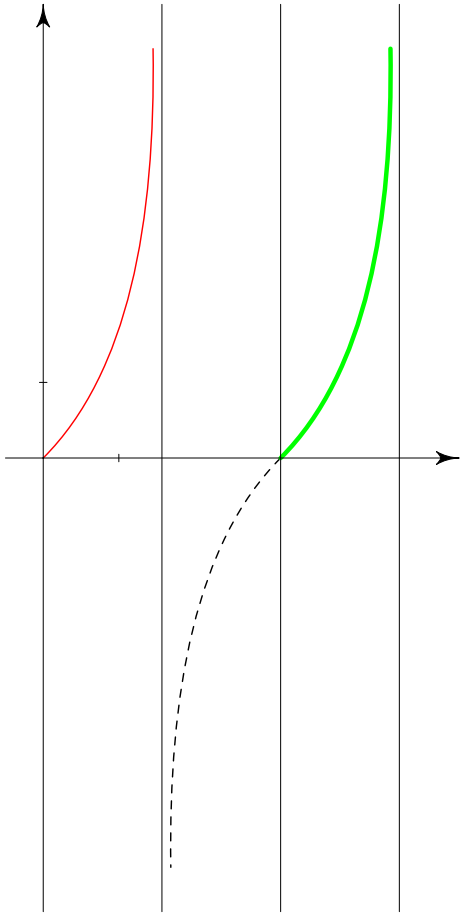
a proto stačí ve `for` cyklu zaměnit pořadí předpisu funkce a proměnné `k`. Na obrázku vidíme grafy funkcí $f(x) = \sin x$ a $f(x) = \arcsin x$ v intervalu od $-\frac{\pi}{2}$ do $\frac{\pi}{2}$, pro srovnání také část jejich zdrojového textu:



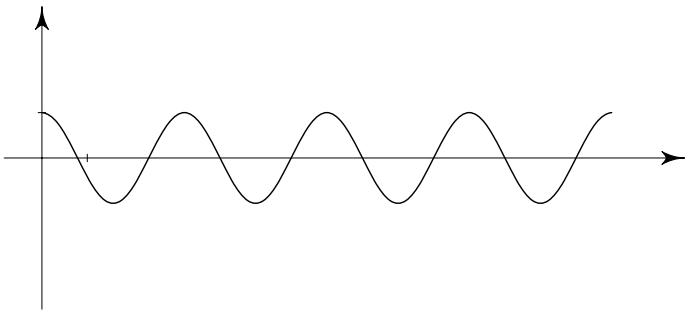
Obrázek 13: Graf funkce $f(x) = \arcsin x$

```
...
path s[];
z1'=(0,h/2); z2'=(w,y1'); z3'=(w/2,0); z4'=(x3',h); z5'=(x3',y1');
for k=-90 step 10 until 90:
z[k]=(sind k,k);
z'[k]=(k,sind k);
endfor;
s0 := for k=-90 step 10 until 80:z[k]...endfor z[90];
s10 := for k=-90 step 10 until 80:z'[k]...endfor z'[90];
s2 := s0;
s3 := s2 xscaled xjed yscaled ((yjed/90)*1.57) shifted z5';
s11 := s10 yscaled yjed xscaled ((xjed/90)*1.57) shifted z5';
pickup pencircle scaled d1;
draw s3; draw s11 dashed evenly scaled 0.5;
...
```

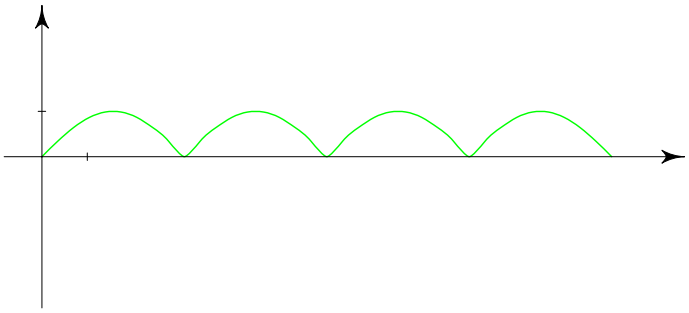
Následující ukázky jsou různými kombinacemi většiny popsaných příkazů, jejichž účelem je ukázat, že práce s Metapostem se může stát i zábavou:



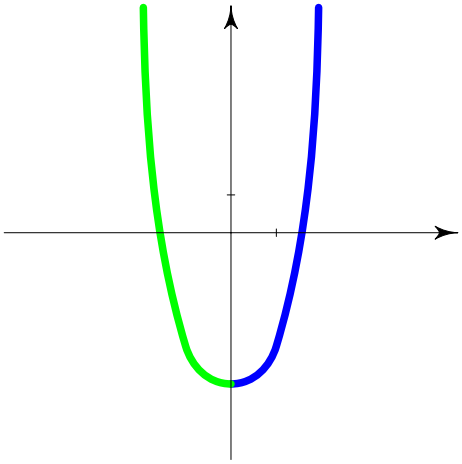
Obrázek 14: $f(x) = \operatorname{tg} x$

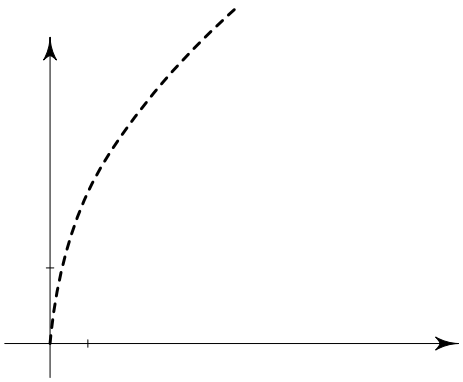


Obrázek 15: $f(x) = \cos 4x$

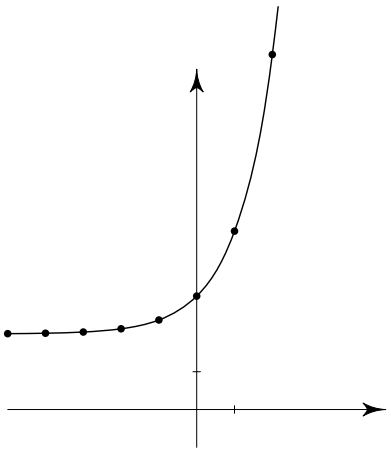


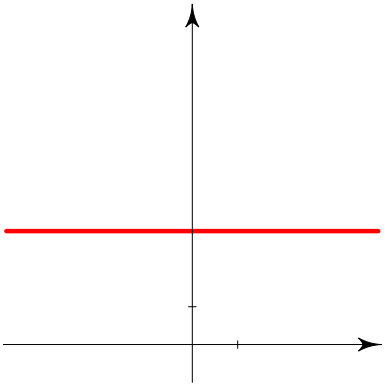
Obrázek 16: $f(x) = |\sin 2x|$





Obrázek 17: $f(x) = x^4 - 4$; $f(x) = 2\sqrt{x}$





Obrázek 18: $f(x) = e^x + 2$; $f(x) = 3$

4. Elementární funkce

V této kapitole představíme *elementární funkce*, jejich základní vlastnosti (úzce související s jejich grafy) a nakonec konkrétní elementární funkce. Elementární funkce je funkce, kterou je možno získat konečným počtem sčítání, odčítání, násobení, dělení a také složení, a to pouze ze základních elementárních funkcí. Patří mezi ně funkce mocninné, logaritmické, exponenciální, goniometrické, cyklometrické, hyperbolické[1] a hyperbolometrické[2]. Podrobněji se dále budu zabývat prvními čtyřmi typy.

4.1. Obecné charakteristiky

Definice 1 *Reálná funkce jedné reálné proměnné je předpis, podle kterého je každému $x \in R$ přiřazeno právě jedno $y \in R$, tedy $y = f(x)$.*

Lze také říci, že reálná funkce jedné reálné proměnné je každá množina uspořádaných dvojic $[x, y] \in R$ taková, pro kterou platí, že ke každému $x \in R$ existuje nejvýše jedno $y \in R$, pro něž $(x, y) \in f$. Po zavedení pojmu funce je také nezbytné definovat její definiční obor $D(f)$ a obor hodnot $H(f)$.

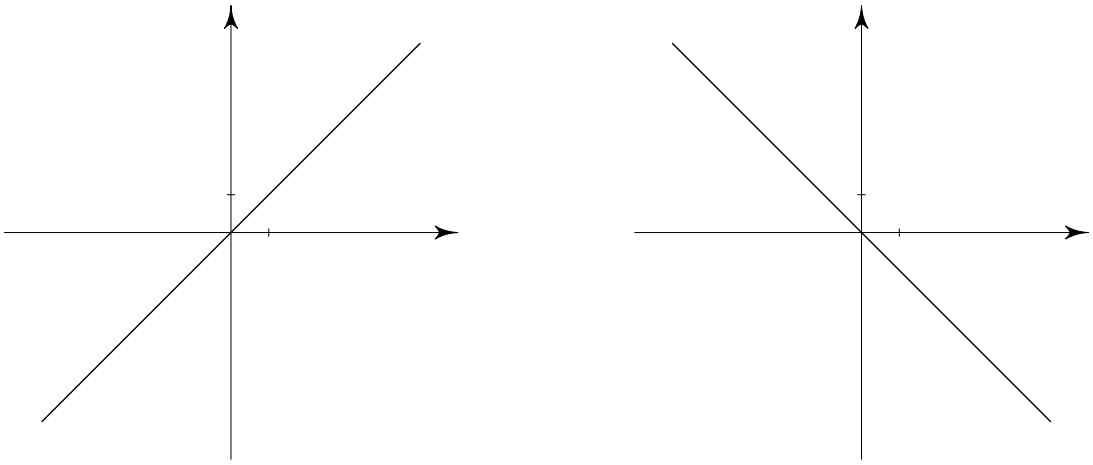
Definice 2 *Definiční obor funkce f je množina všech $x \in R$, k nimž existuje $y \in R$ takové, že $[x, y] \in f$.*

Definice 3 *Obor hodnot funkce f je množina všech $y \in R$, k nimž existuje $x \in R$ tak, že $[x, y] \in f$.*

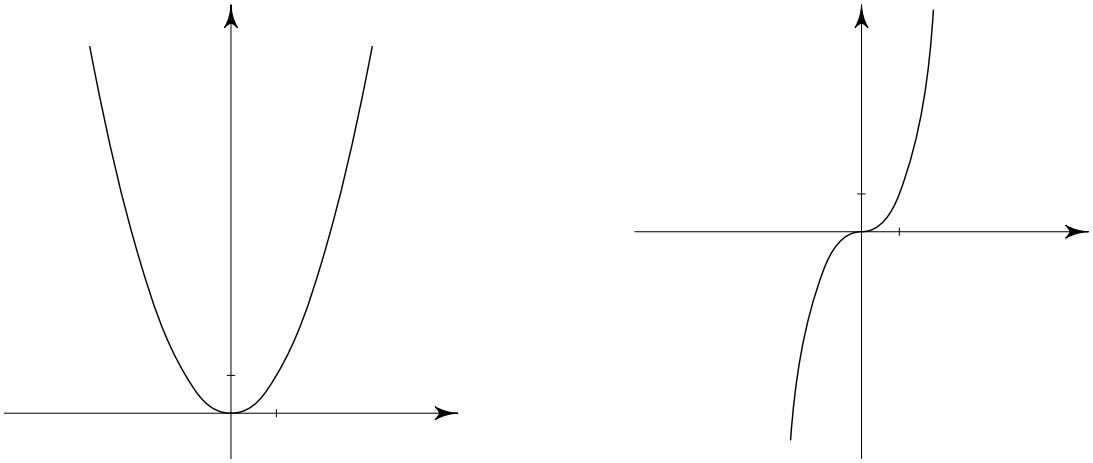
Definice 4 *Funkce f a g jsou si rovny, jestliže mají stejný definiční obor a pro všechna x z tohoto definičního oboru platí: $f(x) = f(g)$.*

Existuje mnoho způsobů, kterými funkci můžeme zadat. Patří mezi ně tabulka, slovní popis, funkční předpis nebo graf.

Definice 5 *Graf funkce f ve zvolené soustavě souřadnic v rovině je množina všech bodů roviny $[x, f(x)]$, kde $x \in D(f)$.*



Obrázek 19: Graf rostoucí a klesající funkce



Obrázek 20: Graf sudé a liché funkce

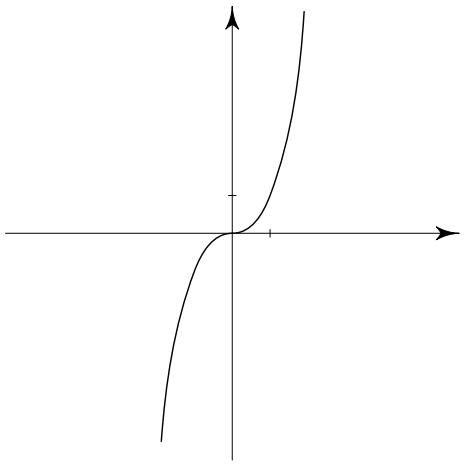
4.2. Mocninné funkce

Mocninné funkce rozlišujeme podle toho, z jaké podmnožiny reálných čísel jejich exponent pochází. Na tom také závisí jejich vlastnosti, obor hodnot a definiční obor. V rámci této kapitoly popíšeme mocninné funkce s přirozeným exponentem a jako zajímavost funkci konstantní.

[1]skupina funkcí analogických funkcím goniometrickým
[2]inverzní funkce k funkcím hyperbolickým

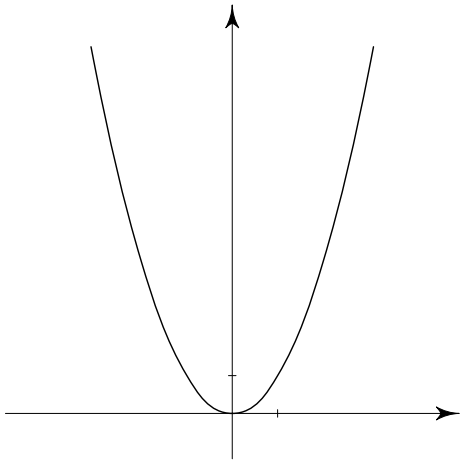
Definice 6 *Mocninná funkce s přirozeným exponentem je každá funkce daná vztahem* $f(x) = x^n$*, kde* $n \in N$*.*

- $f(x) = x^n$, $n \in N$, kde n je **liché**



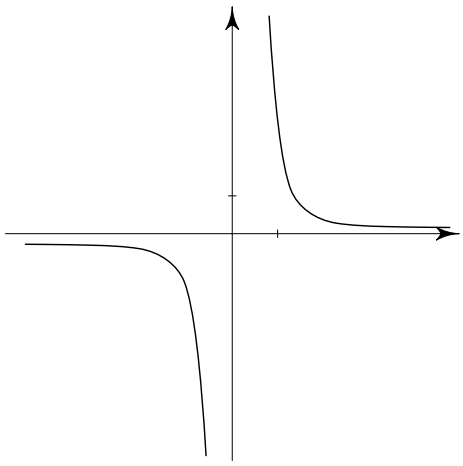
Obrázek 21: Graf funkce $f(x) = x^3$

- $f(x) = x^n$, $n \in N$, kde n je **sudé**



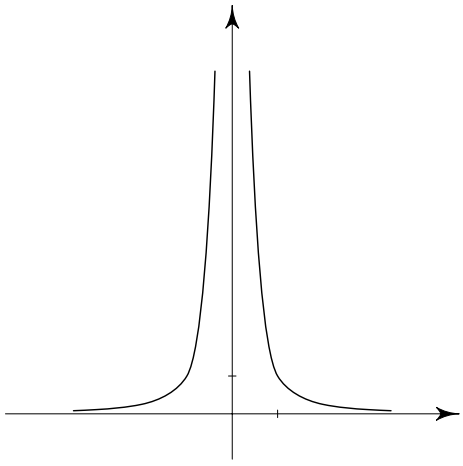
Obrázek 22: Graf funkce $f(x) = x^2$

- $f(x) = x^{-n}$, $n \in N$, kde n je **liché**



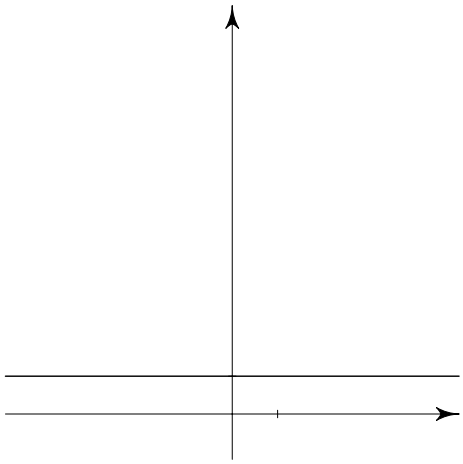
Obrázek 23: Graf funkce $f(x) = x^{-3}$

- $f(x) = x^{-n}$, $n \in N$, kde n je **sudé**



Obrázek 24: Graf funkce $f(x) = x^{-2}$

Existuje také případ, kdy $n = 0$. Funkce má potom tvar $f(x) = 1$ a jde o *konstantní* funkci.



Obrázek 25: Graf funkce $f(x) = 1$

4.3. Exponenciální funkce

Definice 7 *Exponenciální funkce je každá funkce daná předpisem* $f(x) = a^x$*, kde* $a > 0$*,* $a \neq 1$*.*

Proměnná je v tomto případě v exponentu, nikoli v základu, jak tomu bylo u funkcí mocninných. Speciálním případem exponenciální funkce je *přirozená exponenciální funkce*, jejíž základ je roven přímo Eulerovu číslu[3]. Pomocí této funkce lze popsat mnoho jevů, jako například pohlcování elektromagnetického záření.

4.4. Logaritmické funkce

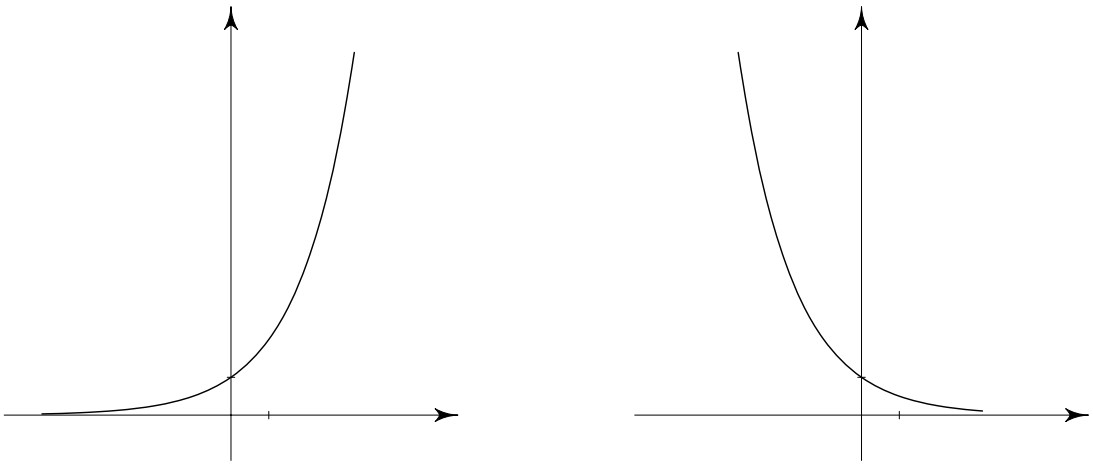
Definice 8 *Logaritmická funkce o základu* a *je funkce daná předpisem* $f(x) = \log_a x$*, kde* $a \in R$*,* $a > 1$*,* $a \neq 0$*.*

Platí, že exponenciální a logaritmická funkce o stejném základu jsou vzájemně inverzní funkce, tedy $y = \log_a x \Leftrightarrow x = a^y$. Srovnáme-li grafy logaritmické a exponenciální funkce, vidíme, že jsou souměrné podle přímky $y = x$. Logaritmus, jehož základem je Eulerovo číslo, se nazývá *přirozený logaritmus* $\ln x$. Pomocí něj

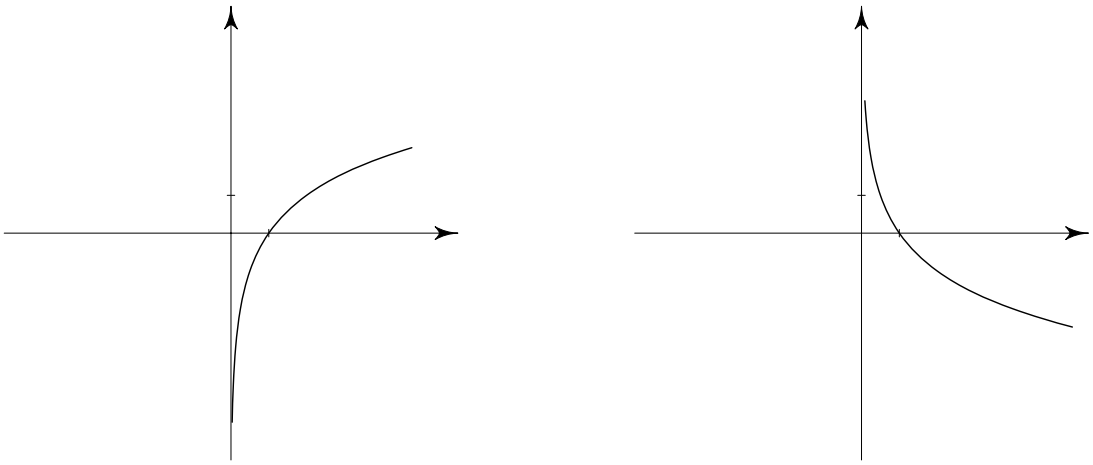
[3]jedna ze základních matematických konstant, $e \doteq 2.71828182846$

lze vyjádřit logaritmus o libovolném základu, a to vztahem

$$\log_a x = \frac{\ln x}{\ln a}.$$



Obrázek 26: Graf funkce $f(x) = a^x$, $a > 1$; $f(x) = a^x$, $0 < a < 1$

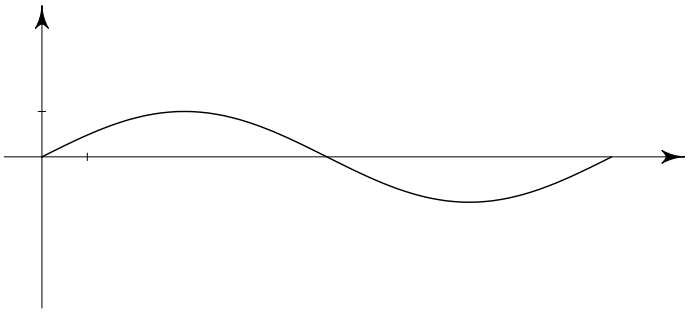


Obrázek 27: Graf funkce $f(x) = \log_a x$, $a > 1$; $f(x) = \log_a x$, $0 < a < 1$

4.5. Goniometrické funkce

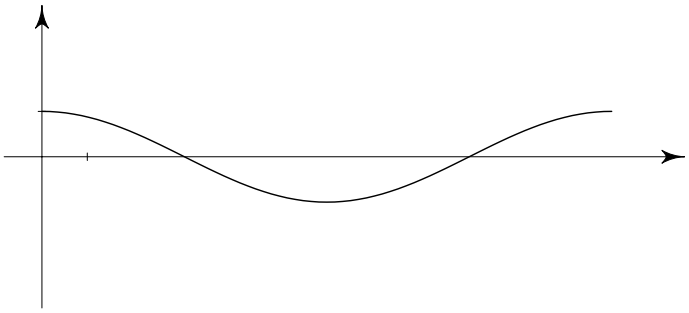
Ke goniometrických funkcím řadíme funkce *sinus*, *kosinus*, *tangens* a *kotangens*. Lze je definovat více způsoby, nejčastěji se využívá **jednotková kružnice**. Je to kružnice, jejíž poloměr je délky jedna a střed se nachází ve středu souřadnicového systému, tedy v bodě $S = [0; 0]$. Sestrojíme-li orientovaný úhel o velikosti t, jehož vrchol je ve středu této kružnice a počáteční rameno splývá s kladným směrem osy x, pak jeho koncové rameno protne kružnici v bodě $[\cos t; \sin t]$.

- funkce **sinus**



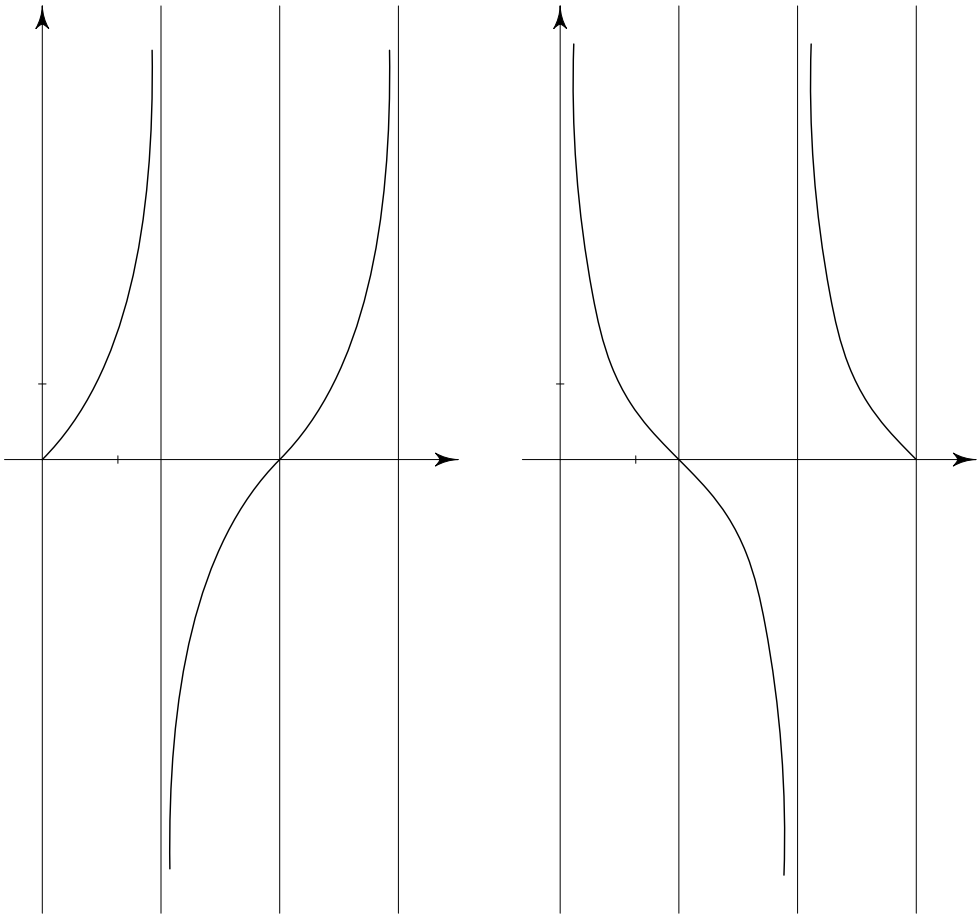
Obrázek 28: Graf funkce $f(x) = \sin x$

- funkce **kosinus**



Obrázek 29: Graf funkce $f(x) = \cos x$

- funkce **tangens**
  - dána rovnicí $f(x) = \frac{\sin x}{\cos x}$
- funkce **kotangens**
  - dána rovnicí $f(x) = \frac{\cos x}{\sin x}$



Graf funkce $f(x) = \operatorname{tg} x$ Graf funkce $f(x) = \operatorname{cotg} x$

Obrázek 30

Závěr

Ve své bakalářské práci jsem se zabývala programovacím jazykem Metapost, a to v souvislosti s grafy elementárních funkcí. Kromě popisu potřebných příkazů a zdrojových textů jsem stručně uvedla základní elementární funkce. Účelem bylo ukázat, co ve zdrojovém souboru jednotlivé příkazy znamenají, jakým způsobem se zadávají, umět vysvětlit tvar získaného grafu konkrétní funkce a sledovat, jak se změní při změně parametrů ve zdrojovém textu. Jako názornou ukázku a pro názornější představu obsahuje práce Metapostem vykreslené grafy nejen základních funkcí, ale také jejich různých transformací.

Hlavním cílem práce bylo ukázat, že i když to tak na první pohled nevypadá, je při kreslení čárové grafiky Metapost nenáročným a užitečným pomocníkem. Jeho umění může využít každý, kdo chce dosáhnout přesně vykreslených obrázků. Stačí porozumět potřebným příkazům, pomocí nich nadefinovat, co po Metapostu požadujeme, a on práci odvede za nás. Navíc, při kreslení analogických obrázků, ušetřme nejen práci, ale i čas. Bylo by možné zabývat se jeho dalšími funkcemi, jako je řešení lineárních rovnic nebo pokročilejší grafika, ty už ale přesahují rámec tématu.

Práce s Metapostem mě utvrdila v tom, že pokud budu v budoucnu potřebovat nakreslit obrázek, vždy ho použiji. Díky zvolené problematice jsem se dozvěděla o tvůrci TEXu Donaldu Knuthovi, zdokonalila se v práci s TEXem, naučila se používat některé příkazy Metapostu a v neposlední řadě jsem si zopakovala poznatky o elementárních funkcích. Někdy pro mě bylo obtížnější nalézt správný postup při programování Metapostu, neboť jsem se s Metapostem setkala při psaní práce poprvé. O to lepší pocit to byl, když se graf pěkně vykreslil. Práce s Metapostem při kreslení grafů pro mne byla velkým přínosem.

Myslím si, že o „umění Metapostu“ by neměl být ochuzen nikdo, kdo pracuje s jakoukoli grafikou. I studenti ve školách s technickým zaměřením nebo přírodovědným zaměřením by jeho funkce jistě využili. Pokud budu mít možnost, ráda mé touto prací získané znalosti předám někomu dalšímu. Totiž, proč bychom dělali věci složitě, když jdou udělat jednoduše.

Literatura

[1] Stříž, P.: Sazba v TEXu a kresba v Metapostu (1. vydání). Bučovice: Nakladatelství Martin Stříž, 2011.

[2] Rybička, J.: LATEX pro začátečníky (2. vydání). Brno: KONVOJ, 1999.

[3] Zdráhal, T.: Zobecněné elementární funkce (1. vydání). Ústí nad Labem: Univerzita J. E. Purkyně v Ústí nad Labem, 2006.

[4] Kubešová, N., Cibulková, E.: MATEMATIKA – přehled středoškolského učiva (2. vydání). Třebíč, 2007.

[5] Kubát, J.: . Sbírka úloh z matematiky pro přípravu k maturitní zkoušce a k přijímacím zkouškám na vysoké školy (1. vydání). Praha: Prometheus, spol. s.r.o., 2004.

[6] www.cstug.cz

[7] www.root.cz/clanky/metapost-datove-typy

[8] www.tug.org/docs/html/metapost

[9] wwww.matematika.cz

[10] programujte.com/clanek/2009032600-donald-ervin-knuth/

[11] apfyz.upol.cz/ucebnice/down/mini/drslat.pdf

[12] phoenix.inf.upol.cz/propedeutika/2/practice4.html

[13] www.cdm.cas.cz/publications/hora/ph_rep2005_mmp.pdf

[14] pav.kourilova.sweb.cz/skriptum-kap7-elfce.pdf

[15] https://cs.wikipedia.org