Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra matematiky

Bakalářská práce

Úvod do programu wxMaxima

Vypracoval: Lukáš Filip
Vedoucí práce: Mgr. Roman Hašek, Ph.D

České Budějovice 2013

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma wxMaxima jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích .....

…….

Děkuji vedoucímu mé bakalářské práce Mgr. Romanu Haškovi za skvělý nápad vytvoření úvodu do programu wxMaxima a tím ho přiblížit širší veřejnosti. Také mu děkuji za trpělivost a za odborné vedení.

Anotace

Název: Úvod do programu wxMaxima
Vypracoval: Lukáš Filip
Vedoucí práce: Mgr. Roman Hašek, Ph.D.

Klíčová slova: wxMaxima, analýza, algebra, příkazy

Obsahem této práce je vytvoření stručného, ale kompletního úvodu do práce s programem wxMaxima při výpočtech na úrovni základní a střední školy. Práce je doplněna o ukázkově řešené příklady a také o ukázková videa pro názornější a rychlejší představení vybraných úkonů a operací prováděných s programem.

Abstract

Title: Introduction to wxMaxima
Author: Lukáš Filip
Supervisor: Mgr. Roman Hašek, Ph.D.

Key words: wxMaxima, calculus, algebra, commands

The subject of this thesis is creating a brief but complex introduction into the wxMaxima program and its usage for calculations in basic schools and high schools. Part of the thesis are also solved examples and videos for a more transparent presentation of some chosen activities and operations dealt with by using the program.

OBSAH

1 ÚVOD

Program wxMaxima patří mezi nejlépe vybavené matematické nástroje. Jeho užívání ale nemusí být pro každého snadné. Proto se pokusím přiblížit základní syntaxi a užívání některých důležitých funkcí, všem zájemcům o program.

Touto bakalářskou prací bych chtěl vytvořit stručný úvod do práce s programem wxMaxima při výpočtech na úrovni základní a střední školy. Práce je doplněna o ukázkové řešené příklady a komentovaná videa.

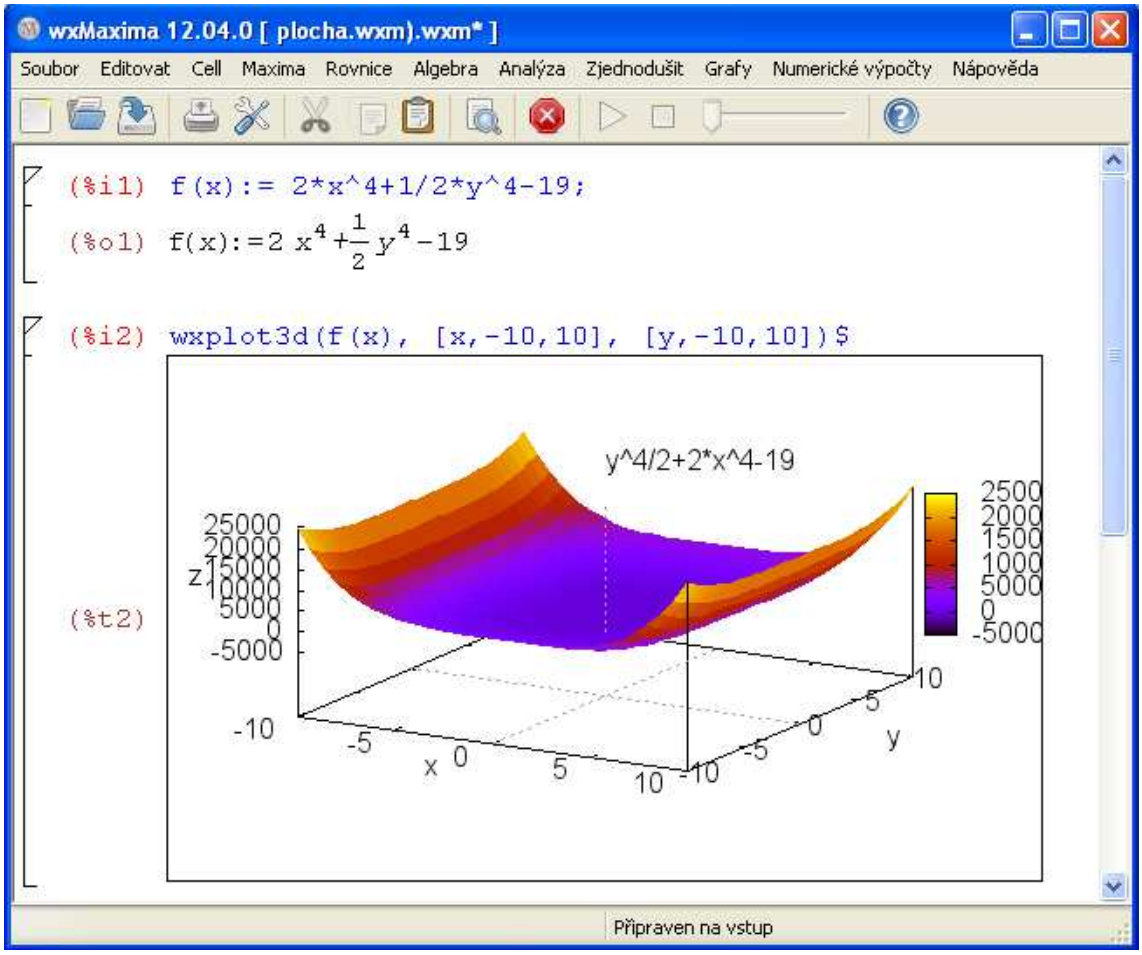
1.1 Historie

Maxima je pokračovatel DOE Macsyma, která vznikla roku 1960. Je to jedinečný systém, díky svému open source charakteru. Macsyma byl první svého druhu počítačové algebry, vytvořil cestu pro programy jako je Maple a Mathematica.[1]

Program wxMaxima představuje grafickou nadstavbu původního programu Maxima. Tato nová verze umožňuje pohodlnější užívání i začínajícím uživatelům. Program funguje na počítačích s operačním systémem Windows i s operačním systémem Linux. Aplikace je totiž multiplatformní (spustitelná v různých operačních systémech). Navíc nabízí výrazně nižší hardwarové požadavky, pohodlnější ovládání a nulovou cenu. Pro studenty je určitě nevýhodou, že program daný matematický problém sice vyřeší, ale už nenabídne postup, jak se k výsledku dopracovat. [3]

2 SEZNÁMENÍ S PROSTŘEDÍM

Po zapnutí programu se nám na monitoru zobrazí hlavní okno, ve kterém si můžeme všimnout několika záložek, obrázků a velkého pracovního pole. Tyto záložky nám pomáhají usnadnit práci s programem i bez kompletní znalosti syntaxe. Podrobněji si některé popíšeme v následující kapitole.



Obrázek 1: Plocha programu

2.1 Soubor

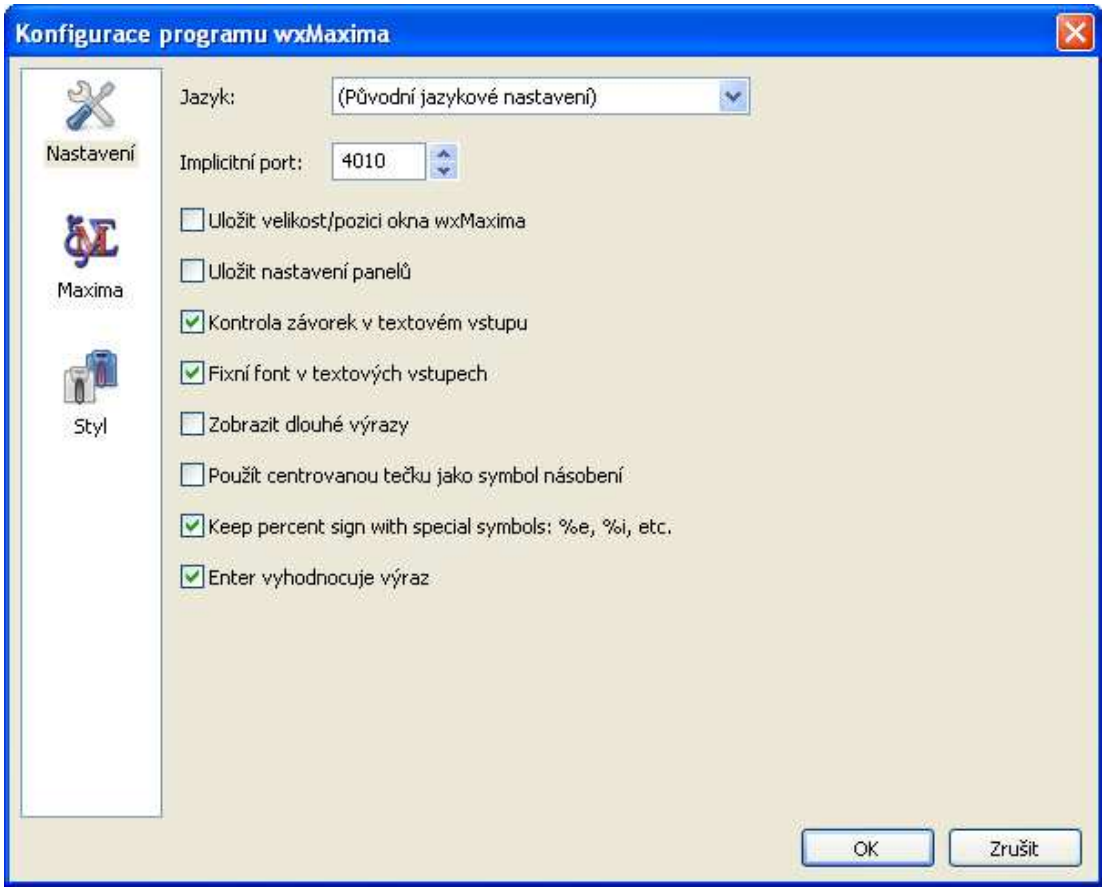
Export – Umožňuje převedení našeho projektu, který se ukládá pouze s koncovkou .wxm. Převod můžeme uskutečnit buďto do publikačního programu TeX s koncovkou .tex nebo můžeme převod uskutečnit do formátu .html, který nám umožňuje vložení příkladů na webové stránky.

2.2 Editovat

Najdi – Pokud v naší práci potřebujeme najít určité slovo nebo výraz, použijeme tento příkaz a on nám vyhledá v textu potřebné slovo. Umožňuje nám také tyto slova nahradit za jiná, což nám ve zdlouhavých textech ulehčí práci s přepisováním chyb.

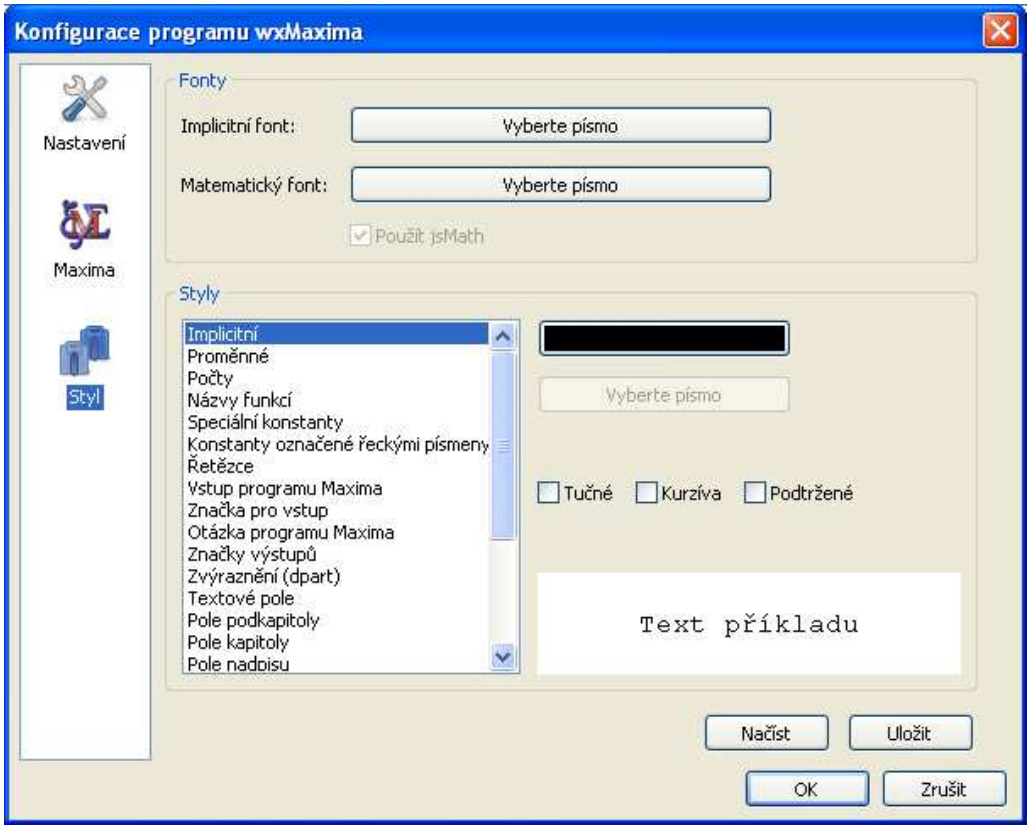
Celá obrazovka – Před použití této funkce je důležité, abyste se podívali, jakou klávesovou zkratkou se funkce Celé obrazovky spouští. Protože pomocí těchto kláves Alt+Enter se nám pracovní plocha roztáhne přes celou obrazovku a jedině stisknutím stejné kombinace kláves se můžeme přepnout zpět do původního nastavení. Pokud tuto kombinaci neznáme, potom již nemáme šance se z roztáhlé plochy dostat.

Nastavení – Důležitá část pro následující práci v programu. Například si zde můžeme změnit způsob vyhodnocení výrazu na samotné tlačítko Enter, které je prvotně nastaveno na skok na další řádek a vyhodnocení příkazu se vykonává pomocí Shift+Enter. Okno pro nastavení vidíme na obrázku 2.



Obrázek 2: Nastavení

Nastavení (Styl) – Spíše zajímavá položka, která nabízí uživateli nastavit si jinou barvu písma u vzorců než u výsledků. Jiné podbarvení u nadpisů než u textového pole. Také dovoluje nastavit font, řez a velikost písma.



Obrázek 3: Nastavení_styl

2.3 Pole

Zde nalezneme vše potřebné, pokud chceme v programu psát komentáře nebo používat program jako textový editor. Můžeme zde vkládat nadpisy různých důležitostí nebo samotný text. Můžeme také vložit obrázek, který se vloží ve své skutečné velikosti a již dále nejde upravovat, proto je potřeba ho upravit na potřebnou velikost ještě před vložením. Na obrázku číslo 4 je uveden příklad, na kterém vidíme, jak v programu vypadá použití nadpisu, textového pole, obrázku s titulkem a matematického vstupu a výstupu.

Úloha na dělení

Maminka má doma 6 dětí, jakým způsobem rozdělí 72 jablek, aby každé z jejich dětí mělo stejně.

Figure 1: Jablka

```
(%i1) 72/6;
(%o1) 12
```

Obrázek 4: Pole

2.4 Maxima

Restart programu Maxima – program vymaže číslování řádků a začíná počítat znovu od jedničky. To se hodí především pro přehlednost, pokud výpočty několikrát v průběhu opravujeme a máme proto přeházené číslování řádků. Pouze zvolíme restart programu a ostatní výpočty začínají od číslice jedna.

Zobrazit proměnné – vypíše všechny proměnné, které jsme během počítání zadali. Jak přiřadit proměnné hodnotu si vysvětlíme v kapitole 3.

Zrušit proměnnou – po zvolení této možnosti se nám zobrazí samostatné okno, do kterého zadáme proměnou, kterou již nebudeme chtít používat. Zrušit proměnnou můžeme také pomocí příkazu „kill“ nebo „remvalue“ a do závorky napíšeme název proměnné.

Změnit 2D výstup – Umožňuje měnit nám styl výstupu. Po kliknutí na tuto možnost se nám ukáže tabulka, která nabízí 3 možnosti: xml, ascii, none. Na obrázku 5 vidíme, jak vypadá stejný výsledek v jiném výstupu.

```
výstup 2D nastaven na xml (primární nastavení)

(%i1) set_display('xml)$

(%i2) log(x^3)/sqrt(5)+b^2/3 ;
```

$$(\%o2)\quad \frac{\log(x^3)}{\sqrt{5}}+\frac{b^2}{3}$$

```
výstup 2D nastaven na ascii

(%i3) set_display('ascii)$

(%i4) log(x^3)/sqrt(5)+b^2/3 ;
                                        3      2
                                   log(x )    b
(%o4)                              ------- + --
                                   sqrt(5)    3

výstup 2D nastaven na none

(%i5) set_display('none)$

(%i6) log(x^3)/sqrt(5)+b^2/3 ;
(%o6) log(x^3)/sqrt(5)+b^2/3
```

Obrázek 5: 2D výstup

2.5 Numerické výpočty

Převést na float – Pokud se nám výsledek vypisuje ve tvaru zlomku, odmocniny nebo například obsahuje konstantu π a my potřebujeme mít výsledek ve tvaru desetinného čísla, použijeme příkaz „float“. Na dalším řádku se nám vypíše výsledek s přesností na 15 desetinných míst.

Nastavení přesnosti – V předchozím příkazu se výsledek vypisoval na 15 desetinných míst, ale může se stát, že výsledek budeme chtít mít na jiný počet desetinných míst. Proto existuje příkaz „fpprec“, kterým si můžeme nastavit přesnost. Výsledek nebude vidět hned, ale vytvoříme ho pomocí následujícího příkazu.

Přesnost bigfloat – Po předchozím nastavení desetinných míst pomocí příkazu „fpprec“ vypíšeme požadovaný výsledek pomocí příkazu „bfloat“

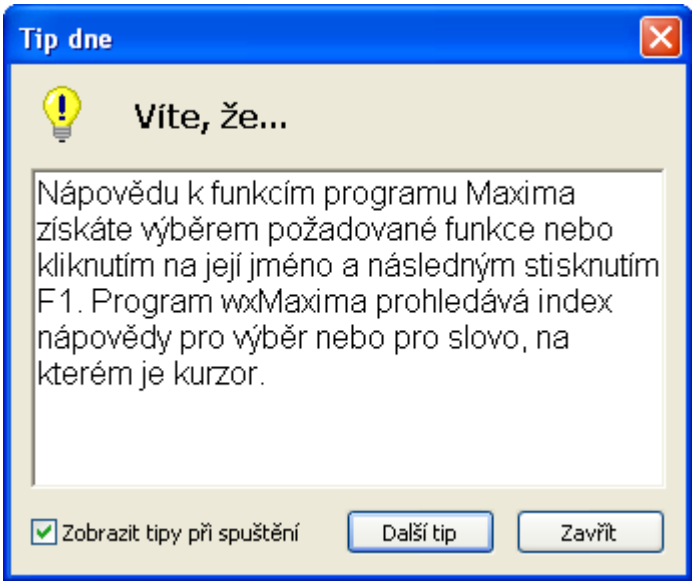
```
(%i1) sqrt(2);
(%o1) √2

(%i2) float(sqrt(2));
(%o2) 1.414213562373095

(%i3) fpprec : 40;
(%o3) 40

(%i4) bfloat(sqrt(2));
(%o4) 1.4142135623730950488016887242096980785_7b0
```

2.6 Nápověda

Příklad – Pokud si nebudeme jistí, který příkaz k čemu slouží, využijeme této chytré nápovědy, která vypíše k požadovanému příkazu vzorové příklady. Na následující ukázce jsme použili příkaz „example“, který slouží k vyvolání příkladu. Použili jsme ho na příkaz „diff“ (derivace) a program nám ukazuje, v jakých situacích se dá námi zvolený příkaz použít.

```
(%i1) example(diff);
(%i2) kill(f,g,h,x,y)
(%o2) done
(%i3) diff(2*x^2+x^3+sin(x),x)
(%o3) cos(x)+3 x^2 +4 x
(%i4) diff(sin(x)*cos(x),x)
(%o4) cos(x)^2 -sin(x)^2
(%i5) diff(sin(x)*cos(x),x,2)
(%o5) -4 cos(x)sin(x)
```

Ukázat tipy – Je to tabulka, která se nám automaticky zobrazuje při startu programu a vypisuje chytré návrhy jak používat program. Použitím tohoto příkazu můžeme tabulku vyvolat i v průběhu naší práce. Mezi návrhy můžeme libovolně přepínat kliknutím na tlačítko „Další tip“.



Obrázek 6: Tip

3 ZÁKLADNÍ POČETNÍ OPERACE

Důležité pro začátek práce v programu je znát, jak se zapisují znaky různých operací. Klasické sčítání „ + “ a odečítání „ - “je stejné jako v každém jiném programu. Násobení se zapisuje pomocí hvězdičky „ * “. Nestačí pouze zápis 2x, ale zápis musí být ve tvaru 2*x, jinak program vypíše chybovou hlášku a výsledek nevypočítá. Dělení se zapisuje pomocí lomítka „ / “.

Pro operaci umocňování se dají použít dva možné zápisy, první možnost je napsat dvě hvězdičky vedle sebe. Druhá používanější možnost je zapsat znak stříšky „^“ (pravý Alt + š). Symbol pro zápis odmocniny program wxMaxima nemá, a proto se odmocnina zapisuje pomocí příkazu „sqrt“. Vyšší odmocniny zapisujeme jako mocninu s racionálním exponentem.

```
(%i1) sqrt(36);
(%o1) 6

(%i2) 32^(1/5);
(%o2) 2
```

Program také neumí pracovat s přirozeným logaritmem, a proto musíme zadávat logaritmus ve tvaru „log“. Pro zapsání čísla v absolutní hodnotě se používá příkaz „abs“.

Jelikož je program vytvořen v Americe, používá se pro oddělení jednotek a desetin desetinná tečka, místo pro nás obvyklé desetinné čárky.

Zápis nejdůležitějších konstant:

π – Ludolfovo číslo	%pi
e - Eulerovo číslo	%e

i – imaginární jednotka	%i
∞ - kladné nekonečno	Inf
-∞ - záporné nekonečno	Minf
∞ - komplexní nekonečno	Infinity
Pravda	True
Nepravda	False

Pokud budeme chtít některé proměnné přiřadit hodnotu, musíme použít symbol dvojtečky. Tím dostane daná proměnná svou hodnotu, kterou si bude udržovat po celou dobu výpočtů, do té doby než jí bude přiřazena nová hodnota. Pokud v průběhu počítání potřebujeme použít stejnou proměnnou s jinou hodnotou, ale nechceme ji přepisovat, můžeme použít příkaz „ev“ do jehož těla vložíme výraz, který chceme vypočítat a hodnoty proměnných oddělené čárkou. Na následujícím příkladu vidíme, jak program pracuje s novými hodnotami pouze v příkazu „ev“. Hodnoty v tomto příkazu můžeme zadávat dvojtečkou nebo rovnítkem.

```
(%i1) a:3;
(%o1) 3

(%i2) b:5;
(%o2) 5

(%i3) c=a+b;
(%o3) c=8

(%i4) ev(d=a+b,a=1,b=2);
(%o4) d=3

(%i5) e=b/a;
(%o5) e=5/3
```

Pokud budeme chtít přiřadit hodnotu funkci, použijeme spojení dvojtečky a rovnítka „ := “.

```
(%i1) f(x):= 12*x^3+8*x;
(%o1) f(x):=12 x^3 +8 x

(%i2) g(x):= f(x)/2;
(%o2) g(x):=f(x)/2

(%i3) ratsimp(g(x));
(%o3) 6 x^3 +4 x
```

Středník na konci řádku se vypisuje automaticky po stisknutí klávesy Enter. Znak dolaru „ $ “ se zapisuje za příkaz, pokud chceme více operací vykonat v jednom řádku. V následujícím příkladu vidíme v prvním řádku, jak se uloží hodnota pouze do proměnné *a*. Ostatní operace se nevykonají. Po použití znaku dolaru se provedou i ostatní operace. Znak apostrofu „ ‘ “ se zapisuje před příkaz, pokud nechceme, aby se vykonal, ale pouze opsal na další řádek.

```
(%i1) a:6 b:3 d=sqrt(a+b);
incorrect syntax: B is not an infix operator

(%i2) a:6$  b:3$  d=sqrt(a+b);
(%o4) d=3

(%i5) determinant(matrix([1,2],[3,1]));
(%o5) -5

(%i6) 'determinant(matrix([1,2],[3,1]));
(%o6) determinant([1 2; 3 1])
```

4 POKROČILÉ VÝPOČTY

4.1 Nejmenší společný násobek a největší společný dělitel

Čísla, která jsou násobky každého ze dvou nebo více kladných celých čísel, se nazývají společné násobky těchto čísel. Nejmenšímu z nich říkáme nejmenší společný násobek. Označujeme ho „n“.

Pokud budeme chtít spočítat nejmenší společný násobek několika čísel, použijeme příkaz „lcm“ z anglického least common multiple. Do závorek poté napíšeme čísla oddělená čárkou, z kterých se má násobek vypočítat.

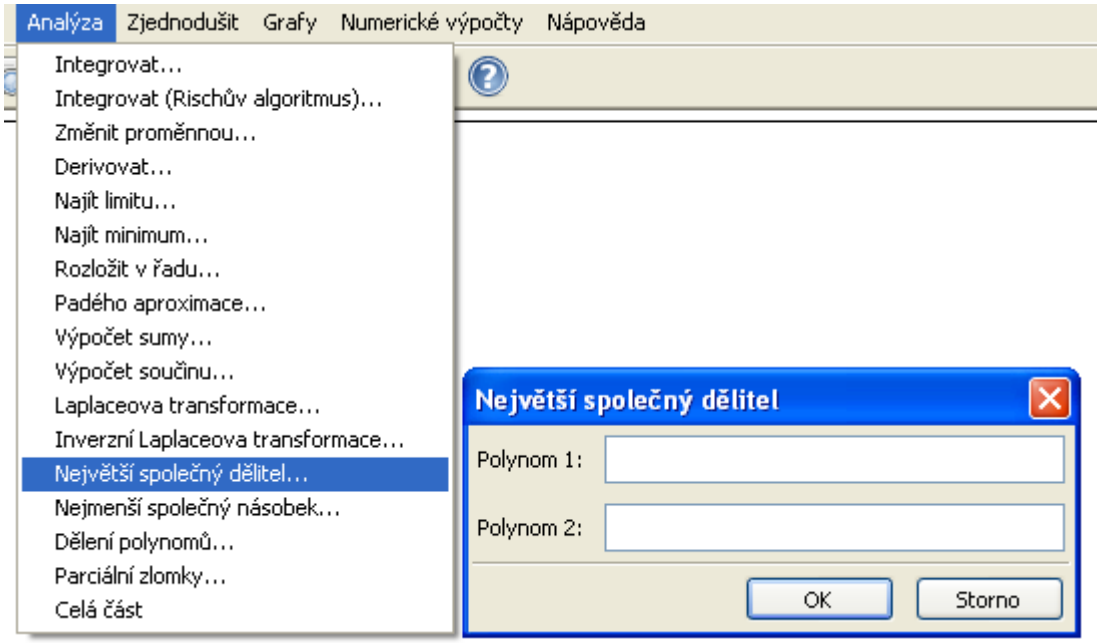
Druhá možnost vypočítání nejmenšího společného násobku je taková, že v horní liště pod záložkou Analýza, zvolíme možnost Nejmenší společný násobek. Tato metoda skrývá nevýhodu v počtu čísel, která chceme zadat. Vyskočí nám totiž okno s pouze dvěma řádky, což umožňuje nacházet nejmenší společný násobek pro dvě čísla.

```
(%i11) lcm(84,12);
(%o11) 84

(%i10) lcm(84,16,56);
(%o10) 336
```

Čísla, která jsou děliteli každého ze dvou nebo více kladných celých čísel se nazývají společní dělitelé těchto čísel. Největšímu z nich říkáme největší společný dělitel. Je to největší číslo, kterým jsou daná čísla dělitelná. Označujeme ho „D“.

Pokud budeme chtít spočítat největší společný dělitel dvou čísel, použijeme příkaz „gcd“ z anglického greatest common divisor. Do závorek poté napíšeme dvojici čísel oddělených čárkou, z kterých se má nejmenší společný násobek vypočítat.

Druhá možnost vypočítání největšího společného dělitele je taková, že v horní liště pod záložkou Analýza zvolíme možnost Největší společný dělitel.



Obrázek 7: Největší společný dělitel

Najít největšího společného dělitele wxMaxima umožňuje pouze pro dvě čísla. Abychom nemuseli takovéto příklady rozepisovat do více řádků, můžeme zanořit více příkazů do sebe.

```
(%i1) gcd(84,24);
(%o1) 12

(%i2) gcd(84,24,56);
Improper argument to gcd:
56
 -- an error. To debug this try: debugmode(true);

(%i3) gcd(gcd(84,24),56);
(%o3) 4
```

Příklad 4.1.1.: Učitelka první třídy paní Britnerová měla na začátku školního roku spravedlivě rozdělit mezi žáky 87 tužek, 45 malých a 116 velkých sešitů. Kolik dětí bylo ve třídě? Kolik tužek, malých a velkých sešitů dostal každý žák?

Určením největší společného dělitele všech tří čísel zjistíme počet všech žáků ve třídě.

```
(%i1) gcd(gcd(87,145),116);
(%o1) 29
```

Vydělením celkového množství jednotlivých školních potřeb počtem žáků ve třídě, získáme hodnotu odpovídající počtu školních pomůcek pro každého žáka.

```
(%i2) tuzek= 87/29;
(%o2) tuzek=3
```

```
(%i3) malych= 145/29;
(%o3) malych=5
```

```
(%i4) velkych= 116/29;
(%o4) velkych=4
```

Do třídy paní Britnerové chodí 29 žáků. Každý žák dostane na začátku školního roku 3 tužky, 5 malých a 4 velké sešity.

Příklad 4.1.2.: V 9:00 se na zastávce setkaly tři autobusy místní dopravy. První autobus má intervaly po 20 minutách, druhý po 25 minutách a třetí po 30 minutách. V kolik hodin se opět setkají na této zastávce?

Nejprve vypočítáme nejmenší společný násobek všech tří časů a tím zjistíme, za kolik minut se autobusy znovu setkají.

```
(%i1) lcm(20,25,30);
(%o1) 300
```

K původnímu setkání v devět hodin přičteme vypočítaný čas převedený na hodiny.

```
(%i2)  9+300/60;
(%o2)  14
```

Všechny tři autobusy MHD se opět potkají ve 14 hodin.

4.2 Zjednodušení výrazů a substituce

Pro zjednodušování výrazů máme na výběr z několika možných příkazů, které nám program nabízí. Každý příkaz má trochu odlišnou funkci a využití. Představíme si nejdůležitější příkazy. Zápis je pro všechny stejný a jednoduchý, stačí napsat název příkazu a do závorky zapsat výraz, který se má zjednodušit. Nakonec si také ukážeme použití substituce, která nám pomůže usnadnit počítání.

Nejdůležitější příkaz pro zjednodušování výrazů je bezesporu příkaz „ratsimp“. V horní liště ho nalezneme v záložce Zjednodušit – Zjednodušit výraz. Jeho síla je v tom, že dokáže zjednodušit téměř jakýkoliv výraz

```
(%i1)  a:(sqrt(a^2-1)+a)^2-(sqrt(a^2-1)-a)^2;
(%o1)  (sqrt(a^2-1)+a)^2-(sqrt(a^2-1)-a)^2
```

```
(%i2)  ratsimp(a);
(%o2)  4 a sqrt(a^2-1)
```

Jeden z příkazů nám slouží k roznásobení závorek a nazývá se „expand“. V horní liště ho můžeme najít v záložce Zjednodušit – Rozložit výraz. Jeho opakem je příkaz „factor“, který roznásobené výrazy opět skládá dohromady na součin výrazů. Najdeme ho v záložce Zjednodušit – Na součin. Opět je z názvu vidět co daný příkaz provádí a můžeme očekávat, že výsledek dostaneme ve tvaru součinu.

```
(%i1) m:(a-3)*(b-2)*(b+5);
(%o1) (a-3)(b-2)(b+5)

(%i2) n: expand(m);
(%o2) a b^2-3 b^2+3 a b-9 b-10 a+30

(%i3) factor(n);
(%o3) (a-3)(b-2)(b+5)
```

Další příkaz, který si uvedeme, je příkaz „radcan“. Tento velmi důležitý příkaz nám pomáhá hlavně se zjednodušováním složitých odmocnin, jak vyplývá i z jeho označení v horní liště. Najdeme jo pod názvem Zjednodušit výraz s odmocninami v záložce Zjednodušit.

```
(%i1) a: (b+sqrt(b^2-2))/(b-sqrt(b^2-2))+(b-sqrt(b^2-2))/(b+sqrt(b^2-2));
(%o1) (b-sqrt(b^2-2))/(sqrt(b^2-2)+b)+(sqrt(b^2-2)+b)/(b-sqrt(b^2-2))

(%i2) radcan(a);
(%o2) 2 b^2-2

(%i3) b: sqrt(1-1/sqrt(2))/sqrt(2);
(%o3) sqrt(1-1/sqrt(2))/sqrt(2)

(%i4) radcan(b);
(%o4) sqrt(sqrt(2)-1)/2^(3/4)
```

Pro počítání s komplexními čísly nejlépe slouží jediný příkaz a to je příkaz „rectform“. Jako jediný příkaz nám ve výsledku vrátí oddělenou reálnou složku od imaginární složky.

```
(%i1) a: (1+%i)/(1+2*%i);
                          %i + 1
(%o1)                   ---------
                        2 %i + 1

(%i2) rectform(a);
                        3   %i
(%o2)                   - - --
                        5   5
```

Substituce neboli nahrazení nám ve složitých příkladech ulehčuje orientaci v průběhu počítání tím, že si za složitý výraz dosadíme jednoduchou proměnnou, kterou ve výsledku opět nahradíme původním výrazem. Příkaz pro použití substituce je „subst“ a skládá se ze tří částí. Do první části zapíšeme znak, který chceme použít k nahrazení. Ve druhé části zapíšeme výraz, který budeme chtít nahradit. Třetí část musí obsahovat výraz, ve kterém budeme substituci provádět nebo alespoň odkaz na tento výraz. Po provedení substituce se již s příkladem dále nic neděje.

Příkaz „subst“ nám nahradí pouze výraz, který stojí samostatně nebo je uzavřen v závorkách. Pokud tomu tak není, substituce se neprovede. K této nápravě slouží příkaz „ratsubst“, který provede substituci kompletně a zároveň ještě celý příklad zjednoduší.

```
(%i1)  a:  (x*(v+z)+v*x)/(v+z+2*x);
                          x (z + v) + v x
(%o1)                     ---------------
                          z + 2 x + v
```

```
(%i2)  subst(A,v+z,a);
                          x A + v x
(%o2)                     ---------
                          z + 2 x + v
```

```
(%i3)  ratsubst(A,v+z,a);
                          x (A + v)
(%o3)                     ---------
                          A + 2 x
```

4.3 Rovnice

Pro řešení rovnic nám slouží velmi důležitý příkaz „solve“, který v překladu z angličtiny znamená „vyřešit“. Příkaz se skládá z dvou částí. První část obsahuje zápis rovnice, ve kterém pokud nebude znak rovná se, tak wxMaxima automaticky považuje příklad za rovný nule. Druhá část příkazu označuje, kterou neznámou má program vypočítat.

Pro řešení soustavy lineárních rovnic slouží příkaz „linsolve“, který se jako předchozí příkaz pro řešení jedné rovnice, skládá ze dvou částí. V první části se zápisy rovnic od sebe oddělují čárkou, stejně jako ve druhé části zápisy neznámých. Jediný rozdíl mezi oběma zápisy je takový, že první i druhá část se vkládá do hranatých závorek.

```
(%i1) solve(x+3-2*x+6*x-5=5,x);
(%o1) [x=7/5]

(%i2) solve((2*z-1)/(5-z)+4=(3+2*z)/(z-2), z);
(%o2) [z=53/16]

(%i3) linsolve([3*x+2*y=8, x-5*y=-3], [x,y]);
(%o3) [x=2,y=1]

(%i4) linsolve([a*x+y=2, (a*x)*2+a*x=4], [x,y]);
(%o4) [x=4/(3 a),y=2/3]
```

Rovnice můžeme také zadávat bez znalosti příkazů, pokud si v horní liště v záložce Rovnice hned na prvním řádku klikneme na Řešit (solve) nebo níže na Řešit lineární systém (linsolve). Po kliknutí na možnost Řešit lineární systém, se nám zobrazí nejprve první tabulka, která se nás ptá, s kolika rovnicemi budeme pracovat. Po zadání potřebného čísla se nám zobrazí druhá tabulka, která se zobrazí také po kliknutí na možnost Řešit. V ní zadáme rovnici a proměnnou nebo rovnice a proměnné. Výsledkem našeho zadávání budou vypočítané hodnoty proměnných.

Příklad 4.4.: Dívky ze dvou tříd si chtějí společně koupit dva volejbalové míče. Jestliže každá z nich přinese 25 Kč, bude jim chybět 200 Kč. Přinese-li každá 32 Kč, zůstane jim 24 Kč. Kolik děvčat je v obou třídách dohromady?

Nejprve si sestavíme rovnice. Počet dívek ve třídě si označíme jako *x*, cenu volejbalového míče jako *m*.

```
(%i1) 2*m=x*25+200;
(%o1) 2 m=25 x+200

(%i2) 2*m=x*32-24;
(%o2) 2 m=32 x-24
```

Vypočteme pomocí soustavy lineárních rovnic.

```
(%i3) linsolve([2*m=x*25+200, 2*m=x*32-24], [m,x]);
(%o3) [m=500, x=32]
```

Odpověď: Volejbalový míč stojí 500Kč. Ve třídách studuje 32 děvčat.

5 OBVOD, OBSAH, POVRCH a OBJEM

Pro výpočet jednoduchých úloh, kdy máme zadané délky stran a počítáme například obvod, nám postačí napsat součet jednotlivých délek a po zmáčknutí tlačítka Enter se nám na dalším řádku zobrazí výsledek. My ovšem budeme počítat jiným způsobem. Na začátku každého příkladu přiřadíme proměnným zadané hodnoty a budeme počítat právě s proměnnými.

Ve složitějších úlohách, kde vypočítáváme neznámou ze vzorečků, budeme pracovat s pomocí rovnic. Rovnice v programu nám ulehčí přesouvání neznámé na levou stranu rovnice a ostatních proměnných na stranu pravou.

Příklad 5.1.: Paní Petrásková si koupila 120 smrkových latí tvaru kvádru s rozměry 7 cm, 4 cm, 6,5 m. Vypočítejte jejich celkový objem v krychlových metrech. Vypočítejte celkovou cenu, jestliže za $1m^3$ latí zaplatila 3600 Kč.

Nejprve si zavedeme hodnoty do proměnných v metrech.

```
(%i1) a:7/100;
(%o1)                          7
                               ---
                               100
```

```
(%i2) b:4/100;
(%o2)                          1
                               --
                               25
```

```
(%i3) c:6.5;
(%o3)                          6.5
```

Nyní podle vzorečku pro výpočet objemu kvádru, vypočítáme objem jedné latě.

```
(%i4) V:a*b*c;
(%o4)                        0.0182
```

Jelikož paní Petrásková zakoupila 120 latí, musíme objem jedné latě vynásobit celkovým počtem.

```
(%i5) V120:V*120;
(%o5) 2.184
```

Když máme vypočítaný celkový objem, můžeme zjistit jeho celkovou hodnotu.

```
(%i6) cena=V120*3600;
(%o6) cena=7862.400000000001
```

Odpověď: Paní Petrásková zaplatí celkem 7862 Kč

Příklad 5.2.: Voda v krychlové nádobě o hraně 2 dm sahá 6 cm pod okraj. Vodu z krychlové nádoby přelijeme do nádoby tvaru válce s průměrem 9 cm. Jak vysoká musí nádoba být, aby se do ní veškerá voda vešla?

Nejprve přiřadíme hodnoty proměnným

```
(%i1) a:20;
(%o1) 20
```

```
(%i2) x:6;
(%o2) 6
```

```
(%i3) d:10;
(%o3) 10
```

Vypočteme objem vody, kterou obsahuje krychlová nádoba.

```
(%i4) V: a^3 - a^2*x;
(%o4) 5600
```

Vypočítáme, do jaké výšky bude sahat voda ve válcové nádobě.

```
(%i5)  v= V/(%pi*(d/2)^2);
(%o5)  v=224/π
```

Výsledek převedeme na numerický výstup, když klikneme v horní liště na záložku Numerické výpočty a zvolíme možnost Přepnout numerický výstup.

```
(%i6)  if numer#false then numer:false else numer:true;
(%o6)  true
```

```
(%i7)  v= V/(%pi*(d/2)^2);
(%o7)  v=71.30141450516911
```

6 MATICE

Matice je obdélníkové nebo čtvercové uspořádání prvků do řádků a sloupců. Matice obsahuje m řádků a n sloupců.

6.1 Zadání matice

Pokud budeme chtít zadat matici ručně, pak budeme postupovat následovně. Nejprve napíšeme označení matice, pod kterým jí můžeme následně vyvolávat a hned za ním napíšeme dvojtečku neboli symbol přiřazení. Tím již máme matici označenou a přichází to nejdůležitější, příkaz pro zadání matice. Matice se označuje slovem „matrix“. Do kulatých závorek pak budeme zadávat prvky matice. Pro přehlednější vkládání se každý řádek píše zvlášť do hranatých závorek. Jak hranaté závorky neboli řádky, tak prvky jednotlivých řádků se oddělují čárkou. Na konci našeho zápisu již stačí napsat známý středník a matice je hotová.

```
(%i1) A: matrix([5,3,1],[6,4,1],[2,4,6]);
                [ 5  3  1 ]
                [         ]
(%o1)           [ 6  4  1 ]
                [         ]
                [ 2  4  6 ]
```

Pokud si budeme chtít usnadnit práci, platí to hlavně u rozměrnějších matic, které mají více řádků a sloupců, můžeme použít druhý způsob zápisu, u kterého odpadá nutnost psaní závorek. Také nám ulehčí práci s maticemi, které jsou například symetrické, protože nebudeme muset vypisovat všechny členy, pouze zvolíme typ matice a ona své zbylé členy vypíše sama.

V horní liště najdeme záložku Algebra, ve které se nachází veškeré potřebné věci k práci s maticemi. Pro tuto chvíli nám postačí „Zadání matice“, k ostatním se dostaneme ještě později.

Po stisknutí řádku s nápisem Zadání matice se nám zobrazí okno, které vidíme na obrázku 8 a které nám nabízí čtyři řádky. První z nich se nás ptá, kolik budeme chtít, aby naše matice měla řádků. Druhý z nich se ptá, kolik budeme chtít, aby naše matice měla sloupců. Ve třetím řádku si zvolíme typ naší matice, na výběr máme z možností „obecná, diagonální, symetrická, antisymetrická“. V posledním řádku zapíšeme označení pro matici.

Obrázek 8: Tvorba matice

Pokud máme vše potřebné vyplněno, můžeme stisknout tlačítko OK a zobrazí se nám nové okno, do kterého zadáme hodnoty jednotlivých prvků matice a ve kterém již můžeme vidět, jak naše matice bude vypadat. Tím máme matici hotovou

```
(%i1) B: matrix(
        [5,3,1],
        [6,4,1],
        [2,4,6]
      );
           [ 5  3  1 ]
           [         ]
(%o1)      [ 6  4  1 ]
           [         ]
           [ 2  4  6 ]
```

6.2 Počítání s maticemi, hodnost matice

Základní operace s maticemi nejsou až na jednu výjimku o nic složitější než obyčejné počítání s reálnými čísly. Nejdůležitější je, abychom maticím přiřadili

označení, tím odpadá přepisování celých matic a můžeme pracovat pouze s jejich označením. Pokud matice násobíme pouhým číslem, je vše stejné jako kdybychom počítali s normálními čísly.

Pokud, ale matice násobíme mezi sebou, nastává jedna velká změna. Znak násobení není hvězdička, ale tečka. Hvězdička nám u matic způsobí takzvané složkové násobení. Roznásobí nám matice stejného tvaru, složku po složce $a_{11}*b_{11}$, $a_{12}*b_{12}$. Rozdíl ve výsledcích mez jednotlivými zápisy vidíme v následující ukázce.

```
(%i1) A: matrix([5,1],[2,4]);
(%o1) matrix([5,1],[2,4])

(%i2) B: matrix([2,2],[0,1]);
(%o2) matrix([2,2],[0,1])

(%i3) A.B ;
(%o3) matrix([10,11],[4,8])

(%i4) A*B ;
(%o4) matrix([10,2],[0,4])
```

Hodnost matice nám udává počet zbylých řádků v matici po Gaussově eliminaci. Ke zjištění této hodnosti nám slouží příkaz „rank“ a jako výsledek nám program vrátí číslici odpovídající hodnosti matice.

6.3 Determinant

Vypočítání determinantu matice je velice jednoduché. Můžeme například celý zápis matice dát do závorek a před závorky napsat příkaz pro vytvoření determinantu „determinant“.

Nebo můžeme využít horní lištu se záložkou Algebra, kde přímo najdeme příkaz Determinant. Ještě před použitím tohoto příkazu, ale musíme označit zápis matice, z které budeme determinant tvořit nebo kliknout na vytvořenou matici, jinak by se nám příkaz nevykonal.

```
(%i1) A: matrix(
        [13,5,8],
        [20,0,12],
        [-20,13,9]
      );
          [ 13   5   8 ]
          [             ]
(%o1)     [ 20   0   12 ]
          [             ]
          [-20  13   9 ]

(%i2) determinant(matrix([13,5,8],[20,0,12],[-20,13,9]));
(%o2) -2048
```

6.4 Transponovaná, Adjungovaná, a Inverzní matice

Matici, která vznikne vzájemným prohozením řádků a sloupců z původní matice A, označujeme jako transponovanou matici a značíme jí A^T. Pro jednotlivé prvky transponované matice platí: $a^T_{mn} = a_{nm}$

Matice adjungovaná k matici a vznikne transponováním matice algebraických doplňků matice A. Adjungovaná matice je důležitá pro výpočet inverzní matice. Označuje se jako A’ nebo adjA.

Inverzní matice A^{-1} k dané matici je taková matice, která po vynásobení s původní maticí a dá jednotkovou matici. Matice Inverzní vznikne, když adjungovanou matici vydělíme determinantem matice.

Všechny tři typy těchto matic (transponovaná, adjungovaná, inverzní) se v programu wxMaxima tvoří podobně. Pokud si označíme vytvořenou matici tím, že na ni klikneme, označení poznáme jejím zšednutím, poté můžeme v horní liště pod záložkou Algebra vybrat jednu z možností. Buďto Transponovat matici, Adjungovaná matice nebo Inverzní matice. Po vybrání jedné z metod se nám vygeneruje zvolená matice.

Ještě máme druhou možnost a to takovou, že každou matici vytvoříme napsáním příkazů ručně. Pro vytvoření transponované matice použijeme příkaz „transpose“, pro vytvoření adjungované matice použijeme příkaz „adjoint“ a pro vytvoření inverzní matice použijeme příkaz „invert“. Do závorek pak můžeme vypsat celou matici, označení matice nebo řádek, ve kterém jsme matici vytvořili.

Příklad 6.1.: Vypočti determinant matice C, která je rovna A^T+2*(B-A*adj(2*B)). Jestliže matice A = [(3;1;2), (2;0;4), (-1;3;1)], matice B = [(1;1;-1), (2;1;2), (3;2;1)].

Nejprve si do programu zapíšeme matice.

```
(%i1) A: matrix(
        [3,1,2],
        [2,0,4],
        [-1,3,1]
      );
```

$$(\%o1)\quad \begin{bmatrix} 3 & 1 & 2 \\ 2 & 0 & 4 \\ -1 & 3 & 1 \end{bmatrix}$$

```
(%i2) B: matrix(
        [1,1,-1],
        [2,1,2],
        [3,2,1]
      );
                [ 1  1  -1 ]
                [          ]
(%o2)           [ 2  1  2  ]
                [          ]
                [ 3  2  1  ]
```

Nyní, když už máme matice zadané, můžeme vypočítat matici C. Pro výpočet transponované matice, použijeme příkaz „transpose“, pro výpočet adjungované matice příkaz „adjoint“ a nesmíme zapomenout, že při násobení matic se zapisuje místo hvězdičky tečka.

```
(%i3) C: transpose(A)+2*(B-A.adjoint(2*B));
                [  29    28   -27 ]
                [                  ]
(%o3)           [  21    18   -9  ]
                [                  ]
                [ -120  -120  131 ]
```

Pro zjištění determinantu z matice C, použijeme příkaz „determinant“.

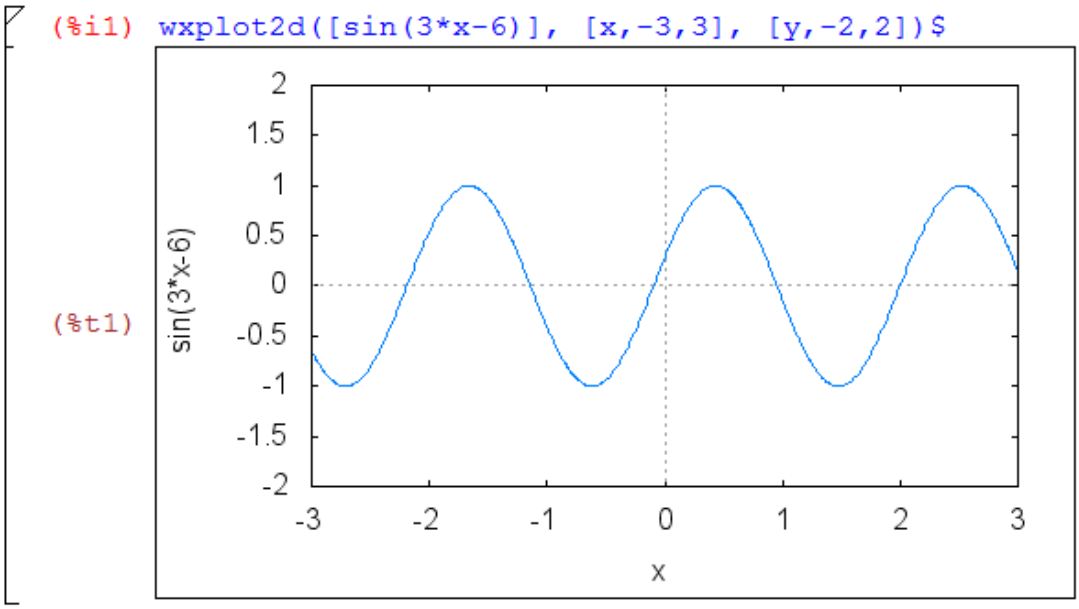
```
(%i4) determinant(C);
(%o4) -6
```

7 GRAF FUNKCE

Vypočítáním některých důležitých vlastností funkce si můžeme vytvořit představu, jak graf vypadá, ale zobrazení grafu je nenahraditelné. Program xwMaxima nabízí vytvoření dvojrozměrného a trojrozměrného grafu.

7.1 2D graf

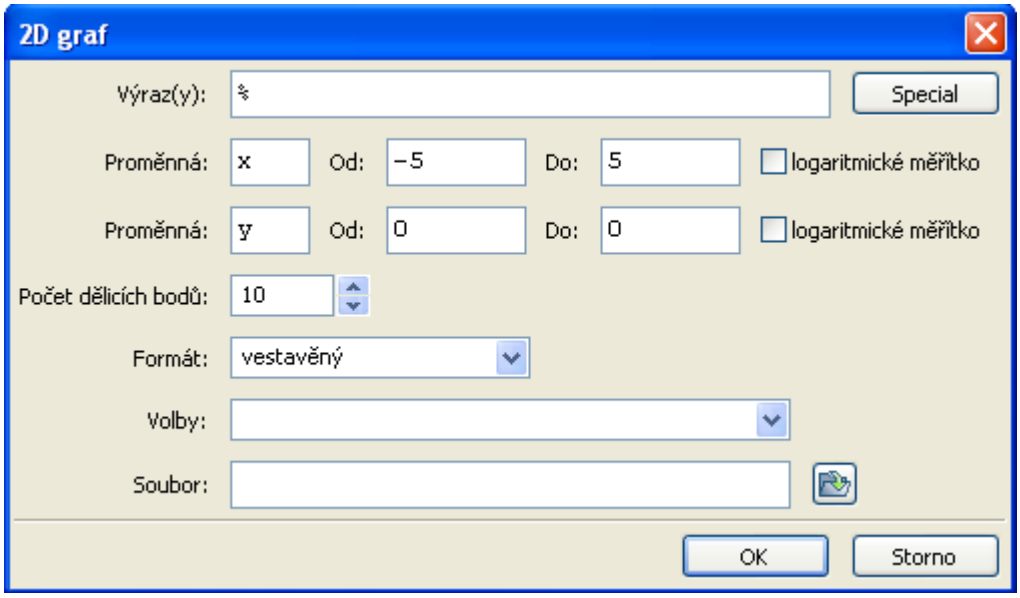
K vytvoření dvojrozměrného grafu funkce slouží příkaz „wxplot2d“. Skládá se ze tří základních částí, každá se píše do hranatých závorek oddělených čárkou. V první části zadáme funkci, ke které chceme vytvořit graf, do druhé a třetí části zapisujeme rozmezí souřadnic na ose x a na ose y. Pokud příkaz zapíšeme jako „plot2d“ vytvoří se nám graf v prostředí gnuplot graph.



Obrázek 9: Graf 2D

Na horní liště programu najdeme záložku Grafy, ve které je políčko 2D graf. Po kliknutí na tuto nabídku se zobrazí okno, ve kterém po nás program požaduje vyplnění políčka s funkcí. Dále zadáme rozsah na ose x a rozsah na ose y. Můžeme zde zadat počet dělících bodů, které jsou primárně nastaveny na 10. Formát grafu je nastaven na vestavěný, což znamená, že se graf vykreslí přímo do programu. Ostatní formáty

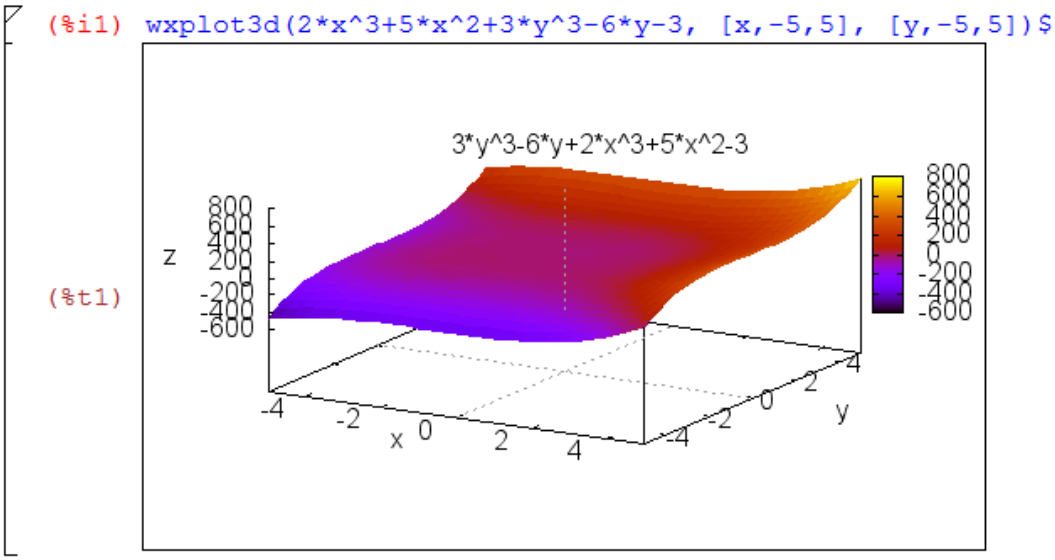
vykreslí graf do samostatného okna, ze kterého se graf může zkopírovat nebo samostatně uložit.



Obrázek 10: Tvorba 2D grafu

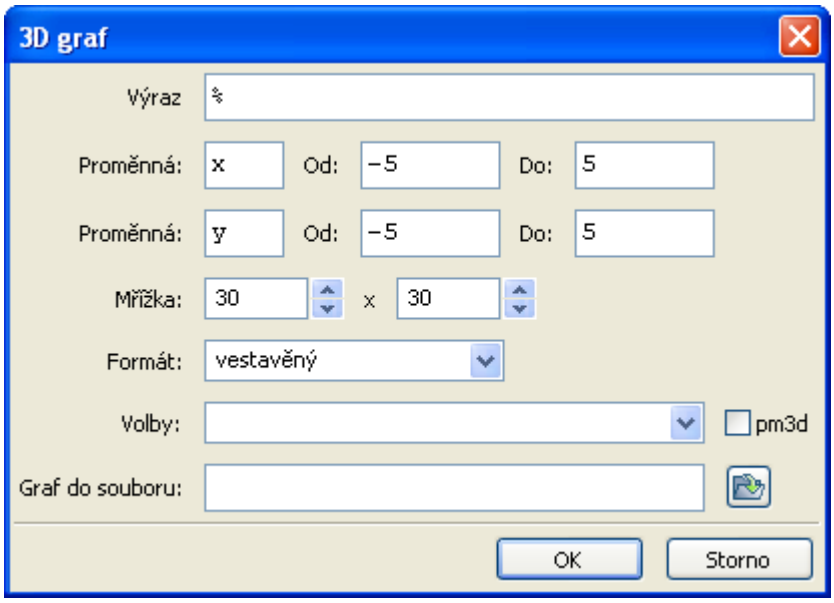
7.2 3D graf

K vytvoření trojrozměrného grafu funkce slouží příkaz „wxplot3d“. Skládá se stejně jako 2D graf, ze tří základních částí. Funkce a rozmezí dvou os, osy x a osy y.



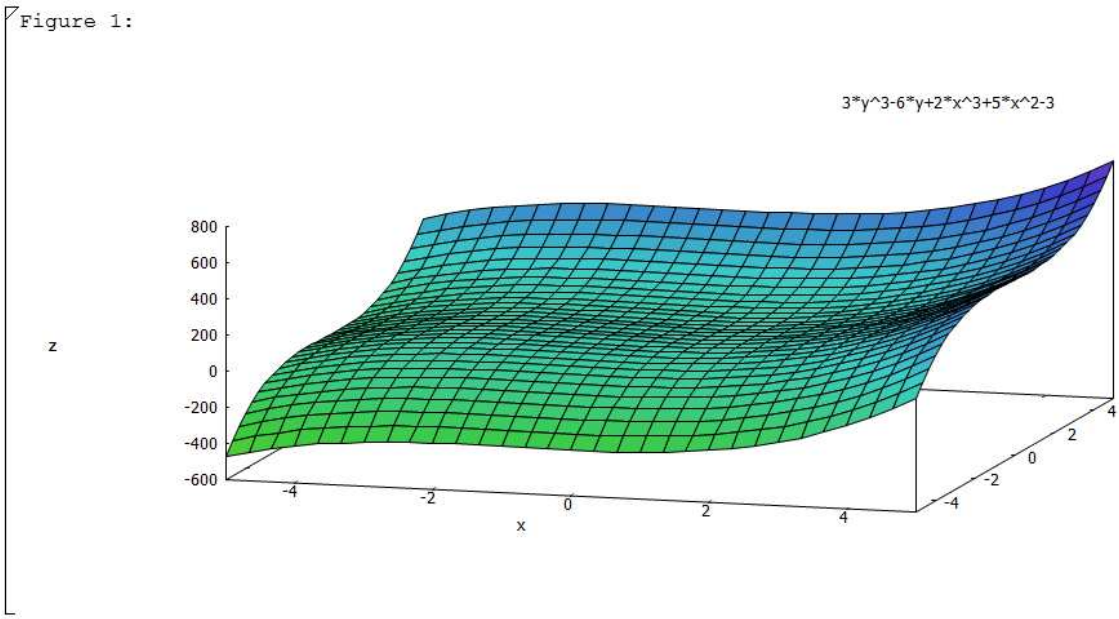
Obrázek 11: Graf 3D

V záložce Grafy, najdeme také políčko 3D graf, které nám pomůže graf vytvořit. Po kliknutí na tuto nabídku se zobrazí podobné okno, jaké už známe. Vyplníme políčko výraz, kam zadáme funkci a vyplníme políčka s proměnnými a jejich rozsahem.



Obrázek 12: Tvorba 3D grafu

Formát grafu je opět nastaven na vestavěný, potom vykreslený graf bude vypadat jako na předchozím obrázku. Pokud zvolíme formát gnuplot, otevře se nám graf v prostředí gnuplot graph. Pokud zvolíme formát openmath, otevře se nám graf v prostředí XMaxima. V obou těchto formátech se grafy dají otáčet a tím nám ulehčí prohlídnout si jejich zobrazení. Následně si tyto grafy můžeme zkopírovat do programu wxMaxima.



Obrázek 13: Zkopírovaný graf

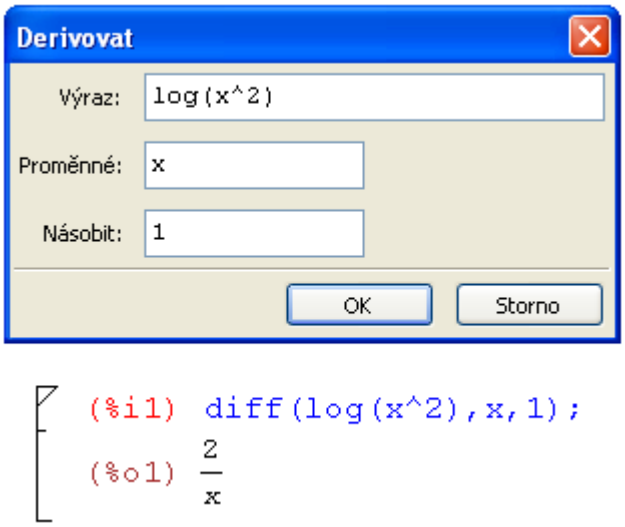
8 DERIVACE

Derivace funkce nám slouží k určení směrnice tečny v daném bodě. Pomocí derivací lze určit lokální extrémy funkce.

Pro výpočet derivace funkce používá program wxMaxima příkaz „diff“. V závorkách za příkazem se musí uvést tři podstatné údaje, které jsou odděleny čárkou. Jako první údaj zapíšeme funkci, z které má vniknout derivace. Můžeme ho zapsat přímo jako funkci nebo na tuto funkci odkázat pomocí čísla řádku nebo názvu funkce. Druhým údajem řekneme programu, podle kterého bodu máme funkci derivovat. Ve třetím údaji zadáme číslovku, která nám udává, kolikátou derivaci funkce chceme vypočítat.

```
(%i1) diff(x^3-sin(3*x),x,1);
(%o1) 3 x^2 -3 cos(3 x)
```

Pro uživatele snadnější a přehlednější výpočet derivace, můžeme získat, když využijeme řádek Derivace v záložce Analýza. Získáme tím přehlednou tabulku se třemi řádky. V prvním řádku po nás program požaduje zadání funkce, z které chceme vypočítat derivaci. Druhý řádek udává neznámou a třetí řádek počet derivací, které se mají vykonat.




```
(%i1) diff(log(x^2),x,1);
                              2
(%o1)                         -
                              x
```

Obrázek 14: Tvorba derivace

9 INTEGRÁL

Termínem "neurčitý integrál" funkce se rozumí množina jejích primitivních funkcí. Pod pojmem určitý integrál rozumíme obsah plochy ve dvojrozměrné rovině, který je omezen grafem funkce *f*, osou x a svislými přímkami $x = a$ a $x = b$.

Pro výpočet neurčitého i určitého integrálu použijeme stejný příkaz „integrate“. Liší se v počtu částí, které zápisy obsahují. Neurčitý integrál se skládá z funkce a proměnné. Určitý integrál má navíc ještě dvě části a jsou to krajní body intervalu. První se zapisuje bod s nižší číselnou hodnotou, poté bod s vyšší hodnotou.

Ve výsledku program nezapisuje konstantu.

```
(%i1) 'integrate((x-2)^2*(x^2+1),x);
```

$$(\%o1)\quad \int (x-2)^2 (x^2+1)\,dx$$

```
(%i2) integrate((x-2)^2*(x^2+1),x);
```

$$(\%o2)\quad \frac{3x^5-15x^4+25x^3-30x^2+60x}{15}$$

```
(%i3) expand(%o2);
```

$$(\%o3)\quad \frac{x^5}{5}-x^4+\frac{5x^3}{3}-2x^2+4x$$

```
(%i4) 'integrate(x*cos(x),x,0,%pi/2);
```

$$(\%o4)\quad \int_0^{\frac{\pi}{2}} x\cos(x)\,dx$$

```
(%i5) integrate(x*cos(x),x,0,%pi/2);
```

$$(\%o5)\quad \frac{\pi}{2}-1$$

10 LIMITA

Z definice limity funkce vyplívá, že pokud se hodnota zadané funkce blíží libovolně blízko k nějakému bodu, potom je právě tento bod označován jako limita.

Pro výpočet limity funkce slouží příkaz „limit“, který se skládá ze tří částí. V první části závorky se zapisuje funkce, do druhé části se zapíše proměnná a do třetí části zapíšeme hodnotu, ke které proměnná směřuje. Jednotlivé části jsou odděleny čárkou.

Jako výsledek limity nám program může vrátit číslo, ∞, -∞ nebo nápis „und“, který znamená, že limita neexistuje.

```
(%i1) limit((x^6+3*x^2+5)/(3*x-x^2),x,inf);
(%o1) -∞

(%i2) limit(log(x^2-8)/(x^2-3*x),x,3);
(%o2) 2

(%i3) limit(sqrt(1-cos(2*x))/x,x,0);
(%o3) und
```

11 PRŮBĚH FUNKCE

Vyšetření průběhu funkce si ukážeme na funkci $x^3/(x-1)$. Nejprve zjistíme **definiční obor** funkce, k tomuto zjištění použijeme příkaz „denom“ což je zkratka pro jmenovatele, který položíme rovno nule. Výsledek nám určí, které body nepatří do definičního oboru.

```
(%i1) f(x):=x^3/(x-1)^2;
                          3
(%o1)                    x
              f(x) := ------
                          2
                      (x-1)

(%i2) solve(denom(f(x))=0,x);
(%o2)                [x=1]
```

Dále zjistíme, v jakých bodech funkce protíná osu x a osu y. Pro zjištění průsečíku s osou y, dosadíme do funkce hodnotu x=0. Pro zjištění průsečíku s osou x, položíme funkci rovnou nule.

```
(%i3) ev(f(x),x=0);
(%o3)                0

(%i4) solve(f(x)=0,x);
(%o4)                [x=0]
```

Zjistíme limity v nekonečnu.

```
(%i5) limit(f(x),x,inf);
(%o5)                 inf

(%i6) limit(f(x),x,minf);
(%o6)                -inf
```

Vytvoříme si první derivaci funkce a položíme jí rovno nule. Vzniknou nám **Stacionární body**. Když si za x zvolíme libovolné číslo mezi těmito body a dosadíme

do první derivace, zjistíme, jestli je funkce v tomto intervalu rostoucí (+) nebo klesající (-).

```
(%i7) f1:factor(diff(f(x),x));
                              (x-3)x^2
(%o7)                         ---------
                                3
                              (x-1)
(%i8) solve(num(f1)=0,x);
(%o8) [x=0, x=3]
(%i9) ev(f1,x=-1);
                                1
(%o9)                           -
                                2
(%i10) ev(f1,x=2);
(%o10) -4
(%i11) ev(f1,x=5);
                               25
(%o11)                         --
                               32
```

Když do druhé derivace funkce dosadíme stacionární body, dostaneme **lokální extrémy**. Pokud vyjde kladné číslo, znamená to lokální minimum. Pokud vyjde záporné číslo, znamená to lokální maximum. Pokud vyjde nula, tak funkce v bodě nemá extrém.

```
(%i12) f2: factor(diff(f1,x));
                              6 x
(%o12)                      ------
                                 4
                            (x - 1)

(%i13) ev(f2,x:0);
(%o13)                         0

(%i14) ev(f2,x:3);
                               9
(%o14)                         -
                               8
```

Když druhou derivaci funkce položíme rovnou nule, pak dostaneme **inflexní body**. Za x si dosadíme číslo z intervalu mezi inflexními body a zjistíme kde je funkce konkávní (-) nebo konvexní (+).

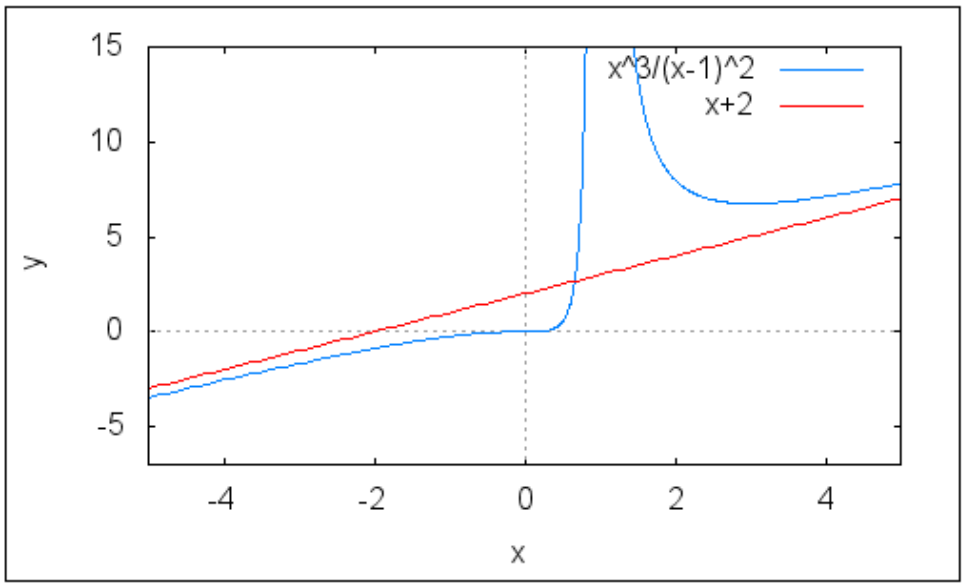
```
(%i15) solve(f2=0,x);
(%o15)                       [x = 0]

(%i16) ev(f2,x=-2);
                               4
(%o16)                       - --
                               27

(%i17) ev(f2,x=2);
(%o17)                         12
```

Poslední co budeme zjišťovat, jsou asymptoty. Asymptoty v nekonečnu jsou přímky $y=kx+q$. Kde k vypočteme pomocí limity $(f(x)/x)$, q vypočteme pomocí limity $(f(x)-kx)$.

```
(%i18) k1: limit(f(x)/x,x,inf);
(%o18) 1

(%i19) q1: limit(f(x)-k1*x,x,inf);
(%o19) 2

(%i20) y1=k1*x+q1;
(%o20) y1=x+2

(%i21) k2: limit(f(x)/x,x,minf);
(%o21) 1

(%i22) q2: limit(f(x)-k2*x,x,minf);
(%o22) 2

(%i23) y2=k2*x+q2;
(%o23) y2=x+2
```

Vytvoříme graf funkce v rozmezí, ve kterém se nacházejí všechny námi vypočítané důležité body.

```
(%i24) wxplot2d([f(x),x+2], [x,-5,5], [y,-7,15]);
plot2d: some values were clipped.
```




```
(%t24)

(%o24)
```

12 ZÁVĚR

Díky tvorbě této bakalářské práce, jsem se dokázal naučit pracovat s programem. Jelikož nebyl vytvořen žádný kompletní manuál pro práci s programem wxMaxima v českém jazyce, musel jsem se vše učit a poznávat od samých začátků. Díky tomu mám z práce velmi dobrý pocit. Mnoho mých výpočtů začínalo stylem pokus omyl, ale postupem času jsem se s programem naučil dobře pracovat. Myslím si, že se mi podařilo vytvořit vyvedenou práci, která je ideální pro nové uživatele a zájemce, ať už se jedná o lidi s předchozími zkušenostmi z podobných matematických prostředí nebo o úplné nováčky hledající jen matematickou pomoc v počítačovém světě.

Práce je určena pro širokou škálu studentů i učitelů, neboť obsahuje výpočty od úrovně základních škol, jako například hledání nejmenšího společného násobku dvou čísel, až po výpočty příkladů na úrovni středních škol, jako například hledání limity funkce v daném bodě. Všechny představované příkazy jsou doplněny o obrázky pro lepší představu a pochopení.

Přílohou bakalářské práce jsou také komentovaná videa, pro detailní postup výpočtů.

Literatura:

[1] BESHENOV, Lyosha. *Maxima, a Computer Algebra System*. [online]. [cit. 2013-04-10]. Dostupné z: http://maxima.sourceforge.net

[2] BUŠA, J., *MAXIMA Open source systém počítačovej algebry*, Fakulta elektrotechniky a informatiky, TU v Košicích, 2006

[3] ČERNÝ, Michal. *WxMaxima a Maxima: všestranný počtář à la Maple*. [online]. 2010-04-20 [cit. 2013-04-20]. Dostupné z: http://www.root.cz/clanky/wxmaxima-a-maxima-vsestranny-poctar-la-maple/

[4] Hašek, R., *Užití Derive ve výuce matematiky*, Europeon a.s., České Budějovice, 2007.

[5] URROZ, E. G., *Introduction to Maxima* [online], http://math.stanford.edu/~paquin/MaximaBook.pdf.

www stránky:

www.maxima.sourceforget.net

www.austromath.at/dug

www.andrejv.github.com/wxmaxima/index.html

www.abclinuxu.cz/clanky/programovani/hra-s-pismenky-wxmaxima

www.root.cz/clanky/wxmaxima-a-maxima-vsestranny-poctar-la-maple/

www.cs.wikipedia.org/wiki/Maxima

Použitý software:

Matematický program - wxMaxima 12.04.0

Obrázky zpracovány v Malování

Videa natočena v CamStudio