UNIVERZITA PALACKÉHO V OLOMOUCI
PEDAGOGICKÁ FAKULTA
Katedra matematiky

Diplomová práce

Bc. et. Bc. Libor Jeřábek

Řešení rovnic v systému počítačové algebry Maxima

Olomouc 2016

vedoucí práce: doc. RNDr. Tomáš Zdráhal, CSc.

Prohlášení

Prohlašuji, že jsem bakalářskou práci s názvem *Řešení rovnic v systému počítačové algebry Maxima* vypracoval samostatně, a že veškerá použitá literatura je uvedena v závěru práce.

V Olomouci dne: 22. dubna 2016

Bc. et. Bc. Libor Jeřábek

Poděkování

Rád bych na tomto místě poděkoval vedoucímu mojí diplomové práce panu doc. RNDr. Tomáši Zdráhalovi, CSc. za obětavou spolupráci i za čas, který mi věnoval při konzultacích.

Obsah

Úvod

Cílem této práce je seznámit čtenáře s počítačovým softwarem MAXIMA CAS. Práce obsahuje stručný manuál, pomocí kterého bude dotyčný schopen ovládat a zadávat potřebné příkazy k řešení různých úloh. Celkový koncept je rozčleněn do čtyř obsáhlých kapitol.

V první kapitole se seznámíme s možností stažení a instalace programu v prostředí operačního systému Microsoft a Android. Následně si představíme základní pracovní prostředí, se kterými se uživatel může setkat, a stručně si je popíšeme. Na tyto jednotlivé prostředí navážeme návodem jak zadávat jednotlivé příkazy a způsob zadávání.

V druhé kapitole si podrobně rozebereme příkazy spojené s řešením rovnic, nebo jejich soustav.

V předposlední kapitole se seznámíme s algebraickým řešením rovnic 3. a 4. řádu. Vysvětlíme si používání Hornerova schématu, rozklad na součin a Cardanovy vzorce.

Součástí poslední čtvrté kapitoly je výzkum a jeho vyhodnocení. Tento průzkum je zaměřený na algebraické řešení rovnic vyššího řádu (3. a 4.). Ve vyhodnocování jsem se pak zaměřil na celkový počet správných řešení jednotlivých příkladů, současně na úspěchy jednotlivých dotazovaných a jejich názor na matematické softwary.

1. Stažení, instalace a první seznámení s programem

1.1 Stažení a instalace

V první části práce se zaměříme na stažení a instalaci programu. Soubory potřebné k instalaci jsou volně stažitelné na webových stránkách *http://maxima.sourceforge.net/download.html*, kde jsou ke stažení verze dostupné pro různé typy operačních systémů - dále již jen jako OS(Microsoft, Linux, Android, …).

1.1.1 Instalace programu v operačním systému Microsoft

Na již výše zmiňovaných webových stránkách si zvolíme možnost stažení instalačních souborů pro OS Windows. Po zvolení požadovaného OS nás dané stránky přesměrují na https://sourceforge.net/projects/maxima/files/Maxima-Windows/, kde zvolíme poslední verzi programu 5.37.3. Po stažení souboru spustíme instalaci programu. Při instalaci budeme vybídnuti k potvrzení instalace. Kliknutím na tlačítko "*Další* >" se přesuneme k oknu s licenčními podmínkami. Po jejich odsouhlasení vybereme složku, ve které se nám vytvoří zástupce programu v nabídce Start a klikneme na tlačítko "*Instalovat*". Po ukončení instalace budeme vyzváni k dokončení instalace kliknutím na tlačítko "Dokončit".

1.1.2 Instalace programu v operačním systému Linux

Na webových stránkách *http://maxima.sourceforge.net/download.html* a vybereme možnost Android. Po kliknutí na Maxima on Android dojde k přesměrování na webové stránky https://sites.google.com/site/maximaonandroid. V odstavci označeném „*Download "Maxima on Android"*" opět vybereme možnost „*here*", kdy posléze naposled budeme přesměrováni na webové stránky https://play.google.com/store/apps/details?id=jp.yhonda&feature=search_result#?t=W251bGwsMSwxLDEsImpwLnlob25kYSJd. Zde již máme možnost stažení a instalování programu. Klinutím na tlačítko „Nainstalovat" budeme dotázáni k tzv. přihlášení. Jedná se o potvrzení typu zařízení, na který chceme program instalovat. Po potvrzení se začne aplikace sama stahovat a instalovat. Nezbytnou podmínkou pro úspěšné stažené a instalování je minimální paměť alespoň 1GB.

1.2 První seznámením s programem

Program můžeme spustit dvěma způsoby. My si nyní ukážeme zápis pomocí obou způsobů, kdy si jen letmo nastíníme zadávání příkladů pomocí jednotlivých příkazů a druhou variantu pomocí přehledné grafického prostředí.

1.2.1 Počítání pomocí konzolové aplikace XMaxima

Při spuštění programu danou aplikací se nám zobrazí prostředí, ve kterém zadáváme příkazy pomocí předem definovaných operací. Při tomto stylu zápisu je nezbytné dobře ovládat dané funkce a umět je správně zapsat. Pro ukázku uvedu pár příkladů pro rovnice, derivace a integrály.

Pro řešení rovnic využíváme příkazu „*solve*“. Vložíme kulatou závorku a hned za ni vložíme hranatou závorku, do které zapíšeme zadání naší rovnice a ukončíme hranatou závorkou. Vložíme čárku, do hranatých závorek uvedeme proměnnou, kterou chceme vypočítat, a ukončíme i kulatou závorku. Na závěr celý příkazový řádek ukončíme středníkem a dáme “*Enter*”.

Př. č. 1. Řešte rovnici:

```
solve([x+1=-5],[x]);
(%o11)                    [x = - 6]
```

Př. č. 2.: Řešte rovnici:

```
solve([x^2+x-2],[x]);
(%o12)                [x = 1, x = - 2]
```

Př. č. 3.: Řešte rovnici:

```
solve([x^3+x^2+x-3],[x]);
(%o13)      [x = (- sqrt(2) %i) - 1, x = sqrt(2) %i - 1, x = 1]
```

Příkazem “*diff*” dáme příkaz k derivování. Zápis se od zadávání rovnic značně liší. Za příkaz diff vložíme kulatou závorku a hned za ni píšeme funkci, kterou chceme derivovat. Tuto funkci oddělíme čárkou a zapíšeme za ni proměnnou, podle které derivujeme. Vložíme čárku a hned za ni uvedeme tentokrát číslo (z množiny přirozených čísel), kterým určíme počet provedených derivací. Ukončíme závorku, vložíme středník a dáme “*Enter*”.

Př. č. 4.: Derivujte funkci

```
diff(x^4+x^2,x,1):
                                   3
(%o25)                        4 x  + 2 x
```

Př. č. 5.: Derivujte funkci:

```
diff(x^3+x^2-x,x,2);
(%o26)                        6 x + 2
```

Př. č. 6.: Derivujte funkci:

```
diff(x^3+y^3+x*y^2+x-y,y,1);
                                 2
(%o34)                    3 y  + 2 x y - 1
```

V případě integrálů zvolíme příkaz "*integrate*". Zápis se provádí obdobně jako u derivace. Za příkaz integrate vložíme kulatou závorku a hned za ni píšeme funkci, kterou chceme integrovat. Tuto funkci oddělíme čárkou a zapíšeme za ni proměnnou, podle které integrujeme. Ukončíme závorku, vložíme středník a dáme "*Enter"*.

Př. č. 7.: Integruj výraz:

```
integrate (x+1, x);
                              2
                             x
(%o27)                       -- + x
                             2
```

Př. č. 8.: Integruj výraz:

```
integrate (x^2+x-1, x);
                          3       2
                          x       x
(%o8)                     -- + -- - x
                          3       2
```

Př. č. 9.: Integruj výraz:

```
integrate (x+y, y);integrate (x+y, y);
                          2
                          y
(%o33)                    -- + x y
                          2
```

Jak jsme si ukázali na těchto 9 příkladech, daný program můžeme využít k vypočítání jakéhokoliv příkladu. Obtížnost je ale v jeho zadávání a je nutná znalost příkazů pro jednotlivé výpočty. Z tohoto důvodu nebudeme daný styl dále více rozebírat, ale zaměříme se na druhý pro uživatele mnohem jednodušší a přehlednější styl zadávání.

1.2.2 Počítání pomocí uživatelského grafického prostředí wxMaxima

V následujícím textu se zaměříme podrobně na přehledné uživatelské prostředí wxMaxima. hned při spuštění je zřejmé že dané prostředí je pro zacházení značně jednodušší. Máme zde již Menu, které obsahuje základní prvky (Soubor, Editovat, View, Cell, Maxima, Rovnice, Algebra, Analýza, Zjednodušit, Grafy, Numerické výpočty, Nápověda).

1.2.2.1 Soubor

Složka Soubor, obsahuje základní prvky jako je New, tedy nová pracovní plocha, Uložit/Uložit jako, kdy si můžeme uložit naše výpočty k pozdějšímu zpracování. Je zde i možnost otevřít libovolný uložený soubor, načíst naposledy použité data, ale i tisk a samozřejmě konec.

1.2.2.2 Editovat

Záložku Editovat, můžeme přirovnat k záložce Upravit, která má obdobné vlastnosti jako v jiných programech. Jsou zde funkce jako je Zpět, Vyjmout, Vložit, Najdi apod.

1.2.2.3 View

View slouží jako záložka s možností volby početních operací, Statistika, Symboly, … Jednotlivé panely nám usnadňují a urychlují práci např. zadávání symbolů, řecké abecedy atd. Další funkcí, která nám umožňuje zvětšit, či zmenšit pracovní plochu je Zoom.

1.2.2.4 Cell

Cell využijeme při vkládání textu do výpočtů. Můžeme vkládat jak obyčejný text, tak i nadpisy, podnadpisy, číslování. Umožní nám vkládat i obrázky. Proto tuto záložku můžeme označit jako textový a grafický editor.

1.2.2.5 Maxima

Záložku Maxima můžeme označit jako funkční záložku. Pomocí ní můžeme resetovat program, změnit výstupy, současně využít jako generátor výstupu pro TeX apod. Už z tohoto výčtu funkcí je zřejmé, že danou záložku budeme využívat jen minimálně.

1.2.2.6 Rovnice

Z názvu je patrné, že danou záložku budeme využívat k řešení rovnic, hledání kořenů polynomu, určení počátečních podmínek, hodnot v bodě apod.

1.2.2.7 Algebra

V sekci algebra máme možnost sestavit matice, inverzní matice, počítání s maticemi, determinant atd.

1.2.2.8 Analýza

Záložka analýza nám umožňuje využití počítání derivací, určení limit, výpočet sumy, součinu, integrálu, nejmenšího společného násobku, největšího společného dělitele a dalších analytických výpočtů.

1.2.2.9 Zjednodušit

Zjednodušit nám pomáhá upravit složité výrazy na jednodušší. A to klasické nebo i výrazy s odmocninami. Dále zde najdeme i funkce pro počítání s logaritmy, výpočet modulo aj.

1.2.2.10 Grafy

V záložce graf najdeme možnost sestrojení grafů v provedení 2D a 3D.

1.2.2.11 Numerické výpočty

1.2.2.12 Nápověda

Již podle názvu je patrné, že v záložce Nápověda se nachází nápověda k programu, současně je zde i možnost si najít si styl a způsob zadávání jednotlivých příkazů.

1.3 Zadávání příkazů do uživatelského rozhraní wxMaxima

1.3.1 Příkazový řádek

Na úvod je potřeba se seznámit se základními pravidly pro zadávání jednotlivých pravidel. Každý námi zadaný výraz musí být ukončen středníkem. Vyhodnocení zadaného příkazu proběhne po stisknutí kláves “Shift+Enter”. Do jednoho řádku lze zapsat i více příkazů, které musejí být odděleny středníkem. Současně můžeme vyhodnotit i více řádků, které můžeme zapsat pomocí klávesy “Enter”. Pokud je uživatel zvyklý na opačný styl zadávání pomocí kláves Enter a Shift+Enter, lze jejich funkčnost vyměnit. V záložce *Editovat* vybereme položku *Možnosti* a v ní zaškrtneme možnost “ Enter vyhodnocuje výraz”. V ukázce si předvedeme vyhodnocení dvou přirozených čísel, zapsaných pomocí jednořádkového a víceřádkového příkazu.

```
(%i1) 1;5;
(%o1)                          1
(%o2)                          5
```

```
(%i1) 1;
5;
(%o1)                    1
(%i2)
(%o2)                    5
```

Všechny zadávané vstupy jsou označeny (%ix)(z anglického slova input), výstup nese označení (%ox)(z anglického slova output) a x je číselné označení zadávaného příkazu v pořadí. Pomocí těchto výrazů se můžeme v následujících příkladech odkazovat na získané výsledky nebo postupy.

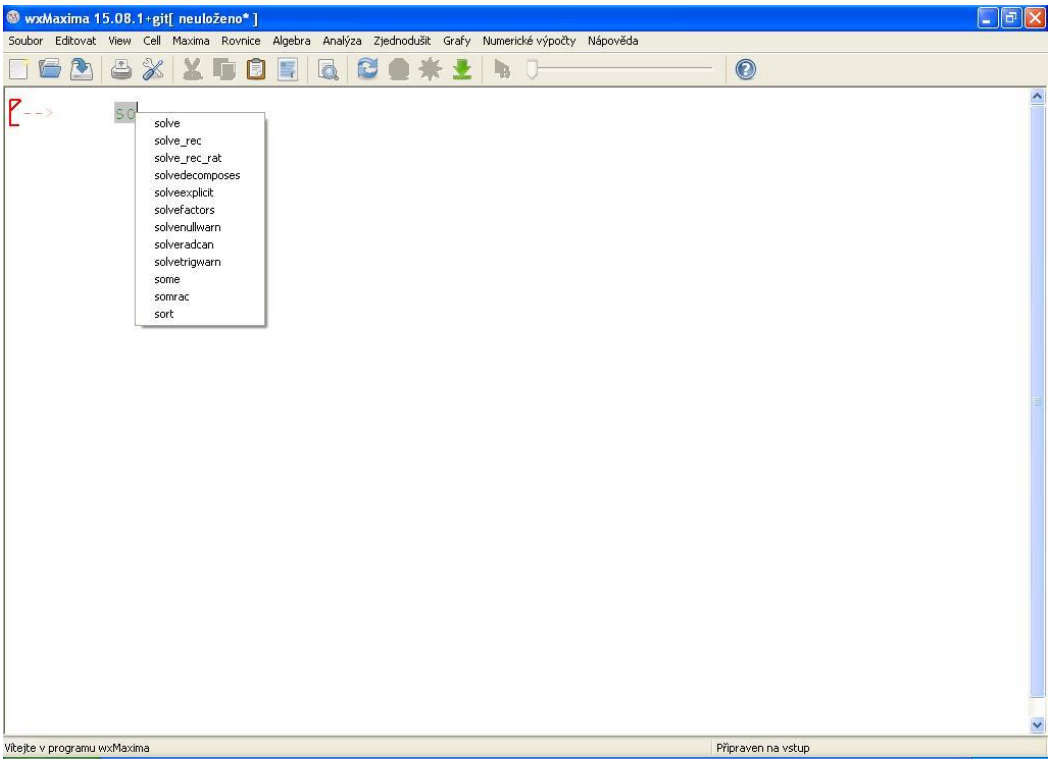
Pro skrytí zobrazených vyhodnocených výrazů použijeme znak $ (dolaru), který zapíšeme na konci příkazu místo středníku. I přesto, že se nám výsledek nezobrazí, můžeme s ním i nadále pracovat. Výsledný výraz můžeme opět vyvolat pomocí příkazu (%ox). Vyhodnocení bez zobrazení a jeho opětovné vyvolání si názorně ukážeme v následující ukázce.

```
(%i1) 1+3$

(%i2) 6+%o1;
(%o2)                    1
```

Pokud chceme zadaný výraz nepočítat, jen ho pouze vypsat do výstupu, vložíme před zadání apostrof ‘. V příkladu použije příkaz *solve()*, pomocí které vypočítáme kořeny rovnice. O jejím podrobnějším zadávání se píše v následující kapitole.

```
(%i1) 'solve([x-1=2],[x]);
(%o8)                    solve([x - 1 = 2], [x])

(%i9) solve([x-1=2],[x]);
(%o9)                    [x = 3]
```

V případě, že zadavatel si nemůže přesně vybavit daný příkaz, může pomocí klávesnic “Ctrl+K” vyvolat okno se seznamem podobně znějících příkazů např. pro funkci solve



Obr. 1.: Pomocné zobrazovací okno

1.3.2 Ukládání, otevírání, export a tisk souborů

Kdykoliv během práce, lze svoji rozpracovanou práci uložit a posléze načíst. Možnost *Uložit, Uložit jako, Otevřít* a *Otevřít naposledy použité* nalezneme v záložce *Soubor*. Další možností, jak uložit soubor, je pomocí klávesnicové zkratky *Ctrl+C*. Pro otevření slouží klávesnice *Ctrl+O*. Soubory jsou uloženy ve formátu s příponou .wxm. Současně program umožňuje možnost *Export*, která nám exportuje naše vypočítané příkazy do jazyka HTML nebo jazyka TEX. Tisk se provádí rovněž přes záložku *Soubor* a možnost *Tisk...* nebo přes klávesovou zkratku *Ctrl+P*.

1.3.3 Základní početní funkce

Nyní se zaměříme na zadávání jednotlivých příkazů. Pro přehlednost a lepší názornost uvedeme vždy i vzorový příklad.

1.3.3.1 Standardní operace

Při obyčejném matematickém počítání využíváme standardní znaky +,-,* a /. Mocniny můžeme zadávat pomocí stříšky ^. Ekvivalentním zápisem je znak **. Odmocniny můžeme uvést jako mocniny, eventuálně druhou odmocninu můžeme zapsat pomocí výrazu sqrt(), kde argumentem je číslo, které chceme odmocnit. U zadávání musí uživatel dbát pozornosti při použití závorek, protože tím dochází ke změně priorit při počítání. Nyní si ukažme zadávání jednotlivých příkazů a jejich možnosti.

```
(%i1) 4+10;
(%o1)                    14

(%i2) 9-5;
(%o2)                    4

(%i3) 3*7;
(%o3)                    21

(%i4) 45/9;
(%o4)                    5

(%i5) 3^3;
(%o5)                    27

(%i6) 3**2;
(%o6)                    9

(%i1) 9^(1/2);
(%o1)                    3

(%i5) sqrt(16);
(%o5)                    4

(%i1) 16^1/2;
(%o1)                    8
```

```
(%i2) 16^(1/2);
(%o2)                    4

(%i3) 2^2+1;
(%o3)                    5
%i4) 2^(2+1);
(%o4)                    8
```

Při násobení čísla a proměnné je nutné vložit vždy mezi ně znak pro krát *. Jinak nám toto zadání vyhodnotí program jako error.

```
(%i1)  2x;
(%o1)  incorrect syntax: x is not an infix operator
       2x;
         ^
(%i2)  2*x;
(%o2)  2*x
```

1.3.3.2 Matematické konstanty

Pomocí programu maxima můžeme využívat tyto matematické konstanty: Eulerovo číslo, imaginární jednotka, Ludolfovo číslo, záporné nekonečno, kladné nekonečno, logická pravda a nepravda. Jejich zadávání je zcela jednoduché až triviální, proto věřím tomu, že postačí pouze názorná ukázka zadávání jednotlivých konstant.

```
(%i1) %e;
      %i;
%pi;
minf;
inf;
true;
false;
(%o3) e
(%o4) i
(%o5)
(%o6)
```

(%o7)

(%o8) *true*

(%o9) *false*

1.3.3.3 Přiřazování hodnot, výrazů a funkcí

U výrazů a hodnot, které mohou obsahovat nějakou proměnnou, provádíme přiřazení pomocí dvojtečky *":"*. V případě vytvoření funkce jedné a více proměnných využije kombinace dvojtečky a rovná se *":="*. Do závorek za označení funkce zapisujeme názvy proměnných oddělené čárkou. Ukažme si na příkladu, jak vytvoříme výraz *f* a funkci *g(x).*

```
(%i1) f:x^2+x-4;
                          2
(%o1)                     x  + x - 4

(%i2) g(x):=x^4+x^3-x^2+3;
                          4       3       2
(%o2)                     g(x) := x  + x  - x  + 3
```

Chceme-li získat hodnotu v určitém bodě námi zadané funkci *g(x), dosadíme* za x požadovanou hodnotu.

```
(%i3) g(2);
(%o3)                          23
```

Pokud bychom chtěli vypočítat hodnotu pro výraz např. *f* (1), zobrazilo by nám vyhodnocení chybu, protože výraz *f* není funkce, a tedy nezadáváme žádný argument. Pokud chceme smazat přiřazené hodnoty, využijeme příkaz *kill(),* kdy argument považujeme název proměnné. V případě, že chceme smazat všechny argumenty, použijeme příkaz *kill(all).*

1.3.3.4 Zobrazení reálných čísel

Pro naše účely budeme využívat záložky *Numerické výpočty*. V dalších krocích budeme pracovat s možnostmi této záložky. Pokud chceme vyčíslit nějaký numerický symbol např. π, e, …, využijeme možnosti *Přepnout numerický výstup*. V případě, že chceme převést pouze daný určitý výraz, využijeme *Převést na float (plovoucí řádková čárka).*

(%i1) %pi;
(%o1)
(%i2) %e;
(%o2) e

(%i3) if numer#false then numer:false else numer:true;
(%o3) true

(%i4) %pi;
(%o4) 3.141592653589793

(%i5) %e;
(%o5) 2.718281828459045

(%i1) %e;
(%o1) e

(%i2) float(%), numer;
(%o2) 2.718281828459045

Potřebujeme-li větší počet zobrazených číslic než je základní šestnácti místní počet, můžeme pomocí *Set bigfloat Precision… změnit* počet číslic., nebo zadáním příkazu *fpprec:x;*, kde x nám určuje počet míst kolik chceme zobrazit. Pokud chceme vypsat upravený počet výrazů, musíme použit *Přesnost bigfloat.*

(%i1) %pi;
(%o1) 3.141592653589793

(%i2) float(%), numer;
(%o2) 2.718281828459045

(%i3) *fpprec: 20;*
(%o3) 20

```
(%i4)  bfloat(%pi), numer;
(%o4)  3.141592653589793116b0
```

Na konci výstupu (%o4) si můžeme všimnout výrazu b0, kde *b* nám odděluje desetinná místa od výrazu 10^0

1.3.3.5 Komplexní čísla

Komplexní čísla zapisujeme podle obecného matematického zápisu *a+bi*, kde a *i* je imaginární jednotka. Zapisujeme ji pouze ve tvaru *i*.

```
(%i1)  r1:1+i;
r2:4+3*i;
(%o1)  i + 1

(%i2)
(%o2)  3 i + 4
```

S komplexními čísly můžeme počítat jako s běžnými čísly.

```
(%i5)  r1+r2;
(%o5)  4 i + 5

(%i6)  r1-r2;
(%o6)  (- 2 i) - 3

(%i7)  r1*r2;
(%o7)  (i + 1) (3 i + 4)

(%i8)  r1/r2;
(%o8)  (i+1)/(3*i+4)
```

Chceme-li získat základní tvar, využijeme funkce *rectform()*. Pokud potřebujeme vypsat z výrazu pouze reálnou složku, volíme funkci realpart(). Pro imaginární složku čísla použijeme příkaz imagpart().

```
(%i9)  rectform(%o8);
(%o9)  %i/25+7/25

(%i10)         realpart(%o9);
(%o10)         7/25

(%i11)         imagpart(%o6);
(%o11)         1/25
```

1.3.4 Počítání s polynomy

V této kapitole si popíšeme práci s polynomy. Ukážeme si jejich zápisy, základní operace a funkce pro práci s nimi. Např. výpočet hodnoty polynomu v daném bodě, dělitelnost, derivaci a další.

1.3.4.1 Zápis polynomu

Polynomy můžeme zadávat pomocí dvou možností. První je popsán v kapitole 1.3.3.3 pomocí *":="* , kde jsme ho definovali jako funkci. Jednotlivé polynomy budeme zapisovat v kanonickém tvaru, členy sestupně v řadě od nejvyšší mocniny po nejmenší. Polynom budeme tedy zapisovat ve tvaru:

```
(%i1)  f(x):=3*x^3+4*x^2-7*x+1;
(%o1)  f(x):=3*x^3+4*x^2-7*x+1
```

Ekvivalentem tohoto zápisu je využití příkazu *define().* Vstupním argumentem pro příkaz je název polynomu s neurčitým výrazem a daný výraz funkce.

```
(%i2)  define(g(x),2*x^3+5*x^2-3*x+1);
(%o2)  g(x):=2*x^3+5*x^2-3*x+1
```

Chceme-li zjistit stupeň polynomu (kromě funkce), použijeme příkaz hipow(). Do argumentu příkazu zadáváme dvě hodnoty. První je název funkce a druhá je neurčitá proměnná.

```
(%i3)  hipow(f(x),x);
(%o3)  3
```

Při práci s polynomy může nastat situace, že budeme potřebovat zjistit hodnotu koeficientu u některého členu polynomu. K tomu nám slouží funkce *coeff()*. I zde je první vstupní složkou argumentu název funkce a druhou složkou je výraz, jehož hodnotu koeficientu chceme zjistit.

```
(%i4)  coeff(f(x),x^2);
(%o4)  4
```

Je patrné, že tato funkce vypíše jen hodnotu koeficientu u daného členu, ale nikoli již ostatní členy. Pokud chceme vypsat i ostatní členy, použijeme příkaz bothcoeff(), který nám na prvním místě vypíše hodnotu koeficientu u daného členu a posléze zbývající členy.

```
(%i5)  bothcoeff(f(x),x);
(%o5)  [-7,3*x^3+4*x^2+1]
```

Zajímá-li nás hodnota polynomu v daném bodě, stačí zadat pouze příkaz *f(c)*. Další možností je použití funkce subst(), kde v argumentu se na prvním místě nachází hodnota, pro kterou výraz počítáme, na druhém místě je výraz, místo kterého zavádíme substituci a jako poslední je název funkce.

```
(%i6)  f(c);
(%o6)  3*c^3+4*c^2-7*c+1

(%i7)  subst(z,x,f(x));
(%o7)  3*z^3+4*z^2-7*z+1
```

1.3.4.2 Základní operace s polynomy

Základní početní operace s polynomy provádíme podobně, jako je uvedeno v kapitole 1.3.3.1. Místo číselných hodnot použijeme názvy polynomů.

```
(%i1)  f(x):=4*x^4+3*x^3-2*x^2+8*x-1;
(%o1)  f(x):=4*x^4+3*x^3-2*x^2+8*x-1

(%i2)  g(x):=3*x^4-2*x^3+9*x^2-5*x+6;
(%o2)  g(x):=3*x^4-2*x^3+9*x^2-5*x+6
```

```
(%i3)  f(x)+g(x);
(%o3)  7*x^4+x^3+7*x^2+3*x+5

(%i4)  f(x)-g(x);
(%o4)  x^4+5*x^3-11*x^2+13*x-7

(%i5)  f(x)*g(x);
(%o5)  (3*x^4-2*x^3+9*x^2-5*x+6)*(4*x^4+3*x^3-2*x^2+8*x-1)

(%i6)  f(x)/g(x);
(%o6)  (4*x^4+3*x^3-2*x^2+8*x-1)/(3*x^4-2*x^3+9*x^2-5*x+6)

(%i7)  f(x)**2;
(%o7)  (4*x^4+3*x^3-2*x^2+8*x-1)^2
```

V případě, že chceme zjistit, zda jsou dva polynomy stejné, použijeme příkaz *is().* Do argumentu vložíme název funkcí, které chceme porovnat a vložíme mezi ně znak rovná se =. Vyhodnocení je ve formě t*rue* (pravda) nebo *false* (nepravda).

```
(%i8)  is(f(x)=g(x));
(%o8)  false
```

Pomocí programu Maxima můžeme provádět rozpis do kanonického tvaru funkcí *expand().* Chceme-li rozložit funkci na součin, využijeme příkazu *factor(),* který slouží pro rozklad s reálnými čísly nebo *gfactor(),* jenž využijeme pro rozklad s komplexními čísly.

```
(%i8)  expand(%o7);
(%o8)  16*x^8+24*x^7-7*x^6+52*x^5+44*x^4-38*x^3+68*x^2-16*x+1

(%i9)  factor(%o8);
(%o9)  (4*x^4+3*x^3-2*x^2+8*x-1)^2
```

```
(%i10)      z(x):=x^2+1;
(%o10)      z(x):=x^2+1

(%i10)      factor(z(x));
(%o11)      x^2+1

(%i10)      gfactor(z(x));
(%o11)      (x-%i)*(x+%i)
```

1.3.4.3 Dělení polynomu

Nyní se zaměříme na dělení polynomů. Můžeme je využit pro další výpočty, ale i pro výpočet rovnic pomocí tzv. Hornerova schématu, které je podrobněji rozepsáno v kapitole č. 3.1. Dělení polynomu mnohočlenem (polynome) provádíme pomocí funkce *divide()*, kde v argumentu na prvním místě uvedeme název funkce, a na druhém místě oddělené čárkou uvedeme výraz, kterým chceme daný polynom dělit.

```
(%i12) divide(f(x),(x+1));
(%o12)      [4*x^3-x^2-x+9,-10]
```

S výsledky můžeme nadále počítat jako s částečným podílem a zbytkem. Výsledek po dělení nám vytvoří seznam, a proto při dalším zpracování se budeme na jednotlivé segmenty výsledku odvolávat pomocí hranatých závorek. Do těchto závorek uvádíme n-tý člen ze seznamu.

```
(%i13) %o12[2]-%o12[1];
(%o13)      -4*x^3+x^2+x-19
```

Jak již bylo zmíněno, pomocí programu Maxima můžeme spočítat kořeny polynomu s využitím Hornerova schématu. K tomuto účelu slouží příkaz horner (). Do argumentu opět uvádíme samotnou funkci, kterou chceme vypočítat, nebo její název a neznámou proměnnou. V případě, že ve funkci je zadaná pouze jedna neznámá, nemusíme tuto proměnnou udávat.

```
(%i14) horner(f(x),x);
(%o14)          x*(x*(x*(4*x+3)-2)+8)-1

(%i15) horner(f(x));
(%o15)          x*(x*(x*(4*x+3)-2)+8)-1
```

1.3.4.4 Derivace polynomu

Derivace je jedna z nezbytných matematických operací. K tomuto účelu použijeme výraz *diff().* Do argumentu uvedeme funkci, kterou chceme derivovat, nebo její název a uvedeme proměnnou, podle které budeme derivovat.

```
(%i16) diff(f(x),x);
(%o16)          16*x^3+9*x^2-4*x+8
```

V případě, že chceme získat derivaci vyššího řádu, zapíšeme za proměnnou, podle níž derivujeme, číslo derivace.

```
(%i17) diff(f(x),x,3);
(%o17)          96*x+18
```

1.3.4.5 Výpočet kořenů polynomu

V předchozí kapitole jsme si ukázali již jeden způsob výpočtu s použitím Hornerova schématu. Pro připomenutí zopakuji příkaz pro zadávání. Jedná se o výraz *horner().* Základním výrazem pro výpočet kořenů polynomu je *solve().* Zadávání do argumentu je stejné jako při zadávání do příkazu pro Hornerovo schéma. Na prvním místě uvedeme funkci nebo její název, kterou chceme vypočítat, a posléze výslednou proměnnou. Je-li v zadávané funkci pouze jedna proměnná, nemusíme ji do příkazu zadávat.

```
(%i18) solve(2*x^3*x+2*x*y^2+3*x*y+2,[y]);
(%o18)          [y=-(sqrt(-16*x^5+9*x^2-16*x)+3*x)/(4*x),y=(sqrt(-16*x^5+9*x^2-
16*x)-3*x)/(4*x)]
```

```
(%i19) h(x):=x^3-6*x^2+11*x-6;
(%o19)          h(x):=x^3-6*x^2+11*x-6

(%i20) solve(h(x));
(%o20)          [x=1,x=2,x=3]
```

Jestliže nám výsledek vychází na množině komplexních čísel a my požadujeme pouze výsledek na množině reálných čísel, použijeme výraz *algsys().* Do argumentu vkládáme nejdříve funkci nebo její název a posléze neurčitou, kterou chceme vyřešit. Při zadávání je nezbytné zadat funkci a proměnnou do hranatých závorek. Za takto zadaný výraz připojíme *realonly:true.*

```
(%i21) z(x):=x^3+x^2+x-3
(%o21)          z(x):=x^3+x^2+x-3

(%i22) algsys([z(x)],[x]);
(%o22)          [[x=sqrt(2)*%i-1],[x=-sqrt(2)*%i-1],[x=1]]

(%i22) algsys([z(x)],[x]),realonly:true;
(%o22)          [[x=1]]
```

1.3.5 Grafy

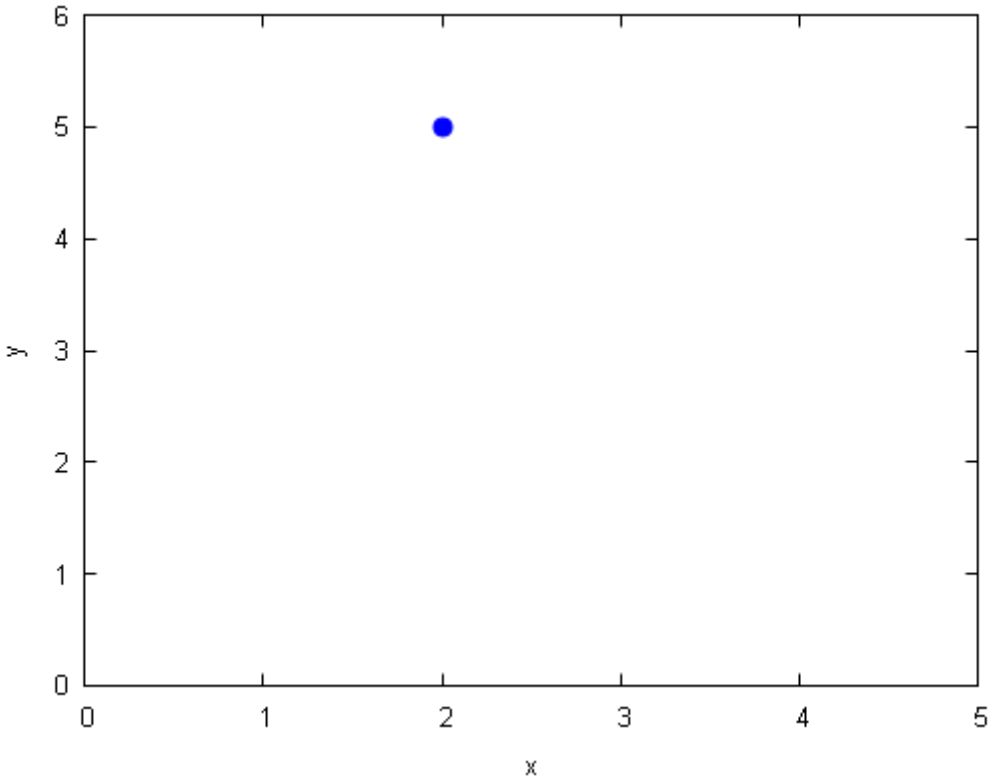
Mezi základní vybavení programu patří i vykreslování grafů 2D i 3D, tedy buď v rovině, nebo v prostoru. Proto se nyní zaměříme na záložku Grafy, která je v hlavním Menu.

1.3.5.1 Zadávání bodu

Jedná se o grafy vykreslené v rovině, tedy na ose *x* a *y.* Mezi základní prvky pro vykreslení patří bod. Při jeho zadávání je nutné nejdříve zadat jeho souřadnice pomocí příkazu *lx:* pro souřadnice na ose *x a ly:* pro souřadnice na ose *y.* Dále zadáme příkaz pro vykreslení grafu *wxplot2d(),* kdy do jeho argumentu uvedeme potřebné informace pro vykreslení bodu. Na prvním místě dáme příkaz pro vytvoření grafické pracovní plochy příkazem *makelist().* Zde do argumentu vložíme v hranatých závorkách funkci *(discrete,lx,ly)*,1, kde discrete nám označuje typ zadávání, lx a ly nás odkazují na souřadnice

bodu a číslovka 1 za závorkou nám udává barvu vyznačeného bodu (v našem případě se jedná o modrou barvu). Po takto nedefinovaném bodu musíme nadefinovat i rozměry zobrazovacího okna, které uvedeme v hranatých závorkách. Na prvním místě v závorce vždy uvedeme o, jakou osu se jedná (*x*,*y*). Na druhém místě uvádíme nejmenší hodnotu zobrazovacího pole a na třetím místě největší hodnotu). Po udání rozměrů pracovní plochy musíme určit, co chceme zobrazit. Opět do hranatých závorek uvedeme funkci *style*, za ni nadefinujeme, jestli chceme vykreslit bod (points), přímku (lines) apod. Číslice za tímto výrazem nám udává velikost bodu. Celý takto na definovaný bod ukončíme znakem dolaru *$*.

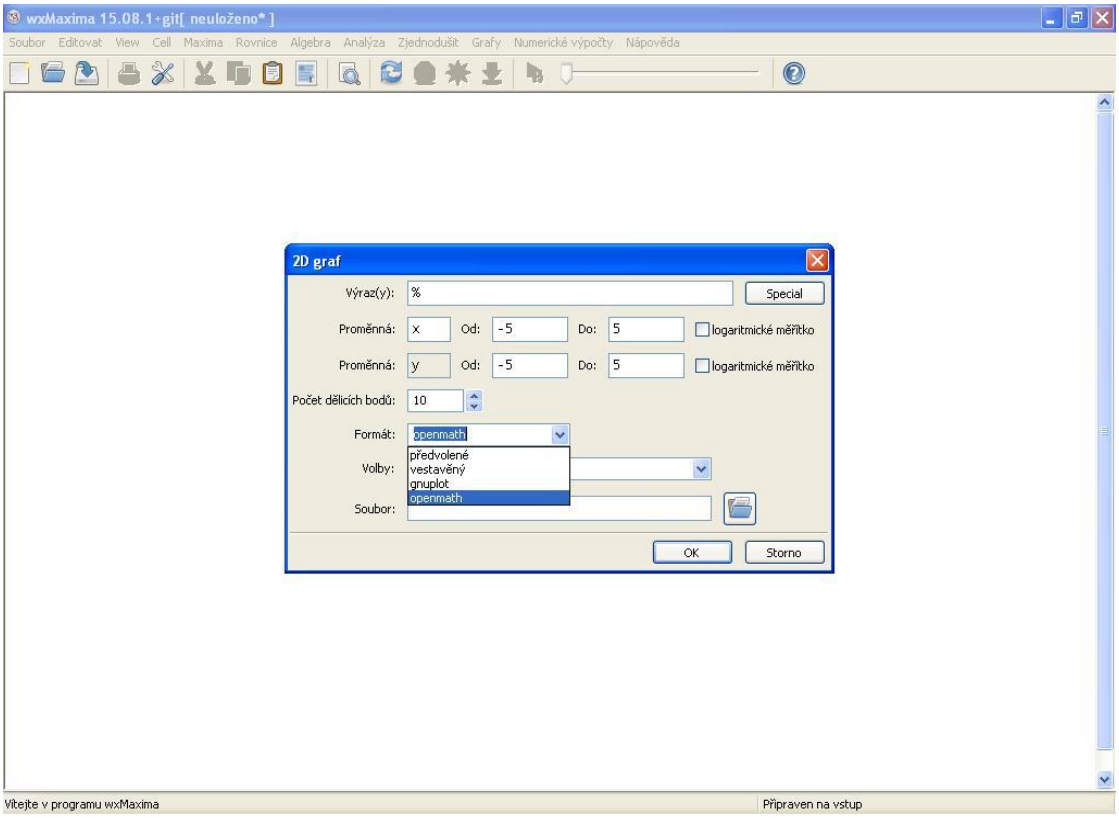
```
(%i1)  lx:[2];
(lx)   [2]

(%i2)  ly:[5];
(ly)   [5]

(%i3) wxplot2d(makelist([discrete,lx,ly],1),
[x,0,5],[y,0,6],[style,[points,3]])$
(%t3)
```



Obr. 2.: Vykreslení bodu

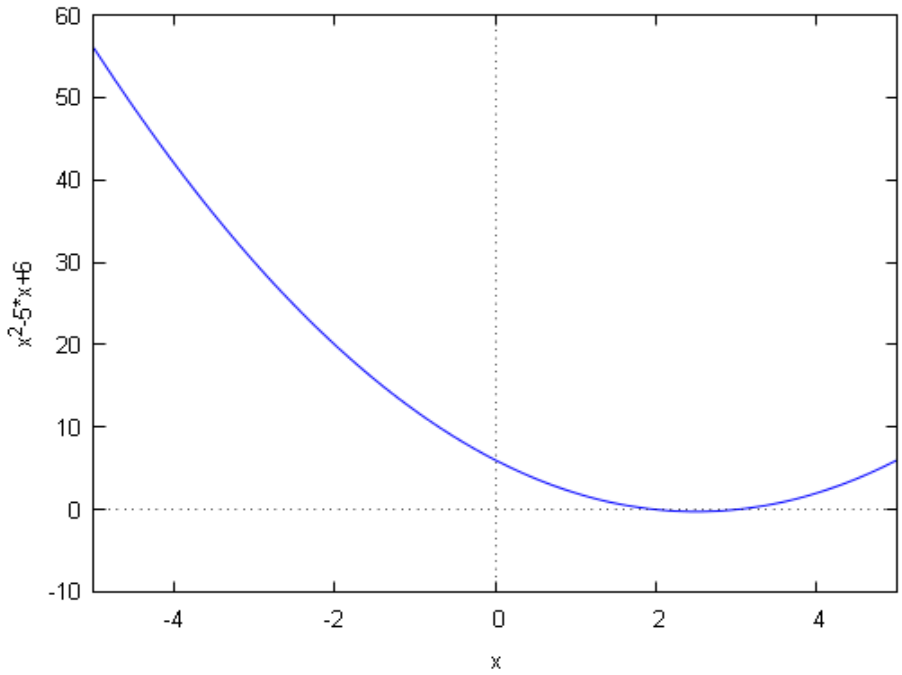
1.3.5.2 Zadávání přímky a jiných funkcí

Z analytické geometrie umíme vyjádřit funkci dvěma způsoby a to ve tvaru obecném nebo parametrickém. V záložce Grafy vybereme možnost *2D graf*... Zobrazí se nám zadávací okno, do kterého můžeme vložit naši funkci v obecném tvaru. Máme-li funkci zadanou parametricky, kliknutím na tlačítku *Special* vybereme možnost *Graf zadaný parametricky*. Současně můžeme zadat rozpětí zobrazovacího okna na ose *x* a *y*. Mimo jiné můžeme zvolit i styl zobrazení. Zvolíme-li při zadávání formát *vestavěný*, zobrazí se nám graf jako součást zobrazovacího okna. Možnost volby *gnuplot* nebo *openmath* nám vykreslí graf v novém vyskakovacím okně. Chceme-li zadat do jednoho grafu dvě funkce, oddělíme zadání jednotlivých funkcí čárkou.



Obr. 3.: Zadávací okno pro grafy 2D

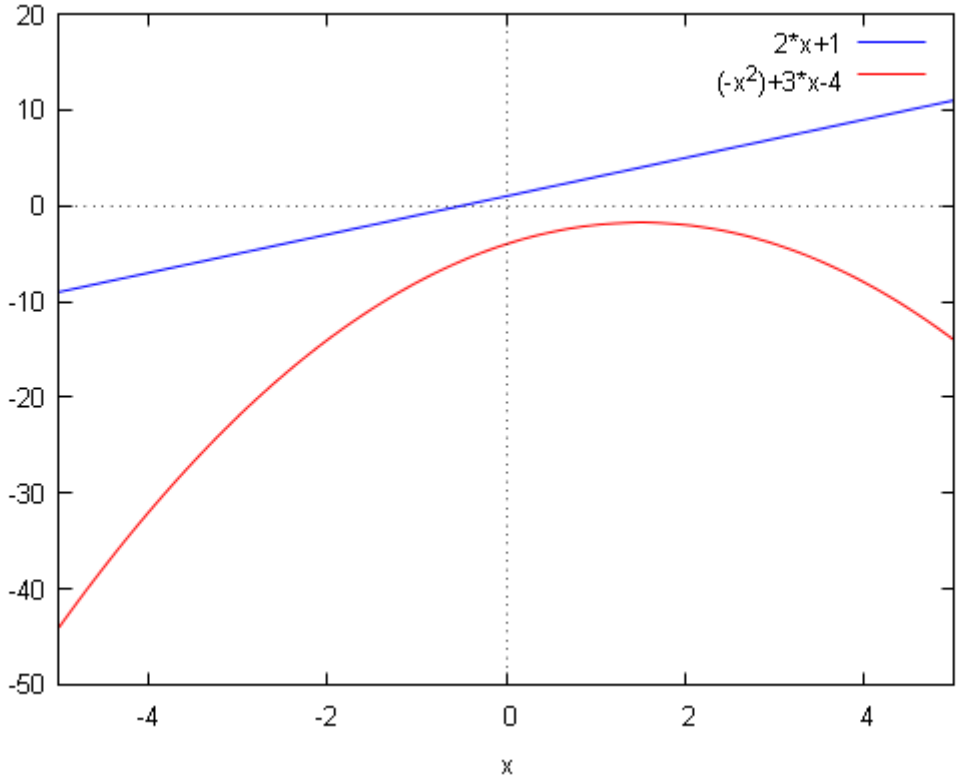
```
(%i1)  wxplot2d([x^2-5*x+6], [x,-5,5])$
(%t1)
```



Obr. 4.: Graf funkce

(%i2) wxplot2d([2*x+1,-x^2+3*x-4], [x,-5,5], [y,-5,5])$

(%t2)



Obr. 5.: vykreslení funkce a

Na obrázku č. 3 můžeme vidět zadávací okno grafu. Obrázek č. 4 nám vykresluje graf kvadratické funkce a na Obr. č. 5 nám ukazuje spojení dvou funkcí v jednom grafu.

2. Řešte vhodně vybrané rovnice pomocí Maxima funkcí solve, linsolve, find_root, allroots, realroots, eliminate.

V následující kapitole si připomeneme využití jednotlivých příkazů a současně v závěru kapitoly uvedeme i soubor několika příkazů, pomocí kterých vyřešíme nejen numericky ale i graficky zadanou úlohu.

2.1 solve

Tento příkaz jsme uvedli a rozebrali v kapitole 1.3.4.5 Výpočet kořenů polynomu, proto ho již nyní nebudeme rozebírat do detailu, pouze na příkladu si názorně připomeneme jeho zadávání.

```
(%i1)  a: x^2+3*x-4;
(%o1)  x^2+3*x-4

(%i2)  solve([a],[x]);
(%o2)  [x=1,x=-4]

(%i2)  solve([x^2+3*x-4],[x]);
(%o2)  [x=1,x=-4]
```

2.2 linsolve

linsolve() používáme k výpočtu soustav rovnic. Do argumentu na prvním místě v hranatých závorkách uvedeme rovnice nebo název funkce. Na druhém místě, opět do hranatých závorek uvedeme neznámé, které se v rovnicích vyskytují, a současně je chceme spočítat. Ukažme si tedy tento postup na následujícím příkladě, kdy máme vyřešit zadanou soustavu rovnic.

```
(%i1)  a1:2*x-3*y+4*z=5
(%o1)  4*z-3*y+2*x=5
```

```
(%i2)  a2:3*x+4*y-2*z=0;
(%o2)  -2*z+4*y+3*x=0

(%i3)  a3:-4*x+2*y+3*z=8;
(%o3)  3*z+2*y-4*x=8

(%i4)  linsolve ([a1,a2,a3], [x, y, z]);
(%o4)  [x=0,y=1,z=2]

(%i5)  linsolve ([2*x-3*y+4*z=5, 3*x+4*y-2*z=0, -4*x+2*y+3*z=8], [x, y, z]);
(%o5)  [x=0,y=1,z=2]
```

V případě, že se jedná o soustavu rovnic, kdy vyjde nekonečně mnoho výsledků, zobrazí se nám i výsledek v závislosti na určité proměnné, kterou posléze můžeme nahradit libovolným číslem. Opět si i tuto variantu demonstrujme na příkladu.

```
(%i1)  a:x+y+2*x=4;
(a)    2*z+y+x=4

(%i2)  b:x-2*y+z=0;
(b)    z-2*y+x=0

(%i3)  c:x-5*y=-4;
(c)    x-5*y=-4

(%i4)  linsolve ([a,b,c], [x, y, z]);
solve: dependent equations eliminated: (3)
(%o4)  [x=%r1,y=(%r1+4)/5,z=-(3*%r1-8)/5]
```

2.3 find_root

Pomocí funkce *find_root()* vypočítáme přibližné kořeny rovnice na určitém intervalu. Výsledek se nám zobrazí pomocí desetinného čísla. První dva členy v argumentu nám určují rovnici nebo funkci, kterou chceme vypočítat, a proměnnou. Další dva členové nám udávají rozmezí intervalu, na němž chceme rovnici vyřešit.

```
(%i1)  find_root (cos(x) - x/6, x, 0, %pi);
(%o1)  1.344751045375789
```

2.4 allroots

Potřebujeme-li získat výsledek v desetinném tvaru, použijeme příkaz *allroots()*. Do argumentu uvedeme naši řešenou rovnici. Pro lepší názornost uvedu rovnici řešenou pomocí příkazu *solve()* a příkazu *allrootas()*.

```
(%i1)  solve([x^3-8*x+6],[x]);
(%o1)  [x=(-(sqrt(3)*%i)/2-1/2)*((sqrt(269)*%i)/3^(3/2)-3)^(1/3)+(8*((sqrt(3)*%i)/2-
1/2))/(3*((sqrt(269)*%i)/3^(3/2)-3)^(1/3)),x=((sqrt(3)*%i)/2-1/2)*((sqrt(269)*%i)/3^(3/2)-
3)^(1/3)+
(8*(-(sqrt(3)*%i)/2-1/2))/(3*((sqrt(269)*%i)/3^(3/2)-
3)^(1/3)),x=((sqrt(269)*%i)/3^(3/2)-3)^(1/3)+8/(3*((sqrt(269)*%i)/3^(3/2)-3)^(1/3))]

(%i2)  allroots(x^3-8*x+6);
(%o2)  [x=0.818558051726674,x=2.32887218197108,x=-3.147430233697754]
```

2.5 realroots

Chceme-li získat pouze reálné kořeny rovnice, zvolíme funkci *realroots()*. Jako u předchozí funkce vložíme do argumentu pouze naši řešenou rovnici.

```
(%i1)  solve([x^4+x^3+x^2+x],[x]);
(%o1)  [x=-%i,x=%i,x=-1,x=0]

(%i2)  realroots(x^4+x^3+x^2+x);
(%o2)  [x=-1,x=0]
```

2.6 eliminate

Zadáním příkazu *eliminate()* odstraníme zadané proměnné ze soustav rovnic. Do argumentu na první místo uvedeme do hranatých závorek odkaz na dané rovnice nebo je jednotlivě vypíšeme. Na druhé pozici uvedeme opět do hranatých závorek proměnné, které chceme z rovnic eliminovat (odstranit). Principiálně můžeme danou metodu přirovnat k metodě řešení soustavy rovnic sčítací metodou. Názorně si tento způsob ukažme na na soustavě tří rovnic, kdy v prvním případě budeme chtít eliminovat pouze *x* a v druhém případě *x, y*.

```
(%i1)  a:3*x+2*y-7*z=2;
(%o1)  -7*z+2*y+3*x=2

(%i2)  b:2*x+3*y-4*z=1;
(%o2)  -4*z+3*y+2*x=1

(%i3)  c:4*x+7*y+2*z=6;
(%o3)  2*z+7*y+4*x=6

(%i4)  eliminate([a,b,c],[x]);
(%o4)  [34*z+13*y-10,2*z+5*y+1]

(%i5)  eliminate([a,b,c],[x,y]);
(%o5)  [-9*(16*z-7)]
```

2.7 Celkové řešení rovnic do 10. řádu

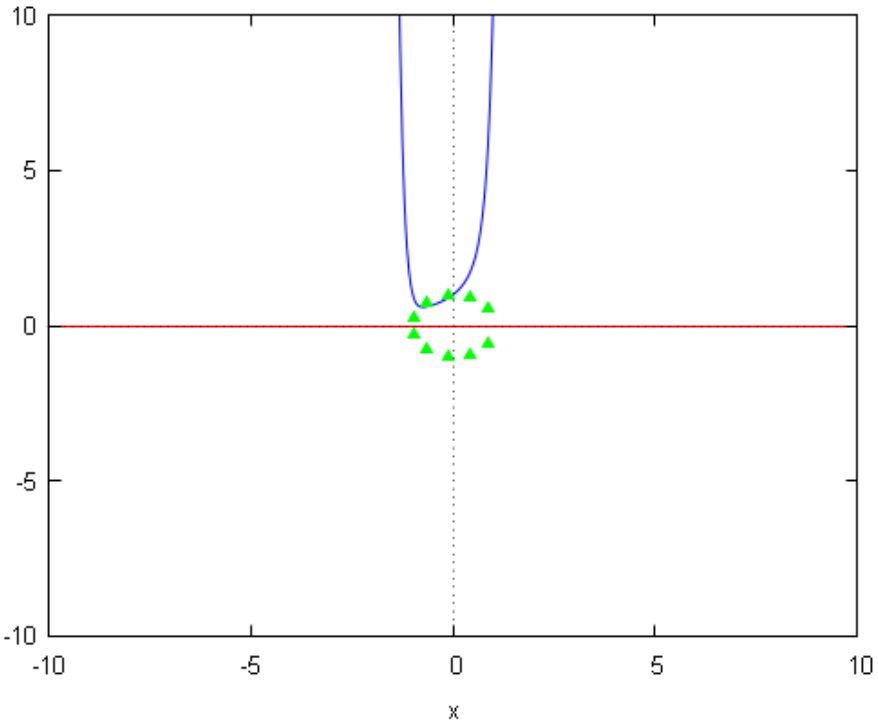
V této kapitole si ukážeme a rozebereme soubor několik provázaných zdrojových příkazů/kódů, které nám umožní spočítat a vykreslit kořeny dané rovnice a rovněž nám ji i vykreslí.

```
(%i1)  [a,b,c,d,e,f,g,h,i,j,k]:[1,1,1,1,1,1,1,1,1,1,1];
(%o1)  [1,1,1,1,1,1,1,1,1,1,1]
```

V prvním kroku vložíme výše uvedený příkaz a nahradíme jedničky koeficienty zadávané rovnice. V případě, že se daný člen v rovnici nevyskytuje, píšeme do zadání 0. Po uložení všech proměnných vložíme další soubor příkazů.

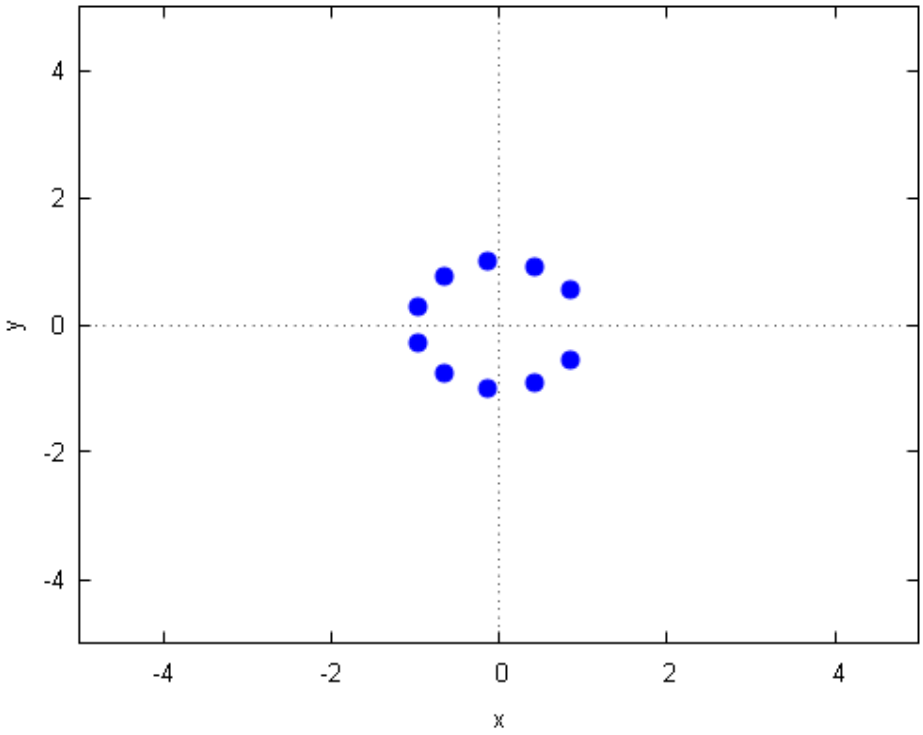
```
(%i25)
allroots(a*x^10+b*x^9+c*x^8+d*x^7+e*x^6+f*x^5+g*x^4+h*x^3+i*x^2+j*x+k);rhs
(realpart(%o2)[1])$rhs(realpart(%o2)[2])$rhs(realpart(%o2)[3])$rhs(realpart(%o2)[4])$rhs(re
alpart(%o2)[5])$rhs(realpart(%o2)[6])$rhs(realpart(%o2)[7])$rhs(realpart(%o2)[8])$rhs(realp
art(%o2)[9])$rhs(realpart(%o2)[10])$rhs(imagpart(%o2)[1])$rhs(imagpart(%o2)[2])$rhs(ima
gpart(%o2)[3])$rhs(imagpart(%o2)[4])$rhs(imagpart(%o2)[5])$rhs(imagpart(%o2)[6])$rhs(i
magpart(%o2)[7])$rhs(imagpart(%o2)[8])$rhs(imagpart(%o2)[9])$rhs(imagpart(%o2)[10])$l
x:[(%o3),(%o4),(%o5),(%o6),(%o7),(%o8),(%o9),(%o10),(%o11),(%o12)]$ly:[(%o13),(%o1
4),(%o15),(%o16),(%o17),(%o18),(%o19),(%o20),(%o21),(%o22)]$wxplot2d([0,a*x^10+b*
x^9+c*x^8+d*x^7+e*x^6+f*x^5+g*x^4+h*x^3+i*x^2+j*x+k,[discrete,lx,ly]],[x,-10,10],[y,-
10,10],[legend,""],[style,[lines,1,2],[lines,1,1],[points,2]])$wxplot2d(makelist([discrete,lx,ly],
1),
[x,-5,5],[y,-5,5],[legend,""],[style,[points,3]])$kill(all)$
(%o2)  [x=0.9096319953545184*%i+0.4154150130018864,x=0.4154150130018864-
0.9096319953545184*%i
,x=0.7557495743542584*%i-0.654860733945285,x=-0.7557495743542584*%i-
0.654860733945285,x=
0.9898214418809328*%i-0.1423148382732852,x=-0.9898214418809328*%i-
0.1423148382732852,x=
0.5406408174555979*%i+0.8412535328311811,x=0.8412535328311811-
0.5406408174555979*%i,x=
0.2817325568414286*%i-0.9594929736144973,x=-0.2817325568414286*%i-
0.9594929736144973]
```

(%t25)



Obr. č. 6.: Grafické řešení rovnice

(%t26)



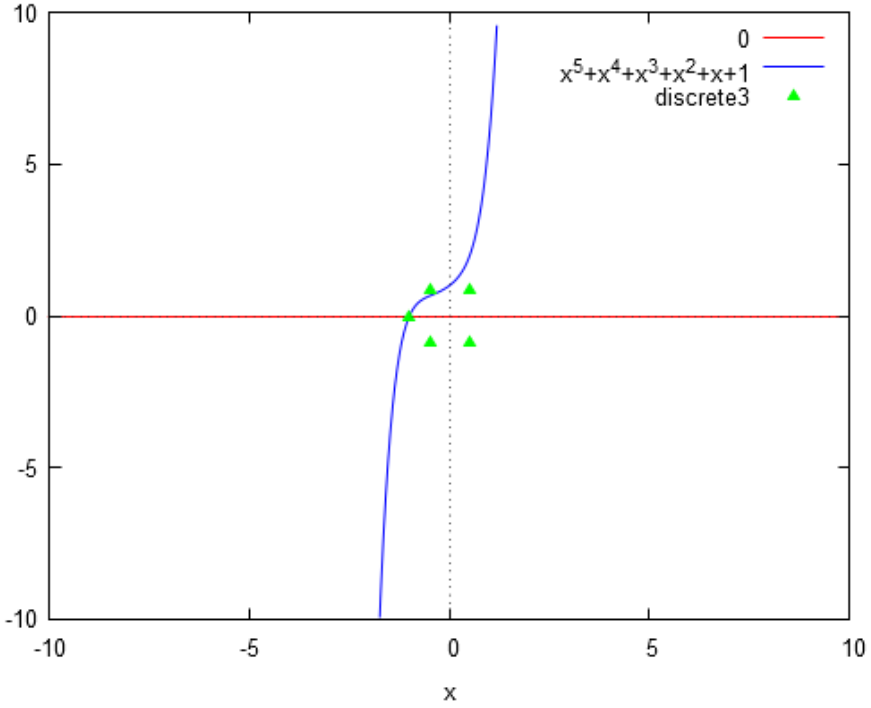
Obr. č. 7.: Zobrazení všech řešení rovnice

Pokud zadáváme rovnici nižšího řádu než je 10 (tedy první koeficienty nabývají hodnoty 0) zobrazí nám program error podle počtu chybějících hodnot. Ale i přes toto upozornění je program a soubor takto zadaných příkazů schopný námi zadané rovnice spočítat a vyhodnotit.

```
(%i1)  [a,b,c,d,e,f,g,h,i,j,k]:[0,0,0,0,0,1,1,1,1,1,1];
(%o1)  [0,0,0,0,0,1,1,1,1,1,1]

(%i27)
allroots(a*x^10+b*x^9+c*x^8+d*x^7+e*x^6+f*x^5+g*x^4+h*x^3+i*x^2+j*x+k);rhs
(realpart(%o2)[1])$rhs(realpart(%o2)[2])$rhs(realpart(%o2)[3])$rhs(realpart(%o2)[4])$rhs(re
alpart(%o2)[5])$rhs(realpart(%o2)[6])$rhs(realpart(%o2)[7])$rhs(realpart(%o2)[8])$rhs(realp
art(%o2)[9])$rhs(realpart(%o2)[10])$rhs(imagpart(%o2)[1])$rhs(imagpart(%o2)[2])$rhs(ima
gpart(%o2)[3])$rhs(imagpart(%o2)[4])$rhs(imagpart(%o2)[5])$rhs(imagpart(%o2)[6])$rhs(i
magpart(%o2)[7])$rhs(imagpart(%o2)[8])$rhs(imagpart(%o2)[9])$rhs(imagpart(%o2)[10])$l
x:[(%o3),(%o4),(%o5),(%o6),(%o7),(%o8),(%o9),(%o10),(%o11),(%o12)]$ly:[(%o13),(%o1
4),(%o15),(%o16),(%o17),(%o18),(%o19),(%o20),(%o21),(%o22)]$wxplot2d([0,a*x^10+b*
x^9+c*x^8+d*x^7+e*x^6+f*x^5+g*x^4+h*x^3+i*x^2+j*x+k,[discrete,lx,ly]],[x,-10,10],[y,-
10,10],[style,[lines,1,2],[lines,1,1],[points,2]])$wxplot2d(makelist([discrete,lx,ly],1),
[x,-5,5],[y,-5,5],[style,[points,3]])$kill(all)$

(%o2)  [x=0.8660254037844386*%i+0.5,x=0.5-
0.8660254037844386*%i,x=0.866025403784439*%i-0.5,
x=-0.866025403784439*%i-0.5,x=-1.0]part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);plot2d: some values were clipped.
```

Warning: excluding 5 points with non-numerical values.

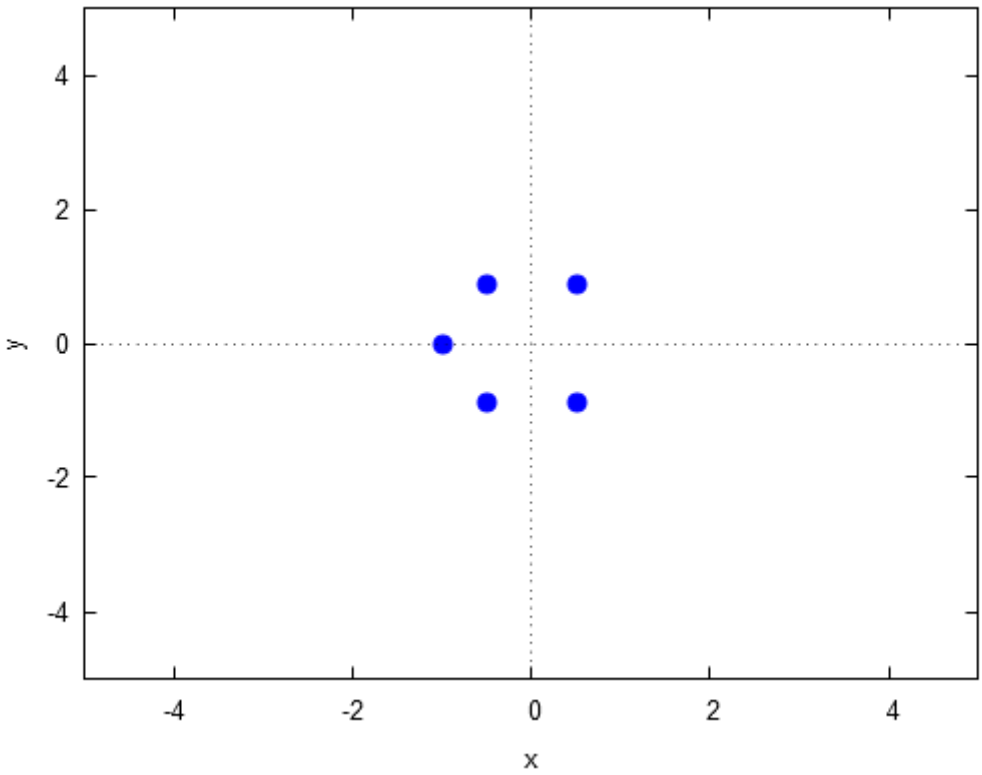
(%t25)



Obr. č. 8.: Grafické řešení rovnice

Warning: excluding 5 points with non-numerical values.

(%t26)



Obr. č. 9.: Zobrazení všech řešení rovnice

Jak je patrné na tomto příkladu, řešíme rovnici 5. řádu. Nahradili jsme prvních 5 členů nulou. Tyto nulové hodnoty zapříčinili vypsání deseti výstupů označených error. Jde právě o deset výstupů, kdy právě k jednomu bodu náleží dva výstupy a to pro *x-ovou* a *y-ovou* složku bodu.

Pro lepší a snadnější manipulaci jsem umístil již hotový soubor na volně přístupné webové stránky, kde je libovolně ke stažení. Odkaz:

http://ulozto.cz/xkgcjbQo/reseni-rovnic-10-radu-wxmx

2.8 Komplikace při řešení rovnic vyššího řádu

Při zkoumání možností programu jsem narazil na jeden nedostatek při řešení úloh. Jednalo se o způsob zadávání a dosazování do výsledků. Pokud jsme chtěli vyřešit konkrétní příklad, mohli jsme zvolit libovolný příkaz a tím i získat konkrétní výsledek. K problému přišlo, pokud jsme nechali vyřešit rovnici obecně a posléze jsme do získaných kořenů dosazovali za koeficienty. Jedná se o to, že program pasivně dosazuje do vypočítaných kořenů a již nadále není schopen dopočítat kořen rovnice. Ukažme si tento problém se zadáváním na konkrétních příkladech.

Nejdříve vyřešíme rovnici

```
(%i1)   solve([x^3-6*x^2+11*x-6],[x]);

(%o1)   [x=1,x=2,x=3]
```

Jak vidíme, přímo zadanou rovnici nám program bezchybně vypočítá. Nyní si ale pro názornost zkusíme nejdřív vyřešit kořeny obecně a posléze za jednotlivé kořeny budeme dosazovat.

```
(%i2)   solve([a*x^3+b*x^2+c*x+d=0],[x]);

(%o2)   [x=(-(sqrt(3)*%i)/2-1/2)*(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-
b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-9*a*b*c+2*b^3)/(54*a^3))^(1/3)-

(((sqrt(3)*%i)/2-1/2)*(3*a*c-b^2))/(9*a^2*(sqrt(27*a^2*d^2+(4*b^3-
18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3))-b/(3*a),x=((sqrt(3)*%i)/2-1/2)*

(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3)-

((-(sqrt(3)*%i)/2-1/2)*(3*a*c-b^2))/(9*a^2*(sqrt(27*a^2*d^2+(4*b^3-
18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3))-b/(3*a),x=

(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3)-

(3*a*c-b^2)/(9*a^2*(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-
b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-9*a*b*c+2*b^3)/(54*a^3))^(1/3))-b/(3*a)]


(%i3)   [a,b,c,d]:[1,-6,11,-6];
(%o3)   [1,-6,11,-6]


(%i6)   x=(-(sqrt(3)*%i)/2-1/2)*(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-
b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-9*a*b*c+2*b^3)/(54*a^3))^(1/3)-

(((sqrt(3)*%i)/2-1/2)*(3*a*c-b^2))/(9*a^2*(sqrt(27*a^2*d^2+(4*b^3-
18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3))-b/(3*a);x=((sqrt(3)*%i)/2-1/2)*

(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3)-

((-(sqrt(3)*%i)/2-1/2)*(3*a*c-b^2))/(9*a^2*(sqrt(27*a^2*d^2+(4*b^3-
18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-
9*a*b*c+2*b^3)/(54*a^3))^(1/3))-b/(3*a);x=
```

(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-9*a*b*c+2*b^3)/(54*a^3))^(1/3)-

(3*a*c-b^2)/(9*a^2*(sqrt(27*a^2*d^2+(4*b^3-18*a*b*c)*d+4*a*c^3-b^2*c^2)/(2*3^(3/2)*a^2)-(27*a^2*d-9*a*b*c+2*b^3)/(54*a^3))^(1/3))-b/(3*a);

(%o4) x=((sqrt(3)*%i)/2-1/2)/((-1)^(1/6)*sqrt(3))+((-1)^(1/6)*(-(sqrt(3)*%i)/2-1/2))/sqrt(3)+2

(%o5) x=((-1)^(1/6)*((sqrt(3)*%i)/2-1/2))/sqrt(3)+(-(sqrt(3)*%i)/2-1/2)/((-1)^(1/6)*sqrt(3))+2

(%o6) x=(-1)^(1/6)/sqrt(3)+1/((-1)^(1/6)*sqrt(3))+2

Jak je patrné na tomto příkladu, musíme vždy vhodně zvolit způsob zadávání. Bohužel v tomto případě neumí program dané koeficienty vhodně dosadit a vypočítat námi hledané kořeny zadané rovnice

3. Typy algebraických rovnic

Na úvod této kapitoly si stručně připomeňme základní typy rovnic, se kterými se můžeme ve škole setkat:

Řád rovnice	Obecný tvar	Název
1.		Lineární
2.		Kvadratická
3.		Kubická
4.		
5. a vyšší	…	

Tab. č. 1.: Přehled typů rovnic

3.1 Řešení algebraických rovnic 3. stupně

Kubickou rovnicí nazýváme každou rovnic, kterou můžeme pomocí ekvivalentních úprav převést do tvaru , kde *x* nazýváme proměnnou, koeficienty a platí:

-
-

Člen nazýváme kubický, kvadratický člen, lineární člen a absolutní člen.

Kubickou rovnici můžeme mít zadanou:

1. **V součinovém tvaru**

kde jsou kořeny rovnice.

2. **Bez absolutního členu**

Při řešení můžeme vytknout z levé strany rovnice *x*. Tak získáme tvar kdy je právě jedním z kořenů dané rovnice a zbývající dva kořeny dopočítáme z kvadratické rovnice.

3. Ostatní

Při obecném řešení můžeme zvolit jednu z mnoha metod. My si nyní některé z nich uvedeme.

Jednou z metod řešení kubických rovnic je tzv. separace kořenů. Jedná se jen o přibližnou metodu zjištění kořenů rovnice. Tato metoda vychází z vlastnosti grafu funkce, kdy graf musí projít přes osu x, tedy musí platit . Řešením je hodnota x, kde k tomu dojde. Je patrné, že se jedná o velmi zdlouhavou, početně náročnou a ne zcela úplně přesnou metodu řešení. Další metodou, kterou lze použít u výpočtu kořenů rovnic, je tzv. Hornerovo schéma. Tuto metodu je vhodné využívat u rovnic řešených na množině celých čísel. Prvním krokem při využití Hornerova schématu je najít všechny dělitele absolutního členu. Sestavíme tabulku, kdy v horním řádku od druhého sloupce jsou vypsány všechny koeficienty jednotlivých členů. Do prvního sloupce na druhém řádku zapíšeme první z dělitelů absolutního členu. Do druhého sloupce, druhého řádku zapíšeme hodnotu koeficientu kubického členu. Tuto hodnotu vynásobíme námi zvoleným dělitelem a tento součin přičteme k hodnotě kvadratického členu a výsledek zapíšeme do třetího sloupce druhého řádku. Opět vezmeme výsledek, vynásobíme s ním dělitele a přičteme ho k lineárnímu členu a výsledek o pět zapíšeme na dané místo. Tento krok opět zopakujeme. Pokud bude výsledek po posledním kroku různý od 0, není daný dělitel jedním z kořenů rovnice a musíme zvolit jiného dělitele. Je-li poslední výsledek roven 0, pak jsme získali první kořen rovnice a další hodnoty, pomocí kterých dalším počítáním získáme zbývající kořeny rovnice. Opět i u dalšího kroku si nejdříve určíme dělitele u posledního nově získaného člena a postup opakujeme, dokud nezískáme všechny kořeny dané rovnice.

Př. 1.: Řešte na množině

	1	-1	-8	12
1	1	0	-8	4
-1	1	-2	-6	18
2	1	1	-6	0
-1	1	0	-6	
1	1	2	-4	
-2	1	-1	-4	
2	1	3	0	

-1	1	2		
1	1	4		
-3	1	0		

Tab. č. 2.: Hornerovo schéma

Další vhodná metoda pro řešení je podobná Hornérovu schématu. Princip spočívá v nalezení jednoho z reálných kořenů dané rovnice a následným vydělením rovnice výrazem Po tomto kroku nám vznikne kvadratická rovnice, kterou jsme schopni již snadno řešit.

Př. 3.: Řešte na množině

Z hodnoty výrazu 28 určíme, že jedním z možných kořenů této rovnice mohou být čísla Výběrem z těchto čísel určíme, že první kořen této rovnice je Proto pokračujeme následně:

Nyní vyřešíme kvadratickou rovnici

Kořeny dané rovnice jsou

Další a poslední metodou pro výpočet kubické rovnice je užití tzv. Cardanových vzorců. Ještě před jejich využitím si je odvoďme. Obecný zápis kubické rovnice je:

Takto vzniklé koeficienty nově označíme jako

Nyní se vhodnou substitucí — zbavíme kvadratického členu

Opět provedeme vhodné nahrazení nově vzniklých koeficientů za proměnné

Získáme nový tzv. redukovaný tvar kubické rovnice

Řešení této rovnice lze hledat v součtu dvou zatím neurčených čísel

Tento tvar dosadíme do redukované rovnice

Pokud umocníme rovnici − získáme výraz − . Porovnáme-li tyto dvě rovnice, můžeme je přirovnat k Vietovým vztahům, kde jsou kořeny kvadratické rovnic.

Součet nám dá první kořen rovnice

Odmocněním výrazů

získáme další řešení

„Veta 1,7. Každé komplexné číslo má práve n rôznych n-tých odmocnín. Je to týchto n čísel:

Pritom značí reálné nezáporné číslo, kterého n-tá mocnina je r.“(Schwarz, str. 24).

„Veta 1,8. Nech je celé číslo a nech — — Potom čísla sú práve všetky n-té odmocniny z čísla1.“ (Schwarz, str. 27).

Chceme-li určit hodnotu určíme jeho hodnoty pro . Dosazením získáme:

Tato řešení je nutno zkombinovat, ale musíme dodržet podmínku –.

1. varianta

2. varianta

Když se podíváme na tyto varianty, můžeme vidět, že se shodují s našimi dříve zjištěnými možnostmi. Proto máme následující tři řešení

Kde – — , – – – , – – – . Po vypočítání hodnot dosadíme získané hodnoty do námi první provedené substituce — a vypočítáme kořeny Při odvozování Cardanových vztahů se použila řada ekvivalentních úprav, které daný výsledek mohou zkreslit a tak i požadovaný hotový výraz nemusí odpovídat skutečnosti. Tento fakt a tedy nutnost správně zvolit postup, jakým budeme danou rovnici počítat, si ukážeme na jednoduché rovnici . Už jen při letmém pohledu na tuto rovnici mohou někteří její řešitelé určit, že jedním z kořenů této rovnice je . Dokažme si tedy na této rovnici nevhodně zvolený postup při jejím řešení, kdy budeme počítat složité rovnice a navíc výsledek rovněž nebude odpovídat řešení rovnice.

Následně zvolíme jinou početní metodu, díky níž určíme přesný výsledek a to za mnohem kratší domu.

Př. 3.: Řešte na množině

a) Použitím Cardanových vzorců:

Zavede substituci

Při použití Cardanových vzorců jsme získali kořeny. Už na první pohled je patrné, že výsledný kořen na množině reálných čísel není správné řešení rovnice, a proto je tento postup pro daný příklad nevhodný. K chybě při řešení došlo důsledkem zaokrouhlování a užitím ekvivalentních úprav. Zkusme nyní tento stejný příklad vyřešit pomocí jiné metody.

b) Použitím Hornerova schématu:

	1	1	1	-3
-3	1	-2	-5	-18
-1	1	0	1	-4
1	1	2	3	0
-3	1	-1	6	
-1	1	1	2	
1	1	3	6	
3	1	5	18	

Tab. č. 3. Hornerovo schéma

Jak je patrné tak zadaná rovnice má na množině reálných čísel pouze jedno řešení .

Získáním odlišných výsledků u tohoto příkladu, je nutné správně zvolit vhodnou metodu řešení. Podobných příkladů, kdy při použití dvou různých metod získáme odlišné výsledky, je několik.

3.2 Řešení algebraických rovnic 4. stupně

Kvartickou rovnicí nazýváme každou rovnici, kterou můžeme pomocí ekvivalentních úprav převést do tvaru , kde *x* nazýváme proměnnou, koeficienty a platí:

-
-

Rovnici 4. řádu můžeme mít zadanou:

1. **V součinovém tvaru**

kde jsou kořeny rovnice.

2. **Bez absolutního členu**

Při řešení můžeme vytknout z levé strany rovnice *x*. Tedy získáme tvar

kdy Postup při řešení je stejný jako při řešení rovnic 3. řádu.

3. **Ostatní**

Řešení daných rovnic je obdobné jako u řešení rovnic 3. řádu. Liší se pouze tvar Cardanových vzorců, proto si ukažme ve zkratce jejich odvozování.

Zavedeme substituci — a zbavíme se kubického členu. Získáme rovnici v redukovaném tvaru

Opět provedeme další substituci − Hodnotu − jsme si zvolili, abychom v dalším kroku získali jednodušší tvar. Nyní si opět zvolíme vhodně podmínky pro výpočet (je nutné dbát na to, aby si zvolené podmínky neodporovaly).

Dosadíme do předchozí rovnice a získáme tvar

Opět zvolíme z dané rovnice dvě podmínky

Opětovným dosazením získáme rovnici ve tvaru

Umocněním předchozí rovnice na druhou

Předchozí tři vztahy nám ukazují, že jsou kořeny rovnice třetího stupně, kterou můžeme zapsat jako

Takto získaná rovnice se nazývá kubická rezolventa rovnice (název rezolventa pochází z lat. resolvere = vyřešení, řešení) a kořeny jsou Potom platí

—

—

—

Vzhledem k rovnici nemůžeme volit odmocniny libovolně (nesmíme zapomínat na to, že jsme danou rovnici umocnili). Proto při prvních dvou odmocninách můžeme volit znamínka libovolně, ale třetí je už určené danou rovnicí. Díky tomu získáme soustavu čtyř rovnic, kdy jejich dvojnásobky můžeme zapsat jako

— — —

— — —

— — —

— — —

Po vyřešení získané hodnoty dosadíme do první substituce a vypočítáme hledané kořeny dané rovnice.

Př. 4.: Řešte na množině

Vyřešíme kořeny této kubické rezolventní rovnice. Dopočítáme hodnoty

Odtud dosazením do původní substituce získáme hodnoty

4. Zkoumání úspěšných řešení rovnic 3. a 4. řádu použitím algebraických metod

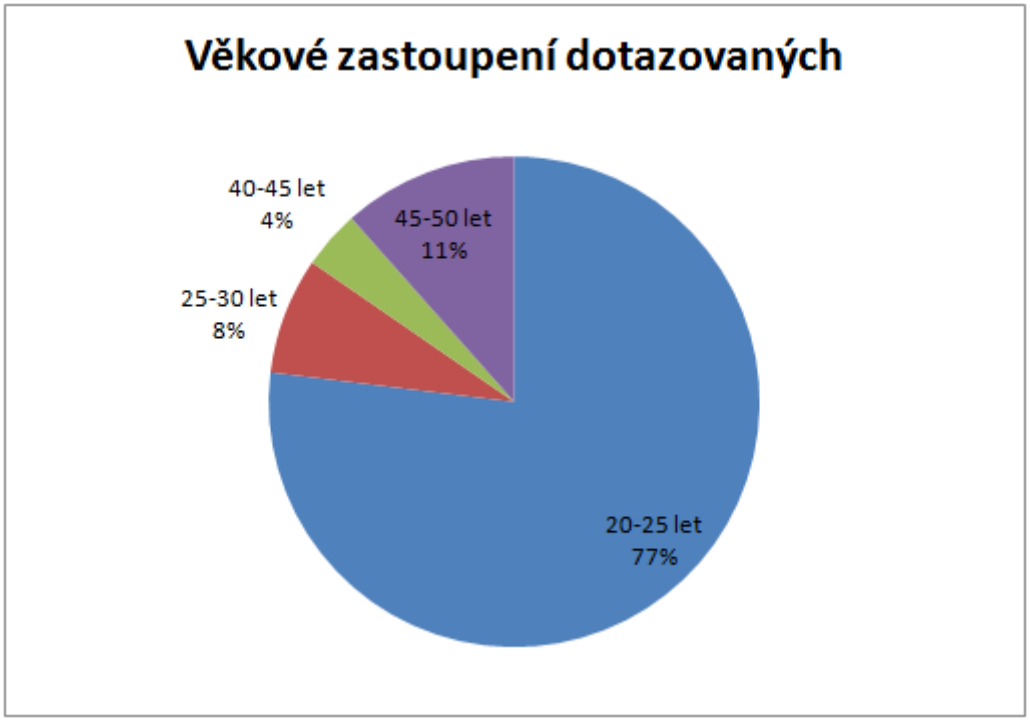
V návaznosti na dané téma mojí diplomové práce jsem vytvořil dotazník, který je zaměřený na řešení rovnic 3. a 4. řádu bez použití jakéhokoliv počítačového softwaru. Cílem výzkumu bylo poukázat na mnohdy výhodnější využívání nejrůznějších matematických softwarů k výpočtu daných příkladů. Zkoumanými subjekty byli žáci 1. ročníku navazujícího magisterského studia učitelství matematiky pro 2. stupeň základních škol, učitelky na středních i základních školách, mimo jiné jsem oslovil i studenty aplikované matematiky na Přírodovědecké univerzitě v Olomouci a v Brně. Celkem se tohoto průzkumu zúčastnilo celkem 26 dotazovaných respondentů, kteří měli za úkol spočítat a odpovědět na zadané příklady.

Celý průzkum probíhal v elektronické podobě prostřednictvím Google účtu a za využití možnosti vytvoření a sdílení formulářů na Google Disku. V úvodu dotazníku byl dotazovaný seznámen s jeho obsahem a náplní. Současně byl vysvětlen a názorně ukázán způsob odpovídání. Spolu s objasněním účelu dotazníku se na první straně vyskytují již první otázky zaměřené na zjištění obecných základních údajů jako je pohlaví, věk, vystudovaná fakulta, typ školy, kde dotyčný vyučuje (v případě, že již učí) a délka praxe. Druhá část dotazníku obsahovala celkem sedm příkladů. Čtyři byly na řešení rovnic 3. stupně a zbývající tři na řešení rovnic 4. stupně. V poslední sekci dotazníku měli dotazovaní možnost se vyjádřit k obtížnosti zadávaných příkladů, a zda by při jejich řešení raději využili nějaký z matematických softwarů. Jako protihodnotu za vyplnění dotazníku jsem zveřejnil pouze pro dotyčné osoby již vypracovanou verzi manuálu jak pracovat v prostředí počítačového softwaru Maxima CAS. Někteří učitelé využili nabídky a zvolený dotazník vyplňovali v tištěné podobě. Takto získaná data jsem posléze přepsal do elektronické podoby a vyhodnotil je. Připojuji zde odkaz na zmiňovaný dotazník.

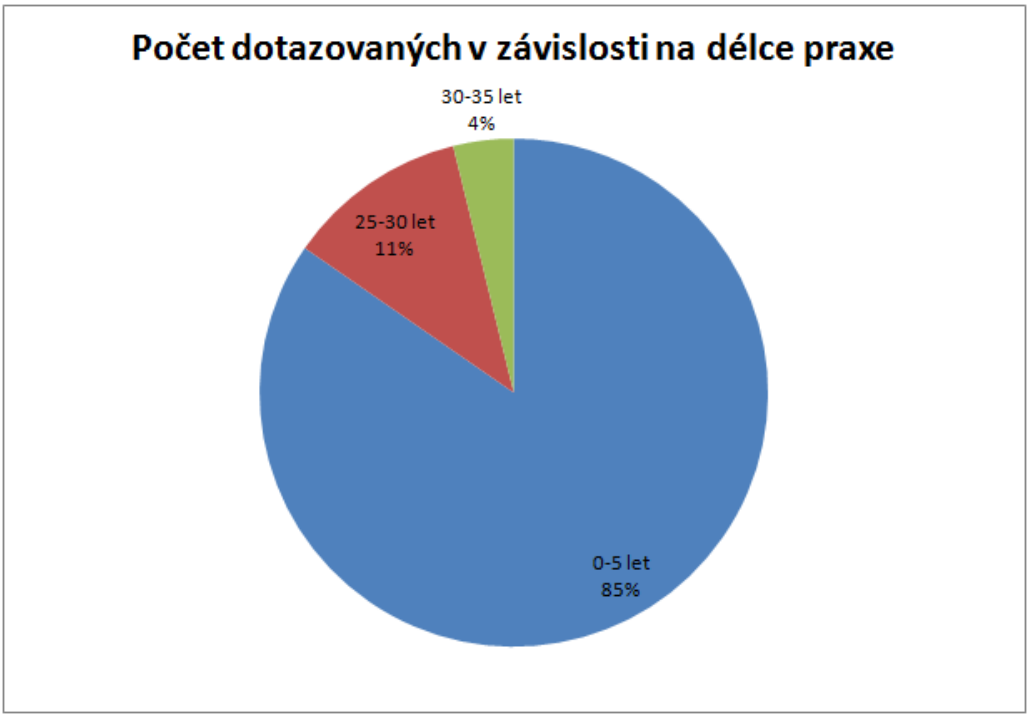
http://goo.gl/forms/kDOl58fcJN

Prostřednictvím následujících grafů obeznámím čtenáře s obecnými údaji o dotazovaných jako je pohlaví, věk, délka praxe, apod.

Graf č. 1.: Počet dotazovaných lidí z pohledu pohlaví



Graf č. 2.: Věkové zastoupení dotazovaných



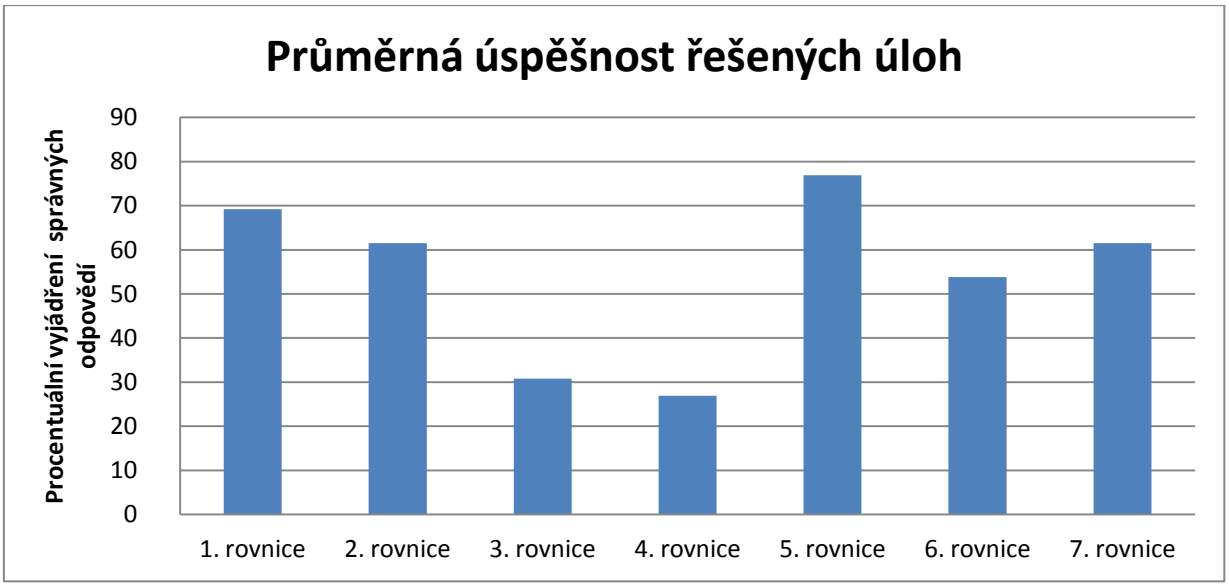
Graf č. 3.: Počet dotazovaných v závislosti na délce praxe

Nyní se již můžeme zaměřit na získaná data a jejich následné vyhodnocení. Jednotlivé výsledky jsou zapsané v tabulce. Jedná se již o upravenou a částečně vyhodnocenou tabulku. Odeslané správné odpovědi jsou v tabulce označeny zeleně a nabývají hodnoty 1. Naopak špatné odpovědi jsou zvýrazněny červeně a jejich hodnota je rovna 0. Spolu se všemi vyjádřenými výsledky se v tabulce vyskytují i součty správných odpovědí jednotlivých dotazovaných a současně součet správných odpovědí na jednotlivé příklady.

Jednotlivé součty jsou barevně zvýrazněny. Hodnoty s největším počtem správných odpovědí jsou zvýrazněny sytě. S nižším počtem bodů přechází barva z tmavě syté barvy do bílé.

Přehled úspěšnosti řešení jednotlivých příkladů								
	1. rovnice	2. rovnice	3. rovnice	4. rovnice	5. rovnice	6. rovnice	7. rovnice	
1	1	0	0	0	1	1	1	4
2	1	0	0	0	1	1	1	4
3	1	0	0	0	0	0	0	1
4	1	1	0	0	1	0	0	3
5	0	1	0	0	0	0	0	1
6	0	1	0	0	0	1	0	2
7	0	0	0	0	0	0	0	0
8	1	0	0	0	1	0	0	2
9	1	1	1	1	1	1	1	7
10	1	1	1	1	1	1	1	7
11	1	1	1	0	1	1	1	6
12	0	0	0	0	1	0	0	1
13	0	0	0	0	1	1	1	3
14	1	1	0	0	1	0	1	4
15	1	1	0	0	1	1	1	5
16	1	1	0	0	1	0	0	3
17	1	1	0	0	1	1	1	5
18	0	0	0	0	1	0	0	1
19	1	1	1	1	1	0	1	6
20	0	0	0	0	0	0	1	1
21	0	0	0	0	0	0	0	0
22	1	1	0	0	1	1	1	5
23	1	1	1	1	1	1	1	7
24	1	1	1	1	1	1	1	7
25	1	1	1	1	1	1	1	7
26	1	1	1	1	1	1	1	7
	18	16	8	7	20	14	16	

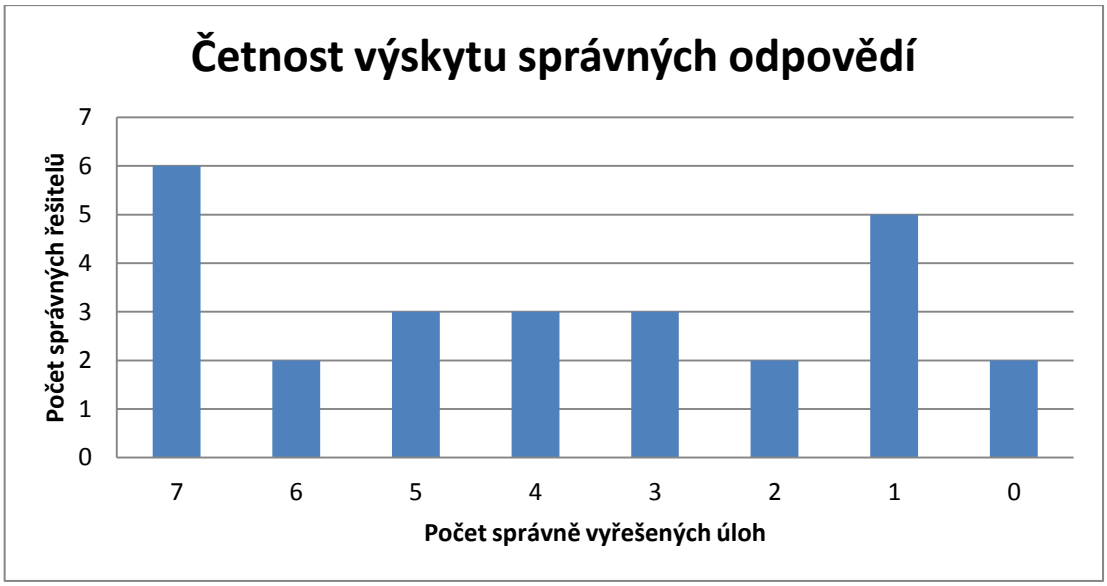
Tab. č. 4.: Přehled úspěšnosti řešení jednotlivých příkladů



Graf č. 4.: Průměrná úspěšnost řešení jednotlivých úloh

Jak je patrné, celková průměrná úspěšnost při řešení příkladů byla pouze 54,4%. Nejhůře dopadli rovnice a . Průměrná úspěšnost těchto dvou rovnic je 28,85%. Tento nízký počet úspěšných řešení můžeme přiřadit výsledkům, které vyšly v komplexních číslech. Následným počítáním s nimi mohlo dojít k numerickým chybám, neboť do odpovědí se zadával nejmenší reálný kořen a součet všech kořenů. Kořeny ostatních rovnic vyšly na množině reálných čísel, a proto počítání s nimi již nedělalo dotazovaným početní obtíže. Naopak mezi nejvíce spočítaných rovnic se řadí rovnice s celými, reálnými kořeny. Nejlépe dopadla rovnice 4. řádu , kterou vyřešilo 77% zúčastněných. Zbývající čtyři rovnice vyřešilo průměrně 61,54%. Důvodem tak relativně nízkého počtu úspěšných řešitelů může být i již zmiňovaný způsob zadávání odpovědí do elektronického dotazníku.

Při počítání se značně prokázala letitá praxe učitelů. Právě tito dotazovaní spočítali všechny příklady správně. Byly i dva studenti, kteří rovněž byli 100% řešiteli. Při podrobnějším zkoumání jejich výsledků a termínu odevzdávání, který se liší pouze v jedné sekundě, lze zvažovat, zda se jedná o studenty, kteří danou látku bezchybně zvládají, nebo opisovali.



Graf č. 5.: Četnost výskytu správných odpovědí

Součástí dotazníku byla i možnost vyjádřit se k daným příkladům. Ze strany studentů se spíše jednalo o hodnocení obtížnosti příkladů než o jeho využití při výuce. Z jejich názorů mnohdy vyplývalo, že by uvítali seznámení s některými matematickými softwary. Někteří žáci znají již online webový server wolframalpha. Tento web můžeme přirovnat k anglické online verzi našeho programu Maxima. Často se setkávám (bylo tomu i v případě tohoto dotazníku), že podobné servery žáci mnohdy využívají pro kontrolu svých výsledků. Pro názornost si dovolím některé z nich ocitovat:

„Některé rovnice se mi zdály složitější, ale spočítat mi snad šly. Ráda bych věděla, jak algebraické programy fungují.“

„Řešení by bylo snazší, pokud by bylo výuce učiva věnováno více pozornosti. O využití softwaru jsem uvažovala, neumím.“

„Něco bylo jednoduché, něco složitější, uvažovala jsem o využití volně dostupného matematického softwaru.“

„Složité zapisování některých hodnot, ano, pro kontrolu wolframalpha.“

„Obtížné, ano uvažoval jsem o použití matematického softwaru.“

Z pedagogického hlediska jsou však zajímavější názory čtyř zúčastněných pedagogů, kteří mají letitou praxi (více než 20 let). Ti poukazují spíše na využívání početního řešení úloh. Jejich snahou je motivovat a nutit žáky využívat již naučené metody a postupy. Používání různých programů ale zcela nezavrhují. Poukazují na výhody při řešení úloh pro lidi, kteří se nezabývají matematikou, nebo pro kontrolu získaných výsledků. Opět uvedu citace z dotazníku od učitelů:

„Rovnice byly vhodné na použití Hornerova schématu nebo substituce. Daly se vypočítat bez matematického software, ale dokážu si představit, že lidé nezabývající se matematikou (s potřebou řešit toto zadání), by bez užití matematického software byli bezradní. I žáci naučené metody zapomenou a berličku v podobě počítačového postupu by uvítali.“

„Protože takové rovnice na SŠ neřešíme, šly vypočítat pomocí středoškolských dovedností (tj. rozkladem na součin). Je však pravda, že ne vždy to bylo hned vidět. Pokud bych chtěla žáky naučit řešit takovéto rovnice, pak se program dá využít snad jen ke kontrole výsledků. Ale je-li vyřešení dané rovnice jen součástí většího úkolu (na VŠ, v technické práci), pak je zjednodušení a zrychlení určitě na místě. Volně šiřitelný software jsem využila pro kontrolu právnosti svých výsledků“

„Nejprve jsem si myslela, že vypočítám s užitím řešení kvadratických rovnic jen 3 a 5 příklad. Ostatní byly pro mne výzvou. A když jsem 4. rovnici zvládla vytýkáním a úpravou, pustila jsem se do dalších. Nakonec jsem řešila 1. a 2. Rovnici, neboť úpravy se hned nenabízely. Příklady nebyly těžké, lze je využít na gymnáziu. Pořadí bylo dobře zvoleno, pokud mělo odradit od klasického řešení. Uvítám seznámení s programem MAXIMA. Setkala jsem se s aplikací pro tablety, kdy stačí rovnici vyfotit a výsledek mi tablet spočítá.“

Je tedy patrné, že učitelé v dnešní době sice mají přehled o určitých matematických softwarech, ale jedná se vesměs o ty nejzákladnější. Toto vede k minimálnímu využívání moderních technologií ve výuce. Nemůžeme říct, že vše lze nahradit počítačovými technologiemi. V současnosti již existuje celá řada programů. Od základních početních, přes konstrukční geometrické až po složité programy, které nám umožní spočítat i velmi obtížné úlohy.

Závěr

Cílem této práce bylo poskytnout čtenáři informace o počítačovém softwaru MAXIMA CAS. Práce je koncipována jako stručný manuál, pomocí kterého bude dotyčný schopen ovládat a zadávat potřebné příkazy k řešení různých úloh. Celková práce je rozčleněna do čtyř obsáhlých kapitol.

V první kapitole jsme se seznámili s možností stažení a instalace programu v prostředí operačního systému Microsoft a Android. Na tento úvod navázal popis základních pracovních prostředí, se kterými se uživatel může setkat, a stručně jsme si je charakterizovali. Následně jsme navázali návodem jak zadávat jednotlivé příkazy a způsob zadávání.

V druhé kapitole jsme probrali příkazy spojené s řešením rovnic, nebo jejich soustav.

V třetí kapitole jsme řešili algebraicky rovnice 3. a 4. řádu. Vysvětlili jsme si používání Hornerova schématu, rozklad na součin a Cardanovy vzorce.

Součástí poslední kapitoly je výzkum a jeho vyhodnocení. Tento průzkum byl zaměřený na algebraické řešení rovnic vyššího řádu (3. a 4.). Vyhodnocování pojednávalo o celkovém počtu správných řešení jednotlivých příkladů, současně i úspěchy jednotlivých dotazovaných a jejich názor na matematické softwary.

Seznam použitých zdrojů

1) BLAŽEK, J. a kol. Algebra a teoretická aritmetika 2. Praha: SPN 1985.

2) BUŠA, J. Maxima: open source systém počítačovej algebry. Košice: Technická univerzita, 2005. Edícia vysokoškolských učebníc. ISBN 8080736405. Dostupné z: http://people.tuke.sk/jan.busa/kega/maxima/maxima_brozura.pdf

3) *Graphics with MAXIMA* [online]. 2011, , 34 [cit. 2016-04-5]. Dostupné z: http://www.austromath.at/daten/maxima/zusatz/Graphics_with_Maxima.pdf

4) Maxima Manual. *Maxima Manual* [online]. 2008 [cit. 2016-02-12]. Dostupné z: http://www.johnlapeyre.com/qinf/qinf_html/maxima_html/maxima.html#SEC_Top

5) SCHWARZ, Štefan. *Základy náuky o riešení rovníc*. 2. dopl. vyd. Bratislava: Vydavateľstvo Slovenskej akadémie vied, 1968.

6) WOOLLETT, L., E. Maxima by Example: Chapter 1: Getting Started. Dostupné na http://www.isys.uni-stuttgart.de/lehre/systemdynamik/fls/Maxima/Woollett_mbe1start.pdf

7) WOOLLETT, Edwin L. *Maxima by Example* [online]. [cit. 2016-03-03]. Dostupné z: http://web.csulb.edu/~woollett/#mbe

Seznam obrázků

Seznam grafů

Seznam tabulek

ANOTACE

Jméno a příjmení:	Bc. et. Bc. Libor Jeřábek
Katedra:	Matematiky
Vedoucí práce:	doc. RNDr. Tomáš Zdráhal, CSc.
Rok obhajoby:	2016

Název práce:	Řešení rovnic v systému počítačové algebry Maxima
Název v angličtině:	Solving Equations in Computer Algebra System Maxima
Anotace práce:	Cílem této práce je vytvořit manuál, pomocí kterého bude čtenář schopen zacházet s algebraickým programem Maxima
Klíčová slova:	Matematika, Maxima, rovnice, rovnice 3. řádu, rovnice 4. řádu
Anotace v angličtině:	The aim of this work is to create a manual by which the reader will be able to treat the algebraic program Maxima
Klíčová slova v angličtině:	Math, Maxima, equation, equation 3. procedure, equation 4. order
Přílohy vázané v práci:	CD s elektronickou podobou bakalářské práce
Rozsah práce:	64 stran
Jazyk práce:	Český jazyk