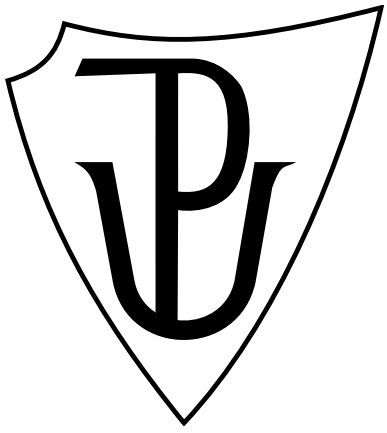
UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA ALGEBRY A GEOMETRIE

BAKALÁŘSKÁ PRÁCE

Řešení algebraických rovnic s programem MAXIMA



Vedoucí bakalářské práce:
Doc. RNDr. Petr Emanovský, Ph.D.
2014

Vypracoval:
Vojtěch Šoupal
M-VT, 3.ročník

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. RNDr. Petra Emanovského, Ph.D. a uvedl jsem veškerou použitou literaturu.

V Olomouci dne 24. 6. 2014

.....

Vojtěch Šoupal

Poděkování

Mé poděkování patří panu Doc. RNDr. Petru Emanovskému, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Bibliografická identifikace

Jméno a příjmení autora	Vojtěch Šoupal
Název práce	Řešení algebraických rovnic s programem MAXIMA
Typ práce	Bakalářská
Pracoviště	Katedra algebry a geometrie
Vedoucí práce	Doc. RNDr. Petr Emanovský, Ph.D.
Rok obhajoby	2014
Abstrakt	Bakalářská práce seznamuje čtenáře s programem Maxima a provádí ho programem za účelem řešení algebraických rovnic. Text je rozdělen do šesti kapitol. První dvě kapitoly se zaměřují na instalaci a základní práci v programu. Ve třetí kapitole čtenář získá znalosti pro práci s polynomy. Čtvrtá a pátá kapitola je věnována řešení algebraických rovnic. V poslední kapitole si čtenář může ověřit své znalosti v kapitole řešených příkladů.
Klíčová slova	MAXIMA, polynomy, algebraické rovnice, řešení algebraických rovnic
Počet stran	63
Počet příloh	0
Jazyk	český

Bibliographical identification

Autor's first name and surname	Vojtěch Šoupal
Title	Solving algebraic equations with the program MAXIMA
Type of thesis	Bachelor
Department	Department of Algebra and Geometry
Supervisor	Doc. RNDr. Petr Emanovský, Ph.D.
The year of presentation	2014
Abstract	The bachelor thesis instruct readers in usage of programme Maxima and guide him through the programme for the purpose of solving algebraic equations. The thesis has six chapters. The first two are focused on installation and its basic work with programme. In the third chapter the readers get knowledge how to work with polynomials. The fourth and the fifth chapter is concentrated on algebraic equations. In the last chapter of thesis reader can verify his knowledge of the usage with resolved examples.
Keywords	MAXIMA, polynomials, algebraic equations, solving algebraic equations
Number of pages	63
Number of appendices	0
Language	czech

Obsah

Seznam tabulek

Použité značení

Symbol	**Význam**
$\mathbb{N}$	Množina přirozených čísel
$\mathbb{R}$	Množina reálných čísel
$\mathbb{C}$	Množina komplexních čísel
$\mathbb{T}$	Číselné těleso
$\mathbb{T}'$	Nadtěleso číselného tělesa $\mathbb{T}$
$\mathbb{T}[x]$	Obor integrity polynomů jedné neurčité x nad $\mathbb{T}$
$f(x), g(x), h(x)$	Prvky oboru integrity $\mathbb{T}[x]$
$stf(x)$	Stupeň polynomu $f(x)$
$\mid$	Relace býti dělitelem
$f'(x)$	Derivace polynomu $f(x)$
$f^{(k)}(x)$	k-tá derivace polynomu $f(x)$
Δ	Diskriminant kvadratické rezolventy kubické rovnice
$rem(f(x), g(x))$	Zbytek po dělení polynomu $f(x)$ polynomem $g(x)$
$W(c)$	Počet znaménkových změn ve Sturmově posloupnosti v bode c

Úvod

Cílem této bakalářské práce je čtenáře seznámit s programem počítačové algebry Maxima, zpracovat vybrané partie algebry a provést čtenáře řešením jejich příkladů za pomoci programu Maxima.

Program maxima se zabývá manipulací symbolických a numerických výrazů. Tyto programy řadíme do tzv. systémů počítačové algebry (označované pod zkratkou CAS z anglického: Computer algebra system). Existují jak komerční, tak volně šiřitelné programy. Mezi komerčními zmíníme programy Maple, Matlab, Mathematica. Mezi volně šiřitelnými jsou to například GAP, Macaulay2, SINGULAR a zejména Maxima. Jako jeden z prvních CAS systémů byl program Macsyma (MAC's SYmbolic Manipulator), který byl vyvinut v roce 1968 v MIT (Massachusetts Institute of Technology).

Maxima byla roku 1982 vyvinuta profesorem Williamem F. Shelterem z Univerzity Texas jako verze programu Macsyma. Program je napsán ve funkcionálním programovacím jazyce LISP, přesněji v jeho dialektu Common Lisp, se kterým se autor tohoto textu seznámil v průběhu svého studia na této fakultě. Roku 1998 se profesoru Shelterovi podařilo získat povolení na zveřejnění zdrojového kódu DOE Macsyma od Department of Energy pod licencí GNU General Public Licence (GNU GPL). Díky této licenci se program stal volně šiřitelným stejně jako všechny jeho odvozená díla. Dva roky poté inicializoval na SourceForge samotný projekt MAXIMA, o jehož údržbu se staral až do své smrti v roce 2001.

Program se stal natolik populární, že se kolem něj vytvořila silná komunita, která se stará o vývoj programu dodnes. Poslední verzí programu na operačním systému Windows je Maxima 5.31.2 (2013-10-08), s kterou budeme pracovat i v našem textu. Maximu lze spustit na více operačních systémech, kromě Windows se jedná také o Linux a MacOS X. Jak již bylo řečeno, Maxima zvládá manipulaci se symbolickými a numerickými výrazy, včetně derivace, integrace, Taylorových polynomů, řešení obecných diferenciálních rovnic a řešení soustavy lineárních rovnic. Zvládá práci s množinami, vektory, maticemi a polynomy, což nás v tomto textu bude zajímat nejvíce.

1 Instalace a první prohlídka programu

1.1 Instalace programu Maxima

V této části textu si projdeme samotnou instalaci programu. Veškeré instalační soubory si můžeme volně stáhnout na stránce `http://sourceforge.net/projects/maxima/files/`, kde si můžeme vybrat z verzí jak pro operační systémy Windows, tak i pro Linux.

1.1.1 Instalace Windows

Po stažení poslední verze programu, tedy verze 5.31.2, spustíme instalační soubor. V průběhu instalace budeme vyzváni k souhlasu s licenčními podmínkami. Po odsouhlasení podmínek vybereme složku, do které bude program nainstalován. V dalším kroku máme možnost vybrat komponenty, které budou nainstalovány. My vybereme možnost *Full instalation*, a tím nainstalujeme vše.

V dalším okně můžeme zvolit pojmenování složky nebo zaškrtnout její nevytvoření v nabídce start. Po stisknutí tlačítka Next se objeví předposlední dialogové okno, kde nastavíme, zda chceme vytvořit zástupce na ploše či nikoliv. V dalším kroku si prohlédneme kompletní nastavení instalace a stisknutím tlačítka *Install* započne samotná instalace. Po přečtení *readme* souboru klikneme na tlačítko *Next* a instalaci ukončíme stisknutím *Finish*.

1.1.2 Instalace Linux

Nyní ukážeme jak program nainstalovat v linuxové distribuci Ubuntu (verze 10.04.1). Jednou z možností je spustit aplikaci *Centrum softwaru pro Ubuntu*, kterou najdeme v záložce *Aplikace*. Poté ve spuštěné aplikaci vyhledáme program wxMaxima a kliknutím na *Nainstalovat* spustíme instalaci.

Druhou možností je instalovat přes *Terminál*. Ten najdeme v záložce *Aplikace* > *Příslušenství*. Zde stačí napsat příkaz „*sudo apt-get install wxmaxima*“. Po stisknutí klávesy enter budeme vyzvání k ověření hesla správce, bez něhož by nešlo aplikaci nainstalovat. Po úspěšné autorizaci se spustí instalace.

1.2 První prohlídka programu

V této podkapitole provedeme čtenáře spuštěním programu a základní prací s příkazovým řádkem, soubory a nápovědou. Zmíníme se také o tom, jak program ukončit.

1.2.1 Spuštění programu

Program můžeme spustit jako konzolovou aplikaci XMaxima nebo s podporou grafického uživatelského rozhraní wxMaxima. V dalším textu budeme pracovat s variantou wxMaxima, která je z pohledu uživatele mnohem pohodlnější než XMaxima. Ovšem mějme na paměti, že se jedná stále o stejný program,
a tedy v obou verzích můžeme využívat stejné funkce. V systémech Windows stačí poklepat na ikonu wxMaxima. V případě Ubuntu najdeme program wxMaxima v záložce *Aplikace* > *Věda* > *wxMaxima* nebo program spustíme v příkazovém řádku příkazem *„wxmaxima“*.

Po zapnutí si můžeme všimnout, že je aplikace vybavena bohatým menu, které nám kromě obvyklých položek jako je vytvoření, otevření, uložení, exportování souboru a vypnutí programu nabízí mnoho dalších funkcí. Díky těmto předdefinovaným funkcím je práce mnohem pohodlnější a efektivnější. Není třeba si pamatovat velké množství příkazů pro naši budoucí práci s programem. Ovšem zkušenější uživatel postupem času nejspíše upřednostní přímo psaní příkazů do příkazového řádku před hledáním potřebné položky v menu.

1.2.2 Příkazový řádek

Abychom mohli začít s programem Maxima pracovat, musíme se nejprve seznámit s pravidly pro zapisování výrazů do příkazového řádku. Každý výraz musí být ukončen středníkem. Samotné vyhodnocení řádku proběhne po stisknutí klávesy *Shift* + *Enter*. Je povoleno do jednoho řádku vepsat více výrazů, ovšem každý z nich musí být ukončen středníkem. Stejně tak je možné vyhodnotit zároveň více řádků, které můžeme psát pomocí klávesy *Enter*. Pokud je uživatel zvyklý na opačné chování kláves *Enter* a *Shift* + *Enter*, může si jejich funkč-

nost v programu zaměnit. V menu vybereme sekci *Editovat* a vybereme položku *Nastavení*. Zde zaškrtneme možnost „Enter vyhodnocuje výraz“. Po stisknutí tlačítka *OK* budou funkce kláves *Shift + Enter* a *Enter* prohozeny. V ukázce provedeme vyhodnocení dvou přirozených čísel v jednořádkovém i víceřádkovém příkazu.

```
(%i1)  1; 20;

(%o1)  1
(%o2)  20

(%i3)  1;
       20;

(%o3)  1
(%o4)  20
```

Všechny vstupy (z anglického input) jsou v příkazovém řádku označeny jako (%ix), výstupy (z anglického output) jako (%ox), kde x je přirozené číslo. Snadno se tedy můžeme odkazovat na již vyhodnocené výrazy.

Pro potlačení zobrazení výsledku vyhodnocení slouží znak $, který zapisujeme na konci příkazu místo středníku. I když výsledek nezobrazíme, máme možnost s ním dále pracovat. Vyhodnocení výrazu bez zobrazení výsledku a následné použití tohoto výrazu si můžeme demonstrovat následující ukázkou.

```
(%i5)  6+21$

(%i6)  %o5/3;

(%o6)  9
```

Nechceme-li výraz vypočítat, ale jen ho vypsat do výstupu, vložíme před výraz apostrof '. V příkladu používáme funkci *diff()*, která slouží k derivování. O tom, s jakými argumenty ji lze volat, se dozvíme v podkapitole 3.5.

```
(%i7)  'diff(x^3 + 3*x + 2, x, 1);
```

$$(\%o7) \quad \frac{d}{d\,x}\left(x^3 + 3\,x + 2\right)$$

Pokročilejší uživatel jistě ocení našeptávač, který po stisknutí kláves *Ctrl+K* automaticky doplní rozepsaný název funkce nebo nabídne možnosti všech funkcí obsahující rozepsaný řetězec znaků.

1.2.3 Uložení, načtení, export a tisk souboru

Během práce můžeme kdykoli svou rozdělanou práci uložit a později znovu načíst. Zmíněné možnosti nalezneme v menu pod položkou *Soubor* nebo použijeme standardní klávesové zkratky *Ctrl+S* pro uložení souboru a *Ctrl+O* pro otevření již uloženého souboru. Soubory jsou ukládány ve formátu **.wxm* jakožto zkratky *wxMaxima dokument*. Program podporuje export do jazyka HTML a také do jazyka TEX, což je i pro tvorbu tohoto textu velmi užitečná funkce programu. Tisk najdeme v menu Soubor a položce *Tisk*. Případně stačí použít zažitou klávesovou zkratku *Ctrl+P*.

1.2.4 Nápověda

Je bezesporu neoddělitelnou součástí programu. Nápověda obsahuje rozsáhlou dokumentaci všech funkcí programu. Jedinou nevýhodou stále zůstává absence české verze tohoto manuálu. Proto může být pro začátečníka obtížné se v samotné nápovědě daných funkcí popsaných v odbornějším anglickém jazyce orientovat. Nápovědu spustíme klávesou *F1* nebo v menu *Nápověda* položka *Nápověda programu Maxima*. Nemusíme ovšem hledat vždy jen v nápovědě. Přímo v příkazovém řádku si můžeme pomoci funkcemi *describe()*, *example()*, *apropos()*.

Funkce *describe()* nám vypíše popis dané funkce, jejíž název vložíme jako argument do závorek. Na výstupu vrací hodnotu *true*, jestliže najde nápovědu zadané funkce v argumentu. Pokud ji nenalezne, vrátí hodnotu *false*.

Nyní jsme schopni zjistit, jaký význam daná funkce má. Ovšem příklady po-

užití jsou mnohokrát užitečnější než stovky slov o dané funkci. Pro tyto situace nám slouží funkce *example()*. Již z názvu můžeme odhadnout, že funkce *example* bude vypisovat příklady použití funkce zadané v argumentu. Program dané příklady nevypisuje pouze jako text, ale sám je zadává a vyhodnocuje, takže se na ně můžeme odkazovat.

Pro hledání funkcí obsahujících daný řetězec použijeme funkci *apropos()*, které jako argument předáme řetězec znaků. Na výstupu poté najdeme seznam funkcí splňujících tuto podmínku. Jelikož se jedná o řetězec znaků, musíme argument vepsat do uvozovek. Chceme-li o daných funkcích zjistit více, použijeme již pro nás známé funkce *describe()* nebo *example()*.

1.2.5 Ukončení programu

Pro ukončení programu wxMaxima zvolíme v menu *Soubor* a možnost *Ukončit*. Aplikace se dá ukončit také klávesovou zkratkou *Ctrl+Q*. Pokud svoji dosavadní práci nemáme uloženou, budeme vyzváni, zda ji chceme uložit či nikoliv.

2 Základní uživatelská výbava

V této kapitole se seznámíme se základní výbavou pro práci v programu wxMaxima. Pro lepší pochopení budeme vždy udávat i názorné příklady.

2.1 Standardní operace

Program používá klasickou infixovou notaci, tedy takovou, kdy se operátor píše mezi dva operandy. Takže jistě nebude žádný problém s klasickými operacemi, které zastupují standardní znaky +, -, * a /. N-tou mocninu čísla x můžeme vyjádřit za pomoci stříšky takto: x^n. Ekvivalentně ji také lze vyjádřit znakem **, tedy x**n. Odmocniny můžeme zapisovat formou mocniny. Pro druhou odmocninu můžeme zvolit funkci *sqrt()*, která jako argument bere číslo, které hodláme odmocnit. Uživatel musí dbát na priority jednotlivých operací a uzávorkování výrazů. V příkladu si ukážeme, jak je uzávorkování důležité.

```
(%i1)  x**3+4;
```

(%o1) x^3+4

```
(%i2)  x**(3+4);
```

(%o2) x^7

2.2 Zápis matematických konstant

Jedná se o tyto konstanty: Eulerovo číslo, imaginární jednotka, Ludolfovo číslo, kladné nekonečno, záporné nekonečno, logická pravda a nepravda. Pro jednoduchost jistě postačí přímá demonstrace zápisu těchto konstant v příkazovém řádku přesně dle již zmíněného pořadí.

```
(%i10) %e;
       %i;
       %pi;
       inf;
       minf;
       true;
       false;
```

(%o10) e

(%o11) i

(%o12) π

(%o13) ∞

(%o14) $-\infty$

(%o15) $true$

(%o16) $false$

2.3 Přiřazení hodnot, výrazů a funkcí

U hodnot a výrazů, které mohou obsahovat i další proměnné, realizujeme přiřazení pomocí dvojtečky. Pro vytvoření funkce jedné nebo více proměnných využíváme kombinace znaků „:=“. Do závorek za jméno funkce vložíme názvy proměnných oddělené čárkou. Vytvoříme tedy výraz f a funkci $g(x)$.

```
(%i17) f: x^2 + 3;
```

(%o17) $x^2 + 3$

```
(%i18) g(x) := x^2 + 3;
```

(%o18) $\mathrm{g}\,(x) := x^2 + 3$

Pro získání hodnoty funkce $g(x)$ v bodě tři použijeme následující příkaz.

```
(%i19) g(3);
```

(%o19) 12

Pokud bychom chtěli vyhodnotit příkaz $f(3)$;, skončilo by vyhodnocení chybou, jelikož f není funkce, a tedy neočekává žádný argument. Přiřazené hodnoty zrušíme příkazem *kill()*, který jako argument bere název proměnné, u které se má přiřazení zrušit. Všechny proměnné zrušíme příkazem *kill(all)*.

2.4 Přesnost a zobrazení reálných čísel

Pro tyto účely si vystačíme s položkou *Numerické výpočty* v hlavním menu. Dále se budeme odkazovat přímo na položky tohoto podmenu. Pokud chceme místo symbolického tvaru čísla používat tvar numerický, použijeme položku v menu *Přepnout numerický výstup*. Pokud chceme převést na numerický tvar pouze daný výraz, použijeme možnost *Převést na float (plovoucí řádová čárka)* (%i21).

```
(%i20) %pi;
```

(%o20) π

```
(%i21) float(%), numer;
```

(%o21) 3.141592653589793

Nestačí-li nám standardní počet šestnácti zobrazených cifer, můžeme pomocí *Nastavit přesnost...* počet cifer zvýšit či snížit (%i23).

```
(%i23) fpprec: 50;
```

(%o23) 50

Změna se ovšem projeví pouze pokud zvolíme v menu možnost *Přesnost bigfloat* (%i24).

```
(%i24) bfloat(%pi);
```

(%o24) 3.14159265358979323846264338327950288419716939937510

Ve výstupu (%o24) nám písmeno „b“ odděluje desetinná místa od exponentu. Výstup bychom tedy mohli napsat také takto:

$$3.14159265358979323846264338327950288419716939937510 \cdot 10^{0}$$

2.5 Komplexní čísla

Komplexní čísla zapisujeme v algebraickém tvaru, tedy ve tvaru $a + bi$, kde a, $b \in \mathbb{R}$ a i je imaginární jednotkou, kterou zapisujeme pomocí znaku %i. Zavedeme si tedy dvě komplexní čísla $z1, z2$.

```
(%i26) z1: 4+3*%i;
       z2: 1+5*%i;
```

(%o26) $3\,i + 4$

(%o27) $5\,i + 1$

S komplexními čísly lze provádět standardní operace. Získáme tedy například podíl zavedených komplexních čísel.

```
(%i28) z1/z2;
```

(%o28) $$\frac{3\,i+4}{5\,i+1}$$

Pro získání základního tvaru použijeme funkci *rectform()*. Chceme-li získat reálnou část komplexního čísla, použijeme funkci *realpart()*. V případě imaginární části použijeme funkci *imagpart()*. Převedeme tedy komplexní číslo z výstupu (%o28) do základního tvaru a poté získáme jeho reálnou část.

```
(%i29) rectform(%);
```

(%o29) $$\frac{19}{26} - \frac{17\,i}{26}$$

```
(%i30) realpart(%);
```

$$(\%o30)\ \frac{19}{26}$$

Absolutní hodnotu komplexního čísla nám vrátí funkce *abs()*. Argument komplexního čísla potom vrací funkce *carg()*. Chceme-li komplexní číslo zobrazit v exponenciálním tvaru, použijeme funkci *polarform()*.

```
(%i31) abs(z1);
```

$$(\%o31)\ 5$$

```
(%i32) carg(z1);
```

$$(\%o32)\ \operatorname{atan}\left(\frac{3}{4}\right)$$

```
(%i33) polarform(z1);
```

$$(\%o33)\ 5\,e^{i\operatorname{atan}\left(\frac{3}{4}\right)}$$

Touto kapitolou jsme získali základní úroveň pro práci v programu Maxima. Pro naše účely je tato úroveň dostačující a můžeme přejít k další kapitole této práce. Pokud budeme v průběhu práce využívat pro nás zatím neznámou funkci, vždy její popis a použití uvedeme.

3 Práce s polynomy v programu Maxima

V této kapitole se zaměříme na práci s polynomy v programu Maxima. Seznámíme se s možnostmi zápisu polynomů, základními operacemi, funkcemi pro práci s polynomy. Ukážeme si výpočet hodnoty polynomu v daném bodě, dělitelnost, největší společný dělitel, derivaci a hledání kořenů polynomu. U důležitých pojmů uvedeme vždy i definici pro připomenutí. Celá teorie je přehledně zpracována ve skriptu [3].

3.1 Zápis polynomů

Ze všeho nejdříve si ujasníme, jak je polynom definován.

Definice 3.1.1. Nechť $\mathbb{T}$ je číselné těleso,$n \in \mathbb{N} \cup \{0\}, a_0, a_1, \ldots, a_n \in \mathbb{T}, x \notin \mathbb{T}$ pak výraz

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

nazveme polynomem jedné neurčité x nad $\mathbb{T}$, kde prvek $a_i x^i$ se nazývá člen i-tého stupně a prvky $a_0, a_1, \ldots, a_n \in \mathbb{T}$ jsou koeficienty $f(x)$. $\mathbb{T}[x]$ označíme množinu všech polynomů jedné neurčité x nad $\mathbb{T}$. Výraz $f(x)$ v tomto tvaru budeme považovat za tzv. *kanonický tvar* polynomu $f(x)$.

Při zápisu polynomu v programu Maxima máme dvě možnosti. S první jsme se již seznámili v podkapitole 2.3, kde jsme za pomoci řetězce „:=“ definovali funkce. Domluvme se, že budeme vždy polynom zapisovat v kanonickém tvaru. Tedy vždy budeme uvádět polynom jako sumu jednotlivých členů (v roznásobeném tvaru) a jednotlivé členy budeme zapisovat sestupně podle výše mocniny. Vytvoříme tedy polynom $f(x)$ takto:

```
(%i1)  f(x) := 2*x^4 - 3*x^3 + 4*x^2 - 5*x + 6;
```

(%o1) $\mathrm{f}\,(x) := 2\,x^4 - 3\,x^3 + 4\,x^2 + (-5)\,x + 6$

Ekvivalentním zápisem je použití funkce *define()*, které jako vstupní argumenty zadáme název polynomu s neurčitou a výraz, který se danému názvu přiřadí. Nyní tedy vytvoříme polynom $g(x)$.

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

(%o2) $\mathrm{g}\,(x) := x^2 - 3\,x + 1$

U každého polynomu, s výjimkou nulového ($f(x) = 0$), můžeme určit stupeň polynomu. Tento pojem nám objasňuje následující definice.

Definice 3.1.2. Nechť $f(x) \in \mathbb{T}[x]$. Pak stupněm polynomu $f(x)$ rozumíme číslo $n \in \mathbb{N} \cup 0$ pro které platí:

$$(a_n \neq 0) \wedge (\forall i \in \mathbb{N}, i > n, a_i = 0)$$

Stupeň polynomu $f(x)$ označujeme symbolem: $stf(x)$.

Předpokládáme-li, že máme polynom v kanonickém tvaru. Pak pro zjištění stupně polynomu slouží funkce *hipow()*. Funkce přijímá dva argumenty. Prvním je polynom a druhým je neurčitá. Zjistíme tedy stupeň polynomu $f(x)$.

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

(%o3) 4

Před použitím je potřeba mít polynom v kanonickém tvaru, jinak bychom mohli dostat mylný výsledek. Pokusíme se zjistit stupeň polynomu $f(x)$ umocněného na druhou.

```
(%i4)  hipow(f(x)^2,x);
```

(%o4) 4

Jak je vidět, dostali jsme opět stejný stupeň, i když by tentokrát stupeň měl mít dvojnásobnou hodnotu. Proto budeme vždy funkce *hipow() a lopow()*

používat na polynomy v kanonickém tvaru.

Při naší práci může nastat situace, kdy budeme potřebovat zjistit koeficienty u jednotlivých členů polynomu. K tomu nám poslouží funkce *coeff()*, případně *bothcoeff()*. Obě jako argument berou polynom a neurčitou s mocninou, u které se má koeficient hledat. Funkce *coeff()* vrací samotný koeficient. V ukázkách obou funkcí budeme hledat koeficient u členu s neurčitou x^3.

```
(%i5)  f(x);
```

$$(\%o5)\quad 2\,x^4 - 3\,x^3 + 4\,x^2 - 5\,x + 6$$

```
(%i6)  coeff(f(x),x^3);
```

$$(\%o6)\quad -3$$

Funkce *bothcoeff()* vrací seznam, ve kterém je jako první uveden hledaný koeficient a jako druhý polynom bez členu se zadanou mocninou.

```
(%i7)  bothcoeff(f(x),x^3);
```

$$(\%o7)\quad [-3, 2\,x^4 + 4\,x^2 - 5\,x + 6]$$

Nyní se pokusíme získat vedoucí člen polynomu. Tedy prvek $a_n x^n$ polynomu $f(x)$ stupně n. Nejprve nalezneme stupeň polynomu n. Pomocí něj funkcí *coeff()* získáme hodnotu koeficientu vedoucího členu. Nakonec vynásobíme hodnotu koeficientu a neurčité v příslušné mocnině n.

```
(%i8)  n:hipow(f(x),x);
       coeff(f(x),x^n)*x^n;
```

$$(\%o8)\quad 4$$

$$(\%o9)\quad 2\,x^4$$

Dále se podíváme, jak je definována hodnota polynomu v bodě.

Definice 3.1.3. Nechť $f(x) \in \mathbb{T}[x]$, $f(x) = \sum_{i=0}^{n} a_i x^i$, $c \in \mathbb{T}$ pak hodnotou

polynomu $f(x)$ v bodě c rozumíme $\sum_{i=0}^{n} a_i c^i$.

Pro získání hodnoty polynomu v daném bodě c stačí zadat příkaz $f(c)$. Také můžeme použít funkci *subst()*, která jako argumenty nejprve bere hodnotu, jakou budeme substituovat, poté co budeme substituovat a jako poslední argument udáváme, kde budeme substituovat. Oba způsoby nám ukáže následující příklad, ve kterém budeme zjišťovat hodnotu polynomu $f(x)$ v bodě c.

```
(%i10) f(c);
```

(%o10) $2\,c^4 - 3\,c^3 + 4\,c^2 - 5\,c + 6$

```
(%i11) subst(c,x,f(x));
```

(%o11) $2\,c^4 - 3\,c^3 + 4\,c^2 - 5\,c + 6$

První možnost je zcela jistě přímočařejší a pohodlnější.

3.2 Standardní operace

Připomeneme si, jak je definován součet a součin polynomů na $\mathbb{T}[x]$.

Definice 3.2.1. Nechť $f(x), g(x) \in \mathbb{T}[x]$, kde $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0 x^0$, $g(x) = b_m x^m + b_{m-1}x^{m-1} + \ldots + b_0 x^0$, kde $n, m \in \mathbb{N} \cup \{0\}$. Na množině $\mathbb{T}[x]$ definujeme součet $f(x) + g(x)$ a součin $f(x) \cdot g(x)$ takto:

a) $f(x)+g(x) = \sum_{i=0}^{max(m,n)} (a_i + b_i) \cdot x^i$, *kde pro* $i > n$ (*resp.* $i > m$) *položíme* $a_i = 0$

(*resp.* $b_i = 0$).

b) $f(x) \cdot g(x) = \sum_{i=0}^{n+m} c_i \cdot x^i$, *kde* $c_i = \sum_{h=0}^{i} a_{i-h} \cdot b_h$.

Se standardními operacemi v programu Maxima jsme se setkali v podkapitole 2.1 a při práci s polynomy tomu není jinak. Můžeme tedy provádět klasické operace tak, jak jsme zvyklí u číselných těles.

```
(%i12) f(x)+g(x);
       f(x)-g(x);
       f(x)*g(x);
       f(x)/g(x);
       f(x)^3;
```

$$(\%o12)\quad 2\,x^4-3\,x^3+5\,x^2-8\,x+7$$

$$(\%o13)\quad 2\,x^4-3\,x^3+3\,x^2-2\,x+5$$

$$(\%o14)\quad \left(x^2-3\,x+1\right)\,\left(2\,x^4-3\,x^3+4\,x^2-5\,x+6\right)$$

$$(\%o15)\quad \frac{2\,x^4-3\,x^3+4\,x^2-5\,x+6}{x^2-3\,x+1}$$

$$(\%o16)\quad \left(2\,x^4-3\,x^3+4\,x^2-5\,x+6\right)^3$$

V ukázce jsme postupně získali výsledky součtu (%o12), rozdílu (%o13), součinu (%o14), podílu (%o15) polynomů $f(x)$ a $g(x)$ a umocnili jsme polynom $f(x)$ (%o16).

Nyní se zaměříme na to, jak je to s rovností dvou polynomů.

Definice 3.2.2. Nechť $f(x), g(x) \in \mathbb{T}[x]$, kde $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0 x^0$, $g(x) = b_m x^m + b_{m-1}x^{m-1} + \ldots + b_0 x^0$, kde $n, m \in \mathbb{N} \cup \{0\}$. Řekneme, že $f(x) = g(x)$ právě když platí:

$$[stf(x) = n = m = stg(x)] \ \wedge [\forall i \in \{0, 1, \ldots, n\}, a_i = b_i]$$

Rovnost dvou polynomů můžeme ověřit v programu maxima za pomoci příkazu *is()*, který vrací hodnotu *true* nebo $false$ v závislosti na vyhodnocení argumentu funkce. Vyzkoušíme tedy rovnost polynomů $f(x)$ a $g(x)$.

```
(%i17) is(f(x) = g(x));
```

$$(\%o17)\quad false$$

3.3 Rozklad polynomu

Polynomy nemusí být vždy v kanonickém tvaru. Mohou být ve tvaru součinu. Za jakých podmínek lze polynom přepsat do tvaru součinu si připomeneme v definici.

Definice 3.3.1. Nechť $f(x) \in \mathbb{T}[x]$, $stf(x) > 1$. Řekneme, že $f(x)$ je rozložitelný (reducibilní) nad $\mathbb{T}$, jestliže

$$\exists g(x), h(x) \in \mathbb{T}[x], stg(x) \geq 1, sth(x) \geq 1 \text{ tak, že } f(x) = g(x) \cdot h(x) \ .$$

V opačném případě je $f(x)$ nerozložitelný (ireducibilní) nad $\mathbb{T}$.

Jak jste si mohli všimnout ve výstupech (%o14), (%o15), (%o16), jsme dostali výsledný polynom v jiném než kanonickém tvaru. Chceme-li polynom zobrazit v kanonickém tvaru, použijeme funkci *expand()*. Funkce *expand()* svůj argument roznásobí a zobrazí ho v kanonickém tvaru.

```
(%i18) expand(%o16);
```

(%o18) $8\,x^{12} - 36\,x^{11} + 102\,x^{10} - 231\,x^9 + 456\,x^8 - 735\,x^7 + 1024\,x^6 - 1257\,x^5 + 1344\,x^4 - 1169\,x^3 + 882\,x^2 - 540\,x + 216$

Pro rozložení na součin (faktorizování) použijeme funkce *factor()*nebo *gfactor()*. Funkce *factor()* se snaží svůj argument rozložit ve tvaru součinu nad reálnými čísly. Tedy nedokáže rozložit polynom $h(x) = x^2 + 1$, což si ukážeme v následující ukázce.

```
(%i19) factor(%o18);
```

(%o19) $\left(2\,x^4 - 3\,x^3 + 4\,x^2 - 5\,x + 6\right)^3$

```
(%i20) h(x):= x^2 + 1;
```

(%o20) $\mathrm{h}\,(x) := x^2 + 1$

```
(%i21) factor(h(x));
```

(%o21) $x^2 + 1$

S tímto rozkladem si ovšem poradí funkce *gfactor()*, která je určena pro rozklad na součin nad komplexními čísly. Ověříme tedy, zda funkce *gfactor()* rozloží polynom $h(x)$.

```
(%i22) gfactor(h(x));
```

(%o22) $(x - i)\,(x + i)$

Další možností, jak zapsat polynom, je pomocí tzv. „Hornerova pravidla“. Uvedeme pouze větu, důkaz je proveden například v [3].

Věta 3.3.1 (Hornerovo schéma)**.** Nechť $\mathbb{T}$ je komutativní těleso a $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \in \mathbb{T}[x], stf(x) = n \geq 1, c \in \mathbb{T}$ a nechť $q(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}$ je částečný podíl a r je zbytek při dělení polynomu $f(x)$ dvojčlenem $(x - c)$. Pak platí: $b_{n-1} = a_n, b_{n-2} = a_{n-1} + cb_{n-1}, \ldots, b_0 = a_1 + cb_1, r = a_0 + cb_0$.

V programu Maxima k němu slouží funkce *horner()*. Argumenty jsou polynom a neurčitá. U polynomu s jednou neurčitou se neurčitá uvádět nemusí a stačí pouze jeden argument.

```
(%i23) f(x);
```

(%o23) $2\,x^4 - 3\,x^3 + 4\,x^2 - 5\,x + 6$

```
(%i24) horner(f(x));
```

(%o24) $x\,(x\,(x\,(2\,x - 3) + 4) - 5) + 6$

```
(%i25) horner(f(x),x);
```

(%o25) $x\,(x\,(x\,(2\,x - 3) + 4) - 5) + 6$

Funkce vrací reprezentaci polynomu $f(x)$ v závislosti na větě 3.3.1, tedy v závislosti na Hornerovu pravidlu. My si nyní ukážeme, že tento tvar opravdu vede na výpočet hodnoty polynomu $f(x)$ v bodě c. Máme tedy polynom obecně v tomto tvaru:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 x^0$$

Postupným roznásobením závorek bychom zjistili, že následující vyjádření polynomu $f(x)$ je s předešlým ekvivalentní.

$$f(x) = a_0 + x(a_1 + x(a_2 + \ldots x(a_{n-1} + a_n x)\ldots))$$

Pak tedy platí:

$$f(c) = a_0 + c(a_1 + c(a_2 + \ldots c(a_{n-1} + a_n c)\ldots))$$

Budeme-li nyní do této rovnice postupně dosazovat vztahy z věty 3.3.1, dostaneme následující tvary polynomu:

$f(c) = a_0 + c(a_1 + c(a_2 + \ldots c(a_{n-1} + b_n c)\ldots))$

$f(c) = a_0 + c(a_1 + c(a_2 + \ldots c(b_{n-1})\ldots))$

$\vdots$

$f(c) = a_0 + c(b_1)$

$f(c) = r$.

To je přesně v souladu s danou větou o Hornerově schématu. Bohužel z této reprezentace na první pohled nevidíme koeficienty $b_0, b_1, \ldots b_{n-1}$ částečného podílu $q(x)$ z věty 3.3.1. Pro získání zmíněných koeficientů čtenáře odkazujeme na podkapitolu 3.4.

Jako poslední se zmíníme o Taylorově rozvoji.

Definice 3.3.2. Nechť $f(x) \in \mathbb{T}[x], stf(x) = n \geq 1, d_0, d_1, \ldots, d_n \in \mathbb{T}, c \in \mathbb{T}$. Polynom

$$d_0 + d_1(x-c) + d_2(x-c)^2 + \ldots + d_n(x-c)^n$$

se nazývá Taylorův rozvoj polynomu $f(x)$ v bodě c, jestliže je roven polynomu $f(x)$.

V programu Maxima k němu náleží funkce *taylor()*. Argumenty jsou polynom, neurčitá, bod v jakém se má rozvoj provést a jako poslední argument se udává počet prvních členů, které se mají zobrazit. Provedeme tedy Taylorův rozvoj polynomu $f(x)$ ve dvou různých bodech.

```
(%i26) taylor(f(x),x,1,4);
       taylor(f(x),x,2,3);
```

$$(\%o26)/\mathrm{T}/ \quad 4 + 2\,(x-1) + 7\,(x-1)^2 + 5\,(x-1)^3 + 2\,(x-1)^4 + \ldots$$
$$(\%o27)/\mathrm{T}/ \quad 20 + 39\,(x-2) + 34\,(x-2)^2 + 13\,(x-2)^3 + \ldots$$

Znak /T/ uvedený za číslem výstupu značí, že se jedná o Taylorův rozvoj a ten tedy může na výstupu obsahovat jen několik svých prvních členů.

3.4 Dělení polynomů

V této sekci si přiblížíme, jak se dají dělit polynomy v programu Maxima.

Definice 3.4.1. Nechť $f(x)$, $g(x) \in \mathbb{T}[x]$. Řekneme, že $f(x)$ dělí $g(x)$ (resp. $g(x)$ je dělitelný $f(x)$, $g(x)$ je násobkem $f(x)$), jestliže $\exists h(x) \in \mathbb{T}[x]$ takový, že $g(x) = f(x) \cdot h(x)$. Tuto skutečnost značíme jako $f(x) \mid g(x)$, v opačném případě $f(x) \nmid g(x)$.

Snadno je vidět, že relace „|“ je reflexivní a tranzitivní. Každý polynom $f(x)$ je dělitelný jedničkou $(1 \mid f(x))$ a dělí nulový polynom $(f(x) \mid 0)$.

Pro dělení polynomů budeme používat funkci *divide()*, která jako vstupní argumenty bere dělenec, dělitel a jako třetí argument přijímá název neurčité, podle které se dělí. Pokud polynomy obsahují pouze jednu neurčitou, stačí uvést první dva argumenty. Funkce vrací seznam, ve kterém je první hodnotou částečný podíl a druhou je zbytek po dělení.

```
(%i28) divide(f(x),(x+1+%i));
```

$$(\%o28) \quad [x - i + 1, 0]$$

```
(%i29) divide(f(x), (x+1));
```

(%o29) $[x+1, 1]$

S výsledkem můžeme dál pracovat a přepsat ho jako součet částečného podílu a zbytku. Jelikož je výsledkem seznam, je potřeba se dostat na jednotlivé prvky seznamu. K tomu slouží hranaté závorky, do kterých se vepíše n-tý prvek ze seznamu. Přepsání výsledku může vypadat například takto.

```
(%i30) %o29[1] + %o29[2]/(x+1);
```

(%o30) $\dfrac{1}{x+1} + x + 1$

Nyní si ukážeme, jak hledat největší společný dělitel dvou polynomů.

Definice 3.4.2. Nechť $f_1(x), \ldots, f_n(x) \in \mathbb{T}[x]$. Polynom $D(x) \in T[x]$ nazveme největší společný dělitel $f_1(x), \ldots, f_n(x) \in \mathbb{T}[x]$, jestliže:
1) $[D(x) \mid f_1(x)] \wedge [D(x) \mid f_2(x)] \wedge \ldots \wedge [D(x) \mid f_n(x)]$
2) $\forall d(x) \in \mathbb{T}[x]$, jestliže $[d(x) \mid f_1(x)] \wedge [d(x) \mid f_2(x)] \wedge \ldots \wedge [d(x) \mid f_n(x)]$, pak $[d(x) \mid D(x)]$

Pokud je stupeň největšího společného dělitele polynomů $f_1(x), \ldots, f_n(x)$ alespoň jedna, pak říkáme, že polynomy jsou soudělné. V opačném případě jsou polynomy nesoudělné.

V programu k tomuto účelu slouží funkce *gcd()*, argumenty jsou dva nebo více polynomů a neurčitá těchto polynomů. U polynomů jedné neurčité opět nemusíme poslední argument uvádět. Funkce vrací hodnotu největšího společného dělitele. V příkladu si funkci vyzkoušíme.

```
(%i31) gcd((x^2 + 2*x + 1), (x+2)*(x+3));
       gcd((x^2 + 2*x + 1), (x+1)*(x+3));
       gcd((x^2 + 2*x + 1), (x+1)*(x+3), (x+1)*(x^3 + 3*x +4));
```

(%o31) 1

(%o32) $x+1$

(%o33) $x+1$

Obdobnou funkcí, která vypočítává největší společný dělitel, je funkce *ezgcd()*. Opět jí můžeme předat jako argumenty i více než dva polynomy. Funkce vrací seznam, ve kterém je první hodnotou největší společný dělitel a ostatními prvky jsou argumenty (tedy polynomy) vydělené největším společným dělitelem.

```
(%i34) ezgcd((x^2 + 2*x + 1), (x+1)*(x+3), (x+1)*(x^3 + 3*x +4));
```

(%o34) $[x+1, x+1, x+3, x^3+3\,x+4]$

Snadno vidíme, že pokud bychom prvním členem seznamu vynásobili ostatní, dostaneme původní polynomy. Tedy funkce pracuje přesně tak, jak jsme si uvedli.

3.5 Derivace polynomu

Definice 3.5.1. Nechť $f(x) \in \mathbb{T}[x], f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n, stf(x) = n \geq 1$. Pak derivací polynomu $f(x)$ rozumíme $f'(x) \in \mathbb{T}[x]$ a platí:

$$f'(x) = a_1 + 2a_2x + 3a_3x^2 + \ldots + na_nx^{n-1} = \sum_{i=1}^{n} ia_ix^{i-1}$$

Derivací polynomu stupně nula rozumíme nulový polynom. Již máme definovanou první derivaci polynomu. Samozřejmě že můžeme uvažovat na množině $\mathbb{T}[x]$ i derivace vyšších řádů. Jejich definici zavedeme matematickou indukcí.

Definice 3.5.2. Nechť $f(x) \in \mathbb{T}[x], k \in \mathbb{N} \cup \{0\}$. Pak k-tou derivaci $f^{(k)}(x)$ polynomu $f(x)$ definujeme indukcí takto:
$f^{(0)} = f(x)$
$f^{(1)} = f'(x)$
$f^{(k)} = (f^{(k-1)}(x))'$ pro $k > 1$

Derivace by určitě neměla chybět v našem rejstříku ovládaných operací. V programu Maxima ji zajišťuje funkce *diff()*. Jejími argumenty jsou polynom,

neurčitá a číslo derivace. Jedná-li se o první derivaci, můžeme poslední argument vypustit. Nejprve provedeme první derivaci $g(x)$, kde vypustíme poslední argument.

```
(%i35) diff(g(x),x);
```

(%o35) $5\,x^4 + 4\,x^3 + 3$

Poté provedeme třetí derivaci $g(x)$ i s posledním argumentem.

```
(%i36) diff(g(x),x,3);
```

(%o36) $60\,x^2 + 24\,x$

3.6 Kořeny polynomu

Hledání kořenů je jednou z nejčastějších úloh při práci s polynomy.

Definice 3.6.1. Nechť $f(x) \in \mathbb{T}[x], c \in \mathbb{T}' \supseteq \mathbb{T}$. Pak řekneme, že c je kořenem polynomu $f(x)$, jestliže platí $f(c) = 0$.

My si ukážeme funkci *solve()*, přijímající polynom, u kterého se mají kořeny hledat. Jednotlivé kořeny nám budou vráceny v seznamu.

```
(%i37) g(x) := x^3 - 9*x -28;
```

(%o37) $\mathrm{g}\,(x) := x^3 - 9\,x - 28$

```
(%i38) solve(g(x));
```

(%o38) $[x = -\sqrt{3}\,i - 2, x = \sqrt{3}\,i - 2, x = 4]$

Požadujeme-li získat jen reálné kořeny, použijeme funkci *algsys()*. Té jako argumenty předáme dva seznamy. Do prvního seznamu vložíme výraz a do druhého vložíme neurčitou, pro kterou se má daný výraz řešit. Za funkci ještě připojíme

příkaz „realonly:true“, který nám zajistí vrácení pouze reálných kořenů. Samotná funkce vrací seznam, ve kterém jsou seznamy kořenů. Nejprve tedy vypočítáme všechny komplexní kořeny polynomu $g(x)$.

```
(%i39) algsys([g(x)], [x]);
```

(%o39) $[[x = \sqrt{3}\,i - 2], [x = -\sqrt{3}\,i - 2], [x = 4]]$

Nyní připojíme příkaz „realonly:true“ a získáme tedy pouze reálné kořeny.

```
(%i40) algsys([g(x)], [x]), realonly:true;
```

(%o40) $[[x = 4]]$

Následující definice nám definuje pojem k-násobného kořene polynomu.

Definice 3.6.2. Nechť $f(x) \in \mathbb{T}[x]$, $k \in \mathbb{N} \cup \{0\}$. Prvek $c \in \mathbb{T}' \supseteq \mathbb{T}$ nazveme k-násobným kořenem polynomu $f(x)$, jestliže $[(x-c)^k \mid f(x)] \wedge [(x-c)^{k+1} \nmid f(x)]$. Je-li $k = 1$, mluvíme o jednoduchém kořenu $f(x)$. Pokud $k = 0$, pak c není kořenem $f(x)$.

Definici můžeme také chápat tak, jak je popsáno v následující poznámce.

Poznámka 3.6.1. Prvek $c \in \mathbb{T}' \supseteq \mathbb{T}$ k-násobným kořenem $f(x) \in \mathbb{T}[x] \Leftrightarrow [f(x) = (x-c)^k \cdot g(x)] \wedge [g(c) \neq 0]$

Již víme, že funkce solve() vrací kořeny polynomu. Zjistíme tedy kořeny následujícího polynomu.

```
(%i41) solve(x^4-10*x^3+36*x^2-54*x+27);
```

(%o41) $[x = 1, x = 3]$

Násobnost jednotlivých kořenů se dozvíme pomocí příkazu „multiplicities;“.

```
(%i42) multiplicities;
```

(%o42) $[1, 3]$

Jednotlivé prvky v seznamu odpovídají násobnosti kořenů vrácených funkcí *solve()*. V našem případě je tedy číslo tři trojnásobným kořenem a číslo jedna je jednoduchým kořenem. O tom se nyní přesvědčíme.

```
(%i43) factor(x^4-10*x^3+36*x^2-54*x+27);
```

(%o43) $(x-3)^3\ (x-1)$

Polynom jsme upravili do tvaru součinu a je tedy na první pohled jasné, že číslo tři je opravdu trojnásobným kořenem. Číslo jedna je kořenem jednoduchým.

4 Algebraické rovnice a jejich řešení

V této kapitole čtenáře seznámíme s pojmem algebraická rovnice. Poté si ukážeme binomické, reciproké a kubické rovnice společně s jejich řešením. Lineárním se v tomto textu věnovat nebudeme.

V minulé kapitole jsme se již seznámili s problematikou rovnosti polynomu. Na tento problém se ovšem můžeme dívat i z jiného pohledu, a to tak, že budeme pro dva polynomy hledat všechna taková α z nadtělesa $\mathbb{T}'$ tělesa $\mathbb{T}$, ve kterých se budou polynomy rovnat. Tedy pro dané dva polynomy $f(x), g(x) \in \mathbb{T}[x]$ budeme řešit rovnici:

$$f(x) = g(x)$$

Tuto rovnici nazveme *algebraickou rovnicí* a všechna $\alpha \in \mathbb{T}'$, která jí budou vyhovovat, označíme jako *řešení dané algebraické rovnice*. Pokud od rovnice odečteme polynom $g(x)$, dostaneme na levé straně rovnice polynom $h(x) = f(x) - g(x)$ a na pravé straně nulu. Rovnice bude tedy v tomto tvaru:

$$h(x) = 0$$

Tuto rovnici nazýváme *obecnou algebraickou rovnicí*. Ta je s původní rovnicí ekvivalentní, a tudíž má stejnou množinu řešení. Tuto úlohu můžeme zase zpětně chápat jako hledání kořenů polynomu $h(x)$. Vícenásobné kořeny polynomu $h(x)$ budeme v algebraických rovnicích označovat jako vícenásobná řešení. Jako stupeň rovnice budeme rozumět stupeň polynomu $h(x)$.

Dále se zaměříme na algebraickou řešitelnost.

4.1 Algebraická řešitelnost

V úvodu kapitoly jsme uvedli, že při řešení algebraických rovnic hledáme všechna řešení α z nadtělesa $\mathbb{T}'$ tělesa $\mathbb{T}$. Uvedeme si tedy definici vytvoření nejmenšího tělesa obsahujícího těleso $\mathbb{T}$ a prvek α.

Definice 4.1.1. Nechť $(\mathbb{T}, +, \cdot)$ je těleso, $(\mathbb{T}', +, \cdot)$ je jeho nadtěleso a $A \subseteq \mathbb{T}'$. Pak těleso $(\mathbb{T}(A), +, \cdot)$, které je průnikem systému všech podtěles tělesa $(\mathbb{T}', +, \cdot)$

obsahující množinu $\mathbb{T} \cup A$, nazveme tělesem vzniklým adjunkcí množiny A k tělesu $\mathbb{T}$ (Rozšířením tělesa $\mathbb{T}$ o množinu A).

Poznámka 4.1.1. Pro $A = \{a\}$ hovoříme o jednoduchém rozšíření, které značíme $\mathbb{T}(a)$.

Při řešení rovnic budeme často rozšiřovat o n-té odmocniny z čísla a, tzv. *radikály.*

Definice 4.1.2. Nechť a je libovolný nenulový prvek z tělesa $\mathbb{T}$, pak jednoduché algebraické rozšíření $\mathbb{T}(\sqrt[n]{a})$ nazveme rozšířením tělesa $\mathbb{T}$ pomocí radikálu $\sqrt[n]{a}$. O tělese $\mathbb{T}(\sqrt[n]{a})$ též říkáme, že vzniklo adjunkcí radikálu $\sqrt[n]{a}$ k tělesu $\mathbb{T}$.

Nyní můžeme definovat pojem *algebraické řešení rovnic.*

Definice 4.1.3. Řekneme, že algebraická rovnice $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0$ stupně $n \in \mathbb{N}, n \geq 1$ nad tělesem $\mathbb{T}$ je algebraicky řešitelná (řešitelná pomocí radikálů), jestliže existuje konečná posloupnost těles tvaru $\mathbb{T} = \mathbb{T}_0 \subseteq \mathbb{T}_1 \subseteq \ldots \subseteq \mathbb{T}_k$ splňující následující podmínky.

1) Těleso $\mathbb{T}_k$ obsahuje všechna řešení dané rovnice.

2) Každé těleso $\mathbb{T}_{i+1}$ je rozšířením tělesa $\mathbb{T}_i$ pomocí nějakého radikálu $\sqrt[n_i]{a_i}$, kde $a_i \in \mathbb{T}_i, i \in \{0, 1, \ldots, k-1\}$.

Nyní se podíváme na různé typy algebraických rovnic.

4.2 Kvadratická rovnice s imaginárními koeficienty

Definice 4.2.1. Kvadratickou rovnicí (rovnicí 2. stupně) nad číselným tělesem $\mathbb{T}$ nazveme rovnici tvaru

$$ax^2 + bx + c = 0 \ ,$$

kde $a, b, c \in \mathbb{T}, a \neq 0$.

Následující věta mluví o řešitelnosti kvadratické rovnice.

Věta 4.2.1. Každá kvadratická rovnice nad číselným tělesem $\mathbb{T}$ je algebraicky řešitelná a její řešení leží v tělese $\mathbb{T}(\sqrt{D})$.

Kvadratické rovnice s imaginárními koeficienty tedy počítáme stejně jako kvadratické rovnice s reálnými koeficienty. Využíváme známého vzorce pro výpočet kořenů

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a} \ ,$$

kde $D = b^2 - 4ac$. Pokud D má nenulovou imaginární složku, můžeme pro výpočet odmocniny diskriminantu využít těchto vzorců pro komplexní číslo $z = u + vi$.

a) Je-li $v > 0$, pak $\sqrt{z} = \pm[\ \sqrt{1/2(u + \sqrt{u^2 + v^2})} + i\sqrt{1/2(-u + \sqrt{u^2 + v^2})}\ ]$.

b) Je-li $v < 0$, pak $\sqrt{z} = \pm[\ \sqrt{1/2(u + \sqrt{u^2 + v^2})} - i\sqrt{1/2(-u + \sqrt{u^2 + v^2})}\ ]$.

Další možností je diskriminant D převést na goniometrický tvar a pomocí vzorce pro druhou odmocninu z komplexního čísla

$$z_k = \sqrt{|z|}[cos(\frac{\alpha + 2k\pi}{2}) + isin(\frac{\alpha + 2k\pi}{2})] \ ,$$

kde $k = \{0, 1\}$, získat potřebné odmocniny diskriminantu.

Příklad 4.1. Řešte rovnici $x^2 + (-1 - 4i)x - i - 5 = 0$ nad $\mathbb{C}$.
Řešení: Řešením této rovnice jsou kořeny polynomu na levé straně rovnice, tedy

$$x_{1,2} = \frac{-(-1 - 4i) \pm \sqrt{5 + 12i}}{2} \ .$$

Nyní využijeme vzorců pro výpočet odmocnin komplexního čísla.

$$\sqrt{5 + 12i} = \pm[\ \sqrt{1/2(5 + \sqrt{5^2 + 12^2})} + i\sqrt{1/2(-5 + \sqrt{5^2 + 12^2})}\ ] = \pm(3 + 2i)$$

Odtud dostáváme

$$x_{1,2} = \frac{-(-1 - 4i) \pm (3 + 2i)}{2} \ ,$$

tedy $x_1 = 2 + 3i$ a $x_2 = -1 + i$.

Samozřejmě si výsledky ověříme za pomoci programu Maxima. Tentokrát nepoužijeme funkci *solve()*, ale použijeme funkci *gfactor()*. Ta rozloží polynom nad komplexními čísly.

```
(%i1)  gfactor(x^2 + (-1-4*%i)*x - 5 -%i);
```

(%o1) $(x - 3\,i - 2)\,(x - i + 1)$

Funkce nám vrátila rozklad polynomu na pravé straně rovnice. Z tohoto rozkladu poté snadno zjistíme kořeny polynomu, a tedy i řešení rovnice. Řešením jsou opravdu $x_1 = 2 + 3i$ a $x_2 = -1 + i$.

4.3 Kubické rovnice

Definice 4.3.1. Kubickou rovnicí nad číselným tělesem $\mathbb{T}$ nazveme rovnici tvaru $a_3x^3 + a_2x^2 + a_1x + a_0 = 0$, kde $a_3, a_2, a_1, a_0 \in \mathbb{T}, a_3 \neq 0$.

Teď, když víme, jak kubická rovnice vypadá, zaměříme se na její řešení. Ukážeme si, jak řešení najít v jednotlivých případech zadání kubické rovnice.

a) **rovnice v součinovém tvaru**

Pokud bychom měli rovnici ve tvaru součinu lineárních činitelů

$$a(x - x_1)(x - x_2)(x - x_3) = 0\ ,$$

jsou řešením právě čísla x_1, x_2, x_3.

b) **rovnice bez absolutního členu**

V rovnici $a_3x^3 + a_2x^2 + a_1x = 0$ na levé straně vytkneme x, tím jsme nalezli jedno řešení $x_1 = 0$. Další dvě řešení vypočítáme z kvadratické rovnice $a_3x^2 + a_2x + a_1 = 0$.

c) **ostatní**

V těchto případech využíváme tzv. *Cardanových vzorců*. My si uvedeme obecný postup bez odvozování. Celý postup si může čtenář podrobně projít v [4] str. 27.

Řešme rovnici $a_3x^3 + a_2x^2 + a_1x + a_0 = 0$.

Nejprve celou rovnici vydělíme koeficientem a_3 a dostaneme rovnici:

$$x^3 + b_2x^2 + b_1x + b_0 = 0 \tag{1}$$

kde $b_2 = \frac{a_2}{a_3}, b_1 = \frac{a_1}{a_3}, b_0 = \frac{a_0}{a_3}$. Poté se substitucí $x = y - \frac{b_2}{3}$ zbavíme kvadratického členе a po úpravě dostaneme tuto rovnici:

$$y^3 + py + g = 0 \tag{2}$$

Dále zavedeme substituci $y = u + v$. Po dosazení do (2) získáme rovnici

$$u^3 + v^3 + (3uv + p)(u + v) + q = 0 \ . \tag{3}$$

Nyní si pro neznámé u a v můžeme dovolit zvolit podmínku $3uv + p = 0$, tj. $uv = -\frac{p}{3}$. Z rovnice (3) pak dostaneme

$$u^3 + v^3 = -q \ . \tag{4}$$

Dále po umocnění podmínky $uv = -\frac{p}{3}$ na třetí dostaneme

$$u^3v^3 = -\frac{p^3}{27} \ . \tag{5}$$

Z posledních dvou vztahů platí, že prvky u^3 a v^3 jsou řešením kvadratické rovnice

$$z^2 + qz - \frac{p^3}{27} = 0 \ . \tag{6}$$

Tuto rovnici nazýváme kvadratickou rezolventou kubické rovnice. Pro diskriminant této rovnice platí

$$D = q^2 + 4\frac{p^3}{27} = 4(\frac{q^2}{4} + \frac{p^3}{27}) \ .$$

Tedy obě řešení leží v tělese $\mathbb{T}(\sqrt{\Delta})$, kde $\sqrt{\Delta}$ je libovolný druhý radikál z prvku $\Delta = \frac{q^2}{4} + \frac{p^3}{27}$, který nazveme diskriminantem kvadratické rezolventy kubické rovnice a platí

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{\Delta}}, \ v = \sqrt[3]{-\frac{q}{2} - \sqrt{\Delta}} \ .$$

Tedy u a v jsou libovolné třetí radikály z prvků u^3 a v^3. Řešení rovnice (2) pak získáme z následujících vzorců, které označujeme jako *Cardanovy vzorce*:

$y_1 = u + v,$

$y_2 = \varepsilon u + \varepsilon^2 v,$

$y_3 = \varepsilon^2 + \varepsilon v,$

kde ε je libovolná primitivní třetí odmocnina z jedné a u, v je libovolná dvojice vázaná vztahem $uv = -\frac{p}{3}$. Poté nezbývá než provést zpětnou substituci a získat tak řešení x_1, x_2, x_3 rovnice (1).

Věta 4.3.1. Každá kubická rovnice nad číselným tělesem $\mathbb{T}$ je algebraicky řešitelná a její řešení leží v $\mathbb{T}(\sqrt{\Delta}, u, \varepsilon)$, kde u je libovolný třetí radikál z u^3.

Následující věta objasňuje řešitelnost kubických rovnic s reálnými koeficienty.

Věta 4.3.2. Je-li

a) $\Delta < 0$, má kubická rovnice tři různá reálná řešení.

b) $\Delta = 0$, má kubická rovnice dvě reálná řešení z toho jedno dvojnásobné.

c) $\Delta > 0$, má kubická rovnice jedno reálné a dvě navzájem komplexně sdružená řešení.

Cardanovy vzorce ovšem nejsou použitelné vždy. My si nyní ukážeme příklad, který tak trochu poukazuje na omezené praktické využití těchto vzorců.

Příklad 4.2. Řešte rovnici $x^3 - 13x - 12 = 0$ nad $\mathbb{C}$.
Řešení: pro diskriminant kvadratické rezolventy této rovnice platí:

$$\Delta = (\frac{q}{2})^2 + (\frac{p}{3})^3 = (-6)^2 + (\frac{13}{3})^3 = 36 - \frac{2197}{27} = -\frac{1225}{27}$$

Dostali jsme záporný diskriminant, což znamená, že rovnice má tři různá reálná řešení. Funkcí *solve()* se o tom nyní přesvědčíme.

```
(%i2)  solve(x^3-13*x-12);
```

(%o2) $[x = 4, x = -1, x = -3]$

Vidíme, že řešením této rovnice jsou čísla $x_1 = 4, x_2 = -1, x_3 = -3$. Pokusíme-li se řešení získat pomocí Cardanových vzorců, dostaneme je v této formě:

$$u = \sqrt[3]{-\tfrac{q}{2} + \sqrt{\Delta}} = \sqrt[3]{6 + \tfrac{35i}{9}\sqrt{3}}\ ,$$

$$v = \sqrt[3]{-\tfrac{q}{2} + \sqrt{\Delta}} = \sqrt[3]{6 - \tfrac{35i}{9}\sqrt{3}}\ ,$$

$$v = -\tfrac{p}{3u}\ ,$$

$$x_1 = u + v = \sqrt[3]{6 + \tfrac{35i}{9}\sqrt{3}} + \sqrt[3]{6 - \tfrac{35i}{9}\sqrt{3}}\ ,$$

$$x_2 = \varepsilon u + \varepsilon^2 v = \varepsilon\sqrt[3]{6 + \tfrac{35i}{9}\sqrt{3}} + \varepsilon^2\sqrt[3]{6 - \tfrac{35i}{9}\sqrt{3}}\ ,$$

$$x_3 = \varepsilon^2 u + \varepsilon v = \varepsilon^2\sqrt[3]{6 + \tfrac{35i}{9}\sqrt{3}} + \varepsilon\sqrt[3]{6 - \tfrac{35i}{9}\sqrt{3}}\ .$$

Tento případ nazýváme *casus irreducibilis* (nerozložitelný případ). Řešit tyto rovnice je možné pomocí tzv. *goniometrického řešení*. My se ovšem v těchto případech zaměříme na výpočet řešení pomocí tzv. *numerických metod pro přibližné řešení algebraických rovnic* v následující kapitole.

4.4 Kvartické rovnice (rovnice 4. stupně)

Definice 4.4.1. Kvartickou rovnicí nad číselným tělesem $\mathbb{T}$ nazveme rovnici tvaru

$a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0$, kde $a_4, a_3, a_2, a_1, a_0 \in \mathbb{T}, a_4 \neq 0$.

Každá kvartická rovnice nad číselným tělesem $\mathbb{T}$ je algebraicky řešitelná. Řešení pomocí radikálů však může být problematické. My se zaměříme pouze na řešení rovnic v programu Maxima. Postup řešení těchto rovnic je uveden v [4] str. 33. Výpočet těchto rovnic je v programu Maxima opravdu snadný. Přistupme tedy rovnou k příkladu.

Příklad 4.3. Řešte rovnici $x^4 + 7x^3 - 23x^2 - 39x + 54 = 0$.

Řešení: Použijeme funkci *solve()*.

```
(%i3)  solve(x^4+7*x^3-23*x^2-39*x+54);
```

(%o3) $[x = 3, x = -2, x = 1, x = -9]$

Na výstupu jsme dostali řešení rovnice $x_1 = 3$, $x_2 = -2$, $x_3 = 1$ a $x_4 = 9$.

Věta 4.4.1. Algebraické rovnice stupně $n > 4$ nejsou obecně vždy algebraicky řešitelné.

V těchto případech je nutné hledat řešení pomocí numerických metod, které si představíme v kapitole 5. Ovšem i pro některé případy těchto rovnic dokážeme najít jednoduše řešení. Jedná se o případy binomických a reciprokých rovnic.

4.5 Binomické rovnice

Definice 4.5.1. Binomickou rovnicí rozumíme algebraickou rovnici ve tvaru:

$$x^n - a = 0,$$

kde $a \in \mathbb{C}, a \neq 0, n \geq 1$. Rovnice má v $\mathbb{C}$ n různých řešení (n-té odmocniny z čísla a).

Binomické rovnice můžeme řešit goniometricky a v některých případech algebraicky. Goniometrické řešení spočívá v tom, že komplexní číslo a vyjádříme v goniometrickém tvaru a poté, dle známého vzorce pro výpočet odmocniny z komplexního čísla, najdeme řešení rovnice. U algebraického způsobu využíváme vzorců pro rozklad dvojčlenu. Obě řešení si ukážeme na jednoduchém příkladu.

Příklad 4.4. Řešte rovnici $x^3 - 5 = 0$.
Řešení:
a) goniometrický způsob
Přepíšeme číslo a do goniometrického tvaru komplexního čísla.

$$a = |5|(cos\ 0\pi + isin\ 0\pi)$$

pomocí vzorce pro n-tou odmocninu komplexního čísla:

$$x_k = \sqrt[n]{|z|}(cos\ \frac{\varphi + 2k\pi}{n} + isin\ \frac{\varphi + 2k\pi}{n}), k \in \{0, 1, \ldots, n-1\}$$

lehce vypočítáme řešení $x_1 = \sqrt[3]{5}, x_2 = -\sqrt[3]{5}(\frac{1-i\sqrt{3}}{2}), x_3 = -\sqrt[3]{5}(\frac{1+i\sqrt{3}}{2})$

b) algebraický způsob

$$x^3 - 5 = x^3 - (\sqrt[3]{5})^3 = (x^3 - \sqrt[3]{5})(x^2 + x\sqrt[3]{5} + \sqrt[3]{5^2})$$

Z rozkladu vidíme, že řešením bude $x_1 = \sqrt[3]{5}$. Další dvě komplexní řešení $x_2 = -\sqrt[3]{5}(\frac{1-i\sqrt{3}}{2}), x_3 = -\sqrt[3]{5}(\frac{1+i\sqrt{3}}{2})$ dostaneme vyřešením kvadratické rovnice $(x^2 + x\sqrt[3]{5} + \sqrt[3]{5^2}) = 0$.

Daný příklad nyní vyřešíme v programu Maxima funkcí *solve()*.

```
(%i4)  solve(x^3 - 5);
```

$$(\%o4) \quad [x = \frac{\sqrt{3}\,5^{\frac{1}{3}}\,i - 5^{\frac{1}{3}}}{2}, x = -\frac{\sqrt{3}\,5^{\frac{1}{3}}\,i + 5^{\frac{1}{3}}}{2}, x = 5^{\frac{1}{3}}]$$

Snadno vidíme, že jsme dostali stejná řešení.

4.6 Reciproké rovnice

Definice 4.6.1. Algebraická rovnice tvaru $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = 0, a_n \neq 0$ nad $\mathbb{C}$ se nazývá:

a) ***reciproká rovnice 1. druhu***: jestliže $a_n = a_0, a_{n-1} = a_1, a_{n-2} = a_2, \ldots,$ obecně $a_{n-k} = a_k, k \in \{0, 1, \ldots, n\}$. Polynom na levé straně této rovnice se nazývá *reciproký polynom 1. druhu*.

b) ***reciproká rovnice 2. druhu***: jestliže $a_n = -a_0, a_{n-1} = -a_1, a_{n-2} = -a_2, \ldots,$ obecně $a_{n-k} = -a_k, k \in \{0, 1, \ldots, n\}$. Polynom na levé straně této rovnice se nazývá *reciproký polynom 2. druhu*.

Pro všechna řešení reciproké rovnice c je řešením také $\frac{1}{c}$. Postup řešení těchto rovnic je následující:

i) Je-li reciproká rovnice 2. druhu, pak je řešením rovnice $c = 1$. Vydělíme-li rovnici dvojčlenem $x - 1$, dostaneme reciprokou rovnici 1. druhu.

ii) Reciproká rovnice 1. druhu, která je lichého stupně, má řešení $c = -1$. Vydělením reciproké rovnice dvojčlenem $x+1$ získáme reciprokou rovnici 1. druhu sudého stupně.

iii) Reciprokou rovnici 1. druhu sudého stupně $n = 2m$ převedeme na algebraickou rovnici poloviního stupně m. Nejprve vydělíme rovnici x^m a poté vytkneme z prvního a posledního člene, druhého a předposledního atp. jejich koeficient. V rovnici zavedeme substituci $y = x + \frac{1}{x}$. Dostaneme tak obecnou rovnici *m-tého* stupně s neznámou y.

Příklad 4.5. Řešte rovnici $x^6 + 2x^5 + x^4 - x^2 - 2x - 1 = 0$

Řešení: Jedná se o reciprokou rovnici 2. druhu, tudíž jedním řešením bude $c_1 = 1$. Nyní vydělíme obě strany rovnice dvojčlenem $x - 1$. Dostaneme tedy rovnici

$$x^5 + 3x^4 + 4x^3 + 4x^2 + 3x + 1 = 0 \ . \tag{1}$$

Rovnice (1) je reciproká 1. druhu lichého stupně. Dalším řešením je $c_2 = -1$. Vydělíme tedy rovnici (1) dvojčlenem $x + 1$. Po úpravě dostaneme následující rovnici

$$x^4 + 2x^3 + 2x^2 + 2x + 1 = 0 \ . \tag{2}$$

Získaná rovnice (2) je 1. druhu sudého stupně. Pokračovat budeme vydělením rovnice x^2

$$x^2 + 2x + 2 + \frac{2}{x} + \frac{1}{x^2} = 0 \ . \tag{3}$$

Z rovnice (3) vytkneme společné koeficienty u prvního a posledního člene. Poté to samé i u druhého a předposledního

$$(x^2 + \frac{1}{x^2}) + 2(x + \frac{1}{x}) + 2 = 0 \ . \tag{4}$$

Zavedeme substituci $y = x + \frac{1}{x}$, kde $x^2 + \frac{1}{x^2} = y^2 - 2$ a dosadíme do rovnice (4). Po úpravě dostaneme rovnici

$$y^2 + 2y = 0 \tag{5}$$

Rovnici (5) snadno vyřešíme a nalezneme řešení vyhovující rovnici (5). Řešení jsou tato: $y_1 = 0, y_2 = -2$. Musíme se ještě vrátit zpět ze substituce $y = x + \frac{1}{x}$. Po dosazení y_1, y_2 dostaneme tyto rovnice.

$$x^2 + 1 = 0 \tag{6}$$

$$x^2 + 2x + 1 = 0 \tag{7}$$

Z rovnice (6) získáme $c_3 = +i$ a $c_4 = -i$. Z rovnice (7) získáme kořeny $c_5 = -1$ a $c_6 = -1$.

Řešením jsou tedy čísla $c_1 = 1, c_2 = -1, c_3 = i, c_4 = -i$, kde řešení $c_2 = -1$ je trojnásobné. O tom se přesvědčíme funkcí *solve()* a příkazem „multiplicities;“ v programu Maxima.

```
(%i5)  solve(x^6 + 2*x^5 + x^4 - x^2 - 2*x -1);
```

(%o5) $[x = -i, x = i, x = 1, x = -1]$

```
(%i6)  multiplicities;
```

(%o6) $[1, 1, 1, 3]$

Ve výstupu vidíme, že jsme řešení spočítali správně a řešení $c_2 = -1$ je opravdu trojnásobné.

5 Numerické metody pro přibližné řešení algebraických rovnic

Program Maxima podporuje přibližné řešení algebraických rovnic. My si pro přiblížení uvedeme danou metodu i s algoritmem výpočtu. Pro lepší pochopení spočítáme ukázkový příklad a poté připojíme řešení za pomoci funkce, která tuto metodu při svém výpočtu využívá v programu Maxima.

Ještě něž se zaměříme na numerické hledání řešení, uvedeme si tzv. *Sturmovu metodu*, která zjišťuje počet reálných kořenů polynomu na daném intervalu. Než vyslovíme samotnou metodu, musíme se nejprve seznámit s pojmem *Sturmova posloupnost*.

Definice 5.0.2. Posloupnost polynomů

$$f(x) = f_0(x), f_1(x), \dots \ f_{n+1}(x), \ kde\ n = stf(x), \ f_0(x), \dots, f_{n+1}(x) \in \mathbb{T}[x]$$

se nazývá Sturmovou posloupností příslušnou polynomu $f(x)$, jestliže platí následující

a) Všechny reálné kořeny polynomu $f_0(x)$ jsou jednoduché.

b) $f_0(x) = f(x)$

c) $f_1(x) = -f_0'(x)$

d) Pro další členy platí: $f_{i+1}(x) = -rem(f_{i-1}(x), f_i(x)), i = \{1, 2, \dots, n\}$, kde $rem(f_{i-1}(x), f_i(x))$ je zbytek po dělení polynomu $f_{i-1}(x)$ polynomem $f_i(x)$.

e) $(f_{n+1}(x) \neq 0) \wedge (stf_{n+1}(x) = 0)$

Důkaz následující věty uvádět nebudeme.

Věta 5.0.1 (Sturmova)**.** Počet reálných kořenů polynomu $f(x)$ v intervalu $\langle a, b\rangle$ je roven $W(b) - W(a)$, kde $W(x)$ je počet znaménkových změn ve Sturmově posloupnosti $f_0(x), f_1(x), \dots, f_{n+1}(x)$ v bodě x. Do znaménkových změn nepočítáme nuly.

Příklad 5.1. Určete počet reálných kořenů polynomu

$$f(x) = x^3 - 8x + 6$$

Řešení: Sestavíme Sturmovu posloupnost polynomu $P(x)$.

$$f_0(x) = x^3 - 8x + 6$$

$$f_1(x) = -3x^2 + 8$$

$$f_2(x) = 16x - 18$$

$$f_3(x) = -1$$

Polynom $f_2(x)$ je zbytek po dělení polynomu $f_0(x)$ polynomem $f_1(x)$ vynásobený číslem tři. Tuto úpravu jsme si mohli dovolit, jelikož nám v metodě jde pouze o znaménka. Stejně tak jsme upravili i $f_3(x)$. Nyní si sestavíme tabulku pro určení počtu reálných kořenů.

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$W(x)$
$-\infty$	-	-	-	-	0
∞	+	-	+	-	3

Tab. 1: Tabulka změn Sturmovy posloupnosti.

Podle *Sturmovy věty* na intervalu $(-\infty, \infty)$, dle vztahu $W(\infty) - W(-\infty) = 3$, jsou tři reálné kořeny.

My se teď podíváme, jak příklad vyřešit za pomoci programu Maxima. Použijeme funkci *nroots()*. Jako argument vložíme polynom $f(x)$ a poté dva krajní body intervalu, ve kterém se mají reálné kořeny hledat. Předchozí příklad bychom vyřešili následujícím příkazem.

```
(%i1)  nroots(x^3 - 8*x +6, minf,inf);

(%o1)  3
```

Ověřili jsme tedy, že polynom $f(x)$ má tři reálné kořeny. Jako první metodu numerického určování kořenů polynomu si uvedeme *metodu půlení intervalu*.

5.1 Metoda půlení intervalů

Buď $f(x) \in \mathbb{T}[x]$ polynom mající na intervalu (a, b) jeden jednoduchý kořen. Určíme posloupnost bodů $x_1, x_2, \ldots, x_m$, která konverguje ke kořenu. Body posloupnosti určíme následovně:

1. $x_1 = a, x_2 = b$
2. $x_3 = \dfrac{x_1 + x_2}{2}$
3. Pro konstrukci dalšího bodu x_4 využijeme interval $\langle x_1, x_3\rangle$, pokud $f(x_1)f(x_3) < 0$. Interval $\langle x_3, x_2\rangle$ využijeme v případě, kdy $f(x_2)f(x_3) < 0$. Pro $f(x_1)f(x_3) = 0$ je x_3 kořenem polynomu $f(x)$.

Takto půlíme intervaly až do té doby, dokud nenalezneme přibližný kořen, který by vyhovoval naší zvolené odchylce. Jako přibližný kořen označíme prvek x_i, který vznikl půlením intervalu o délce menší než 2ε, kde ε je předem zadaná odchylka.

Příklad 5.2. Určete přibližnou hodnotu kořenu polynomu $f(x) = x^3 - 8x + 6$ na intervalu $(0, 1)$ s přesností $\varepsilon = 10^{-2}$.
Řešení: Sestavíme tabulku.

a	$\frac{a+b}{2}$	b	$sgnf(a)$	$sgnf(\frac{a+b}{2})$	$sgnf(b)$	$\lvert(a,b)\rvert$
0	1/2	1	+	+	-	1
1/2	3/4	1	+	+	-	0,5
3/4	7/8	1	+	-	-	0,25
3/4	13/16	7/8	+	+	-	0.125
13/16	27/32	7/8	+	-	-	0,0625
13/16	53/64	27/32	+	-	-	0,03125
13/16	105/128	53/64	+	-	-	0,015625 $< 2\varepsilon$

Tab. 2: Metoda půlení intervalu

Jelikož je splněna podmínka o délce intervalu a odchylky přesnosti ε, bude kořen polynomu $f(x)$ ležet v intervalu $(c - \varepsilon, c + \varepsilon)$, kde

$$c = \frac{\frac{13}{16} + \frac{105}{128}}{2} = \frac{209}{256} = 0.81640625\ .$$

Příklad nyní vyřešíme za pomoci programu Maxima, ve kterém bude výsledek samozřejmě přesnější. Budeme k tomu potřebovat funkci *find_root()*, té jako argumenty předáme polynom $f(x)$ a krajní body intervalu, na kterém chceme kořen hledat. Na výstupu obdržíme přibližnou hodnotu kořene polynomu $f(x)$. Řešení tedy bude vypadat následovně.

```
(%i2)  find_root(x^3 - 8*x +6,0,1);
```

(%o2) 0.81855805172667

Přesvědčíme se, zda jsme v požadované odchylce.

```
(%i3)  %i2 - 0.81640625;
```

(%o3) 0.0021518017266738

Vidíme, že hodnota je menší než naše zvolené ε, a tedy příklad byl vyřešen správně. Než přejdeme k další numerické metodě, seznámíme čtenáře s funkcí *realroots()*. Funkce vypočítává racionální aproximaci kořenů. Funkce přijímá na vstupu polynom a nepovinný argument ε, který udává s jakou přesností se mají kořeny hledat. Tato funkce využívá *Sturmovu metodu* pro určení intervalů s jedním reálným kořenem, ve kterých poté pomocí *metody půlení intervalu* kořeny aproximuje. Vypočteme tedy zmíněný příklad i za pomoci funkce *realroots()*. Příkazy „numer:true“ a „numer:false“ určují, zda nám bude výsledek vrácen v numerickém nebo racionálním tvaru.

```
(%i4)  realroots(x^3 - 8*x + 6), numer:false;
```

(%o4) $[x = -\frac{105610233}{33554432}, x = \frac{27466251}{33554432}, x = \frac{78143983}{33554432}]$

```
(%i5)  realroots(x^3 - 8*x + 6), numer:true;
```

(%o5) $[x = -3.147430211305618, x = 0.81855806708336,$
$x = 2.328872174024582]$

Opět jsme nalezli (s jistou odchylkou) stejný kořen $x = 0.81855806708336$. Nyní se můžeme zaměřit na další metodu.

5.2 Newtonova metoda (metoda tečen)

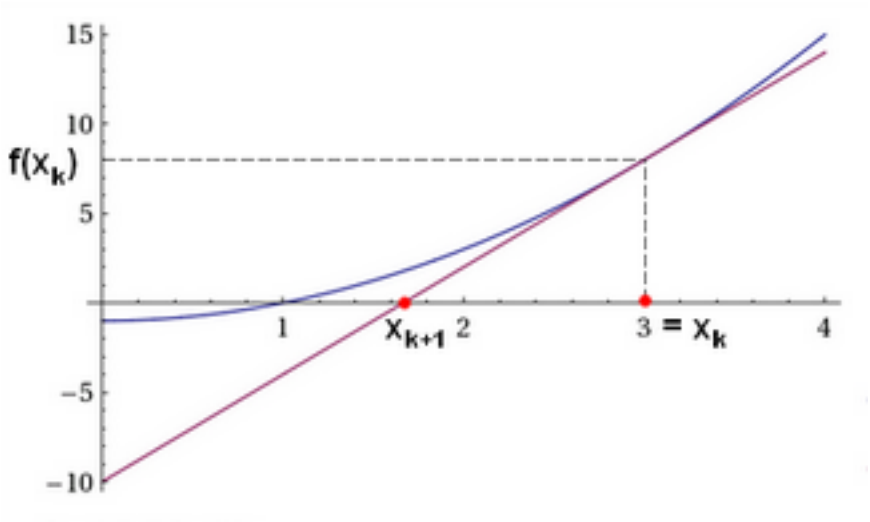
Je iterační metoda (Iterační metodou chápeme proces opakování stejného postupu výpočtu, který vede k dosáhnutí požadovaného výsledku). Newtonova metoda hledá přibližnější řešení rovnice $f(x) = 0$ v místě, ve kterém tečna ke grafu funkce sestrojená z bodu $f(x_k)$ protíná osu x. Bod x_k označujeme jako k-tou aproximaci kořene polynomu $f(x)$. Směrnice tečny z bodu x_k je $f'(x_k)$. Daný průsečík tečny s osou x označíme x_{k+1}. Vypočítat ho můžeme pomocí vztahu v definici 5.2.1.

Definice 5.2.1. Nechť polynom $f(x) \in \mathbb{T}[x]$ má kořen $c \in \mathbb{R}$. Iterační metoda:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, k = \{0, 1, \ldots, n\}, x_k \in \mathbb{R}, x_{k+1} \in \mathbb{R}$$

se nazývá Newtonova metoda.

Pro $k = 0$ označujeme prvek $x_0 \in \mathbb{R}$ jako původní odhad (aproximaci) kořene polynomu $f(x)$. Iteraci opakujeme a získáváme bližší aproximace $(x_1, x_2, \ldots x_n)$ až do té doby, kdy je hodnota $f(x_n)$ dostatečně blízko nuly. Pro lepší pochopení uvedeme grafické znázornění této metody. Na obrázku 1 je znázorněna aproximace z bodu x_k na bod x_{k+1}. Tečna v bodě $f(x_k)$ je označena fialovou barvou. Graf funkce je označen modrou barvou.



Obr. 1: Newtonova metoda tečen

Příklad 5.3. Proveďte první tři iterace Newtonovy metody a určete přibližnou hodnotu řešení rovnice $f(x) = 0$, kde $f(x) = x^3 - 8x + 6$ s počáteční aproximací $x_0 = 1$.

Řešení: Sestavíme tabulku aproximací.

k	x_k
0	1
1	0,8
2	0.81842105263158
3	0.81855804403377

Tab. 3: Newtonova metoda - tabulka aproximací.

Z tabulky vidíme, že přibližné řešení je $x_3 = 0.81855804403377$.

V programu Maxima použijeme funkci newton(). Nejdříve si ovšem musíme načíst knihovnu obsahující tuto funkci. Knihovnu načteme příkazem „load(newton);“. Poté již můžeme použít funkci newton(), které předáme polynom a hodnotu počáteční aproximace. Náš konkrétní příklad bychom vyřešili takto:

```
(%i6)  load(newton);
```

(%o6) $C : /PROGRA\ 2/MAXIMA\ 1.2/share/maxima/5.31.2/share/numeric/newton.mac$

```
(%i7)  newton(x^3 - 8*x +6,1);
```

(%o7) 0.81855805172667

Jistě jste si všimli, že u této funkce neudáváme přesnost. Pro určení řešení s přesností na x desetinných míst využijeme našich znalostí z kapitoly 2.4 a použijeme příkaz „fpprec: x;“. Spočítáme tedy příklad s přesností na čtyři desetinná místa.

```
(%i8)  fpprec: 4;
```

(%o8) 4

```
(%i9)  newton(x^3 - 8*x + 6,1);
```

(%o9) $8.184b - 1$

Vidíme, že se nám opravdu podařilo určit řešení s přesností na čtyři desetinná místa. Jako další uvedeme funkci *allroots()*, která využívá tzv. Jenkinsův-Traubův algoritmus. Samotnou metodu vzhledem k její náročnosti rozebírat nebudeme. Čtenář si ji může nastudovat na tomto odkazu [1]. My si ukážeme rovnou použití této funkce, která dokáže vypočítat i komplexní kořeny. Jako jediný argument funkci předáme polynom, u kterého se mají kořeny hledat. Na výstupu obdržíme seznam kořenů. Vyřešme tedy následující příklad.

Příklad 5.4. Vypočtěte přibližné hodnoty všech kořenů polynomu $f(x) = x^3 - 8x + 6$.
Řešení:

```
(%i10) allroots(x^3 - 8*x +6);
```

(%o10) $[x = 0.81855805172667, x = 2.32887218197108, x = -3.147430233697754]$

[1] `http://dml.cz/bitstream/handle/10338.dmlcz/141061/PokrokyMFA_46-2001-1_4.pdf`

Abychom se přesvědčili, že funkce zvládá i hledání komplexních kořenů, použijeme funkci *allroots()* na polynom $x^2 + 1$.

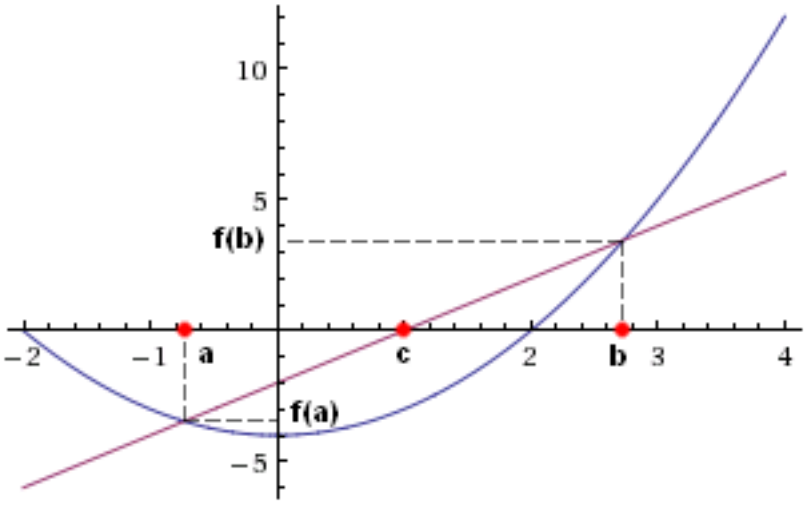
```
(%i11) allroots(x^2 +1);
```

(%o11) $[x = 1.0\,i, x = -1.0\,i]$

Funkce opravdu vrací komplexní kořeny.

5.3 Regula falsi (Metoda tětiv)

Tato metoda je nazývána také metodou tětiv a my si nyní přiblížíme proč. Spojíme-li úsečkou dva body o souřadnicích $[a, f(a)], [b, f(b)]$ dané křivky $f(x)$ (tuto úsečku nazýváme tětivou křivky), jejichž součin funkčních hodnot je záporný $f(a) \cdot f(b) < 0$, pak průsečík přímky procházející body $[a, f(a)], [b, f(b)]$ s osou x je bod c, který je zpřesnění řešení rovnice $f(x) = 0$. Celou situaci si můžeme prohlédnout na obrázku 2.



Obr. 2: Regula falsi - metoda tětiv

Výpočet hodnoty c můžeme interpretovat následujícím vztahem:

$$c = \frac{a \cdot f(b) + b \cdot f(a)}{f(b) - f(a)}$$

Další přesnější řešení hledáme buď v intervalu (a, c), pokud je $f(a) \cdot f(c) < 0$, nebo v intervalu (c, b) pokud je $f(c) \cdot f(b) < 0$. Tento postup aplikujeme do té doby, dokud není splněna podmínka $|f(c)| < \varepsilon$, kde ε je předem zvolená odchylka. Metodu si předvedeme znovu na našem ukázkovém příkladu.

Příklad 5.5. Určete přibližné řešení rovnice $f(x) = 0$, kde $f(x) = x^3 - 8x + 6$ na intervalu $(0, 1)$ s přesností $\varepsilon = 10^{-4}$.

Řešení: Sestavíme tabulku aproximací.

i	a_i	c_i	b_i	$f(c_i)$
1	0	0.85714285714286	1	-0.22740524781341
2	0	0.82584269662921	0.85714285714286	-0.043503508664919
3	0	0.81989795698322	0.82584269662921	-0.0080214714248297
4	0	0.81880328911885	0.81989795698322	-0.0014687968464179
5	0	0.81860289556027	0.81880328911885	-0.0002.6860460898647798
6	0	0.818566250449	0.81860289556027	-0.000049109264609770165

Tab. 4: Regula falsi (metoda tětiv) - tabulka aproximací

Vidíme, že v šesté iteraci hodnota $f(c_6)$ splňuje podmínku $|f(c_i)| < \varepsilon$. Řešením rovnice tedy je číslo $c_6 = 0.818566250449$.

Zda jsme počítali správně, bychom rádi ověřili pomocí programu Maxima. Ten však funkci, která by aproximovala řešení dané rovnice pomocí metody tětiv, neobsahuje. Musíme se tedy spolehnout sami na sebe a danou rekurzivní funkci si naprogramovat. Algoritmus výpočtu je poměrně jednoduchý. Můžeme si ho shrnout do těchto kroků:

1) Mějme interval (a, b), ve kterém leží řešení rovnice $(f(a) \cdot f(b) < 0)$.
2) Na intervalu (a, b) určíme bod c, který je dán vztahem $c = \frac{a \cdot f(b) + b \cdot f(a)}{f(b) - f(a)}$
3) Jestliže je splněna podmínka $|f(c)| < \varepsilon$ funkce vrátí bod c a výpočet bude ukončen.
4) Podmínka v kroku 3) nebyla splněna a volíme tedy interval, ve kterém se nachází řešení. To zvolíme pomocí následující podmínky: jestliže je $f(a) \cdot f(c) < 0$, pak $b = c$, jinak $a = c$.
5) Vracíme se na bod 2) s novým intervalem (a, b) a pokračujeme ve výpočtu.

Funkce naprogramovaná v programu Maxima by mohla vypadat asi takto:

Výpis kódu 1: Regula falsi (metoda tětiv)

```
1  regula_falsi(expr,a,b,e) :=
2     block([c,fa,fb,fc],
3        fa:subst(a,x,expr),
4        fb:subst(b,x,expr),
5        if (fa*fb<0)
6           then (c: (a*fb - b*fa)/(fb-fa),
7              fc:subst(c,x,expr),
8              (if (fa*fc<0)
9                 then b:c
10                else a:c),
11                if(abs(fc) < e)
12                   then c
13                   else regula_falsi(expr,a,b,e))
14          else false);
```

Pojďme si ji popsat řádek po řádku:

1. Na prvním řádku naši funkci pojmenujeme a určíme argumenty funkce. Argument *expr* nám bude reprezentovat daný polynom $f(x)$ z rovnice $f(x) = 0$. Argumenty a a b budou krajní body intervalu (a, b), ve kterém leží řešení rovnice. Argument e bude představovat danou odchylku.
2. Příkaz „block()“ můžeme chápat jako tělo celé funkce. A v něm na tomto řádku do listu (do hranatých závorek) deklarujeme proměnné *c, fa, fb, fc*.
3. Do proměnné *fa* vložíme hodnotu výrazu *expr* v bodě *a*.
4. Do proměnné *fb* vložíme hodnotu výrazu *expr* v bodě *b*.
5. Podmínkou ověříme, zda v daném intervalu skutečně leží kořen. Pokud ano, pokračujeme na řádek 6, v opačném případě funkce skočí na 14. řádek a vrátí hodnotu *false*.
6. Příkaz *then(...)* zahrnuje pravdivou větev větvení. Na tomto řádku do proměnné c vložíme průsečík přímky procházející body a, b s osou x.
7. Do proměnné *fc* vložíme hodnotu výrazu *expr* v bodě *c*.
8. Nyní, když máme bod c a hodnotu fc, potřebujeme zjistit, ve kterém intervalu dále hledat přibližnější řešení. To nám zajistí další podmínka, ve které

se ptáme, zda řešení leží v intervalu (a, c).

9. Předchozí podmínka byla splněna, a tedy řešení leží v intervalu (a, c). Proto musíme přepsat hodnotu b na hodnotu c.
10. Předchozí podmínka nebyla splněna a řešení tedy zákonitě musí ležet v intervalu (c, b). Proto musíme přepsat hodnotu a na hodnotu c.
11. V této chvíli tedy víme, ve kterém intervalu leží řešení. Nyní ověříme, zda získaný bod c vyhovuje zastavující podmínce $|fc| < e$.
12. Podmínka byla splněna a funkce vrací bod c, který je přibližným řešením vyhovujícím odchylce e.
13. Podmínka splněna nebyla a my tedy musíme hledat ještě přesnější řešení. Voláme tedy funkci znovu, tentokráte ovšem s jiným, menším intervalem. Takto funkce pokračuje do té doby, než bude podmínka na 11. řádku splněna.
14. Podmínka na 5. řádku nebyla splněna a tedy funkce byla volána s intervalem, ve kterém nelze touto funkcí nalézt přibližné řešení.

Funkci zkompilujeme.

```
(%i12) regula_falsi(expr,a,b,e) :=
          block([c,fa,fb,c,fc],
             fa:subst(a,x,expr),
             fb:subst(b,x,expr),
             if (fa*fb<0)
                then (c: (a*fb - b*fa)/(fb-fa),
                   fc:subst(c,x,expr),
                   (if (fa*fc<0)
                      then b:c
                      else a:c),
                      if(abs(fc) < e)
                         then c
                         else regula_falsi(expr,a,b,e))
                else false);
```

(%o12) regula_falsi $(expr, a, b, e) :=$ block$([c, fa, fb, c, fc], fa :$ subst $(a, x, expr)$,
$fb :$ subst $(b, x, expr)$, if $fa\ fb < 0$then$(c : \dfrac{a\,fb - b\,fa}{fb - fa}, fc :$ subst $(c, x, expr)$, if fa
$fc < 0$then$b : c$else$a : c$, if $|fc| < e$then$c$elseregula_falsi $(expr, a, b, e)))$

Nyní máme aparát, kterým si můžeme příklad 5.5 ověřit.

Příklad 5.6. Ověřte správnost řešení příkladu 5.5 pomocí navržené funkce *regula_ falsi()*.
Řešení: Máme určit přibližné řešení rovnice $f(x) = 0$, kde $f(x) = x^3 - 8x + 6$ na intervalu $(0, 1)$ s přesností $\varepsilon = 10^{-4}$. Použijeme funkci *regula_ falsi()*.

```
(%i13) regula_falsi(x^3 -8*x+6, 0, 1,10^(-4));
```

(%o13) 0.818566250449
Řešení je opravdu totožné. Tudíž jsme počítali správně.

My tímto můžeme přejít k poslední kapitole, ve které budeme řešit úlohy za pomoci již osvojených funkcí a znalostí.

6 Řešené příklady

Tato kapitola by měla sloužit jako praktické využití předešlých kapitol. Čtenář by měl být schopen příklady vyřešit za pomoci již známých funkcí. Kapitola neslouží jako sbírka úloh, ale jako návod pro řešení daných typových úloh. Proto budeme uvádět vždy jen jeden ukázkový příklad k danému typu příkladu.

Příklad 6.1. Vyjádřete polynom $f(x) = x^7+3x^6-9x^5-8x^4+3x^3-6x^2+3x-5$ v mocninách dvojčlenu $(x-2)$.
Řešení: Použijeme funkci *taylor()*.

```
(%i1)  taylor(x^7+3*x^6-9*x^5-8*x^4+3*x^3-6*x^2+3*x-5, x, 2, 8);
```

(%o1)/T/ $95+63\,(x-2)+492\,(x-2)^2+619\,(x-2)^3+362\,(x-2)^4+111\,(x-2)^5+17\,(x-2)^6+(x-2)^7+\ldots$
Na výstupu (%o1) jsme dostali potřebný Taylorův rozvoj.

Příklad 6.2. Vyjádřete polynom $f(x) = x^9-6x^6-13x^5+4x^4+2x^2-5x+6$ pomocí Hornerova schématu.
Řešení: Použijeme funkci *horner()*.

```
(%i2)  horner(x^9-6*x^6-13*x^5+4*x^4+2*x^2-5*x+6);
```

(%o2) $x\left(x\left(x^2\left(x\left(x\left(x^3-6\right)-13\right)+4\right)+2\right)-5\right)+6$
Na výstupu (%o2) jsme dostali potřebný vyjádření polynomu $f(x)$.

Příklad 6.3. Zjistěte, zda je bod $c = i$ kořenem následujícího polynomu, případně kolikanásobný:

$$f(x) = x^6+(6-3i)x^5-(2+18i)x^4-(12+2i)x^3-(3+12i)x^2-(18-i)x+6i$$

Řešení: Nejprve funkcí *solve()* zjistíme kořeny polynomu $f(x)$ a poté zjistíme jejich násobnost pomocí příkazu „multiplicities;“.

```
(%i3)  solve(x^6+(6-3*%i)*x^5-(2+18*%i)*x^4-(12+2*%i)*x^3
         -(3+12*%i)*x^2-(18-%i)*x+6*%i);
```

(%o3) $[x = i, x = -i, x = -6]$

```
(%i4)  multiplicities;
```

(%o4) $[4, 1, 1]$

Bod $c = i$ je čtyřnásobným kořenem polynomu $f(x)$.

Příklad 6.4. Určete polynom, který má stejné kořeny jako polynom $f(x)$, ale všechny jednoduché.

$$f(x) = x^5 - (13 - 2i)x^4 + (15 - 26i)x^3 + (205 + 32i)x^2 - (16 - 384i)x - 192$$

Řešení: Funkcí *solve()* zjistíme kořeny $f(x)$. Poté již snadno vytvoříme výsledný polynom.

```
(%i5)  solve(x^5-(13-2*%i)*x^4+(15-26*%i)*x^3+(205+32*%i)*x^2
       -(16-384*%i)*x-192);
```

(%o5) $[x = -i, x = -3, x = 8]$

```
(%i6)  expand((x +%i)*(x+3)*(x-8));
```

(%o6) $x^3 + i\,x^2 - 5\,x^2 - 5\,i\,x - 24\,x - 24\,i$

Řešením je polynom $g(x) = x^3 + ix^2 - 5x^2 - 5ix - 24x - 24i$.

Příklad 6.5. Určete všechny reálné kořeny polynomu

$$f(x) = x^3 + (9 - 6i)x^2 - (106 + 108i)x - 336i - 504$$

bez užití numerických metod.

Řešení: Použijeme funkci *algsys()* s příkazem „realonly:true;“.

```
(%i7)  algsys([x^3+(9 -6*%i)*x^2-(106 + 108*%i)*x-336*%i-504],[x]),
       realonly:true;
```

(%o7) $[[x = -14], [x = -4]]$

Hledanými reálnými kořeny jsou $x_1 = -14$ a $x_2 = -4$.

Příklad 6.6. Určete polynom stejného stupně jako $f(x)$, který bude mít kořeny o jedna větší než polynom $f(x)$.

$$f(x) = x^5 - 12x^4 - 22x^3 + 384x^2 + 621x - 972$$

Řešení: Funkcí *solve()* najdeme kořeny polynomu $f(x)$. Jelikož nám funkce vrátila jen čtyři kořeny, musí být jeden z kořenů dvojnásobný. Najdeme ho a poté již sestavíme nový polynom tak, že k získaným kořenům přičteme číslo jedna.

```
(%i8)
solve(x^5-12*x^4-22*x^3+384*x^2+621*x-972);
```

(%o8) $[x = 1, x = -4, x = -3, x = 9]$

```
(%i9)  multiplicities;
```

(%o9) $[1, 1, 1, 2]$

```
(%i10) expand((x-2)*(x+3)*(x+2)*(x-10)^2);
```

(%o10) $x^5 - 17\,x^4 + 36\,x^3 + 368\,x^2 - 160\,x - 1200$

Hledaným polynomem je polynom $g(x) = x^5 - 17x^4 + 36x^3 + 368x^2 - 160x - 1200$.

Příklad 6.7. Řešte rovnici $f(x) = \frac{39555875}{262144}$, kde

$$f(x) = 3x^6 + 5x^4 - 3x^3 + x^2 - 25$$

Řešení: Použijeme funkci *solve()*.

```
(%i11) solve(f(x) = 39555875/262144);
```

(%o11) $[x = \frac{15}{8}, 0 = 98304\,x^5 + 184320\,x^4 + 509440\,x^3 + 856896\,x^2 + 1639448\,x + 3073965]$

Řešením rovnice je $x = \frac{15}{8}$.

Příklad 6.8. Řešte v $\mathbb{R}$ reciprokou rovnici $x^5 + 3x^4 + x^3 + x^2 + 3x + 1 = 0$.

Řešení: použijeme funkci *algsys()* s parametrem „realonly:true;“.

```
(%i12) algsys([x^5 + 3*x^4 + x^3 +x^2 +3*x +1],[x]), realonly:true;
```

(%o12) $[[x = -1], [x = \frac{\sqrt{5}-3}{2}], [x = -\frac{\sqrt{5}+3}{2}]]$

Řešením jsou tyto kořeny: $x_1 = -1, x_2 = \frac{\sqrt{5}-3}{2}, x_3 = -\frac{\sqrt{5}+3}{2}$.

Příklad 6.9. Řešte v $\mathbb{C}$ binomickou rovnici $x^8 - 256 = 0$.

Řešení: Použijeme funkci *solve()*.

```
(%i13) solve(x^8 - 256);
```

(%o13) $[x = \frac{2\,i+2}{\sqrt{2}}, x = 2\,i, x = \frac{2\,i-2}{\sqrt{2}}, x = -2, x = -\frac{2\,i+2}{\sqrt{2}}, x = -2\,i,$
$x = -\frac{2\,i-2}{\sqrt{2}}, x = 2]$

Řešením jsou tedy všechna x ve výstupu %o13.

Příklad 6.10. Na intervalu $(1, 2)$ určete metodou půlení intervalu přibližné řešení rovnice

$$x^6 + 6x^4 + 3x^2 - 5x - 80 = 0$$

Řešení: Použijeme funkci *find_root()*.

```
(%i14) find_root(x^6+6*x^4+3*x^2-5*x-80, 1,2);
```

(%o14) 1.726020056479832

Přibližným řešením rovnice je $x = 1.726020056479832$.

Příklad 6.11. Určete počet kladných, reálných kořenů polynomu $f(x) = x^6 + 3x^5 - 6x^4 - 2x^3 - x^2 + 11x - 6$ a poté je určete i s jejich případnou násobností.

Řešení: Pro zjištění počtu kladných reálných kořenů vhodně použijeme funkci *nroots()*. Funkcí *realroots()* získáme kořeny polynomu a příkazem „multiplicities;“ zjistíme jejich násobnost.

```
(%i15) nroots (x^6+3*x^5-6*x^4-2*x^3-x^2+11*x-6,0,inf);
```

(%o15) 3

```
(%i16) realroots(x^6+3*x^5-6*x^4-2*x^3-x^2+11*x-6);
```

(%o16) $[x = 1, x = -\frac{145202797}{33554432}, x = \frac{30487399}{33554432}]$

```
(%i17) multiplicities;
```

(%o17) $[2, 1, 1]$

Kladnými, reálnými kořeny jsou $x_1 = \frac{30487399}{33554432}$ a dvojnásobný kořen $x_{2,3} = 1$.

Příklad 6.12. Metodou tečen řešte rovnici $x^4 - 5x^3 - 1435x^2 - 1250x + 42000 = 0$ s počáteční aproximací $x_0 = 5$.

Řešení: Použijeme funkci *newton()* v knihovně „newton“.

```
(%i18) load("newton");
```

(%o18) $C : /PROGRA\ 2/MAXIMA\ 1.2/share/maxima/5.31.2/share/numeric/newton.mac$

```
(%i19) newton(x^4-5*x^3-1435*x^2-1250*x+42000,5);
```

(%o19) $4.99191671278307 4b0$

Přibližným řešením je $x = 4.991916712783074$.

Příklad 6.13. Nad $\mathbb{C}$ určete přibližné řešení rovnice

$$x^6 + 5x^5 - 8x^4 + 6x^3 - 7x^2 + 5x - 7 = 0$$

Řešení: Použijeme funkci *allroots()*, která dokáže počítat i přibližné komplexní kořeny.

```
(%i20) allroots(x^6 + 5*x^5 - 8*x^4 + 6*x^3 - 7*x^2 + 5*x -7);
```

(%o20) $[x = 0.83455013470849\, i + 0.52430058107377,$

$x = 0.52430058107377 - 0.83455013470849\, i,$

$x = 0.83108885609152\, i - 0.4449931728215,$

$x = -0.83108885609152\, i - 0.4449931728215,$

$x = 1.26275530396683,$

$x = -6.421370120471371]$

Na výstupu (%o20) jsme dostali kořeny jako prvky seznamu.

Závěr

Práce čtenáře v první kapitole seznamuje s nejzákladnější výbavou a prací v programu Maxima. Po zvládnutí toho nejzákladnějšího si čtenář osvojí funkce pro práci s polynomy jedné neurčité. V dalších dvou kapitolách se seznámí s algebraickými rovnicemi a jejich řešením za pomoci programu Maxima. Poté utvrdí své nabyté znalosti v poslední kapitole řešených příkladů. V ní čtenář nalezne jednoduchá řešení mnohdy složitých algebraických rovnic za pomoci programu Maxima. To bylo hlavním cílem této práce. Domnívám se, že se povedl splnit. Dalším dílčím cílem bylo práci napsat tak, aby daná problematika byla lehce zvládnutelná i pro nové uživatele programu. O to jsem se pokusil a své snažení podpořil častými a přehlednými ukázkami z programu. Tento text by mohl sloužit jako podpůrný text pro studium algebraických rovnic a polynomů jedné neurčité.

Na trhu najdeme jistě i výkonnější CAS systémy než je program Maxima. Ovšem většina z těch, co nalezneme, budou komerční, a tudíž pro většinu uživatelů nedostupné. Pokud budeme uvažovat jen volně šiřitelné programy, je Maxima dle mého zcela jistě správnou volbou, ne-li tou nejlepší.

Literatura

[1] 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`

[2] DODIER, R. *Maxima 5.30.0 Manual* [online]. 2013 [cit. 2014-01-30]. Dostupné z: `http://maxima.sourceforge.net/docs/manual/en/maxima.html`

[3] EMANOVSKÝ, P., *Algebra 2 (pro distanční studium). Skriptum PdF UP, Olomouc*, 2001. ISBN 80 – 244 – 0289 – 0

[4] EMANOVSKÝ, P., *Algebra 3 (pro distanční studium). Skriptum PdF UP*, Olomouc, 2002. ISBN 80 – 244 – 0490 – 7

[5] KOŘÍNEK, V., *Základy algebry*, Praha : NČSAV , 1956.

[6] TREFILÍKOVÁ, Z. *Systém počítačové algebry Maxima* [online]. 2011 [cit. 2014-02-07]. Bakalářská práce. Masarykova univerzita, Přírodovědecká fakulta. Vedoucí práce Roman Plch. Dostupné z: `http://is.muni.cz/th/262630/prif_b/`

[7] Zlámalová, L. *Numerické metody pro hledání kořenů polynomu* [online]. 2006 [cit. 2014-02-07]. Bakalářská práce. Masarykova univerzita, Přírodovědecká fakulta. Vedoucí práce Jiří Zelinka. Dostupné z: `http://is.muni.cz/th/43682/prif_b/`

[8] *Maxima, a Computer Algebra System* [online]. [cit. 2014-02-07]. Dostupné z: `http://maxima.sourceforge.net/`