MASARYKOVA UNIVERZITA

PEDAGOGICKÁ FAKULTA

Katedra matematiky

MUNI
PED

Numerické metody pro řešení polynomických rovnic

Bakalářská práce

Brno 2019

Vedoucí práce:
RNDr. Břetislav Fajmon, Ph.D.

Autor práce:
Iveta Trombiková

Bibliografický záznam

TROMBIKOVÁ, Iveta *Numerické metody pro řešení polynomických rovnic*. Brno: Masarykova univerzita, Fakulta pedagogická, Katedra matematiky, 2019. Vedoucí bakalářské práce RNDr. Břetislav Fajmon, Ph.D.

Anotace

Bakalářská práce s názvem „Numerické metody pro řešení polynomických rovnic“ popisuje základní pojmy z oblasti polynomů a věnuje se vybraným numerickým metodám, jmenovitě metodě půlení intervalu, metodě prosté iterace a Newtonově metodě. Tato práce rovněž obsahuje programy pro jednotlivé numerické metody, díky nimž jsou pak metody porovnány a studenti mohou vidět jejich světlé či stinné stránky. Práce je navíc doplněna o webové aplikace pomáhající při řešení polynomických rovnic.

Annotation

The bachelor thesis “Numerical Methods for Polynomial Equations” describes basic terms from the field of polynomials and deals with chosen numerical methods, namely the bisection method, the simple iteration method and Newton method. The thesis also contains programmes for the numerical methods, thanks to which are methods compared and students can see their benefits or deficiencies. The thesis is supplemented with additional website’s applications which help to solve polynomial equations.

Klíčová slova

Polynomy, polynomické rovnice, numerické metody, kořen polynomu, Newtonova metoda, metoda prosté iterace, metoda půlení intervalu.

Keywords

Polynomials, polynomial equations, numerical methods, a root of the polynomial, Newton method, a simple iteration method, a bisection method.

Prohlášení

„Prohlašuji, že jsem bakalářskou práci zpracovala samostatně, s využitím pouze citovaných pramenů, dalších informací a zdrojů v souladu s Disciplinárním řádem pro studenty Pedagogické fakulty Masarykovy univerzity a se zákonem č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.“

V Brně dne 27. března 2019 Iveta Trombiková

Poděkování

Velké díky patří mému vedoucímu doktoru Břetislavu Fajmonovi, za jeho vřelý přístup, ochotu a trpělivost, s kterou vedl celou bakalářskou práci. Také bych na tomto místě chtěla poděkovat mé rodině a přátelům za podporu během psaní.

Obsah

Úvod

Polynomy jsou jedním ze základních kamenů matematiky a díky svému širokému uplatnění se s nimi setkávají lidé skoro každodenně. Když pomineme základní polynomické rovnice, které řeší snad každý z nás v různých situacích (třeba když si chceme spočítat cenu svého nákupu), tak bez polynomů by se neobešli například stavební inženýři při konstrukci budov, cest či třeba horských drah. Důležité uplatnění však nalezly i při modelování různých situací, například pro odhad ekonomického růstu nebo pro simulaci dopravy. Polynomy jako učební látka je předmětem žáků již na základních školách (při výpočtech jednoduchých polynomických rovnic), ale v celé šíři jsou zájmem hlavně studentů technicky zaměřených vysokých škol.

Jelikož je velké procento žáků, kteří se s polynomy a konkrétně také s řešením polynomických rovnic setkají, vznikla tato bakalářské práce, jejímž cílem bylo prozkoumat oblast polynomů, shrnout základní pojmy a vybrat několik metod pro výpočet polynomických rovnic, tyto metody popsat, *naimplementovat*[1] a porovnat jejich rychlost a účinnost.

Vše bylo realizováno ve třech kapitolách, které nesou názvy podle svého obsahu. V první kapitole „Polynomy“ jsou shrnuty základní definice a pojmy z oblasti polynomů, mnohdy doplněné o vzorové příklady. Čtenář zde tedy může najít například, co je to stupeň polynomu**,** derivace polynomu či vysvětlené metody jako je Hornerovo schéma.

Druhá kapitola se věnuje vybraným numerickým metodám, které jsou zde vysvětleny, popsány a jejich fungování ukázáno na vzorovém příkladě.

Poslední kapitola vysvětluje fungování programů vytvořených pro výpočet jednotlivých numerických metod a také skrze sérii několika příkladů porovnává a vyhodnocuje účinnost těchto metod.

Bakalářská práce byla tedy pojata tak, aby pomohla studentům matematiky (či jiných oborů) ve studiu polynomů a usnadnila jim práci při řešení polynomických rovnic.

[1] naprogramovat

1. POLYNOMY

Pro počítání s polynomickými rovnicemi je nutno nejdříve definovat základní pojmy z oblasti polynomů, které jsou uvedeny v následujících podkapitolách. Text je rovněž obohacen ukázkovými příklady, které slouží k lepší ilustraci látky. Materiál k napsání této kapitoly je čerpán z knih: (Horák, 1977), (Rosický, 2000), (Budínová, 2013), částečně také z přednášek předmětu Algebra 3 či výukových internetových zdrojů.

1.1 Základní pojmy

Na polynomy můžeme nahlížet dvojím způsobem. Můžeme je brát jako funkce jedné proměnné *x* nebo je definovat jako posloupnost.

Definice 1: Polynomem stupně *n* rozumíme funkci tvaru
$\boldsymbol{P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + ax + a_0}$, kde $a_0, a_1, \ldots\ a_n \in R, a_n \neq 0$.

Prvky $a_0, a_1, \ldots\ a_n$ se nazývají koeficienty polynomu. Člen a_n nazýváme řídícím koeficientem polynomu a koeficient a_0 **absolutním členem**[2] polynomu. Polynom, jehož všechny členy jsou rovny nule, nazýváme **nulovým polynomem** a značíme jej $\bar{o} = (0,0,\ldots,0)$. Polynomu, jehož absolutní člen je roven jedné a ostatní rovny nule, říkáme **jednotkový polynom** a značíme jej $\bar{\jmath} = (1,0,\ldots,0)$.

Poznámka: Nabízí se také varianta definovat polynomy pomocí posloupností jejich koeficientů. Například polynom $f(x) = 3x^4 - 2x^2 + 15$ je možné ztotožnit s posloupností (15, 0, 2, 0, 3) nebo polynom $g(x) = 2x^3 + x^2 - 2x + 1$ s posloupností $(1, -2, 1, 2)$. Místo konečných posloupností různých délek však můžeme uvažovat nekonečné posloupnosti o konečném počtu nenulových členů. Množinu takových to posloupnosti, které jsou prvky tělesa[3] T označíme T[x] a tuto množinu, pak budeme nazývat množinou polynomů nad tělesem T.

Na základní definici polynomu již můžeme stavět další důležité pojmy, jedním z nich je **stupeň polynomu**.

---

[2] Člen, jenž je pouhým koeficientem, neobsahuje žádné proměnné.

[3] Číselným tělesem rozumíme uspořádanou trojici $(T, +, \cdot)$, kde T je podmnožina množiny komplexních čísel C taková, že $0 \in T, 1 \in T$ a platí:

1. $\forall x, y \in T: x + y \in T \wedge x \cdot y \in T$ (je uzavřená vzhledem k operacím sčítání a násobení)
2. $\forall x \in T: (-1) \cdot x \in T$ (je uzavřená pro opačné prvky)

$\forall x \in T: x \neq 0 \Rightarrow \frac{1}{x} \in T$ (je uzavřený na převrácené hodnoty nenulových prvků)

Definice 2: Nechť f je nenulový polynom nad tělesem T. Pak největší nenulové přirozené číslo n v exponentu proměnné *x se* nazývá stupeň polynomu, značíme $st(f)$, když

$$f = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0, \qquad a_n \neq 0.$$

Polynomy, jež mají stupeň menší než jedna (nulový polynom a polynom stupně nula), se obvykle nazývají **konstantními polynomy**. Polynomy stupně jedna nazveme **polynomy lineárními**, stupně dva **kvadratickými** a stupně tři **kubickými**. Vyšší stupně polynomu již nemají své specifické názvy.

Příklad 1: Určete stupně polynomů $f_1(x) = x^3 + 3x^2 - 2x,\ f_2(x) = 12x$ a $f_3(x) = 0$.
Řešení: Polynom $f_1 = x^3 + 3x^2 - 2x$ je stupně 3, protože nejvyšší mocnina, která se v polynomu vyskytuje je u x^3, píšeme tedy $st(f_1) = 3$. Polynom $f_2 = 12x$ má nejvyšší mocninu jedna, proto $st(f_2) = 1$. U posledního polynomu $f_3 = 0$, je nevyšší mocninou nula, proto se jedná o nulový polynom a píšeme $st(f_3) = 0$.

1.2 Operace na množině polynomů

Už tedy víme, jak polynom vypadá, dále pro něj budeme definovat základní operace. Začneme s operacemi **sčítání** a **násobení**.

Definice 3: Nechť $f = (a_0, a_1, \ldots), g = (b_0, b_1, \ldots) \in$ T[x]. Pak:
Součet $f + g$ definujeme: $f + g = (a_0 + b_0, a_1 + b_1, \ldots)$,
Součin $f \cdot g$ definujeme: $f \cdot g = (c_0, c_1, c_2, \ldots)$, kde $c_0 = a_0 b_0;\ c_1 = a_1 b_0 + a_0 b_1;$ $c_2 = a_2 b_0 + a_1 b_1 + a_0 b_2$ atd.

Příklad 2: Jsou dány polynomy $f(x) = x^3 + 2x - 4$ a $g(x) = 2x^3 + 5$, jaký je jejich součet a součin?
Řešení:

$$f + g = (x^3 + 2x - 4) + (2x^3 + 5) = x^3 + 2x^3 + 2x - 4 + 5 = 3x^3 + 2x + 1$$

$$f \cdot g = (x^3 + 2x - 4) \cdot (2x^3 + 5) =$$
$$= x^3 \cdot 2x^3 + x^3 \cdot 5 + 2x \cdot 2x^3 + 2x \cdot 5 - 4 \cdot 2x^3 - 4 \cdot 5 =$$
$$= 2x^6 + 5x^3 + 4x^4 + 10x - 8x^3 - 20 =$$
$$= 2x^6 + 4x^4 - 3x^3 + 10x - 20$$

Z výše uvedené definice je zřejmé, že $f + g$ a $f \cdot g$ bude taktéž polynom. Z tohoto zjištění plyne Věta 1.

Věta 1: $(T[x], +, \cdot)$ je oborem integrity [4]pro každé těleso T.

Definice součtu a součinu nám umožňují dívat se na množinu polynomu jako na **vektorový prostor**.

Věta 2: Nechť $T_n[x]$ je množinou všech polynomů s takto definovanými operacemi sčítání a násoben, pak $(T_n[x], +, \cdot)$ tvoří vektorový prostor dimenze $n + 1$.

Ze základních operací ještě chybí definovat **dělení**. Níže je uvedena definice pro dělení dvou polynomů.

Definice 4: Nechť $f, g \in$ T[x]. Říkáme, že polynom g dělí polynom f, píšeme $g|f$, jestliže existuje polynom h, takový, že:

$$f = g \cdot h$$

V opačném případě říkáme, že polynom g nedělí polynom f a píšeme $g \nmid f$.

Z této definice pak plynou následující vlastnosti.

Věta 3: Nechť $f, g \in$ T[x], $f \neq 0, g \neq 0$. Pak platí:

1) $f|f$.
2) Pro všechna $u, v \in$ T[x] a všechna $f_1, f_2 \in$ T[x] platí: jestliže $g|f_1 \wedge g|f_2$, pak také $g|(f_1 u + f_2 v)$.
3) Nechť $h \in$ T[x], pak platí: jestliže $h|g \wedge g|f$, pak také $h|f$.
4) $g|f \Rightarrow st\ g \leq st\ f$.
5) $g|f \wedge f|g \Leftrightarrow$ existuje takový nenulový prvek $c \in T$, že platí $g = cf$.

Je však zřejmé, že dělení není vždy celočíselné, je tedy potřeba definovat **dělení polynomu se zbytkem**.

---

[4] Oborem integrity je taková struktura $(T[x], +, \cdot)$, pro kterou platí:

1) $(T[x], +)$ je komutativní grupa, což znamená, že:

- je uzavřená k operaci sčítání: $\forall x, y \in T[x] : x + y \in T$
- platí zde asociativní zákon $\forall x, y, z \in T[x]: (x + y) + z = x + (y + z)$
- obsahuje neutrální prvek (v našem případě 0): $\exists n \in T[x]: n + x = n \wedge x + n = n, \forall x \in T[x]$
- existují v ní prvky inverzní $\forall x \in T[x]: x + (-x) = 0$
- platí komutativní zákon: $\forall x, y, z \in T[x]: x * y = y * x$

2) $(T[x], \cdot)$ je komutativní monoid, jež je uzavřený k operaci násobení, platí zde asociativní zákon, obsahuje neutrální prvek (zde je to 1) a je komutativní
3) Platí distributivní zákony:
$\forall x, y, z \in T[x]: x \cdot (y + z) = (x \cdot y) + (x \cdot z) \wedge (y + x) \cdot z = (y \cdot z) + (x \cdot z)$
4) V $(T[x], +, \cdot)$ neexistují netriviální dělitelé nuly: $\forall x, y \in T[x]: x, y \neq 0$, pak $x \cdot y \neq 0$

Definice 5: Nechť f, $g \in$ T[x]. V T[x] lze provést dělení se zbytkem polynomu f polynomem g, jestliže existují polynomy $q, r \in$ T[x], takové, že:

(1) $f = g \cdot q + r$

(2) $st\,(r) < st\,(g)$

Polynom q se pak nazývá **podíl** a polynom r **zbytek** tohoto dělení.

Z této definice vyplývá, že pro $g = 0$ nelze provést dělení se zbytkem, protože podmínka (2) nebude splněna. Mohou tedy nastat situace, kdy dělení se zbytkem provést nelze, nebo podíl a zbytek nejsou jednoznačně určeny. Aby byla definice úplná doplníme ji o dvě dodatečné věty.

Věta 4: Nechť f, $g \in$ T[x]. Je-li vedoucí koeficient u polynomu g jednotkou tělesa T, pak v T[x] lze provést dělení se zbytkem polynomu f polynomem g.

Věta 5: Nechť T je těleso a $g \in$ T[x] je polynom, jehož vedoucí koeficient je různý od nuly. Pak pro libovolné $f \in$ T[x] existuje nejvýše jedna dvojce polynomů q, r tak, že

$$f = g \cdot q + r; \qquad st(r) < st(g).$$

Jak vypadá dělení dvou polynomů je ukázáno na následujícím příkladu.

Příklad 3: Jsou dány polynomy: $f(x) = 8x^8 + 16x^5 + 3x^3 - 7;\ g(x) = 4x^3 + 3$, určete jejich podíl.

Řešení:

$$(8x^8 + 10x^4 + 3x^3 - 7) \div (4x^4 + 3) = 2x^4 + 1 + \frac{3x^3 - 10}{4x^4 + 3}$$

$$\underline{-(8x^8 + 6x^4)}$$

$$4x^4 + 3x^3 - 7$$

$$\underline{-(4x^4 + 3)}$$

$$3x^3 - 10$$

Postup dělení je podobný jako u dělení celých čísel: vydělíme $8x^8 \div 4x^4 = 2x^4$, poté zpětně násobíme, $2x^4 \cdot 4x^4 = 8x^8$, $2x^4 \cdot 3 = 6x^4$, výsledný součin $8x^8 + 6x^4$ následně odečteme od dělence a postupujeme stejným způsobem, dokud není stupeň dělitele větší než stupeň dělence (polynom $3x^3 - 10$ je stupně 3 a stupeň dělence $st(g) = 4$). Tento polynom je pak zbytkem.

1.3 Hodnota a kořen polynomu

Pro práci s polynomy jsou důležitými pojmy **hodnota** a **kořen polynomu**.

Definice 6: Nechť R je okruh a $f = a_n x^n + a_{n-1}x^{n-1} + \cdots a_1 x + a_0$ je polynom z R[x], pak $f(c)$ je hodnota polynomu f v bodě c ∈ $\mathbb{R}$ jestliže:

$$a_n c^n + a_{n-1}c^{n-1} + \cdots + a_1 c + a_0 \in \mathrm{R}$$

Je-li $f(c) = 0$, pak se prvek c nazývá kořen polynomu f.

Poznámka: Z toho plyne, že každý prvek c ∈ $\mathbb{R}$ je kořenem nulového polynomu, a naopak nenulový polynom stupně nula nemá žádný kořen.

Pro hodnotu polynomu platí následující vztahy.

Věta 6: Nechť R je okruh, $f, g \in \mathrm{R}[x]$ a c ∈ R . Pak platí:

(1) Když $f = g$, pak i $f(c) = g(c)$

(2) $(f \pm g)(c) = f(c) \pm g(c)$

(3) $(f \cdot g)(c) = f(c) \cdot g(c)$

Příklad 4: Určete hodnotu polynomu $f(x) = x^4 - 2x^3 + 3x - 5$ pro $c_1 = 1$ a $c_2 = 2$.

Řešení: Počítáme-li hodnotu pro $c_1 = 1$, tak v polynomu f jednoduše dosadíme za x hodnotu 1. Výsledek je tedy -3. Pro $c_2 = 2$ postupujeme analogicky a výsledek je 1.

$$f(1) = 1^4 - 2 \cdot 1^3 + 3 \cdot 1 - 5 = -3$$

$$f(2) = 2^4 - 2 \cdot 2^3 + 3 \cdot 2 - 5 = 1$$

S pojmem hodnoty polynomu se váže následující věta.

Věta 7: Nechť R je okruh, pak c ∈ R je kořenem polynomu $f \in \mathrm{R}[x]$ právě když polynom $(x - c) \mid f$.

Tato věta pak dává prostor pro definici **násobnosti kořene**.

Definice 7: Nechť R je okruh, $f \in \mathrm{R}[x]$ a c je kořenem polynomu f, pak přirozené číslo k se nazývá násobnost kořene c, když platí:

(1) $(x - c)^k \mid f$

(2) $(x - c)^{k+1} \nmid f$

Pro $k = 1$ říkáme kořenu **jednoduchý**.

Například polynom $f(x) = x^3 - 6x^2 + 12x - 8$ lze rozložit na součin $f(x) = (x-2)(x-2)(x-2)$, z čehož plyne, že tento polynom má trojnásobný kořen $x = 2$.

Podíváme-li se podrobně na jednotlivé struktury, zjistíme, že množina racionálních čísel $\mathbb{Q}$ je tělesem, ale ne všechny polynomické rovnice v nich mají řešení. Například polynom $f = x^2 + 1$ má kořeny $k = \pm i$, což jsou kořeny, které patří až do tělesa komplexních čísel. Hledáme-li strukturu, v níž má každá polynomická rovnice řešení, musíme definovat **algebraicky uzavřené těleso**.

Definice 8: Těleso T je algebraicky uzavřené právě tehdy, když každý polynom $f \in T[x]$ stupně alespoň 1 má v tělese T alespoň jeden kořen.

Z tohoto zjištění plyne **základní věta algebry**. Ta má několik znění, které jsou svým významem ekvivalentní. Níže je uvedeno jedno její znění a ostatní věty jsou vyvozovány jako její důsledky.

Věta 8: (Základní věta algebry) Každý polynom s komplexními koeficienty, jehož stupeň je alespoň 1, má alespoň 1 komplexní kořen, tj. $\forall f \in \mathbb{C}[x]\ \exists c \in \mathbb{C}: f(c) = 0$.
Jinými slovy: Těleso komplexních čísel je algebraicky uzavřené (Čechová, 2013).

Věta 9: Nechť T je těleso a nenulový polynom $f \in T[x]$, pak má tento polynom nejvýše $n = st(f)$ kořenů, počítá-li se i s jejich násobností (Čechová, 2013).

Věta 10: Libovolný polynom stupně $n \geq 1$ s komplexními koeficienty lze rozložit na součin lineárních polynomů v $\mathbb{C}[x]$ (Čechová, 2013).

Věta 11: Nechť $\mathbb{C}$ je těleso komplexních čísel, polynom $f(x) \in \mathbb{C}[x]$ je stupně n a jeho kořeny jsou $x_1, x_2, \dots, x_n$. Pak lze tento polynom vyjádřit jako součin *normovaných*[5] polynomů a nenulové konstanty $a \in \mathbb{C}$: $f(x) = a(x - x_1)(x - x_2) \dots (x - x_n)$, kde $a = a_n$ je vedoucí koeficient polynomu $f(x)$ (Čechová, 2013).

[5] Polynom nazýváme normovaným, když se jedná o polynom nenulový a jeho vedoucí koeficient se rovná jedné.

1.4 Rozložitelnost, největší společný dělitel

Pro práci s polynomy je také důležitý pojem **rozložitelnost**, resp. **nerozložitelnost** neboli *reducibilita* resp. *ireducibilita* polynomu.

Definice 9: Polynom $f \in$ T[x] je reducibilní nad tělesem T, právě když existují polynomy $g, h \in$ T[x] tak, že platí $f = g \cdot h, st(g) \geq 1, st(h) \geq 1$.

Definice 10: Polynom $f \in$ T[x] nazýváme ireducibilní nad tělesem T, právě když $st(f) \geq 1$a neexistují polynomy $g, h \in$ T[x], tak aby $f = g \cdot h, st(g) \geq 1, st(h) \geq 1$.

Věta 12: Každý nekonstantní polynom f lze nad tělesem T rozložit na součin konstanty a normovaných ireducibilních polynomů.

Například polynom $f(x) = 3x^3 - 18x^2 + 36x - 24$ bychom mohli rozložit na $f(x) = 3 \cdot (x-2)(x-2)(x-2)$.

O reducibilitě nemusíme nutně uvažovat nad tělesem, ale i nad jednoduššími strukturami. Například polynom $f(x) = x^2 - 6x + 9$ je rozložitelný již na množině celých čísel. Jeho rozklad vypadá $f(x) = (x-3)(x-3)$ a kořeny tohoto polynomu by byl jeden dvojnásobný kořen $x = 3$, který patří do oboru celých čísel. Naopak polynom $g(x) = x^2 + 2$ je rozložitelný až nad tělesem komplexních čísel. Jeho rozklad je $g\,(x) = (x - \sqrt{2}i)(x + \sqrt{2}i)$ a kořeny $x_{1,2} = \pm\sqrt{2}i \in \mathbb{C}$.

Výše uvedené poznatky nám teď umožňují definovat jak **společného dělitele**, tak i **největšího společného dělitele** dvou polynomů.

Definice 11: Polynom $h \in$ T[x] je společný dělitel polynomů $f, g \in$ T[x] právě tehdy, když platí $h|g \ \wedge h|f$.

Definice 12: Nechť $f, g, d \in$ T[x]. Říkáme, že d je největší společný dělitel polynomů f, g (značíme $NSD(f, g)$), když platí:

1. $d \mid f \ \wedge d \mid g$.
2. Pro všechny polynomy $h \in$ T[x] platí: jestliže $h|f \ \wedge h|g$, pak $h|d$.

Pro výpočet největšího společného dělitele jsou níže uvedeny dvě metody, jednou z nich (tou častější) je **Eukleidův algoritmus**.

Věta 13 (Eukleidův algoritmus): Nechť $f,\ g \in$ T[x] a platí $st(f) \geq st(g) > 0$. Označme $g = g_0$, dělením $f \div g$ vytvoříme posloupnost:

$$f = q_1 \cdot g + g_1, \qquad st(g_1) < st(g_0)$$
$$g = q_2 \cdot g_1 + g_2, \qquad st(g_2) < st(g_1)$$
$$g_1 = q_3 \cdot g_2 + g_3, \qquad st(g_3) < st(g_2)$$
$$\dots$$
$$g_{n-3} = q_{n-1} \cdot g_{n-2} + g_{n-1}, \ st(g_{n-1}) < st(g_{n-2})$$
$$g_{n-2} = q_n \cdot g_{n-1} + g_n, \qquad st(g_n) < st(g_{n-1}),$$
$$q_0, q_1, \dots, q_n \in \mathbb{R}.$$

Postupujeme dokud $g_n = 0$. Největším společným dělitelem polynomů f, g je pak člen g_{n-1}, tedy $NSD(f,g)$ nalezneme jako poslední nenulový zbytek v této posloupnosti dělení polynomů.

Důkaz: Z poslední rovnice je zřejmé, že $g_{n-1}|g_{n-2}$, z předposlední rovnice zase, že $g_{n-1}|g_{n-3}$. Takto postupujeme, až z prvních dvou rovnic dostaneme vztahy $g_{n-1}|g$ a $g_{n-1}|f$, což znamená, že polynom g_{n-1} je společným dělitelem polynomů f a g. Zbývá dokázat, že polynom g_{n-1} je největším společným dělitelem f a g. Zvolme tedy d jako společný dělitel obou polynomů, z první rovnice je pak zřejmé, že $d|g_1$, tím pádem $d|g_2$ a tak dále, dokud nedojdeme ke vztahu $d|g_{n-1}$. Jelikož každý společný dělitel dělí g_{n-1}, je tento polynom největším společným dělitelem f a g.

Použití této metody je ukázáno na následujícím příkladu.

Příklad 4: Najděte největšího společného dělitele $f(x) = x^4 - 2x^3 - 2x^2 + 7x - 6$ a $g(x) = 2x^3 - 4x^2 - x + 2$.

Řešení: Podle definice vydělíme polynom $f(x)$ polynomem $g(x)$, protože $st(f) = 4$ a $st(g) = 3$, což znamená, že $st(f) > st(g)$. Pro následné lepší počítání s polynomy vynásobíme $f(x) \cdot 2$.

$$(2x^4 - 4x^3 - 4x^2 + 14x - 12) \div (2x^3 - 4x^2 - x + 2) = x + \frac{-3x^2 + 12x - 12}{2x^3 - 4x^2 - x + 2}$$
$$\underline{-(2x^4 - 4x^3 - x^2 + 2x)}$$
$$-3x^2 + 12x - 12$$

Z toho plyne, že $q_1(x) = x$, $g_1(x) = -3x^2 + 12x - 12$, tento polynom můžeme normovat na polynom $g_1(x) = x^2 - 4x + 4$, podle definice pokračujeme dělením $g(x)$ polynomem $g_1(x)$.

$(2x^3 - 4x^2 - x + 2) \div (x^2 - 4x + 4) = 2x + 4 + \dfrac{7x - 14}{x^2 - 4x + 4}$

$\underline{-(2x^3 - 8x^2 + 8x)}$

$4x^2 - 9x + 2$

$\underline{-(4x^2 - 16x + 16)}$

$7x - 14$

Pokračujeme analogicky $q_2(x) = 2x + 4$, $g_2(x) = 7x - 14$. Dělíme $g_1(x)$ normovaným polynomem $g_2(x)$.

$(x^2 - 4x + 4) \div (x - 2) = x - 2$

$\underline{-(x^2 - 2x)}$

$-2x + 4$

$\underline{-(-2x + 4)}$

0

$g_3(x) = 0$, největším společným dělitelem je tedy polynom $g_2(x) = x - 2$. $NSD(f, g) = c \cdot (x - 2)$, kde $c \in \mathbb{R}$.

Poznámka: Polynomy $g_1(x), g_2(x), \ldots g_n(x)$ normujeme, abychom předešli zbytečném počítáním se zlomky, v čemž se často chybuje. Největší společný dělitel je proto určen jednoznačně, až na vynásobení polynomu číslem c z tělesa T.

Největšího společného dělitele dvou polynomů je možno také spočítat pomocí rozkladů jednotlivých polynomů na součin, k tomu využijeme **Hornerovo schéma**.

Věta 14 (Hornerovo schéma): Nechť $c \in T$ je libovolné a $f(x), g(x) \in R[x]$, $st(f) = n \geq 1$, jsou tvaru:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0, \text{ kde } a_i \in R, a_n \neq 0,$$

$$g = x - c.$$

Pak existuje polynom q, $st(q) = n - 1$, $q(x) = b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + \cdots + b_0$, a platí

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = (x - c)(\mathrm{b}_{n-1} x^{n-1} + \cdots + b_0) + f(c)$$

(Fišnarová, 2008).

Tuto situaci vepisujeme do tabulky:

	a_n	a_{n-1}	…	a_1	a_0
c	b_{n-1}	b_{n-2}	…	b_0	$f(c)$

Tabulka 1: Tabulka pro Hornerovo schéma

Důkaz: Roznásobením pravých stran výše uvedených rovnic a jejich úpravami dostaneme následující vztahy:

$$a_n = b_{n-1}$$
$$a_{n-1} = b_{n-2} - cb_{n-1} \Rightarrow b_{n-2} = cb_{n-1} + a_{n-1}$$
$$a_{n-2} = b_{n-3} - cb_{n-2} \Rightarrow\ b_{n-3} = cb_{n-2} + a_{n-2}$$
$$\dots$$
$$a_0 = f(c) - cb_0 \Rightarrow f(c) = cb_0 + a_0$$

Dosadíme-li je za jednotlivé členy a_n, a_{n-1}, a_{n-2}, … , a_0 dostaneme:

$a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 = b_{n-1}x + (b_{n-3} - cb_{n-2})x^{n-1} + \cdots + f(c) - cb_0$

Členy na pravé straně roznásobíme $b_{n-1}x + \ b_{n-3}x^{n-1} - cb_{n-2}x^{n-1} + \cdots + f(c) - cb_0$ a následně vytkneme $x - c \ \Rightarrow (x-c)(b_{n-1}x^{n-1} + b_{n-2}x^{n-2}) + \cdots + \ f(c)$, což dokazuje, že $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 = (x-c)(b_{n-1}x^{n-1} + \cdots + b_0) + f(c)$ platí.

Jak vypadá práce s Hornerovým schématem v praxi, ukážeme na následujícím příkladu.

Příklad 5: Určete největšího společného dělitele polynomů $f(x) = x^4 + 5x^3 + 7x^2 + 5x + 6$ a $g(x) = x^4 - 1$.

Řešení: Polynomy se snažíme rozložit na součin polynomů nižšího stupně (nejlépe stupně jedna):

$$f(x) = x^4 + 5x^3 + 7x^2 + 5x + 6$$

	$1 = a_4$	$5 = a_3$	$7 = a_2$	$5 = a_1$	$6 = a_0$
-2	$1 = b_3$	$3 = b_2$	$1 = b_1$	$3 = b_0$	$0 = f(-2)$
-3	$1 = d_2$	$0 = d_1$	$1 = d_0$	$0 = f(-3)$	

Tabulka 2: Výpočet kořenu polynomu $f(x) = x^4 + 5x^3 + 7x^2 + 5x + 6$ pomocí Hornerova schématu

V Hornerově schématu vycházíme z námi vyjádřených vztahů, které jsou uvedeny výše. První koeficient a_4 (v tomto případě 1) opisujeme $a_4 = b_3$. Poté námi zvolené číslo $c = -2$ vynásobíme s b_3 a sečteme s koeficientem $a_3 = 5$, čímž dostaneme člen $b_2 = 3$, takto postupujeme s dalšími koeficienty. Pokud $c \cdot b_0 + a_0 = 0$, pak dané číslo $c = -2$ je kořenem polynom $f(x)$ a ten lze proto rozložit na součin polynomů $(x + 2)(\ x^3 + 3x^2 + x + 2)$.

Další kořeny nalezneme stejným způsobem, vstupní vektor koeficientu nového Hornerova schématu však tvoří polynom $f_1(x) = x^3 + 3x^2 + x + 2$, tj. hodnoty $b_{n-1}, b_{n-2}, \dots, b_0$, které jsme právě získali v předešlém kroku. Polynom stupně dvě

určený koeficienty d_2, d_1, d_0 obsahuje komplexní kořeny, proto je můžeme vypočítat pomocí **Vietových vztahů**[6] či vzorce pro výpočet kvadratické rovnice[7]. Rozklad $f(x)$ pak má tvar $f(x) = (x+2)(x+3)(x^2+1)$. Vietovy vztahy či vzorec pro kvadratické rovnice lze použít vždy, když se jedná o polynom stupně dva bez ohledu na to, jedná-li se o kořeny komplexní či nikoli. Výpočet je mnohdy rychlejší, protože nemusíme hledat číslo c.

Analogicky pokračujeme s druhým polynomem:

$$g(x) = x^4 - 1$$

Rozklad podle Hornerova schématu:

	1	0	0	0	-1
1	1	1	1	1	0
-1	1	0	1	0	

Tabulka 3: Hledání kořeny polynomu $g(x) = x^4 - 1$ pomocí Hornerova schématu

V tomto případě lze polynom rozložit také pomocí algebraických vzorců[8]. Jeho rozklad vypadá. $g(x) = (x+1)(x-1)(x^2+1)$. Největším společným dělitelem je pak součin všech polynomů, které mají oba polynomy totožné, což je (x^2+1). Tento polynom je určen jednoznačně až na násobek reálným číslem.

$$NSD(f,g) = \mathrm{c} \cdot (x^2+1), \mathrm{c} \in \mathbb{R}$$

Poznámka: Hornerovo schéma lze také využít k hledání koeficientů **Taylorova rozvoje**[9], když zvolíme $c = 1$.

Například pro polynom $f(x) = x^3 - 2x + 1$ vypadá rozklad podle Hornerova schématu následovně:

[6] Předpokládáme, že polynom je ve tvaru $ax^2 + bx + c$. Vietovy vztahy pak vypadají:

$$x_1 + x_2 = -\frac{b}{a};\ x_1 \cdot x_2 = \frac{c}{a}\ \text{(Hašek, 2018)}$$

[7] Předpokládáme, že polynom je ve tvaru $ax^2 + bx + c$ kořeny se poté vypočítají pomocí vzorce pro kvadratické rovnice

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

[8] Mezi základní algebraické vzorce patří: $(A+B)^2 = A^2 + 2AB + B^2$, $(A-B)^2 = A^2 - 2AB + B^2$, $A^2 - B^2 = (A-B) \cdot (A+B)$

[9] Nechť $f(x) \in R[x]$; $c \in R$. Taylorův polynom o středu c polynomu f má pak tvar:

$$f(x) = a_n(x-c)^n + \cdots + a_2(x-c)^2 + a_1(x-c) + a_0$$

A nechť $f \in R[x],\ st(f) = n \geq 1$; nechť $c \in R$. Pak existuje právě jeden Taylorův polynom o středu c polynomu f, a sice:

$$f(x) = f(c) + \frac{f'(c)}{1!}(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n$$

	1	0	-2	1
1	1	1	-1	0
1	1	2	1	
1	1	3		
1	1			

Tabulka 4: Vztah mezi Taylorovým rozvojem a Hornerovým schématem pro polynom $f(x) = x^3 - 2x + 1$

$c = x_0$ a čísla $0, 1, 3$ a 1 jsou koeficienty Taylorova rozvoje, který má tvar:

$$f(x) = (x-1) + 3(x-1)^2 + (x-1)^3$$

1.5 Derivace polynomu

Proto, aby bylo možné využít námi zvolené numerické metody pro výpočet polynomických rovnic, je důležité definovat **derivaci polynomu**. Tu využijeme následně v Newtonově metodě.

Definice 13: Nechť $f \in \mathbb{R}[x]$, kde $f = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$. Derivací polynomu f je pak polynom $f' \in \mathbb{R}[x]$ definován vztahem:

(1) Když $st(f) = 0$, pak $f' = 0$

(2) Když $st(f) > 0$, pak $f' = n \cdot a_n \cdot x^{n-1} + \cdots + 2a_2 x^1 + a_1$

Pro polynomy a jejich derivace pak platí následující vztahy

Věta 15: Nechť $f, g \in T[x]$, pak:

(1) $st(f) = n \geq 1$, pak $st(f') = n - 1$

(2) $(f \pm g)' = f' \pm g'$

(3) $(f_1 + \cdots + f_k)' = f_1' + \cdots + f_k{}'$

(4) $(f \cdot g)' = f' \cdot g + f \cdot g'$

(5) $(f_1 \cdot \ldots \cdot f_k)' = f_1' \cdot f_2 \cdot \ldots \cdot f_k + \cdots + f_1 \cdot \ldots \cdot f_{k-1} \cdot f_k{}'$

(6) $(f^k)' = k \cdot f' \cdot f^{k-1}, k \in \mathbb{N}$

Níže jsou uvedeny vztahy mezi kořeny polynomu f a kořeny derivovaného polynomu f'.

Věta 16: Nechť $f \in T[x], c \in T$ a $k \in \mathbb{N}$ přičemž $k \geq 2$. Je-li c k-násobným kořenem polynomu f, pak v polynomu f' je c $(k-1)$-násobným kořenem. Pokud je c jednoduchým kořenem v polynomu f, pak není kořenem v polynomu f'.

Důkaz: Nechť f je polynom stupně k a c je k-násobným kořenem polynomu f, pak podle vztahu (5) ve větě 15 můžeme derivaci toho polynomu zapsat jako:

$$[(x-x_1)^{l_1}(x-x_2)^{l_2}\dots(x-x_r)^{l_r}(x-c)^k]' =$$
$$k\cdot(x-x_1)^{l_1}(x-x_2)^{l_2}\dots(x-x_r)^{l_r}(x-c)^{k-1}+l_1\cdot(x-x_1)^{l_1-1}(x-x_2)^{l_2}\dots$$
$$(x-x_r)^{l_r}(x-c)^k+l_2\cdot(x-x_1)^{l_1}(x-x_2)^{l_2-1}\dots(x-x_r)^{l_r}(x-c)^k+\cdots$$
$$+l_r\cdot(x-x_1)^{l_1}(x-x_2)^{l_2}\dots(x-x_r)^{l_r-1}(x-c)^k.$$

Závorku $(x-c)^{k-1}$ lze vytknout ze všech členů, ale závorku $(x-c)^k$ už vytknout nelze. Tedy c je $(k-1)$-násobným kořenem polynomu f'. Tuto větu je možno využít při hledání násobných kořenů polynomu.

Příklad 6: V polynomu $f=x^5-ax^2-ax+1$ určete a tak, aby číslo $c=-1$ bylo alespoň dvojnásobným kořenem (Budínová, 2013).

Řešení: Nejdříve polynom zderivujeme $f'(x)=5x^4-2ax-a$ a určíme hodnoty v bodě $x=-1$ polynomů f a f'.

$$f(-1)=-1-a+a+1=0$$
$$f'(-1)=5+2a-a=5+a$$

Nyní řešíme rovnice: $0=0$ a $5+a=0$, z druhé rovnice nám plyne, že $a=-5$, což znamená, že hledaný polynom má tvar $f=x^5+5x^2+5x+1$. Správnost můžeme ověřit tak, že polynom $f=x^5+5x^2+5x+1$ vydělíme polynomem $(x+1)^2$, což se rovná x^3-2x^2+3x+1 a zbytek je nula, proto je $c=-1$ dvojnásobným kořenem polynomu $f=x^5+5x^2+5x+1$.

Věta 17: Nechť $f\in\mathbb{R}[x]$, $st(f)\geq 1$ a $f'\in\mathbb{R}[x]$ je derivací polynomu f, pak $g\in\mathbb{R}[x]$ je polynom, pro který platí:

$$f=NSD(f,f')\cdot g$$

Kořeny polynomu g a polynomu f jsou tedy totožné, ale polynom g obsahuje pouze kořeny jednoduché. To znamená, vydělíme-li polynom f největším společným dělitelem f a f' dostaneme polynom g, jenž skýtá stejné kořeny jako f, ale pouze násobnosti 1, což je velmi užitečné pro hledání kořenů polynomických rovnic, jak je ukázáno na následujícím příkladu.

Příklad 7: Najděte kořeny polynomu $f=16x^4-64x^3+120x^2-112x+49$.

Řešení: Nejdříve spočítáme derivaci polynomu f.

$$f'=64x^3-192x^2+240x-112$$

Následně spočítáme $NSD(f, f')$ pomocí Eukleidova algoritmu:

$$(16x^4 - 64x^3 + 120x^2 - 112x + 49) \div (64x^3 - 192x^2 + 240x - 112) = \frac{1}{4}x - \frac{1}{4} + \frac{12x^2 - 24x + 21}{64x^3 - 192x^2 + 240x - 112}$$

$$\underline{-(16x^4 - 48x^3 + 60x^2 - 28x)}$$
$$-16x^3 + 60x^2 - 84x + 49$$
$$\underline{-(-16x^3 + 48x^2 - 60x + 28)}$$
$$12x^2 - 24x + 21$$

Nově vzniklý polynom $p_1 = 12x^2 - 24x + 21$ můžeme normovat na polynom $p_1 = 4x^2 - 8x + 7$ a pokračujeme:

$$(64x^3 - 192x^2 + 240x - 112) \div (4x^2 - 8x + 7) = 16x - 16$$

$$\underline{-(64x^3 - 128x^2 + 112x)}$$
$$-64x^2 + 128x - 112$$
$$\underline{-(-64x^2 + 128x - 112)}$$
$$0$$

Největším společným dělitelem f a f' je tedy polynom $g_1 = 4x^2 - 8x + 7$. $NSD(f, f') = c \cdot (4x^2 - 8x + 7)$. Tímto polynomem vydělíme polynom f.

$$(16x^4 - 64x^3 + 120x^2 - 112x + 49) \div (4x^2 - 8x + 7) = 4x^2 - 8x + 7$$

$$\underline{-(16x^4 - 32x^3 + 28x^2)}$$
$$-32x^3 + 92x^2 - 112x + 49$$
$$\underline{-(-32x^3 + 64x^2 - 56x)}$$
$$28x^2 - 56x + 49$$
$$\underline{-(-28x^2 - 56x + 49)}$$
$$0$$

Polynom $g = 4x^2 - 8x + 7$ má tedy stejné kořeny jako polynom f jenom násobnosti 1. Vypočítáme tedy kořeny polynomu g a to buď pomocí Vietových vztahů nebo vzorcem pro výpočet kvadratické rovnice.

$$x_{1,2} = 1 \pm i \cdot \frac{\sqrt{3}}{2}$$

Kořeny původního polynomu f jsou tedy dva komplexně sdružené dvojnásobné kořeny $x_{1,2} = 1 + i \cdot \frac{\sqrt{3}}{2}$ a $x_{3,4} = 1 - i \cdot \frac{\sqrt{3}}{2}$.

1.6 Polynomické rovnice

Veškerá předchozí teorie nám umožňuje přejít k jádru bakalářské práce, a tím je řešení **polynomických rovnic**.

Definice 14 (Algebraická rovnice): Nechť $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ je polynom stupně n, kde $a_n \neq 0$. Algebraickou rovnici stupně n rozumíme rovnici ve tvaru $P_n(x) = 0$, tj. $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$.

Řešit algebraické (neboli polynomické) rovnice znamená hledat jejich kořeny ξ_i. Přičemž v rovnici je tolik kořenů, jaký je stupeň polynomu $P_n(x)$, počítáme-li s jejich násobností (Věta 9). Kořeny polynomických rovnic mohou být jak reálné, tak komplexní, přičemž když jsou všechny a_i reálná, tak jsou komplexní kořeny vždy komplexně sdružená čísla $\xi_1 = a + bi$ a $\xi_2 = a - bi$.

Hledáme-li pouze racionální kořeny, můžeme použít Hornerovo schéma (Věta 14), přičemž za c dosazujeme následující čísla.

Věta 18: Nechť polynom $f(x) \in T[x]$ je tvaru $f = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, $c = \frac{p}{q}$ je racionálním kořenem, tak že $p \in \mathbb{Z} \wedge p \neq 0, q \in \mathbb{N}$ a platí:

(1) Čísla p a q jsou nesoudělná.

(2) $p|a_o$ a $q|a_n$.

Příklad 8: Najděte kořeny polynomu $f(x) = 6x^3 + 7x^2 - x - 2$.

Řešení: Hledáme čísla p a q, tedy taková, že $p|2$ a $q|6$.

$$p = \pm 1;\ \pm 2 \qquad\qquad q = 1;\ 2; 3; 6$$

Kořeny rovnice pak mohou být čísla $c = \frac{p}{q}$:

$$c = \pm 1; \pm\frac{1}{2}; \pm\frac{1}{3}; \pm\frac{1}{6}; \pm 2; \pm\frac{2}{3}$$

Zdali jsou tato čísla kořeny rovnice ověříme pomocí Hornerova schématu.

	6	**7**	**−1**	**−2**
1	6	13	12	10
−1	6	1	−2	0
$\frac{1}{2}$	6	10	4	0
$-\frac{1}{2}$	6	4	−3	$-\frac{1}{2}$
$\frac{1}{3}$	6	9	2	$-\frac{4}{3}$
$-\frac{1}{3}$	6	5	$-\frac{8}{3}$	$-\frac{10}{9}$

	6	**7**	**−1**	**−2**
$\frac{1}{6}$	6	8	$\frac{1}{3}$	$-\frac{35}{18}$
$-\frac{1}{6}$	6	6	−2	$-\frac{5}{3}$
2	6	19	37	72
−2	6	−5	9	−20
$\frac{2}{3}$	6	11	$\frac{19}{3}$	$\frac{20}{9}$
$-\frac{2}{3}$	6	3	−3	0

Tabulka 5: Hornerovo schéma pro výpočet kořenů polynomu $f(x) = 6x^3 + 7x^2 - x - 2$

Zjistili jsme, že kořeny rovnice $6x^3 + 7x^2 - x - 2 = 0$ jsou $x_1 = -1, x_2 = \frac{1}{2}, x_3 = -\frac{2}{3}$.

Ve výjimečných případech lze k nalezení kořenů polynomu použít i algebraické vzorce či Vietovy vztahy (viz Příklad 5). Mezi další možné metody, které ještě v této kapitole nebyly zmíněny, jsou vzorce pro **kubické** a **kvartické rovnice** (rovnice 3. a 4. stupně). Pro rovnice vyššího řádu, než je čtyři, žádný univerzální vzorec neexistuje[10]. Vzorce jsou však již docela složité a užívají se jen zřídka, proto je zde uveden pouze vzorce pro výpočet kubických rovnic, které využívají tzv. **Cardanových vzorců**.

Věta 19: Nechť $a, b, c \in R$ a $x^3 + ax^2 + bx + c = 0$ je kubická rovnice, pro výpočet jejich kořenů nejdříve zavedeme substituci $x = y - \frac{a}{3}$, rovnice vypadá:

(1) $y^3 + py + q = 0,$

kde $p = b - \frac{a^2}{3}$ a $q = c + \frac{2a^3 - 9ab}{27}$. Předpokládejme, že $p \neq 0$. Dále zavedeme neznámé u, v splňující podmínky $u + v = y$. Dosazením podmínky do původní rovnice nám vznikne:

(2) $u^3 + v^3 + (3uv + p)(u + v) + q = 0$

a abychom anulovali závorku $(3uv + p)$, zavedeme $3uv + p = 0$. Dostaneme tak vztah $uv = -\frac{p}{3}$. Tento vztah umocníme na třetí $u^3 \cdot v^3 = \frac{p^3}{27}$ a spolu s upraveným vztahem (2)

[10] Neexistenci obecného vzorce pro řešení libovolné rovnice stupně vyššího než 4 dokázal Niels Henrik Abel ve svém díle „O algebraických rovnicích“ vydaného roku 1824.

$u^3 + v^3 = -q$ na ně můžeme nahlížet jako na Vietovy vzorce pro kořeny u^3, v^3 kvadratické rovnice.

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

Kořeny se budou rovnat:

$$(3) \quad u^3 = -\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}} \quad a \quad v^3 = -\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}$$

Diskriminantem rovnice označíme hodnotu $D_3 = \frac{q^2}{4} + \frac{p^3}{27}$. Rovnice tak budou mít devět řešení, ale kořeny jsou pouze ta, která vyhovují podmínce $v = -\frac{p}{3u}$. Každá ze tří hodnot u tedy odpovídá jedné hodnotě v.

Nechť čísla $\varepsilon_1, \varepsilon_2$ jsou komplexně sdružené kořeny rovnice $x^3 - 1 = 0$, pak se rovnají

$$\varepsilon_{1,2} = \frac{-1 \pm \sqrt{3}}{2}.$$

Kořeny rovnice (1) tak můžeme zapsat:

(4) $y_1 = u + v$

(5) $y_2 = \varepsilon_1 u + \varepsilon_2 v$

(6) $y_3 = \varepsilon_2 u + \varepsilon_1 v$

Po dosazení u, v do výše uvedených vztahů dostaneme:

$$y_1 = \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

$$y_2 = \varepsilon_1 \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \varepsilon_2 \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

$$y_3 = \varepsilon_2 \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \varepsilon_1 \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

(Slovák, Panák, Bulant, Návrat, & Veselý, 2013)

Příklad 9: Určete řešení rovnice $x^3 - 9x^2 + 36x - 28 = 0$ (Páv, 2010).

Řešení: Nejdříve chceme rovnici upravit do tvaru $y^3 + py + q = 0$, proto zavedeme substituci $x = y - \frac{a}{3}$. V zadání se $a = -9$, takže substituce má tvar $x = y + 3$.

$$(y + 3)^3 - 9 \cdot (y + 3)^2 + 36 \cdot (y + 3) - 28 = 0$$

$$y^3 + 9y^2 + 27y + 27 - 9y^2 - 54y - 81 + 36y + 108 - 28 = 0$$

$$y^3 + 9y + 26 = 0$$

Teď si vyjádříme u a v podle vztahu (3):

$$u = \sqrt[3]{-\frac{26}{2} + \sqrt{\frac{26^2}{4} + \frac{9^3}{27}}} = \sqrt[3]{-13 + \sqrt{13^2 + 3^3}} = \sqrt[3]{-13 + \sqrt{196}} = \sqrt[3]{-13 + 14} =$$

$$= \sqrt[3]{1}$$

Za u si můžeme zvolit přímo číslo 1 a odpovídající v poté dostaneme dosazením do podmínky $v = -\frac{p}{3u}$:

$$v = -\frac{9}{3 \cdot 1} = -3$$

Kořeny rovnice $y^3 + 9y + 26 = 0$ se tedy rovnají (podle vztahů (4)-(6)):

$$y_1 = 1 - 3 = -2$$

$$y_2 = \varepsilon_1 - 3\varepsilon_2 = 1 + 2i\sqrt{3}$$

$$y_3 = 1\varepsilon_2 - 3\varepsilon_1 = 1 - 2i\sqrt{3}$$

Máme-li kořeny námi upravené rovnice, dosadíme do substituce $x = y + 3$ a získáme kořeny naší původní rovnice $x^3 - 9x^2 + 36x - 28 = 0$.

$$x_1 = -2 + 3 = 1$$

$$x_2 = 1 + 2i\sqrt{3} + 3 = 4 + 2i\sqrt{3}$$

$$x_3 = 1 - 2i\sqrt{3} + 3 = 4 - 2i\sqrt{3}$$

Můžeme si povšimnout, že komplexní kořeny jsou vždy **komplexně sdružená čísla**.

I přesto, že jsme zde uvedli již řadu metod pro výpočet polynomických rovnic, stále je ještě mnoho případů, kdy nemůžeme kořeny určit, jedním z nich je následující příklad.

Příklad 10: Je dán polynom $f(x) = x^6 + x^5 + 4x^4 + 3x^3 + 5x^2 + 2x + 2$. Nalezněte jeho kořeny.

Řešení: Nejdříve zkusíme určit, zdali má polynom nějaké násobné kořeny (viz Příklad 7). Po provedení Eukleidova algoritmu dospějeme k závěru, že největším společným dělitelem polynomu $f = x^6 + x^5 + 4x^4 + 3x^3 + 5x^2 + 2x + 2$ a jeho derivace $f' = 6x^5 + 5x^4 + 16x^3 + 9x^2 + 10x$ je jedna, $NSD(f, f') = 1$, což znamená, že polynom nemá žádné násobné kořeny. Dále zkusíme najít racionální kořeny tohoto polynomu, pomocí Hornerova schématu (Věta 18).

	1	1	4	3	5	2	2
-1	1	-1	5	-2	7	-5	7
1	1	1	5	8	13	15	17
-2	1	-2	8	-13	31	-60	122
2	1	2	8	19	43	88	178

Tabulka 6: Hornerovo schéma pro výpočet kořenů polynomu $f(x) = x^6 + x^5 + 4x^4 + 3x^3 + 5x^2 + 2x + 2$

Z tabulky je zřejmé, že polynom neobsahuje žádné racionální kořeny, takže kořeny daného polynomu jsou iracionální nebo komplexní čísla. Jelikož je daný polynom stupně 6, nelze na něj uplatnit ani žádný vzorec, protože existují vzorce maximálně pro rovnice 4. stupně. Pro výpočet kořenů tedy musíme použít numerické metody, které jsou rozebrány v následující kapitole.

2. NUMERICKÉ METODY

V této kapitole se budeme zabývat řešením polynomických rovnic, které často nedokážeme vypočítat žádným jiným způsobem. Zabývejme se však nejprve úvahami, které naleznou reálné kořeny dané polynomické rovnice. Ne vždy je nalezení (zejména iracionálních) kořenů jednoduché a často dokážeme určit pouze jejich přibližnou hodnotu, je proto důležité nejdříve stanovit, v jakém intervalu se kořeny nachází (hledáme právě jeden kořen v daném intervalu). Tento proces se nazývá **separace kořenů rovnice**. Při hledání jednotlivých kořenů je užitečná následující věta.

Věta 20: Je-li funkce f spojitá[11]na intervalu $\langle a, b\rangle$, leží v tomto intervalu alespoň jeden kořen rovnice $f(x) = 0$, když platí:

$$f(a) \cdot f(b) < 0.$$

Tato věta předpokládá, že funkční hodnoty v krajních bodech daného intervalu mají opačná znaménka, z čehož plyne, že křivka, která je grafem levé strany rovnice $f(x) = 0$, někde protíná osu x. Jinými slovy, hodnota nějakého prvku se rovná 0 a jak je zmíněno v předchozí kapitole, kořenem rovnice je prvek c právě tehdy, když $f(c) = 0$. Tato věta však nezaručuje nalezení všech kořenů a jejich intervalů. Interval nemusí být totiž vždy zvolen vhodně, a i když věta nebude platit, může se v daném intervalu kořen (či více kořenů) nacházet, jak je vidět na následujícím příkladu.

Příklad 11: Nalezněte kořeny polynomu $f(x) = x^2 - 1$.
Řešení: Najít tyto kořeny znamená vyřešit rovnici $x^2 - 1 = 0$. Je zřejmé, že polynom bude mít dva kořeny (Věta 9), či jeden dvojnásobný kořen, protože $st(f) = 2$. Kořeny bychom mohli najít rozkladem na součin pomocí algebraických vzorců.

$$x^2 - 1 = 0$$
$$(x - 1)(x + 1) = 0$$

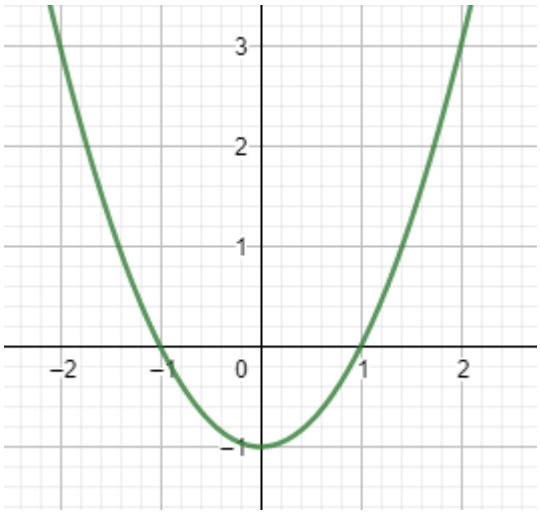
Kořeny daného polynomu jsou $x_1 = 1, x_2 = -1$.

Kdybychom se však řídili předchozí větou a zvolili interval $I = \langle -2, 2\rangle$, vyšlo by $f(-2) = 3$ a $f(2) = 3$. Jejich součin $3 \cdot 3 > 0$. Také dvojnásobné kořeny se vymykají pravdivosti této věty. Například graf polynomu $g(x) = x^2 + 2x + 1$ se osy x pouze

[11] Funkce $f(x)$ je spojitá v bodě a, když $\lim_{x \to a} f(x) = f(a)$. Funkce je spojitá na intervalu I, právě tehdy když je spojitá v každém bodě tohoto intervalu (Spojitost funkce, 2018).

dotkne, ale nenabývá nikde záporných hodnot, tudíž $f(a) \cdot f(b) \geq 0, \forall a, b \in R$. To je ilustrací faktu, že podmínka $f(a) \cdot f(b) < 0$ je u polynomu dostatečná, nikoli nutná pro existenci kořene polynomu f na intervalu $\langle a, b \rangle$.

Dalším důležitým vodítkem pro nalezení řešení polynomických rovnic je tedy základní znalost vlastností funkcí a zejména nakreslený graf. Kdybychom uplatnili tento postup na předchozí příklad (Příklad 11) a nejdříve nakreslili graf funkce $f = x^2 - 1$ (Obrázek 1), bylo by zřejmé, že námi zvolený interval $I = \langle -2, 2 \rangle$ je příliš velký. Navíc bychom v tomto příkladu mohli kořeny rovnice určit z grafu rovnou (graf totiž protíná osu x v bodech -1 a 1).



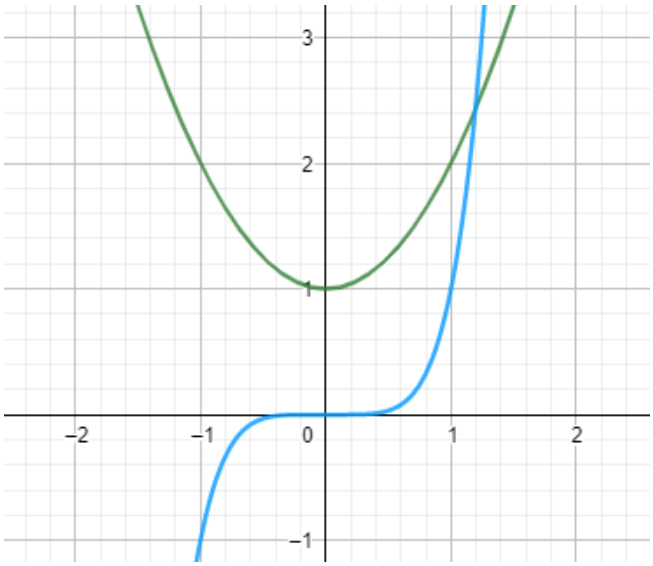
Obrázek 1: Graf funkce $f = x^2 - 1$

V některých případech je možné si úlohu upravit na tvar $f_1(x) = f_2(x)$, kde f_1 a f_2 jsou funkce a řešení úlohy je v místě, kde se grafy obou funkcí protnou. Tento případ je znázorněn na příkladu níže.

Příklad 12: Najděte řešení polynomické rovnice $x^5 - x^2 - 1 = 0$.

Řešení: Rovnici si upravíme na tvar $x^2 + 1 = x^5$, což znamená, že $f_1(x) = x^2 + 1$ a $f_2(x) = x^5$. Nakreslíme jejich grafy (Obrázek 2). Z grafu vyplývá, že řešením je x-ová souřadnice průsečíku grafů obou funkcí, tedy $x \in \langle 1,2 \rangle$.

S takto připraveným teoretickým základem již můžeme přistoupit k popisu jednotlivých numerických metod. Pro tuto bakalářskou práci jsou vybrány tři, jmenovitě: **metoda půlení intervalu**, **Newtonova metoda** a **metoda prosté iterace**. Každé zvlášť je níže věnována jedna podkapitola. Materiál pro tuto kapitolu je čerpán z učebního materiálu doktora Fajmona a magistry Růžičkové (2003), knihy Ivany Horové (1999) a knihy Canale a Chapra (2010).



Obrázek 2: Grafy funkcí $f_1(x) = x^2 + 1$ (zeleně) a $f_2(x) = x^5$ (modře)

2.1. Metoda půlení intervalu

Této metodě se také často přezdívá *bisekce* a je poměrně jednoduchá. Nejdříve volíme interval $\langle a, b\rangle$, pro nějž je daná funkce $f(x)$ spojitá a musí platit, že $f(a) \cdot f(b) < 0$, což znamená, že obsahuje kořen. Přitom v tomto intervalu musí ležet pouze jeden kořen dané funkce. Tento výchozí interval označme jako $\langle a_0, b_0\rangle$ a „rozpůlíme" jej. Získáme tak bod $x_0 = \frac{a_0+b_0}{2}$. Poté vybíráme jednu z následujících podmínek:

(1) Je-li $f(a_0) \cdot f(x_0) < 0$, leží kořen v intervalu $\langle a_0, x_0\rangle$ a položíme $a_1 = a_0$ a $b_1 = x_0$. Pro tento interval $\langle a_1, b_1\rangle$ postup opakujeme.

(2) Je-li $f(x_0) \cdot f(b_0) < 0$, leží kořen v intervalu $\langle x_0, b_0\rangle$ a položíme $a_1 = x_0$ a $b_1 = b_0$. Postup opakujeme pro interval $\langle a_1, b_1\rangle$.

(3) Je-li $f(x_0) = 0$, je x_0 kořenem rovnice.

Metoda bisekce je znázorněna na Obrázku 3. Tímto způsobem nám vznikne posloupnost intervalů: $\langle a_0, b_0\rangle \supset \langle a_1, b_1\rangle \ldots \supset \langle a_n, b_n\rangle \ldots$, kde, jak je uvedeno výše, platí $f(a_n) \cdot f(b_n) < 0$ a $n \in N_0$.

V půlení pokračujeme tak dlouho, dokud nenarazíme na kořen rovnice $f(x_n) = 0$ nebo se dostatečně nepřiblížíme toleranci ε a to tak, že pro nějaké k platí:

$$b_k - a_k < 2\varepsilon$$

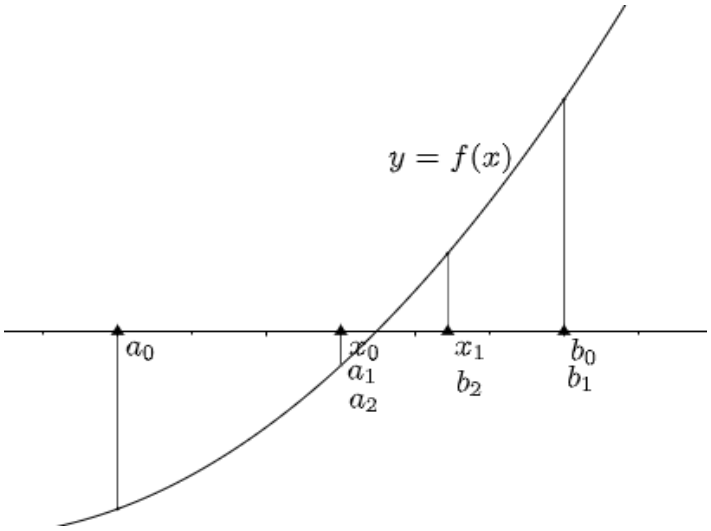
Přibližnou hodnotou námi hledaného kořene je střed posledního nalezeného intervalu.

$$x_k = \frac{a_k + b_k}{2}$$

Kořen se tak bude nacházet v posledním intervalu a bude se lišit od své přesné hodnoty nejvýše o polovinu své délky

$$|x_k - \xi| < \varepsilon.$$

Metoda půlení intervalu je velmi spolehlivá a lze ji použít vždy. Jediným problémem může být špatné určení intervalu, protože nachází-li se v něm více kořenů, nalezneme pouze jeden z nich. Nutno také dodat, že je tato metoda velmi pomalá, a proto se často využívá pouze k zúžení intervalu. Použití je ukázáno na následujícím příkladu.



Obrázek 3: Ukázka metody půlení intervalu

Příklad 13: Metodou bisekce najděte kořen funkce $f(x) = x^5 - x^2 - 1$ s přesností $\varepsilon = 0{,}01$.

Řešení: Tento příklad jsme řešili již výše (Příklad 12) a určili jsme, že kořen se nachází v intervalu $I \in \langle 1; 2\rangle$, máme tedy počáteční interval $\langle a_0, b_0\rangle$. Vypočítáme teď střed x_0 tohoto intervalu.

$$x_0 = \frac{a_0 + b_0}{2}$$

$$x_0 = \frac{1+2}{2} \Rightarrow x_0 = 1{,}5$$

Poté spočítáme funkční hodnoty v bodech a_0, b_0 a x_0.

$$f(a_0) = -1, f(b_0) = 27,\ f(x_0) = 4{,}34375$$

Teď musíme ověřit, která z podmínek (1) – (3) platí. Z uvedených hodnot vyplývá, že se jedná o podmínku (1) a to $f(a_k) \cdot f(x_k) < 0$. Položíme tedy $a_1 = 1$ a $b_1 = 1{,}5$ a dále počítáme s intervalem $\langle 1; 1{,}5\rangle$. Jelikož máme zvolenou přesnost výsledku, musíme zkontrolovat, zda platí podmínka $b_k - a_k < 2\varepsilon$, což po dosazení hodnot zjistíme, že neplatí $1 > 0{,}02$, proto pokračujeme dále. Interval $\langle 1; 1{,}5\rangle$ opět rozpůlíme a postupujeme stejným způsobem. Pro přehlednost můžeme řešení vepisovat do tabulky. Jelikož není

určující přesná funkční hodnota v jednotlivých bodech, stačí pro lepší přehled psát pouze znaménka + a −.

k	a_k	b_k	x_k	$f(a_k)$	$f(b_k)$	$f(x_k)$
0	1	2	1,5	−	+	+
1	1	1,5	1,25	−	+	+
2	1	1,25	1,125	−	+	−
3	1,125	1,25	1,1875	−	+	−
4	1,1875	1,25	1,21875	−	+	+
5	1,1875	1,21875	1,203125	−	+	+
6	1,1875	1,203125	1,1953125			

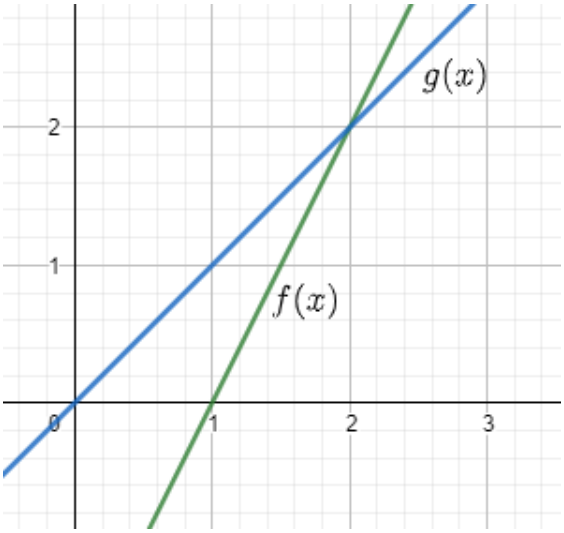
Tabulka 7: Hledání kořene polynomu $f(x) = x^5 - x^2 - 1$ metodou půlení intervalu

Takto pokračujeme až $k = 6$, kde už $b_6 - a_6 < 2 \cdot \varepsilon$, tedy $1{,}203125 - 1{,}1875 < 0{,}02$. Kořenem je pak $x_6 \doteq 1{,}195$.

2.2. *Metoda prosté iterace*

Iterační metody jsou založeny na řešení ekvivalentní úlohy $x = g(x)$. Funkce g se nazývá iterační funkce a místo kořenů původní rovnice hledáme **pevný bod funkce** g. Definujme tedy, co je to pevný bod.

Definice 15: Číslo ξ se nazývá pevný bod funkce $g(x)$, jestliže platí $g(\xi) = \xi$, tj. bod který funkce $g(x)$ zobrazí sama na sebe. Tento bod je pak řešením rovnice $g(x) = x$.

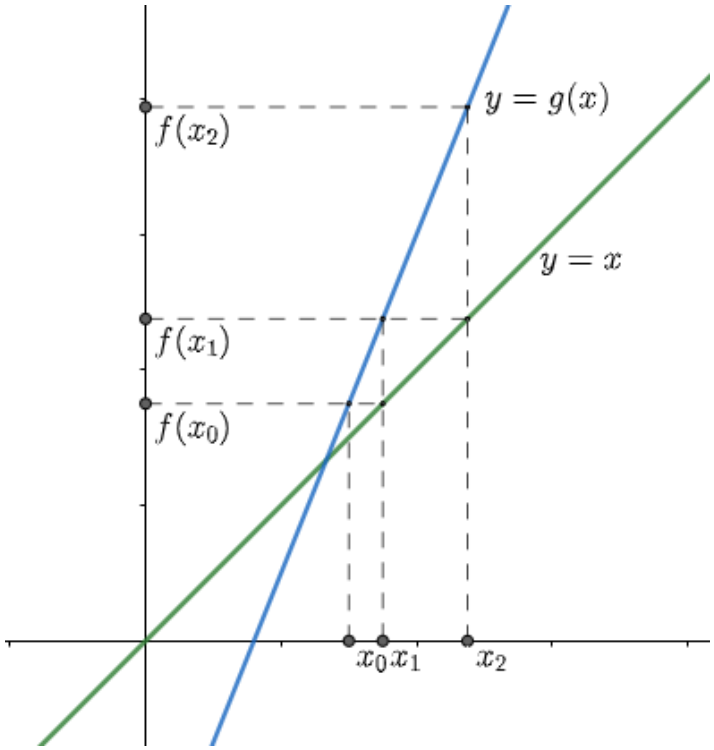


Obrázek 4: Graf funkcí $f(x) = 2x - 2$ a $g(x) = x$

Z definice vyplývá, že pevný bod funkce $y = g(x)$ je průsečíkem grafu funkce s přímkou $y = x$. Například funkce $f(x) = 2x - 2$ se protne s přímkou $y = x$ v bodě

$\xi = 2$ (Obrázek 4), který je tím pádem pevným bodem funkce $y = 2x - 2$. Platí tedy podmínka $g(\xi) = \xi$ a funkční hodnota v bodě $\xi = 2$ se rovná 2, $f(2) = 2$, bod se zobrazí sám na sebe.

Iterační metoda však může často divergovat a k pevnému bodu nemusíme vůbec dojít (Obrázek 5, např. pro iterační funkci $g(x) = \frac{5}{2}x - 2$ a počáteční bod $x_0 = 1{,}8$). Aby tedy iterační funkce $g(x)$ konvergovala, musí platit **Banachova věta o pevném bodě**.



Obrázek 5: Příklad divergence iterační funkce

Věta 21 (Banachova věta o pevném bodě): Nechť $g(x)$ je funkce na uzavřeném intervalu $\langle a, b\rangle$, která:

(1) zobrazuje interval $\langle a, b\rangle$ do sebe

(2) je diferencovatelná[12] na intervalu $\langle a, b\rangle$ a splňuje zde pro nějakou reálnou konstantu $L \in (0,1)$ nerovnost $|g'(x)| < L$

Pak má funkce $g(x)$ na intervalu $\langle a, b\rangle$ jediný pevný bod ξ. Je-li x_0 libovolný bod intervalu $\langle a, b\rangle$ a definujeme-li posloupnost $\{x_k\}_{k=0}^{\infty}$ vztahem $x_{k+1} = g(x_k)$, pak tato posloupnost konverguje k pevnému bodu x_*. Odhad chyby při aproximaci bodu ξ pomocí členů posloupnosti je

$$|x_{k+1} - \xi| \leq \frac{L}{1-L}|x_{k+1} - x_k| \text{ (Mařík, 2012a).}$$

[12] Diferencovatelná funkce je taková funkce, která má v daném bodě konečnou derivaci (Bárta, Pražák a kol., 2010)

Aplikuje-li se Banachova věta k-krát na funkci $g(x)$, nazýváme výsledek k-tá iterace funkce g. Například $g(g(x))$ je druhá iterace funkce g.

Protože nalezení L a ověření podmínky může být někdy obtížné, omezuje se často funkce na určitý počet kroků a z dosažených výsledků se pak určuje konvergence funkce. K zastavení iterace se používá podmínka $|x_{k+1} - x_k| < \varepsilon$, kde ε je námi zadaná tolerance. Fungování metody prosté iterace si ukážeme na následujícím příkladu.

Příklad 14: Metodou prosté iterace najděte kořen funkce $f(x) = x^5 + x^2 - 3$ s přesností $\varepsilon = 0{,}01$.

Řešení: Z grafu funkce $f(x) = x^5 + x^2 - 3$ (Obrázek 6) vyplývá, že kořen bude ležet v intervalu $I \in \langle 1; 2\rangle$ nebo přesněji $I \in \langle 1; 1{,}2\rangle$. Hledáme proto vhodnou iterační funkci. Možností, jak zvolit iterační funkci je nekonečně mnoho, například:

$$x^5 = 3 - x^2 \Rightarrow x = \pm\sqrt[5]{3 - x^2}$$

$$x^2 = 3 - x^5 \Rightarrow x = \pm\sqrt{3 - x^5}$$

My budeme počítat hned s prvním vztahem, $g(x) = \sqrt[5]{3 - x^2}$. Aby byla zajištěna konvergence funkce, potřebujeme nejdříve ověřit podmínky z Banachovy věty. Začneme podmínkou (2). Funkci $g(x)$ zderivujeme a hledáme maximum $|g'(x)|$ na intervalu $I = \langle 1; 1{,}2\rangle$.

$$g'(x) = \frac{1}{5}\frac{(-2x)}{(3 - x^2)^{\frac{4}{5}}} \qquad |g'(x)| = \frac{1}{5}\frac{2x}{(3 - x^2)^{\frac{4}{5}}}$$

Abychom našli maximum, funkci ještě jednou zderivujeme a položíme ji rovnou nule. Tak najdeme body podezřelé z extrému (stacionární body).

$$|g''(x)| = \frac{2 \cdot \sqrt[5]{(3 - x^2)^4} + \frac{16x^2}{5 \cdot \sqrt[5]{(3 - x^2)}}}{5 \cdot \sqrt[5]{(3 - x^2)^9}} = 0$$

Řešením této rovnice jsou komplexní kořeny ($x_{1,2} = \pm i\sqrt{5}$), proto funkce $|g'(x)|$ nemá lokální extrémy. Na intervalu $I = \langle 1; 1{,}2\rangle$ je funkce spojitá, tím pádem svého maxima nabývá v jednom z krajích bodů intervalu (Weierstrassova věta[13]). Dosazením krajních bodů do předpisu funkce $|g'(x)|$ a porovnáním těchto funkčních hodnot zjistíme, že je funkce na tomto intervalu rostoucí, tj. hledané maximum se nachází v bodě $x = 1{,}2$.

[13] Nechť funkce $f(x)$ je spojitá na uzavřeném intervalu $\langle a, b\rangle$. Potom je na tomto intervalu ohraničená a nabývá zde své největší a nejmenší hodnoty, tj. existují čísla $x_1, x_2 \in \langle a, b\rangle$ s vlastností $f(x_1) \leq f(x) \leq f(x_2)$, pro všechna $x \in \langle a, b\rangle$. (Mařík, 2012b)

$$|g'(1)| = \frac{1}{5}\frac{2 \cdot 1}{(3-1^2)^{\frac{4}{5}}} = 0{,}2297 \quad < \quad |g'(1{,}2)| = \frac{1}{5}\frac{2 \cdot 1{,}2}{(3-1{,}2^2)^{\frac{4}{5}}} = 0{,}3363$$

Podmínka (2) tak platí např. pro $L = 0{,}4$.

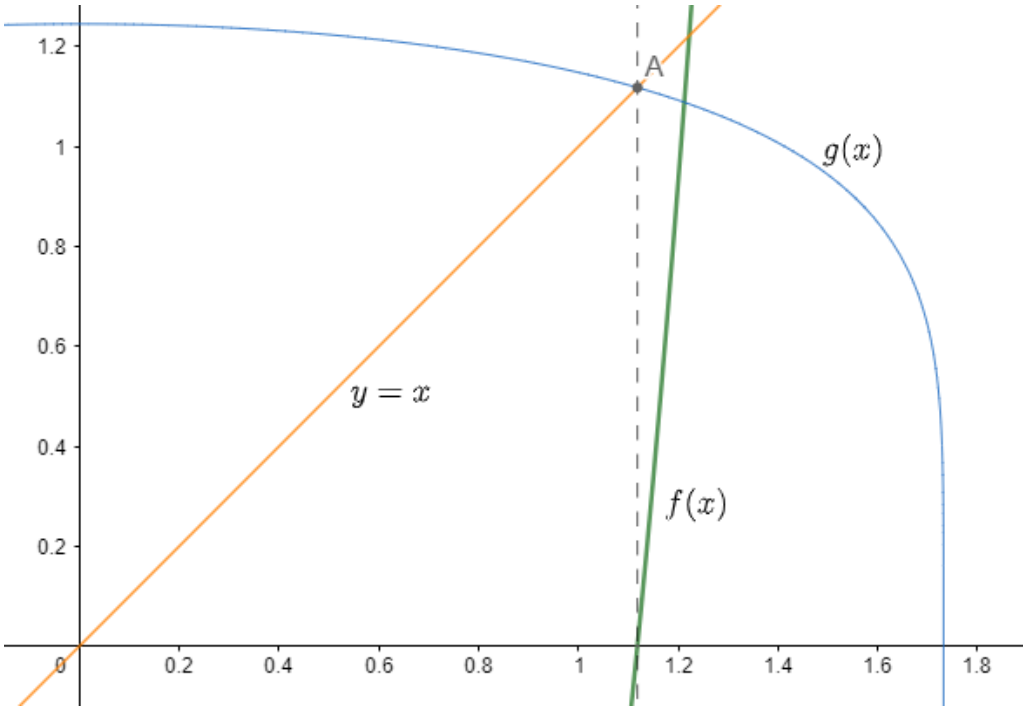
Dále ověříme Podmínku (1). Funkční hodnoty v krajích bodech jsou $g(1) = 1{,}148698$ a dále $g(1{,}2) = 1{,}093012$. Tedy samotná funkce $g(x)$ je klesající a krajní hodnoty se zobrazí do intervalu $\langle 1; 1{,}2\rangle$, tudíž tato funkce do tohoto intervalu zobrazuje jakoukoli hodnotu z tohoto intervalu. Celkově Banachova věta platí pro funkci *g(x)* na intervalu $I = \langle 1; 1{,}2\rangle$ pro jakékoli $x_0 \in \langle 1; 1{,}2\rangle$. Budeme tedy aproximovat podle předpisu $g(x) = \sqrt[5]{3-x^2}$ a zvolíme $x_0 = 1$. V dalším kroku dosadíme počáteční aproximaci do předpisu a vypočítáme x_1, $x_1 = \sqrt[5]{3-1^2} \doteq 1{,}148698355$.

$$x_2 = \sqrt[5]{3-1{,}148698355} \Rightarrow x_2 \doteq 1{,}109397785$$

$$x_3 = \sqrt[5]{3-1{,}109397785^2} \Rightarrow x_3 \doteq 1{,}120874995$$

$$x_4 = \sqrt[5]{3-1{,}120874995^2} \Rightarrow x_3 \doteq 1{,}117612697$$

Výpočet již můžeme zastavit, protože je splněna podmínka $|x_k - x_{k-1}| < \varepsilon$, tedy $|1{,}117612697 - 1{,}120874995| < 0{,}01$. Hledaným kořenem je $x_4 \doteq 1{,}1176$.



Obrázek 6: Graf funkcí $f(x) = x^5 + x^2 - 3$, $g(x) = \sqrt[5]{3-x^2}$ a $y = x$

Podíváme-li se se na grafy funkcí $g(x) = \sqrt[5]{3-x^2}$ a $y = x$ (Obrázek 6), je vidět, že námi nalezený bod $x_4 \doteq 1{,}1176$ odpovídá v grafu bodu A, proto se opravdu jedná o pevný bod funkce.

2.3. *Newtonova metoda (metoda tečen)*

Jelikož je mnohdy těžké určit správnou iterační funkci, zavedla se Newtonova metoda, která je speciálním případem iterační metody a pro svoji účelnost se stala jednou z nejrozšířenějších a nejpoužívanějších numerických metod.

Definice 16: Nechť f je iterační funkce a rovnice $f(x) = 0$ má jednoduchý kořen ξ, tj. $f'(\xi) \neq 0$. Pak pro funkci

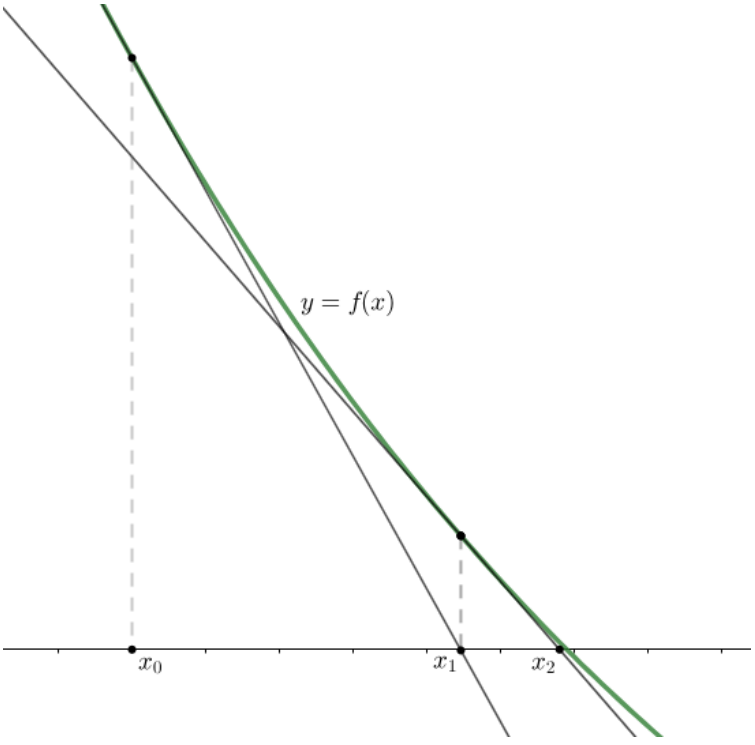
$$g(x) = x - \frac{f(x)}{f'(x)}$$

je ξ pevným bodem a iterační metoda určena touto funkcí ve tvaru

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad f'(x^k) \neq 0, k = 0,1, \dots$$

se nazývá Newtonova metoda.

Newtonovu metodu můžeme rovněž popsat graficky, díky čemuž můžeme tuto metodu nazvat také **metodou tečen** (Obrázek 7). Nejdříve si zvolíme počáteční



Obrázek 7: Newtonova metoda

aproximaci kořene x_0 a bodem $[x_0, f(x_0)]$ vedeme tečnu ke grafu funkce f. Průsečík dané tečny s osou x označíme x_1 a tímto bodem $[x_1, f(x_1)]$ opět vedeme tečnu ke grafu funkce f, čímž vznikne průsečík x_2. Postup opakujeme, dokud není splněna podmínka $|x_k - x_{k-1}| < \varepsilon$. Kořenem ξ je pak bod x_{k-1}.

Newtonovu metodu můžeme také odvodit pomocí Taylorova rozvoje (víc níže), který nám zároveň přiblíží, jakým tempem metoda konverguje.

$$f(x_{k+1}) = f(x_k) + f'(x_k)(x_{k+1} - x_k) + \frac{f''(\xi)}{2!}(x_{k+1} - x_k)^2,$$

kde ξ leží v intervalu od x_k do x_{k+1}. Ukončíme-li rozvoj po první derivaci dostaneme rovnici

$$f(x_{k+1}) \cong f(x_k) + f'(x_k)(x_{k+1} - x_k)$$

V průsečíku s osou x se bude $f(x_{k+1}) = 0$, rovnici tak přepíšeme a následně upravíme do již známého tvaru:

$$f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Taylorův rozvoj může být rovněž použit pro odhad chyby k-té aproximace kořene získaného Newtonovou metodou.

Věta 22: Nechť $x_k \in I \wedge \xi \in I$ a funkce $f(x)$ má na tomto intervalu I druhou derivaci, pak

$$(1) \quad |\xi - x_k| \leq \frac{M}{2m}(x_k - x_{k-1})^2$$

$$(2) \quad |\xi - x_k| \leq \frac{M}{2m}(\xi - x_{k-1})^2,$$

kde $M = \max |f''(x)|$ a $m = \min |f'(x)|$ pro $x \in I$.

Stejně jako iterační metody i Newtonova metoda nemusí vždy konvergovat. Většinou je to zapříčiněno špatnou volbou počáteční aproximace x_0. Uveďme tedy podmínky, které zajišťují konvergenci metody.

Věta 23 (Fourierova podmínka): Nechť má rovnice $f(x) = 0$ v intervalu $\langle a, b\rangle$ jediný reálný kořen a nechť f', f'' mají spojité derivace, které na intervalu $\langle a, b\rangle$ nemění znaménko, přičemž $f'(x) \neq 0, \forall x \in \langle a, b\rangle$. Počáteční aproximace x_0 je tedy ten z krajních bodů a, b, ve kterém je znaménko funkce stejné jako znaménko f'' na intervalu $\langle a, b\rangle$. Newtonova metoda pak bude konvergovat.

Příklad 15: Newtonovou metodou najděte kořen funkce $f(x) = x^5 - x^2 - 1$ s přesností $\varepsilon = 0{,}01$.

Řešení: Již víme, že kořen funkce bude ležet v intervalu $I \in \langle 1,2\rangle$ (Příklad 12). Teď je třeba správně zvolit počáteční aproximaci x_0. Ověříme tedy Fourierovu podmínku (Věta 23). Nejdříve určíme, jaká jsou znaménka v krajních bodech intervalu.

$$f(1) = 1^5 - 1^2 - 1 \;\Rightarrow\; f(1) = -1$$
$$f(2) = 2^5 - 2^2 - 1 \;\Rightarrow\; f(2) = 27$$

Dále potřebujeme znaménka porovnat s druhou derivací funkce $f(x)$, proto funkci dvakrát zderivujeme.

$$f'(x) = 5x^4 - 2x$$
$$f''(x) = 20x^3 - 2$$

Následně opět dosadíme krajní body intervalu a porovnáme získaná znaménka.

$$f''(1) = 20 \cdot 1^3 - 2 \;\Rightarrow\; f''(1) = 18$$
$$f''(2) = 20 \cdot 2^3 - 2 \;\Rightarrow\; f''(2) = 158$$

Za počáteční aproximaci x_0 zvolíme bod $x = 2$, protože znaménka jsou v obou případech kladná a funkce je na intervalu $I \in \langle 1,2\rangle$ konvexní. Nyní již můžeme začít počítat, předpis tedy vypadá (Definice 16).

$$x_{k+1} = x_k - \frac{{x_k}^5 - {x_k}^2 - 1}{5{x_k}^4 - 2x_k}$$

Dosadíme první aproximaci $x_0 = 2$ a pak pokračujeme, dokud není splněna podmínka $|x_k - x_{k-1}| < \varepsilon$.

$$x_1 = 2 - \frac{2^5 - 2^2 - 1}{5 \cdot 2^4 - 2 \cdot 2} \Rightarrow x_1 \doteq 1{,}644737$$

$$x_2 = 1{,}645 - \frac{1{,}644737^5 - 1{,}644737^2 - 1}{5 \cdot 1{,}644737^4 - 2 \cdot 1{,}644737} \Rightarrow x_2 \doteq 1{,}394561$$

$$x_3 = 1{,}395 - \frac{1{,}394561^5 - 1{,}394561^2 - 1}{5 \cdot 1{,}394561^4 - 2 \cdot 1{,}394561} \Rightarrow x_3 \doteq 1{,}250053$$

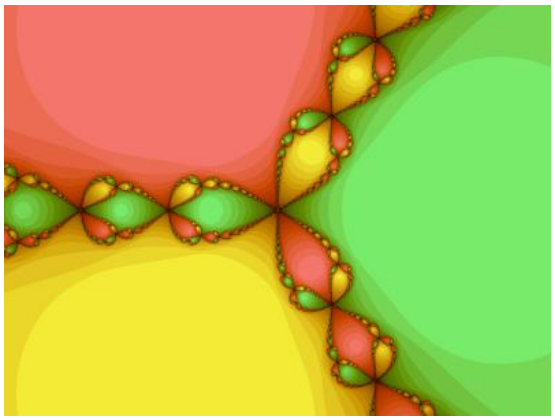
$$x_4 = 1{,}250 - \frac{1{,}250053^5 - 1{,}250053^2 - 1}{5 \cdot 1{,}250053^4 - 2 \cdot 1{,}250053} \Rightarrow x_4 \doteq 1{,}199608$$

$$x_5 = 1{,}2 - \frac{1{,}199608^5 - 1{,}199608^2 - 1}{5 \cdot 1{,}199608^4 - 2 \cdot 1{,}199608} \Rightarrow x_5 \doteq 1{,}193926$$

Jelikož $|1{,}199608 - 1{,}193926| < 0{,}01$ kořenem je $x_5 \doteq 1{,}1939$.

Newtonovu metodu můžeme rovněž použít pro výpočet rovnic s komplexními kořeny $f(z) = 0$. Je však nutné zvolit počáteční aproximaci z_0 komplexní, abychom došli ke komplexnímu kořenu. K jakému kořenu dojdeme, záleží na počáteční aproximaci z_0. Obarvením komplexní roviny podle zadané počáteční aproximace (toutéž barvou

obarvíme ty, jež vedou k nalezení stejného kořene) se vytvoří *fraktál*[14](Obrázek 8 a 9 (Tišnovský, 2006)).



Obrázek 8: Newtonova fraktální množina polynomu $z^3 - 1 = 0$

Obrázek 9: Newtonova fraktální množina polynomu $z^8 - 1 = 0$

Na konci první kapitoly byl zmíněn příklad (Příklad 10), jež jsme s dosavadní znalostí metod nedokázali vypočítat. Teď však by nalezení jeho kořenů neměl být problém.

Příklad 17: Nalezněte kořeny polynomu $f(x) = x^6 + x^5 + 4x^4 + 3x^3 + 5x^2 + 2x + 2$ s přesností $\varepsilon = 0{,}01$.

Řešení: Již víme, že tento polynom nemá žádné racionální kořeny, takže se bude jednat buďto o kořeny iracionální nebo komplexní. Předpokládejme, že námi hledaný kořen je komplexní číslo, použijeme proto Newtonovu metodu.

Nejdříve polynom zderivujeme.

$$f'(x) = 6x^5 + 5x^4 + 16x^3 + 9x^2 + 10x + 2$$

Předpis pro k-tou aproximaci vypadá:

$$x_{k+1} = x_k - \frac{{x_k}^6 + {x_k}^5 + 4{x_k}^4 + 3{x_k}^3 + 5{x_k}^2 + 2x_k + 2}{6{x_k}^5 + 5{x_k}^4 + 16{x_k}^3 + 9{x_k}^2 + 10x_k + 2}$$

Dále určíme počáteční aproximaci $x_0 = 1 + i$. Další aproximace jsou tedy:

$$x_1 = 1 + i - \frac{(1+i)^6 + (1+i)^5 + 4(1+i)^4 + 3(1+i)^3 + 5(1+i)^2 + 2(1+i) + 2}{6(1+i)^5 + 5(1+i)^4 + 16(1+i)^3 + 9(1+i)^2 + 10(1+i) + 2}$$

$$x_1 = 0{,}698813 + 0{,}924332i$$

$$x_2 = 0{,}435852 + 0{,}892621i$$

$$x_3 = 0{,}219409 + 0{,}905221i$$

$$x_4 = 0{,}071556 + 0{,}948134i$$

[14] Fraktál je geometrický útvar, ve kterém se stále opakuje určitý motiv či tvar.

$$x_5 = 0{,}00809 + 0{,}987446i$$

$$x_6 = -0{,}00008 + 0{,}99549i$$

Zde zastavíme výpočet, protože $|x_6 - x_5| < 0{,}01$, tudíž kořenem je $x \doteq i$. Protože komplexní kořeny jsou vždy komplexně sdružená čísla, je zřejmé, že druhý kořen se bude rovnat $x_2 = -i$.

Další kořeny je možné najít různými způsoby. Můžeme například opět použít Newtonovu metodu, avšak pro jinou počáteční aproximaci. Například při zadání $x_0 = 1 + \sqrt{2}i$ dostaneme, že dalšími kořeny jsou opět komplexně sdružená čísla $x_{3,4} = \pm\sqrt{2}i$. Poté opět můžeme pokračovat Newtonovou metodou nebo vydělíme polynom $f(x)$ výše vypočítanými kořeny a dostaneme:

$$(x^6 + x^5 + 4x^4 + 3x^3 + 5x^2 + 2x + 2) \div [(x^2 + 1) \cdot (x^2 + 2)] = x^2 + x + 1$$

Tento polynom stupně 2 již lehce spočítáme pomocí vzorce pro výpočet kořenů kvadratické rovnice.

$$x_{5,6} = \frac{-1 \pm \sqrt{1 - 4 \cdot 1 \cdot 1}}{2 \cdot 1} \Rightarrow x_5 = -\frac{1}{2} + i \cdot \frac{\sqrt{3}}{2};\ x_6 = -\frac{1}{2} - i \cdot \frac{\sqrt{3}}{2}$$

S pomocí numerických metod tedy dokážeme vypočítat kořeny jakéhokoli polynomu.

3. PROGRAMY A VÝSLEDKY

Pro programování numerických metod existuje několik možností, jako je například MATLAB, jazyk S či jazyk R. MATLAB® je programovací prostředí určeno většinou pro akademickou sféru, programuje se v něm ve stejnojmenném jazyce MATLAB. Tento název vznikl zkrácením slov *matrix laboratory* („maticová laboratoř“) a je založen na maticích. MATLAB tak dokáže počítat s maticemi, vykreslit 2D či 3D grafy funkcí, vytvořit počítačovou simulaci, analyzovat a interpretovat data či vytvářet nové aplikace. Nejdříve byl využíván pouze pro matematické výpočty, dnes je však rozšířeným jazykem, který se uplatňuje v mnoha sférách.

Naproti tomu je jazyk S statistický jazyk. Byl vyvinut v Bellových laboratořích Johnem Chambersem v roce 1976 (Becker, 1994). Jazyk S je i přes několik svých aktualizací dnes již zastaralý a málo používaný. Vyvinul se z něj však jazyk R, jež je určený zejména pro analýzu dat a jejich grafické vykreslení. V základní sestavě obsahuje mnoho statistických funkcí, ale může být použit i k řešení systému lineárních rovnic, maticových výpočtů či psaní vlastních funkcí a skriptů. Vidíme, že jazyk R a MATLAB jsou si v mnoha ohledech podobné, avšak na rozdíl od MATLABu je jazyk R volně dostupný, a proto byl použit pro tuto bakalářskou práci. Programy jsou psány v aplikaci RStudio, jež slouží jako grafické vývojové prostředí pro jazyk R.

3.1. Představení programů

V následujících podkapitolách jsou popsány programy pro výpočet numerických metod uvedených v Kapitole 2. Jedná se o programy: „Půlení intervalu“, „Metoda prosté iterace“, „Newtonova metoda pro reálné kořeny“ a „Newtonova metoda pro komplexní kořeny“. Kódy všech programů jsou vloženy do příloh.

3.1.1. Metoda půlení intervalu

Kód pro tuto metodu se skládá z několika částí. Ve funkci *polynom_vzor* (Obrázek 10) je vložen ukázkový polynom. Ten je zde jako příklad pro uživatele, který se seznamuje s fungováním programu. Jeho definování přímo ve zdrojovém kódu programu usnadňuje volání funkce, protože stačí zadat požadovaný interval (stačí do konzole napsat například *puleni (1, 2)* a program provede půlení vzorového polynomu v intervalu $I = \langle 1,2 \rangle$).

```
3   #vzorový polynom, počítáme v intervalu I=<0,1>
4   polynom_vzor <- function(x) {
5     return(x ^ 3 + x - 1)
6   }
```

Obrázek 10: Kód vzorového polynomu

Následuje funkce *puleni*, která implementuje algoritmus metody půlení intervalu. Při zadávaní je možno využít vzorového polynomu, nebo zadat svůj vlastní. Ten zadáváme například ve tvaru: *puleni (-2, 0, polynom = x ^ 3 + x - 1)*. Metodu rovněž můžeme ohraničit na určitý počet kroků (Obrázek 11 a 12), při nastavení *i_max = 0*, je počet kroků zanedbán a program skončí, až dosáhne zadané (či předdefinované) přesnosti ε (v programu označeno jako e).

```
75      #ověření, zdali jsme dosáhli určeného počtu kroků
76      i <- i + 1
77      if(i_max > 0 && i > i_max){
78        print(paste(sep = "", "Kořen s danou přesností nenalezen",
79                    ", počet kroků=", i - 1))
80        break
81      }
```

Obrázek 11: Zastavení průběhu metody půlení intervalu po určitém počtu kroků a ignorování kroků při nastavení i_max=0

```
> puleni(0, 1, i_max = 4)
[1] "0: s=0,5"
[1] "1: s=0,75"
[1] "2: s=0,625"
[1] "3: s=0,6875"
[1] "4: s=0,65625"
[1] "Kořen s danou přesností nenalezen, počet kroků=4"
>
```

Obrázek 12: Půlení intervalu, příklad na omezený počet kroků

Funkce pak kontroluje několik podmínek, než dojde k samotnému půlení. Nejdříve zkontroluje, zdali je interval zadán vzestupně, není-li tomu tak, hodnoty v intervalech vymění (Obrázek 13 a 14). Dále ověřuje, zdali se v intervalu nachází kořen daného polynomu (Obrázek 15 a 16) a také, zda není jeden z krajních bodů kořenem polynomu (Obrázek 17 a 18).

```
30    #ověření, zda je interval zadán vzestupně
31    if(a > b) {
32      t <- a
33      a <- b
34      b <- t
35    }
```

Obrázek 13: Kontrola vzestupnosti krajních bodů v intervalu

```
> puleni(1, 0, i_max = 4)
[1] "0: s=0,5"
[1] "1: s=0,75"
[1] "2: s=0,625"
[1] "3: s=0,6875"
[1] "4: s=0,65625"
[1] "Kořen s danou přesností nenalezen, počet kroků=4"
>
```

Obrázek 14: Ukázka kontroly a úprava intervalu pro polynom $f(x) = x^3 + x - 1$

```
42    #ověření, že v daném intervalu se nachází kořen
43    if(f_a * f_b > 0) {
44      print(paste(sep="", "Máte špatně zadaný interval: ",
45                  "f(", n2s(a), ")=", n2s(f_a), ", f(", n2s(b), ")=",
46                     n2s(f_b)))
47      return(NA)
48    }
```

Obrázek 15: Kontrola, zdali je kořen polynomu v daném intervalu

```
> puleni(1,2)
[1] "Máte špatně zadaný interval: f(1)=1, f(2)=9"
[1] NA
```

Obrázek 16: Kontrola, zda se v intervalu $I = \langle 1,2 \rangle$ *polynomu* $f(x) = x^3 + x - 1$ *nachází kořen*

```
49    #ověření, zdali je jeden z krajích bodů intervalu kořenem polynomu
50    if(abs(f_a) <= e) {
51      print(paste(sep = "", "Kořenem je ", n2s(a)))
52      return(a)
53    }
54    if(abs(f_b) <= e) {
55      print(paste(sep = "", "Kořenem je ", n2s(b)))
56      return(b)
57    }
```

Obrázek 17: Kontrola, zda je krajní bod intervalu kořenem polynomu

```
> puleni(0, 1, polynom = x ^ 3 + 2 * x ^ 2 - x - 2)
[1] "Kořenem je 1"
[1] 1
>
```

Obrázek 18: Polynom $f(x) = x^3 + 2x^2 - x - 2$ *má v krajním bodě* $x = 1$ *intervalu* $I = \langle 0,1 \rangle$ *kořen*

Poté už program provede algoritmus půlení (Obrázek 19 a 20). Ten je realizován cyklem *while* a provádí půlení, dokud není dosaženo přesnosti ε nebo určeného počtu kroků, jak je zmíněno výše. Program je také ukončen, když je kořen nalezen přesně, tj. jeho funkční hodnota je rovná nule. To se stává pouze ve výjimečných případech, programy totiž mají problém vypočítat absolutní nulu. Jedním z příkladu, kdy se tato podmínka využívá, je při hledání kořenů polynomů nalezených hned po jednom aproximačním kroku (např. u polynomu $f(x) = 2x$ v intervalu $I = \langle -1,1 \rangle$). V těchto případech není dosaženo zadané přesnosti ε ani počtu kroků (pokud není nastaveno *i_max =1*).

```
63    #cyklus, který provádí půlení, dokud není dosaženo přesnosti e
64    while(abs(a - b) > e) {
65      f_a <- pol(a)
66      f_b <- pol(b)
67      s <- (a + b) / 2
68      f_s <- pol(s)
```

Obrázek 19: Algoritmus půlení 1. část

```
89      #ověření, s jakým intervalem budeme nadále počítat
90      if(f_a * f_s < 0) {
91        b <- s
92      }
93      else {
94        a <- s
95      }
```

Obrázek 20: Algoritmus půlení 2. část

Hodnoty jsou poté vypsány, jak přímo v konzoli, tak i v samostatném souboru pomocí funkce *write* (Obrázek 21 a 22). Vypisování do souboru poskytuje lepší přehled a umožňuje pracovat s výsledky i po opuštění konzole.

```
59    #vytvoření souboru, kde se uloží vypočítáné hodnoty, a pojmenování hodnot
60    soubor <- file("Puleni_tabulka.csv", "w")
61    write("krok;a;b;s;f(a);f(b);f(s)", soubor)
```

Obrázek 21: Funkce write - 1.část

```
70      #ukládání hodnot do souboru
71      write(paste(sep = "", i, ";", n2s(a), ";", n2s(b), ";", n2s(s),
72                  ";", n2s(f_a), ";", n2s(f_b), ";", n2s(f_s)), soubor)
73      print(paste(sep = "", i, ": ", "s=", n2s(s)))
```

Obrázek 22: Funkce write - 2.část

Jelikož je soubor ukládán s koncovkou .csv, který je kompatibilní například i s aplikací Excel od firmy Microsoft, bylo nutné přidat do programu ještě funkci *n2s (number to string)* (Obrázek 23), protože jazyk R odděluje desetinná čísla pomocí tečky a ne čárky, jak je zvykem v České republice. Čísla se tak mnohdy převáděla automaticky na data (1.5 → 1. května).

```
18  #převedení tečky u desetinných čísel na čárku
19  n2s <- function(n) {
20    format(n, decimal.mark = ",")
21  }
```

Obrázek 23: Funkce n2s

3.1.2. Metoda prosté iterace

Stejně jako pro metodu půlení intervalu, tak také v programu pro metodu prosté iterace je uveden vzorový polynom (Obrázek 24), kde je však již vepsána iterační funkce k danému polynomu $f(x) = x^3 + x - 1$.

```
3   #vzorový polynom, počáteční hodnota x_0=0,5
4   polynom_vzor <- function(x) {
5     return((1 - x) ^ (1 / 3))
6   }
```

Obrázek 24: Kód vzorové iterační funkce

Jelikož jsou podmínky pro ověřování správnosti intervalu pro tuto metodu docela složité a vyžadovaly by mnoho vstupů od uživatele, nejsou v programu ověřovány. Důležité pro tuto metodu je tedy určení maximálního počtu kroků (Obrázek 11), po kterém se program zastaví a uživatel může z dosažených výsledků rozhodnout, zdali posloupnost aproximací diverguje (Obrázek 25) či konverguje (Obrázek 26). V programu je taktéž možné počítat bez určeného počtu kroků, při nastavení *i_max = 0,* v takovémto případě by si měl být uživatel jistý, že funkce je konvergentní, jinak se program zacyklí.

```
> iterace(1,i_max=6)
[1] "x_0: 1"
[1] "x_1: 0"
[1] "x_2: 1"
[1] "x_3: 0"
[1] "x_4: 1"
[1] "x_5: 0"
[1] "x_6: 1"
[1] "Kořen s danou přesností nenalezen, počet kroků=6"
```

Obrázek 25: Ukázka divergence posloupnosti aproximací funkce $g(x) = \sqrt{1-x}$ při zadání $x_0 = 0$

```
> iterace(0.5, i_max = 6)
[1] "x_0: 0,5"
[1] "x_1: 0,7937005"
[1] "x_2: 0,5908801"
[1] "x_3: 0,7423639"
[1] "x_4: 0,6363102"
[1] "x_5: 0,7138008"
[1] "x_6: 0,6590061"
[1] "Kořen s danou přesností nenalezen, počet kroků=6"
>
```

Obrázek 26: Ukázka konvergence posloupnosti aproximací funkce $g(x) = \sqrt{1-x}$ při zadání $x_0 = 0{,}5$

Aproximace kořene je prováděna opět pomocí cyklu *while* ve funkci *iterace*, ve které na rozdíl od *puleni* zadáváme přímo počáteční aproximaci x_0. Jelikož program nepodporuje odmocniny ze záporných hodnot, je metoda prosté iterace určena pouze pro reálné kořeny (Obrázek 27).

```
> iterace(2, i_max = 6)
[1] "x_0: 2"
[1] "x_1: NaN"
Error in while (abs(x_k - x) > e) { :
  missing value where TRUE/FALSE needed
>
```

Obrázek 27: Ukázka, jak se metoda chová při zadání $x_0 = 2$, kdy jako další aproximace vychází komplexní číslo

Výsledky jsou stejně jako u předchozí metody vypsány do konzole i uloženy do souboru pomocí funkce *write*.

3.1.3. Newtonova metoda

Pro Newtonovu metodu jsou vytvořeny dva programy, pro reálné a komplexní kořeny. Program „Newtonova metoda pro reálné kořeny“ má podobné prvky jako předchozí programy. Jak i ostatní, i tento umožňuje počítat se vzorovým polynomem (Obrázek 10) a rovněž ověřuje některé podmínky: zdali je interval zadán vzestupně (Obrázek 13 a 14), zdali je interval zadán správně (Obrázek 15 a 16) a jestli nejsou krajní hodnoty kořeny polynomu (Obrázek 17 a 18).

Princip metody je popsán ve funkci *prubeh*. Potřebná derivace se počítá automaticky pomocí funkce *D* (vyjma případu, kdy se jedná o vzorový polynom, zde je již derivace zadána) (Obrázek 28). Zadaný polynom však musí obsahovat proměnnou *x*, protože funkce *D* derivuje právě podle této proměnné.

```
36    #výpočet derivace polynomu a převedení výrazu na funkci
37    else {
38
39      pol <- function (x) {}
40      body(pol) <- exp
41
42      derBody <- D(exp, "x")
43
44      der <- function (x) {}
45      body(der) <- derBody
46
47    }
```

Obrázek 28: Derivace polynomu

Tělo programu opět tvoří cyklus *while*, a abychom nemuseli rozlišovat, zda se jedná o funkci rostoucí nebo klesající, převede program funkci automaticky na funkci rostoucí (Obrázek 29) a poté zahájí cyklus.

```
61    #převedení na rostoucí funkci
62    if(f_a > f_b) {
63      v <- a
64      a <- b
65      b <- v
66    }
```

Obrázek 29: Převedení na rostoucí funkci

Pro ukončení programu je možno využít požadované přesnosti ε (je na uživateli, zda se rozhodne využít předdefinovanou přesnost $\varepsilon = 0{,}001$ nebo svou vlastní) či metodu ohraničit na určitý počet kroků (automaticky je nastaveno *i_max = 1000*).

Hodnoty se jako i u předchozích programů vypisují přímo do konzole a také do souboru *Newton_tabulka.csv*.

Program „Newtonova metoda pro komplexní kořeny“ je téměř totožný jako pro kořeny reálné, jsou zde však vynechány ověřovací podmínky, protože v komplexní rovině budeme těžko uvažovat interval, na němž bychom podmínky ověřovali. Do programu tedy zadáváme přímo počáteční aproximaci x_0.

3.2. *Porovnání programů a příklady*

Cílem této podkapitoly je porovnat vlastnosti jednotlivých programů (a tím pádem i jimi určených numerických metod). Pozor bude věnován rychlosti metod (kolik je potřeba na určení kořene kroků), zdali se metody ve výsledcích shodují, popřípadě jak se liší od „skutečného kořene“ (lze zkoumat, jedná-li se o racionální kořeny) a jak se s metodou pracuje (je těžké najít interval či počáteční aproximaci či nikoli).

Příklad 18: Určete kořen polynomu $f(x) = x^3 + x - 1$ s přesností $\varepsilon = 0{,}001$.
Diskuze: Pro programy „Metoda půlení intervalu“ a „Newtonova metoda“ byl kořen počítán v intervalu $I = \langle 0, 1\rangle$, pro program „Metoda prosté iterace“ bylo nutné nejdříve zvolit vhodnou iterační funkci a k ní počáteční aproximaci. Jako iterační funkce byla zvolena $g(x) = \sqrt[3]{1-x}$ s počáteční aproximací $x_0 = 0{,}5$.

Metoda prosté iterace vypočítala příklad v devatenácti krocích, $x_{19} = 0{,}682626670619523$, metoda půlení intervalu došla k výsledku v devíti krocích $x_9 = 0{,}6826171875$ a Newtonově metodě stačily na výpočet čtyři kroky $x_4 = 0{,}682327803946513$. Přehled jednotlivých intervalů a kořenů je ukázán v následující tabulce.

Polynom $f(x) = x^3 + x - 1$	Interval nebo poč. aproximace	Počet kroků	Kořen
Metoda půlení intervalu	$I = \langle 0, 1 \rangle$	9	0,6826171875
Metoda prosté iterace	$x_0 = 0{,}5$	19	0,682626670619523
	$x_0 = 0{,}6$	16	
Newtonova metoda	$I = \langle 0, 1 \rangle$	4	0,682327803946513
	$x_0 = i$	5	$-0{,}341164 + 1{,}161541\mathrm{i}$
	$x_0 = -i$	5	$-0{,}341164 - 1{,}161541\mathrm{i}$

Tabulka 8: Kořeny polynomu $f(x) = x^3 + x - 1$

Polynom $f(x) = x^3 + x - 1$ nemá racionální kořeny, proto nelze uvažovat odchylku od „skutečného“ řešení. Je však vidět, že metody se s výsledkem shodují. Nejpomalejší a také nejpracnější byla metoda prosté iterace. I přes to, že upravíme počáteční aproximaci na $x_0 = 0{,}6$ metoda stále potřebuje na výpočet kořene šestnáct kroků, což ji i v tomhle případě činí nejpomalejší.

K výpočtu dalších kořenů můžeme využít pouze program „Newtonova metoda pro komplexní kořeny“, protože se jedná o komplexní kořeny. S počáteční aproximací $x_0 = i$ dospějeme ke kořenu v pátém kroku $x_5 = -0{,}341164 + 1{,}161541\mathrm{i}$, posledním kořenem tedy bude k němu komplexně sdružené číslo $\overline{x_5} = -0{,}341164 - 1{,}161541\mathrm{i}$, ke kterému v metodě dojdeme opět po pěti krocích při zadání $x_0 = -i$.

Příklad 19: Určete kořen polynomu $f(x) = x^5 - x^3 + x - 2$ s přesností $\varepsilon = 0{,}001$.
Diskuze: Pro metodu půlení intervalu a Newtonovu metodu budeme počítat v intervalu $I = \langle 1{,}2 \rangle$, pro metodu prosté iterace je zvolena iterační funkce $g(x) = \sqrt[5]{x^3 - x + 2}$ a počáteční aproximace $x_0 = 0{,}5$.

V tomto příkladě metoda prosté iterace fungovala rychleji a v počtu kroků si může podat ruce s Newtonovu metodou. Půlení intervalu je zde zdlouhavější. Je zajímavé si povšimnout, že při zvětšení intervalu $I = \langle -10, 10 \rangle$ se počet kroků Newtonovy metody zvětší na 13 a půlení intervalu na 14, při ještě větším rozpětí $I = \langle -20, 20 \rangle$ metoda půlení intervalu předběhne Newtonovu metodu o jeden krok, půlení má 15 kroků, Newtonova metoda 16. Naopak při zmenšení intervalu $I = \langle 1; 1{,}3 \rangle$ dospěje Newtonova metoda k výsledku za 3 kroky a půlení intervalu za 8. Iterační metoda při zmenšení intervalu dospěje k výsledku v 5 krocích. I v tomto případě je však docela obtížné najít iterační

funkci a zbylé kořeny je možno opět dopočítat pouze Newtonovou metodou, protože se jedná o kořeny komplexní.

Polynom $f(x) = x^5 - x^3 + x - 2$	Interval nebo poč. aproximace	Počet kroků	Kořen
Metoda půlení intervalu	$I = \langle 1, 2\rangle$	9	1,206055
	$I = \langle 1; 1{,}3\rangle$	8	
	$I = \langle -10, 10\rangle$	14	
	$I = \langle -20, 20\rangle$	15	
Metoda prosté iterace	$x_0 = 0{,}5$	6	1,20526034969988
	$x_0 = 1$	5	
Newtonova metoda	$I = \langle 1, 2\rangle$	6	1,205569
	$I = \langle 1; 1{,}3\rangle$	3	
	$I = \langle -10, 10\rangle$	13	
	$I = \langle -20, 20\rangle$	16	
	$x_0 = i$	7	$0{,}5 + 0{,}8660254\mathrm{i}$
	$x_0 = -i$	7	$0{,}5 - 0{,}8660254\mathrm{i}$
	$x_0 = -1 + i$	5	$-1{,}102785 + 0{,}665457i$
	$x_0 = -1 - i$	5	$-1{,}102785 - 0{,}665457i$

Tabulka 9: Kořeny polynomu $f(x) = x^5 - x^3 + x - 2$

Příklad 20: Určete všechny kořeny polynomu $f(x) = 2x^4 - 11x^2 + 5$.

Diskuze: Tento polynom má všechny kořeny reálné a racionální, proto se budeme snažit určit jejich přesnou hodnotu. Z grafu (Obrázek 30) i vzorce je vidět, že funkce je sudá, bude tedy stačit najít pouze dva kořeny a zbylé dva se budou lišit pouze o znaménko.

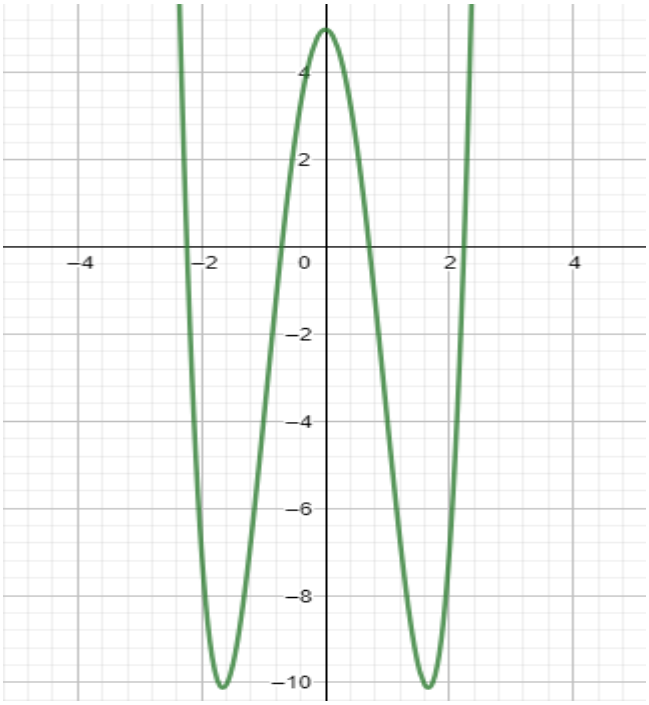
Nejdříve budeme počítat v intervalu $I = \langle 0{,}5;\ 1\rangle$, pro Newtonovu metodu a metodu půlení intervalu. Pro Newtonovu metodu totiž nemůžeme stanovit interval $I = \langle 0, 1\rangle$, protože pak bychom ve výpočtu dělili nulou[15], což nelze. Pro metodu prosté iterace budeme počítat s iterační funkci $g(x) = \sqrt{\frac{2x^4+5}{11}}$ a počáteční aproximaci

[15] Z grafu funkce je zřejmé, že daná funkce bude na intervalu klesat, proto jako počáteční aproximaci volíme číslo 0, $x_0 = 0$. Dosadíme-li do předpisu Newtonovy metody pro tento polynom, jistíme, že rovnice nemá řešení.

$$x_{k+1} = x_k - \frac{2x_k^4 - 11x_k^2 + 5}{8x_k^3 - 22x_k}$$

$$x_1 = 0 - \frac{2 \cdot 0 - 11 \cdot 0 + 5}{8 \cdot 0 - 22 \cdot 0}$$

$x_0 = 0{,}5$. V tomto případě chceme dojít k přesnému výsledku, proto jsme nastavili $\varepsilon = 0{,}000000001$. Nemůžeme však nastavit $\varepsilon = 0$, protože programy mají problém nalézt absolutní nulu a mnohdy se mohou i zacyklit.



Obrázek 30: Graf funkce $f(x) = 2x^4 - 11x^2 + 5$

Výpočty pro druhý kořen jsou uvedeny v tabulce, pro metodu prosté iterace však musela být zvolena jiná iterační funkce $g(x) = \sqrt[4]{\frac{11x^2-5}{2}}$.

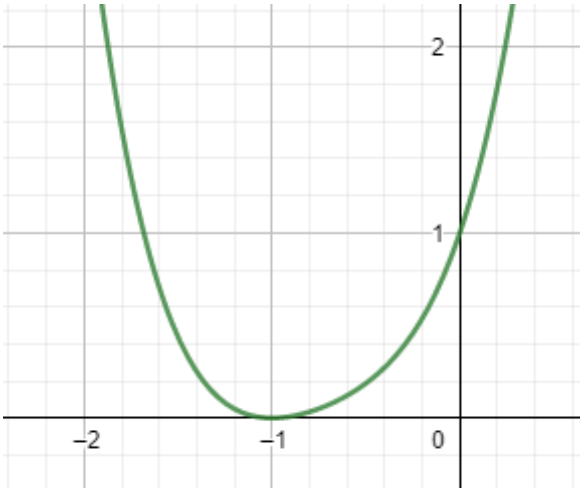
Polynom $f(x) = x^5 - x^3 + x - 2$	Interval nebo poč. aproximace	Počet kroků	Výsledek
Metoda půlení intervalu	$I = \langle 0{,}5;\ 1\rangle$	22	0,7071068
	$I = \langle 1;\ 3\rangle$	24	2,236068
Metoda prosté iterace	$x_0 = 0{,}5$	10	0,7071068
	$x_0 = 1$	29	2,236068
Newtonova metoda	$I = \langle 0{,}5;\ 1\rangle$	4	0,7071068
	$I = \langle 1;\ 3\rangle$	6	2,236068

Tabulka 10: Kořeny polynomu $f(x) = x^5 - x^3 + x - 2$

Všechny kořeny daného polynomu budou tedy $x_1 = 0{,}7071068$, $x_2 = -0{,}7071068$, $x_3 = 2{,}236068$, $x_4 = -2{,}236068$.

Příklad 21: Určete kořeny polynomu $f(x) = x^4 + 3x^3 + 4x^2 + 3x + 1$.

Diskuze: Z Obrázku 32 je vidět, že nikdy graf funkce neprotne osu *x* a tím pádem není splněna podmínka $f(a) \cdot f(b) < 0;\ \forall a, b \in \mathbb{R}$, která je pro programy numerických metod klíčová. Nemůže zde tedy použít programy „Metodu půlení intervalu“ a ani „Newtonova metoda pro reálné kořeny“, protože nesplnění podmínky programy zastaví.



Obrázek 31: Graf funkce $f(x) = x^4 + 3x^3 + 4x^2 + 3x + 1$

Je tak učiněno z důvodu, že uživatel může mnohdy špatně určit interval a program by se pokaždé zacyklil. Můžeme však využít program „Newtonova metoda pro komplexní kořeny“, který tuto podmínku neobsahuje a rovnou počítá se zadanou počáteční aproximací.

Polynom $f(x) = x^4 + 3x^3 + 4x^2 + 3x + 1$	Poč. aproximace	Počet kroků	Výsledek
Newtonova metoda	$x_0 = 0$	15	$-1{,}000081$
	$x_0 = i$	7	$-0{,}5 + 0{,}8660254i$
	$x_0 = -i$	7	$-0{,}5 - 0{,}8660254i$

Tabulka 11: Kořeny polynomu $f(x) = x^4 + 3x^3 + 4x^2 + 3x + 1$

Na příklad by šlo však nahlížet i jiným způsobem. Víme, že tento polynom obsahuje buď komplexní nebo násobné kořeny. S násobností kořenů by si poradily metody popsané v Kapitole 1, např. **Hornerovo schéma** nebo by bylo možné použít Větu 17 a vydělit polynom $NSD(f, f')$. V našem případě $NSD(f, f') = (x + 1)$, polynom proto můžeme rozložit na součin $f(x) = (x^3 + 2x^2 + 2x + 1)(x + 1)$ a graf $f_2(x) = x^3 + 2x^2 + 2x + 1$ už protíná osu *x*. Pro tento polynom již můžeme využít numerických metod. Počítání však můžeme ještě zjednodušit, jelikož z předchozího řešení je zřejmé, že polynom obsahuje dvojnásobný kořen $x = -1$, polynom $f_2(x)$ tak lze vydělit ještě jednou výrazem $(x + 1)$ a rozložit tak na $f(x) = (x^2 + x + 1)(x + 1)^2$.

Z toho rozkladu je zřejmé, že $f_3(x) = x^2 + x + 1$ má komplexní kořeny, které poté dokážeme určit Newtonovou metodou. Výsledkem budou komplexně sdružené kořeny $x_{3,4} = -0{,}5 \pm 0{,}8660254i$.

Příklad 22: Najděte kořeny polynomu $f(x) = 147x^4 + 84x^3 - 40x^2 - 28x - 3$.

Diskuze: Zkusíme hledat kořeny v intervalech $I = \langle -1, 0\rangle$ a $I = \langle 0, 1\rangle$ pro Newtonovu metodu a metodu půlení intervalu. Pro metodu prosté iterace budeme počítat s iterační funkcí $g(x) = \sqrt[4]{\frac{-84x^3+40x^2+28x+3}{147}}$ a počáteční aproximací $x_0 = 0$.

Polynom $f(x) = 147x^4 + 84x^3 - 40x^2 - 28x - 3$	Interval nebo poč. aproximace	Počet kroků	Kořen
Metoda půlení intervalu	$I = \langle -1, 0\rangle$	22	$-0{,}5773503$
	$I = \langle -0{,}8;\ 0\rangle$	22	$-0{,}1428571$
	$I = \langle 0, 1\rangle$	22	$0{,}5773503$
Metoda prosté iterace	$x_0 = 0$	7	$0{,}5773503$
Newtonova metoda	$I = \langle -1, 0\rangle$	7	$-0{,}5773503$
	$I = \langle -0{,}8;\ 0\rangle$	5	$-0{,}5773503$
	$I = \langle 0, 1\rangle$	6	$0{,}5773503$

Tabulka 12: Kořeny polynomu $f(x) = 147x^4 + 84x^3 - 40x^2 - 28x - 3$

Z výpočtu je zřejmé, že v intervalu $I = \langle -1, 0\rangle$ leží více kořenů, protože zúžením intervalu nalezneme pro metodu půlení intervalu vždy jiný kořen. Naopak zužováním či zvětšováním intervalu u Newtonovy metody dostaneme vždy stejný kořen. Správné intervaly, v kterých bychom měli kořeny hledat jsou tedy $I_1 = \langle -1;\ -0{,}5\rangle$, $I_2 = \langle -0{,}5;\ -0{,}35\rangle$, $I_3 = \langle -0{,}35;\ 0\rangle$. Hodnoty intervalu musíme volit právě tímto způsobem, protože Newtonova metoda při zadání některých intervalů (např. $I = \langle -0{,}5;\ -0{,}3\rangle$) najde kořen úplně mimo daný interval (např $x = 0{,}5773503$). Níže je upravená tabulka pro metodu půlení intervalu a Newtonovu metodu.

Polynom $f(x) = 147x^4 + 84x^3 - 40x^2 - 28x - 3$	Interval	Počet kroků	Kořen
Metoda půlení intervalu	$I_1 = \langle -1;\ -0{,}5 \rangle$	21	$-0{,}5773503$
	$I_2 = \langle -0{,}5;\ -0{,}35 \rangle$	18	$-0{,}4285714$
	$I_3 = \langle -0{,}35;\ 0 \rangle$	18	$-0{,}1428571$
	$I_4 = \langle 0, 1 \rangle$	22	$0{,}5773503$
Newtonova metoda	$I_1 = \langle -1;\ -0{,}5 \rangle$	7	$-0{,}5773503$
	$I_2 = \langle -0{,}5;\ -0{,}35 \rangle$	4	$-0{,}4285714$
	$I_3 = \langle -0{,}35;\ 0 \rangle$	4	$-0{,}1428571$
	$I_4 = \langle 0, 1 \rangle$	6	$0{,}5773503$

Tabulka 13: Kořeny polynomu $f(x) = 147x^4 + 84x^3 - 40x^2 - 28x - 3$

3.3. Shrnutí a doporučení

Z výše uvedených příkladů můžeme vyvodit některé vlastnosti jednotlivých numerických metod. Nejrychlejší metodou (metodou s nejmenším počtem kroků) byla skoro ve všech případech Newtonova metoda. Její účinnost však docela rapidně klesala se zvětšujícím se intervalem. Také je nutno dodat, že v Příkladě 20 a 22 byl problém najít interval, ve kterém bychom mohli kořen hledat, protože pro některé intervaly nebyla splněna Fourierova podmínka a v intervalu $I = \langle -0{,}5; 0{,}5 \rangle$ počítala metoda „špatné“ kořeny. Např. pro $I = \langle 0{,}1; 1 \rangle$ našla kořen $x = 2{,}236068$, který do toho intervalu vůbec nepatří a podobně. U Newtonovy metody je proto důležité dbát na správný výběr intervalu. Co však staví Newtonovu metodu, v tomto případě lépe řečeno program „Newtonova metoda pro komplexní kořeny“, před všechny ostatní je výpočet komplexních kořenů.

Velkou výhodou metody půlení intervalu bylo její všestranné využití. Bylo možno ji využít vždy, nehledě na vybraný intervalu. Jediným „zádrhelem“ jsou násobné kořeny. Metoda si poradí, i když se v intervalu nachází více kořenů, ale v takovémto případě najde pouze jeden z nich. Každopádně bychom se měli vyhnout počítání více kořenů v jednom intervalu a měli bychom být schopni vždy najít interval, kde je kořen právě jeden. Největší nevýhodou metody je její zdlouhavost. Někdy se oproti ostatním metodám lišila až o 15 kroků (Příklad 22), což by při ručním počítání určitě zabralo nějaký čas. Největší prodleva byla pozorovatelná na velmi malých intervalech, naopak metoda dobře fungovala ve velkých rozpětích intervalu. V Příkladě 19 byla metoda půlení intervalu dokonce rychlejší než Newtonova metoda.

Nejvíce nevyváženou metodou ze všech byla metoda prosté iterace. V některých příkladech se v rychlosti rovnala Newtonově metodě (Příklad 19), jindy zaostávala i za metodou půlení intervalu (Příklad 18). Celkově by se však dala zařadit mezi rychlejší metody výpočtu. Největším problémem u této metody bylo určování iteračních funkcí a počátečních aproximací. Někdy jsme funkci našli hned na první pokus, jindy bylo hledání složitější. Značný problém nalezení iteračních funkcí byl v Příkladu 22. To, že Newtonova byla rychlejší a snadnější pro práci se však dalo předpokládat, jelikož Newtonova metoda je speciálním případem iterační metody.

Kdybychom chtěli shrnout nejefektivnější řešení polynomických rovnic, určitě bychom začali s nakreslením grafu, který uživateli ledacos napoví a ukáže mu, jaký interval pro výpočet kořenů zvolit. Doporučením, jak si tuto práci usnadnit je grafický kalkulátor Geogebra (Hohenwarter, 2002), který byl v této bakalářské práci použit k ilustrací všech grafů.

Jak je zmíněno výše, z grafu se dá ledacos odhadnout. Vidíme-li, že graf neprotíná osu *x* (Příklad 21) odhadujeme, že kořeny polynomu jsou buďto násobné nebo komplexní. Výše je zmíněno, jakým způsobem se dá takovýto případ vyřešit. Pro snadnější výpočet však můžeme využít volně dostupných programů na webových stránkách. Derivaci je možné vypočítat v programu „Derivace polynomu“ (je přiložen v Přílohách), jehož princip je ekvivalentní s derivací použitou v programu „Newtonova metoda pro reálné kořeny“ a „Newtonova metoda pro komplexní kořeny“ a je rovněž naimplementován v jazyce R. Největší společný dělitel polynomu a jeho derivace pak můžeme být jednoduše spočítán na webové stránce zabývající se online kalkulačkami (Anton, Polynomial Greatest Common Divisor, 2018). Stránka je sice v angličtině, ale velmi dobře zpracována, takže by neměl být problém se na ni správně orientovat. Jediným problémem by mohlo být zadávání polynomu, který je zde vyobrazen jako posloupnost, např. polynom $f(x) = x^3 - 2x^2 + 1$ bychom zadali jako $(1\ -2\ 0\ 1)$.

V dalším kroku dělíme polynom největším společným dělitelem polynomu a jeho derivace. K tomu můžeme opět využít volně dostupných programů. Na internetových stránkách jich existuje celá řada, v této bakalářské práci jsou uvedeny dva. Jeden se nachází již na výše zmíněné stránce (Anton, Polynomial division, 2018). Uživatel si může vybrat, chce-li počítat dělení přesně nebo mu stačí pouze zaokrouhlený výsledek. Stránka taktéž zobrazuje všechny kroky dělení, které jsou doprovázeny slovním popisem. Co však tuto online kalkulačku staví nad ostatní je možnost pracovat i s komplexními čísly.

Druhu stránkou je (Wolfram Research, 2018), i tato stránka nabízí přehled jednotlivých kroků dělení, a navíc se polynomy zadávají klasicky, ne jako posloupnosti. Dělení komplexními čísly však není možné provést.

Zbavíme-li se násobných kořenů, zbývá využít dostupných numerických metod. Nejjednodušší cestou se jeví použití metody půlení intervalu na zúžení intervalu a poté použití Newtonovy metody. Tím by mělo být dosáhnuto výsledku s nejmenším možným počtem kroků.

Pro komplexní kořeny zbývá použít pouze Newtonovu metodu (program „Newtonova metoda pro komplexní kořeny“). Kdybychom však chtěli využít znalostí z Kapitoly 1, je možno najít komplexní kořeny i pomocí Hornerova schématu.

Závěr

Tato bakalářská práce prozkoumala oblast polynomů a k řešení polynomických rovnic popsala tři numerické metody, a to metodu půlení intervalu, metodu prosté iterace a Newtonovu metodu. Tyto metody byly naprogramovány ve volně dostupném jazyce R a fungování jednotlivých částí programů bylo vysvětleno v poslední kapitole. Jakýkoli uživatel by tedy po přečtení této bakalářské práce neměl mít problém s používáním těchto programů.

Poslední kapitola také uvádí 5 příkladů (Příklad 18-23), které pomáhají k následnému porovnávání vybraných numerických metod a tím poukazuje na jejich světlé i stinné stránky. V této kapitole jsou rovněž popsány některé volně dostupné webové aplikace, které usnadňují výpočty potřebné k určení správného výsledku. Například je zde uvedena webová stránka Geogebra, která pomáhá určit správný interval, ve kterém leží kořeny dané polynomické rovnice, nebo různé online kalkulačky, které pomáhají řešit problém s násobnými kořeny polynomů, na něž jsou některé numerické metody krátké.

Tato bakalářské práce tak může pomoci studentům k lepšímu pochopení látky polynomů a s pomocí programů by si měli poradit s jakoukoli polynomickou rovnicí.

Myslím si, že i když jsou polynomy a numerické metody předmětem již mnoha bakalářských či diplomových prací, každá z nich vrhá světlo pouze na kousek problematiky a tím, že každá práce uchopí tuto látku jinak, je stále ještě dost prostoru pro budoucí práce přijít s něčím novým a unikátním. Studentům, kteří by tedy chtěli na tuto bakalářskou práci navázat, bych doporučila pokračovat ve vytvoření programů, které by mohly více objasnit a usnadnit další výpočty. Zejména bych uvítala program na výpočet Hornerova schématu, pomocí kterého je možné řešit některé polynomické rovnice a zejména pro celočíselné kořeny jsou mnohdy lepší volbou než výše zmíněné numerické metody. Velmi užitečné by bylo i naprogramování např. dělení polynomů či Eukleidův algoritmus pro hledání největšího společného dělitele dvou polynomů, aby se student nemusel spoléhat na webové stránky.

Bibliografie

Anton. (22. prosinec 2018). *Polynomial division*. Načteno z Online calculators: https://planetcalc.com/7718/

Anton. (22. prosinec 2018). *Polynomial greatest common divisor*. Načteno z Online calculators: https://planetcalc.com/7760/

Bárta, T., Pražák, D. a kol. (2010) *Jednoznačnost a diferencovatelnost řešení.* Načteno z Matematicko-fyzikální fakulta Univerzity Karlova: http://www.karlin.mff.cuni.cz/~barta/pcODR/Kapitola-JednDiferenc/JednoznDifferenc.pdf

Becker, R. A. (1994). A Brief History of S. *AT&T Bell Laboratories*.

Budínová, I. (2013). *Polynomy* . Brno: Masarykova univerzita, Pedagogická fakulta.

Čechová, P. (2013). *Základní věta algrebry a její důkazy* (Diplomová práce). Brno: Masarykova univerzita, Přírodovědecká fakulta.

Fajmon, B., & Růžičková, I. (2003). *IS-VUT-MAT103.* Brno: VUT v Brně, Fakulta elektrotechniky a komunikačních technologií.

Fišnarová, S. (22. září 2008). *Polynomy a racionální lomené funkce.* Načteno z Kiwi.mendelu.cz: http://user.mendelu.cz/fisnarov/zvm/prednasky/polynomy_text.pdf

Hašek, R. (5. července 2018). *Vietovy vzorce*. Načteno z Pedagogická fakulta Jihočeská univerzita v Českých Budějovicích: http://home.pf.jcu.cz/~hasek/Algebra4/ALG4%20046.jpg

Hohenwarter, M. (únor 2002). *GeoGebra: Ein Softwaresystem für dynamische Geometrie und Algebra der Ebene*. Německo: Paris Lodron University, Salzubrg, Austria. Načteno z https://www.geogebra.org/

Horák, P. (1977). *Polynomy.* Brno: Univerzita Jana Evangelisty Purkyně.

Horová, I. (1999). *Numerické metody.* Brno: Masarykova univerzita, Přírodovědecká fakulta.

Chapra, S. C., & Canale, R. P. (2010). *Mumerical Methods for Engineers.* New York: McGraw-Hill.

Mařík, R. (31. říjen 2012a). *Přibližné řešení rovnic.* Načteno z Kiwi.mendelu.cz: http://user.mendelu.cz/marik/mat-web/mat-webse23.html

Mařík, R. (31. říjen 2012b). *Věty o spojitých funkcích.* Načteno z Kiwi.mendelu.cz: http://user.mendelu.cz/marik/mat-web/mat-webse5.html#x10-70005

Páv, D. (2010). *Řešení kubických rovnic* (Diplomová práce). Liberec: Technická univerzita v Liberci, Fakulta přírodovědně-humanitní a pedagogická.

Rosický, J. (2000). *Algebra.* Brno: Masarykova univerzita, Přírodovědecká fakulta.

Slovák, J., Panák, M., Bulant, M., Návrat, A., & Veselý, M. (2013). *Matematika drsně a svižně .* Brno: Masarykova univerzita.

Spojitost funkce (21. červenec 2018). Načteno z Matematika.cz: https://matematika.cz/spojitost-funkce

Tišnovský, P. (28. březen 2006). *Fraktály v počítačové grafice XXIII*. Načteno z Root.cz: https://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-xxiii/#k04

Wolfram Research, I. (22. prosinec 2018). *Long Division of Polynomials*. Načteno z Essential Mathematica for Students of Science: Tutorial Approach to Mastery of Mathematica -- from Wolfram Library Archive: http://library.wolfram.com/webMathematica/Education/LongDivide.jsp

Seznam obrázků:

Seznam tabulek:

Přílohy

Program „Půlení intervalu“

```
#Puleni intervalu

#vzorový polynom, počítáme v intervalu I=<0,1>
polynom_vzor <- function(x) {
  return(x ^ 3 + x - 1)
}
#převedení tečky u desetinných čísel na čárku
n2s <- function(n) {
  format(n, decimal.mark = ",")
}

#funkce, kde (a,b) = interval, e = přesnost, polynom = námi
#zadaný polynom, i_max = maximální počet kroků, při zadání
#i_max = 0, nekonečně mnoho kroků
puleni <- function(a, b, e = 0.001, polynom = NULL,
i_max = 1000) {

  #převedení zadaného polynomu na výraz (expresssion)
  exp <- substitute(polynom)

  #když není zadán žádný polynom, program počítá se vzorovým
  #polynomem
  if(is.null(exp)) {
    pol <- polynom_vzor
  }

  #převedení výrazu na funkci
  else {
    pol <- function (x) {}
    body(pol) <- exp
  }

  #ověření, zda je interval zadán vzestupně
  if(a > b) {
    t <- a
    a <- b
    b <- t
  }

  #výpočet funkčních hodnot
  f_a <- pol(a)
  f_b <- pol(b)
  i <- 0

  #ověření, že v daném intervalu se nachází kořen
  if(f_a * f_b > 0) {
    print(paste(sep="", "Máte špatně zadaný interval: ",
                "f(", n2s(a), ")=", n2s(f_a), ", f(", n2s(b),
                ")=", n2s(f_b)))
    return(NA)
  }
```

```
  #ověření, zdali je jeden z krajích bodů intervalu kořenem
  #polynomu
  if(abs(f_a) <= e) {
    print(paste(sep = "", "Kořenem je ", n2s(a)))
    return(a)
  }
  if(abs(f_b) <= e) {
    print(paste(sep = "", "Kořenem je ", n2s(b)))
    return(b)
  }

  #vytvoření souboru, kde se uloží vypočítáné hodnoty,
  #a pojmenování hodnot
  soubor <- file("Puleni_tabulka.csv", "w")
  write("krok;a;b;s;f(a);f(b);f(s)", soubor)

  #cyklus, který provádí půlení, dokud není dosaženo přesnosti e
  while(abs(a - b) > e) {
    f_a <- pol(a)
    f_b <- pol(b)
    s <- (a + b) / 2
    f_s <- pol(s)

    #ukládání hodnot do souboru
    write(paste(sep = "", i, ";", n2s(a), ";", n2s(b), ";",
          n2s(s), ";", n2s(f_a), ";", n2s(f_b), ";", n2s(f_s)),
          soubor)
    print(paste(sep = "", i, ": ", "s=", n2s(s)))

    #ověření, zdali jsme dosáhli určeného počtu kroků
    i <- i + 1
    if(i_max > 0 && i > i_max){
      print(paste(sep = "", "Kořen s danou přesností nenalezen",
                  ", počet kroků=", i - 1))
      break
    }

    #ověření, zda jsem nalezli kořen přesně
    if(f_s == 0) {
      print(paste(sep = "", "Kořenem je ", n2s(s)))
      break
    }

    #ověření, s jakým intervalem budeme nadále počítat
    if(f_a * f_s < 0) {
      b <- s
    }
    else {
      a <- s
    }
  }
  close(soubor)
}
```

Program „Metoda prosté iterace“

```
#Metoda prosté iterace

#vzorový polynom, počáteční hodnota x_0=0,5
polynom_vzor <- function(x) {
  return((1 - x) ^ (1 / 3))
}

#převedení tečky u desetinných čísel na čárku
n2s <- function(n) {
  format(n, decimal.mark = ",")
}

#funkce, kde x = počáteční aproximace, e = přesnost,
#polynom = námi zadaný polynom, i_max = maximální počet kroků,
#při zadání i_max = 0, nekonečně mnoho kroků
iterace <- function(x, e = 0.001, polynom = NULL, i_max = 1000)
{

  #převedení zadaného polynomu na výraz (expresssion)
  exp <- substitute(polynom)

  #když není zadán žádný polynom, program počítá se vzorovým
  #polynomem
  if(is.null(exp)) {
    pol <- polynom_vzor
  }

  #převedení výrazu na funkci
  else {
    pol <- function (x) {}
    body(pol) <- exp
  }

  #vytvoření souboru, kde se uloží hodnoty, a pojmenování hodnot
  soubor <- file("Iterace_tabulka.csv", "w")
  write("krok;x", soubor)
  write(paste(sep = "", "0;", n2s(x)), soubor)
  print(paste(sep = "", "x_0: ", n2s(x)))

  x_k <- pol(x)

  write(paste(sep = "", "1;", n2s(x_k)), soubor)
  print(paste(sep = "", "x_1: ", n2s(x_k)))

  i <- 2

  #cyklus, který provádí aproximaci kořene, dokud není dosaženo
  #přesnosti e
  while (abs(x_k - x) > e) {
    x <- x_k
    x_k <- pol(x)

    #ukládání hodnot do souboru
    write(paste(sep = "", i, ";", n2s(x_k)), soubor)
```

```
    print(paste(sep = "", "x_",i,": ", n2s(x_k)))

    #ověření, zdali jsem dosáhli požadovaného počtu kroků
    i <- i + 1
    if (i_max > 0 && i > i_max) {
      print(paste(sep = "", "Kořen s danou přesností nenalezen",
                  ", počet kroků=", i - 1))
      break
    }
  }
  close(soubor)
}
```

Program „Newtonova metoda pro reálné kořeny“

```
#Newtonova metoda pro reálné kořeny

#vzorový polynom, počítáme v intervalu I=<0,1>
polynom_vzor <- function(x) {
  return(x ^ 3 + x - 1)
}

#derivace vzorového polynomu
derivace_vzor <- function(x) {
 return(3 * x ^ 2 + 1)
}

#princip Newtonovy metody
prubeh <- function(x, pol, der) {
  return(x - pol(x) / der(x))
}

#převedení tečky u desetinných čísel na čárku
n2s <- function(n) {
  format(n, decimal.mark = ",")
}

#funkce, kde <a,b> je interval, e = přesnost, polynom = zadaný
#polynom, i_max = maximální počet kroků, při zadání i_max = 0,
#nekonečně mnoho kroků
newton <- function(a, b, e = 0.001, polynom = NULL,
i_max = 1000) {

  #převedení zadaného polynomu na výraz (expresssion)
  exp <- substitute(polynom)

  #když není zadán žádný polynom, program počítá se vzorovým
  #polynomem a jeho derivací
  if(is.null(exp)) {
    pol <- polynom_vzor
    der <- derivace_vzor
  }

  #výpočet derivace polynomu a převedení výrazu na funkci
  else {
```

```
  pol <- function (x) {}
  body(pol) <- exp

  derBody <- D(exp, "x")

  der <- function (x) {}
  body(der) <- derBody

}

#ověření, zda je interval zadán vzestupně
if(a > b) {
  t <- a
  a <- b
  b <- t
}

#výpočet funkčních hodnot
f_a <- pol(a)
f_b <- pol(b)
i <- 2

#převedení na rostoucí funkci
if(f_a > f_b) {
  v <- a
  a <- b
  b <- v
}


#vytvoření souboru, kde se uloží vypočítáné hodnoty,
#a pojmenování hodnot
soubor <- file("Newton_tabulka.csv", "w")
write("krok;x", soubor)

#ověření, zdali jeden z krajích bodů intervalu není kořenem
#polynomu
if(abs(f_a) <= e) {
  print(paste(sep = "", "Kořenem je ", n2s(a)))
  return(a)
}
if(abs(f_b) <= e) {
  print(paste(sep = "", "Kořenem je ", n2s(b)))
  return(b)
}

#ověření, že v daném intervalu se nachází kořen
if(f_a * f_b > 0) {
  print(paste(sep="", "Máte špatně zadaný interval: ",
              "f(", n2s(a), ")=", n2s(f_a), ", f(", n2s(b),
              ")=", n2s(f_b)))
  return(NA)
}

#ukládání hodnot do souboru
write(paste(sep = "", 0, ";", n2s(b)), soubor)
```

```
  print(paste(sep = "", "x_0: ", n2s(b)))

  x <- prubeh(b, pol, der)
  write(paste(sep = "", 1, ";", n2s(x)), soubor)
  print(paste(sep = "", "x_1: ", n2s(x)))

  #cyklus, který provádí aproximaci kořene, dokud není dosaženo
  #přesnosti e
  while(abs(x - b) > e) {
      b <- x
      x <- prubeh(b, pol, der)

      #ukládání hodnot do souboru
      write(paste(sep = "", i, ";", n2s(x)), soubor)
      print(paste(sep = "", "x_", i, ": ", n2s(x)))

      #ověření, zdali jsem dosáhli požadovaného počtu kroků
      i <- i + 1
      if(i_max > 0 && i > i_max) {
        print(paste(sep = "", "Kořen s danou přesností
              nenalezen",", počet kroků=", i - 1))
        break
      }
  }
  close(soubor)
}
```

Program „Newtonova metoda pro komplexní kořeny“

```
#Newtonova metoda pro komplexní kořeny

#vzorový polynom, počítáme pro počáteční aproximaxi x_0 = 1+1i
polynom_vzor <- function(x) {
  return(x ^ 6 + x ^ 5 + 4 * x ^ 4 + 3 * x ^ 3 + 5 * x ^ 2 + 2 *
         x + 2)
}

#derivace vzorového polynomu
derivace_vzor <- function(x) {
  return(6 * x ^ 5 + 5 * x ^ 4 + 16 * x ^ 3 + 9 * x ^ 2 + 10 * x
         + 2)
}

#princip Newtonovy metody
prubeh <- function(x, pol, der) {
  return(x - pol(x) / der(x))
}

#převedení tečky u desetinných čísel na čárku
n2s <- function(n) {
  format(n, decimal.mark = ",")
}

#funkce, kde x je počáteční aproximace, e = přesnost,
#polynom = #zadaný polynom, i_max = maximální počet kroků, při
#zadání i_max = 0, nekonečně mnoho kroků
```

```
newton_complex <- function(a, e = 0.001, polynom = NULL,
i_max = 1000) {

  #převedení zadaného polynomu na výraz (expresssion)
  exp <- substitute(polynom)

  #když není zadán žádný polynom, program počítá se vzorovým
  #polynomem a jeho derivací
  if(is.null(exp)) {
    pol <- polynom_vzor
    der <- derivace_vzor
  }

  #výpočet derivace polynomu a převedení výrazu na funkci
  else {

    pol <- function (x) {}
    body(pol) <- exp

    derBody <- D(exp, "x")

    der <- function (x) {}
    body(der) <- derBody

  }

  f_a <- pol(a)
  i <- 2

  #vytvoření souboru, kde se uloží vypočítáné hodnoty,
  #a pojmenování hodnot
  soubor <- file("Newton_komplex_tabulka.csv", "w")
  write("krok;x", soubor)

  #ověření, zdali je počáteční aproximace kořenem polynomu
  if(abs(f_a) <= e) {
    print(paste(sep = "", "Kořenem je ", n2s(a)))
    return(a)
  }

  #ukládání hodnot do souboru
  write(paste(sep = "", 0, ";", n2s(a)), soubor)
  print(paste(sep = "", "x_0: ", n2s(a)))

  x <- prubeh(a, pol, der)
  write(paste(sep = "", 1, ";", n2s(x)), soubor)
  print(paste(sep = "", "x_1: ", n2s(x)))

  #cyklus, který provádí aproximaci kořene, dokud není dosaženo
  #přesnosti e
  while(abs(x - a) > e) {
    a <- x
    x <- prubeh(a, pol, der)

    #ukládání hodnot do souboru
    write(paste(sep = "", i, ";", n2s(x)), soubor)
```

```
    print(paste(sep = "", "x_", i, ": ", n2s(x)))

    #ověření, zdali jsem dosáhli požadovaného počtu kroků
    i <- i + 1
    if(i_max > 0 && i > i_max) {
      print(paste(sep = "", "Kořen s danou přesností nenalezen",
                  ", počet kroků=", i - 1))
      break
    }
  }
  close(soubor)
}
```

Program „Derivace polynomu“

```
#Derivace polynomu

derivace <- function (polynom) {

  exp <- substitute(polynom)
  derBody <- D(exp, "x")

  der <- function (x) {}
  body(der) <- derBody

  return(der)
}
```