Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Iterační metody řešení nelineárních rovnic a jejich využití

Jonáš Urban

Bakalářská práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jonáš Urban**
Osobní číslo:	**I14191**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Iterační metody řešení nelineárních rovnic a jejich využití**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Práce by měla nejprve teoreticky popsat jednotlivé iterační metody pro řešení nelineárních rovnic, popis je vhodné doplnit ilustračními příklady a grafy. Jednotlivé metody by měly být na několika příkladech srovnány, srovnání by mělo být formulováno i v obecné rovině – kdy lze jednotlivé metody využít, pro který problém je která metoda výhodnější a podobně. Jednotlivé metody by měly být implemetovány ve vhodném programovacím jazyce, případně v MATLABu, který je pto tuto problematiku též vhodný. Dále by měla práce obsahovat možné využití popsaných metod v praxi – ekonomie, manažerské rozhodování atd.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

FEISTAUER, Miloslav a Václav KUČERA. Základy numerické matematiky. Vyd. 1. Praha: Matfyzpress, 2014, 73 s. ISBN 978-80-7378-264-1. RALSTON, Anthony. Základy numerické matematiky: příručka pro university ČSR. 2. čes. vyd. Praha: Academia, 1978, 635 s. HOROVÁ, Ivana.: Numerické metody. Skriptum MU, Brno: MU, 1999. VITÁSEK, Eduard.: Numerické metody. Praha: SNTL. 1987.

Vedoucí bakalářské práce: **Mgr. Alena Pozdílková, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **12. května 2019**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 20. března 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 29. 7. 2019

Jonáš Urban

PODĚKOVÁNÍ

Rád bych tímto způsobem poděkoval především vedoucí své práce Mgr. Aleně Pozdílkové, Ph.D., která mi poskytla cenné rady a připomínky během psaní, a dále své rodině a přítelkyni za neocenitelnou podporu během psaní práce.

ANOTACE

Cílem práce je představit několik základních iteračních metod pro řešení nelineárních funkcí. V první kapitole vytyčíme některé základní pojmy nutné pro další práci. V druhé kapitole se budeme věnovat jednotlivým metodám, pro které bude vysvětlen princip jejich fungování. Konkrétně půjde o metody bisekce, Regula falsi, prosté iterace, sečen, Newtonovu, Steffensenovu, Müllerovu, Maehlyovu a Laguerrovu metodu. Ve třetí kapitole budou představeny různé typy příkladů, na kterých budou tyto metody testovány a vyhodnocovány. Zde prokážeme, že neexistuje univerzální metoda pro všechny typy příkladů. V poslední kapitole pak bude popsán software vytvořený v programovacím jazyce Java, který na zadanou funkci aplikuje metody popsané v této práci a představí uživateli proces řešení, tj. tabulku iterací, vyhodnocení jednotlivých ukončovacích kritérií a graf konvergence.

KLÍČOVÁ SLOVA

Nelineární rovnice, numerické metody, iterační metody, kořen, řešení

TITLE

Iterative methods of solving non-linear equations and their utilization

ANNOTATION

The aim of this work is to present few basic iterative methods for solving non-linear equations. In the first chapter some of the basic terms required for further reading will be explained. In the second chapter we will focus on the methods, for which basic principle of operation will be explained. Specifically, bisection, false position, fixed-point iteration, Newton's, secant, Steffensen's, Muller's, Maehly's and Laguerre's methods will be covered. In the third chapter distinct types of examples will be presented and numerical experiments will be carried out. In this chapter we will prove that no universal method for every type of equation exists. In the last chapter a program created in Java programming language will be described. This application applies every of the explained methods on the provided equation and presents a step-by-step solution including iteration tables, terminating conditions evaluation and convergence graph.

KEYWORDS

Non-linear equation, numerical methods, iterative methods, root, solution

OBSAH

Seznam obrázků 9

Seznam tabulek 10

Úvod 11

1 Kapitola 12

1.1 Definice problému 12

1.2 Počáteční odhad 12

1.3 Konvergence metody 13

1.4 Ukončení řešení 14

2 Kapitola 16

2.1 Základní metody 16

2.1.1 Metoda bisekce 16

2.1.2 Metoda Regula falsi 17

2.1.3 Metoda prosté iterace 19

2.1.4 Newtonova metoda 23

2.1.5 Metoda sečen 27

2.1.6 Steffensenova metoda 31

2.2 Metody pro řešení polynomů 34

2.2.1 Newtonova metoda 35

2.2.2 Newtonova-Maehlyova metoda 37

2.2.3 Müllerova metoda 38

2.2.4 Laguerrova metoda 39

3 Kapitola 41

3.1 Příklady a analýza metod 41

3.1.1 Kvadratické rovnice 42

3.1.2 Goniometrické funkce 44
3.1.3 Exponenciální funkce 47
3.1.4 Logaritmické funkce 48
3.1.5 Polynomy 50
4 Kapitola 54
Závěr 55
Použitá literatura 57
Přílohy 58

SEZNAM OBRÁZKŮ

Obrázek 1: Pevný bod iteračních funkcí g(x) 20
Obrázek 2: Ukázka prvních tří iterací Newtonovy metody 24
Obrázek 3: Ukázka výsledných funkcí po dělení polynomu nalezeným kořenem 36
Obrázek 4: Hlavní okno aplikace po spuštění 62
Obrázek 5: Vstupní sekce aplikace 62
Obrázek 6: Zadání vstupu metod a iteračních funkcí g(x) 63
Obrázek 7: Ukázka chyby vstupu u metody bisekce 63
Obrázek 8: Ukázka řešení funkce $fx = x3 - x2 - x$ Newtonovou metodou 64
Obrázek 9: Ukázka vykreslení funkce s detailem první iterace Newtonovy metody 65
Obrázek 10: Ilustrace vlivu hodnoty rozlišení při vykreslování grafu 66
Obrázek 11: Souhrn výsledků všech testovaných metod 67
Obrázek 12: Graf konvergence 67

SEZNAM TABULEK

Tabulka 1: Výsledky pokusů pro funkci $fkv1(x)$ 42
Tabulka 2: Výsledky pokusů pro funkci $fkv2(x)$ 43
Tabulka 3: Výsledky pokusů pro funkci $fkv3(x)$ 43
Tabulka 4: Výsledky pokusů pro funkci $fgon1(x)$ 45
Tabulka 5: Výsledky pokusů pro funkci $fgon2(x)$ 45
Tabulka 6: Výsledky pokusů pro funkci $fgon3(x)$ 45
Tabulka 7: Výsledky pokusů pro funkci $fexp1(x)$ 47
Tabulka 8: Výsledky pokusů pro funkci $fexp2(x)$ 47
Tabulka 9: Výsledky pokusů pro funkci $fexp3(x)$ 48
Tabulka 10: Výsledky pokusů pro funkci $flog1(x)$ 49
Tabulka 11: Výsledky pokusů pro funkci $flog2(x)$ 49
Tabulka 12: Výsledky pokusů pro funkci $flog3(x)$ 49
Tabulka 13: Výsledky pokusů pro funkci $fpol1(x)$ při hledání 1. kořene 51
Tabulka 14: Výsledky pokusů pro funkci $fpol1(x)$ při hledání všech kořenů.....51
Tabulka 15: Výsledky pokusů pro funkci $fpol2(x)$ při hledání násobného kořene 51
Tabulka 16: Výsledky pokusů pro funkci $fpol2(x)$ při hledání všech kořenů.....51
Tabulka 17: Výsledky pokusů pro funkci $fpol3(x)$ při hledání násobného kořene 52
Tabulka 18: Výsledky pokusů pro funkci $fpol3(x)$ při hledání všech kořenů.....52

ÚVOD

Současná matematika, ač značně rozvinutá a prozkoumaná, dokáže přesně a jednoduše vyřešit pouze nepatrný zlomek nelineárních rovnic pomocí různých vzorců, k tomu navíc existuje pravděpodobnost, že výsledek nelze vyjádřit s konečným počtem desetinných číslic. V takových případech se budeme muset spokojit s dostatečně přesným odhadem, k jeho získání už ale naštěstí známe velké množství numerických metod, z nichž mnohé spadají do kategorie iteračních. Tímto pojmem označujeme takové metody, jejichž způsob užití spočívá v opakující se aplikaci určité operace, přičemž výsledek, který při každé aplikaci získáme, se postupně přibližuje přesnému řešení. Důvodem pro existenci většího počtu iteračních metod je jednoduše fakt, že ne každou metodu lze použít na libovolnou funkci. Tato práce se tedy bude věnovat deseti vybraným metodám a pokusí se dát čtenáři skrze teoretický výklad a následné praktické experimenty určitý přehled, která metoda je pro daný typ příkladu nejefektivnější, nebo alespoň funkční.

Pro účely analýzy řešení je pak součástí práce i kompaktní aplikace, která čtenáři poskytne jednoduchý přehled o efektivitě všech popsaných metod v konkrétním případě.

1 KAPITOLA

1.1 Definice problému

Mějme funkci $f: \mathbb{R} \to \mathbb{R}$. Naším úkolem je najít přibližnou hodnotu kořene funkce f, tj. takovou hodnotu x^*, pro kterou platí

$$f(x^*) = 0. \tag{1.1}$$

V praxi se může běžně stát, že taková funkce má kořenů hned několik, nebo naopak žádný, a tak je vhodné její průběh nejprve prošetřit jinou technikou, a určit takové intervaly, pro které platí následující věta:

Věta 1: *Nechť je funkce $f(x), x \in \langle a, b \rangle$, která nabývá v krajních bodech intervalu hodnot s opačnými znaménky, tj. $f(a)f(b) < 0$. Pak v tomto intervalu leží minimálně jeden kořen funkce $f(x)$. Pokud navíc platí, že $f'(x)$ má na celé délce intervalu neměnící se znaménko a je různá od nuly, existuje na intervalu právě jeden kořen x^*.*

Postupu, kdy se průběh funkce vyšetří a určí se intervaly splňující podmínky ve Větě 1 se říká separace kořenů rovnice. Při zadání intervalu, který obsahuje více než jeden kořen, dokáže většina metod konvergovat k jednomu z nich, nicméně nelze předem odhadnout, který to bude. Proto je vhodné provést důkladnou separaci kořenů, abychom mohli použít metody pro nalezení konkrétního kořene, který chceme.

1.2 Počáteční odhad

Iterační metody vyžadují ke své funkci několik vstupních parametrů. Prvním z nich je počáteční aproximace neboli odhad řešení zkoumané funkce. Tím je například interval $\langle a, b \rangle$, který splňuje podmínky uvedené v první části Věty 1. Druhou možností je, že metoda vyžaduje pouze jeden bod – odhad přibližného řešení. Tyto vstupní parametry lze získat pouhým pohledem na graf funkce, máme-li ho k dispozici nebo vytvořením tabulky $[x_i,\ f(x_i)]$, do které zaneseme funkční hodnoty $f(x_i)$ pro námi zvolené hodnoty x_i. Na základě Věty 1 nám stačí najít dvě po sobě jdoucí hodnoty x_i a x_{i+1}, které mají funkční hodnoty s opačným znaménkem, a ty použít jako krajní body intervalu, případě jen jednu z nich jako jednobodový odhad.

1.3 Konvergence metody

Konvergence metody nám zaručuje, že k dostatečně přesnému řešení dojdeme v konečném počtu iterací. Jak již bylo řečeno dříve, cílem iteračních metod není najít přesné řešení, ale pouze se k němu přiblížit na určitou uspokojivou přesnost. Z principu fungování iteračních metod pak vychází, že v každé iteraci bude existovat nějaká chyba $e_i = x_i - x^*$, kde x^* je přesné řešení a x_i je odhad řešení v i-té iteraci. Pokud výsledkem iterační metody není přibližné řešení, ale interval, který řešení obsahuje, považujme za chybu jeho velikost, tj. $e_i = b_i - a_i$.

Metoda konverguje k přesnému řešení, pokud

$$\lim_{i \to \infty} |e_i| = 0.$$

Konvergence jednotlivých metod je u každé rozdílná, ať už jde rychlost, nebo zda vůbec konverguje. Z toho důvodu bude konvergence popsána v druhé kapitole pro každou metodu zvlášť, včetně jejího důkazu.

Uveďme si nyní další pojem z oblasti konvergence, a to tzv. řád konvergence. Mějme vztah

$$\lim_{i \to \infty} \frac{|e_{i+1}|}{|e_i|^r} = C, \tag{1.2}$$

kde C je reálné číslo různé od 0 a r je řád konvergence. Nejčastěji rozlišujeme tři druhy konvergence:

- lineární, pokud $r = 1$ a $C < 1$,
- superlineární, pokud $1 < r < 2$,
- kvadratická, pokud $r = 2$.

Řád konvergence nám pro danou metodu říká, kolikrát více platných cifer získáme v každé iteraci. V případě lineární konvergence tak získáme s každou iterací stejný počet platných číslic a s kvadratickou konvergencí se počet nových platných číslic s každou iterací zdvojnásobuje.

Konstantu C můžeme taktéž nazvat rychlost konvergence. Čím nižší je hodnota konstanty, tím rychlejší je konvergence, neboť v blízkosti kořene platí

$$|e_{i+1}| \approx C|e_i|^r.$$

Obecně lze říci, že kvadratická konvergence je rychlejší než superlineární, a ta je rychlejší než lineární, nicméně samotná konstanta C může tyto rychlosti značně ovlivnit. například lineární konvergence s C = 0,001 bude mnohem rychlejší, než kvadratická konvergence s C = 0,8.

Každá metoda konverguje jinou rychlostí, což závisí především na volbě počáteční aproximace. Obecně platí, že čím rychleji metoda konverguje, tím přesnější počáteční odhad vyžaduje. V praxi je tak ideální zvolit nejprve pomalejší metodu, která dokáže relativně spolehlivě upřesnit řešení funkce, a její výsledek následně použít pro jemnější a rychlejší metodu.

Rychlost konvergence nám říká, jak rychle se přiblížíme k přesnému řešení s každou iterací. Je však důležité si uvědomit, že skutečná rychlost výpočtu (například počítačovým programem) závisí na ceně iterace, tj. počtu a typu výpočetních operací, které je potřeba vykonat pro získání výsledku. Rychle konvergující metody mají zpravidla drahé iterace, a v praxi tak mohou být i přes jejich nižší počet pomalejší [9].

1.4 Ukončení řešení

Protože níže zkoumané metody prvotní odhad k řešení pouze přibližují, je velice pravděpodobné, že se k přesnému řešení nedostaneme v konečném počtu iterací. Z tohoto důvodu musíme definovat nějaké podmínky, které nám zastaví výpočet ve chvíli, kdy už lze považovat výsledek za dostatečně dobrý.

Úplně nejjednodušší podmínkou pro ukončení výpočtu je stanovit maximální počet iterací. Obecně lze však toto řešení považovat za nevhodné, protože u výsledku nemáme absolutně žádné informace o jeho přesnosti.

Vhodnějším způsobem je tedy výpočet zastavovat po dosažení určitě požadované přesnosti. V První řadě si definujeme požadovanou přesnost $\varepsilon > 0$ a výpočet zastavíme, pokud je $f(x_i) < \varepsilon$. Toto kritérium může fungovat, pokud je funkce dobře podmíněná, tj. křivka funkce svírá v průsečíku s osou x velký úhel. Pokud je ale tento úhel malý, interval, ve kterém $f(x_i) < \varepsilon$ může být příliš velký a za přibližné řešení může být označena hodnota, která je od přesného řešení stále poměrně daleko.

Jako další možnost se nám nabízí sledovat pokrok v řešení po jednotlivých iteracích. Zvolenou přesnost e nebudeme porovnávat s funkční hodnotou v nalezeném bodě, ale s velikostí posunu přibližného řešení po každé iteraci. Toto kritérium definujeme jako $|x_i - x_{i-1}| < \varepsilon$. Pokud se přibližné řešení v každé iteraci posouvá jen minimálně, můžeme se domnívat, že je již velmi blízko přesnému řešení. Toto kritérium ale ve skutečnosti trpí stejným nedostatkem jako předchozí varianta, jen obráceně, tj. pokud je křivka funkce k ose x téměř kolmá, může se řešení v každé iteraci přibližovat s minimálními rozdíly, avšak hodnota $f(x_i)$ bude stále značně rozdílná od 0.

Závěrem tedy můžeme říci, že neexistuje žádné univerzální zastavovací kritérium, které nám zajistí dostatečně přesný výsledek pro jakýkoliv problém, který řešíme. Volba kritéria tak bude vždy záviset na použité metodě a charakteru zkoumané funkce.

2 KAPITOLA

2.1 Základní metody

2.1.1 Metoda bisekce

Metoda bisekce, též známa jako metoda půlení intervalu, je asi nejjednodušší a nejméně náročná na vlastnosti zkoumané funkce. Její jednoduchost je ale vyvážena ne příliš velkou rychlostí.

Vstupem metody je interval $\langle a, b\rangle$, o kterém víme, že obsahuje kořen funkce. Cílem metody je postupně zmenšovat tento interval jeho půlením tak, aby stále obsahoval přesné řešení. Podmínkou pro správné fungování metody je spojitost funkce a splnění podmínky dané ve Věta **1** na celé délce zadaného intervalu.

Algoritmus

V každé iteraci dojde nejprve k nalezení bodu m, který je středem zadaného intervalu:

$$m = \frac{a + b}{2}$$

Tento způsob výpočtu nemusí být vhodný v počítačové logice, kde jsme omezeni maximální hodnotou použitého datového typu, jelikož v extrémních případech může dojít k tzv. přetečení neboli situaci, kdy součet velkých čísel může přesáhnout tuto hodnotu. Použijeme tedy takový výpočet, který nevyužívá sčítání:

$$m = a + \frac{b - a}{2} \tag{2.1}$$

V dalším kroku otestujeme, zda $f(m) = 0$. Pokud ano, dostali jsme přesné řešení a výpočet můžeme ukončit. V opačném případě otestujeme interval $\langle a, m\rangle$ na pravidlo o opačnosti znamének. Pokud $f(a)f(m) < 0$, pak víme, že tento interval obsahuje řešení, jinak řešení obsahuje druhý interval $\langle m, b\rangle$. Bod m tedy označíme podle výsledku porovnávání znamének novým krajním bodem intervalu a celý proces opakujeme, dokud není splněna ukončovací podmínka, tj. $(b - a) < \varepsilon$, kde ε je předem zvolená požadovaná přesnost.

Konvergence

Velikost intervalu, nebo též chyba, může být po i iteracích zapsána dvěma způsoby: $|b_i - a_i| = \frac{b-a}{2^i}$. Použijeme-li vztah (1.2) o řádu konvergence, do kterého za dosadíme $r = 1$, dostaneme

$$\lim_{i\to\infty} \frac{\|e_{i+1}\|}{\|e_i\|^r} = \lim_{i\to\infty} \frac{\frac{b-a}{2^{i+1}}}{\frac{b-a}{2^i}} = \frac{1}{2}$$

Lze tedy říci, že metoda bisekce konverguje lineárně s rychlostí 1/2.

Velkou výhodou metody bisekce je, že můžeme předem vypočítat potřebný počet iterací. Stačí nám k tomu jen počáteční interval a požadovaná přesnost ε. Výsledek iterace je dostatečně blízko přesnému řešení právě tehdy, když platí $\frac{(b-a)}{2^i} < \varepsilon$, což lze upravit na $\frac{(b-a)}{\varepsilon} < 2^i$. Počet iterací tedy lze vypočítat jako

$$i \geq \log_2 \frac{(b-a)}{\varepsilon}, \qquad i \in \mathbb{N}. \tag{2.2}$$

Výhody a nevýhody

Výhodou metody je především nízká výpočetní cena, které je dosaženo tím, že metoda nijak nevyužívá například derivaci funkce. Lze jí tedy bez problémů použít i na funkce, které nejsou diferencovatelné. Nevýhodou je pak rychlost konvergence, která je v porovnání s jinými metodami značně pomalá. Obecně je tedy metoda bisekce využívána k získání přesnější počáteční aproximace pro jinou, rychlejší metodu, která by se mohla potýkat s nejistotou konvergence při volbě příliš vzdálené počáteční aproximace.

2.1.2 Metoda Regula falsi

Tato metoda je vylepšením metody půlení intervalu, která, jak bylo zmíněno v jejím hodnocení výše, nevyužívá k získání další aproximace žádné bližší informace o průběhu funkce a jejím charakteru. Metoda regula falsi pro získání dalšího bodu využívá aproximace funkce pomocí

přímky vedené body $[x_0, f(x_0)]$ a $[x_1, f(x_1)]$, jejíž řešení je další aproximací. Stejně jako v případě bisekce je druhý krajní bod nového, menšího intervalu zvolen tak, aby nový interval splňoval podmínky ve Věta 1.

Iterační funkce této metody je tedy odvozena z rovnice sečny procházející krajními body intervalu, tj. $y = f(x_1) + \frac{f(x_1)-f(x_0)}{x_1-x_0}(x - x_1)$. Hledáme průsečík této funkce s osou x, dosadíme tedy $y = 0$ a vyjádříme x:

$$x = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_1).$$

Obecný tvar iterační funkce tedy můžeme zapsat takto:

$$x_{i+1} = x_i - \frac{x_i - x_s}{f(x_i) - f(x_s)} f(x_i), \tag{2.3}$$

kde index s označuje nejvyšší index, pro který platí $f(x_{i+1})f(x_s) < 0$.

Metoda regula falsi konverguje lineárně. Důkaz je poměrně obsáhlý, nalézt ho můžeme například v [8].

Algoritmus

Stejně jako v případě metody bisekce je vstupem interval, o kterém víme, že obsahuje kořen funkce. Další podmínkou je pak aby funkční hodnoty v krajních bodech intervalu měly opačná znaménka a funkce byla na celém intervalu spojitá.

V případě, že druhá derivace funkce na celém zvoleném intervalu nemění znaménko, nastává zvláštní situace, kdy ve všech iteracích až do nalezení dostatečně přesného řešení se jeden z krajních bodů intervalu nemění. Z výpočetního hlediska tak stojí za zvážení, zda se nevyplatí provést nejdříve tuto výpočetně náročnější kontrolu, a následně pak v průběhu hledání řešení ušetřit několik operací tím, že nebude třeba kontrolovat opačnost znamének, protože jeden z krajních bodů vždy zůstane stejný. Tímto „pevným“ bodem bude pak ten, pro který platí, že znaménko funkční hodnoty je stejné, jako znaménko druhé derivace na zvoleném intervalu. Úprava iterační funkce potom spočívá pouze v nahrazení členu x_s buď bodem x_0 nebo x_1.

Volba ukončovacího kritéria opět závisí na průběhu funkce v okolí kořene, volit můžeme mezi

velikostí kroku $|x_{i+1} - x_i| < \varepsilon$ či funkční hodnotou v novém bodě $|f(x_{i+1})| < \varepsilon$.

Výhody a nevýhody

Vlastnosti metody regula falsi jsou díky podobnosti s metodou bisekce víceméně shodné. Vynecháme-li volitelnou kontrolu průběhu funkce pomocí druhé derivace, je metoda aplikovatelná i na nediferencovatelné funkce. Konvergence je navíc zaručena pro jakýkoliv počáteční interval splňující podmínky uvedené v úvodu podkapitoly popisující algoritmus. Cena jednotlivých iterací je však v porovnání s metodou bisekce vyšší z důvodu vyššího počtu vyhodnocování funkce.

2.1.3 Metoda prosté iterace

Metoda prosté iterace je taktéž známa jako fixed-point iterace. Její fungování vychází z předpokladu, že pro danou funkci $f(x)$ existuje nějaká funkce $g(x)$, pro kterou platí, že pevný bod funkce $g(x)$, tj. bod, pro který platí $g(x) = x$, je zároveň kořenem funkce $f(x)$.

Dokažme si nyní toto tvrzení. Mějme spojitou funkci $g(x) \in \langle a, b\rangle,\ \forall x \in \langle a, b\rangle$. Pak tato funkce má v intervalu $\langle a, b\rangle$ pevný bod. Pokud $g(a) \neq a$ a $g(b) \neq b$, předpokládáme, že $g(a) > a$ a $g(b) < b$. Mějme nyní funkci $h(x) = g(x) - x$. Pak $h(a) = g(a) - a > 0$ a $h(b) = g(b) - b < 0$. Tímto jsme splnili podmínku opačnosti znamének uvedenou ve Věta 1, a tím pádem jsme dokázali existenci takového $x^* \in \langle a, b\rangle$, pro které platí $g(x^*) = x^*$ a zároveň $h(x^*) = 0$.

Dokažme si existenci jediného pevného bodu na intervalu $\langle a, b\rangle$. K tomu nám poslouží Lipschitzova podmínka. Mějme funkci $g \in C\langle a, b\rangle$ a konstantu $q \in \langle 0, 1)$, pro které platí

$$|g(x) - g(y)| \leq q|x - y|, \qquad \forall x, y \in \langle a, b\rangle.$$

Dále mějme dva pevné body, ξ, η. Podle výše zmíněných pravidel by muselo platit

$$|\xi - \eta| = |g(\xi) - g(\eta)| \leq q|\xi - \eta| < |\xi - \eta|,$$

což není možné, a existence více než jednoho pevného bodu je tak vyloučena.

Tato metoda tedy spočívá v odvození jedné či více funkcí $g(x)$. Jednou variantou je použít vztah $f(x) = g(x) - x$, tedy $g(x) = f(x) + x$, nebo vyjádřením x ze zkoumané funkce. Pro získané funkce $g(x)$ poté hledáme pevný bod.

Uveďme si několik příkladů funkcí $g(x)$. Mějme například funkci $f(x) = x^3 - x - 1$. Výše zmíněnými postupy odvodíme:

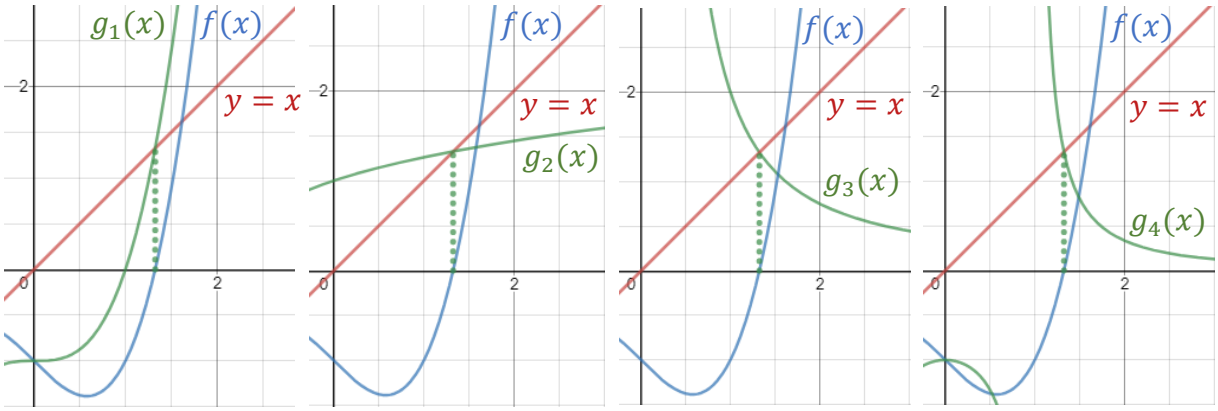
(1) $g_1(x) = x^3 - 1$,

(2) $g_2(x) = \sqrt[3]{x+1}$,

(3) $g_3(x) = \frac{x+1}{x^2}$,

(4) $g_4(x) = \frac{1}{x^2-1}$.

Sestrojme nyní grafy těchto funkcí a doplňme je o graf funkce $y = x$, tj. funkce, jejíž průsečík s funkcí $g(x)$ značí její pevný bod. Z obrázku je zřejmé, že všechny odvozené funkce prochází stejným pevným bodem, který se shoduje s kořenem funkce $f(x)$. Zdaleka ne všechny funkce ale lze použít jako iterační funkci. Odůvodnění přijde v sekci o konvergenci.



Obrázek 1: Pevný bod iteračních funkcí g(x)

Algoritmus

Místo přímého hledání kořene funkce $f(x)$ hledáme pevný bod funkce $g(x)$. Vstupem metody je počáteční aproximace v podobě jednoho bodu x_0. Další body pak získáváme opakovanou

aplikací iterační funkce $g(x)$, tedy

$$x_{i+1} = g(x_i). \tag{2.4}$$

Pokud $x_{i+1} = x_i$, nalezli jsme přesné řešení a výpočet můžeme ukončit. V opačném případě pokračujeme, dokud není splněna zvolená ukončovací podmínka. Za tu se nabízí volba rozdílu mezi iteracemi, kterou můžeme upravit díky tomu, že při konvergenci k pevnému bodu dochází ke snižování rozdílu mezi x_i a $g(x_i)$, použijeme tedy $|x_i - g(x_i)| < \varepsilon$.

Konvergence

Ačkoliv různých variant funkce $g(x)$ je několik, k výsledku nelze dojít libovolnou funkcí. Uveďme si pojem přitahující (atraktivní) a odpuzující (repulzivní) pevný bod. Mějme funkci $g(x)$ splňující výše zmíněné podmínky s právě jedním pevným bodem ξ. Jestliže pro jakékoliv x v blízkosti bodu ξ platí

$$\left|\frac{g(x) - g(\xi)}{x - \xi}\right| < 1, \tag{2.5}$$

ξ je přitahující pevný bod. Pokud naopak platí

$$\left|\frac{g(x) - g(\xi)}{x - \xi}\right| > 1, \tag{2.6}$$

jde o odpuzující bod. Pokud má funkce odpuzující pevný bod, nebude možné ji použít pro vyřešení původní funkce, jelikož bude divergovat.

Uveďme si Lagrangeovu větu o střední hodnotě:

Věta 2: *Nechť $a, b \in \mathbb{R}$, $a < b$, f je funkce spojitá na intervalu $\langle a, b\rangle$ a nechť $f'(x)$ existuje pro všechna $x \in (a, b)$. Pak existuje $\xi \in (a, b)$, splňující*

$$f'(\xi) = \frac{f(b) - f(a)}{(b - a)} \tag{2.7}$$

Všimněme si, že tento výraz je stejný jako vztahy (2.5) a (2.6) pro určení charakteru pevného bodu. Nahrazením tedy získáváme, že pro libovolný bod ζ v blízkosti pevného bodu ξ platí, že pokud $g'(\zeta) < 1$, pevný bod je přitahující a naopak.

Dokažme si nyní konvergenci posloupnosti $\{x_i\}$.

Věta 3: *Nechť $g(x)$ je spojitě diferencovatelná funkce na neprázdném intervalu $\langle a, b\rangle$, pro kterou platí $g(x) \in \langle a, b\rangle \; \forall x \langle a, b\rangle$. Nechť existuje také číslo $K \in \langle 0, 1)$, pro které platí $|g'(x)| \leq K \quad \forall x \in \langle a, b\rangle$. Potom na daném intervalu existuje pouze jediný pevný bod, ke kterému posloupnost $\{x_i\}$ konverguje pro libovolné $x_0 \in \langle a, b\rangle$.*

Mějme chybu v *i*-té iteraci $e_i = x_i - \xi$, kde ξ je pevný bod funkce $g(x)$. Tu lze podle iterační funkce a definice pevného bodu napsat také jako $e_i = g(x_{i-1}) - g(\xi)$. Zvolme si nyní libovolné $\eta_i \in \langle \xi, x_{i-1}\rangle$ a položme následující vztah:

$$e_i = g(x_{i-1}) - g(\xi) = g'(\eta_i)(x_{i-1} - \xi) = g'(\eta_i)e_{i-1}.$$

Z věty 2 plyne, že $|g'(\eta_i)| \leq K < 1$, a tedy $|e_i| \leq K|e_{i-1}|$, neboli

$$|e_i| \leq K|e_{i-1}| \leq K^i|e_0|$$

Protože víme, že $0 \leq K < 1$, můžeme říci, že

$$\lim_{i\to\infty}|e_i| = \lim_{i\to\infty} K^n|e_0| = 0.$$

Tímto jsme dokázali konvergenci fixed-point iterace při splnění podmínek uvedených ve Věta **3**. Všimněme si, že posloupnost konverguje za podmínky, že $g'(x_i) < 1$, což se shoduje s výše popsanou definicí přitahujícího pevného bodu.

Řád a rychlost konvergence si prokažme pomocí následující věty:

Věta 4: *Nechť je funkce $g(x)$ spojitě diferencovatelná na celém intervalu obsahujícím pevný bod ξ, a nechť $0 < g'(\xi) < 1$. Pak fixed-point iterace konverguje lineárně s rychlostí $|C = g'(\xi)|$.*

Opět využijme Větu 2 o střední hodnotě s $\eta \in \langle x_i, \xi\rangle$:

$$g'(\eta_i) = \frac{g(x_i) - g(\xi)}{x_i - \xi} = \frac{x_{i+1} - \xi}{x_i - \xi},$$

a tedy pro řád konvergence $r = 1$ platí:

$$\lim_{i\to\infty}\frac{|e_{i+1}|}{|e_i|}=\lim_{i\to\infty}\frac{|x_{i+1}-\xi|}{|x_i-\xi|}=\lim_{i\to\infty}|g'(\eta_i)|=|g'(\xi)|=C<1.$$

Výhody a nevýhody

Výhodou metody prosté iterace je opět její jednoduchost na implementaci – k vypočítání iterace stačí pouze jediné vyhodnocení funkce. Celková cena iterace se ale může diametrálně lišit, jelikož nevíme, jaké všechny operace zahrnuje právě vyhodnocení funkce. Za výhodu se dá taktéž považovat, že metoda může ve výjimečných případech konvergovat i při nesplnění podmínek, konkrétně pak podmínky $g'(\xi)<1$. Příkladem je funkce $\sin(x)$. Ta má pevný bod v 0 a derivace v tomto bodě je rovna jedné. I přes to bude posloupnost konvergovat.

2.1.4 Newtonova metoda

Tato metoda je jednou z nejrychleji konvergujících metod, a je z ní odvozeno několik dalších metod pro různé specifické případy. My si nyní představíme její základní podobu.

Newtonova metoda spočívá v odvození tečny k funkci v bodě aproximace, a jako následující odhad je pak zvolen průsečík této tečny s osou x, proto se jí též říká metoda tečen. Tato metoda je nejefektivnější v těsném okolí kořene funkce, proto je v praxi ideální použít nejprve jinou metodu, kterou tento dostatečně blízký odhad získáme, a poté teprve aplikovat Newtonovu metodu.

Odvoďme si nyní iterační funkci pro Newtonovu metodu. Jak bylo řečeno výše, následující odhad x_{i+1} je řešením tečny vedené k funkci f v bodě předchozího odhadu x_i. Vzorec pro výpočet tečny je $y-y_0=f'(x_0)(x-x_0)$, kde x_0 a y_0 jsou souřadnice bodu dotyku. Nahraďme je tedy hodnotami x_i a $f(x_i)$, a dále $y=0$ a $x=x_{i+1}$. Dostáváme

$$-f(x_i)=f'(x_i)(x_{i+1}-x_i).$$

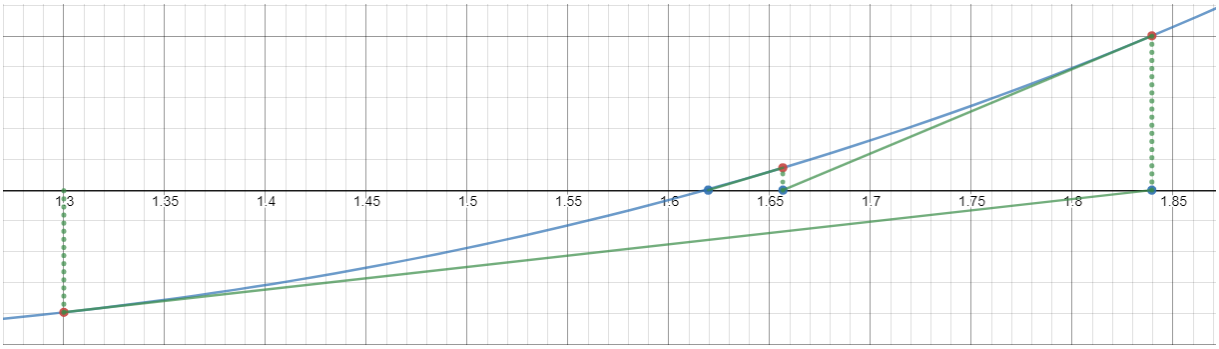
Z této rovnice několika úpravami vyjádříme x_{i+1} a zíkáme

$$x_{i+1}=x_i-\frac{f(x_i)}{f'(x_i)}. \tag{2.8}$$

Algoritmus

Vstupním parametrem Newtonovy metody je, kromě požadované přesnosti, pouze počáteční odhad. Pomocí výše zmíněného vzorce se vypočítá tečna v bodě odhadu, a následně její řešení, které je použito jako odhad pro další iteraci.

Přirozeně před dalším výpočtem ověříme, zda nový odhad není přesným řešením funkce, abychom ukončili výpočet, přestože pravděpodobnost trefy do přesného řešení je v případě, že řešíme funkci danou křivkou, mnohem nižší než u předchozích metod. Jako zastavovací kritéria se nám opět nabízí $|f(x_i) < \varepsilon|$ nebo $|x_i - x_{i-1}| < \varepsilon$, dle tvaru křivky. Za zvážení však stojí doplnění o kritérium maximální odchylky od původního odhadu, například v podobě $|x_0 - x_i| > \varepsilon_{max}$, jelikož nízká hodnota derivace funkce může vést ke značné divergenci.



Obrázek 2: Ukázka prvních tří iterací Newtonovy metody

Konvergence

Podmínkou pro konvergenci Newtonovy metody je spojitost funkce f na celém intervalu $\langle a, b\rangle$ a existence $f'(x)$ pro libovolné $x \in \langle a, b\rangle$. Newtonova metoda konverguje kvadraticky, což si dokážeme nyní.

Pro tento účel pohlížejme na iterační funkci Newtonovy metody jako na fixed-point iteraci, tj. $g(x) = x - \frac{f(x)}{f'(x)}$, pro kterou hledáme pevný bod $g(\xi) = \xi$.

Věta 5: *Nechť funkce $f(x)$ je dvakrát spojitě diferencovatelná na intervalu $\langle a, b\rangle$ a nechť $\xi \in \langle a, b\rangle$ je kořenem funkce $f(x)$ a $f'(\xi) \neq 0$. Pak existuje takové $\delta > 0$, pro které platí,*

že posloupnost $\{x_i\}$ *konverguje ke kořeni* ξ *kvadraticky pro každou počáteční aproximaci* $x_0 \in \langle \xi - \delta, \xi + \delta \rangle$.

Jelikož předpokládáme jednoduchý kořen, tj. $f'(\xi) \neq 0$, tak existuje nějaké $\delta_1 > 0$, pro které platí, že i $f'(x) \neq 0 \; \forall x \in \langle \xi - \delta_1, \xi + \delta_1 \rangle$, čili i iterační funkce $g(x)$ je na tomto intervalu definovaná a spojitá, a protože ve Věta 5 předpokládáme, že je funkce $f(x)$ dvakrát spojitě diferencovatelná, je i $g'(x)$ definovaná a spojitá. Toho využijeme v následujícím vztahu:

$$g'(x) = 1 - \frac{f'^2(x) - f(x)f''(x)}{f'^2(x)} = \frac{f(x)f''(x)}{f'^2(x)}.$$

Protože předpokládáme, že $f(\xi) = 0$, je i

$$g'(\xi) = \frac{f(\xi)f''(\xi)}{f'^2(\xi)} = 0.$$

Tím pádem existuje nějaké $\delta > 0$, pro které platí, že $|g'(x)| < 1 \; \forall x \in \langle \xi - \delta_1, \xi + \delta_1 \rangle$.

Vrátíme-li se opět do kapitoly o metodě prosté iterace, vyčteme z Věty 3 o konvergenci této metody, že pokud je na daném intervalu derivace menší než 1, posloupnost bude konvergovat k pevnému bodu, čímž je dokázána i konvergence Newtonovy metody při volbě počáteční aproximace $x_0 \in \langle \xi - \delta_1, \xi + \delta_1 \rangle$.

Tvrzení, že Newtonova metoda konverguje kvadraticky, dokážeme pomocí Taylorova rozvoje funkce $f(\xi)$ okolo bodu x_i:

$$f(\xi) = f(x_i) + f'(x_i)(\xi - x_i) + \frac{1}{2}f''(\eta_i)(\xi - x_i)^2, \qquad \eta_i \in \langle \xi, x_i \rangle.$$

Víme, že $f(\xi) = 0$, a $f'(x_i) \neq 0$, tudíž po vydělení $f'(x_i)$ získáme

$$\frac{f(x_i)}{f'(x_i)} + (\xi - x_i) = -\frac{1}{2}\frac{f''(\eta_i)}{f'(x_i)}(\xi - x_i)^2,$$

Podle vztahu (2.8) nahradíme $\frac{f(x_i)}{f'(x_i)}$ za $x_i - x_{x+1}$ a po vzájemném odečtení x_i získáme

$$x_{i+1} - \xi = \frac{1}{2}\frac{f''(\eta_i)}{f'^{(x_i)}}(\xi - x_i)^2, \qquad e_{i+1} = e_i^2 \frac{1}{2}\frac{f''(\eta_i)}{f'(x_i)},$$

tedy

$$\frac{|e_{i+1}|}{|e_i|^2} = C, \qquad C = \frac{1}{2}\frac{f''(\eta_i)}{f'(x_i)}.$$

Newtonova metoda tedy konverguje kvadraticky s rychlostí C.

Na závěr podkapitoly o konvergenci Newtonovy metody si uveďme Fourierovy podmínky, které nám ji zaručí:

(1) $f(a)f(b) < 0$,

(2) $f'(x) \neq 0,\ \forall x \in \langle a, b \rangle$,

(3) $f''(x)$ na celé délce intervalu nemění znaménko,

(4) jako počáteční aproximaci x_0 volíme takový bod, pro který platí $f(x_0)f''(x_0) > 0$.

Modifikace pro rovnice s násobným kořenem

Druhá z výše uvedených Fourierových podmínek nám říká, že funkce nesmí mít nulovou derivaci v žádném bodě na intervalu, ve kterém volíme počáteční aproximaci. Té (a v některých případech ani dalším) však nelze vyhovět, má-li funkce násobný kořen, tj. takový kořen, pro který platí

$$f(\xi) = f'(\xi) = f''(\xi) = \ldots = f^{(m-1)}(\xi) = 0, \quad f^{(m)}(\xi) \neq 0,$$

kde m značí násobnost kořene. Naštěstí pro nás však Fourierovy podmínky konvergenci pouze zaručují, jejich nesplnění tak neznamená že metoda nebude konvergovat. Ve skutečnosti Newtonova metoda konverguje i k násobným kořenům, avšak už ne kvadraticky. V [7] lze nalézt důkaz, že Newtonova metoda konverguje ke kořeni s násobností $m > 1$ lineárně s rychlostí $\frac{m-1}{m}$.

Původní iterační funkci (2.8) však můžeme drobně upravit na

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)}. \tag{2.9}$$

Tato upravená iterační funkce již bude konvergovat kvadraticky. Její hlavní nevýhodou je však nutnost znalosti násobnosti kořene. Pokud tuto hodnotu neznáme přesně, špatný odhad může vést k pomalejší konvergenci než u základní podoby metody nebo i k divergenci.

Výhody a nevýhody

Tato metoda konverguje v těsné blízkosti kořene velice rychle, což je zřejmé při pohledu na graf funkce. Postupným zmenšováním intervalu, ve kterém pracujeme, lze čím dál přesněji aproximovat křivku funkce přímkou, pro kterou Newtonova metoda z principu fungování najde řešení ihned.

Nevýhodou je pak nepředvídatelnost při volbě příliš vzdáleného odhadu, konvergence v takových podmínkách totiž není zaručena. Další nevýhodou je pak cena iterace. Jelikož v předchozích metodách jsme si vystačili s jediným vyhodnocením funkce v každé iteraci, zde musíme přidat její derivaci, která může být z výpočetního hlediska poměrně náročná. V praxi však tato negativa vyvažuje již zmíněná rychlost konvergence, a tak se při splnění podmínek jedná o velmi vhodnou metodu i za cenu dražší iterace.

2.1.5 Metoda sečen

Ukázali jsme, že Newtonova metoda konverguje velice rychle, obzvláště pak s dobrou počáteční aproximací. Jejím velkým nedostatkem je však potřeba vyhodnocení derivace funkce, která může být velmi náročná. Metoda sečen vychází z Newtonovy metody, a tedy využívá podobného principu, ale obejde se bez derivace funkce. Tato výhoda je ale samozřejmě vyvážena o něco pomalejší konvergencí, protože na rozdíl od Newtonovy metody nemáme k dispozici přesné informace o průběhu funkce získané derivací, ale pouze jejich přibližnou aproximaci.

Metoda sečen vychází z aproximace derivace funkce v daném bodě pomocí následujícího

vztahu:

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Po dosazení do iterační funkce Newtonovy metody a krátké úpravě dostáváme následující vztah:

$$x_{i+1} = x_i - \frac{f(x_i)}{\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \tag{2.10}$$

Název vychází z geometrické reprezentace této metody. V každé iteraci se použijí poslední dvě aproximace a sestrojí se sečna ke křivce funkce procházející právě body $[x_i, f(x_i)]$ a $[x_{i-1}, f(x_{x-1})]$. Jako další bod se pak použije průsečík této sečny s osou x. Čím blíže k sobě tyto body jsou, tím lépe je aproximována tečna, kterou bychom získali Newtonovou metodou. To je také důvod, proč v praxi metoda sečen konverguje zprvu pomaleji, avšak jak se aproximace blíží kořeni (a body se přibližují), konvergence značně zrychluje.

Algoritmus

Jelikož se jedná o dvoukrokovou metodu, tj. každá iterace využívá dvou předchozích aproximací, je potřeba na začátku uvést jako počáteční odhad dva body. Pomocí iterační funkce (2.10) se pak získá třetí bod, který se použije v další iteraci společně s bodem s vyšším indexem z předchozí iterace. Na počátku tedy volíme x_0 a x_1 a vypočítáme x_2. V další iteraci použijeme body x_1 a x_2, získáme x_3 a tak dále.

Zastavovací kritéria mohou být díky podobnosti Newtonově metodě volena stejně, tj. $|f(x_i) < \varepsilon|$ a $|x_i - x_{i-1}| < \varepsilon$. Stejně tak opět stojí za zvážení i maximální odchylka pro následující aproximaci $|x_0 - x_i| > \varepsilon_{max}$.

Konvergence

Konvergence metody sečen je taktéž silně závislá na volbě počáteční aproximace. Jak již bylo řečeno, vlivem pouhé aproximace derivace je konvergence metody sečen pomalejší, nicméně je stále rychlejší než lineární.

Věta 6: *Nechť je funkce f dvakrát spojitě diferencovatelná na nějakém okolí U bodu ξ, pro který platí $f(\xi) = 0$ a $f'(\xi) \neq 0$. Pak existuje $\delta > 0$ takové, že posloupnost $\{x_i\}$ generovaná metodou sečen konverguje ke kořeni ξ pro jakoukoliv počáteční aproximaci $x_0, x_1 \in \langle \xi - \delta, \xi + \delta \rangle$ s řádem $(1 + \sqrt{5})/2 \approx 1{,}618$.*

Před samotným důkazem si nejprve odvodíme jinou formu iterační funkce této metody:

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= \frac{x_i\big(f(x_i) - f(x_{i-1})\big) - f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \end{aligned}$$

Dokažme si nyní poslední Větu. Mějme chybu v *i*-té iteraci $e_i = x_i - \xi$. Pak

$$\begin{aligned} e_{i+1} = x_{i+1} - \xi &= \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} - \xi \\ &= \frac{(x_{i-1} - \xi)f(x_i) - (x_i - \xi)f(x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= \frac{e_{i-1}f(x_i) - e_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= e_i e_{i-1}\left(\frac{\frac{f(x_i)}{e_i} - \frac{f(x_{i-1})}{e_{i-1}}}{f(x_i) - f(x_{i-1})}\right) \end{aligned}$$

Protože $f(\xi) = 0$, můžeme tento tvar upravit následovně:

$$e_{i+1} = e_i e_{i-1}\left(\frac{\frac{f(x_i) - f(\xi)}{x_i - \xi} - \frac{f(x_{i-1}) - f(\xi)}{x_{i-1} - \xi}}{f(x_i) - f(x_{i-1})}\right)$$

Nahraďme nyní členy ve jmenovateli výrazem $F(x) = \frac{f(x) - f(\xi)}{x - \xi}$:

$$e_{i+1} = e_i e_{i-1}\left(\frac{F(x_i) - F(x_{i-1})}{f(x_i) - f(x_{i-1})}\right) \tag{2.11}$$

Dle Věty 2 o střední hodnotě dostáváme $F'(\eta_i) = \frac{F(x_i)-F(x_{i-1})}{x_i-x_{i-1}}$, tedy

$$F(x_i) - F(x_{i-1}) = F'(\eta_i)(x_i - x_{i-1}),\ \ \eta_i \in \langle x_i, x_{i-1}\rangle \tag{2.12}$$

Dále mějme Taylorův rozvoj $f(\xi) = f(x) + f'(x)(\xi - x) + \frac{1}{2}f''(\eta)(\xi - x)^2$ pro $\eta \in \langle \xi, x\rangle$, ze kterého vyjádříme $\frac{1}{2}f''(\eta) = \frac{f(\xi)-f(x)-f'(x)(\xi-x)}{(\xi-x)^2}$. Všimněme si, že pravá část tohoto výrazu je ekvivalentní k $F'(x)$, resp. $\left(\frac{f(x)-f(\xi)}{x-\xi}\right)'$, a toho využijeme dosazením do (2.12):

$$F(x_i) - F(x_{i-1}) = \frac{1}{2}f''(\eta_i)(x_i - x_{i-1}).$$

Vraťme se k rovnici (2.11), do které dosadíme právě získaný výraz:

$$e_{i+1} = e_i e_{i-1}\left(\frac{1}{2}f''(\eta_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}\right)$$

Po několika drobných úpravách můžeme opět použít Větu o střední hodnotě, a získáme tak

$$e_{i+1} = e_i e_{i-1}\frac{f''(\eta_i)}{2f'(\theta_i)},\ \ \theta \in \langle x_i, x_{i-1}\rangle \tag{2.13}$$

Jelikož předpokládáme, že f je dvakrát spojitě diferencovatelná a že $f'(\xi) \neq 0$ na nějakém okolí kořene U, můžeme říci, že existuje takové číslo $M \geq \frac{f''(\eta_i)}{2f'(\theta_i)}$, $M \in U$. Při volbě $x_0, x_1 \in U$, která splňují $|e_0|, |e_1| < \frac{1}{M+1}$ lze z (2.13) vyvodit, že $e_{i+1} < \frac{M}{M+1}$, tedy že metoda bude konvergovat. Nyní si dokážeme řád metody neboli takové číslo p, pro které platí $\lim_{i\to\infty}\frac{|e_{i+1}|}{|e_i|^p} = C,\ C > 0$.

Označme si $S_n = \frac{|e_{i+1}|}{|e_i|^p}$, pak $|e_{i+1}| = S_i|e_i|^p$ a $|e_i| = S_{i-1}|e_{i-1}|^p$, tedy

$$|e_{i+1}| = S_i(S_{i-1}|e_{i-1}|^p)^p = S_i S_{i-1}^p |e_{i-1}|^{p^2}.$$

Dosaďme do vzorce pro výpočet řádu konvergence:

$$\frac{|e_{i+1}|}{|e_i||e_{i-1}|} = \frac{S_i S_{i-1}^p |e_{i-1}|^{p^2}}{S_{i-1}|e_{i-1}|^p |e_{i-1}|} = S_i S_{i-1}^{p-1} |e_{i-1}|^{p^2-p-1}$$

Jelikož rychlost C nemůže být nulová, musí platit $\lim_{i\to\infty} \frac{|e_{i+1}|}{|e_i||e_{i-1}|} = S_i S_{i-1}^{p-1} \lim_{i\to\infty} |e_{i-1}|^{p^2-p-1} > 0$. Protože ale $\lim_{i\to\infty} |e_i| = 0$, musí platit $p^2 - p - 1 = 0$, abychom došli k jinému než nulovému výsledku. Jediným kladným řešením této rovnice, a tedy i řádem konvergence je $\frac{1+\sqrt{5}}{2} \approx 1.618$, metoda sečen tedy konverguje superlineárně.

Výhody a nevýhody

Nepochybnou výhodou této metody je že nevyužívá derivaci. Nejen že to může snížit výpočetní náročnost, a tím i cenu iterace, ale hlavně lze tuto metodu použít na nediferencovatelné funkce. Nižší cena iterace může navíc v některých případech i vyvážit ztrátu v rychlosti konvergence oproti Newtonově metodě.

2.1.6 Steffensenova metoda

Než se budeme věnovat samotné metodě, je potřeba se nejprve seznámit s Aitkenovou δ^2-metodou. Jedná se o způsob, jakým lze urychlit konvergenci libovolné lineárně konvergentní řady generované iterační metodou. Mějme tedy posloupnost $\{x_i\}_{i=0}^{\infty}$, $x_i \neq \xi$, $\lim_{i\to\infty} x_i = 0$, $i = 0, 1, 2, \ldots$ splňující podmínky

$$x_{i+1} - \xi = (C + \gamma_i)(x_i - \xi), \quad |C| < 1, \quad \lim_{i\to\infty} \gamma_i = 0, \quad k = 0, 1, 2, \ldots$$

Pak posloupnost $\{\hat{x}\}$ generovaná podle předpisu

$$\hat{x}_i = x_i - \frac{(x_{i+1} - x_i)^2}{x_{i+2} - 2x_{i+1} + x_i} \tag{2.14}$$

Bude splňovat $\lim_{i\to\infty} \frac{\hat{x}_i - \xi}{x_i - \xi} = 0$, a tudíž konvergovat rychleji, než posloupnost $\{x_i\}$. Toto tvrzení je dokázáno v [2].

Nyní, když jsme si představili Aitkenovu metodu, je na čase ji využít v praxi. Ze vzorce (2.14)

je zřejmé, že pro výpočet členu urychlené posloupnosti je třeba znát dva následující členy původní posloupnosti. Vzpomeňme si nyní na metodu prosté iterace a její iterační funkci $x_{i+1} = g(x_i)$. Pak $x_{i+2} = g(x_{i+1}) = g\big(g(x_i)\big)$. Dosaďme tyto výrazy do vzorce (2.14) a upravme (pro přehlednost nahradíme $x_{i+1} = y_i$ a $x_{i+2} = z_i$):

$$x_i - \frac{(y_i - x_i)^2}{z_i - 2y_i + x_i} = \frac{[\![x_i(z)]\!]_i - 2y_i + x_i) - (y_i - x_i)^2}{z_i - 2y_i + x_i}$$

$$= \frac{z_i x_i - 2y_i x_i + x_i^2 - (y_i^2 - 2y_i x_i + x_i^2)}{z_i - 2y_i + x_i} = \frac{z_i x_i - y_i^2}{z_i - 2y_i + x_i}$$

$$= \frac{x_i g\big(g(x_i)\big) - \big(g(x_i)\big)^2}{g\big(g(x_i)\big) - 2g(x_i) + x_i} = \varphi(x_i) \tag{2.15}$$

Takto získanou iterační funkci $x_{i+1} = \varphi(x_i)$ využívá Steffensenova metoda.

Při odvozování iterační funkce jsme vycházeli z metody prosté iterace, kde $g(\xi) = \xi$. Za tohoto předpokladu, a dále že $g'(\xi) \neq 1$, dokážeme pomocí L'Hospitalova pravidla, že

$$\varphi(\xi) = \frac{g\big(g(\xi)\big) + \xi g'\big(g(\xi)\big)g'(\xi) - 2g(\xi)g'(\xi)}{g'\big(g(\xi)\big)g'(\xi) - 2g'(\xi) + 1} = \frac{\xi + \xi g'^2(\xi) - 2\xi g'(\xi)}{g'^2(\xi) - 2g'(\xi) + 1} = \frac{\xi + \xi - \xi}{1}$$
$$= \xi,$$

a tedy že při hledání kořene funkce $f(x)$ pomocí Steffensenovy metody hledáme opět pevný bod funkce $g(x)$, resp. $\varphi(x)$ a jako zastavovací kritérium se tak opět nabízí $|x_i - g(x_i)| \leq \varepsilon$.

Řád metody závisí na zvolené iterační funkci $g(x)$. Je-li tato funkce spojitě diferencovatelná až do řádu $p + 1$ a iterační funkce $x_{i+1} = g(x_i)$ je řádu p, pak je Steffensenova metoda řádu $2p - 1$ pro $p > 1$. V případě, že $p = 1$ a $g'(\xi) \neq 1$, je metoda minimálně řádu 2 [2].

Výhody a nevýhody

Rychlost konvergence je srovnatelná s Newtonovou metodou, na rozdíl od ní ale nevyžaduje vyhodnocení derivace zkoumané funkce. Z tohoto úhlu pohledu se jedná o značné vylepšení oproti metodě sečen, která taktéž nevyžaduje výpočet derivace, její konvergence je však pouze superlineární. V porovnání s metodou prosté iterace je pak zjevnou výhodou rychlost, více však fakt, že Steffensenova metoda dokáže konvergovat i k původně odpuzujícím pevným bodům.

Samotná konvergence však zaručena není, tato metoda totiž trpí stejnou slabinou jako Newtonova, a to že při nevhodné volbě počáteční aproximace bude výsledná posloupnost s největší pravděpodobností divergovat. Další nevýhodou pak může být samotná cena iterace vyplývající ze složitější iterační funkce.

2.2 Metody pro řešení polynomů

Polynomem rozumíme matematický výraz v podobě součtu *n* členů sestávajících z koeficientu a mocniny neznámé. Obecný tvar polynomu tedy můžeme zapsat takto:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

Než však přejdeme k jednotlivým metodám pro hledání kořenů polynomu, představíme si nejprve techniky pro zjištění jejich počtu a hranic, neboli intervalu, na kterém se nacházejí. Tyto informace nám totiž pomohou lépe odhadnout vstupní aproximace pro níže uvedené metody, a potencionálně tak snížit počet potřebných iterací.

Způsobů, jak určit hranice kořenů je hned několik, liší se pouze velikostí výsledného rozsahu. Jedním ze způsobů je položení následující nerovnosti

$$\frac{1}{1 + \frac{B}{|a_0|}} \leq |\xi_k| \leq 1 + \frac{A}{|a_n|},$$

kde $A = \max(|a_{n-1}|, \ldots, |a_0|)$ a $B = \max(|a_n|, \ldots, |a_1|)$. Dalšími způsoby pak můžeme určit horní hranici:

$$|\xi_k| \leq \max\left\{1, \sum_{j=0}^{n-1} \left|\frac{a_j}{a_n}\right|\right\}$$

$$|\xi_k| \leq 1 + \max\left\{\left|\frac{a_0}{a_n}\right|, \left|\frac{a_1}{a_n}\right|, \ldots, \left|\frac{a_{n-1}}{a_n}\right|\right\}.$$

Důkazy těchto nerovností lze nalézt v [2] a tamtéž odkazovaných zdrojích.

Pro získání počtu kladných i záporných kořenů, reálných i komplexních, můžeme použít *Descartovo pravidlo znamének*. Počet kladných kořenů je roven buď počtu znaménkových změn v koeficientech polynomu $P(x)$, nebo o sudé číslo menší. Počet záporných kořenů se pak zjistí aplikací stejného pravidla na polynom $P(-x)$. Dle definice má polynom tolik kořenů, jakého je řádu. Je-li tedy reálných kořenů méně než stupeň polynomu, je zbytek tvořen komplexními kořeny, které, má-li polynom pouze reálné koeficienty, tvoří páry ($c \pm di$, kde $c, d \in \mathbb{R}$).

2.2.1 Newtonova metoda

Jelikož jsou polynomy specifickým případem nelineární funkce, lze pro hledání jejich kořenů použít libovolnou z výše zmíněných metod, z nichž nejvhodnější je hlavně Newtonova metoda. Při pohledu na její iterační funkci

$$x_{i+1} = x_i - \frac{P(x_i)}{P'(x_i)}$$

si však vzpomeneme, že je třeba vypočítat hodnotu polynomu a také jeho derivace. K tomuto účelu lze efektivně využít Hornerovo schéma.

Mějme polynom $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$. Hodnotu polynomu v libovolném bodě x vypočítáme z koeficientů $a_n \ldots a_0$ následujícím způsobem:

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + b_n x \\ &\vdots \\ b_0 &= a_0 + b_1 x = P(x) \end{aligned}$$

Výsledek b_0 je pak hodnotou polynomu v bodě x. Derivaci v tomto bodě pak získáme opětovnou aplikací tohoto postupu, tentokrát ale na koeficienty b_n až b_1:

$$\begin{aligned} c_n &= b_n \\ c_{n-1} &= b_{n-1} + c_n x \\ &\vdots \\ c_1 &= b_1 + c_2 x = P'(x) \end{aligned}$$

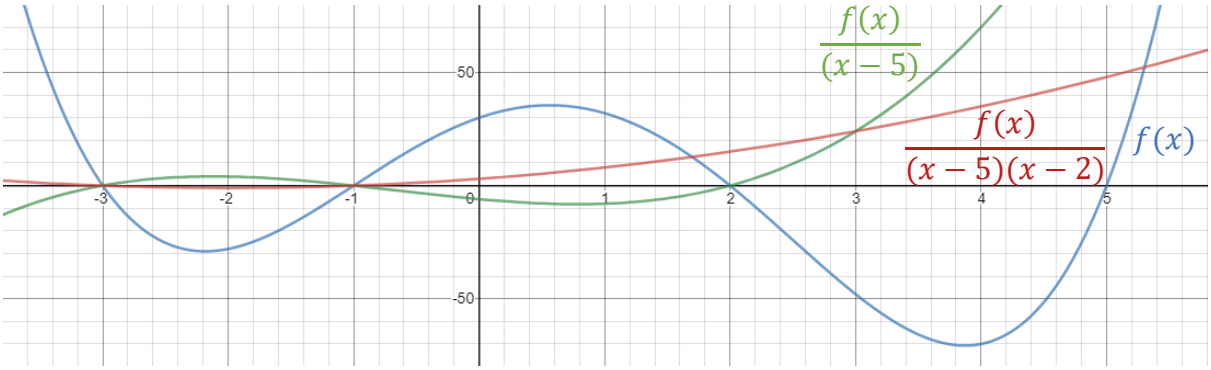
Algoritmus

Jako počáteční aproximaci je vhodné volit takový bod, který je větší než největší kořen zkoumaného polynomu. K určení takové aproximace nám mohou posloužit výše zmíněné techniky určování hranic kořenů. Pomocí Hornerova schémata pak vypočítáme hodnoty polynomu a jeho derivace a aplikujeme Newtonovu metodu pro nalezení prvního kořene ξ_1.

Jelikož je Newtonova metoda použitelná k nalezení pouze jednoho kořene, je třeba doplnit

algoritmus další technikou, například metodou snižování stupně.

Protože polynom lze zapsat také jako $\prod_{n=0}^{p}(x-\xi_n)$, kde ξ_n jsou kořeny, můžeme po nalezení jednoho z kořenů celý polynom vydělit výrazem $(x-\xi)$, kde ξ je právě nalezený kořen. Získáme tak polynom nižšího stupně se stejnými zbývajícími, dosud neobjevenými kořeny. Na něm poté celý proces opakujeme, dokud nenalezneme všechny kořeny.



Obrázek 3: Ukázka výsledných funkcí po dělení polynomu nalezeným kořenem

Výhody a nevýhody

Díky využití Hornerova schématu není třeba v porovnání s klasickou Newtonovou metodou počítat derivaci funkce, což značně přispěje k ceně každé iterace. V praxi však tuto výhodu může snadno převážit nízká rychlost konvergence v případě příliš vzdálené počáteční aproximace, což se může velmi snadno stát, použijeme-li hodnotu z výše zmíněných metod pro nalezení horní hranice. Ty nám sice dávají jistotu, že žádný kořen nebude vyšší, nic však nevíme o tom, jak je ve skutečnosti nejvyšší kořen vzdálený od získané horní hranice. Například pro polynom $x^4-6x^3+3x^2+26x-24$ získáme nejnižší horní hranici 27, ve skutečnosti je ale nejvyšší kořen v bodě 4. Abychom alespoň částečně předešli tomuto zpomalení, můžeme využít tzv. zdvojenou Newtonovu metodu s iterační funkcí $x_{i+1}=x_i-2\frac{P(x_i)}{P\prime(x_i)}$, která s každou iterací přibližuje odhad dvojnásobně. Protože se už ale z geometrického pohledu nejedná o tečnu, může se stát, že aproximace kořen přeskočí, tj. nastane stav $x_{i+1}<\xi$. V takovém případě algoritmus postupuje dále klasickou Newtonovou metodou, protože se dá předpokládat, že už se nacházíme v dostatečné blízkosti kořene.

2.2.2 Newtonova-Maehlyova metoda

Výše popsaná varianta Newtonovy metody bude teoreticky samozřejmě fungovat, v praxi se ale potýká se zásadním nedostatkem způsobeným metodou snižování stupně. Jak bylo řečeno v úvodu práce, tyto metody nemají za cíl nalézt přesné řešení, ale pouze jeho dostatečně blízký odhad, a jelikož by metoda snižování stupně využívala k odvození polynomu nižšího stupně místo přesného kořene pouze jeho aproximaci, mohlo by nám konečné řešení pro několik posledních kořenů vyjít zcela špatně. Tento problém řeší Maehlyova úprava Newtonovy metody, které k fungování stačí pouze aproximace předchozích kořenů.

Mějme polynom nižšího stupně a jeho derivaci vyjádřené následovně:

$$P_{(j)}(x) = \frac{P(x)}{(x-\tilde{\xi}_1)\dots(x-\tilde{\xi}_j)},$$

$$P'_{(j)}(x) = \frac{P'(x)}{(x-\tilde{\xi}_1)\dots(x-\tilde{\xi}_j)} - \frac{P(x)}{(x-\tilde{\xi}_1)\dots(x-\tilde{\xi}_j)}\sum_{n=1}^{j}\frac{1}{x-\tilde{\xi}_n}.$$

Dosazením do Newtonovy metody dostaneme

$$x_{i+1} = x_i - \frac{P_{(j)}(x_i)}{P'_{(j)}(x_i)} = x_i - \frac{\dfrac{P(x_i)}{(x_i-\tilde{\xi}_1)\dots(x_i-\tilde{\xi}_j)}}{\dfrac{P'(x_i)}{(x_i-\tilde{\xi}_1)\dots(x_i-\tilde{\xi}_j)} - \dfrac{P(x_i)}{(x_i-\tilde{\xi}_1)\dots(x_i-\tilde{\xi}_j)}\sum_{n=1}^{j}\dfrac{1}{x_i-\tilde{\xi}_n}},$$

a úpravou získáme

$$x_{i+1} = x_i - \frac{P(x_i)}{P'(x_i) - \sum_{n=1}^{j}\dfrac{P(x_i)}{x_i-\tilde{\xi}_n}}. \tag{2.16}$$

Algoritmus

Iterační funkci (2.16) označme $x_{i+1} = \varphi_j(x_i)$, kde j označuje počet již známých kořenů, nebo jejich aproximací. Znamená to tedy, že takovouto funkci použijeme pro nalezení kořene ξ_{j+1}. Každý takto nalezený kořen je pak využit ve výpočtu pro nalezení kořenů následujících. Jak bylo zmíněno v předchozí kapitole, lze opět použít zdvojenou modifikaci metody

s návratem k původní v momentě přeskočení kořene. Výhodou této modifikace je již zmíněná nezávislost na chybách předchozích aproximací a s využitím zdvojené metody i poměrně rychlá, kvadratická konvergence.

2.2.3 Müllerova metoda

Tato metoda je zobecněním metody sečen. Ačkoliv ji lze aplikovat na kteroukoliv funkci, je dobře použitelná především pro řešení polynomů, proto si ji představíme zde. Zatímco u metody sečen jsme zkoumanou funkci aproximovali přímkou, zde použijeme aproximaci parabolou procházející třemi zadanými body. Průsečík s osou x bude opět další aproximací, která, společně s dvěma předchozími, bude sloužit k sestrojení další paraboly.

Mějme tedy zadány body x_0, x_1 a x_2 jako počáteční aproximace. Pomocí následujících vztahů vypočítáme koeficienty pro rovnici kvadratické funkce:

$$a = \frac{(x_0 - x_2)\big(P(x_1) - P(x_2)\big) - (x_1 - x_2)\big(P(x_0) - P(x_2)\big)}{(x_0 - x_2)(x_1 - x_2)(x_1 - x_0)},$$

$$b = \frac{(x_0 - x_2)^2\big(P(x_1) - P(x_2)\big) - (x_1 - x_2)^2\big(P(x_0) - P(x_2)\big)}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)},$$

$$c = P(x_2)$$

Pomocí těchto koeficientů nyní můžeme sestrojit kvadratickou rovnici

$$Q(x) = a(x - x_2)^2 + b(x - x_2) + c,$$

jejíž řešení, které je na ose x bližší x_2, je další aproximací. Správné řešení této rovnice získáme ze vztahu

$$x_{i+1} = x_i - \frac{2c}{b + sign(b)\sqrt{b^2 - 4ac}}. \tag{2.17}$$

Dle [8] lze ukázat, že řád Müllerovy metody je minimálně 1,84…, tedy větší z kořenů funkce $q^3 - q^2 - q - 1 = 0$.

2.2.4 Laguerrova metoda

Tato metoda je význačná tím, že dokáže při libovolné počáteční aproximaci v drtivé většině případů konvergovat k jednomu z kořenů zkoumaného polynomu. Laguerrova metoda je poměrně oblíbená, a to i přes to, že je výpočetně náročnější, a především vyžaduje znalost stupně polynomu.

Mějme polynom $P(x) = (x - \xi_1)(x - \xi_2) \dots (x - \xi_n)$. Jeho derivaci pak můžeme vyjádřit ve tvaru $P'(x) = (x - \xi_2) \dots (x - \xi_n) + (x - \xi_1)(x - \xi_3) \dots (x - \xi_n) + \dots + \dots (x - \xi_{n-1})$, nebo také $P'(x) = P(x)\left(\frac{1}{x-\xi_1} + \frac{1}{x-\xi_2} + \dots + \frac{1}{x-\xi_n}\right)$.

Původní polynom $P(x)$ nyní zlogaritmujeme a derivujeme, čímž získáme

$$ln|P(x)| = ln|x - \xi_1| + ln|x - \xi_2| + \dots + ln|x - \xi_n|$$

$$[ln|P(x)|]' = \frac{1}{x - \xi_1} + \frac{1}{x - \xi_2} + \dots + \frac{1}{x - \xi_n}. \tag{2.18}$$

Všimněme si, že tvar (2.18) je stejný jako činitel ve výše zmíněné, druhé variantě zápisu derivace polynomu. Tento tvar si tedy pro následné použití označíme

$$G = \frac{P'(x)}{P(x)}.$$

Stejným způsobem získáme i tvar odvozený z negované druhé derivace a označíme

$$H = -[ln|P(x)|]'' = \frac{1}{(x - \xi_1)^2} + \frac{1}{(x - \xi_2)^2} + \dots + \frac{1}{(x - \xi_n)^2} = G^2 - \frac{P''(x)}{P(x)}.$$

Předpokládejme nyní, že vzdálenost prvního kořene ξ_1 od počáteční aproximace x je $a = x - \xi_1$ a že ostatní kořeny jsou sdruženy ve vzdálenosti $b \approx x - \xi_i$. Pak můžeme vyjádřit

$$G = \frac{1}{a} + \frac{n-1}{b}, \qquad H = \frac{1}{a^2} + \frac{n-1}{b^2},$$

kde n značí stupeň polynomu. Po vyřešení soustavy těchto rovnic pro a získáme následující vztah

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}}.$$

Znaménko ve jmenovateli volíme tak, aby absolutní hodnota celého jmenovatele byla co nejvyšší. Další aproximaci pak získáme položením $x_{i+1} = x_i - a$.

Jak již bylo zmíněno, situace, kdy Laguerrova metoda nekonverguje ke kořeni polynomu jsou vzácností, a tak se jedná o vhodnou volbu pro určení přesnější aproximace pro jiné metody. Je však nutné upozornit na fakt, že výraz pod odmocninou může vyjít záporný, metoda totiž může konvergovat i ke komplexním kořenům, nebo během konvergence k reálnému kořeni projít přes komplexní mezivýsledky. Při vlastní implementaci je tak nutné zahrnout i základní komplexní aritmetiku.

3 KAPITOLA

3.1 Příklady a analýza metod

Pro praktické otestování metod a jejich vlastností zvolme následující postup. Nejprve stanovíme několik různých příkladů nelineárních rovnic, na kterých otestujeme každou metodu, porovnáme výsledky a vytvoříme závěr, jak typ funkce, volba metody a počáteční aproximace mohou ovlivňovat průběh výpočtu. Pro každý typ rovnic jsou jednotlivé příklady voleny tak, aby představovaly různé scénáře průběhu funkce (např. prudkost stoupání či klesání, násobnost kořene nebo průběh funkce v okolí kořene). Mějme tedy pro tyto účely následující příklady:

- kvadratické rovnice
  - $f_{kv1}(x) = x^2 - 2x - 2$
  - $f_{kv2}(x) = -x^2 + 3x - 2{,}2$
  - $f_{kv3}(x) = 4x^2 + 4x + 1$
- goniometrické funkce
  - $f_{gon1}(x) = \sin(x)$
  - $f_{gon2}(x) = 2\cos(x) - 2$
  - $f_{gon3}(x) = \sin(x) + 0{,}5x - 2$
- exponenciální funkce
  - $f_{exp1}(x) = 8^{x-2} - x$
  - $f_{exp2}(x) = e^{-x} - 1$
  - $f_{exp3}(x) = e^{-x^2} - 0{,}5$
- logaritmické funkce
  - $f_{log1}(x) = \ln(x)$
  - $f_{log2}(x) = \log_2(x^2 + 6x) - 5$
  - $f_{log3}(x) = \ln\left(\frac{(2x+1)^3}{(3x-1)^4}\right)$
- polynomy
  - $f_{pol1}(x) = -5x^4 - 3x^3 + 3x^2 + x$
  - $f_{pol2}(x) = x^3 + x^2 - x - 1$
  - $f_{pol3}(x) = x^6 - 2x^5 - 3x^4 + 4x^3 - x^2 + 4x - 2$

V následujících podkapitolách jsou uvedeny výsledky řešení těchto příkladů všemi metodami, které bylo možné aplikovat. Všechny příklady byly řešeny na požadovanou přesnost $\varepsilon = 10^{-6}$ a maximální počet 1000 iterací. Jako zastavovací kritérium bylo požadováno splnění obou podmínek, tj. $f(x_i) < \varepsilon$ a $|x_{i+1} - x_i| < \varepsilon$, aby mohlo být poukázáno, jak nevhodná volba kritéria může negativně ovlivnit přesnost nebo délku výpočtu. Poslední sloupec, podle kterého je tabulka řazena, je celková efektivita metody definovaná jako součin $(f(\xi) + 1) * iterace * (čas + 1)$. Řád a rychlost konvergence do výpočtu zahrnuty nejsou, jelikož přesnost a samotná existence těchto údajů je závislá na počtu iterací. U rychle konvergujících metod by tak tyto údaje vůbec nemusely být k dispozici, nebo výsledek značně zkreslovat. Mějme však na paměti, že tento ukazatel je pouze orientační a skutečná efektivita metody v praxi může záviset na mnoha dalších faktorech (výkon testovacího stroje, doplňující činnosti mezi iteracemi, rozlišovací schopnosti při měření času apod.).

3.1.1 Kvadratické rovnice

- $f(x) = x^2 - 2x - 2$
- $g_1(x) = f(x) + x, \quad g_2(x) = x - \frac{f(x)}{4}, \quad g_3(x) = \sqrt{2x + 2}, \quad g_4(x) = \frac{x^2 - 2}{2}$

Metoda	*ξ*	*f(ξ)*	*Řád m.*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Laguerrova m.	2,732051	4,44E-16	NaN	NaN	2	0	2
Steffensenova m., $g_3(x)$	2,732051	4,33E-08	NaN	NaN	3	0	3
Newtonova m.	2,732051	0	1,998055	0,283489	4	0	4
Maehlyova m., 1. kořen	2,732051	0	1,998055	0,283489	4	0	4
Newtonova m. pro pol., 1. kořen	2,732051	0	1,998055	0,283489	4	0	4
M. sečen	2,732051	1,17E-11	1,600016	0,364673	5	0	5
Steffensenova m., $g_4(x)$	2,732051	1,27E-10	1,628899	0,071969	5	0	5
Steffensenova m., $g_1(x)$	2,732051	6,44E-09	1,998552	1,269620	5	0	5
Steffensenova m., $g_2(x)$	2,732051	2,73E-08	0,210427	2,58E-07	3	1	6
M. regula falsi	2,732051	3,16E-08	1,000000	0,071797	7	1	14
M. prosté it., $g_2(x)$	2,732051	3,46E-07	0,999996	0,133966	7	1	14
M. bisekce	2,732051	3,05E-07	1	0,5	21	2	63,00002
M. prosté it., $g_3(x)$	2,732051	7,00E-07	1,000000	0,366025	14	6	98,00007

Tabulka 1: Výsledky pokusů pro funkci $f_{kv1}(x)$

- $f(x) = -x^2 + 3x - 2{,}2$
- $g_1(x) = f(x) + x, g_2(x) = x - f(x), g_3(x) = \sqrt{3x - 2{,}2}, g_4(x) = \frac{x^2+2{,}2}{3}$,
  $g_5(x) = -\frac{2{,}2}{x} + 3$

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***Iterace***	***Čas (ms)***	***Efektivita***
Laguerrova m.	1,7236068	0	NaN	NaN	1	0	1
Steffensenova m., $g_1(x)$	1,7236068	2,42E-10	1,025715	0,001117	4	0	4
Steffensenova m., $g_4(x)$	1,7236068	2,17E-09	0,741488	1,84E-04	5	0	5
Newtonova m. pro pol., 1. kořen	1,7236068	0	1,989234	2,038598	5	0	5
Newtonova m.	1,7236068	0	1,982136	1,955062	5	1	10
Maehlyova m., 1. kořen	1,7236068	0	1,989234	2,038598	5	1	10
Steffensenova m., $g_2(x)$	1,7236068	2,49E-10	1,856245	1,096515	6	1	12
M. sečen	1,7236068	0	0,853778	3,38E-04	8	2	24
M. prosté it., $g_1(x)$	1,7236075	3,28E-07	3,132037	2,80E+13	20	1	40,000013
Steffensenova m., $g_3(x)$	1,7236068	6,26E-12	0,395645	6,15E-06	17	2	51
Steffensenova m., $g_5(x)$	1,7236068	4,10E-11	0,924799	4,45E-04	23	2	69
M. regula falsi	1,7236066	1,07E-07	1,000001	0,381974	15	4	75,000008
M. bisekce	1,7236066	9,45E-08	1	0,5	20	3	80,000008
M. prosté it., $g_5(x)$	1,7236087	8,43E-07	1,488096	291,22382	38	2	114,00010
M. prosté it., $g_3(x)$	1,7236088	9,15E-07	1,897403	165981,44	82	5	492,00045

Tabulka 2: Výsledky pokusů pro funkci $f_{kv2}(x)$

- $f(x) = 4x^2 + 4x + 1$
- $g_1(x) = f(x) + x, \quad g_2(x) = x - \frac{f(x)}{4}, \quad g_3(x) = \frac{-4x-1}{4x}$

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***Iterace***	***Čas (ms)***	***Efektivita***
Laguerrova m.	-0,5	0	NaN	NaN	1	0	1
Steffensenova m., $g_2(x)$	-0,5	0	1,070184	0,887584	13	1	26
Steffensenova m., $g_1(x)$	-0,5	0	1,116172	1,513044	18	1	36
Maehlyova m., 1. kořen	-0,499999	3,64E-12	1,296611	14,693790	19	1	38
Newtonova m. pro pol., 1. kořen	-0,499999	3,64E-12	1,296611	14,693790	19	1	38
Newtonova m.	-0,499999	3,64E-12	1,000008	0,500050	19	2	57
M. sečen	-0,499999	4,18E-12	1,000089	0,618755	28	5	168
M. prosté it., $g_2(x)$	NaN	NaN	0,999006	0,992165	1000	63	NaN

Tabulka 3: Výsledky pokusů pro funkci $f_{kv3}(x)$

Do výsledků nebyly zahrnuty metody, které divergovaly (většinou met. Prosté iterace vlivem špatné volby iterační funkce), Müllerova metoda, která vzhledem k principu svého fungování

v případě kvadratických funkcí nalezne řešení hned v první iteraci a Maehlyova s Newtonovou metodou při hledání druhého kořene, protože v dané situaci řešily téměř lineární funkci, řešení tak našly shodně v druhé iteraci. V posledním příkladě pak nejsou zahrnuty intervalové metody bisekce a regula falsi, protože se jedná o funkci s jediným kořenem se sudou násobností.

Ve všech třech případech tabulkám dle výše definované efektivity podle očekávání dominují pokročilejší funkce určené především pro řešení polynomů a Steffensenova metoda využívající urychlení konvergence pomocí Aitkenovy δ^2-metody. Ze základních metod se pak nejvíce osvědčila Newtonova metoda následována metodou sečen a intervalovými metodami regula falsi a bisekce, čímž se potvrdila teorie o vzájemném poměru rychlostí těchto metod. Metodu prosté iterace je těžké přímo hodnotit, jelikož její výsledky závisí na uživateli a jeho volbě iterační funkce, její efektivita tak může značně kolísat. Příkladem konvergující, ale extrémně neefektivní iterační funkce je funkce $g_2(x)$ ve třetím příkladě. Z tabulky můžeme vidět, že po tisíci iteracích metoda stále nenalezla dostatečně přesný výsledek, přesto k němu velice pomalu konvergovala.

Třetí příklad přinesl také potvrzení teorie týkající se volby ukončovacího kritéria v závislosti na podmíněnosti funkce. Vzhledem k sudé násobnosti kořene je funkce v jeho okolí velmi špatně podmíněna, a ukončovací kritérium $f(x_i) < \varepsilon$ bylo splněno v některých případech již po polovičním počtu provedených iterací, skutečná odchylka od přesného řešení však byla vzhledem ke zvolené přesnosti stále příliš velká (téměř 0,0005).

3.1.2 Goniometrické funkce

- $f(x) = \sin(x)$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x), \;\; g_3(x) = xsin(x) + x$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Newtonova m.	3,141593	0	1,932675	0,002258	4	0	4
M. prosté it., $g_1(x)$	3,141593	0	2,999925	0,166552	4	0	4
M. regula falsi	3,141593	0	1,507635	0,004792	5	0	5
M. sečen	3,141593	0	1,507635	0,004792	5	0	5
Müllerova m.	3,141593	0	2,307120	70,596490	5	0	5
Steffensenova m., $g_2(x)$	0	0	5,041681	0,033234	5	0	5

Steffensenova m., $g_3(x)$	3,141593	3,00E-09	1,997104	0,663194	6	0	6
M. prosté it., $g_2(x)$	6,283185	0	2,995501	0,163892	5	1	10
M. bisekce	3,141593	1,51E-07	1	0,5	22	0	22,000003
Steffensenova m., $g_1(x)$	3,141593	1,24E-12	NaN	NaN	33	0	33

Tabulka 4: Výsledky pokusů pro funkci $f_{gon1}(x)$

- $f(x) = 2\cos(x) - 2$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x), \;\; g_3(x) = xcos(x)$

Metoda	***ξ***	***f(ξ)***	***Řád metody***	***Rychlost konv.***	***Iterace***	***Čas (ms)***	***Efektivita***
Steffensenova m., $g_1(x)$	4,32E-06	1,86E-11	NaN	NaN	16	0	16
Steffensenova m., $g_2(x)$	4,04E-06	1,63E-11	1,358714	19,361398	20	0	20
Newtonova m.	8,44E-07	7,13E-13	1,145108	1,539071	20	1	40
Steffensenova m., $g_3(x)$	1,22E-04	1,49E-08	NaN	NaN	22	2	66
M. sečen	1,28E-06	1,63E-12	0,551048	0,001181	28	2	84
M. prosté it., $g_1(x)$	9,98E-04	9,96E-07	1,707759	5398,071844	986	11	11832,0118
M. prosté it., $g_2(x)$	NaN	NaN	0,170262	9,205843	1000	10	NaN
M. prosté it., $g_3(x)$	NaN	NaN	0,999021	0,996078	1000	6	NaN

Tabulka 5: Výsledky pokusů pro funkci $f_{gon2}(x)$

- $f(x) = \sin(x) + 0{,}5x - 2$
- $g_1(x) = f(x) + x, g_2(x) = x - f(x), g_3(x) = -2\sin(x) + 4, g_4(x) = \frac{xsin(x)+0{,}5x^2}{2}$

Metoda	***ξ***	***f(ξ)***	***Řád metody***	***Rychlost konv.***	***Iterace***	***Čas (ms)***	***Efektivita***
Newtonova m.	5,462807	0	1,987732	0,282052	4	0	4
Steffensenova m., $g_1(x)$	5,462807	1,03E-09	1,074062	0,001609	5	0	5
Steffensenova m., $g_3(x)$	5,462807	2,62E-09	1,997863	0,413947	5	0	5
Steffensenova m., $g_4(x)$	5,462807	2,29E-09	0,539752	1,06E-05	5	0	5
Müllerova m.	5,462807	0	1,604611	0,031906	6	0	6
M. sečen	5,462807	0	1,688041	1,090172	7	0	7
M. regula falsi	5,462807	2,87E-08	1,000000	0,118866	9	0	9
M. prosté it., $g_2(x)$	5,462807	1,65E-07	0,999994	0,181930	9	0	9,000001
Steffensenova m., $g_2(x)$	5,462807	2,14E-11	1,895651	0,036794	30	0	30
M. bisekce	5,462807	1,56E-07	1	0,5	22	1	44,000007

Tabulka 6: Výsledky pokusů pro funkci $f_{gon3}(x)$

Stejně jako v případě kvadratických rovnic i v těchto tabulkách nejsou zahrnuty metody, které během výpočtu buď divergovaly, nebo se, v případě metod prosté iterace, zacyklily.

Pochopitelně zde již nenajdeme ani metody pro řešení polynomů, k těm se však ještě vrátíme.

Až na několik málo výjimek výsledky opět odpovídají teorii. Pojďme se ale spíše zaměřit na zvláštní situace, které jsou povětšinou způsobeny nevhodnou kombinací iteračních funkcí $g(x)$ a počátečních aproximací. Volbou aproximací pro první příklad v okolí bodu π jsme chtěli docílit konvergence k tomuto konkrétnímu kořeni, nicméně s iterační funkcí $g_2(x)$ nakonec konvergovala metoda prosté iterace i Steffensenova k okolním kořenům 0 a 2π.

Při vzpomínce na odvození iterační funkce Steffensenovy metody nebo při pohledu na dosavadní výsledky jsme si mohli všimnout, že pokud pro nějakou funkci $g(x)$ konvergují obě metody, je z nich vždy rychlejší právě Steffensenova. Tomu ale neodpovídají výsledky prvního a třetího příkladu. Za příčinu této skutečnosti lze označit nevhodnou volbu zastavovacího kritéria. Podmínku $f(x_i) < \varepsilon$ totiž metoda splňuje hned v druhé iteraci, avšak nesplňuje zároveň kritérium maximální velikosti rozdílu mezi výsledky iterací. Algoritmus tak produkuje další výpočty, které generují nové aproximace střídavě v bodě dostatečně přesného řešení a okolí tohoto bodu, které je větší než požadovaná přesnost. Toto okolí se pak postupně zužuje, dokud není dostatečně malé, aby výsledky dvou po sobě jdoucích iterací byly menší než zadaná tolerance. Ve skutečnosti tedy navzdory výsledkům praxe stále odpovídá teorii, a tento jev byl způsoben nevhodným, avšak úmyslným chováním testovacího nástroje.

Stejně jako u kvadratických rovnic i zde se setkáváme s extrémně pomalou konvergencí u druhého příkladu, který obsahuje pouze kořeny se sudou násobností. Ta je způsobena tím, že v okolí pevného bodu je derivace iteračních funkcí téměř rovna jedné, což má za následek pouze malé krůčky mezi iteracemi.

Zacyklením nebo divergencí pak skončily funkce $g_3(x)$ u metody prosté iterace v prvním i třetím příkladě, a funkce $g_2(x)$ u druhého příkladu pomalu konvergovala k mnohem vzdálenějšímu kořeni.

Z těchto pozorování lze vyvodit závěr, že k řešení goniometrických funkcí není příliš vhodné používat metody založené na hledání pevného bodu, obzvláště pak metodu prosté iterace. Jejich periodický průběh totiž může relativně snadno způsobit nežádoucí efekty, jako je zacyklení nebo konvergence k jinému kořeni. Newtonovu metodu a metodu sečen lze považovat za mnohem spolehlivější, avšak pouze při velmi dobré počáteční aproximaci. Navzdory slabším

výsledkům tak z tohoto experimentu vychází jako ideální intervalové metody, protože vylučují jakoukoliv divergenci. Jejich slabší výkon pak lze kompenzovat předčasným zastavením při získání dostatečně přesné aproximace např. pro Newtonovu metodu. V případě počítačového zpracování pak samozřejmě lze výhodně použít i početně náročnější Müllerovu metodu.

3.1.3 Exponenciální funkce

- $f(x) = 8^{x-2} - x$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - \frac{f(x)}{3}$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Newtonova m.	2,426241	0	1,998825	1,281019	6	1	12
Müllerova m.	2,426241	0	1,715234	0,262509	6	1	12
Steffensenova m., $g_2(x)$	0,016159	6,80E-12	0,973439	6,74E-04	7	1	14
M. sečen	2,426241	3,51E-12	1,618575	1,183656	8	1	16
M. prosté it., $g_2(x)$	2,426241	4,76E-07	0,999994	0,348377	16	2	48,00002
M. bisekce	2,426241	8,63E-08	1	0,5	21	4	105,00001
M. regula falsi	2,426241	7,38E-07	1	0,535804	24	10	264,00019
Steffensenova m., $g_1(x)$	2,426241	4,65E-11	1,900976	3,003140	429	60	26169

Tabulka 7: Výsledky pokusů pro funkci $f_{exp1}(x)$

- $f(x) = e^{-x} - 1$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x), \;\; g_3(x) = xe^{-x}$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Newtonova m.	3,58E-16	0	1,999722	0,498577	7	0	7
M. sečen	-7,66E-12	7,66E-12	1,619031	0,660185	8	0	8
M. prosté it., $g_1(x)$	0	0	2,017140	0,576717	10	0	10
Müllerova m.	1,07E-12	1,07E-12	1,741436	0,144430	6	1	12
M. bisekce	1,19E-07	1,19E-07	1	0,5	23	0	23,000003
Steffensenova m., $g_2(x)$	4,12E-18	0	1,999889	0,998681	32	0	32
M. regula falsi	4,37E-07	4,37E-07	1	0,418024	18	1	36,000016
Steffensenova m., $g_3(x)$	4,12E-18	NaN	1,000096	0,999932	1000	10	NaN

Tabulka 8: Výsledky pokusů pro funkci $f_{exp2}(x)$

- $f(x) = e^{-x^2} - 0{,}5$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x)$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Steffensenova m., $g_1(x)$	0,832555	4,16E-10	0,615363	7,43E-06	3	0	3
Newtonova m.	0,832555	0	1,989290	0,207582	4	0	4
Steffensenova m., $g_2(x)$	0,832555	2,97E-09	0,200107	1,21E-07	4	0	4
M. regula falsi	0,832555	1,57E-09	1,000014	0,009017	6	0	6
M. prosté it., $g_1(x)$	0,832555	1,14E-07	1,000003	0,167452	8	0	8,000001
Müllerova m.	0,832555	0	1,668613	0,087948	5	1	10
M. sečen	0,832555	1,28E-11	1,623591	0,434524	6	1	12
M. bisekce	0,832555	1,72E-07	1	0,5	22	0	22,000004

Tabulka 9: Výsledky pokusů pro funkci $f_{exp3}(x)$

Tato sekce nám bohužel nepřinesla žádné nové poznatky, avšak dalším výskytem určitých jevů nám potvrdila jejich platnost. Z teoretické části víme, že Steffensenova metoda dokáže konvergovat k odpuzujícím pevným bodům, ale ačkoliv je tato metoda význačná rychlou konvergencí, u takovýchto bodů tomu může být přesně naopak. Dokonalou ukázkou je první příklad s iterační funkcí $g_1(x)$ a druhý příklad s funkcí $g_3(x)$. která v bodě počáteční aproximace prudce stoupá, a další body využívané ve výpočtu Steffensenovy metody (obzvláště třetí $g\big(g(x_i)\big)$) tak leží neúměrně daleko. Tento velký rozdíl pak má za následek, že během výpočtu další iterace se hodnota původního odhadu násobí číslem jen velmi nepatrně nižším než 1, další iterace tak vyjde jen těsně vedle předchozího odhadu.

Další charakteristikou Steffensenovy metody je určitá míra nepředvídatelnosti. V prvním příkladě jsme volili jako počáteční aproximaci bod $x_0 = 3$, kterému je nejblíže ostatními metodami nalezený kořen $\xi \approx 2{,}426241$. S funkcí $g_2(x)$ však metoda konvergovala ke vzdálenějšímu kořeni $\xi \approx 0{,}016259$.

3.1.4 Logaritmické funkce

- $f(x) = \ln(x)$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x)$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Steffensenova m., $g_2(x)$	1	0	NaN	NaN	3	0	3
Newtonova m.	1	0	2,000798	0,503845	5	0	5
Müllerova m.	1	1,38E-14	1,710386	0,094960	5	0	5
M. prosté it., $g_2(x)$	1	0	1,997738	0,489983	5	1	10
M. regula falsi	1,0000002	1,88E-07	1,000000	0,278652	12	0	12,000002
M. sečen	1	0	1,636228	0,877883	7	1	14
M. bisekce	0,9999999	1,19E-07	1	0,5	22	0	22,000003

Tabulka 10: Výsledky pokusů pro funkci $f_{log1}(x)$

- $f(x) = \log_2(x^2 + 6x) - 5$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x)$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Müllerova m.	3,403124	0	1,792466	0,154097	5	0	5
M. sečen	3,403124	6,94E-13	1,619339	0,277579	5	0	5
Newtonova m.	3,403124	0	1,997307	0,120196	6	0	6
Steffensenova m., $g_2(x)$	3,403124	2,92E-09	1,998864	0,051076	4	1	8
M. regula falsi	3,403124	1,06E-07	0,9999999	0,189892	9	0	9,000001
M. prosté it., $g_2(x)$	3,403124	2,80E-07	0,9999997	0,422638	16	0	16,000004
M. bisekce	3,403124	5,48E-08	1	0,5	22	0	22,000001

Tabulka 11: Výsledky pokusů pro funkci $f_{log2}(x)$

- $f(x) = \ln\left(\frac{(2x+1)^3}{(3x-1)^4}\right)$
- $g_1(x) = f(x) + x, \;\; g_2(x) = x - f(x)$

Metoda	*ξ*	*f(ξ)*	*Řád metody*	*Rychlost konv.*	*Iterace*	*Čas (ms)*	*Efektivita*
Newtonova m.	1,149161	1,91E-14	2,001105	0,802707	6	0	6
M. sečen	1,149161	6,79E-14	1,685055	2,161403	6	0	6
Müllerova m.	1,149161	1,52E-13	1,656118	0,084902	6	1	12
M. bisekce	1,149161	4,48E-07	1	0,5	22	0	22,000010
Steffensenova m., $g_1(x)$	1,149161	7,49E-11	2,011378	1,781612	10	2	30
M. regula falsi	1,149162	7,88E-07	1,000000	0,587401	28	1	56,000044
M. prosté it., $g_1(x)$	NaN	NaN	2,374131	14,487903	1000	20	NaN

Tabulka 12: Výsledky pokusů pro funkci $f_{log3}(x)$

Po provedení experimentů s uvedenými logaritmickými funkcemi můžeme dojít k závěru, že samotná volba dobré počáteční aproximace je velmi důležitá součást řešení. Mnoho

popsaných metod totiž svým principem generují následující odhady v okolí kořene, přičemž není nijak definováno, jak velké toto okolí bude, či na jaké straně od kořene se další aproximace objeví. V případě logaritmických funkcí je pak tato skutečnost rizikovým faktorem, protože se kořen těchto funkcí obvykle nachází těsně vedle krajního bodu definičního oboru. Existuje tak celá řada počátečních aproximací blízkých kořeni, které svůj výpočet ukončí pádem mimo definiční obor funkce, a tímto způsobem mohou selhat i do této chvíle spolehlivé metody jako je Newtonova či metoda sečen. Pokud tedy řešitel nezná princip fungování těchto metod nebo nemá určitou představu o průběhu funkce a jejím definičním oboru, nabízí se jako nejideálnější volba pro tento typ rovnic intervalové metody. V opačném případě podávaly nejlepší výsledky metody Müllerova, Newtonova a z ní odvozená metoda sečen.

3.1.5 Polynomy

Při řešení polynomů můžeme naplno využít popsané metody pro hledání všech kořenů. Abychom ale udrželi výsledkové tabulky přehledné, budeme pro každý z příkladů evidovat dvě tabulky, kde v první budou prezentovány výsledky metod pro nalezení jednoho kořene a v druhé pak pouze výsledky modifikované Newtonovy a Maehlyovy metody pro přímé porovnání i u ostatních kořenů.

- $f(x) = -5x^4 - 3x^3 + 3x^2 + x$
- $g_1(x) = f(x) + x, g_2(x) = x - f(x), g_3(x) = \sqrt[4]{\frac{-3x^3+3x^2+x}{5}}, g_4(x) = \sqrt[3]{\frac{-5x^4+3x^2+x}{3}}$, $g_5(x) = \sqrt{\frac{5x^4+3x^3-x}{3}}$

Metoda	*ξ*	*f(ξ)*	*Řád m.*	*Rychlost k.*	*Iterace*	*Čas (ms)*	*Efektivita*
Steffensenova m., $g_3(x)$	0,689898	2,77E-10	1,258698	0,001321	4	0	4
Steffensenova m., $g_5(x)$	0,689898	6,30E-11	1,530276	0,044775	5	0	5
Laguerrova m.	0,689898	3,06E-16	NaN	NaN	3	1	6
M. prosté it., $g_3(x)$	0,689898	6,51E-07	1,000012	0,130328	7	0	7,000005
M. sečen	0,689898	1,57E-09	1,622293	2,102623	8	0	8
Steffensenova m., $g_4(x)$	-0,289898	1,39E-11	2,002429	0,258544	5	1	10
Newtonova m.	0,689898	0	1,997474	2,980762	6	1	12
Maehlyova m., 1. kořen	0,689898	0	1,997474	2,980762	6	1	12
Newtonova m. pro pol., 1. kořen	0,689898	0	1,997474	2,980762	6	1	12
M. prosté it., $g_4(x)$	-0,289898	1,95E-07	1,000006	0,333363	12	1	24,000005
M. bisekce	0,689898	3,74E-07	1	0,5	20	1	40,000015

Steffensenova m., $g_2(x)$	0,689898	8,59E-11	0,913980	2,20E-04	33	1	66
M. regula falsi	0,689898	8,17E-07	1	0,557215	25	2	75,000061

Tabulka 13: Výsledky pokusů pro funkci $f_{pol1}(x)$ při hledání 1. kořene

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***Iterace***	***Čas (ms)***	***Efektivita***
Maehlyova m., 4. kořen	-1	0	0	0	2	0	2
Newtonova m. pro pol., 4. kořen	-1	0	0	0	2	1	4
Maehlyova m., 1. kořen	0,689898	0	1,997474	2,980762	6	1	12
Newtonova m. pro pol., 1. kořen	0,689898	0	1,997474	2,980762	6	1	12
Maehlyova m., 3. kořen	-0,289898	0	1,992540	1,319680	5	2	15
Maehlyova m., 2. kořen	-1,17E-19	0	1,998696	4,377783	8	1	16
Newtonova m. pro pol., 2. kořen	2,01E-19	0	1,998654	4,376465	8	1	16
Newtonova m. pro pol., 3. kořen	-0,289898	1,02E-14	1,992538	1,319671	5	4	25

Tabulka 14: Výsledky pokusů pro funkci $f_{pol1}(x)$ při hledání všech kořenů

- $f(x) = x^3 + x^2 - x - 1$
- $g_1(x) = f(x) + x,\ g_2(x) = x - f(x),\ g_3(x) = \sqrt[3]{-x^2 + x + 1},$
  $g_4(x) = \sqrt{-x^3 + x + 1}$

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***Iterace***	***Čas (ms)***	***Efektivita***
Laguerrova m.	-1,0000002	5,04E-14	1,121471	0,963903	11	0	11
Steffensenova m., $g_3(x)$	-1	0	0,630490	0,026616	16	0	16
Steffensenova m., $g_1(x)$	-1	0	0,631508	0,021909	21	1	42
Maehlyova m., 1. kořen	-1,000001	9,52E-13	1,296575	16,170857	21	2	63
Newtonova m. pro pol., 1. kořen	-1,000001	9,52E-13	1,296575	16,170857	21	2	63
Newtonova m.	-1,000001	9,52E-13	1,100863	1,717891	21	3	84
M. sečen	-0,999999	3,04E-12	1,000052	0,618443	27	3	108

Tabulka 15: Výsledky pokusů pro funkci $f_{pol2}(x)$ při hledání násobného kořene

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***Iterace***	***Čas (ms)***	***Efektivita***
Maehlyova m., 3. kořen	1	0	0	0	3	0	3
Newtonova m. pro pol., 3. kořen	1	0	0	0	3	0	3
Maehlyova m., 2. kořen	-0,9999998	1,19E-13	0,568082	2,85E-04	4	0	4
Newtonova m. pro pol., 2. kořen	-0,9999996	2,40E-07	0,625869	3,87E-04	4	0	4,000001
Maehlyova m., 1. kořen	-1,000001	9,52E-13	1,296575	16,170857	21	2	63
Newtonova m. pro pol., 1. kořen	-1,000001	9,52E-13	1,296575	16,170857	21	2	63

Tabulka 16: Výsledky pokusů pro funkci $f_{pol2}(x)$ při hledání všech kořenů

- $f(x) = x^6 - 2x^5 - 3x^4 + 4x^3 - x^2 + 4x - 2$
- $g_1(x) = f(x) + x,\ g_2(x) = x - f(x),\ g_3(x) = \sqrt[6]{2x^5 + 3x^4 - 4x^3 + x^2 - 4x + 2},$

$$g_4(x) = \sqrt[5]{\frac{x^6-3x^4+4x^3-x^2+4x-2}{2}},\ g_5(x) = \sqrt[4]{\frac{x^6-2x^5+4x^3-x^2+4x-2}{3}},$$
$$g_6(x) = \sqrt[3]{\frac{-x^6+2x^5+3x^4+x^2-4x+2}{4}},\quad g_7(x) = \sqrt{x^6-2x^5-3x^4+4x^3+4x-2},$$
$$g_8(x) = \frac{-x^6+2x^5+3x^4-4x^3+x^2+2}{4}$$

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***It.***	***Čas (ms)***	***Efektivita***
Steffensenova m., $g_6(x)$	0,496158	6,69E-12	1,570318	0,025785	4	0	4
Laguerrova m.	2,545282	1,73E-14	1,121471	0,963903	3	1	6
Steffensenova m., $g_4(x)$	2,545282	5,74E-07	0,503184	1,94E-05	8	0	8,000005
Müllerova m.	2,545282	5,11E-14	2,577445	1795,841336	5	1	10
Steffensenova m., $g_5(x)$	2,545282	5,20E-07	1,996551	0,945794	5	1	10,000005
Steffensenova m., $g_3(x)$	2,545282	3,63E-07	1,991017	0,247416	7	1	14,000005
Maehlyova m., 1. kořen	2,545282	5,11E-14	1,996553	2,057211	6	2	18
Newtonova m. pro pol., 1. kořen	2,545282	5,11E-14	1,996553	2,057211	6	2	18
Steffensenova m., $g_7(x)$	2,545282	3,87E-08	2,008600	49,021695	12	1	24,000001
M. prosté it., $g_6(x)$	0,496158	4,16E-07	1,000006	0,377950	20	1	40,000017
M. sečen	2,545282	6,34E-10	1,617527	1,582951	19	2	57,000000
Newtonova m.	2,545282	5,11E-14	1,996553	2,057211	6	11	72
M. bisekce	2,545282	1,19E-07	1	0,5	26	2	78,000009
M. regula falsi	2,545282	5,98E-07	1,0000001	0,582694	36	6	252,000151
M. prosté it., $g_3(x)$	2,545282	8,50E-07	0,9999996	0,843928	104	4	520,000442

Tabulka 17: Výsledky pokusů pro funkci $f_{pol3}(x)$ při hledání násobného kořene

Metoda	***ξ***	***f(ξ)***	***Řád m.***	***Rychlost k.***	***Iterace***	***Čas (ms)***	***Efektivita***
Maehlyova m., 1. kořen	2,545282	5,11E-14	1,996553	2,057211	6	2	18
Newtonova m. pro pol., 1. kořen	2,545282	5,11E-14	1,996553	2,057211	6	2	18
Maehlyova m., 3. kořen	0,496158	0	1,999491	1,634634	7	2	21
Maehlyova m., 2. kořen	1,207775	0	1,998726	2,739652	9	2	27
Newtonova m. pro pol., 3. kořen	0,496158	0	1,999460	1,634297	7	3	28
Newtonova m. pro pol., 2. kořen	1,207775	0	1,998720	2,739514	9	4	45
Maehlyova m., 4. kořen	-1,751045	7,59E-14	1,996089	0,986311	27	11	324
Newtonova m. pro pol., 4. kořen	-1,751045	0	1,997735	0,998682	26	14	390

Tabulka 18: Výsledky pokusů pro funkci $f_{pol3}(x)$ při hledání všech kořenů

Jako ve všech pokusech, mezi divergující metody patří zejména různé varianty Steffensenovy metody a metody prosté iterace. V prvním příkladě navíc divergovala i jinak spolehlivá Müllerova metoda z důvodu špatně zvolené počáteční aproximace. To bychom mohli říci i u druhého příkladu, pojďme si ale upřesnit, co je na daném výběru konkrétně špatně.

Müllerova metoda spočívá ve zkonstruování paraboly probíhající zadanými body. Násobné kořeny polynomů mívají obvykle tu vlastnost, že jejich křivka bývá lehce ostřejší než u kvadratické funkce. To má za následek, že při volbě bodů na intervalu, kde druhá derivace funkce nemění znaménko, vrchol křivky zkonstruované Müllerovou metodou nedosáhne až k vrcholu křivky polynomu, a tím pádem ani neprotne osu x. Při řešení takových funkcí touto metodou je tedy vhodné zvolit alespoň jeden počáteční bod mimo tento interval.

U druhého příkladu jsme se v případě metod pro nalezení jednoho kořene soustředili na násobný kořen v bodě $\xi = -1$. Stejně jako v případě obdobného druhého příkladu v kvadratických rovnicích se potvrdila teorie o rozdílném chování některých metod při vyšší násobnosti kořene, a to konkrétně pokles řádu konvergence Newtonovy metody a metody sečen z kvadratické, resp. superlineární, na lineární a výrazné zpomalení konvergence u funkce $g_3(x)$ metody prosté iterace. Zajímavý výsledek pak přinesla funkce $g_2(x)$ u obou metod, které shodou okolností našly přesné řešení hned v první iteraci – avšak jiného kořene, než jaký jsme hledat chtěli, proto je v tabulce nenalezneme. Z tabulky byly také odstraněny výsledky Newtonovy a Maehlyovy metody při druhém hledání „stejného“ kořene.

Nyní, když máme výsledky kvadratických rovnic i polynomů, můžeme vidět, že velmi stabilní i rychlou metodou je Laguerrova metoda. Tyto experimenty tak dávají za pravdu v praxi získaným poznatkům, že se jedná o metodu velmi spolehlivou, a i přes náročnější výpočet uspokojivě rychlou. Schopnost nalézt komplexní kořeny je pak už jen přidanou hodnotou.

Na závěr ještě porovnejme metody, jimž byly vyhrazeny tabulky zvlášť. Navzdory očekávání se jejich výsledky příliš neliší. To je pochopitelné především při hledání prvního kořene, kdy obě metody postupují dle stejného algoritmu. Překvapivě se ale metody shodují i u posledního kořene, kde by teoretická chyba Newtonovy metody měla být nejznatelnější. Drobné odchylky ale můžeme nalézt u ostatních kořenů, pro něž přesnější výsledky přinesla Maehlyova metoda, v případě shody pak byla alespoň rychlejší. Potvrzuje se tak teorie říkající, že Maehlyova metoda by měla být všeobecným zdokonalením Newtonovy.

4 KAPITOLA

Součástí praktické části práce je tvorba desktopové aplikace s grafickým rozhraním, která uživateli poskytne v přehledné podobě veškeré informace o výpočtu a jeho průběhu.

Aplikace je vytvořena v programovacím jazyce Java s grafickým rozhraním vytvořeným pomocí nadstavby JavaFX. Důvodem této volby je fakt, že se jedná o celosvětově rozšířenou technologii funkční na mnoha zařízeních, pokud by tedy v budoucnu vznikl požadavek na modifikaci aplikace pro jinou platformu, bude se jednat o minimální množství práce.

Pokročilejší matematické úkony, které standartní knihovny jazyka nepodporují, jako je práce s funkcí zajišťuje externí knihovna mXparser (dostupné z: http://mathparser.org/). Jedná se o obsáhlou knihovnu určenou mimo jiné pro vytváření, úpravu a vyhodnocování funkcí a jejich derivací, či řešení soustav rovnic. Jelikož se jinak jedná o malou aplikaci, nebylo nutné využít žádné další rozšiřující komponenty, a i struktura souborů zdrojových kódů v adresáři projektu je jednoduše organizována do tří částí:

- *src/main*: obsahuje hlavní soubor, ze kterého se aplikace spouští a ve kterém je definováno i celé grafické rozhraní, třídy vykreslovacích pláten dědící z vestavěné třídy *Canvas* a soubor *Controller.java* obsahující hlavní logiku aplikace.

- *src/methods*: obsahuje třídy s kódem řešení zadané funkce jednotlivými iteračními metodami. Jelikož je aplikace navržena pro řešení vždy jen jedné funkce najednou, využívají tyto třídy návrhový vzor *jedináček* a hlavní třídě poskytují pouze referenci na kolekci s daty výsledků. Výjimku tvoří metody generující více sad výsledků, které před zahájením dalšího výpočtu vrátí referenci na nově vytvořenou kopii kolekce. Dále se zde nachází rozhraní deklarující sdílené metody, které tyto třídy implementují a třída *IterationRow.java*, jejíž instance představuje kompletní sadu výsledků v jedné iteraci. Jedná se o objekt, který je vypisován do tabulek během prezentace výsledků.

- *src/source*: zde se nachází jednoduchá třída pro vyvolávání informačních a chybových dialogů a vlastní implementace základních operací nad polynomy a komplexními čísly, které bylo třeba implementovat, jelikož použitá knihovna tyto funkce neposkytovala.

ZÁVĚR

Cílem této práce bylo představit čtenáři několik základních i specifičtějších iteračních metod pro řešení nelineárních rovnic. Po teoretickém výkladu si dále práce kladla za cíl jednotlivé metody porovnat na konkrétních příkladech a uvést doporučení týkající se volby metody podle zkoumané funkce. Z tohoto ohledu přinesly provedené experimenty poměrně přehledné výsledky. V první řadě je však vhodné zmínit, že doporučení se bude lišit podle toho, zda je při řešení k dispozici počítač, nebo jen „tužka a papír“. V prvním případě totiž budeme předpokládat, že použitá metoda již je nějakým způsobem implementována a rozhodujícím faktorem tak bude délka řešení, respektive cena iterací. Naopak pro papírové řešení bude doporučení voleno tak, aby byly jednotlivé iterace co nejméně náročné na vedlejší výpočty se sekundárním zaměřením na jejich počet.

Pro řešení většiny typů rovnic se osvědčila Laguerrova metoda, pro kterou je četnými experimenty dokázáno, že dokáže téměř vždy konvergovat. Cena její iterace je však poměrně vysoká, jelikož vyžaduje vyhodnocení funkce, její první i druhé derivace a poměrně velké množství matematických operací zahrnujících i rozhodování kvůli znaménku. Navíc je tato metoda primárně určená pro řešení polynomů. Z důvodů těchto nedostatků může být mnohdy výhodnější použít Müllerovu metodu, avšak pouze za předpokladu, že se nejedná o kořen se sudou násobností. Chování této metody v takové situaci je blíže popsáno ve třetí kapitole v rámci shrnutí experimentů na polynomech. Jako třetí možnost se pak nabízí Steffensenova metoda, která v experimentech dosahovala výsledků srovnatelných s dříve zmíněnými metodami. Její výsledky jsou však podmíněny volbou vhodné iterační funkce.

Pokud řešitel nemá k dispozici počítač, je třeba využít jednodušší metody, které v experimentech pochopitelně dopadly o něco hůře. Je-li zkoumaná funkce diferencovatelná a výpočet derivace je zvládnutelný, lze velmi efektivně využít Newtonovu metodu, jejíž výsledky byly mnohdy srovnatelné se složitějšími metodami. Nelze-li derivaci funkce použít, nabízí se využít metodu sečen, která s Newtonovou sdílí velké množství vlastností, včetně podobné rychlosti konvergence, která v praxi nebyla o mnoho nižší. Metoda prosté iterace trpí stejným nedostatkem jako její urychlená varianta, a to že v závislosti na volbě iterační funkce může být metoda velmi efektivní, nebo naopak rychle divergovat.

Intervalové metody bisekce a regula falsi v experimentech končily ve spodní polovině tabulky. Tyto metody totiž nevyužívají téměř žádné informace o charakteru zkoumané funkce v daném intervalu, což má za následek pomalou ale za to jistou konvergenci.

Závěrem lze říci, že výsledky experimentů uvedených v této práci se shodují s teorií popsanou v druhé kapitole.

POUŽITÁ LITERATURA

[1] HAVELKOVÁ, Eva. *Metody pro řešení nelineárních rovnic*. Praha, 2015. Bakalářská práce. Univerzita Karlova, Matematicko-fyzikální fakulta.

[2] HOROVÁ, Ivana a Jiří ZELINKA. *Numerické metody*. 2., rozš. vyd. Brno: Masarykova univerzita v Brně, 2004. 294 s. ISBN 80-210-3317-7.

[3] KOPECKY, Karen A. *Root-Finding Methods* [online]. 2007 [cit. 2019-05-02]. Dostupné z: http://www.karenkopecky.net/Teaching/eco613614/Notes_RootFindingMethods.pdf.

[4] LAMBERS, Jim. *Lecture 4 Notes* [online]. 2010 [cit. 2019-05-05]. Dostupné z: https://www.math.usm.edu/lambers/mat772/fall10/lecture4.pdf.

[5] MEKWI, Wankere R. *Iterative Methods for Roots of Polynomials*. Oxford, 2001. Diplomová práce. University of Oxford, Exeter College.

[6] MOŠOVÁ, Vratislava. *Numerické metody.* 1. vyd. Olomouc: Univerzita Palackého, 2003. 147s. ISBN 80-244-0620-9.

[7] RALL, Luis B. Convergence of the newton process to multiple solutions. *Numerische Mathematik*. 1966, 9(1), 23-37.

[8] STOER, Josef a Roland BULIRSCH. *Introduction to numerical analysis*. 2nd ed. New York: Springer, 1996. ISBN 9780387978789.

[9] VITÁSEK, Emil. *Numerické metody*. 1. vyd. Praha: SNTL - Nakladatelství technické literatury, 1987. 512 s. ISBN 04-009-87.

PŘÍLOHY

PŘÍLOHA A – UKÁZKA KÓDU – VÝPOČET ITERACE

Metoda bisekce

```
while ((b - a) > tol && f(m) > tol) {
	m = a + ((b - a) / 2);
	if (Math.signum(f(a)) != Math.signum(fm)) {
		b = m;
	} else {
		a = m;
	}
}
```

Metoda Regula falsi

```
while (Math.abs(m - mPredchozi) > tol && f(m) > tol) {
	m = b - ((b - a) * f(b)) / (f(b) - f(a));
	if (Math.signum(f(a)) != Math.signum(fm)) {
		b = m;
	} else {
		a = m;
	}
}
```

Metoda prosté iterace

```
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
	x = g(x);
}
```

Newtonova metoda

```
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
	x = x - (f(x) / df(x));
}
```

Metoda sečen

```
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
	x = x1 - (f(x1) * (x1 - x0) / (f(x1) - f(x0)));
	x0 = x1;
	x1 = x;
}
```

Steffensenova metoda

```
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
	x = (x * g(g(x)) - Math.pow(g(x), 2)) / (g(g(x)) - 2 * g(x) + x);
}
```

Maehlyova metoda

```
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
	double sum = 0;
	for (int i = 0; i < roots.size(); i++) {
		sum += f(x) / (x - roots.get(i));
	}
	x = x - f(x) / (df(x) - sum);
}
```

Müllerova metoda

```
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
	//koeficient a
	double a = ((x0 - x2) * (f(x1) - f(x2)) - (x1 - x2) * (f(x0) -
	f(x2))) / ((x0 - x2) * (x1 - x2) * (x1 - x0));
	//koeficient b
	double b = (Math.pow((x0 - x2),2) * (f(x1) - f(x2)) - Math.pow((x1 -
	x2),2) * (f(x0) - f(x2))) / ((x0 - x2) * (x1 - x2) * (x0 - x1));
	//koeficient c
	double c = f(x2);
	x = x2 - ((2*c)/(b + Math.signum(b)*(Math.sqrt(Math.pow(b, 2) -
	4*a*c))));
}
```

Laguerrova metoda

```
//one: objekt komplexního čísla s hodnotou 1 + 0i
//n: objekt komplexního čísla s reálnou hodnotou rovnou stupni polynomu
while (Math.abs(x - xPredchozi) > tol && f(x) > tol) {
      //Laguerrova metoda využívá komplexní aritmetiku, místo
      //standartních matematických operací jsou tak použity vlastní
      //implementované metody pro operace s komplexními čísly
      g = df(x).divide(f(x));
      h = g.power(2).substract(ddf(x).divide(f(x)));
      //volba mezi variantami jmenovatele
      denomPlus = g.add((n.substract(one)
                  .multiply(
                  n.multiply(h).substract(g.power(2)))).sqrt());
      denomMinus = g.substract((n.substract(one)
                  .multiply(
                  n.multiply(h).substract(g.power(2)))).sqrt());
      denominator = ComplexNumber.max(denomPlus, denomMinus);
      x = x.substract(n.divide(denominator));
}
```

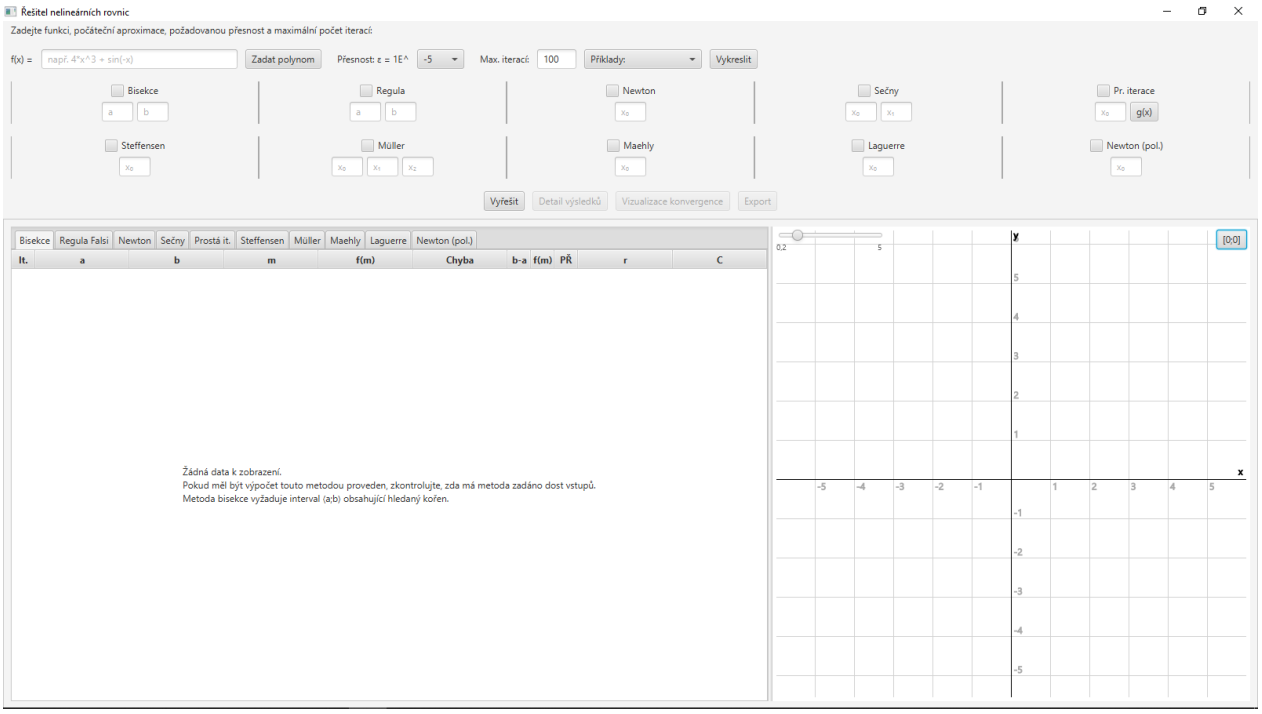
Metody pro vyhodnocení funkce či její derivace – ukázka syntaxe externí knihovny

```
//inicializace objektů
//inicializace neznámé: první argument je písmeno, kterým se ve výrazu
//vyjadřuje, druhý pak počáteční hodnota
Argument x = new Argument("x", 0.0);
//inicializace výrazů: první argument je funkce ve formátu String,
//druhý je pak výše inicializovaná proměnná, která se ve funkci nachází.
//V případě derivace má řetězec funkce tvar "der(x^2 - x, x)",
//kde druhým argumentem vyjadřujeme, podle které proměnné se má
//funkce derivovat
Expression e = new Expression("x^3 – x^2 – x - 1", x);
Expression de = new Expression(
      "der(" + e.getExpressionString() + ", x)", x);

//vyhodnocení funkcí v bodě
private double f(double point){
      x.setArgumentValue(point);
      return e.calculate();
}
private double df(double point){
      x.setArgumentValue(point);
      return de.calculate();
}
```

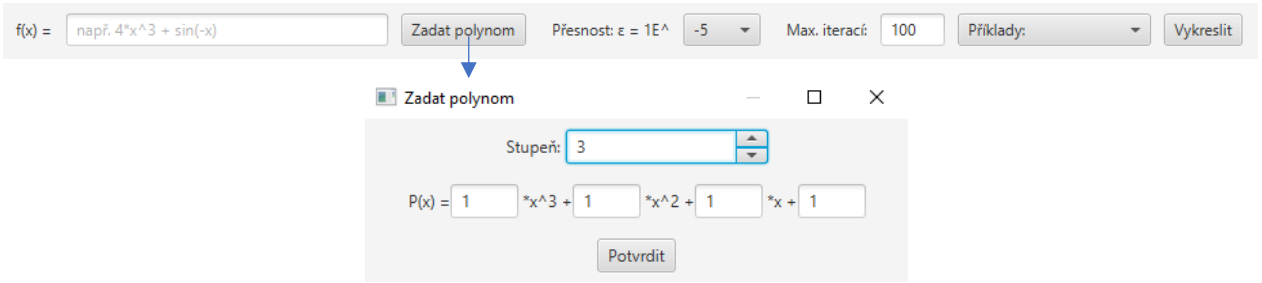
PŘÍLOHA B – UŽIVATELSKÁ PŘÍRUČKA K APLIKACI

Hlavní okno aplikace je rozděleno na tři sekce – pole pro vstup, výstupní tabulky a vykreslovací okno.



Obrázek 4: Hlavní okno aplikace po spuštění

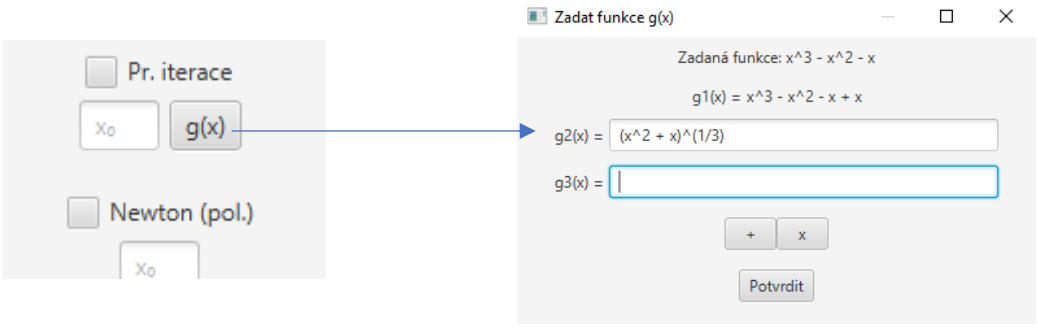
První krok při řešení funkce je její zadání do aplikace. To se provede buď vepsáním funkce do příslušného textového pole, v případě polynomu lze využít jednoduchý dialog vyvolaný tlačítkem ‘Zadat polynom’, kde si uživatel zvolí stupeň polynomu (minimální hodnota je 2) a vyplní jednotlivá pole koeficientů. Třetí variantou je pak výběr z rozbalovací nabídky příkladů.



Obrázek 5: Vstupní sekce aplikace

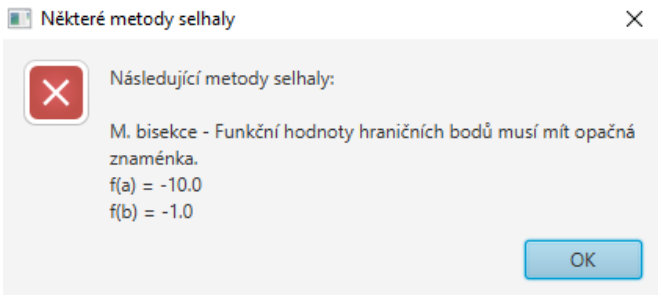
Jakmile je funkce zadaná, lze si ji nejprve nechat načrtnout stisknutím tlačítka ‘Vykreslit’ (v případě výběru z nabídky příkladů proběhne vykreslení samo). V rozbalovací nabídce si pak lze zvolit požadovanou přesnost a maximální počet iterací.

Následně si uživatel pomocí zaškrtávacích políček vybere metody, které chce na funkci aplikovat, a ke každé z nich zadá do příslušných polí počáteční aproximace. V případě metody prosté iterace a Steffensenovy metody je pak možné zadat další iterační funkce pomocí dialogu vyvolaného tlačítkem ‘g(x)’ umístěným u metody prosté iterace. V něm jsou na prvních dvou řádcích předepsané zadaná funkce a automaticky vygenerovaná iterační funkce ve tvaru $g_1(x) = f(x) + x$. Další funkce se pak vepisují do prázdných textových polí. V případě potřeby lze přidat libovolné množství dalších polí pomocí tlačítka ‘+’ ve spodní části dialogu a tlačítkem ‘x‘ ze seznamu naopak prázdná pole odstranit. Uložení zadaných iteračních funkcí se pak provede stisknutím tlačítka ‘Potvrdit’.



Obrázek 6: Zadání vstupu metod a iteračních funkcí g(x)

Jakmile jsou vybrány metody a zadány aproximace, lze zahájit výpočet stiskem tlačítka ‘Vyřešit’. Aplikace vyřeší zadanou funkci všemi vybranými metodami, pokud je to možné. V případě špatného vstupu se objeví chybová zpráva se stručným popisem chyby. Taková situace přeruší výpočet pouze pro danou metodu, výsledky ostatních metod, mají-li správný vstup, budou k dispozici.



Obrázek 7: Ukázka chyby vstupu u metody bisekce

Po dokončení výpočtu si lze prohlédnout kompletní průběh výpočtu po jednotlivých iteracích v sekci výstupních tabulek. Zde je k dispozici pro každou implementovanou metodu jedna záložka obsahující tabulku s výsledky. V případě metod, které produkují více sad výsledků (metoda prosté iterace a Steffensenova metoda generující výsledky pro každou iterační funkci a Newtonova metoda pro polynomy s Maehlyovou metodou generující výsledky pro každý reálný kořen) jsou jednotlivé tabulky uspořádány do záložek v harmonikovém zobrazení s patřičnými popisky. Každá tabulka obsahuje následující sloupce:

- číslo iterace,
- počáteční aproximace v dané iteraci,
- nová aproximace,
- funkční hodnota v bodě nové aproximace,
- chyba,
- ukončovací kritéria (meziiterační rozdíl, funkční hodnota, divergence a přesné řešení),
- řád metody,
- rychlost konvergence.

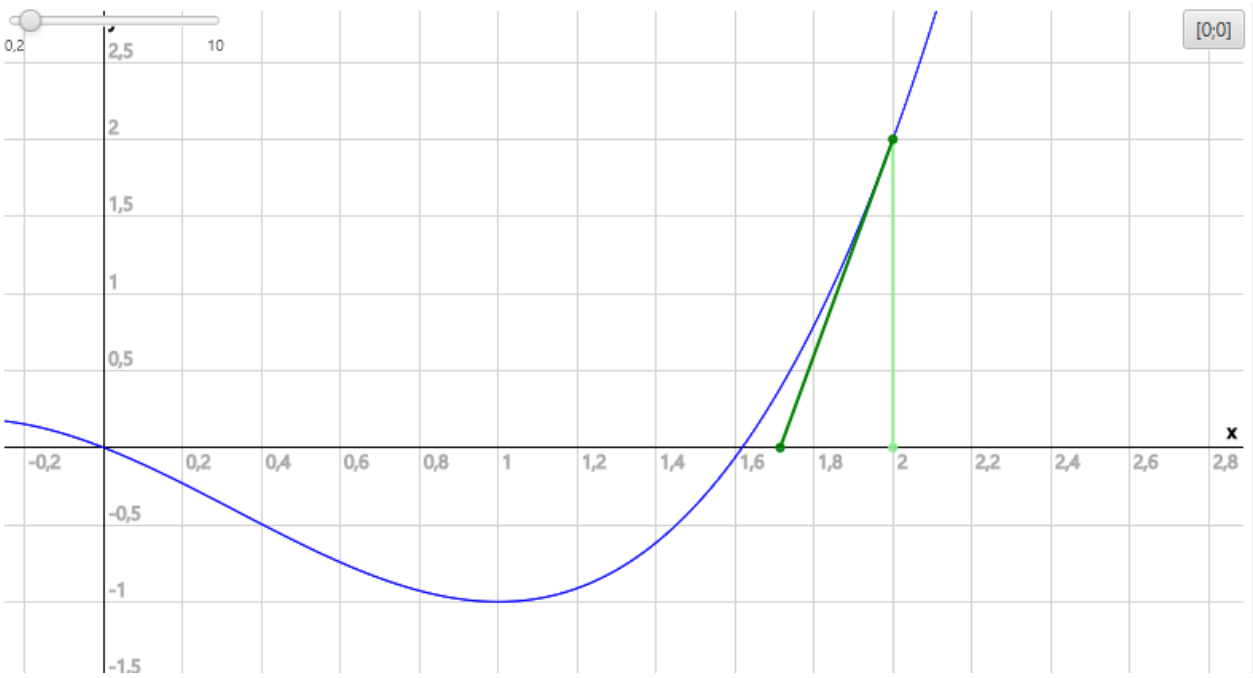
It.	x_i	x_{i+1}	$f(x_{i+1})$	Chyba	$x_{i+1}-x_i$	$f(x_i)$	Div	PŘ	r	C
0	2	1,7142857144	0,3848396506	0,0962517256					0	0
1	1,7142857144	1,6265780732	0,0311947663	0,0085440844					0	0
2	1,6265780732	1,6181106971	0,0002775561	0,0000767084					1,94612451	0,8129786063
3	1,6181106971	1,618033995	2,267558230e-08	6,267376618e-09		✓			1,99712299	1,036480239
4	1,618033995	1,6180339887	0	0	✓	✓		✓	NaN	NaN

Obrázek 8: Ukázka řešení funkce $f(x) = x^3 - x^2 - x$ Newtonovou metodou

Výpočet chyby probíhá po dokončení výpočtu všemi metodami. Z výsledků je vybráno řešení s nejnižší funkční hodnotou, které je označeno za nejlepší řešení a v porovnání s ním se poté určuje hodnota chyby všech ostatních metod. Nejde tedy o odchylku od přesného řešení, ale pouze od nejbližšího odhadu.

Při výběru iterace v tabulce se ve vykreslovacím okně k průběhu funkce doplní grafická interpretace získání nové aproximace. V tomto okně se lze pohybovat pomocí následujících funkcí:

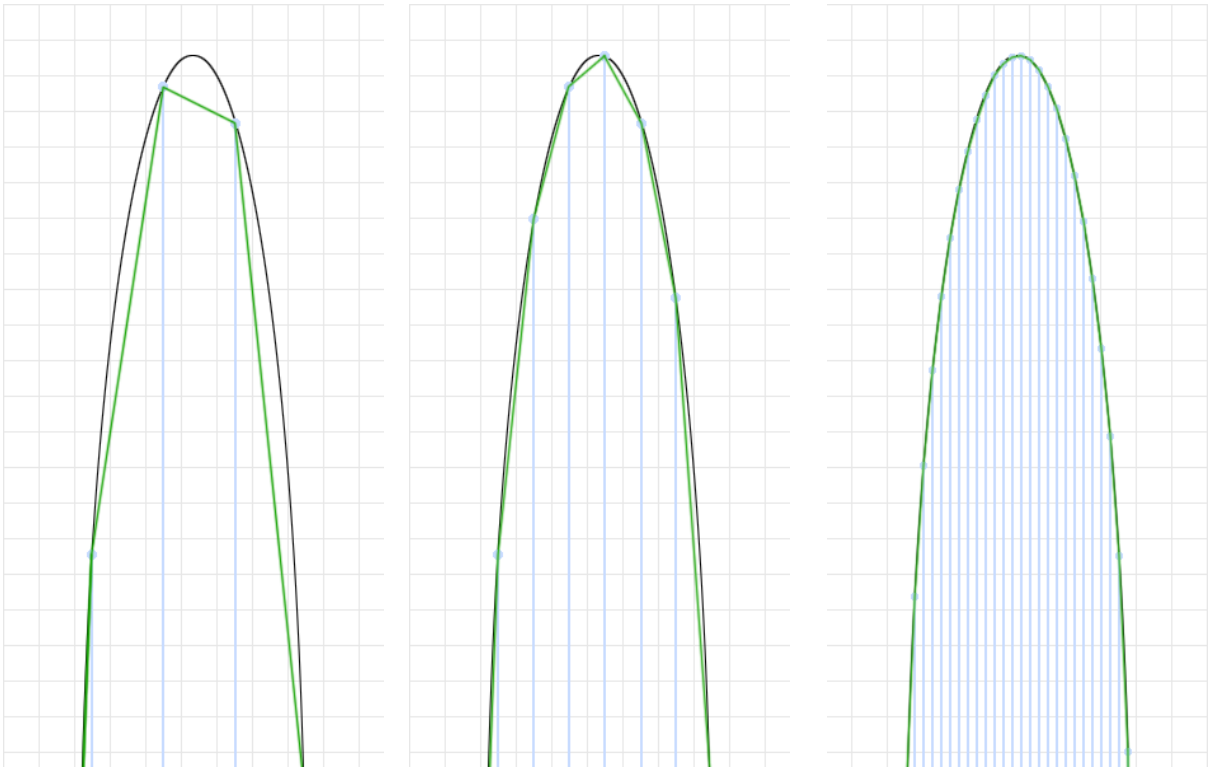
- posun: tažení levým tlačítkem myši,
- přiblížení a oddálení: kolečko myši,
- změna měřítka jen jedné osy:
    - zmenšení a zvětšení podél osy x: kolečko myši s držením klávesy Ctrl,
    - zmenšení a zvětšení podél osy y: kolečko myši s držením klávesy Alt,
- Obnovení náhledu do původního měřítka a zaměření na počátek: tlačítko '[0;0]',
- Změna rozlišení grafu: posuvník v levém horním rohu okna náhledu.



Obrázek 9: Ukázka vykreslení funkce s detailem první iterace Newtonovy metody

Změna rozlišení grafu umožňuje zpřesnit vykreslování průběhu funkce, především v oblastech s přechodem z prudkého stupání do prudkého klesání při větším oddálení. Ve výchozím rozlišení s hodnotou 1 aplikace vypočítává funkční hodnotu (svislou polohu bodu) právě jednou pro každý obrazový bod a následně vykresluje spojnici mezi těmito body. V situaci, kdy funkce prudce stoupá a ve vedlejším pixelu naopak prudce klesá, může dojít k jevu, kdy pro jeden z bodů vyjde funkční hodnota menší než maximum funkce, a ve vedlejším bodě (kde již funkce klesá) vyjde ještě nižší – vykreslené maximum tak bude níže, než má ve skutečnosti být. Zvýšení rozlišení, například na hodnotu 4, způsobí, že aplikace vypočítá funkční hodnotu pro celkem čtyři rovnoměrně rozdělené body v intervalu odpovídajícímu jednomu pixelu

a vykreslí spojnice mezi všemi po sobě jdoucími body. To má za následek věrnější kopírování průběhu funkce. Celý tento efekt je ilustrován na následujících obrázcích:



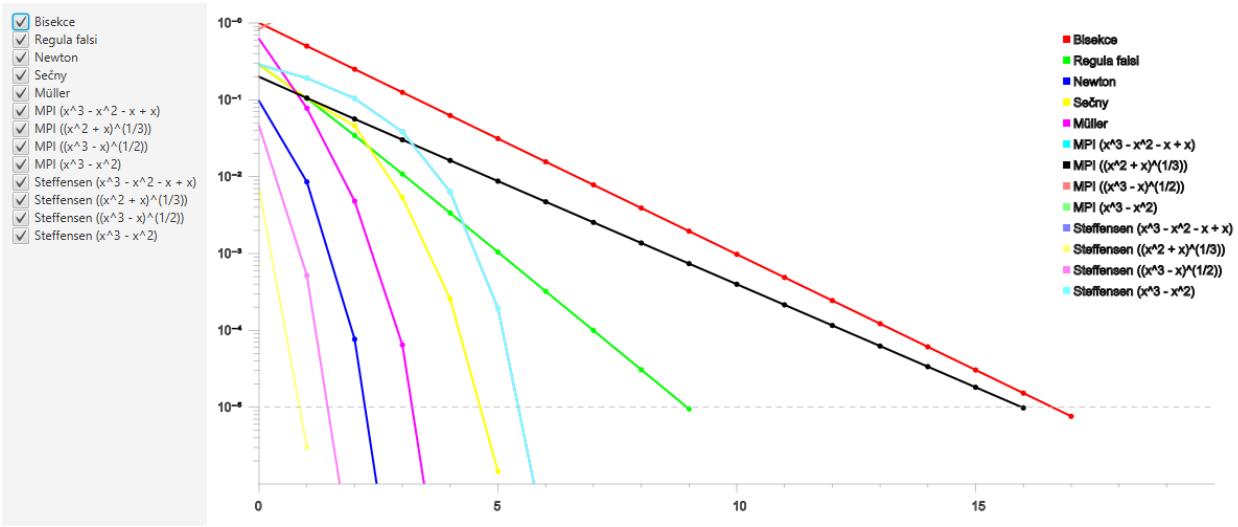
Obrázek 10: Ilustrace vlivu hodnoty rozlišení při vykreslování grafu

Kromě prohlížení jednotlivých tabulek iterací pro každou metodu má uživatel k dispozici i celkový přehled v podobě tabulky s klíčovými hodnotami pro všechny metody. Ten lze zobrazit tlačítkem ‘Detail výsledků’. Každý řádek této tabulky odpovídá jedné metodě a lze zde nalézt informace týkající se nalezeného řešení, jeho funkční hodnoty, řád metody, rychlost konvergence, počet potřebných iterací a celkový čas, který samotný výpočet zabral.

Metoda	g(x)	ξ	f(ξ)	Řád metody	Rychlost konv.	Iterace	Čas
M. bisekce		1,6180343628	1,353301094e-06	1	0,5	18	1 ms
M. regula falsi		1,6180330808	3,285019242e-06	1,0000032	0,3090301126	12	1 ms
Newtonova m.		1,6180339887	0	1,99712299	1,036480239	5	0 ms
M. sečen		1,6180339884	1,439908193e-09	1,59154841	0,767656551	7	1 ms
Müllerova m.		1,6180339887	0	1,75633051	0,1520882875	6	1 ms
M. prosté it.	x^3 - x^2 - x + x	---	---	---	---	3	0 ms
M. prosté it.	(x^2 + x)^(1/3)	1,618035527	5,565393628e-06	0,99999962	0,5393418954	20	1 ms
M. prosté it.	(x^3 - x)^(1/2)	---	---	---	---	7	1 ms
M. prosté it.	x^3 - x^2	---	---	---	---	3	0 ms
Steffensenova m.	x^3 - x^2 - x + x	1,6180339887	2,493560913e-13	2,12691443	14,5519515983	8	0 ms
Steffensenova m.	(x^2 + x)^(1/3)	1,618033989	8,148894892e-10	1,22542672	0,0013124552	3	0 ms
Steffensenova m.	(x^3 - x)^(1/2)	1,6180339894	2,470763594e-09	0,48850066	2,338884520e-06	4	0 ms
Steffensenova m.	x^3 - x^2	1,6180339887	2,493560913e-13	2,12691443	14,5519515983	8	1 ms

Obrázek 11: Souhrn výsledků všech testovaných metod

Poslední formou výstupu aplikace je graf konvergence jednotlivých metod, který lze otevřít tlačítkem ‘Vizualizace konvergence’. V levé části tohoto okna si může uživatel vybrat, pro které metody se má graf vykreslovat. Svislá osa grafu označuje velikost chyby a má logaritmické měřítko, vodorovná osa pak vyjadřuje číslo iterace. Rozsah vodorovné osy se přizpůsobuje maximální hodnotě zobrazovaných metod. Pokud je však nejvyšší hodnota výrazně vyšší než druhá nejvyšší, může se jednat o divergující nebo velmi pomalu konvergující metodu, která by zbytečně zmenšovala vykreslovací prostor pro ostatní metody a zhoršovala tak čitelnost grafu. V takovém případě je rozsah osy určen druhým nejvyšším počtem iterací.



Obrázek 12: Graf konvergence

Všechny popsané výstupy aplikace, tedy tabulky řešení jednotlivých metod, závěrečný souhrn

a graf konvergence lze uložit pomocí tlačítka ‘Exportovat’. Jeho stisknutí vyvolá nejprve informativní dialog s instrukcemi, poté se otevře systémový dialog pro výběr adresáře. Uživatel si vybere libovolný adresář, a po potvrzení budou do zvoleného adresáře vyexportovány tabulky iterací s vygenerovaným názvem v podobě ‘*název metody*.csv‘. V případě, že v adresáři již jeden nebo více takových souborů existuje, dostává nový soubor zvyšující se číselnou koncovku, dokud se ho nepovede vytvořit. Stejným způsobem se vytvoří soubor ‘Shrnutí.csv’ obsahující data z tabulky vyvolané tlačítkem ‘Detail výsledků’ a graf konvergence se všemi metodami jako obrázek s názvem ‘Konvergence.png‘.