

# **Numerické řešení nelineárních rovnic v programu Mathematica**

Jakub Kovařík



# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jakub Kovařík**

Osobní číslo: **A14746**

Studijní program: **B3902 Inženýrská informatika**

Studijní obor: **Informační a řídicí technologie**

Forma studia: **kombinovaná**

Téma práce: **Numerické řešení nelineárních rovnic v programu Mathematica**

Téma anglicky: **Numerical Solutions of Nonlinear Equations in the Mathematica Environment**

Zásady pro vypracování:

1. Seznamte se s problematikou numerického řešení nelineárních rovnic.
2. Definujte základní pojmy související s nelineárními rovnicemi a metodami jejich numerického řešení.
3. Nastudujte a popište vybrané numerické metody užívané při řešení nelineárních rovnic. Zaměřte se zejména na metodu půlení intervalu, metodu regula falsi, iterační metodu, Newtonovu metodu a metodu sečen.
4. Jednotlivé metody otestujte na vhodně zvolených příkladech.
5. Vytvořte programy v software Mathematica na použití těchto metod a demonstруйте je na ukázkových příkladech.
6. Porovnejte přesnosti jednotlivých metod, které jste použil, a uveďte jejich výhody a nevýhody.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. HOROVÁ, Ivana a Jiří ZELINKA. Numerické metody. Vyd. 2., rozš. Brno: Masarykova Univerzita v Brně, 2004, 285 s. ISBN 80-210-3317-7.
2. NAVARA, Mirko a Aleš NĚMEČEK. Numerické metody. 1. vyd. Praha: Česká technika – nakladatelství ČVUT, 2005, 142 s. ISBN 80-010-2689-2.
3. KUBÍČEK, Milan, Miroslava DUBCOVÁ a Drahoslava JANOVSÁ. Numerické metody a algoritmy. 2. oprav. vyd. Praha: Vydavatelství VŠCHT Praha, 2005, 188 s. ISBN 80-708-0558-7.
4. ISAACSON, Eugene. Analysis of numerical methods. New York: Dover Publications, 1994, 541 s. ISBN 04-866-8029-0.
5. HOFFMAN, Joe D. Numerical methods for engineers and scientists. 2nd ed., rev. and expanded. New York: Marcel Dekker, c2001, xi, 823 p. ISBN 08-247-0443-6.
6. ZAROWSKI, Christopher J. An introduction to numerical analysis for electrical and computer engineers. Hoboken, NJ: Wiley, 2004, xvi, 586 s. ISBN 04-714-6737-5.

Vedoucí bakalářské práce:

**Mgr. Jana Řezníčková, Ph.D.**

Ústav matematiky

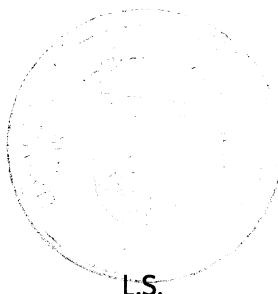
Datum zadání bakalářské práce:

**6. března 2015**

Termín odevzdání bakalářské práce:

**22. května 2015**

Ve Zlíně dne 6. března 2015



L.S.

doc. Mgr. Milan Adámek, Ph.D.  
*děkan*

prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## Prohlašuji, že

- беру на вѣдомі, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- беру на вѣдомі, že bakalářské práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- беру на вѣдомі, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- беру на вѣдомі, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- беру на вѣдомі, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- беру на вѣдомі, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

## Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně



.....  
podpis autora

## **ABSTRAKT**

Cílem bakalářské práce je nastudovat a popsat numerické metody řešení nelineárních rovnic. Použití jednotlivých metod bude názorně ilustrováno na konkrétních příkladech, k jejichž řešení bude využit software Mathematica.

Klíčová slova: nelineární numerické metody, metoda půlení intervalů, metoda regula falsi, metoda prosté iterace, Newtonova metoda, metoda sečen, Wolfram Mathematica

## **ABSTRACT**

The purpose of this bachelor thesis is describe numeric methods used to solve nonlinear equations. Software Wolfram Mathematica is used to illustrate progress of calculation of some selected numeric methods.

Keywords: numerical methods, bisection method, false position method, fixed-point iteration method, Newton's method, secant method, Wolfram Mathematica

Velmi děkuji Mgr. Janě Řezníčkové, Ph.D. za cenné rady, věcné připomínky a vstřícnost při vypracování bakalářské práce.

## OBSAH

<b>1</b>	<b>ÚVOD .....</b>	<b>9</b>
<b>I</b>	<b>TEORETICKÁ ČÁST .....</b>	<b>9</b>
<b>2</b>	<b>CHYBY A JEJICH ODHADY .....</b>	<b>11</b>
2.1	PLATNÉ CIFRY.....	12
2.2	CHYBY PŘI VÝPOČTU .....	13
2.2.1	Iterační chyba .....	14
2.2.2	Chyba aproximace .....	15
2.2.3	Chyba zaokrouhlení .....	15
2.2.4	Celková chyba .....	15
2.3	PŘÍKLADY.....	17
<b>3</b>	<b>ŘEŠENÍ NELINEÁRNÍCH ROVNIC .....</b>	<b>21</b>
3.1	HLEDÁNÍ KOŘENŮ.....	25
3.1.1	Separace kořenů .....	25
3.1.2	Zpřesnění výsledků .....	26
<b>4</b>	<b>METODA PŮLENÍ INTERVALŮ.....</b>	<b>27</b>
<b>5</b>	<b>METODA REGULA FALSI.....</b>	<b>30</b>
<b>6</b>	<b>METODA PROSTÉ ITERACE .....</b>	<b>34</b>
<b>7</b>	<b>NEWTONOVA METODA.....</b>	<b>42</b>
<b>8</b>	<b>METODA SEČEN.....</b>	<b>45</b>
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>46</b>
<b>9</b>	<b>METODA PŮLENÍ INTERVALŮ.....</b>	<b>48</b>
9.1	IMPLEMENTACE ALGORITMU .....	48
9.2	PŘÍKLADY.....	49
<b>10</b>	<b>METODA REGULA FALSI.....</b>	<b>53</b>
10.1	IMPLEMENTACE ALGORITMU .....	53
10.2	PŘÍKLADY.....	54
<b>11</b>	<b>METODA PROSTÉ ITERACE .....</b>	<b>57</b>
11.1	IMPLEMENTACE ALGORITMU .....	57
11.2	PŘÍKLADY.....	58
<b>12</b>	<b>NEWTONOVA METODA.....</b>	<b>62</b>
12.1	IMPLEMENTACE ALGORITMU .....	62
12.2	PŘÍKLADY.....	63

<b>13</b>	<b>METODA SEČEN.....</b>	<b>66</b>
13.1	IMPLEMENTACE ALGORITMU .....	66
13.2	PŘÍKLADY.....	67
<b>14</b>	<b>POUŽITÍ SOFTWARE MATHEMATICA .....</b>	<b>69</b>
	<b>ZÁVĚR.....</b>	<b>71</b>
	<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>72</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>73</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>74</b>
	<b>SEZNAM TABULEK .....</b>	<b>75</b>
	<b>SEZNAM PŘÍLOH .....</b>	<b>76</b>



## 1 ÚVOD

Tato bakalářská práce se zabývá řešením nelineárních rovnic s jednou neznámou pomocí vybraných numerických metod. Cílem je využití softwaru Wolfram Mathematica od společnosti **Wolfram Research**. Program je dostupný pro tři hlavní operační systémy *Windows*, *Linux* a *OS X*. Disponuje vlastním programovacím jazykem zvaným **Wolfram Language**, který umožňuje poměrně snadno realizovat nejrůznější matematické algoritmy. Program je koncipován jako pomůcka při výuce a demonstraci matematiky. Lze tedy na něm snadněji a rychle porovnávat jednotlivé matematické postupy, a to nejenom pro řešení nelineárních rovnic numerickými metodami.

Postupně bylo implementováno celkem 5 funkcí, které jsou schopny řešit nelineární rovnice. Každá z nich využívá jednu z následujících numerických metod - metoda půlení intervalů (bisekce), metoda regula falsi, metoda prosté iterace, Newtonova metoda a metoda sečen.

V textu je porovnána rychlost konvergence, vhodné typy matematických funkcí, výpočetní složitost a počet kroků jednotlivých metod vedoucí k dosažení požadovaného výsledku. Vstupem každé z vytvořených funkcí je kromě řešené rovnice také požadovaná přesnost nebo relativní chyba výsledku. Na výstupu je pak možné kromě výsledku zobrazit i průběh výpočtu v každém kroku a sledovat změnu v závislosti na definované přesnosti výpočtu, počátečním odhadu, typu vstupní funkce apod.

# I. TEORETICKÁ ČÁST

## 2 CHYBY A JEJICH ODHADY

Ještě než budou ukázány vybrané numerické metody k řešení nelineárních rovnic, je vhodné zdůraznit některé základní skutečnosti, na něž je možné narazit nejenom při výpočtu nelineárních rovnic, ale jsou společné pro všechny metody numerické matematiky. Jedná se především o chyby zvolené numerické metody, chyby vznikající při zaokrouhlování v průběhu výpočtu a vliv pořadí těchto operací na celkový výsledek.

Často se používají numerické metody k výpočtu rovnic odvozených z fyzikálních zákonů, které se snaží popsat reálné situace. Pro lepší představu to například může být fyzikálně a matematicky složitý problém jako je proudění tekutin a modelování v problémech klimatu. Uvedenou situaci nazveme fyzikálním jevem, který se nejprve převede na matematický model (tj. soustava rovnic popisující uvedenou reálnou situaci). Bohužel rovnice popisující tyto složité jevy nelze ve většině případů řešit analytickým způsobem a je nutné k jejich výpočtu použít metody numerické matematiky. To znamená vytvořit numerický model, tj. soustavu lineárních algebraických rovnic, kterou jsme schopni řešit (nejčastěji pomocí počítače). I řešení nelineárního matematického modelu je v konečné fázi převedeno na soustavu lineárních rovnic. Právě popsany postup lze zobrazit následujícím schématem.



Obr. 2.1 Postup řešení pomocí numerického modelu

Pod pojmem problém si můžeme představit například popis fyzikálního jevu nebo jen problém čistě matematický, který se bude řešit uvedeným postupem. Matematický model představuje rovnici nebo soustavu rovnic, které popisují fyzikální model, například by to mohly být Maxwellovy rovnice, pokud by se jednalo o popis fyzikálního děje související s elektromagnetickým polem apod. Numerický model pak umožňuje řešit tyto složité rovnice převedením na posloupnost jednoduchých matematických operací (sčítání, odčítání, násobení atd.). Numerické výpočty se v dnešní době provádí téměř výhradně pomocí počítače, protože je to mnohonásobně rychlejší a spolehlivější. Navíc spoustu problémů by ani nebylo v silách člověka v rozumném čase vyřešit, nehledě na možné chyby z nepozornosti, které jsou při použití počítače eliminovány. Samotné algoritmy numerických metod pracují s čísly. Používají se přitom běžné matematické operace jako jsou sčítání, odčítání, násobení atd. Ať už je zpracovává počítač nebo člověk na papíře, používá pouze aproximace přesných hodnot. Níže jsou ukázány matematické pojmy sloužící k posouzení správnosti výsledného řešení. Numerická metoda musí podat kromě výsledku výpočtu také odhad chyby tohoto řešení. Bez této informace je výsledek téměř bezcenný. [1] [2] [7]

## 2.1 Platné cifry

První důležitým pojmem při posuzování správnosti výpočtu, jsou tzv. **platné cifry**. Jsou to takové číslice, u kterých se předpokládá, že jsou ve výsledku správné. Vstupem jednotlivých numerických algoritmů budou opět čísla, např. počáteční odhad hodnoty nebo přibližný interval, na kterém se nachází kořen. Je velmi důležité vědět, kolik platných číslic bude mít výsledek řešení, protože pokud nemáme vůbec žádnou další informaci (pouze číslo), tak bez znalosti odhadu chyby nelze rozhodnout, zda je postup a výsledek vyhovující nebo ne. U jednotlivých metod uvedených v této práci budou ukázány vztahy pro určení odhadu chyby v každém kroku výpočtu (tj. každého mezi-výsledku). Poté podle následující definice je možné odvodit počet platných cifer.

### Definice 2.1

Aproximace  $\tilde{x}$  čísla  $x$  má  $s$  platných cifer, jestliže  $s$  je největší celé nezáporné číslo takové, že platí

$$\left| \frac{x - \tilde{x}}{x} \right| \leq \frac{1}{2} 10^{-s} \quad (2.1)$$

[1] [7]

Jinými slovy relativní chyba pro jednu platnou číslici musí být menší než 0,05 (5 %). Pro dvě platné číslice musí být relativní chyba menší nebo rovna 0,005 (0,5 %) atd. Nejčastěji na konci výpočtu je rovněž potřeba znát kolik desetinných míst je ve výsledku správně zaokrouhleno. Návod jak na to podává následující definice.

### Definice 2.2

Nechť  $x$  je reálné číslo, které má obecně nekonečné dekadické vyjádření. Číslo  $x^{(d)}$ , které má  $d$  desetinných míst, je správně zaokrouhlenou hodnotu čísla  $x$ , platí-li

$$|x - x^{(d)}| \leq \frac{1}{2} 10^{-d}. \quad (2.2)$$

V takto správně zaokrouhleném čísle jsou **všechny cifry platné**. [1] [7]

Číslo obsahuje na zvoleném místě platnou cifru právě tehdy, když se od přesného čísla liší nejvýše o 5 jednotek řádu příslušného následujícímu desetinnému místu. [7] Příklad níže ukazuje, jak pomocí těchto definic správně zaokrouhlovat.

**Příklad 2.1**

Zaokrouhlete přesné číslo 53486 na 3 platné číslice.

Pohledem na zadané číslo nejprve zjistíme, že na 3. místě je čtverka v pozici stovek. To znamená, že správně zaokrouhlené číslo na uvedený počet platných cifer se od nezaokrouhleného nesmí lišit o více než 5 desítek. Výsledek tedy bude 53500. Ověření chyby se provede následovně

$$\left| \frac{53486 - 53500}{53486} \right| \leq \frac{1}{2} 10^{-3},$$
$$2,449 \cdot 10^{-4} \leq 0,5 \cdot 10^{-3}.$$

**2.2 Chyby při výpočtu**

V předchozí části bylo zmíněno zaokrouhlování na zvolený počet desetinných míst. Každou takovou operací vznikají chyby, které se mohou v průběhu výpočtu zvětšovat nebo naopak navzájem rušit. K posouzení jejich velikostí se využívá následujících pojmů.

**Definice 2.3**

Nechť  $x$  je přesné číslo a nechť  $\tilde{x}$  značí aproximaci  $x$ , potom rozdíl

$$\tilde{x} - x, \quad (2.3)$$

se nazývá **absolutní chyba**. Každé nezáporné číslo  $\alpha$ , pro které platí

$$|\tilde{x} - x| \leq \alpha, \quad (2.4)$$

se nazývá **odhad absolutní chyby**. Absolutní chyba vztažená na hodnotu přesné veličiny

$$\left| \frac{\tilde{x} - x}{x} \right|, \quad x \neq 0 \quad (2.5)$$

se nazývá **relativní chyba**. Každé nezáporné číslo  $\delta$ , pro které platí

$$\left| \frac{\tilde{x} - x}{x} \right| \leq \delta, \quad x \neq 0 \quad (2.6)$$

se nazývá **odhad relativní chyby**. [1] [7]

Uvedený příklad byl záměrně vybrán, tak aby ukázal rozdíly mezi absolutní a relativní chybou.

### Příklad 2.2

Jsou zadány dvě dvojice čísel  $x_1 = 1,84$ ,  $\tilde{x}_1 = 1,85$  a  $x_2 = 5,89$ ,  $\tilde{x}_2 = 5,88$ . Jaké jsou absolutní a relativní chyby?

Absolutní chyba a relativní chyba první dvojice čísel je

$$\begin{aligned} |\tilde{x}_1 - x_1| &= 1,85 - 1,84 = 0,01 \\ \left| \frac{\tilde{x}_1 - x_1}{x_1} \right| &= \frac{1,85 - 1,84}{1,84} = 0,054 \end{aligned}$$

Absolutní chyba a relativní chyba druhé dvojice čísel je

$$\begin{aligned} |\tilde{x}_2 - x_2| &= 5,89 - 5,88 = 0,01 \\ \left| \frac{\tilde{x}_2 - x_2}{x_2} \right| &= \frac{5,89 - 5,88}{5,89} = 0,017 \end{aligned}$$

Zatímco absolutní chyba každé dvojice čísel je stejná, jejich relativní chyba je naopak různá. V průběhu výpočtů je možné rozlišit následující druhy chyb:

- chyby v zadaných parametrech problému,
- algebraické chyby vytvořené v průběhu výpočtu (např. v důsledku nepozornosti),
- iterační chyby,
- chyby aproximací,
- chyby zaokrouhlení.

První dva typy chyb nebudou dále rozebírány, protože je nelze rozumně popsat a předvídat. Bude se předpokládat, že zadání příkladu je v pořádku a rovněž je možné vyloučit chyby z nepozornosti například použitím počítače.

#### 2.2.1 Iterační chyba

Je chyba vznikající v každém kroku iterační metody, jejíž mezivýsledky se asymptoticky přibližují k řešení a naopak chyby těchto mezivýsledků se musejí limitně blížit nule.

Údaj o této chybě v každém kroku iterační metody je rozhodujícím faktorem k určení, zda daný výsledek je uspokojivou aproximací řešení a umožňuje rovněž omezit počet iteračních kroků (tj. nepokračovat ve výpočtu, pokud už bylo dosaženo požadované přesnosti) a mnohdy tak výrazně urychlit výpočet. [5]

### 2.2.2 Chyba aproximace

Je rozdíl mezi přesným řešením přesně popsaného problému a mezi přesným řešením přibližně popsaného problému. Tato chyba může být redukována pouze lepší aproximací (matematickým popisem) uvedeného problému. Jako příklad si lze představit nahrazení derivace diferencí. Rozdíl přesných výsledků při použití těchto postupů bude právě chyba aproximace. [5]

### 2.2.3 Chyba zaokrouhlení

Je chyba způsobená konečnou délkou zobrazení čísla v paměti počítače. Nepřesnost rovněž vzniká i při ručním výpočtu na papír, neboť se také používají čísla konečné délky. Rozsah hodnot, které může číslo reprezentované v počítači mít, se liší od použité počítačové architektury (například 32 nebo 64 bitový systém). Nicméně se nejedná o definitivní omezení. Programově lze zajistit číslo téměř „libovolné“ velikosti například tak, že se rozloží na více částí, které obsáhnou standardní nabízené datové typy a na tyto díly se pak nahlíží jako na jedno číslo. Software Mathematica má toto omezení automaticky vyřešeno a je možné pouhým zadáním příkazu nastavit počet cifer, které se při výpočtu použijí. Jedná se o podstatné ulehčení při vytváření algoritmů oproti jiným programovacím jazykům, kde většinou bez přídavných knihoven by bylo nutné celou tuto (ne zrovna jednoduchou) funkčnost doplnit. [5]

### 2.2.4 Celková chyba

Jak plyne ze schématu a popisu na obrázku 2.1, matematický model popisující např. fyzikální děj je možné zapsat následující rovnicí

$$Y = F(x_1, \dots, x_n),$$

kde  $Y$  je hledaná veličina, která je jednoznačně určena hodnotami  $(x_1, \dots, x_n)$ . Pokud byl daný problém fyzikální, tak se s největší pravděpodobností dopustíme první nepřesnosti při výpočtu. Tato nepřesnost ovšem nesouvisí s numerickými metodami, ale má původ v použité fyzikální teorii a její kvalita a přesnost už je otázkou fyzikální, a ne matematickou, proto nebude dále rozebírána. Nyní se uvedená závislost nahradí

obecně nějakou numerickou metodou

$$y = f(x_1, \dots, x_n).$$

Veličina  $Y$  popsaná funkcí  $F$  se nahradila novou funkční závislostí. Tato náhrada bude záležet na použitém druhu numerické metody, kterých je celá řada. Celý postup se prováděl z toho důvodu, že nelze nalézt řešení analytickou cestou, proto je nutné volit numerické postupy. Navíc hodnoty  $(x_1, \dots, x_n)$  nejsou známy zcela přesně, ale velmi často se dosazují jejich aproximace  $\tilde{x}_i$ . Potom rovnice přejde na tvar

$$\hat{y} = f(\tilde{x}_1, \dots, \tilde{x}_n).$$

Navíc kvůli zaokrouhlování mezivýsledků není možné provádět všechny výpočty přesně, proto výsledná hodnota bude ve tvaru

$$\tilde{y} = \tilde{f}(\tilde{x}_1, \dots, \tilde{x}_n).$$

[1] Celková chyba výpočtu bude rozdíl veličiny  $Y$  a vypočtené hodnoty  $\tilde{y}$ .

$$Y - \tilde{y} = Y - y + [f(x_1, \dots, x_n) - f(\tilde{x}_1, \dots, \tilde{x}_n)] + [f(\tilde{x}_1, \dots, \tilde{x}_n) - \tilde{f}(\tilde{x}_1, \dots, \tilde{x}_n)]. \quad (2.7)$$

Postup odvození lépe vynikne, pokud se rozepíší chyby v jednotlivých krocích. Rozepíší se přitom:

- **chyba metody:**  $Y - y = F(x_1, \dots, x_n) - f(x_1, \dots, x_n)$
- **primární chyba:**  $y - \hat{y} = f(x_1, \dots, x_n) - f(\tilde{x}_1, \dots, \tilde{x}_n)$
- **sekundární chyba:**  $\hat{y} - \tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_n) - \tilde{f}(\tilde{x}_1, \dots, \tilde{x}_n).$

Uvedeným rozepsáním je už tvar výsledku celkové chyby snáze odvoditelný. Primární chybu lze odhadnout následujícím způsobem.



**Věta 2.1**

Nechť existuje množina  $U = \{x_i : |x_i - \tilde{x}_i| \leq \alpha_i, i = 1, \dots, n\}$  a nechť funkce  $f(x_1, \dots, x_n)$  je spojitě diferencovatelná na  $U$ . Pak

$$|f(x_1, \dots, x_n) - f(\tilde{x}_1, \dots, \tilde{x}_n)| \leq \sum_{i=1}^n A_i \alpha_i, \quad (2.8)$$

kde

$$A_i = \sup_U \left| \frac{\partial f}{\partial x_i}(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n) \right|.$$

Odhad chyby lze odvodit například z Taylorova rozvoje funkce  $f(\tilde{x})$  okolo bodu  $x$ .

$$f(\tilde{\mathbf{x}}) = f(\mathbf{x}) + \sum_{i=1}^n \Delta x_i \frac{\partial f(\mathbf{x})}{\partial x_i} + \sum_{i,j=1}^n \Delta x_i \Delta x_j \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} + \dots$$

Při zanedbání součinů chyb  $\Delta x_i \Delta x_j$  platí pro odhad primární chyby

$$|f(x_1, \dots, x_n) - f(\tilde{x}_1, \dots, \tilde{x}_n)| \doteq \left| \sum_{i=1}^n \Delta x_i \frac{\partial f(\mathbf{x})}{\partial x_i} \right| \lesssim \sum_{i=1}^n |\Delta x_i| \left| \frac{\partial f(\mathbf{x})}{\partial x_i} \right|. \quad (2.9)$$

Odhad sekundární chyby lze provést teprve tehdy, až je algoritmus rozepsán do posloupnosti aritmetických operací. [1][7]

**2.3 Příklady**

Část zadání následujících příkladů byla převzata z [1].

**Příklad 2.3**

Najděte primární chybu, která vznikne, jestliže přibližných čísel je použito k výpočtu:

- součtu dvou a obecně  $n$  čísel
- součinu dvou čísel
- podílu dvou čísel
- mocniny čísla, kdy exponent je znám přesně.

**a) Součet  $n$  čísel**

Nejprve bude jednodušší uvažovat pouze chybu při součtu dvou čísel a následně tento získaný závěr zobecnit. Nechť  $a$  a  $b$  jsou správně zaokrouhlená čísla na  $d$  desetinných míst a nechť  $A$  a  $B$  jsou přesná čísla. Potom

$$A = a + \varepsilon_1, \quad B = b + \varepsilon_2, \quad |\varepsilon_1|, |\varepsilon_2| \leq \frac{1}{2}10^{-d}$$

Primární chyba je ve tvaru

$$(A + B) - (a + b) = (a + \varepsilon_1 + b + \varepsilon_2) - (a + b) = \varepsilon_1 + \varepsilon_2.$$

Chyba součtu dvou čísel je algebraickým součtem jejich odchylek. Nyní se ověří, zda tomu bude podobně při součtu obecně  $n$  čísel.

Nechť  $a_1, a_2, \dots, a_n$  je seznam správně zaokrouhlených  $n$  čísel na  $d$  desetinných míst a nechť  $A_1, A_2, \dots, A_n$  je seznam přesných čísel. Potom

$$A_i = a_i + \varepsilon_i, \quad i = 1 \dots n, \quad |\varepsilon_i| \leq \frac{1}{2}10^{-d},$$

kde

$$a_i = A_i - \varepsilon_i.$$

Primární chybu poté vypočteme jako

$$\sum_{i=1}^n A_i - \sum_{i=1}^n a_i = \sum_{i=1}^n A_i - \sum_{i=1}^n (A_i - \varepsilon_i).$$

Sčítání je komutativní operace, proto je možné provést následující úpravu

$$\sum_{i=1}^n A_i - \sum_{i=1}^n (A_i - \varepsilon_i) = \sum_{i=1}^n (A_i - A_i + \varepsilon_i) = \sum_{i=1}^n \varepsilon_i.$$

Primární chyba součtu  $n$  čísel je suma všech odchylek. Je zřejmé, že každé zaokrouhlené číslo přispěje do celkového součtu právě odchylkou od přesné hodnoty, proto je celková chyba algebraickým součtem všech odchylek.

**b) Součin dvou čísel**

Pokud se použije stejné označení jako v předchozí části, tak pro chybu součinu

dvou čísel platí

$$\begin{aligned} A \cdot B - a \cdot b &= (a + \varepsilon_1)(b + \varepsilon_2) - a \cdot b \\ &= a\varepsilon_2 + b\varepsilon_1 - \varepsilon_1\varepsilon_2. \end{aligned}$$

### c) Podíl dvou čísel

Nechť  $u$  a  $v$  jsou správně zaokrouhlená čísla na  $d$  desetinných míst a nechť  $U$  a  $V$  jsou přesná čísla potom platí

$$u = U - \varepsilon, \quad v = V - \gamma, \quad |\varepsilon|, |\gamma| \leq \frac{1}{2}10^{-d}.$$

Primární chyba bude ve tvaru

$$\begin{aligned} f(x_1, x_2) &= \frac{x_1}{x_2}, \quad \frac{\partial f}{\partial x_1} = \frac{1}{x_2}, \quad \frac{\partial f}{\partial x_2} = -\frac{x_1}{x_2^2} \\ \left| \frac{U}{V} - \frac{U - \varepsilon}{V - \gamma} \right| &\leq \left| \frac{\varepsilon}{v^2} - \frac{u\gamma}{v^2} \right|. \end{aligned}$$

### d) Mocnina čísla, kdy exponent je znám přesně

Nechť  $a$  je správně zaokrouhlené číslo na  $d$  desetinných míst a nechť  $A$  a  $k$  jsou přesná čísla, potom platí

$$f(x) = x^k, \quad \frac{df}{dx} = kx^{k-1}, \quad |\varepsilon| \leq \frac{1}{2}10^{-d}.$$

Primární chyba se získá ve tvaru

$$|A^k - a^k| = |A^k - (A - \varepsilon)^k| \leq k(A - \varepsilon)^{k-1}.$$

Zadání následujícího příkladu bylo převzato z [1].

### Příklad 2.4

Nechť je dáno  $n$  čísel  $\tilde{a}_1, \dots, \tilde{a}_n$ , je správně zaokrouhleno na  $d_i$  desetinných míst. Chceme spočítat součet na  $d = \min d_i$  desetinných míst. Ukažte, že je výhodnější nejprve všechna čísla sečíst a výsledek zaokrouhlit na  $d$  míst, než nejprve každé číslo  $\tilde{a}_i$  zaokrouhlit na  $d$  míst a pak sečíst.

Nechť

$$A_i = a_i + \varepsilon_i, \quad i = 1, \dots, n, q \quad |\varepsilon_i| \leq \frac{1}{2}10^{-d_i}.$$

Jestliže každý ze součinů  $a_i$  se zaokrouhlí na  $d$  desetinných míst a výsledek se označí  $(a_i)_z$ , potom

$$(a_i)_z = A_i - \varepsilon_i + \gamma_i, \quad |\gamma_i| \leq \frac{1}{2}10^{\min d_i},$$

a pro primární chybu platí

$$\sum_{i=1}^n A_i - \sum_{i=1}^n (a_i)_z = \sum_{i=1}^n (A_i - A_i + \varepsilon_i - \gamma_i) = \sum_{i=1}^n \varepsilon_i - \sum_{i=1}^n \gamma_i.$$

Nyní se všechna čísla sečtou a celkový výsledek zaokrouhlí. V tomto případě bude primární chyba

$$\sum_{i=1}^n A_i - \left( \sum_{i=1}^n a_i \right)_z = \sum_{i=1}^n \varepsilon_i - \gamma,$$

kde

$$|\gamma| \leq \frac{1}{2}10^{\min d_i}.$$

Porovnáním primárních chyb je vidět, že pokud se čísla nejprve sečtou a až poté zaokrouhlí, bude chyba menší, než při zaokrouhlování každého čísla zvlášť.

### 3 ŘEŠENÍ NELINEÁRNÍCH ROVNIC

V následujících kapitolách budou ukázány některé metody řešení nelineárních rovnic a jejich praktický výpočet pomocí počítače a softwaru **Wolfram Mathematica**. U každé metody je uveden postup jak použít k řešení uvedený software. Mathematica má již vestavěný příkaz **FindRoot**, který je schopen po zadání intervalu, kde se nachází jeden kořen, takové problémy řešit. Proto bude možné vypočítané výsledky porovnávat a ověřovat tak správnost výpočtu. Řešením rovnice se bude rozumět vektor  $\xi$ , pro který platí

$$\mathbf{f}(\xi) = 0, \quad (3.1)$$

kde vektory  $\mathbf{f}$  a  $\xi$  mají stejný počet prvků  $k$

$$\begin{aligned} f_1(x_1, x_2, \dots, x_k) &= 0 \\ f_2(x_1, x_2, \dots, x_k) &= 0 \\ &\vdots \\ f_k(x_1, x_2, \dots, x_k) &= 0. \end{aligned}$$

Pokud je  $k = 1$ , získá se pouze jedna rovnice s jednou neznámou. Za předpokladu, že  $k > 1$ , se získá soustava rovnic a je nutné využít principy a postupy k řešení soustav nelineárních rovnic. V této práci budou ukázány pouze vybrané metody vhodné pro řešení skalární rovnice ( $k = 1$ )

$$f(\xi) = 0, \quad (3.2)$$

kde všechny možné funkce  $f$  budou u každé metody přesně určeny. Například některé z metod mohou řešit pouze vybrané funkce a použití řešení na jiné druhy by nevedlo ke správnému výsledku. Níže uvedené postupy se budou aplikovat na rovnice, které nebudeme schopni vyřešit analytickou metodou (to je jeden z důvodů vzniku všech numerických metod). Na začátku výpočtu pomocí vybrané numerické metody bude nutné kromě řešené rovnice uvést také interval, kde se řešení nachází, nebo blízký počáteční odhad. Metody vyžadující interval s jedním kořenem zaručeně konvergují, ale rychlost, s jakou se přibližuje postup ke správnému řešení, bývá menší, než u metod založených na jednom blízkém počátečním odhadu. Tyto typy metod ovšem nemají konvergenci vždy zaručenou. [1] [4] [5] [6] [7]

Následující příklady budou nejčastěji zadány jako úloha, kde úkolem je najít reálné řešení rovnice  $f(\xi) = 0$ , kde  $f$  je na intervalu  $[a, b]$  spojitá.

**Definice 3.1**

Buď  $x_0 \in \mathbb{R}$ . Řekneme, že funkce  $f$  je v bodě  $x_0$  spojitá, jestliže

$$\lim_{x \rightarrow x_0} f(x) = f(x_0).$$

[8]

**Definice 3.2**

Buď  $f$  funkce,  $I \subseteq D(f)$  interval. Řekneme, že  $f$  je spojitá na intervalu  $I$ , jestliže je spojitá v každém vnitřním bodě intervalu  $I$  a patří-li levý(pravý) koncový bod do  $I$ , je v něm spojitá zprava (zleva). Píšeme  $f \in C(I)$  a je-li  $I = [a, b]$ , také  $f \in C[a, b]$ . [8]

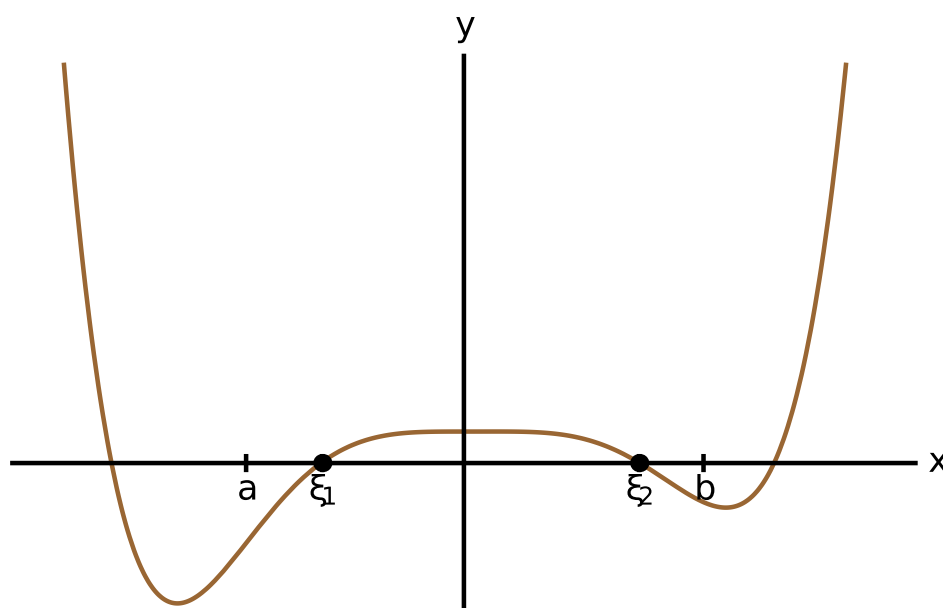
Předpoklad spojitosti je velmi důležitý, neboť umožňuje rozhodnout, zda uvedená funkce má na definovaném intervalu řešení. Například bude stačit, když funkční hodnoty v krajních bodech intervalu budou mít opačná znaménka. Potom je zřejmé, že musí existovat bod, ve kterém je funkční hodnota nulová (tj. hledané řešení rovnice). Přesněji to popisuje následující věta, která je uvedena bez důkazu. Ten lze nalézt například v [8].

**Věta 3.1 (Bolzanova)**

Nechť  $f \in C[a, b]$ . Pak  $f$  nabývá všech hodnot mezi svou největší a nejmenší hodnotou. [8]

**Důsledek:** Pokud  $f(a)f(b) < 0$ , pak existuje bod  $\xi \in (a, b)$  tak, že  $f(\xi) = 0$ .

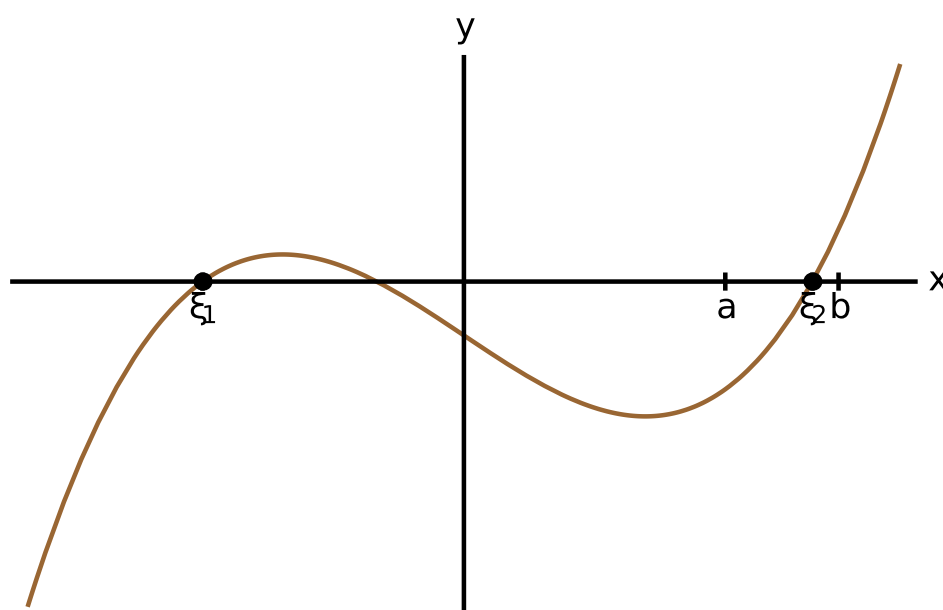
Tato podmínka není nutná. Tento důsledek neříká nic o tom, zda existuje nebo neexistuje kořen za předpokladu, že obě funkční hodnoty v krajních bodech budou mít stejné znaménko [ $f(a)f(b) > 0$ ]. Na obrázku 3.1 je zakreslen graf, a v něm vyznačené krajní body zvoleného intervalu  $[a, b]$ . Na takto vybraném intervalu může mít funkce řešení, i když nemá v krajních bodech intervalu opačná znaménka.



Obr. 3.1 Ukázka nevhodně zvoleného intervalu

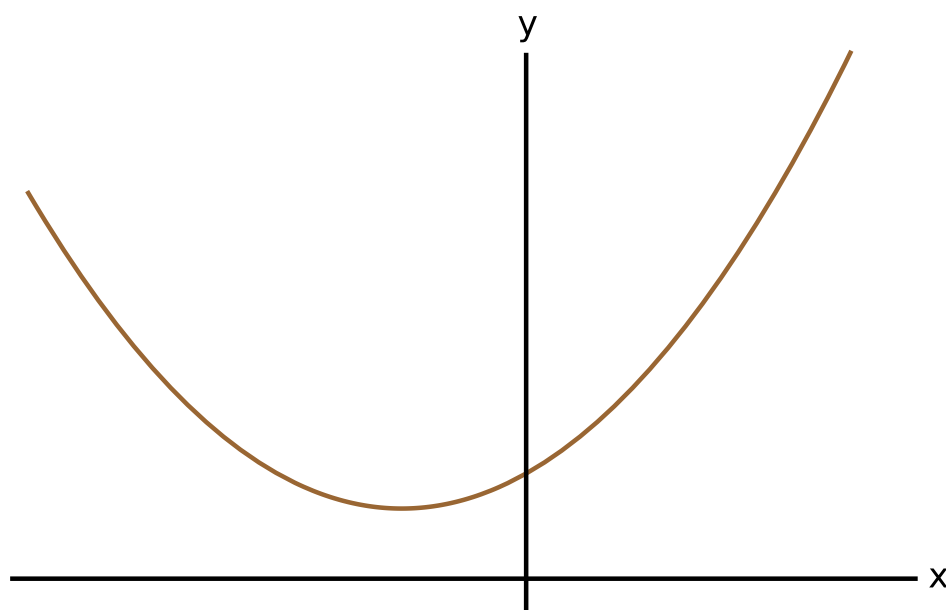
Pro numerické metody jako je např. metoda bisekce (půlení intervalů) a metoda regula falsi (viz kapitoly 4 a 5) se na počátku řešení objeví první problém, a to nalezení vhodného intervalu, ve kterém se bude nacházet právě jeden kořen. Na obrázcích níže jsou graficky znázorněny dobré, ale i špatné volby intervalu.

Pokud budeme v případě grafu na obrázku 3.1 zmenšovat interval, tak nakonec opravdu nastane situace, kdy v krajních bodech budou funkční hodnoty opačného znaménka. Vhodně zvolený interval pro řešení nelineárních rovnic pomocí metody vyžadující k výpočtu počáteční interval s jedním kořenem by byl například následující.



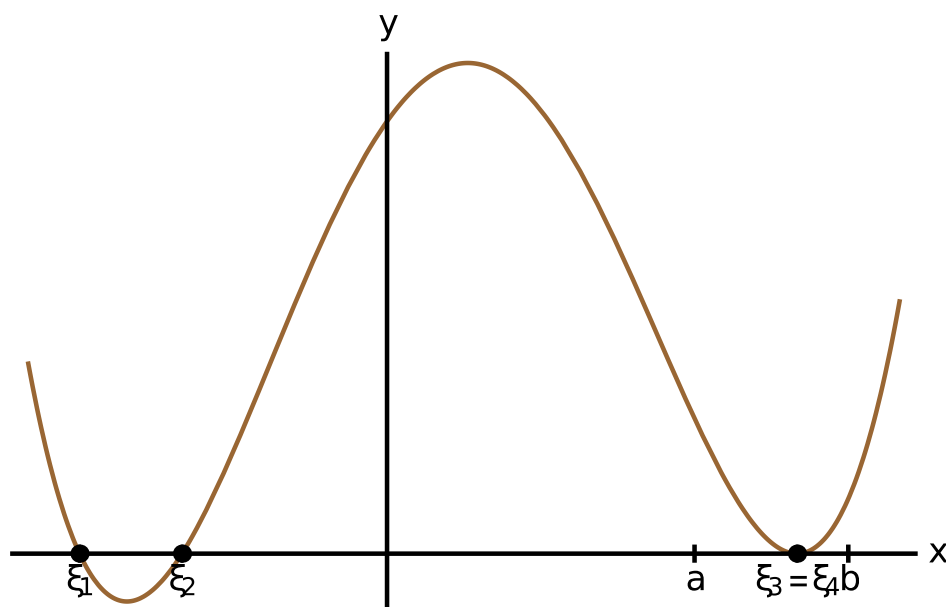
Obr. 3.2 Ukázka vhodně zvoleného intervalu

Níže je znázorněna zřejmě ihned pochopitelná situace. Má smysl se zabývat funkcemi, které protínají osu  $x$ , což pro tu na následujícím obrázku neplatí.



Obr. 3.3 Funkce neprotíná osu  $x$

Nicméně existuje problém, který zmenšením intervalu nevyřešíme, používají-li se metody založené na opačných znaménkách v krajních bodech intervalu (například metoda bisekce). Situace je znázorněna na obrázku 3.4.

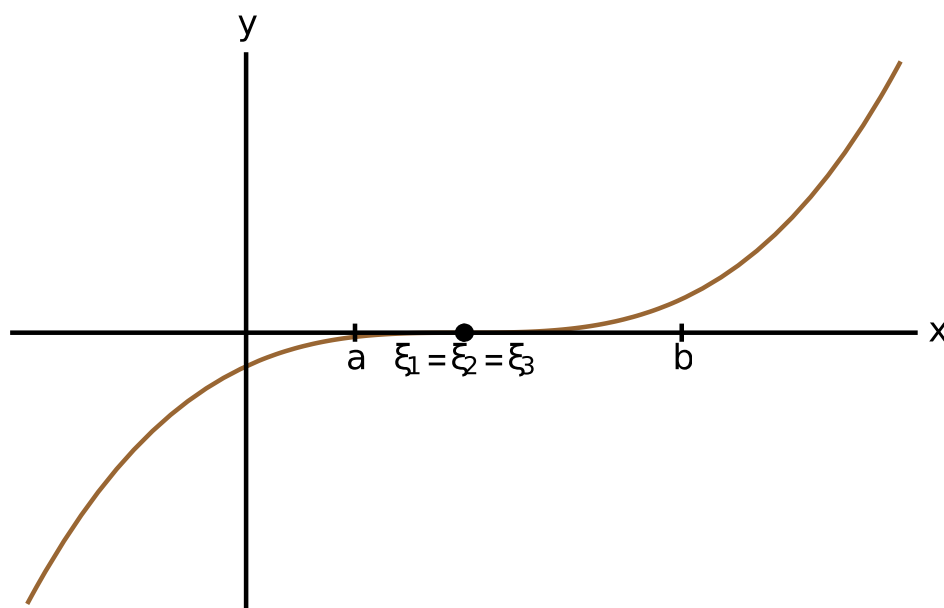


Obr. 3.4 Problém kořenů sudé násobnosti

Konkrétně nalezení kořenu vpravo, tedy obecně kořeny sudé násobnosti. Je opravdu potřeba myslet na to, že volba intervalů je pro takovéto metody klíčová.



Na druhou stranu nalezení kořenů liché násobnosti není pro metody založené na opačných znaménkách v krajních bodech intervalu problémem. [5]



Obr. 3.5 Kořen liché násobnosti

### 3.1 Hledání kořenů

Postup nalezení kořene rovnice (3.2) lze většinou rozdělit do dvou částí

- separace kořenů
- zpřesnění výsledku.

#### 3.1.1 Separace kořenů

Nalezení intervalů, ve kterém se nachází právě jeden kořen, se nazývá separace kořenů. Bohužel nelze ji obecně algoritmizovat a většina postupů bude převážně založena na jedné (popř. kombinaci více) z uvedených metod: [5]

- grafické zobrazení funkce
- postupné hledání
- předchozí zkušenosti s daným nebo podobným problémem
- nahrazení problému zjednodušeným a přibližným
- využití poznatků z předchozí části výpočtu za předpokladu, že problém má více částí.

V příkladech uvedených výše se pohledem na graf dají ihned vybrat vhodné intervaly pro nalezení kořenů. Může ovšem nastat situace, kdy vykreslení funkce v důsledku diskrétní volby hodnot na definičním oboru vede k tomu, že některé z kořenů zůstanou přehlédnuty. Zvětšením bodů (zpřesněním) při vykreslování grafu se situace jistě zlepší, ale nelze to považovat za obecné řešení. Implementovat program, který by řešil kromě výpočtu i separaci kořenů, lze tedy zajistit pouze ve speciálních případech (např. u polynomů).

Další z postupů, pokud není možné funkci jednoduše vykreslit (například při absenci počítače), je velmi podobná metoda založená na zvolení pevného kroku a výpočtu několika funkčních hodnot na zvoleném intervalu. Z vypočítaných funkčních hodnot se zjistí (na základě znaménka) intervaly, ve kterých leží kořen. Opět pro ni platí stejné omezení jako pro grafické zobrazení. V důsledku konečné délky kroku může nastat situace, že dojde k přehlédnutí některých kořenů. [5]

### 3.1.2 Zpřesnění výsledků

Jakmile se nalezne vhodný interval pro řešení, tak můžeme využít následujících postupů

- pokus, omyl
- metody se zadáním intervalu
- metody se zadáním přibližného řešení.

První uvedený postup „pokus, omyl“ jednoduše spočívá v hádání hodnoty  $\xi$  a vypočtení funkční hodnoty  $f(\xi)$ . Pokud je vypočtená funkční hodnota blízká nule, je možné skončit a označit takové  $\xi$  jako řešení. Pokud ne, vybere se nový odhad a celý postup se opakuje. Hledání kořene tímto způsobem je ve velké většině nesmysl, protože existují přesné postupy, jak nalézt co nejrychleji správné řešení.

Jedním z nich jsou metody, jejichž vstupem je interval (získaný postupy uvedenými v části separace kořenů), na kterém se nachází jeden kořen. Pokud se na zadaném intervalu nachází právě jeden kořen, tak tyto metody konvergují k řešení **vždy**, ovšem rychlost konvergence může být velmi pomalá. Do této skupiny se řadí například metoda půlení intervalů a regula falsi.

Dalším typem jsou metody, jejichž vstupem je počáteční odhad řešení (získaný postupy uvedenými v části separace kořenů). Nejsou tak robustní jako předchozí, protože při nevhodném (a přesto velmi blízkém odhadu) se může stát, že metody divergují. Ovšem pokud je odhad zvolen správně, rychlost konvergence bývá zpravidla větší. Do této skupiny se řadí např. metoda prosté iterace, Newtonova metoda a metoda sečen.[5]

#### 4 METODA PŮLENÍ INTERVALŮ

První metodou k řešení nelineárních numerických rovnic, která zde bude uvedena, je **metoda půlení intervalů**, někdy nazývaná také jako **metoda bisekce**. Tento způsob řešení nelineárních rovnic je založen na principu opačného znaménka v krajních bodech intervalu, na kterém se nachází hledaný kořen, a podmínce, že funkce je na uvedeném intervalu spojitá. Dále se předpokládá, že na zvoleném intervalu leží pouze jeden kořen (byl odvozen například jedním z postupů uvedených v části separace kořenů), jehož hodnotu známe velmi přibližně a chceme ji zpřesnit. Přesnější postup této metody je následující: [1]

- Zvolí se vhodný interval  $I = [a, b]$ , přičemž pro funkci platí  $f \in C[a, b]$  (použito stejné označení jako v definici spojitosti funkce).
- Vypočte se bod  $s$ , pro který platí  $s = \frac{a+b}{2}$  (odtud název metoda půlení intervalů).
- Interval  $I$  se rozdělil na dva  $[a, s]$  a  $[s, b]$ . Za předpokladu, že se uvnitř intervalu nachází pouze jeden kořen, musí platit  $[f(a)f(s) \leq 0 \wedge f(s)f(b) \geq 0]$  nebo  $[f(a)f(s) \geq 0 \wedge f(s)f(b) \leq 0]$ . Žádná jiná situace nemůže nastat.
- Za předpokladu, že  $f(s) = 0$ , výpočet je u konce a řešením je právě  $s$ .
- V opačném případě se vybere interval s opačnými znaménky a celý postup se opakuje, dokud nebude dosaženo požadované přesnosti nebo dokud nebude  $f(s) = 0$ .

Je zřejmé, že pro hodnoty  $a_i$  (nejmenší bod vybraného intervalu  $I_i$ ) a  $b_i$  (největší bod vybraného intervalu  $I_i$ ) musí platit

$$a_0 \leq a_1 \leq \dots \leq a_n \leq \xi \leq b_n \leq \dots \leq b_1 \leq b_0. \quad (4.1)$$

Protože vybíráme vždy za  $a_{i+1}$  buď  $a_i$  nebo  $s_i$  (za  $b_{i+1}$  buď  $s_i$  nebo  $b_i$ ), potom posloupnost  $\{a_n\}_{n=1}^{\infty}$  je neklesající a posloupnost  $\{b_n\}_{n=1}^{\infty}$  je nerostoucí. Pro délku prvního intervalu po půlení platí

$$a_1 - b_1 = \frac{a_0 - b_0}{2},$$

pro délku druhého

$$a_2 - b_2 = \frac{a_0 - b_0}{4},$$

obecně pro  $n$ -tý interval platí

$$a_n - b_n = \frac{a_0 - b_0}{2^n}. \quad (4.2)$$

Opakovaným půlením intervalu jeho délka konverguje k nule. Obě posloupnosti jsou tedy omezené. Supremum pro množinu všech prvků posloupnosti  $\{a_n\}_{n=1}^{\infty}$  je právě  $\xi$ . Analogicky infimum pro množinu všech prvků posloupnosti  $\{b_n\}_{n=1}^{\infty}$  je rovněž  $\xi$ . Podle následující věty mají takovéto posloupnosti vlastní limitu.

**Věta 4.1**

Každá neklesající shora ohraničená posloupnost  $\{a_n\}_{n=1}^{\infty}$  má vlastní limitu a platí

$$\lim_{n \rightarrow \infty} a_n = \sup\{a_1, a_2, \dots\}.$$

Podobně každá nerostoucí zdola ohraničená posloupnost  $\{b_n\}_{n=1}^{\infty}$  má vlastní limitu. Přitom platí

$$\lim_{n \rightarrow \infty} b_n = \inf\{b_1, b_2, \dots\}.$$

[8]

Na základě této věty je možné psát

$$\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n = \xi. \quad (4.3)$$

Poslední věc, kterou zbývá dokázat, je, že  $\xi$  je kořenem rovnice (přesněji, že  $f(\xi) = 0$ ). Na základě podmínky výběru intervalů platí

$$f(a_n)f(b_n) \leq 0 \Rightarrow f(\xi)f(\xi) \leq 0 \Rightarrow [f(\xi)]^2 \leq 0. \quad (4.4)$$

V oboru reálných čísel musí být druhá mocnina libovolného čísla kladná nebo rovna nule. Potom z rovnice jednoznačně plyne, že

$$f(\xi) = 0$$

a takovéto  $\xi$  je právě hledaným kořenem.

Nyní pomocí výše uvedených poznatků je možné sestavit větu pro řešení numerických rovnic metodou půlení intervalů.

**Věta 4.2**

Nechť  $f \in C[a, b]$ ,  $f(a)f(b) < 0$  a necht'  $f$  má v intervalu  $[a, b]$  jediný kořen  $\xi$ . Pak metoda bisekce generuje posloupnost  $s_n = \frac{a_n + b_n}{2}$ ,  $n = 0, 1, 2, \dots$ , která konverguje ke kořenu  $\xi$  a aproximuje ho podle vztahu

$$|s_n - \xi| \leq \frac{b - a}{2^{n+1}}. \quad (4.5)$$

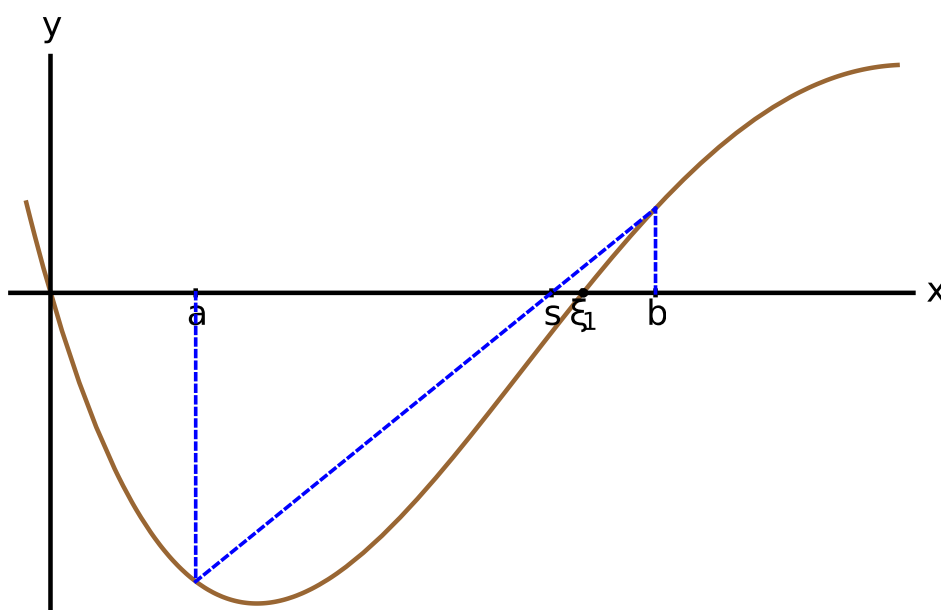
[1]

**Poznámka:** Jelikož  $s_n, \xi \in [a_n, b_n]$  a  $s_n$  se navíc nachází uprostřed tohoto intervalu, musí absolutní hodnota jejich rozdílu být menší nebo rovna velikosti poloviny celého intervalu. Délka intervalu  $|a_n - b_n|$  je rovna  $\frac{a - b}{2^n}$  a polovina tohoto intervalu pak odpovídá pravé straně nerovnice (4.5).

## 5 METODA REGULA FALSI

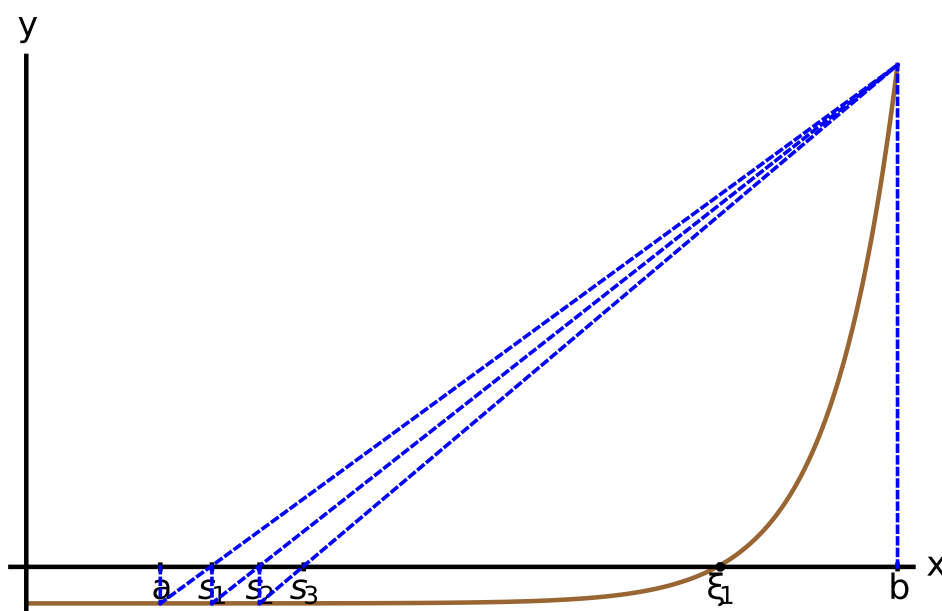
Jak se ukázalo v předchozí kapitole, metoda půlení intervalů nemůže při řešení spojitě funkce na intervalu, kde se nachází jeden kořen, nikdy selhat. Její konvergence je ovšem poměrně pomalá. Jisté zlepšení v některých případech může představovat metoda nazývaná **regula falsi**. Princip výpočtu je podobný s metodou bisekce, až na volbu intervalů pro následující kroky.

Zamyslíme-li se nad krajními body intervalu, kdy jeden z nich je vždy uprostřed předcházejícího, zjistíme, že tato volba nemusí být pokaždé nejlepší. Princip výběru nového intervalu je znázorněn na obrázku níže.



Obr. 5.1 Ilustrace metody regula falsi

V určitých případech je lepší, když je interval zvolen tak, aby jeho krajní body byly co nejblíže řešení rovnice, což právě zvolením středu ve většině případů nemusí být. Na obrázku je absolutní hodnota funkční hodnoty v bodě  $a$  mnohem větší než funkční hodnota v bodě  $b$ . Lze tedy předpokládat, že kořen bude ležet spíše blíže k bodu  $b$  než k  $a$ . A tomu se přizpůsobuje volba intervalů v této metodě. Ty se nerozdělí na polovinu, ale v poměru funkčních hodnot  $f(a)/f(b)$ . Jinými slovy, princip volby je založen na tom, že kořen bude ležet vždy blíže ke krajnímu bodu, jehož absolutní hodnota funkční hodnoty je menší. Uvedená úvaha ale není pokaždé pravdivá. Jeden z příkladů je znázorněn na obrázku 5.2.



Obr. 5.2 Pomalá konvergence metody regula falsi z důvodu velké nelinearity funkce

Použití metody regula falsi je vhodné pro funkce, u kterých se předpokládá na hledaném intervalu průběh blízký lineárnímu. V této situaci by se zvolení poloviny intervalu ukázalo jako vhodnější. Absolutní hodnota funkční hodnoty je v bodě  $a$  výrazně menší než v bodě  $b$ , ovšem kořen leží blíže k bodu  $b$ , což jde ruku v ruce s předpokladem linearitu funkce. Jak už bylo řečeno výše, zbytek výpočtu je velmi podobný s metodou půlení intervalů. [2]

Při volbě nového intervalu se tedy bude uvažovat, že funkce je na něm přibližně lineární. Do nového kroku vybereme bod  $s$ , který je průsečíkem přímky spojující  $f(a)$  a  $f(b)$  s osou  $x$ . Z podobnosti trojúhelníků na obrázku 5.1 dostáváme

$$\begin{aligned}\frac{f(a)}{s-a} &= \frac{f(b)}{s-b}, \\ sf(a) - bf(a) &= sf(b) - af(b), \\ s &= \frac{af(b) - bf(a)}{f(b) - f(a)},\end{aligned}$$

po úpravě

$$s = \frac{f(b)}{f(b) - f(a)}a - \frac{f(a)}{f(b) - f(a)}b. \quad (5.1)$$

Jako nový interval do dalšího kroku bude zvolen  $I_{i+1} = [a, s]$  nebo  $I_{i+1} = [s, b]$  na základě znaménka funkčních hodnot v těchto bodech, stejně jako u metody půlení intervalů. Celý postup se opakuje, dokud není dosažena požadovaná přesnost nebo jiná podmínka vedoucí k ukončení výpočtu. Uvedenou metodu popisuje následující věta.

**Věta 5.1**

Nechť  $f \in C[a, b]$ ,  $f(a)f(b) < 0$  a necht'  $\xi$  je jediný kořen v  $[a, b]$ . Pak posloupnost  $\{x_i\}_{i=0}^{\infty}$  určená metodou regula falsi konverguje pro libovolné počáteční aproximace  $x_0, x_1 \in [a, b]$ ,  $f(x_0)f(x_1) < 0$ , ke kořenu  $\xi \in (a, b)$  funkce  $f$ . [1]

Zatímco u metody půlení intervalů byla chyba výsledku v každém kroku přesně stanovena, tak zde nemáme většinou o chybě přesnou vypovídající hodnotu a nezbyvá nic jiného, než se spolehnout na následující větu o univerzálním odhadu chyby.

**Věta 5.2 (univerzální odhad chyby)**

Nechť funkce  $f$  má na intervalu  $I = (x, \xi)$  spojitou derivaci a necht' existuje číslo  $m > 0$  takové, že  $\forall x \in (x, \xi) : m \leq |f'(x)|$ . Pak

$$|\xi - x| \leq \frac{|f(x_i)|}{m}. \quad (5.2)$$

[2]

**Poznámka:** Číslo  $m$  je možné vyjádřit také jako

$$m = \min_{x \in I(x, \xi)} |f'(x)|.$$

Ne vždy je snadné toto minimum zjistit. Uvedenou větu o univerzálním odhadu chyby je možné použít na výsledek získaný libovolnou numerickou metodou, ovšem obecně v některých případech nemusí být její použití snadné.

Dosud uvedené metody (bisekce a regula falsi) pro nalezení kořene na zadaném intervalu mají zaručenou konvergenci. Při dodržení podmínek spojitosti funkce a vhodné volby intervalu nemůžou tyto dva postupy nikdy selhat, protože kořeny leží vždy uvnitř intervalů, které se postupně zmenšují. Metoda bisekce definuje maximální chybu řešení danou jako polovina aktuálního intervalu. Konvergence je poměrně pomalá (viz příklady v kapitole 9.2). Naproti tomu konvergence metody regula falsi může být výrazně rychlejší, ale slova „může být“ zde mají svá opodstatnění. Tento závěr platí především pro funkce, jejichž průběh je na hledaném intervalu přibližně lineární. Aplikací metody regula falsi na některé jiné typy funkcí (viz například obrázek 5.2) bývá konvergence neporovnatelně pomalejší než u metody bisekce a tento postup se tak stává téměř nepoužitelný. Je nutné zdůraznit, že se zde myslí nepoužitelný ve smyslu neúměrného



množství kroků (volby nových intervalů), ne ve smyslu toho, že by algoritmus nevedl ke správnému výsledku.

Pokud rychlost konvergence ani jednoho z doposud uvedených postupů pro řešení nelineárních rovnic nevyhovuje, je nutné se poohlédnout po dalších metodách numerické matematiky. V následujících kapitolách budou ukázány metody založené na jiném principu. Jejich konvergence záleží na počátečním odhadu, ale bývá zpravidla mnohem rychlejší než konvergence dosud zmíněných metod.

## 6 METODA PROSTÉ ITERACE

Metoda je známá v anglicky knihách jako **fixed-point iteration**, v česky psaných knihách spíše pod názvem **metoda prosté iterace**. Princip je založen na převedení řešení rovnice  $f(x) = 0$  do tvaru  $x = g(x)$ , což lze vždy provést zvolením  $g(x)$  jako

$$g(x) = x - f(x), \quad (6.1)$$

a hledání takového řešení, pro které platí  $x = \xi$ , tedy  $\xi = g(\xi)$ , což je ekvivalentní s  $f(\xi) = 0$ . Bod  $\xi$  se označuje jako **pevný bod** funkce  $g$ . Geometricky to znamená hledat průsečík přímky  $y = x$  s křivkou  $y = g(x)$ . Na začátku výpočtu se volí počáteční odhad řešení (tedy jedno číslo). To je zásadní rozdíl oproti doposud uvedeným metodám (metoda půlení intervalů a metoda regula falsi), kde vstupem byl vždy počáteční interval neboli dvojice čísel. A získáme posloupnost, kde následující prvek závisí na předchozím podle vztahu

$$x_{i+1} = g(x_i). \quad (6.2)$$

Funkce  $g$  se nazývá **iterační funkcí** a vztah (6.2) **iterační metodou** nebo také **metodou prosté iterace**. Taková funkce  $g$  musí být pochopitelně vhodně zvolena, často se používá pojem, že se funkce zobrazuje **do sebe**. To znamená, že rozdíl dvou vzorů je větší než rozdíl obrazů.[1] [2] [4]

Metoda prosté iterace patří mezi tzv. **jednokrokové metody**, neboť výpočet  $x_{i+1}$  závisí pouze na předchozím prvku  $x_i$ . Může se stát, že by mohl výpočet záviset také například na  $x_{i-1}$ , popřípadě na dalších předchozích hodnotách. Takové metody se nenazývají jednokrokovými, ale **vícekrokovými**

$$x_{i+1} = g(x_i, x_{i-1}, x_{i-2} \dots, x_{k-j+1}), \quad j \geq 2. \quad (6.3)$$

Například závisí-li výpočet  $x_{i+1}$  na dvou předchozích hodnotách, nazývají se tyto metody jako **dvoukrokové** atd. [1]

Iterační metody potřebují na začátku výpočtu zvolit odhad hledaného řešení  $x_0$ . Bohužel ještě to neznamena, že pokud je odhad „dostatečně blízko“ ke kořenu, bude algoritmus řešení konvergovat. Rovněž postup pro nalezení vhodného  $x_0$  není jednoduché algoritmizovat (viz kapitola 2). Většinou se musí použít prostředky analytické matematiky k nalezení přibližného odhadu, který se použije jako počáteční hodnota do numerického algoritmu a nebo je už přibližný odhad znám např. z předchozích měření, výpočtů apod. Je možné rovněž využít grafických metod a nahradit (proložit) funkci ve zkoumaném intervalu přímkou a provést výpočet odhadu. V neposlední řadě je možné použít pro řešení numerických rovnic jinou metodu, která nemá tak rychlou konver-

genci, ale nevyžaduje určení  $x_0$  na větší počet desetinných míst. Princip této metody spočívá v provedení odhadu, který bude vstupní hodnotou do iteračních algoritmů, které mohou konvergovat mnohem rychleji. [5]

Následující věta shrnuje poznatky uvedené v odstavcích výše a definuje podmínky, za kterých postup výpočtu vede k řešení.

### Věta 6.1

Nechť  $g(x)$  splňuje Lipschitzovu podmínku

$$|g(x) - g(x')| \leq \lambda |x - x'|, \quad (6.4)$$

pro všechny hodnoty  $x, x'$  v uzavřeném intervalu  $I = [x_0 - \rho, x_0 + \rho], \rho > 0$ , kde pro tzv. Lipschitzovou konstantu  $\lambda$  platí

$$0 \leq \lambda < 1. \quad (6.5)$$

Nechť je nyní  $x_0$  zvoleno tak, že

$$|x_0 - g(x_0)| \leq (1 - \lambda)\rho. \quad (6.6)$$

Potom platí následující tvrzení:

- všechny iterace  $x_i$  definované podle vztahu (6.2) leží v intervalu  $I$

$$x_0 - \rho \leq x_i \leq x_0 + \rho$$

- iterace konvergují ke stejnému bodu  $\xi$  (existence)

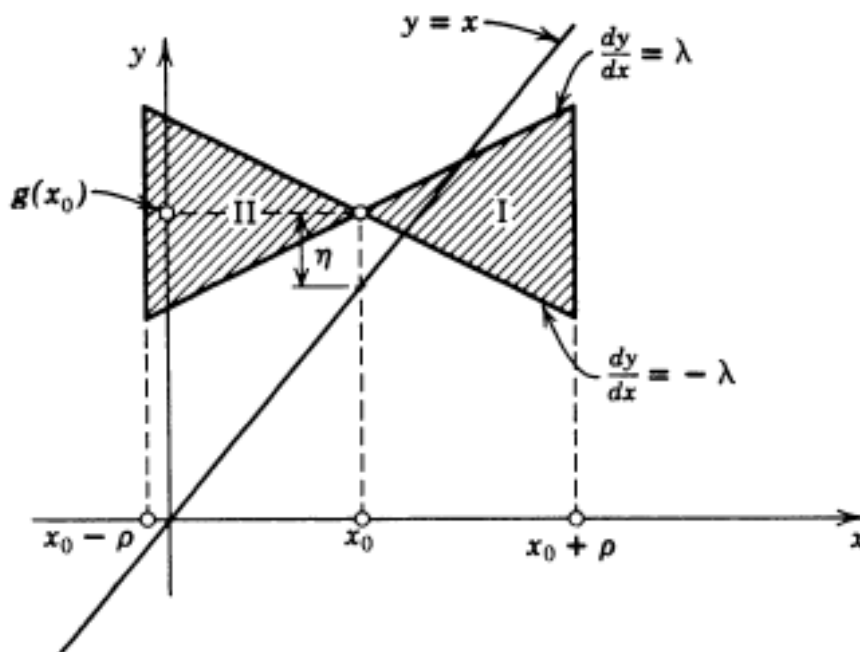
$$\lim_{i \rightarrow \infty} x_i = \xi, \text{ neboli } |x_i - \xi| \leq \lambda \rho$$

- $\xi$  je jediným kořenem na intervalu (jednoznačnost)  $I = [x_0 - \rho, x_0 + \rho]$ .

[4]

**Důsledek:** Jestliže platí  $|g(x)| \leq \lambda < 1$  pro  $|x - x_0| \leq \rho$  a pro volbu  $x_0$  je splněna rovnice (6.6), potom všechny závěry vyplývající z věty 6.1 jsou platné.

Důkazy jednotlivých tvrzení lze nalézt v [4]. Geometrická interpretace této věty je znázorněna na obrázku 6.1, aby bylo možné si lépe představit skutečnosti, které z ní vyplývají.



Obr. 6.1 Grafické znázornění podmínek konvergence metody prosté iterace [4]

Obrázek zobrazuje případ, kdy  $g(x_0) - x_0 = \eta > 0$  a trojúhelníky I a II jsou ohraničeny přímkami se směrnici  $\pm\lambda$ . Uvnitř oblastí I a II se na intervalu  $[x_0 - \rho, x_0 + \rho]$  nacházejí funkční hodnoty  $g(x)$ . Z obrázku lze vyčíst:

- Pokud je  $\lambda \geq 1$ , tak přímka  $y = x$  neprotne horní hranici oblasti I.
- Pokud je  $\lambda < 1$  a zároveň  $\eta > (1 - \lambda)\rho$ , tak přímka  $y = x$  neprotne horní hranici oblasti I, a proto nemusí protnout případnou funkci  $g(x)$  na intervalu  $[x_0 - \rho, x_0 + \rho]$ .

[4]

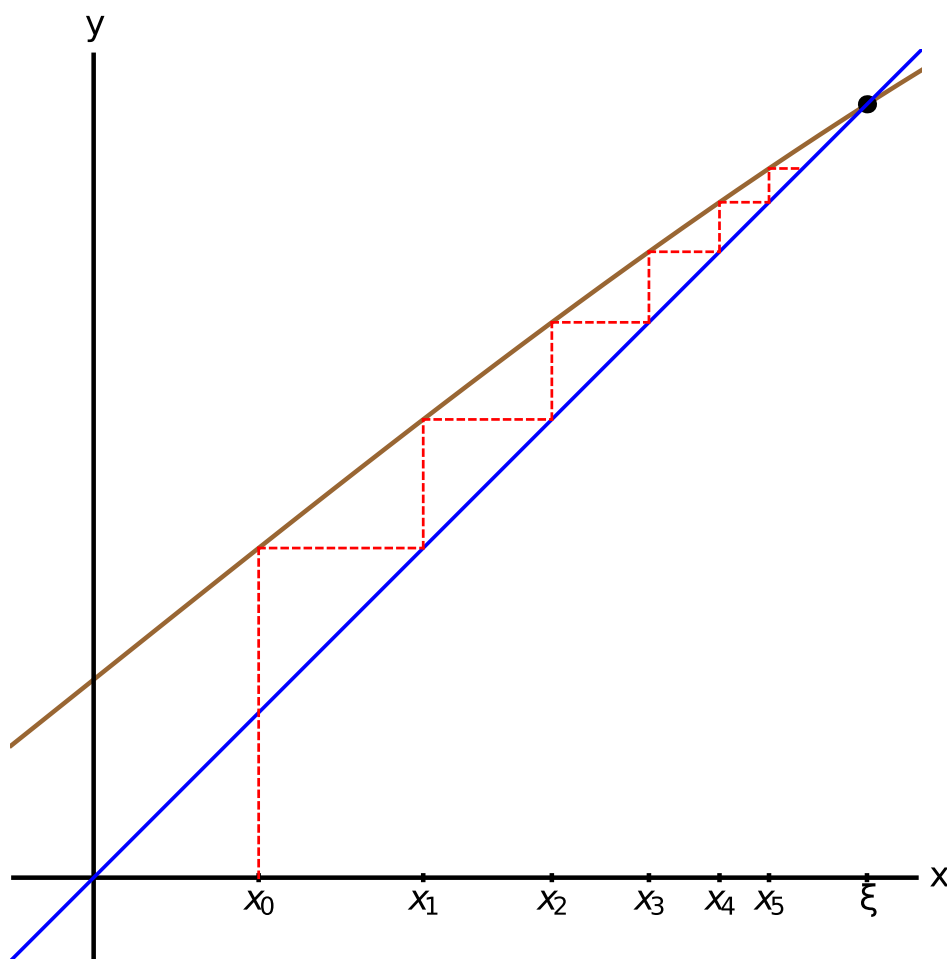
Proč je na obrázku znázorněna přímka  $y = x$  a dvě trojúhelníkové oblasti? Na začátku kapitoly bylo řečeno, že se problém  $f(x) = 0$  převede na ekvivalentní rovnici z hlediska výsledného řešení  $f(g(x)) = 0$  neboli  $g(x) = x$ . Tedy geometricky je to hledání průsečíku  $g(x)$  s osou  $x$ . Trojúhelníkové oblasti vyznačují prostor, ve kterých by měla ležet funkce  $g$ .

Proč jsou to právě takové oblasti? Funkce  $g(x)$  musí na definovaném intervalu splňovat Lipschitzovu podmínku, tj. rozdíl funkčních hodnot v krajních bodech je menší než délka tohoto intervalu. Zjednodušeně řečeno, přímka spojující krajní body intervalu  $[x_0 - \rho, g(x_0 - \rho)]$  a  $[x_0 + \rho, g(x_0 + \rho)]$  musí mít absolutní hodnotu směrnice menší než jedna.

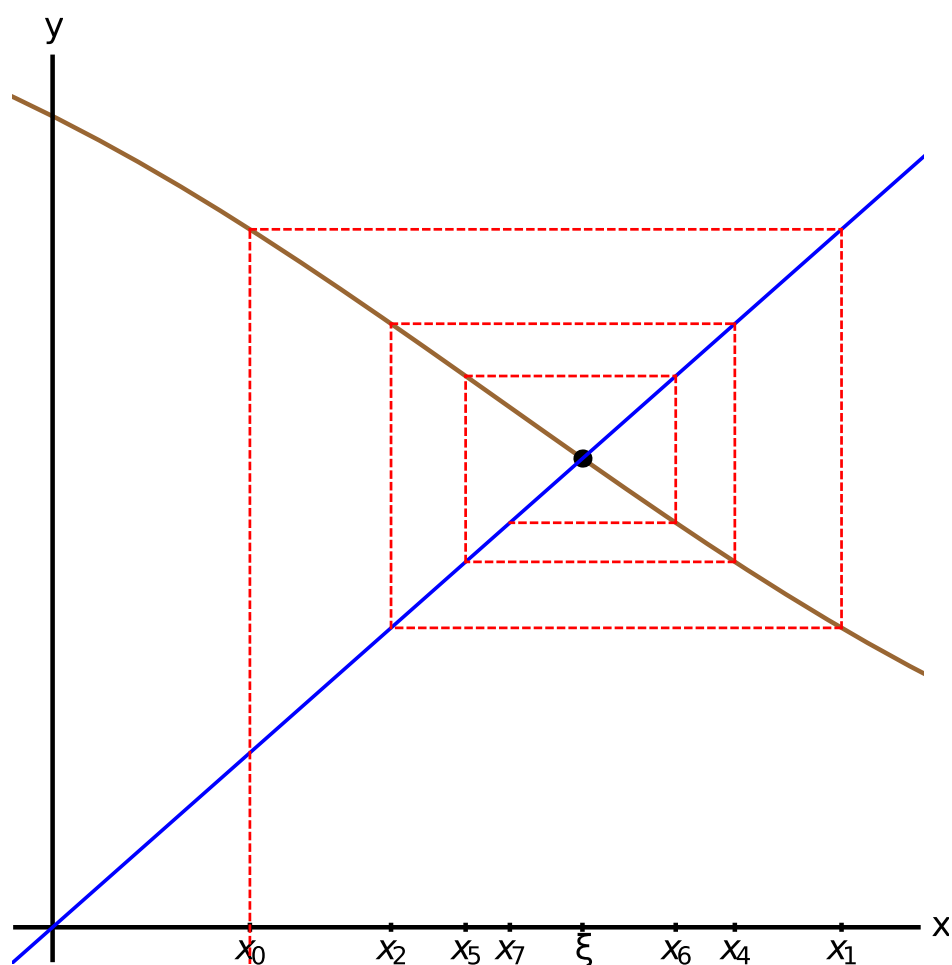
Odkud se vzala podmínka vztahující se na  $\eta$ ? Budeme-li zvětšovat rozdíl  $g(x_0) - x_0$ , nastane situace, kdy přímka  $y = x$  neprotne trojúhelníkovou oblast I. Maximální velikost  $\eta$  je, když přímka prochází pravým spodním vrcholem oblasti I. Odtud i geometricky je možné odvodit, že musí platit  $\eta > (1 - \lambda)\rho$ . Korektně to vyplývá z důkazů, které lze najít v [4].

Jinými slovy, podmínky  $\lambda < 1$  a  $\eta > (1 - \lambda)\rho$  jsou **postačující** k předpokladu existence kořene pro libovolnou Lipschitzovsky spojitou funkci  $g(x)$  splňující podmínky (6.4) a (6.5). [4]

Na grafech níže je znázorněn postup řešení metodou prosté iterace řešení. Modrá křivka na grafech značí funkci  $y = x$ , hnědá je funkce  $g$  a červeně přerušované úsečky ilustrují postupnou volbu nové hodnoty  $x_i$  do dalšího kroku.

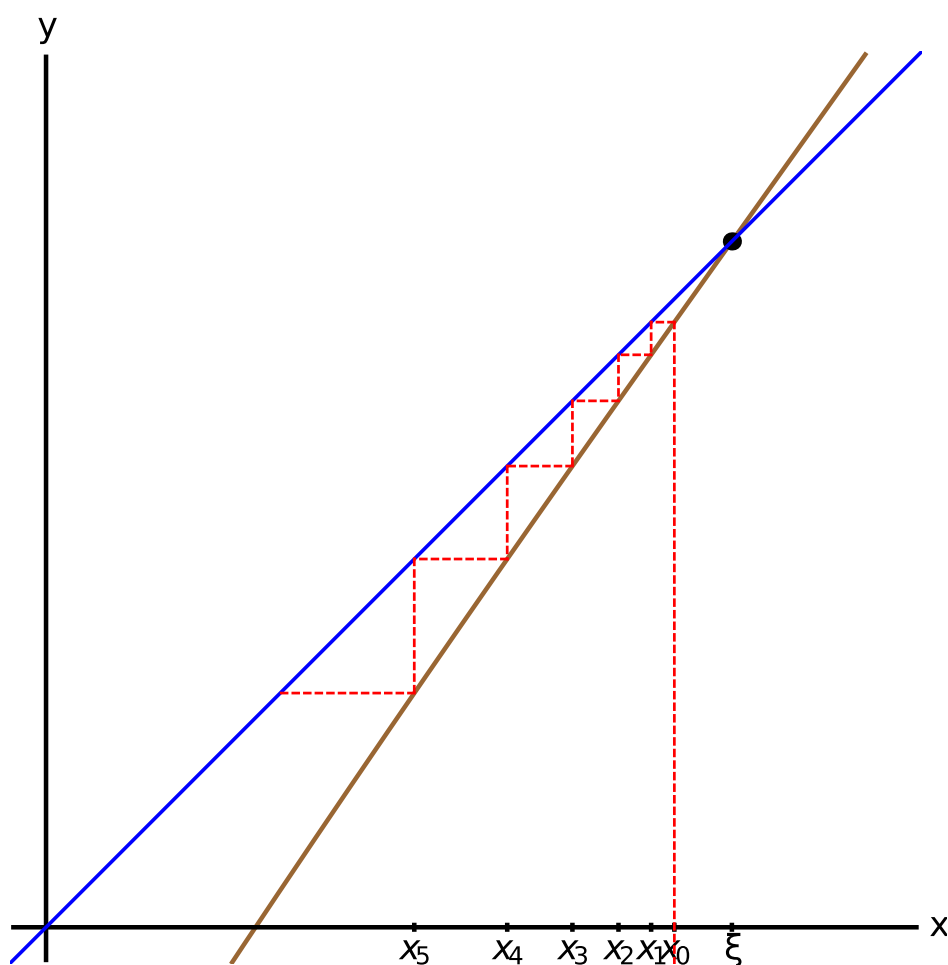


Obr. 6.2 Řešení rovnice iterační metodou (rostoucí funkce)



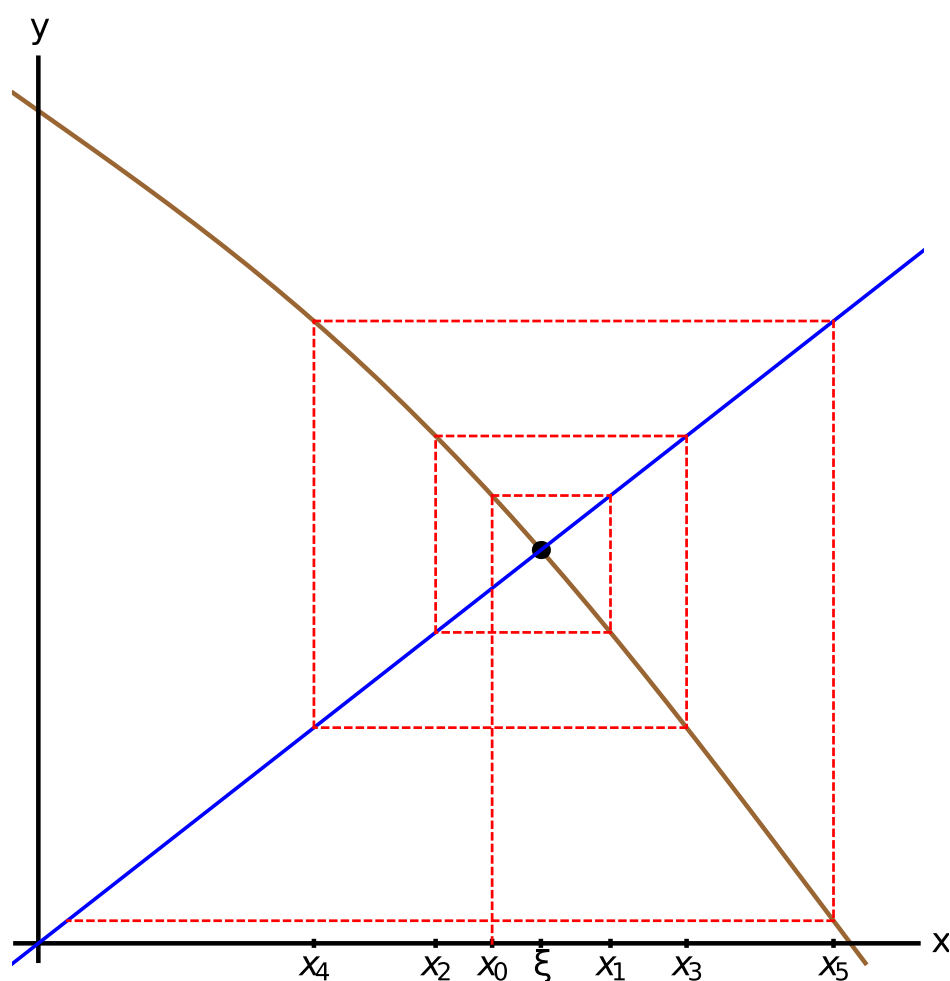
Obr. 6.3 Řešení rovnice iterační metodou (klesající funkce)

Je možné si všimnout, že zatímco u rostoucí funkce na řešeném intervalu se  $x_i$  postupně zvětšuje (nebo zmenšuje, pokud by byl odhad stanoven vpravo od kořene), tak naopak u klesající funkce dochází střídavě k výběru hodnot vpravo a vlevo od průsečíku (řešení). Na dalších dvou obrázcích je znázorněn případ, kdy funkce  $g$  nesplňuje předpoklady uvedené ve větě na začátku kapitoly.



Obr. 6.4 Nesprávně zvolená iterační funkce (rostoucí funkce)

Poslední dva obrázky zachycují nevhodně zvolené iterační funkce. Přestože počáteční odhad  $x_0$  je velmi blízko řešení (mnohem blíže než na obrázcích 6.2 a 6.3), tak metoda (posloupnost) diverguje. Nejen volba počátečního odhadu, ale rovněž správná volba iterační funkce je pro uvedený postup klíčová a rozhoduje i o rychlosti konvergence.



Obr. 6.5 Nesprávně zvolená iterační funkce (klesající funkce)

Následující věta umožňuje určit absolutní chybu výsledku při použití metody prosté iterace.

**Věta 6.2**

Nechť funkce  $g$  splňuje předpoklady věty 10.0, pak pro posloupnost  $\{x_i\}_{i=0}^{\infty}$ ,  $x_0 \in [a, b]$ ,  $x_i = g(x_{i-1})$ , platí

$$|x_i - \xi| \leq \frac{\lambda^i}{1 - \lambda} |x_0 - x_1|, \forall i \geq 1. \quad (6.7)$$

[1] [2]

Nyní zbývá podat návod, jak najít vhodnou iterační funkci. Z rovnice (6.1) plyne, že vhodná iterační funkce vznikne vyjádřením  $x$  z rovnice  $f(x)$ . Pojem vyjádření  $x$  z rovnice není nutné v tomto případě chápat doslovně. Ve většině případů bude stačit



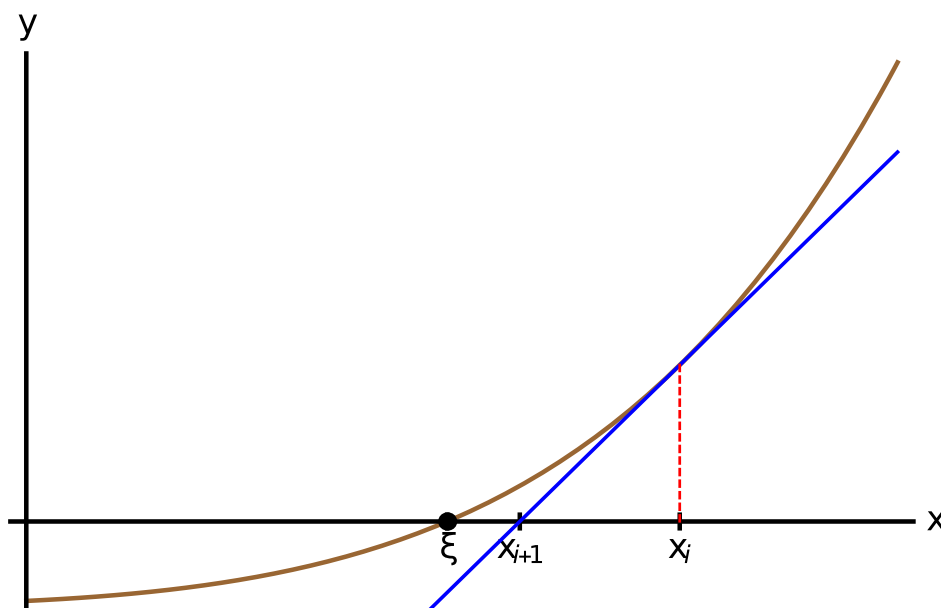
získat výraz, kde na pravé straně je  $x = \dots$ . Což ale někdy nemusí být jednoduše proveditelné. Navíc takto získané iterační funkce  $g$  nemusí vést ke správnému výsledku, neboť musí rovněž platit  $|g(x)| \leq \lambda < 1$ . Viz např. z rovnice  $2x^3 + x - 1 = 0$  se uvedeným postupem získají dvě iterační funkce

$$g_1(x) = \sqrt[3]{\frac{1-x}{2}}$$
$$g_2(x) = 1 - 2x^3.$$

Obě dvě funkce vyhovují rovnici (6.1), nicméně druhá z nich nevede k řešení, protože  $|g(x)| > 1$ . Použití nevhodné iterační funkce (taková, která nevyhovuje uvedeným větám) znázorňují obrázky 6.4 a 6.5.

## 7 NEWTONOVA METODA

**Newtonova metoda** (někdy je nazývána také jako **Newton-Rhapsonova metoda** či **metoda těčen**) je jednou z nejznámějších numerických metod pro řešení nelineárních rovnic. Výhodou je v případě blízkého počátečního odhadu rychlá konvergence. Jediná nepříjemnost je nutnost výpočtu derivace  $f'(x)$ . Ilustrace této metody je znázorněna na obrázku 7.1.



Obr. 7.1 Grafické znázornění Newtonovy metody

Princip metody spočívá ve výpočtu derivace v počátečním bodě (odhadu). Do dalšího kroku se použije průsečík tečny s osou  $x$ . Postup výpočtu je tedy následující:

- volba počátečního odhadu  $x_0$
- výpočet derivace  $f'(x_0)$  a zjištění průsečíku  $x_i$  tečny s osou  $x$
- opakování předchozího kroku s  $x_i$ .

Graf na obrázku 7.1 poslouží jako pomůcka pro odvození vztahu pro volbu odhadu  $x_{i+1}$  do dalšího kroku. Pro rovnici přímky platí

$$y = kx + q$$

kde  $k = f'(x_i)$  a  $q = f(x_i) - kx_i = f(x_i) - f'(x_i)x_i$ . Průsečík s osou  $x$  se odvodí položením  $y = 0$ , tedy

$$0 = f'(x_i)x_{i+1} + f(x_i) - f'(x_i)x_i.$$

Vyjádřením  $x_{i+1}$  se získá vztah pro volbu nového odhadu kořene funkce  $f$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \quad (7.1)$$

Jak už bylo řečeno, na počátku celého výpočtu stojí odhad  $x_0$ , podobně jako tomu bylo u metody prosté iterace. Jaká omezení by mohla platit pro volbu  $x_0$ ? S přihlédnutím k tomu, že derivace určuje směrnici tečny a tato tečna musí protínat osu  $x$ , nemůže pro bod  $x_0$  být  $f'(x_0) = 0$ , tedy tečna rovnoběžná s osou  $x$  (konstantní funkce). V případě implementace algoritmu, kde uživatel bude vyzván k zadání tohoto počátečního odhadu, je nutné nejprve provést výše uvedenou kontrolu a výpočet v případě, že derivace je v  $x_0$  nulová, ukončit. Nezbyývá nic jiného, než zvolit jiný počáteční odhad. Newtonova metoda je speciálním případem metody prosté iterace, právě když

$$g(x) = x - \frac{f(x)}{f'(x)}. \quad (7.2)$$

[1] [2] [3] [6]

#### Věta 7.1

Nechť  $f$  je spojitá funkce na intervalu  $I = [a, b]$  a nechť pro všechna  $x \in I$  existují derivace  $f'(x)$  a  $f''(x)$ . Jestliže se na intervalu  $I$  nachází kořen  $\xi$ , tj. platí  $f(\xi) = 0$  a  $f'(\xi) \neq 0$ , potom existuje  $\delta > 0$  a posloupnost určená rovnicí

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad f'(x_i) \neq 0, \quad k = 0, 1, \dots \quad (7.3)$$

konverguje k  $\xi$  pro libovolné  $x_0 \in [\xi - \delta, \xi + \delta]$ . [1]

Funkce  $g$  splňuje na intervalu  $[\xi - \delta, \xi + \delta]$  předpoklady pro konvergenci posloupnosti určenou metodou prosté iterace. To znamená, že posloupnost určená vztahem (7.3) konverguje pro každou počáteční aproximaci  $x_0 \in [\xi - \delta, \xi + \delta]$  ke kořenu  $\xi$ . Zjednodušeně řečeno, Newtonova metoda konverguje pro každý „blízký“ odhad  $x_0$ . [2]

Odhad chyby pro Newtonovu metodu popisuje následující věta.

**Věta 7.2**

Označíme-li

$$m = \min_{[a,b]} |f'(x)|, \quad M = \max_{[a,b]} |f''(x)|,$$

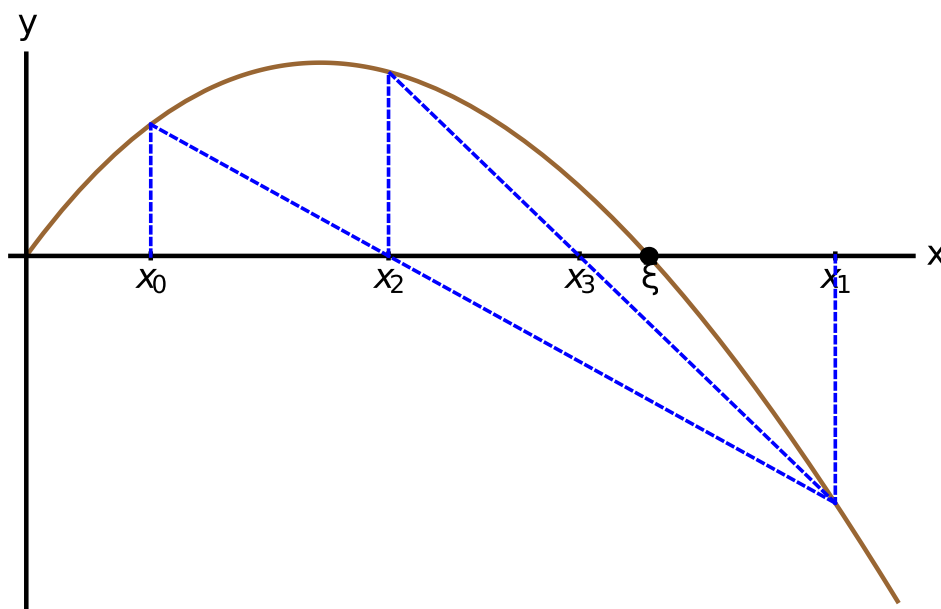
kde  $x_0 \in [a, b]$ ,  $\xi \in [a, b]$  a v intervalu se znaménka  $f'$  a  $f''$  nemění, potom platí

$$|x_{i+1} - \xi| \leq \frac{M}{2m} |x_i - \xi|^2. \quad (7.4)$$

[1] [2] [3]

## 8 METODA SEČEN

Už při vytváření algoritmu pro metodu tečen bylo naznačeno, že výpočet derivace lze zjednodušit a nahradit (při připuštění jisté chyby) výpočtem dvou funkčních hodnot ve dvou velmi blízkých bodech. Za předpokladu, že derivace nebude vypočtena symbolicky, tak se jedná o výpočetně náročnou operaci v porovnání s běžnými operacemi sčítání, násobení apod., neboť je nutné navíc použít opět prostředky numerické matematiky, a celý výpočet se tak výrazně prodlužuje. Mathematica zvládá symbolické derivování, proto by nemělo být znatelné výrazné zpomalení Newtonovy metody právě v důsledku výpočtu derivace. Přesto má smysl se zabývat další metodou nazývanou jako **metoda sečen**, která nevyžaduje k výpočtu znalost první derivace. Principem je spíše podobná metodě regula falsi uvedené v kapitole 5.



Obr. 8.1 Řešení rovnice metodou sečen

Odhad řešení  $x_i$  leží na spojnici dvou předcházejících bodů výpočtu  $[x_{i-2}, f(x_{i-2})]$  a  $[x_{i-1}, f(x_{i-1})]$ . Tím se do jisté míry zamezuje tomu, což bývá pro metodu regula falsi typické, a to, že se v průběhu výpočtu jeden z krajních bodů nemění (viz například obrázek 5.2). Při použití metody sečen se nepřihlíží ke znaménkům v krajních bodech intervalů, ale vždy se použijí dvě předcházející hodnoty, což je právě ten podstatný rozdíl oproti metodě regula falsi. Proto na začátku celého výpočtu potřebujeme dva blízké odhady. Představa dvou bodů mohla být podobná jako potřeba počátečního intervalu u metody bisekce a regula falsi. Opět podobně jako u metody regula falsi z trojúhelníkové podobnosti na obrázku 8.1 se odvodí průsečík s osou  $x$  přímky spojující  $f(x_{i-1})$  a  $f(x_i)$ . Platí

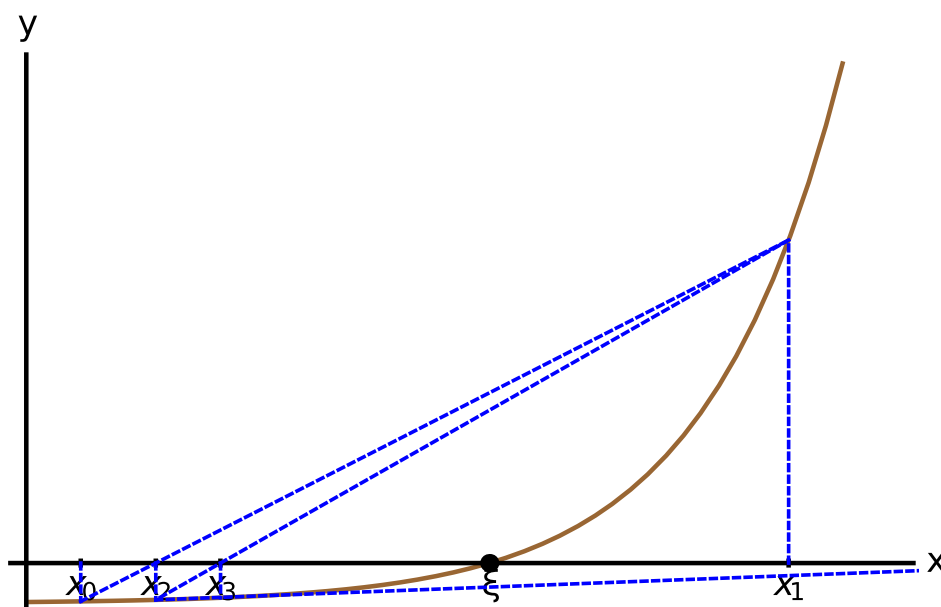
$$\frac{f(x_i)}{x_{i+1} - x_i} = \frac{f(x_{i-1})}{x_{i+1} - x_{i-1}},$$

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

Po úpravě

$$x_{i+1} = \frac{f(x_i)x_{i-1} - f(x_{i-1})x_i}{f(x_i) - f(x_{i-1})}. \quad (8.1)$$

U metody sečen není zaručena její konvergence, neboť neudrží kořen ve zmenšujícím se intervalu, jako je to u metody regula falsi.



Obr. 8.2 Funkce nevhodná pro řešení metodou sečen

Divergence metody sečen lze dosáhnout pouze změnou počátečních odhadů kořene. Opět zbývá uvést, jak zjistit odhad chyby v každém kroku výpočtu. Lze doporučit větu o univerzálnímu odhadu chyby a vztah (5.2), stejně jako u metody regula falsi. [1][2]

## II. PRAKTICKÁ ČÁST

## 9 METODA PŮLENÍ INTERVALŮ

### 9.1 Implementace algoritmu

Jak už bylo naznačeno v úvodu kapitoly o řešení nelineárních rovnic, je nutné zadat na počátku výpočtu interval, ve kterém se nachází právě jeden kořen. Vzhledem k obecně velké složitosti programově implementovat vyhledávání intervalu s jedním kořenem, bude přenechána tato volba uživateli. Jádro algoritmu výpočtu potom může vypadat následovně:

---

```
While[True,
  (* vypocet stredy a funkcnich hodnot v krajnich bodech *)
  fa = fg /. x -> a;
  fb = fg /. x -> b;
  s = (a + b) / 2;
  fs = fg /. x -> s;

  (* vypis mezivysledku *)
  If[progress, Print[N[{it, a, b, s, fs}, wprec]]];

  (* kontrola pocetu platnych cifer *)
  If[Abs[(b - a)/2] <= minReqErr, Return[{x -> N[s, wprec]}]];

  (* kontrola nalezeni presneho reseni *)
  If[fs == 0, Return[{x -> N[s, wprec]}]];

  (* zvoleni intervalu do dalsiho kroku *)
  If[Negative[fa * fs], b = s, a = s];

  (* kontrola zda se na intervalu [a, b] nachazi vice korenu *)
  If[Positive[fa] && Positive[fb],
    Message[NMBisection::badRange];
    Throw[$Failed]
  ];

  (* zastaveni vypoctu po dosazeni maximalniho poctu opakovani *)
  If[++it == maxIt,
    Message[NMBisection::maxIt, it];
    Throw[$Failed]
  ];
];
```

---



Při vytváření algoritmu se naráží na problém, kdy celý výpočet ukončit. Je možné zadat pevný počet kroků a výpočet po  $n$  opakování zastavit s tím, že nebude dosaženo požadované přesnosti. Nebo naopak probíhalo několik výpočtů zbytečně a výsledek je platný na více cifer než bylo požadováno a zbytečně se tím tak prodloužil čas výpočtu.

Lepší je zavést podmínku, která výpočet přeruší, až bude výsledek v uživatelské zadané přesnosti. Ten tedy zadá maximální absolutní chybu výsledku a jakmile je tato podmínka splněna, dojde k ukončení výpočtu. Rovněž bude dobré zajistit ukončení výpočtu po maximálním počtu opakování. Uživatel nemusí tušit, že k výpočtu jeho zvolené přesnosti je nutné neúměrně velké množství kroků a výpočet by tak mohl trvat dlouhou dobu. Je vhodnější mít možnost dvojí kontroly nad probíhajícím výpočtem. Níže jsou vypsána používaná kritéria pro zastavení výpočtu [1] [2] [6]

$$|s_n - \xi| \leq \epsilon, \quad (9.1)$$

$$|f(s_n)| \leq \epsilon, \quad (9.2)$$

$$\left| \frac{s_n - s_{n-1}}{s_n} \right| \leq \epsilon. \quad (9.3)$$

Při vytváření algoritmu se použije první z uvedených podmínek, protože hodnota výrazu na levé straně  $|s_n - \xi|$  je v každém kroku výpočtu známa podle vztahu (4.5). Při podmínce (9.2) dojde k zastavení výpočtu poté, co je funkční hodnota aktuálního řešení menší, než uživatelem definovaná. Není právě moc vhodná, neboť nemáme zaručeno, že se funkce přibližuje k ose  $x$  pouze v blízkosti kořene. Poslední z uvedených tří kritérií pro zastavení výpočtu ukazuje relativní chybu aktuálního řešení od předcházejícího (v minulém kroku). Při praktickém výpočtu se po uživateli požaduje zadat tuto chybu, ale tu uživatel většinou nezná. Co ho nejvíce zajímá, je kolik desetinných míst bude ve výsledku správně (tj. zadá počet desetinných míst). K tomu bude nutné využít poznatky z kapitoly 2.1. Z požadovaného počtu desetinných míst se poté dopočítá přesnost  $\epsilon$ , která se použije jako kritérium k zastavení výpočtu (tedy na základě nerovnosti (9.1)).

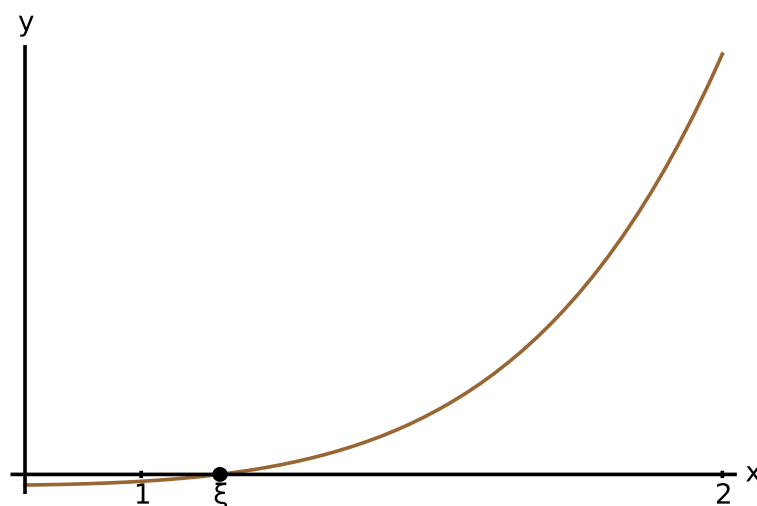
## 9.2 Příklady

Níže jsou uvedeny vybrané příklady na řešení nelineárních rovnic metodou půlení intervalů. Při výpočtu se využívá algoritmus a vytvořená funkce v programu Mathematica, jejíž část byla uvedena v předchozí sekci. Jak používat tuto funkci popisuje poslední kapitola na konci práce.

**Příklad 9.1**

Metodou půlení intervalů (bisekce) nalezněte kořen rovnice  $x^6 - x - 1 = 0$  na intervalu  $I = [1, 2]$  s přesností na 15 desetinných míst.

Daná rovnice má na intervalu  $I$  jeden kořen (viz obrázek).



Obr. 9.1 Funkce  $f(x) = x^6 - x - 1$

S využitím softwaru Mathematica a výše popsaného algoritmu se po zadání příkazu

```
NMBisection[x^6-x-1, {x, 1, 2}, AccuracyGoal -> 15,  
ShowProgress -> True]
```

získá následující tabulka

Tab. 9.1 Postup výpočtu kořene rovnice  $x^6 - x - 1 = 0$   
metodou bisekce

it	a	b	x	f(x)
1	1,0000000000000000	2,0000000000000000	1,5000000000000000	8,890625000000000
2	1,0000000000000000	1,5000000000000000	1,2500000000000000	1,564697265625000
3	1,0000000000000000	1,2500000000000000	1,1250000000000000	-0,097713470458984
4	1,1250000000000000	1,2500000000000000	1,1875000000000000	0,616653025150299
5	1,1250000000000000	1,1875000000000000	1,1562500000000000	0,233268924988806
6	1,1250000000000000	1,1562500000000000	1,1406250000000000	0,061577832108014
...	...	...	...	...
...	...	...	...	...
48	1,1347241384015092	1,1347241384015234	1,1347241384015163	$3,843327568481908 \cdot 10^{-15}$
49	1,1347241384015163	1,1347241384015199	1,1347241384015181	$-1,443117215796558 \cdot 10^{-14}$
50	1,1347241384015181	1,1347241384015199	1,1347241384015190	$-5,293922294741855 \cdot 10^{-15}$
51	1,1347241384015190	1,1347241384015199	1,1347241384015194	0

První sloupec **it** označuje číslo iterace (kroku). Další dva sloupce **a**, **b** značí krajní body zvoleného intervalu, **x** pak jeho střed. Poslední sloupec značí funkční hodnotu v tomto bodě. Řešením je tedy  $x \rightarrow 1,13472413840152$ . Všechny cifry ve výsledku jsou platné. Pro kontrolu je možné použít vestavěný příkaz FindRoot.

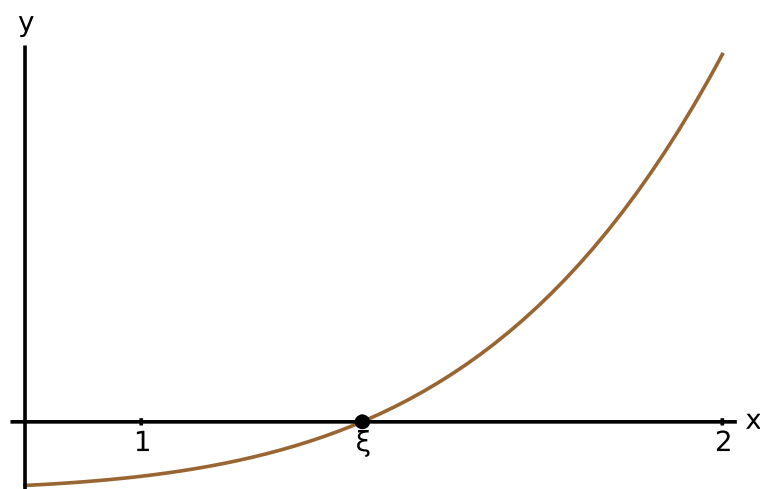
### Příklad 9.2

Metodou půlení intervalů (bisekce) vypočítejte  $\sqrt[5]{5}$  s přesností na 15 desetinných míst.

Nejprve je nutné pátou odmocninu z čísla 5 převést na rovnici v tvaru  $f(x) = 0$ . Jedná se vlastně o hledání čísla, které po umocnění „na pátou“ dá výsledek 5, tj.

$$x^5 - 5 = 0.$$

Ještě než začne samostatný výpočet, je rovněž potřeba zjistit interval, na kterém se nachází kořen. Ze znalostí mocninné funkce plyne, že kořen musí ležet v intervalu  $[1, 2]$  (spíše blíže k číslu 1, ale jako počáteční odhad postačí uvedený interval), což potvrzuje i následující graf.

Obr. 9.2 Funkce  $f(x) = x^5 - 5$ 

Opět využitím vytvořeného programu a zadání příkazu  
`NMBisection[x^6-x-1, {x, 1, 2}, AccuracyGoal -> 15,`  
`ShowProgress -> True]`  
 získá následující tabulka

Tab. 9.2 Postup výpočtu kořene rovnice  $x^5 - 5 = 0$   
 metodou bisekce

it	a	b	x	f(x)
1	1,0000000000000000	2,0000000000000000	1,5000000000000000	2,5937500000000000
2	1,0000000000000000	1,5000000000000000	1,2500000000000000	-1,9482421875000000
3	1,2500000000000000	1,5000000000000000	1,3750000000000000	-0,085113525390625
4	1,3750000000000000	1,5000000000000000	1,4375000000000000	1,138175010681152
5	1,3750000000000000	1,4375000000000000	1,4062500000000000	0,499366670846939
6	1,3750000000000000	1,4062500000000000	1,3906250000000000	0,200560622848570
...	...	...	...	...
...	...	...	...	...
48	1,3797296614612122	1,3797296614612193	1,3797296614612158	$8,161427698079727 \cdot 10^{-14}$
49	1,3797296614612122	1,3797296614612158	1,3797296614612140	$1,724091134919237 \cdot 10^{-14}$
50	1,3797296614612140	1,3797296614612158	1,3797296614612149	$-1,494577146660983 \cdot 10^{-14}$
51	1,3797296614612140	1,3797296614612149	1,3797296614612145	0

Značení sloupců je stejné jako v předcházejícím příkladě. Výsledkem je tedy  
 $x \rightarrow 1,37972966146121$ . Všechny cifry ve výsledku jsou platné.

## 10 METODA REGULA FALSI

### 10.1 Implementace algoritmu

Algoritmus metody regula falsi je téměř totožný jako u metody půlení intervalů. Liší se pouze volbou dělicího bodu („středu“) intervalu. Je zde ovšem problém v už zmíněném zjištění chyby výsledku a zastavení výpočtu po uživatelem zvolené přesnosti. Pokud by program měl vracet výsledek přímo s daným počtem platných cifer, musela by se vypočítat chyba ze vztahu (5.2), což obnáší i zjištění minimální velikosti derivace na řešeném intervalu. Když bude například derivace funkce na intervalu monotónně klesající (rostoucí), potom se nejmenší hodnota nachází právě v krajním bodě tohoto intervalu. To je ale pouze jeden z případů, jak může vypadat vstupní funkce. Implementovat algoritmus, který by tento problém řešil obecně, je náročné a celá tato příprava by mohla stát spoustu výpočetního času. Proto bude uživatel pouze vyzván k zadání největší relativní chyby mezi dvěma řešeními. Jako kritérium pro zastavení výpočtu byl tedy zvolen vztah (9.3), doplněný o zadání maximálního počtu kroků (iterací) tak, aby bylo možné lépe kontrolovat průběh výpočtu.

---

```
While[True,
  (* vypocet deliciho bodu a funkcnich hodnot v krajnich bodech *)
  fa = fg /. x -> a;
  fb = fg /. x -> b;
  s = (b * fa + a * fb) / (fa + fb);
  fs = fg /. x -> s;

  (* vypis mezivysledku *)
  If[progress, Print[N[{it, a, b, s, fs},wprec]]];

  (* kontrola zda byla dosazena zadana odchylka *)
  If[Abs[fs] <= reqErr, Return[{x -> N[s, wprec]}]];

  (* kontrola nalezen presneho reseni *)
  If[fs == 0, Return[{x -> s}]];

  (* zvoleni intervalu do dalsiho kroku *)
  If[Negative[fa * fs], b = s, a = s];

  (* kontrola zda se na intervalu [a, b] nachazi vice korenu *)
  If[Positive[fa] && Positive[fb],
    Message[NMRegFalsi::badRange];
    Throw[$Failed]
  ];
];
```

```
(* zastavení výpočtu po dosažení maximalního počtu opakování *)
If[++it == maxIt,
  Message[NMRegFalsi::maxIt, it];
  Throw[$Failed]
];
```

## 10.2 Příklady

### Příklad 10.1

Metodou Regula Falsi nalezněte řešení rovnice  $x - \cot x = 0$  na intervalu  $[\frac{\pi}{4}, \frac{\pi}{2}]$ . Výpočet ukončete, jakmile bude relativní chyba mezi dvěma po sobě jdoucími řešeními menší než  $1 \cdot 10^{-10}$ . Pokuste se rovněž odhadnout chybu řešení.

Opět využitím vytvořeného programu a zadáním příkazu

```
NMRegFalsi[x - Cot[x], {x, Pi /4, Pi /2},
RelativeError -> 10^{-10}, ShowProgress -> True]
```

se získá následující tabulka.

Tab. 10.1 Postup výpočtu kořene rovnice  $x - \cot x = 0$   
metodou regula falsi

it	a	b	s	f(s)
0	0,785398163397448	1,570796326794897	0,8798016929768852	0,05279205517056876
1	0,785398163397448	0,8798016929768852	0,8611634349792692	0,002272892303819781
2	0,785398163397448	0,8611634349792692	0,8603693991617098	0,0000981240970258853
3	0,785398163397448	0,8603693991617098	0,8603351351367500	$4,236626861198189 \cdot 10^{-6}$
4	0,785398163397448	0,8603351351367500	0,8603336557751336	$1,829223667249623 \cdot 10^{-7}$
...	...	...	...	...
...	...	...	...	...
9	0,785398163397448	0,8603335890196118	0,8603335890193898	$2,744733943182598 \cdot 10^{-14}$
10	0,785398163397448	0,8603335890193898	0,8603335890193802	$1,185078017288932 \cdot 10^{-15}$
11	0,785398163397448	0,8603335890193802	0,8603335890193798	0

Výsledkem je tedy  $x \rightarrow 0,8603335890193798$ . Po 11 krocích se získal výsledek, jehož relativní chyba od předcházejícího je menší než  $1 \cdot 10^{-10}$ . Bohužel stále nemáme žádnou vypovídající hodnotu o tom, kolik má výsledek platných číslic.

Ke zjištění chyby výsledku se použije věta o univerzálnímu odhadu chyby. Nejprve se odvodí minimální derivace na daném intervalu. V tomto příkladě je to poměrně snadné, neboť funkce je na celém intervalu klesající, takže  $m = f'(\frac{\pi}{2}) = 2$  a pro odhad chyby platí

$$|x - \xi| \leq \frac{f(x)}{m} \leq \frac{f(0,86033)}{2} \leq 1,11 \cdot 10^{-16}.$$

To odpovídá 15 správně zaokrouhleným desetinným místům, neboť

$$|x_{11} - \xi| \leq 1,11 \cdot 10^{-16} \leq 0,5 \cdot 10^{-15}. \quad (10.1)$$

Výsledek je ve tvaru  $x \rightarrow 0,860333589019379$ , kde všechny cifry jsou platné.

Za pouhých 11 kroků se získal výsledek s 15 platnými ciframi. Výše uvedený příklad je tedy velmi vhodný pro řešení metodou regula falsi. Největším problémem ovšem zůstává určení odhadu chyby. Zde to bylo poměrně snadné, ale obecně to vždy nemusí být ihned zřejmé. Aby bylo možné porovnat rychlost konvergence metody bisekce a metody regula falsi, bude použito stejné zadání jako v prvním příkladu v předcházející kapitole. Díky šesté mocnině to není funkce, která by měla průběh blízký lineárnímu, proto zřejmě nebude metoda regula falsi příliš vhodná.

### Příklad 10.2

Metodou regula Falsi nalezněte kořen rovnice  $x^6 - x - 1 = 0$  na intervalu  $I = [1, 2]$  s přesností na 15 desetinných míst.

Zadáním příkazu

```
NMRegFalsi[x^6-x-1=0, {x, 1, 2}, RelativeError -> 10^{-10},  
ShowProgres -> True]
```

se získá následující tabulka

Tab. 10.2 Postup výpočtu kořene rovnice  $x^6 - x - 1 = 0$   
metodou regula falsi

it	a	b	s	f(s)
0	1,0000000000000000	2,0000000000000000	1,016129032258065	-0,9153677138206430
1	1,016129032258065	2,0000000000000000	1,030674754131172	-0,8319214145135880
2	1,030674754131172	2,0000000000000000	1,043716600659940	-0,7510231552637691
3	1,043716600659940	2,0000000000000000	1,055347031043141	-0,6737806512483779
4	1,055347031043141	2,0000000000000000	1,065667281981327	-0,6010309811208594
...	...	...	...	...
...	...	...	...	...
97	1,134724099633521	2,0000000000000000	1,134724105290876	$-3.406299825392923 \cdot 10^{-7}$
98	1,134724105290876	2,0000000000000000	1,134724110122661	$-2,909223751115230 \cdot 10^{-7}$
99	1,134724110122661	2,0000000000000000	1,134724114249352	$-2,484685213714889 \cdot 10^{-7}$

Výpočet byl zastaven po 100 iteracích. Porovnáním obou metod se zjistí, že pouze 7 desetinných míst je správně, a to bylo k výpočtu použito téměř dvojnásobné množství kroků oproti metodě půlení intervalů. Samozřejmě pokračováním ve výpočtu by se dospělo ke stejnému výsledku, nicméně tento příklad demonstroval nevhodnost použití metody regula falsi na některé typy funkcí.

U obou příkladů se typicky jeden z krajních bodů zvoleného intervalu dříve nebo později nemění. V posledním příkladě se již od začátku výpočet krajního bodu  $b$  vůbec nezměnil. Tuto situaci nejlépe znázorňuje obrázek 5.2.

Jak už bylo řečeno, rychlost konvergence této metody silně závisí na průběhu funkce  $f$ . Nejlepší výsledky lze dosáhnout při aplikaci na řešení přibližně lineárních funkcí na tomto intervalu.



## 11 METODA PROSTÉ ITERACE

### 11.1 Implementace algoritmu

Samotné jádro algoritmu je velmi jednoduché. Jedná se o výpočet vztahu (6.2). Jak už bylo zmíněno, na začátku je ovšem nutné uvést jednak počáteční odhad  $x_0$  a také iterační funkci  $g$ . I když by zřejmě bylo možné určitými postupy tyto dvě informace vyvodit ze zadání příkladu, přenechá se pro jednoduchost tato volba uživateli. Uživatel bude vyzván rovněž k zadání relativní odchylky mezi dvěma iteracemi, která bude sloužit jako podmínka k zastavení výpočtu (viz vztah (9.3)) podobně jako tomu bylo u metody regula falsi. Rovněž bude umožněno kontrolovat výpočet zadáním maximálního počtu iterací. Část algoritmu by mohla vypadat například takto:

---

```
While[True,
  (* vypocet aktualniho reseni z predchazejiciho *)
  xAct = iterFnc /. x -> xPrev;
  fxAct = fg /. x -> xAct;

  (* vypis mezivysledku *)
  If[progress, Print[{it, xAct, fxAct}]];

  (* overeni relativni chyby *)
  If[Abs[(xAct-xPrev)/xPrev] <= minReqErr, Return[{x -> N[xAct,
    prec]}]];

  (* zastaveni vypoctu po prekroceni maximalniho poctu iteraci *)
  If[++it == maxIt,
    Message[NMIter::maxIt, it];
    Throw[$Failed]
  ];
];
```

---

## 11.2 Příklady

### Příklad 11.1

Metodou prosté iterace vypočítejte kořen rovnice  $x - e^{-x/5} = 0$  na intervalu  $I = [\frac{3}{4}, 1]$ . Počáteční odhad volte  $x_0 = 1$ . Výpočet ukončete, jakmile je relativní chyba výsledku vůči předchozí hodnotě rovna 0,00001.

Nejprve se určí iterační funkce  $g$ . Jedna z nich je například (vznikla osamostatněním  $x$  na levé straně rovnice)

$$g(x) = e^{-x/5},$$

protože platí

$$\begin{aligned} g'(x) &= -\frac{1}{5} \cdot e^{-x/5}, \\ g'\left(\frac{3}{4}\right) &\doteq -0,172, \\ g'(1) &\doteq -0,164. \end{aligned}$$

A také

$$g''(x) = \frac{1}{25} \cdot e^{-x/5},$$

tedy funkce na intervalu  $I$  monotónně klesá a platí, že  $|g'(x)| < 1, \forall x \in I$  a takto zvolená iterační funkce konverguje pro libovolné  $x_0 \in I$ . Zvolíme např. jednu z krajních hodnot a zadáním příkazu

```
NMIter[x-Exp[(-x/5)], {x, 3/4, 1}, Exp[(-x/5),  
RelativeError -> 10^{-10}, ShowProgres -> True]]
```

se získá následující tabulka.

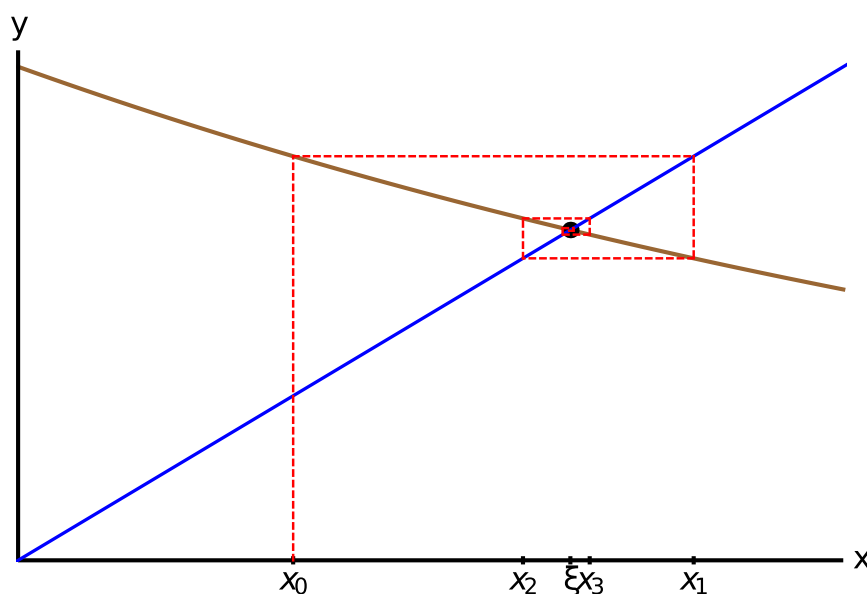
Tab. 11.1 Postup výpočtu kořene rovnice  $x - e^{-x/5} = 0$   
metodou prosté iterace

it	x	f(x)
0	1,0000000000000000	0,18126924692201818
1	0,818730753077981	-0,03022674879057074
2	0,848957501868552	0,00511676313398226
...	...	...
...	...	...
5	0,844583430036431	$4,16431073070724418 \cdot 10^{-6}$
6	0,844579265725700	$-7,03418392764111563 \cdot 10^{-7}$
7	0,844579969144093	$1,18818608533300575 \cdot 10^{-7}$

Předběžným výsledkem, zatím bez znalosti chyby, je tedy

$x \rightarrow 0,844579969144093$ .

Pro lepší představu je uveden obrázek znázorňující počáteční průběh výpočtu.



Obr. 11.1 Grafické znázornění výpočtu  $x - e^{-x/5} = 0$  metodou  
prosté iterace

Barevné označení křivek je shodné s grafy v části 6. Nyní je nutné uvést, kolik desetinných míst je ve výsledku správně. S využitím vztahu (6.7) se získá

$$|x_7 - \xi| \leq \frac{0,172}{1 - 0,172} |1 - 0,1812692469220| \leq 1,46121 \cdot 10^{-7}.$$

To odpovídá 6 správně zaokrouhleným desetinným místům, neboť

$$|x_7 - \xi| \leq 1,46121 \cdot 10^{-7} \leq 0,5 \cdot 10^{-6}.$$

Výsledek je ve tvaru  $x \rightarrow 0,844579$ , kde všechny cifry jsou platné.

### Příklad 11.2

Metodou prosté iterace vypočítejte kořen rovnice  $x - 1 - \frac{1}{2}e^{-x} = 0$  na intervalu  $I = [1, 2]$ . Počáteční odhad volte  $x_0 = 1$ . Výpočet ukončete, jakmile je relativní chyba výsledku vůči předchozí hodnotě rovna 0,00001.

Obdobně jako v předchozím příkladu se zvolí vhodná iterační funkce, jednou z nich je například

$$g(x) = 1 + \frac{1}{2}e^{-x}.$$

Posloupnost určená touto funkcí konverguje pro libovolné  $x \in I$ , protože platí

$$\begin{aligned} g'(x) &= -\frac{1}{2}e^{-x}, \\ g'(1) &= -\frac{1}{2}e^{-1} \doteq -0,18394, \\ g'(2) &= -\frac{1}{2}e^{-2} \doteq -0,067668, \\ g''(x) &= \frac{1}{2}e^{-x}. \end{aligned}$$

Funkce  $g$  na celém intervalu  $I$  monotónně klesá, přičemž platí  $|g(x)| < 1, \forall x \in I$ .

Po zadání příkazu

```
NMIter[x-1-(1/2)e^(-x), {x, 1}, 1 + (1/2) Exp[-x]
RelativeError -> 10^{-10}, ShowProgress -> True]
```

se získá následující tabulka.

Tab. 11.2 Postup výpočtu kořene rovnice  $x - 1 - \frac{1}{2}e^{-x} = 0$   
metodou prosté iterace

it	x	f(x)
0	1,0000000000000000	0,030904456686543513
1	1,153035263899177	-0,004803311255371589
2	1,157838575154549	0,000756329906438590
...	...	...
...	...	...
5	1,157182413799824	$-2,93657023031634878 \cdot 10^{-6}$
6	1,157185350370054	$4,61585142774723067 \cdot 10^{-7}$
7	1,157184888784912	$-7,25542259605305162 \cdot 10^{-7}$

Předběžným výsledkem, zatím bez znalosti chyby, je tedy  $x \rightarrow 1,157184888784912$ . Nyní je nutné uvést, kolik desetinných míst je ve výsledku správně. S využitím vztahu (6.7) se získá

$$\begin{aligned}
 |x_7 - \xi| &\leq \frac{0,67668}{1 - 0,67668} |1,157184888784912 - 1,157185350370054| \\
 &\leq 9,66057 \cdot 10^{-7}.
 \end{aligned}$$

To odpovídá 6 správně zaokrouhleným desetinným místům, neboť

$$|x_7 - \xi| \leq 9,66057 \cdot 10^{-7} \leq 0,5 \cdot 10^{-6}.$$

Výsledek je ve tvaru  $x \rightarrow 1,157184$ , kde všechny cifry jsou platné.

## 12 NEWTONOVA METODA

### 12.1 Implementace algoritmu

Vzhledem k tomu, že Newtonova metoda je speciálním případem metody prosté iterace uvedené v kapitole 6, není potřeba dodatečně vytvářet nový algoritmus. Použije se tedy stávající s tím, že vstupem pro iterační funkci bude vztah (7.2). Tedy definice funkce pro výpočet kořenů pomocí Newtonovy metody bude v sobě pouze zapouzdřovat volání funkce pro výpočet metodou prosté iterace.

---

```
NMNewton[f_, {x_, x0_}, options_] :=
  NMIter[f, {x, x0}, x - f/D[f, x], options]
```

---

Nepříjemnost a pravděpodobně početně nejnáročnější operací zpomalující celý výpočet je nutnost znalosti derivace funkce  $f$  v každém kroku výpočtu. Derivace je definována jako

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}. \quad (12.1)$$

Pro počítač je tento výpočet časově náročný (rovněž k němu používá numerické metody), a proto je někdy vhodné převést derivaci do následující podoby

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x} \quad (12.2)$$

pro vhodně malé  $\Delta x$ . Pro jednu iteraci se tak funkce vyčísluje dvakrát, což bývá ve většině případů rychlejší než výpočet derivace.

Na druhou stranu Mathematica umožňuje symbolické derivování, které je oproti numerickému výpočtu rychlejší. Pro tento případ není nutné brát výpočet derivace jako velký problém. K zastavení výpočtu se používají úplně stejné podmínky jako u metody prosté iterace (maximální počet iterací a největší relativní chyba mezi dvěma po sobě jdoucími výsledky).

## 12.2 Příklady

### Příklad 12.1

Newtonovou metodou nalezněte kořen rovnice  $x^6 - x - 1 = 0$  na intervalu  $[1, 2]$ .

Newtonovou metodou dostáváme

$$x_{i+1} = x_i - \frac{x_i^6 - x_i - 1}{6x_i^5 - 1}. \quad (12.3)$$

Za počáteční odhad se například zvolí  $x_0 = 1,5$ . Po zadání příkazu `NMNewton[x^6-x-1 {x, 1.5}, ShowProgress -> True]` se získá následující tabulka.

Tab. 12.1 Postup výpočtu kořene rovnice  $x^6 - x - 1 = 0$   
Newtonovou metodou

it	x	f(x)
0	1,5000000000000000	8,890630000000000
1	1,300490883590462	2,537264143440597
2	1,139455590275526	0,049235251006873
3	1,134777625237109	0,000550323858393
4	1,134724145316217	$7,113585207090821 \cdot 10^{-8}$
5	1,134724138401519	$1,189046717127037 \cdot 10^{-15}$

Předběžným výsledkem, zatím bez znalosti chyby, je tedy  $x \rightarrow 1,134724138401519$ .

K jejímu odhadu se využije vztah (7.4). Pro první derivaci platí

$$f'(x) = 6x^5 - 1,$$

ta je na intervalu  $[1, 2]$  monotónně rostoucí a  $m$  se vypočte jako

$$m = \min_{[1,2]} |6x^5 - 1| = 5. \quad (12.4)$$

Pro druhou derivaci platí

$$f''(x) = 30x^4,$$

ta je na intervalu  $[1, 2]$  rovněž monotónně rostoucí a  $M$  se vypočte jako

$$M = \max_{[1,2]} |30x^4| = 480.$$

Potom pro odhad chyby dostáváme

$$|x_{i+1} - \xi| \leq \frac{480}{10} |1,134724145316217 - 1,134724138401519|^2 \doteq 2,29503 \cdot 10^{-15},$$

což odpovídá 14 platným desetinným místům, protože

$$2,29503 \cdot 10^{-15} \leq \frac{1}{2} 10^{-14}.$$

Výsledek je ve tvaru  $x \rightarrow 1,13472413840151$ , kde všechna čísla jsou platná.

### Příklad 12.2

Newtonovou metodou vypočítejte  $\sqrt[5]{5}$ . Výpočet ukončete, jakmile je relativní chyba dvou posledních iterací menší než 0,00001.

Příklad už byl vyřešený metodou bisekce. Jedná se o hledání kořene rovnice

$$x^5 - 5 = 0. \quad (12.5)$$

Jako počáteční odhad se zvolí například  $x_0 = 1,5$ . Po zadání příkazu `NMNewton[x^5-5, {x, 1.5}, ShowProgress -> True]` se získá následující tabulka.

Tab. 12.2 Postup výpočtu kořene rovnice  $x^5 - 5 = 0$   
Newtonovou metodou

it	x	f(x)
0	1,500000000000000	2,593750000000000
1	1,379245351827272	-0,008769285828370
2	1,379758353642661	0,0005199093635911
3	1,379727986876797	-0,000030342544610
4	1,379729759283435	$1,7724891474003948 \cdot 10^{-6}$
5	1,379729655747146	$-1,0353600664603852 \cdot 10^{-7}$



Předběžným výsledkem, zatím bez znalosti chyby, je tedy

$x \rightarrow 1,379729655747146$ .

Stejně jako v minulém příkladě se odvodí velikost chyby na základě vztahu (7.4).

Pro  $m$  a  $M$  platí

$$m = \min_{[1,2]} |5x^4| = 5,$$

$$M = \max_{[1,2]} |20x^3| = 160.$$

Potom absolutní odchylka je

$$|x_{i+1} - \xi| \leq \frac{160}{5} |1,379729759283435 - 1,379729655747146|^2 \doteq 3,31316 \cdot 10^{-6},$$

což odpovídá 5 platným desetinným místům, protože

$$3,31316 \cdot 10^{-6} \leq \frac{1}{2} 10^{-5}.$$

Výsledek je ve tvaru  $x \rightarrow 1,37972$ , kde všechna čísla jsou platná.

## 13 METODA SEČEN

### 13.1 Implementace algoritmu

Už v předchozí kapitole bylo uvedeno, že jednotlivé metody mohou mít velice podobný postup výpočtu. Metoda sečen je velmi podobná metodě regula falsi, až na to, že se nebudou zjišťovat znaménka v krajních bodech intervalu a do dalšího kroku se tak vždy použijí dvě předcházející hodnoty bez ohledu na znaménka jejich funkčních hodnot.

---

```
While[True,
  (* vypocet vysledku na zaklade predchozich hodnot *)
  fxPrev      = fg /. x -> xPrev;
  fxPrevPrev  = fg /. x -> xPrevPrev;
  tmp         = xPrev;
  xPrev       = xPrev - fxPrev*(xPrev - xPrevPrev) / (fxPrev -
    fxPrevPrev);
  xPrevPrev   = tmp;

  (* vypis mezivysledku *)
  If[progress, Print[{it, xPrev, fxPrev}]];

  (* overeni relativni chyby *)
  If[Abs[fxPrev] <= minReqErr, Return[{x -> N[xPrev, prec]}]];

  (* zastaveni vypoctu po prekročení maximalního počtu iterací *)
  If[++it == maxIt,
    Message[NMSec::maxIt, it];
    Throw[$Failed]
  ];
];
```

---

Vstupem jsou dva počáteční odhady a v cyklu se provádí výpočet nového odhadu vždy na základě dvou předchozích. Výpočet končí, jakmile je dosaženo požadované přesnosti, přičemž jako rozhodující kritérium se použije vztah (9.3). Uživatel bude vyzván k zadání dvou odhadů řešení a právě požadované přesnosti. Výpočet rovněž končí, když je dosaženo maximálního počtu kroků, který lze rovněž specifikovat vstupním parametrem.

## 13.2 Příklady

Pro srovnání se použije následující příklad, který byl už několikrát vyřešen v předcházejících metodách.

### Příklad 13.1

Metodou sečen nalezněte kořen rovnice  $x^6 - x - 1 = 0$  na intervalu  $[1, 2]$ . Výpočet ukončete, jakmile je relativní chyba dvou po sobě jdoucích výsledků menší než 0,000001.

Po zadání příkazu

```
NMSec[x^6-x-1, {x, 1, 2}, RelativeError -> 0.000001,  
ShowProgress -> True]
```

se získá následující tabulka.

Tab. 13.1 Postup výpočtu kořene rovnice  $x^6 - x - 1 = 0$   
metodou sečen

it	x	f(x)
0	1,016129032258065	-1,000000000000000
1	1,190577768676637	-0,915367713820643
2	1,117655830941552	0,6574656967175045
3	1,132531550216133	-0,1684911678388040
4	1,134816808004853	-0,02243728618794979
5	1,134723645948705	0,000953564064100893
6	1,134724138291216	-5,066165711602604 · 10 <sup>-6</sup>
7	1,134724138401520	-1,134761595673584 · 10 <sup>-9</sup>

Předběžným výsledkem, zatím bez znalosti chyby, je tedy

$x \rightarrow 1,134724138401520$ . S využitím věty o univerzálním odhadu chyby platí

$$|x_7 - \xi| \leq \frac{|f(x_7)|}{m} = \frac{1,13476 \cdot 10^{-9}}{5} \doteq 5,674 \cdot 10^{-10},$$

což odpovídá 9 správně zaokrouhleným desetinným místům. Výsledek je ve tvaru  $x \rightarrow 1,134724138$ , kde všechna čísla jsou platná.

**Příklad 13.2**

Metodou sečen vypočtete  $\sqrt[5]{5}$ . Výpočet ukončete, jakmile je relativní chyba dvou posledních výsledků menší než 0,00001.

Po zadání příkazu

```
NMSec[x^5-5, {x, 1, 2}, ShowProgress -> True,  
RelativeError -> 0.000001]
```

se získá následující tabulka

Tab. 13.2 Postup výpočtu kořene rovnice  $x^5 - 5 = 0$  metodou sečen

it	x	f(x)
0	1,129032258064516	-4,000000000000000
1	1,618445012799713	-3,165440707620006
2	1,296158100255089	6,104263105753632
3	1,354228333650487	-1,341610924788847
4	1,383078752471551	-0,4453031136798379
5	1,379603865034384	0,06097914451711710
6	1,379729052181782	-0,002278951694927873
7	1,379729661572327	-0,00001103982380838389
8	1,379729661461215	$2,013291289600973 \cdot 10^{-9}$

Předběžným výsledkem, zatím bez znalosti chyby, je tedy

$x \rightarrow 1,379729661461215$ . S využitím věty o univerzálním odhadu chyby platí

$$|x_8 - \xi| \leq \frac{|f(x_8)|}{m} = \frac{2,01329 \cdot 10^{-9}}{4} \doteq 5,0332 \cdot 10^{-10},$$

což odpovídá minimálně 9 správně zaokrouhleným desetinným místům. Výsledek je ve tvaru  $x \rightarrow 1,379729661$ , kde všechny číslice jsou platné.

## 14 POUŽITÍ SOFTWARE MATHEMATICA

V této práci byly při výpočtech příkladů použity vlastní funkce v programu Mathematica. Níže je uveden podrobnější návod na jejich použití, včetně uvedení všech dostupných volitelných parametrů ovlivňujících výpočet. Nejprve je nutné toto dodatečné rozšíření připojit k programu tak, aby byly všechny funkce viditelné a bylo možné je používat. Postup připojení tzv. balíčku je následující:

1. Spuštění programu Mathematica a otevření nového „notebooku“  
File->New->Notebook(.nb).
2. Zadání příkazu `Needs["NonLinNumMethods`", <CESTA K SOUBORU>]` se specifikací cesty k souboru `nonlinnummethods.m` pomocí druhého parametru. Uvedený soubor je přiložen k této práci. Rovněž ve stejné složce musí být přítomny soubory jednotlivých numerických metod.

Nyní lze používat v Mathematice následující funkce:

`NMBisection[<ROVNICE>, {<PROMĚNNÁ>, <LEVÝ BOD>, <PRÁVÝ BOD INTERVALU>}, <VOLBY>]`

`NMRegFalsi[<ROVNICE>, {<PROMĚNNÁ>, <LEVÝ BOD>, <PRÁVÝ BOD INTERVALU>}, <VOLBY>]`

`NMIter[<ROVNICE>, {<PROMĚNNÁ>, <POČÁTEČNÍ ODHAD>}, <ITERAČNÍ FCE>, <VOLBY>]`

`NMNewton[<ROVNICE>, {<PROMĚNNÁ>, <POČ. ODHAD>}, <VOLBY>]`

`NMSec[<ROVNICE>, {<PROMĚNNÁ>, <POČ. ODHAD 1>, <POČ. ODHAD 2>}, <VOLBY>]`

Příkazy využívají následující numerické metody (ve stejném pořadí): metoda bisekce, regula falsi, metoda prosté iterace, Newtonova metoda a metoda sečen.

Funkce hledají kořen rovnice určený parametrem `ROVNICE`. Ten může být zadán ve tvaru `f == g` nebo ve tvaru `f` (potom se automaticky předpokládá `f == 0`), kde `f` a `g` značí matematické funkce s jednou nezávisle proměnnou, jejíž název se musí u všech příkazů specifikovat parametrem `PROMĚNNÁ` (například `x`). Za ním následuje určení intervalu, na kterém se nachází jeden kořen. Interval se specifikuje na místě parametrů `LEVÝ BOD` a `PRÁVÝ BOD` oddělených čárkou. Určení počátečního odhadu( $\hat{u}$ ) se provádí na místě `POČ. ODHAD <1,2>`. Tato vymezení intervalu i odhadu( $\hat{u}$ ) musí být reálná čísla. Parametr specifikovaný na místě `ITERAČNÍ FCE` určuje iterační funkci pro metodu prosté iterace.

Všechny příkazy obsahují část určující nastavení výpočtu (parametr `VOLBY`). Seznam voleb je následující:

- `WorkingPrecision` -> <ČÍSLO>
- `AccuracyGoal` -> <ČÍSLO> (pouze pro `NMBisection`)
- `RelativeError` -> <ČÍSLO> (kromě `NMBisection`)

- `MaxIterations` -> `<ČÍSLO>`
- `ShowProgress` -> `<TRUE, FALSE>`

Všechny volby jsou nepovinné a v případě jejich absence v zadání příkazu se použijí následující výchozí hodnoty:

- `WorkingPrecision` -> `MachinePrecision`
- `AccuracyGoal` -> `MachinePrecision - 5`
- `RelativeError` -> `0.00001`
- `MaxIterations` -> `20`
- `ShowProgress` -> `false`

Význam jednotlivých parametrů je následující:

- `WorkingPrecision` určuje, jaká přesnost (počet cifer) bude použita v průběhu celého výpočtu.
- `AccuracyGoal` určuje, kolik platných desetinných cifer bude obsaženo ve výsledku.
- `RelativeError` určuje kritérium pro zastavení výpočtu v případě, že maximální relativní chyba mezi dvěma po sobě jdoucími výsledky je menší než definovaná hodnota.
- `MaxIterations` určuje kritérium pro zastavení výpočtu v případě, že počet iterací překročí definovanou hodnotu.
- `ShowProgress` umožňuje zobrazit mezivýsledky v průběhu celého výpočtu.

## ZÁVĚR

V této bakalářské práci bylo popsáno celkem 5 numerických metod (metoda bisekce, metoda regula falsi, metoda prosté iterace, Newtonova metoda a metoda sečen) určených pro řešení nelineárních rovnic. Ke každé z uvedených metod byla vytvořena v softwaru Wolfram Mathematica funkce, která umožňuje tyto rovnice řešit.

Podnět k implementaci tohoto dodatečného rozšíření byl především demonstrační a výukový, neboť je možné u všech funkcí zobrazit průběh výpočtu. Program Mathematica má samozřejmě vlastní a jednoduše dostupné nástroje k řešení těchto problémů, ale nelze v nich zobrazit postup výpočtu, počet iteračních kroků a ani nelze jednoduše říci, která metoda byla použita při řešení zadané rovnice, neboť nejsou volně k dispozici zdrojové kódy jednotlivých funkcí.

Nově vytvořené nástroje si nekladou za cíl rychlejší a přesnější provádění výpočtů oproti původním funkcím vytvořených tvůrci softwaru Mathematica, ale mají sloužit k porovnávání průběhu a rychlosti výpočtu jednotlivých metod, a tím usnadnit jejich pochopení, přiblížit některé nedostatky nebo naopak přednosti jedné metody oproti druhé.

Ovšem zdánlivě jednoznačné závěry, které mohou plynout z porovnání všech metod na jediném vybraném příkladě, je potřeba brát s jistou rezervou. Může se stát, že jedna z metod bude pro tento druh příkladu zcela nevhodná a výpočet bude trvat delší čas než u ostatních metod. Nelze však ihned prohlásit, že tato metoda je nejhorší a její konvergence je pomalá, protože na jiných typech příkladů můžou být výsledky zcela opačné. Pro získání smysluplného porovnání, je potřeba vyzkoušet více druhů příkladů a teprve poté usoudit, kterou metodu bude vhodné dále používat při výpočtech.

## SEZNAM POUŽITÉ LITERATURY

- [1] HOROVÁ, Ivana a Jiří ZELINKA. *Numerické metody*. Vyd. 2., rozš. Brno: Masarykova Univerzita v Brně, 2004, 285 s. ISBN 80-210-3317-7.
- [2] NAVARA, Mirko a Aleš NĚMEČEK. *Numerické metody*. 1. vyd. Praha: Česká technika - nakladatelství ČVUT, 2005, 142 s. ISBN 80-010-2689-2.
- [3] KUBÍČEK, Milan, Miroslava DUBCOVÁ a Drahoslava JANOVSKEÁ. *Numerické metody a algoritmy*. 2. oprav. vyd. Praha: Vydavatelství VŠCHT Praha, 2005, 188 s. ISBN 80-708-0558-7.
- [4] ISAACSON, Eugene. *Analysis of numerical methods*. New York: Dover Publications, 1994, 541 s. ISBN 04-866-8029-0.
- [5] HOFFMAN, Joe D. *Numerical methods for engineers and scientists*. 2nd ed., rev. and expanded. New York: Marcel Dekker, c2001, xi, 823 p. ISBN 08-247-0443-6.
- [6] ZAROWSKI, Christopher J. *An introduction to numerical analysis for electrical and computer engineers*. Hoboken, NJ: Wiley, 2004, xvi, 586 s. ISBN 04-714-6737-5.
- [7] ČERMÁK, Libor a Rudolf HLAVIČKA. *Numerické metody*. Vyd. 2. Brno: Akademické nakladatelství CERM, 2008, 110 s. ISBN 978-80-214-3752-4.
- [8] DOŠLÁ, Zuzana a Jaromír KUBEN. *Diferenciální počet jedné proměnné*. 1. vyd. Brno: Masarykova univerzita, 2003, 209 s. ISBN 80-210-3121-2.
- [9] WOLFRAM RESEARCH, Inc. *Wolfram Language and System Documentation Center* [online]. [cit. 2015-01-07]. Dostupné z: <http://reference.wolfram.com><sup>1)</sup>
- [10] WOLFRAM RESEARCH, Inc. *Wolfram MathWorld* [online]. [cit. 2015-01-07]. Dostupné z: <http://mathworld.wolfram.com/>
- [11] RYBIČKA, Jiří. *LaTeX pro začátečníky*. 3. vyd. Brno: Konvoj, 2003, 238 s. ISBN 80-730-2049-1.<sup>2)</sup>
- [12] LOMTATIDZE, Lenka a Roman PLCH. *Sázíme v LaTeXu diplomovou práci z matematiky*. 1. vyd. Brno: Masarykova univerzita, 2003, 122 s. ISBN 80-210-3228-6.<sup>2)</sup>

---

<sup>1)</sup>Referenční příručka k softwaru, ve kterém byly realizovány algoritmy numerických metod.

<sup>2)</sup>Publikace související se sazbou textu nikoliv s numerickými metodami.



**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

$\mathbb{R}$	množina reálných čísel
$\forall x$	pro všechna $x$
$x \in M$	$x$ patří do množiny $M$
$\sup M$	supremum množiny $M$
$\inf M$	infimum množiny $M$

## SEZNAM OBRÁZKŮ

Obr. 2.1	Postup řešení pomocí numerického modelu . . . . .	11
Obr. 3.1	Ukázka nevhodně zvoleného intervalu . . . . .	23
Obr. 3.2	Ukázka vhodně zvoleného intervalu . . . . .	23
Obr. 3.3	Funkce neprotíná osu $x$ . . . . .	24
Obr. 3.4	Problém kořenů sudé násobnosti . . . . .	24
Obr. 3.5	Kořen liché násobnosti . . . . .	25
Obr. 5.1	Ilustrace metody regula falsi . . . . .	30
Obr. 5.2	Pomalá konvergence metody regula falsi z důvodu velké nelinearity funkce . . . . .	31
Obr. 6.1	Grafické znázornění podmínek konvergence metody prosté iterace [4]	36
Obr. 6.2	Řešení rovnice iterační metodou (rostoucí funkce) . . . . .	37
Obr. 6.3	Řešení rovnice iterační metodou (klesající funkce) . . . . .	38
Obr. 6.4	Nesprávně zvolená iterační funkce (rostoucí funkce) . . . . .	39
Obr. 6.5	Nesprávně zvolená iterační funkce (klesající funkce) . . . . .	40
Obr. 7.1	Grafické znázornění Newtonovy metody . . . . .	42
Obr. 8.1	Řešení rovnice metodou sečen . . . . .	45
Obr. 8.2	Funkce nevhodná pro řešení metodou sečen . . . . .	46
Obr. 9.1	Funkce $f(x) = x^6 - x - 1$ . . . . .	50
Obr. 9.2	Funkce $f(x) = x^5 - 5$ . . . . .	52
Obr. 11.1	Grafické znázornění výpočtu $x - e^{-x/5} = 0$ metodou prosté iterace .	59

## SEZNAM TABULEK

Tab. 9.1	Postup výpočtu kořene rovnice $x^6 - x - 1 = 0$ metodou bisekce . . .	51
Tab. 9.2	Postup výpočtu kořene rovnice $x^5 - 5 = 0$ metodou bisekce . . . . .	52
Tab. 10.1	Postup výpočtu kořene rovnice $x - \cot x = 0$ metodou regula falsi . .	54
Tab. 10.2	Postup výpočtu kořene rovnice $x^6 - x - 1 = 0$ metodou regula falsi .	56
Tab. 11.1	Postup výpočtu kořene rovnice $x - e^{-x/5} = 0$ metodou prosté iterace	59
Tab. 11.2	Postup výpočtu kořene rovnice $x - 1 - \frac{1}{2}e^{-x} = 0$ metodou prosté iterace	61
Tab. 12.1	Postup výpočtu kořene rovnice $x^6 - x - 1 = 0$ Newtonovou metodou	63
Tab. 12.2	Postup výpočtu kořene rovnice $x^5 - 5 = 0$ Newtonovou metodou . .	64
Tab. 13.1	Postup výpočtu kořene rovnice $x^6 - x - 1 = 0$ metodou sečen . . . .	67
Tab. 13.2	Postup výpočtu kořene rovnice $x^5 - 5 = 0$ metodou sečen . . . . .	68

## SEZNAM PŘÍLOH

P I.      Rozšiřující programový balíček

## **PŘÍLOHA P I. ROZŠÍŘUJÍCÍ PROGRAMOVÝ BALÍČEK**

Na přiloženém CD je k dispozici programový balíček pro software Wolfram Mathematica. Obsahuje rozšiřující funkce pro výpočet nelineárních rovnic pomocí numerických metod uvedených v této práci.