Výpočet a využití nulových bodů nelineárních rovnic obsažených v inženýrských aplikacích z řídících nebo informačních technologií

Solution and application of zero points of nonlinear equations included in engineering application of control or information technologies

Jakub Fojtů

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub FOJTŮ**
Osobní číslo:	**A07723**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Výpočet a využití nulových bodů nelineárních rovnic obsažených v inženýrských aplikacích z řídicích nebo informačních technologií**

Zásady pro vypracování:

1. **Zpracujte literární rešerši na téma výpočet nulových bodů (kořenů) nelineárních rovnic. Zaměřte se zejména na typy rovnic, které se vyskytují v technologických procesech z učební látky obsažené v předmětech garantovaných některým ústavem Fakulty aplikované informatiky.**
2. **Navrhněte a naprogramujte ve zvoleném systému počítačové algebry vhodné metody pro nalezení kořenů nelineárních rovnic, které se vyskytnou v inženýrských aplikacích z řídicích nebo informačních technologií, např. při modelování koncentračních polí v tuhé fázi při pracích procesech.**
3. **Porovnejte získané výstupy nebo uveďte využití nulových bodů ve vybraných inženýrských aplikacích.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **FIALKA, M.; CHARVÁTOVÁ, H. Modelling of extraction process in tannery by computer algebraic system. In Topical Problems of Fluid Mechanics 2007. Institute of Thermomechanics, Academy of Sciences of the Czech Republic, Prague, February 28 -- March 2nd, 2007, 37-40.**
2. **CHARVÁTOVÁ, H. Modelování chemického odvápňování holiny. Dizertační práce. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta technologická, 2007.**
3. **KOLOMAZNÍK, K. Analýza dynamických systémů. Skriptum, Brno: VUT v Brně, určeno pro FT ve Zlíně, 1988.**
4. **KOLOMAZNÍK, K. Teorie technologických procesů III. Skriptum, Brno: VUT v Brně, určeno pro FT ve Zlíně, 1978.**
5. **KOLOMAZNÍK, K. Modelování zpracovatelských procesů. Skriptum, Brno: VUT v Brně, určeno pro FT ve Zlíně, 1990.**
6. **VITÁSEK, E. Numerické metody. Praha: SNTL, 1987.**

Vedoucí bakalářské práce: **RNDr. Miloslav Fialka, CSc.**
Ústav matematiky

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Práce se zabývá výpočtem a využitím nulových bodů nelineárních rovnic. Teoretická část práce popisuje numerické metody, které vedou k určení nulových bodů rovnic. V praktické části práce jsou tyto metody realizovány v prostředí Mathematica. K nalezení nulových bodů jsou také použity přímo funkce softwaru Mathematica. Význam hledání nulových bodů je ukázán na příkladech z inženýrských aplikací, zejména z řídících nebo informačních technologií. Cílem práce je popsat způsoby řešení nelineárních rovnic a ukázat jejich využití v praxi.

Klíčová slova: nulový bod, kořen, numerická metoda, nelineární, transcendentní rovnice

ABSTRACT

Work deals with the calculation and application of the zero points of nonlinear equations. The theoretical part describes numerical methods that lead to the determination of zero points of equations. In the practical part, these methods are implemented in the Mathematica environment. To find the zero points are also used directly function of the software Mathematica. Importance of seeking zero points is shown in examples of engineering applications, especially of control or information technologies. The aim is to describe methods of solving nonlinear equations and show their practical use.

Keywords: zero point, root, numerical method, nonlinear, transcendental equation

Děkuji vedoucímu bakalářské práce panu RNDr. Miloslavu Fialkovi, CSc. za odborné vedení a podporu v průběhu vypracovávání práce.

Dále děkuji prof. Ing. Petru Dostálovi, CSc. za poskytnutí modelu chemického reaktoru.

Prohlašuji, že

- beru na vědomí, 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;
- beru na vědomí, že bakalářská 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;
- beru na vědomí, 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;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, 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;
- beru na vědomí, 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 diplomanta

OBSAH

ÚVOD

V dnešní době dokážeme věci, kterým lze přiřadit hodnotu, matematicky popsat. Tak můžeme nejrůznější problémy analyzovat, simulovat, optimalizovat… Tento matematický popis často tvoří rovnice nebo více rovnic.

Rovnice lze rozdělit na algebraické rovnice, označované též jako polynomiální rovnice, a nealgebraické rovnice, označované též jako transcendentní rovnice. Jako algebraickou rovnici n-tého stupně o jedné neznámé označujeme rovnici ve tvaru $a_n x^n + a_{n-1} x^{n-1} + + \ldots + a_1 x + a_0 = 0$, kde levou stranu rovnice tvoří polynom n-tého stupně s $a_n \neq 0$ za předpokladu $n \geq 1$. Pokud rovnici nelze vyjádřit ve tvaru algebraické rovnice, pak mluvíme o rovnici nealgebraické. Transcendentní rovnice dále můžeme rozdělit na nižší a vyšší. Mezi vyšší patří diferenciální rovnice, kterými se tato práce nezabývá.

Číslo, po jehož dosazení za neznámou proměnnou se rovná levá a pravá strana rovnice, nazýváme kořenem rovnice nebo také nulovým bodem. Název nulový bod vznikl zřejmě z geometrického znázornění. Rovnici převedeme do tvaru, kdy je na jedné straně nula a vyneseme ji jako funkci do kartézské soustavy souřadnic. Tam kde funkce protne osu x se nachází nulový bod. Množinu všech kořenů označujeme jako řešení rovnice. Algebraické rovnice do čtvrtého stupně jsou obecně vždy analyticky řešitelné. Je dokázáno, že nemůže existovat obecný vzorec řešící jakoukoli rovnici pátého a vyššího stupně. Potom je třeba řešení hledat numerickými postupy.

Numerickým hledáním kořene rovnice je míněn takový postup, kdy se hodnoty konkrétně vyčíslují. Nenalezneme tedy obecné řešení jako při analytickém postupu. Tato práce se zabývá numerickými postupy pro nalezení kořenů nelineárních transcendentních rovnic. Postup je obecně takový, že sestavíme iterační funkci. Tu vyčíslíme pro zvolenou počáteční aproximaci či více počátečních aproximací kořene a jako výsledek obdržíme přesnější hodnotu aproximace kořene. Iterační funkci vyhodnotíme s novou aproximací a celý postup stále opakujeme, dokud není výsledný nulový bod dostatečně přesný. Tento proces lze provést různými metodami, které se liší svou náročností při výpočtu, robustností a rychlostí zpřesňování kořene, tj. rychlostí konvergence. Takových metod je celá řada a teoretická část práce popisuje nejpoužívanější z nich. V praktické části je na ukázku realizována Brentova metoda v prostředí Mathematica, realizace ostatních metod jsou přiloženy na CD. Dále je ukázána možnost řešit rovnice přímo funkcemi v Mathematice a na závěr je uvedeno pár příkladů praktických aplikací, ve kterých se tyto rovnice řeší.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÍ POJMY

1.1 Definice metrického prostoru

Libovolnou množinu *X* bodů budeme nazývat **metrickým prostorem** [14], pokud je na množině *X* dána tzv. *vzdálenost*, což je jakákoliv jednoznačná nezáporná reálná funkce $\varrho(x,y)$, která je definována pro každou dvojici $x,y \in X$ a která splňuje tyto tři podmínky:

1) $\varrho(x,y) = 0$, když a jen když $x = y$;
2) $\varrho(x,y) = \varrho(y,x)$ $\forall x,y \in X$ (symetrie);
3) $\varrho(x,y) + \varrho(y,z) \geq \varrho(x,z)$ $\forall x,y,z \in X$ (trojúhelníková nerovnost).

Metrický prostor budeme označovat symbolem

$$R = (X,\varrho).\varrho$$

1.2 Kontraktivní zobrazení

Nechť *R* je metrický prostor. Zobrazení φ prostoru *R* do prostoru *R* se nazývá *kontraktivní* [10], pokud existuje takové číslo $\alpha < 1$, že pro libovolné dva body $x,y \in R$ platí nerovnost

$$\varrho(\varphi(x),\varphi(y)) \leq \alpha\varrho(x,y). \quad (1)$$

Každé kontraktivní zobrazení je spojité. Podle (1) platí, že pokud $x_n \to x$, pak také $\varphi(x_n) \to \varphi(x)$. Symbol „→“ znamená zobrazení.

1.3 Banachova věta o pevném bodě

Pevným bodem [1] budeme nazývat bod *x* zobrazení φ, pro který platí $\varphi(x) = x$. Neboli pevné body jsou řešení rovnice $\varphi(x) = x$.

Každé kontraktivní zobrazení φ, definované v úplném metrickém prostoru *R*, má právě jeden pevný bod.

1.4 Důkaz Banachovy věty o pevném bodě

Nechť x_0 je libovolný bod prostoru R [10]. Položme $x_1 = \varphi(x_0)$, $x_2 = \varphi(x_1) = \varphi^2(x_1)$ atd. Značení $\varphi^2(x)$ znamená $\varphi\big(\varphi(x)\big)$. Obecně nechť $x_n = \varphi(x_{n-1}) = \varphi^n(x_0)$.

Ukážeme, že posloupnost $\{x_n\}$ je cauchyovská. Pro dané $m \geq n$, dostaneme

$$\varrho(x_n, x_m) = \varrho\big(\varphi^n(x_0), \varphi^m(x_0)\big) \leq \alpha^n \varrho(x_0, x_{m-n}) \leq$$

$$\leq \alpha^n [\varrho(x_0, x_1) + \varrho(x_1, x_2) + \cdots + \varrho(x_{m-n-1}, x_{m-n})] \leq$$

$$\leq \alpha^n \varrho(x_0, x_1)(1 + \alpha + \alpha^2 + \cdots + \alpha^{m-n-1}) \leq \alpha^n \varrho(x_0, x_1) \frac{1}{1-\alpha}.$$

Jelikož $\alpha < 1$, je pro dostatečně velká přirozená čísla n tento výraz libovolně malý. Vzhledem k úplnosti prostoru R posloupnost $\{x_n\}$, která je cauchyovská, má v tomto prostoru limitu. Položme

$$x = \lim_{n\to\infty} x_n.$$

Potom vzhledem ke spojitosti zobrazení φ je

$$\varphi(x) = \varphi(\lim_{n\to\infty} x_n) = \lim_{n\to\infty} \varphi(x_n) = \lim_{n\to\infty} x_{n+1} = x.$$

Tím je dokázána existence pevného bodu. Dokažme ještě jeho jednoznačnost. Pokud

$$\varphi(x) = x, \qquad \varphi(y) = y,$$

pak z nerovnosti (1) plyne nerovnost

$$\varrho(x, y) \leq \alpha\varrho(x, y),$$

z čehož vzhledem k tomu, že $\alpha < 1$, plyne

$$\varrho(x, y) = 0, \text{ tj. } x = y.$$

2 ŘEŠENÍ OBECNÉ NELINEÁRNÍ ROVNICE

Pro řešení obecné nelineární rovnice

$$f(x) = 0 \tag{2}$$

existuje více různých iteračních metod. Princip těchto metod spočívá v iteračním procesu [12], [13]: Kontraktivní zobrazení $\varphi(x)$ nazveme *iterační funkcí*. Pokud víme, že se kořen $x = x^*$ rovnice (2) nachází na dostatečně malém intervalu, zvolíme v tomto intervalu počáteční odhad x_0 (blízký k x^*) a sestrojíme posloupnost bodů $x_1, x_2, \dots, x_n, \dots$podle rekurentního předpisu

$$x_k = \varphi_k(x_0, x_1, \dots, x_{k-1}). \tag{3}$$

Rekurentní předpis (3) budeme tvořit tak, aby za jistých předpokladů posloupnost $\{x_n\}$ konvergovala ke kořenu x^*. Různou volbou φ_k (závisející na funkci f) dostáváme různé iterační metody.

Často se iterační funkce $\varphi(x)$ volí tak, že hledané řešení x^* je také řešením rovnice

$$x = \varphi(x) \tag{4}$$

přičemž posloupnost $\{x_n\}$ konstruujeme podle vztahu

$$x_k = \varphi(x_{k-1}),\ k = 1,2, \dots \tag{5}$$

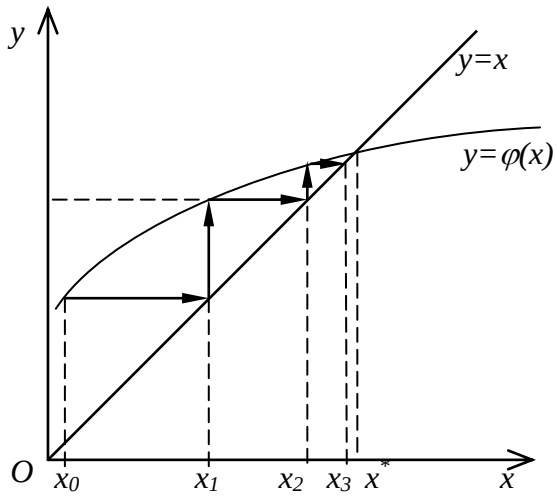
Metody, u kterých se funkce φ se vzrůstajícím indexem iterace nemění, nazýváme *stacionární* a užívají se nejčastěji.

Funkce φ bývá často diferencovatelná. Pokud v intervalu, ve kterém hledáme řešení, platí

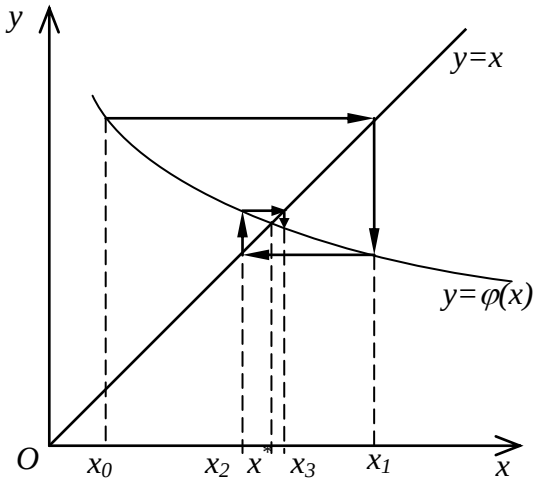
$$|\varphi'(x)| \leq K < 1, \tag{6}$$

postupné aproximace budou konvergovat k x^*. Čím menší bude K, tím rychlejší bude konvergence.

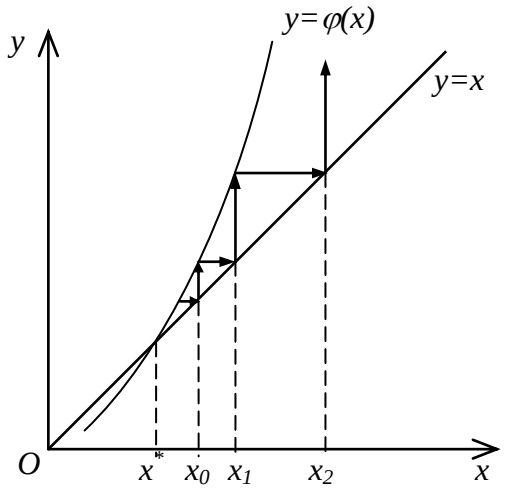
Na obrázcích (Obr. 1) až (Obr. 4) jsou znázorněny kvalitativně odlišné případy iteračního postupu. Na prvních dvou obrázcích je $|\varphi'(x)| < 1$ a posloupnost $\{x_n\}$ bude konvergovat ke kořenu. Na dalších dvou obrázcích je $|\varphi'(x)| \geq 1$ a posloupnost $\{x_n\}$ bude divergovat, iterační proces tedy nepovede k řešení rovnice.



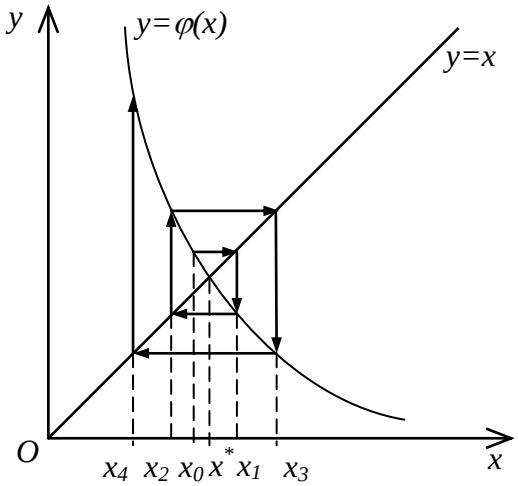
Obr. 1. Iterační proces pro $\varphi' \in (0,1)$.



Obr. 2. Iterační proces pro $\varphi' \in (-1,0)$.



Obr. 3. Iterační proces pro $\varphi' \in (1,\infty)$.



Obr. 4. Iterační proces pro $\varphi' \in (-\infty,-1)$.

2.1 Řád iterace

O rychlosti konvergence iteračního procesu vypovídá tzv. *řád iterace* či *konvergence* [12], [13]. Rozumíme jím takové reálné číslo $m \geq 1$, pro které platí

$$\lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k|^m} = C, \tag{7}$$

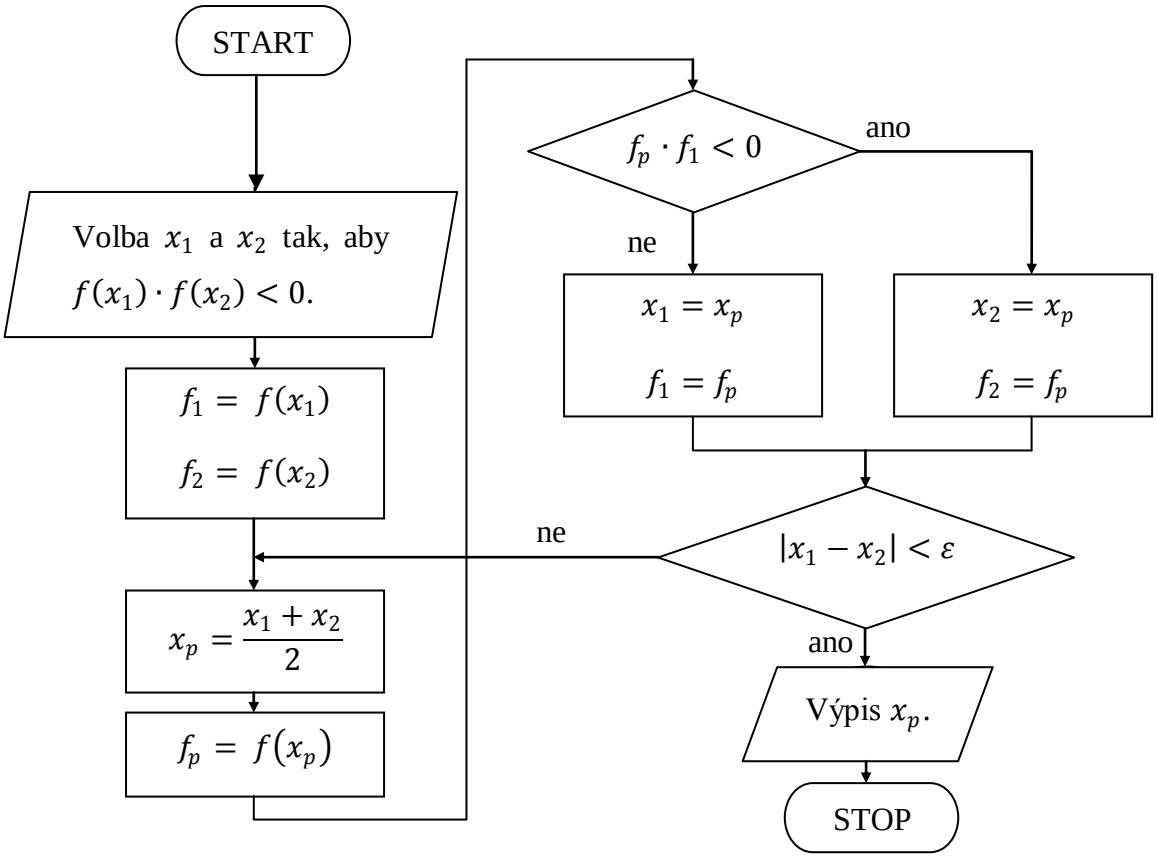
kde C je konstanta, ε_k chyba k – té iterace, tj.

$$\varepsilon_k = x_k - x^*. \tag{8}$$

x^* je přesný kořen rovnice (2). Přitom předpokládáme, že konstanta C je různá od nuly a že v případě $m = 1$ je menší než 1. O metodě, jejíž řád konvergence je 1, říkáme, že konverguje lineárně, při $m > 1$ superlineárně, při $m = 2$ mluvíme o konvergenci kvadratické atd.

2.2 Metoda půlení intervalu

Metoda půlení intervalu [12], [13], někdy označovaná jako bisekce, je jednoduchá metoda, která konverguje sice pomalu, zato však vždy. Je proto vhodná pokud neznáme bližší informace o poloze kořene rovnice (2). Počáteční interval tak můžeme zmenšit a použít sofistikovanější metodu s vyšším řádem konvergence. Metoda půlení intervalu vychází z předpokladu, že funkce $f(x)$ z rovnice (2) je spojitá. Potom stačí nalézt dva body x_1 a x_2 takové, že funkční hodnoty v bodech mají rozdílná znaménka. Tedy $f(x_1) \cdot f(x_2) < 0$. Ze spojitosti potom vyplývá, že mezi x_1 a x_2 leží aspoň jeden kořen. Metoda funguje tak, že vezmeme bod ležící uprostřed mezi body x_1 a x_2, označme jej jako x_p. Tj. $x_p = (x_1 + x_2)/2$. Ke konstrukci dalšího bodu použijeme ten z intervalů $[x_1, x_p]$ a $[x_p, x_2]$, pro který platí $f(x_k) \cdot f(x_p) < 0$. Tj. opět interval, jehož krajní body mají opačná znaménka funkčních hodnot. Krajní body intervalu označme opět x_1 a x_2. Tímto způsobem pokračujeme dále. Postup je znázorněn ve vývojovém diagramu na (*Obr. 5*).



Obr. 5. Vývojový diagram metody půlení intervalu.

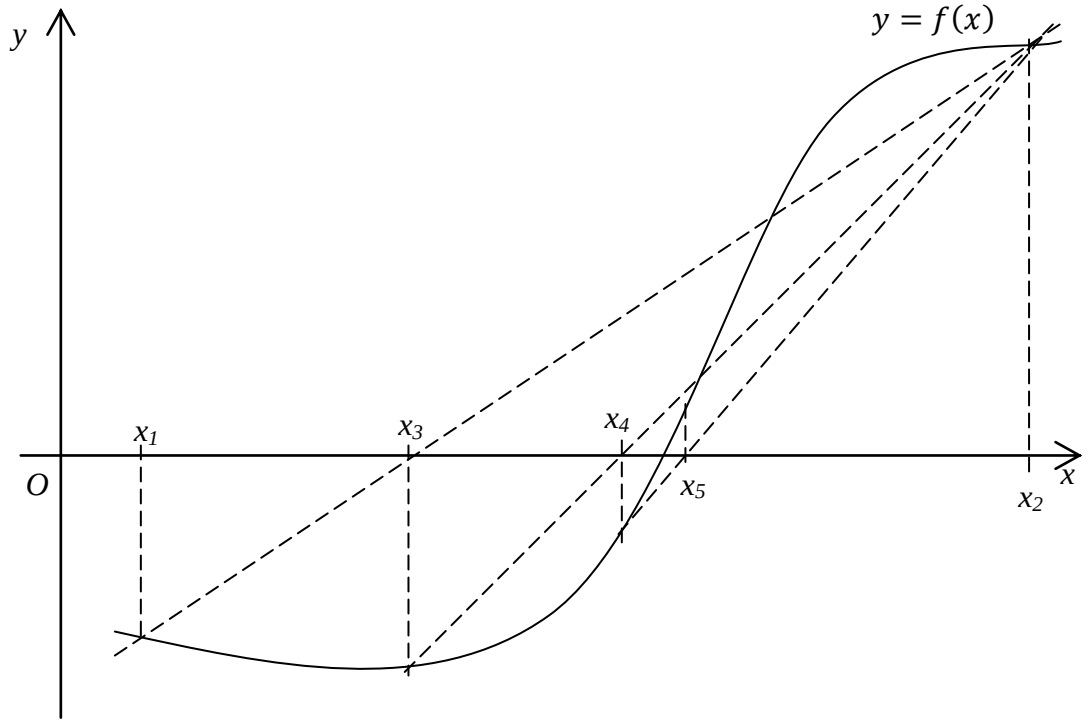
Vzniklá iterační metoda je zřejmě nestacionární. Rychlost konvergence není vysoká. V každém kroku se chyba zmenší na polovinu, platí tedy

$$|x_1 - x_2| = 2^{-n} r, \tag{9}$$

kde $|x_1 - x_2|$ je velikost získaného intervalu, r velikost původního intervalu a n počet půlení intervalu. Po 10 půleních se tedy původní interval zmenší více než 1000krát.

2.3 Metoda regula falsi

Viz [13]. Jedná se rovněž o jednoduchou metodu, která se podobá metodě půlení intervalu. Vyjdeme opět z intervalu $[x_1, x_2]$ takového, že platí $f(x_1) \cdot f(x_2) < 0$. Oproti metodě půlení intervalu se však nekonstruuje další bod v polovině intervalu, ale křivka daná rovnicí $y = f(x)$ se lokálně nahradí přímkou a průsečík této přímky s osou x bude další bod. Nahrazující přímka prochází body $[x_1, f(x_1)], [x_2, f(x_2)]$. Tím nám vzniknou dva intervaly a jako další vezmeme opět ten, jehož krajní body mají opačná znaménka funkčních hodnot.



Obr. 6. Metoda regula falsi.

Nastane-li situace tak, jako na (Obr. 6), sestrojíme x_4 pomocí intervalu $[x_3, x_2]$, bod x_5 pomocí intervalu $[x_4, x_2]$, x_6 pomocí intervalu $[x_4, x_5]$ atd., tj.

$$x_3 = \frac{f(x_2)}{f(x_2) - f(x_1)} x_1 + \frac{f(x_1)}{f(x_1) - f(x_2)} x_2, \tag{10}$$

$$x_4 = \frac{f(x_2)}{f(x_2) - f(x_3)} x_3 + \frac{f(x_3)}{f(x_3) - f(x_2)} x_2,$$

$$x_5 = \frac{f(x_2)}{f(x_2) - f(x_4)} x_4 + \frac{f(x_4)}{f(x_4) - f(x_2)} x_2$$

atd.

Jedná se opět o nestacionární metodu. Pro některé speciální funkce (např. konvexní funkce), je ale metoda regula falsi stacionární, neboť potom jeden krajní bod intervalu, z něhož určujeme další aproximaci, bude stále tentýž.

Rychlost konvergence metody je opět nejvýše lineární, tj. malá, tento fakt je ale vyvážen tím, že konverguje vždy.

2.4 Aitkenova – Steffensenova metoda

Iterační metody 1. řádu se dají urychlit podle následující úvahy [13], [5]. Pokud je ve vzorci (7) $m = 1$ a index k dostatečně velký, budou přibližně platit nerovnosti

$$\varepsilon_{k+1} \approx C\varepsilon_k,$$

$$x_{k+1} - x^* \approx C(x_k - x^*),$$

$$x_k - x^* \approx C(x_{k-1} - x^*),$$

z kterých vypočítáme kořen x^* (jen přibližně).

$$x^* \approx x = \frac{x_{k-1}x_{k+1} - x_k^2}{x_{k+1} - 2x_k + x_{k-1}} = x_{k-1} - \frac{(x_k - x_{k-1})^2}{x_{k+1} - 2x_k + x_{k-1}} \tag{11}$$

Použijeme-li tři získané iterace x_{k-1}, x_k a x_{k+1} z nějaké iterační metody, výsledek z rovnice (11) dává obvykle lepší aproximaci kořene, než kterákoli iterace použitá k jeho získání. Tento postup lze ovšem použít jen u iteračních metod 1. řádu.

Můžeme-li z rovnice (2) vyjádřit x, získáme rovnici ve tvaru $x = g(x)$.

Tzn. $\varphi(x) = g(x)$. Potom $x_k = g(x_{k-1})$, $x_{k+1} = g(x_k) = g\big(g(x_{k-1})\big)$. Hodnotu x z rovnice (11) budeme pokládat za další aproximaci kořene x^*. Dostaneme tak vzorec

$$x_k = x_{k-1} - \frac{(g(x_{k-1}) - x_{k-1})^2}{g(g(x_{k-1})) - 2g(x_{k-1}) + x_{k-1}}. \tag{12}$$

Iterační předpis daný vzorcem (12) se nazývá Aitkenova – Steffensenova metoda pro výpočet kořene rovnice $x = g(x)$. Pokud volíme počáteční aproximaci x_0 dostatečně blízko kořenu x^* a pokud $g'(x^*) \neq 1$, konverguje tato metoda kvadraticky. Pokud $g'(x^*) = 1$, konvergence je pomalá.

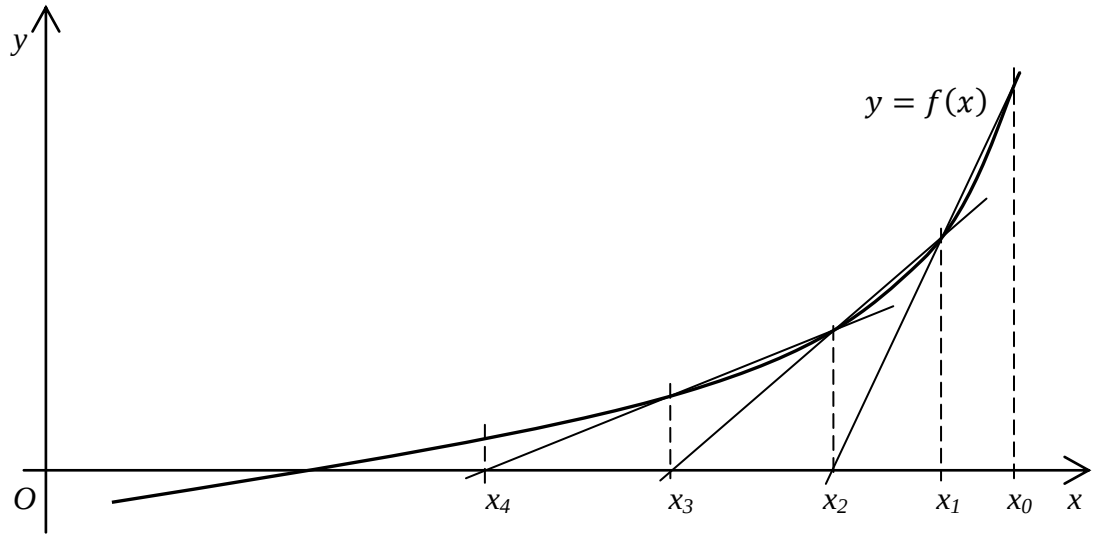
2.5 Metoda sečen

Metoda sečen [13], [3] vychází z metody regula falsi (lze ji i odvodit po aproximaci derivace v Newtonově metodě). Od metody regula falsi se liší tím, že funkce v krajních bodech intervalu, na němž se konstruuje další aproximace, nemusí mít opačná znaménka. Při výpočtu se použijí vždy dvě předchozí aproximace. Iterace jsou dány vzorcem

$$x_{k+1} = \frac{f(x_k)}{f(x_k) - f(x_{k-1})} x_{k-1} + \frac{f(x_{k-1})}{f(x_{k-1}) - f(x_k)} x_k. \tag{13}$$

Název metody pochází z její geometrické interpretace: x_{k+1} je x-ová souřadnice průsečíku osy x a přímky procházející body $[x_{k-1}, f(x_{k-1})]$ a $[x_k, f(x_k)]$, tj. sečna funkce $f(x)$. Iterační postup je znázorněn na (Obr. 7).

Jedná se už o stacionární metodu. Dá se odvodit, že pokud hledáme jednoduchý kořen, řád konvergence je $m = \frac{1}{2}(1 + \sqrt{5}) \approx 1{,}618$. Oproti metodě regula falsi konverguje podstatně rychleji (oproti Newtonově metodě pomaleji), nemusí ale konvergovat vždy.



Obr. 7. Metoda sečen.

2.6 Newton - Raphsonova metoda

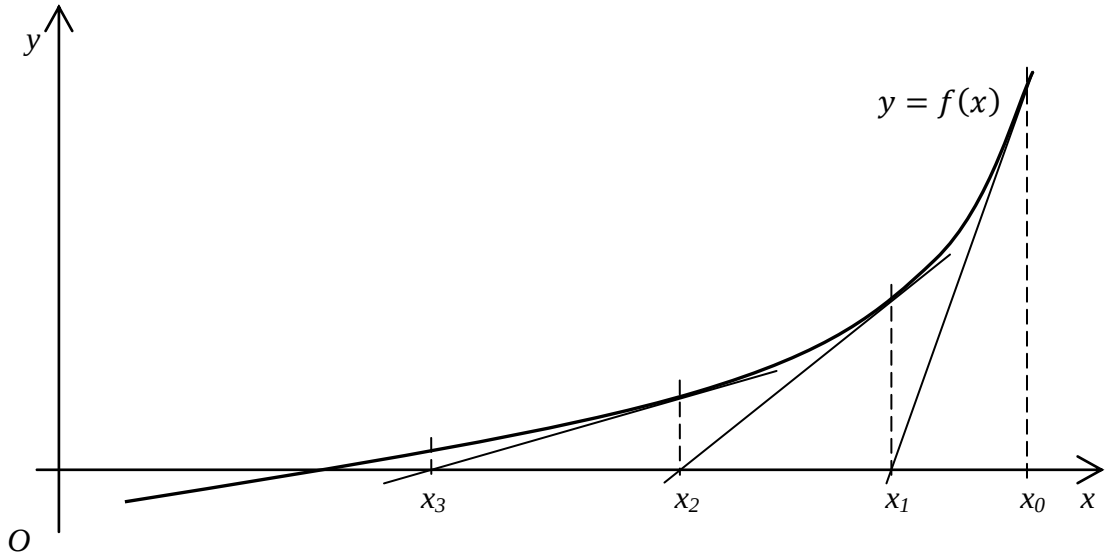
Newton - Raphsonova metoda, zkráceně Newtonova metoda, viz [12], [13], [5], [3], je jedna z nejpoužívanějších metod. V programu Mathematica je to výchozí metoda u příkazu FindRoot. Nazývá se také metoda tečen, díky geometrické představě. Princip je podobný jako u metody sečen, jen se nekonstruuje sečna ze dvou bodů, ale tečna z jednoho bodu. Postup lze vidět na (Obr. 8). Odvození iteračního předpisu pro tvorbu posloupnosti $\{x_n\}$ je následující:

Předpokládejme, že známe x_k. Bodem $[x_k, f(x_k)]$ vedeme tečnu ke křivce $y = f(x)$ a průsečík tečny s osou x považujeme za x_{k+1}. Do rovnice tečny

$$y = f(x_k) + f'(x_k)(x - x_k)$$

proto dosadíme $y = 0$, vyjádříme x a položíme $x_{k+1} = x$. Tím dostaneme předpis

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{14}$$



Obr. 8. Newtonova metoda (tečen).

Dá se odvodit, že Newtonova metoda konverguje vždy, pokud zvolíme počáteční aproximaci dostatečně blízko ke kořenu. (Toho dosáhneme nějakou jednodušší metodou, která konverguje vždy, např. metoda bisekce). V případě jednoduchého kořene konverguje Newtonova metoda kvadraticky, v případě vícenásobného kořene lineárně. Pro iterace v blízkosti kořene už přibližně platí, že se počet správných desetinných míst každou iterací zdvojnásobuje.

V případě větší vzdálenosti od kořene lze použít iteračního předpisu

$$x_{k+1} = x_k - \alpha \frac{f(x_k)}{f'(x_k)}, \tag{15}$$

kde $0 < \alpha \leq 1$. Tímto postupem můžeme zabránit divergenci metody způsobené nevhodnou počáteční aproximací.

Pokud chceme zachovat kvadratickou rychlost konvergence pro vícenásobný kořen, v předpisu (15) položíme α rovno násobnosti kořene. To lze dokázat složitějšími úvahami.

Vedle analytického vyčíslení derivace f' ji můžeme vyčíslit také numericky. Např. v případech, kdy by analytické derivování bylo složité či nemožné. Např. aproximací

$$f'(x_k) = \frac{f(x_k + h) - f(x_k)}{h} \tag{16}$$

pro vhodně malé h.

2.7 Steffensenova metoda

Od Newtonovy metody a vztahu (16) je již jen krůček k metodě Steffensenově [5], [3]. V aproximaci derivace funkce (16) označme h jako h_k a nebude to již znamenat číslo s konstantní hodnotou, ale číslo, které se s rostoucím indexem *k* blíží k nule. Budeme volit

$$h_k = \min(\varepsilon, |f(x_k)|), \tag{17}$$

kde funkce min(*M*) je rovna minimálnímu prvku množiny *M*, ε je daná tolerance (chyba) výpočtu kořene. Iterační předpis je potom ve tvaru

$$x_{k+1} = x_k - \frac{f(x_k)h_k}{f(x_k + h_k) - f(x_k)}. \tag{18}$$

Steffensenova metoda konverguje stejně rychle jako Newtonova metoda, tedy kvadraticky.

2.8 Čebyševovy metody

Iterační metody s vyšším řádem konvergence než dva už neodvodíme z geometrické interpretace nebo názoru. Takových metod je celá řada. Odvoďme metody jedné třídy, Čebyševovy iterační metody [12]:

Předpokládejme, že rovnice (2) $f(x) = 0$ má na intervalu $[a, b]$ kořen x^* a funkce $f(x)$ je spojitá. Potom k funkci $y = f(x)$ existuje inverzní funkce $x = F(y)$ na intervalu $[c, d]$, který je obrazem původního intervalu $[a, b]$. Pro navzájem inverzní funkce platí identity

$$x \equiv F[f(x)] \quad x \in [a,b], \tag{19}$$
$$y \equiv f[F(y)] \quad y \in [c,d].$$

Z toho plyne, že

$$x^* = F(0). \tag{20}$$

Taylorovým rozvojem v bodě y dostaneme

$$x^* = F(0) = F(y) + \sum_{n=1}^{r} (-1)^n \frac{F^{(n)}(y)}{n!} y^n + R, \tag{21}$$

kde R představuje zbytek Taylorova polynomu. Po dosazení vztahů y $= f(x)$ a $x = F(y)$ do (21) lze rovnost zapsat jako

$$x^* = x + \sum_{n=1}^{r} (-1)^n \frac{F^{(n)}[f(x)]}{n!} [f(x)]^n + R. \tag{22}$$

Pro jednoduchost zápisu položme

$$F^{(n)}[f(x)] = a_n(x). \tag{23}$$

Vytvoříme iterační funkci $\varphi_r(x)$

$$\varphi_r(x) = x + \sum_{n=1}^{r} (-1)^n \frac{a_n(x)}{n!} [f(x)]^n \tag{24}$$

a iterační proces bude dán předpisem

$$x_{k+1} = \varphi_r(x_k) \quad k = 0,1,2 \ldots \tag{25}$$

Řád iterace m tohoto procesu bude roven $m = r + 1$. Je-li x_0 zvoleno dostatečně blízko k x^*, potom je splněna podmínka (6)

$$|\varphi_r{}'(x)| < 1$$

a iterační proces bude konvergovat.

Funkci $\varphi_r(x)$ lze vyjádřit explicitně pomocí derivací $f, f', f'', \ldots$, neboť z rovnice (19) derivováním získáme

$$F'[f(x)]f'(x) = 1$$
$$F''[f(x)][f'(x)]^2 + F'[f(x)]f''(x) = 0$$

atd. Neboli podle označení (23)

$$a_1(x)f'(x) = 1 \tag{26}$$

$$a_2(x)[f'(x)]^2 + a_1(x)f''(x) = 0$$

$$a_3(x)[f'(x)]^3 + 3a_2(x)f'(x)f''(x) + a_1(x)f'''(x) = 0$$

atd. Z těchto vztahů lze zpětně nalézt $a_1(x)$, $a_2(x)$, ... a sestrojit z nich $\varphi_r(x)$. Pro $r = 1$ dostaneme $a_1(x) = 1/f'(x)$ a obdržíme tedy

$$\varphi_1(x) = x - \frac{f(x)}{f'(x)}$$

což odpovídá Newtonově metodě. Pro $r = 2$ získáme iterační předpis metody třetího řádu

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(x_k)f^2(x_k)}{2[f'(x_k)]^3}. \tag{27}$$

Pro $r = 3$ získáme metodu čtvrtého řádu

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(x_k)f^2(x_k)}{2[f'(x_k)]^3} - \frac{\{3[f''(x_k)]^2 - f'''(x_k)f'(x_k)\}f^3(x_k)}{6[f'(x_k)]^5}. \tag{28}$$

A tímto způsobem bychom mohli pokračovat dále. Čím vyššího řádu metoda bude, tím vyšší derivace bude nutné vyčíslovat.

2.9 Metoda tečných hyperbol

Odvoďme ještě jednu třídu metod vyšších řádů. Metoda se nazývá Halleyova nebo Richmondova metoda nebo metoda tečných hyperbol [12]. Jednotlivé iterace kořene funkce budeme hledat ve tvaru

$$x_{k+1} = x_k + c_k, \tag{29}$$

kde c_k představuje okolí bodu x_k a funkci $f(x)$ zde rozvedeme v Taylorovu řadu. Čím vyššího řádu rozvoj bude, tím bude aproximace přesnější a metoda bude konvergovat rychleji, daň za to je ovšem opět vyčíslování vyšších derivací, podobně jako u Čebyševovy metody. Podívejme se postupně na Taylorovy rozvoje funkce různých řádů.

2.9.1 Taylorův rozvoj 1. řádu:

$$f(x_k) + f'(x_k)c_1 = 0 \tag{30}$$

Po vyjádření c_1 z (30),

$$c_1 = -\frac{f(x_k)}{f'(x_k)}, \tag{31}$$

obdržíme Newtonovu metodu.

2.9.2 Taylorův rozvoj 2. řádu:

$$f(x_k) + f'(x_k)c_2 + \frac{1}{2}f''(x_k)c_2^2 = 0 \tag{32}$$

Kvadratickou rovnici (32) pro c_2 nelze řešit obecně (bez odmocnin), a proto aproximujeme kvadratický člen

$$c_2^2 \approx c_1 c_2,$$

kde c_1 je Newtonův přírůstek (31). Rovnice (32) má potom tvar

$$f(x_k) + f'(x_k)c_2 + \frac{1}{2}f''(x_k)c_2\left[-\frac{f(x_k)}{f'(x_k)}\right] = 0.$$

Řešením této lineární rovnice pro c_2 dostaneme Richmondovu (Halleyovu) metodu třetího řádu

$$x_{k+1} = x_k - \frac{2f(x_k)f'(x_k)}{2[f'(x_k)]^2 - f(x_k)f''(x_k)}. \tag{33}$$

Když bychom do rovnice (32) dosadili za kvadratický člen $c_2^2 \approx c_1^2$ z Newtonovy metody, získáme Čebyševovu metodu třetího řádu, rovnice (27). To vysvětluje, proč Richmondova iterace obvykle konverguje rychleji než Čebyševova.

Tab. 1. Srovnání metod pro rovnici $x^3 - 10 = 0$.

k	Metoda		
	Newtonova	Čebyševova	Richmondova
0	2,000 000 000 000 000	2,000 000 000 000 000	2,000 000 000 000 000
1	2,166 666 666 666 667	2,152 777 777 777 778	2,153 846 153 846 154
2	2,154 503 616 042 078	2,154 434 688 394 754	2,154 434 690 002 592
3	2,154 434 692 236 913	2,154 434 690 031 884	2,154 434 690 031 884
4	2,154 434 690 031 884		

V tabulce (Tab. 1) můžeme srovnat iterační procesy Newtonovy, Čebyševovy a Richmondovy metody pro rovnici $x^3 - 10 = 0$, při $x_0 = 2$. Počet platných desetinných míst je vyznačen podtržením. Newtonova metoda konverguje kvadraticky a počet platných míst se každou iterací přibližně zdvojnásobuje. Čebyševova a Richmondova metoda konvergují kubicky (jsou třetího řádu) a počet platných míst se přibližně ztrojnásobuje, přičemž Richmondova metoda konverguje o něco rychleji.

2.9.3 Taylorův rozvoj 3. řádu:

$$f(x_k) + f'(x_k)c_3 + \frac{1}{2}f''(x_k)c_3^2 + \frac{1}{6}f'''(x_k)c_3^3 = 0 \tag{34}$$

Zde nahradíme

$$c_3^2 \approx c_2 c_3,$$

$$c_3^3 \approx c_2^2 c_3$$

a po dosazení do (34) a vyjádření c_3 dostaneme

$$c_3 = \frac{-f(x_k)}{f'(x_k) + \frac{1}{2}f''(x_k)c_2 + \frac{1}{6}f'''(x_k)c_2^2} \tag{35}$$

atd., funkci můžeme rozvinout do libovolného řádu.

2.9.4 Taylorův rozvoj *k* - tého řádu:

$$f(x_k) + f'(x_k)c_k + \frac{1}{2}f''(x_k)c_k^2 + \frac{1}{6}f'''(x_k)c_k^3 + \ldots + \frac{1}{k!}f^{(k)}(x_k)c_k^k = 0 \tag{36}$$

Nahradíme

$$c_k^2 \approx c_{k-1}c_k,\ c_k^3 \approx c_{k-1}^2 c_k, \ldots, c_k^k \approx c_{k-1}^{k-1}c_k.$$

Odtud

$$c_k = \frac{-f(x_k)}{f'(x_k) + \frac{1}{2}f''(x_k)c_{k-1} + \frac{1}{6}f'''(x_k)c_{k-1}^2 + \ldots + \frac{1}{k!}f^{(k)}c_{k-1}^{k-1}} \tag{37}$$

Takto můžeme tvořit metody stále vyšších a vyšších řádů podle předpisu (29).

2.10 Metoda inverzní kvadratické interpolace

Metodu inverzní kvadratické interpolace, viz [3], [4], můžeme odvodit z následující myšlenky. U metody sečen se další bod iterace konstruuje jako průsečík osy *x* s přímkou,

která je dána dvěma body funkce. Pokud použijeme z funkce tři body, můžeme sestrojit parabolu, a aproximace funkce bude přesnější. Na tomto principu je založena Müllerova metoda. Potíž může být v tom, že parabola nemusí osu *x* protnout. Proto musí Müllerova metoda provádět výpočet v komplexní aritmetice. Abychom se tomu vyhnuli, nepoložíme parabolu v proměnné *x*, ale v proměnné *y*. Odtud slovo „inverzní" v názvu metody. Tato parabola vždy protíná osu *x*. Průsečík najdeme, když položíme $y = 0$. Viz (Obr. *9*). Algoritmus metody realizujeme takto: Na začátku zvolíme body x_1 a x_2, mezi kterými hledáme nulový bod funkce $f(x)$. Bod x_3 zvolíme v polovině intervalu $[x_1, x_2]$, $x_3 = 0{,}5(x_1+x_2)$. Funkční hodnoty v těchto bodech označme y_1 až y_3. Těmito třemi body $[x_1, y_1]$, $[x_2, y_2]$ a $[x_3, y_3]$ proložíme parabolu v proměnné *x* pomocí Lagrangeova interpolačního polynomu druhého řádu $L_2(y)$.

$$x = L_2(y) = \tag{38}$$

$$= x_1 \frac{(y-y_2)(y-y_3)}{(y_1-y_2)(y_1-y_3)} + x_2 \frac{(y-y_1)(y-y_3)}{(y_2-y_1)(y_2-y_3)} + x_3 \frac{(y-y_1)(y-y_2)}{(y_3-y_1)(y_3-y_2)}$$

Hledáme takové *x*, že $f(x) = y = 0$, proto

$$x = L_2(0) = \tag{39}$$

$$= x_1 \frac{y_2 y_3}{(y_1-y_2)(y_1-y_3)} + x_2 \frac{y_1 y_3}{(y_2-y_1)(y_2-y_3)} + x_3 \frac{y_1 y_2}{(y_3-y_1)(y_3-y_2)}.$$

Vztah (39) se dá rozepsat do podoby

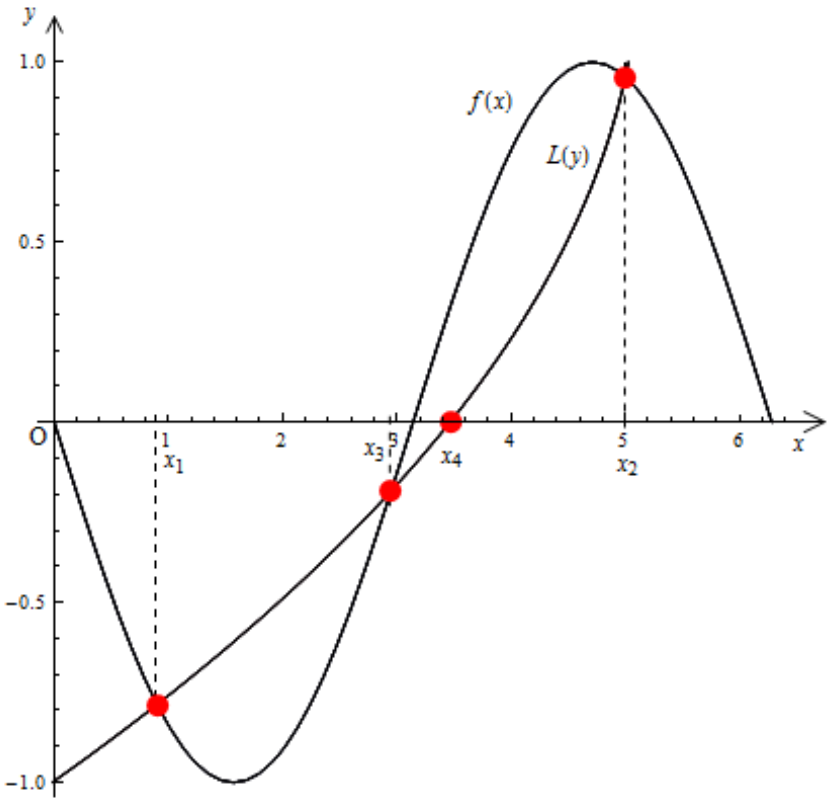
$$x = x_3 + \frac{P}{Q}, \tag{40}$$

kde

$$R = \frac{y_3}{y_2}, S = \frac{y_3}{y_1}, T = \frac{y_1}{y_2}, Q = (T-1)(R-1)(S-1),$$

$$P = S[T(R-T)(x_2-x_3) - (1-R)(x_3-x_1)]$$

Vypočítané x ze vztahu (39) nebo (40) označme jako x_4. Další parabolu proložíme body x_2, x_3 a x_4 a celý postup opakujeme. Řád konvergence této metody $m \approx 1{,}839$, rychlost konvergence je tedy superlineární.



Obr. 9. Metodu inverzní kvadratické interpolace.

2.11 Brentova metoda

Metoda půlení intervalu spolu s metodou sečen a metodou inverzní kvadratické interpolace jsou základem Brentovy metody [3], [4], [2]. V programu Mathematica tuto metodu používá funkce FindRoot po zadání dvou počátečních hodnot. V programu Matlab ji používá funkce fzero. Výhoda Brentovy metody je, že nepoužívá derivace funkce, zaručeně konverguje ke kořenu, tzn. je spolehlivá a po několika počátečních krocích nabývá superlineární rychlosti konvergence, tzn. chyba se rychle zmenšuje. Dva startovací body musíme zvolit tak, aby jejich funkční hodnoty měly opačná znaménka. První iterace se provede metodou sečen, či metodou bisekce, má – li lepší výsledek. V dalších krocích je přednostně použita metoda inverzní kvadratické interpolace. Pokud dostaneme lepší výsledek metodou sečen, použijeme tuto metodu. Nakonec se určí, jestli nedostaneme lepší aproximaci kořene metodou půlení intervalu a pokud ano, použijeme ji. Rychlost

konvergence metody je proto nejvýše superlineární jako u metody inverzní kvadratické interpolace. Přesný algoritmus metody je popsán v praktické části práce.

2.12 Metoda prosté iterace

Metoda prosté iterace [3], nazývaná také metoda postupných aproximací, je založena na myšlence převést rovnici (2) $f(x) = 0$ na ekvivalentní tvar

$$x = g(x). \tag{41}$$

Funkce $g(x)$ tedy odpovídá iterační funkci $\varphi(x)$ ze vztahu (4). Posloupnost postupných aproximací kořene budeme tvořit (podle vztahu (5))

$$x_k = g(x_{k-1}), \qquad k = 1,2, \dots \tag{42}$$

Problém této metody je to, že nekonverguje vždy. Jako postačující podmínky konvergence použijeme nerovnost (43) či silnější (44). Viz (1) a (6).

$$|g(x) - g(y)| \leq \alpha|x - y|, \qquad 0 \leq \alpha < 1 \tag{43}$$

$$|g'(x)| \leq K < 1 \tag{44}$$

Aby posloupnost konvergovala, musí nerovnosti platit pro všechna x, y z intervalu, na kterém hledáme řešení.

Vhodnou úpravou rovnice (2) na rovnici (41) můžeme dostat řadu různých konkrétních metod. Např. pro $g(x) = x - f(x)/f'(x)$ dostaneme Newtonovu metodu.

Rychlost konvergence metody závisí na chování funkce $g(x)$ v nulovém bodě x^*. Je – li splněna podmínka (43) nebo (44), dají se dokázat následující tvrzení.

- Pokud $g'(x^*) \neq 0$, řád konvergence je roven jedné a platí $|x_k - x^*| \leq \alpha|x_{k+1} - x^*|$.
- Pokud $g'(x^*) = 0$ a $g''(x^*) \neq 0$, řád konvergence je roven dvěma.
- Obecně, pokud jsou derivace $g^{(s)}(x^*) = 0, s = 1,2 \dots, r-1$ a $g^{(r)}(x^*) \neq 0$, konvergence je řádu r.

Zkusme to ověřit na následujícím příkladu. Mějme nelineární rovnici

$f(x) = x^2 - 2x - 3 = 0$ s kořeny $x_1^* = -1$ a $x_2^* = 3$. Prozkoumejme konvergenci ke kořenu $x_2^* = 3$ pro několik iteračních funkcí $g(x)$.

1. $g(x) = (x^2 - 3)/2$, $g'(x) = x$, $|g'(3)| = 3$, pro $x_0 \neq 3$ nenastane konvergence.
2. $g(x) = \sqrt{2x+3}$, $g'(x) = 1/\sqrt{2x+3}$, $|g'(3)| = 1/3$, nastane lineární konvergence. Např. pro x_0 z intervalu $[2;4]$, v němž $|g'(x)| \leq 1/\sqrt{7}$.
3. $g(x) = 2 + 3/x$, $g'(x) = -3/x^2$, $|g'(3)| = 1/3$, nastane lineární konvergence. Např. pro x_0 z intervalu $[2;4]$, v němž $|g'(x)| \leq 3/4$.
4. $g(x) = (x^2 + 3)/(2x - 2)$, $g'(x) = 2(x^2 - 2x - 3)/(2x - 2)^2$, $|g'(3)| = 0$, $g''(3) = 1/2$, nastane kvadratická konvergence. Např. pro x_0 z intervalu $[2{,}5; 3{,}5]$, v němž $|g'(x)| < 0{,}39$. Tato iterační funkce odpovídá Newtonově metodě.

2.13 Iterační metody pro násobné kořeny

U zmíněných iteračních metod hraje také roli násobnost kořene [13], [3]. Všechna předchozí tvrzení o rychlostech konvergence metod platí pouze v případě, že aproximujeme jednoduchý kořen rovnice (2). Použijeme – li tyto metody pro kořen násobnosti větší než jedna, zůstane za předpokladu, že výchozí aproximace byla zvolena dostatečně blízko k hledanému kořenu, jejich konvergence většinou zachována, jejich řád konvergence však klesne na jedničku.

Pokud známe násobnost kořene, dá se většina metod modifikovat tak, že jejich řád konvergence zůstane zachován. Např. speciálně Newtonova metoda:

$$x_{k+1} = x_k - r\frac{f(x_k)}{f'(x_k)}, \tag{45}$$

kde r je příslušná násobnost. Konkrétní modifikace jiných metod nebudeme uvádět, protože jejich význam je značně omezen. Hlavně proto, že násobnost hledaného kořene většinou není známa.

Metody však můžeme jednoduše univerzálně upravit tak, aby jejich řád konvergence zůstal zachován a byl nezávislý na násobnosti kořene. Vyjdeme z poznatku, že položíme – li

$$u(x) = \frac{f(x)}{f'(x)}, \tag{46}$$

jsou kořeny rovnice

$$u(x) = 0$$

stejné jako kořeny rovnice (2) a jsou všechny jednoduché. K tomu abychom dostali metodu, jejíž řád iterace nebude záviset na násobnosti kořene, stačí tedy pouze v předchozí

metodě zaměnit funkci $f(x)$ za $u(x)$. Nevýhodou může být, že musíme počítat jednu derivaci navíc.

2.14 Dosažitelná přesnost nulového bodu

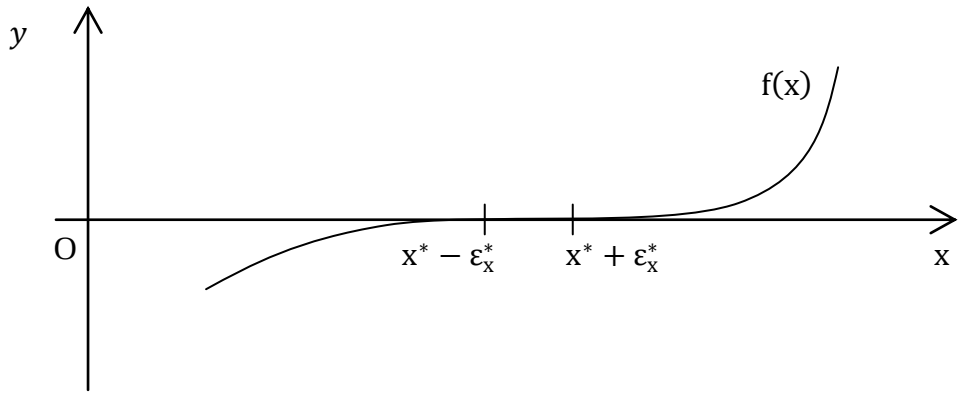
V obecném případě můžeme numerickými metodami získat pouze přibližnou aproximaci nulového bodu. k - tá aproximace kořene je limitována dosažitelnou přesností kořene [3]. Pokud je x_k aproximace jednoduchého kořene rovnice (2), $f(x) = 0$, pomocí věty o střední hodnotě obdržíme

$$f(x_k) = f(x_k) - f(x^*) = f'(\xi)(x_k - x^*),$$

kde ξ je nějaký bod ležící mezi x_k a x^*. Za předpokladu, že při výpočtech pracujeme jen s přibližnými hodnotami $\varphi(x_k) = f(x_k) + \delta_k$, přičemž $\delta_k \leq \delta$, pak nejlepší výsledek, jakého můžeme dosáhnout, je $\varphi(x_k) = 0$. V tomto případě $|f(x_k)| \leq \delta$, takže potom

$$|x_k - x^*| = \frac{|f(x_k)|}{|f'(\xi)|} \leq \frac{\delta}{|f'(\xi)|} \approx \frac{\delta}{|f'(x^*)|} = \varepsilon_x^*, \tag{47}$$

pokud se f' v blízkosti kořene příliš nemění. Vyčíslit x^* s menší chybou než ε_x^* nelze. ε_x^* se nazývá *dosažitelná přesnost kořene* x^*. Všimněme si, že když je velikost směrnice $|f'(x^*)|$ v kořenu x^* malá, dosažitelná přesnost ε_x^* je velká, viz (*Obr. 10*). V tom případě je výpočet kořene x^* *špatně podmíněný problém*.



Obr. 10. Dosažitelná přesnost kořene.

Podobnou úvahou pro kořen násobnosti r dostaneme dosažitelnou přesnost

$$\varepsilon_x = \left(\frac{\delta\, r!}{f^{(r)}(x^*)}\right)^{1/r}. \tag{48}$$

Exponent $1/r$ zapříčiňuje to, že výpočet násobného kořene je obecně špatně podmíněná úloha. Např. pro $f(x) = x^r$ je $x^* = 0$ kořen násobnosti r a $\varepsilon_x = \delta^{1/r}$. Pro $r = 15$ a $\delta = 10^{-15}$ dostaneme $\varepsilon_x = 0{,}1$!

2.15 Kořeny polynomů

Polynom

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ \ldots + a_1 x + a_0 \tag{49}$$

stupně n má obecně n komplexních kořenů [13], [3]. Všechny popsané iterační metody lze použít k hledání reálných kořenů funkce $f(x) = p_n(x)$. Pro zrychlení konvergence metody při násobných kořenech můžeme metodu upravit vztahem (46). V praxi nás často zajímají i komplexní kořeny. Pro nalezení komplexních kořenů lze některé popsané metody mírně upravit nebo použít jiné metody zaměřené speciálně na tento problém, viz [13].

Podívejme se jen na nejznámější Newtonovu metodu. Není zde třeba žádných úprav. Pro výpočet komplexních kořenů stačí zvolit jako počáteční aproximaci komplexní číslo a pak pracovat v komplexní aritmetice.

V případě že nás zajímají všechny kořeny polynomu, tak po nalezení reálného kořene x^* polynom $p_n(x)$ vydělíme členem $(x - x^*)$ a stupeň polynomu se sníží o jedna. Po nalezení komplexního kořene x^*, je také kořenem komplexně sdružené číslo $\bar{x}^*$. Potom dělíme polynom $p_n(x)$ kvadratickým polynomem $(x - x^*)(x - \bar{x}^*)$, jehož koeficienty jsou reálná čísla. Tím se nám sníží stupeň polynomu $p_n(x)$ o dva a pokračujeme dál v hledání.

3 ŘEŠENÍ SOUSTAV NELINEÁRNÍCH ROVNIC

Problematika řešení soustavy nelineárních rovnic iteračními metodami je formálně podobná problematice řešení jedné rovnice. Mnohé z metod určených pro řešení jedné rovnice lze zobecnit na řešení soustavy rovnic [3], [12], [13]. Nemilé je, že to neplatí pro metodu bisekce ani pro metodu regula falsi. Co více, pro soustavy nelineárních rovnic nebyla objevena žádná univerzální metoda, která by dokázala spolehlivě určit dostatečně dobrou počáteční aproximaci. Vyhovující počáteční aproximaci proto musíme odhadnout. Pomoci nám může znalost konkrétního problému nebo odhad řešení za zjednodušených předpokladů, např. aproximací nelineárního problému vhodným problémem lineárním.

Mějme soustavu n nelineárních rovnic o n neznámých

$$\begin{matrix} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{matrix} \quad \text{neboli} \quad \mathbf{f(x)} = \mathbf{o}, \tag{50}$$

kde

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{f(x)} = \begin{pmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{pmatrix} \equiv \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix} \quad \text{a} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Řešením soustavy (50) je každý číselný vektor $\mathbf{x}^*$, pro který platí $\mathbf{f(x^*)} = \mathbf{o}$.

3.1 Newtonova metoda

Zobecněním Newtonovy metody pro jednu rovnici získáme Newtonovu metodu pro soustavu rovnic [3] [12] [13].

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}^{-1}(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) \tag{51}$$

Kde $\mathbf{J}(\mathbf{x})$ je Jacobiho matice:

$$\mathbf{J}(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \dfrac{\partial f_2(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial f_n(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_n(\mathbf{x})}{\partial x_n} \end{pmatrix} \tag{52}$$

Iterační předpis (51) lze přepsat do podoby

$$\mathbf{J}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + \mathbf{f}(\mathbf{x}_k) = \mathbf{o} \tag{53}$$

a výpočet organizovat tak, že nejdříve vyřešíme soustavu lineárních rovnic

$$\mathbf{J}(\mathbf{x}_k)\mathbf{d}_\mathrm{k} = -\,\mathbf{f}(\mathbf{x}_k) \text{ a pak určíme } \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_\mathrm{k} \tag{54}$$

Newtonova metoda konverguje, pokud počáteční aproximace $\mathbf{x}_0$ byla zvolena dostatečně blízko kořene $\mathbf{x}^*$ a rychlost konvergence je kvadratická. Tyto vlastnosti jsou tedy stejné jako v jednodimenzionálním případě.

3.2 Diskretizovaná Newtonova metoda

Pokud nechceme analyticky vyčíslovat parciální derivace v Jacobiho matici, můžeme je aproximovat diferenčními podíly

$$\frac{\partial f_i(\mathbf{x})}{\partial x_j} \approx \Delta_{ij}(\mathbf{x}, \mathbf{h}) = \frac{f_i(x_1, \ldots, x_j + h_j, \ldots, x_n) - f_i(\mathbf{x})}{h_j},$$

kde $h_j \neq 0$ jsou vhodně zvolené parametry, vektor $\mathbf{h} = (h_1, h_2, \ldots, h_n)^\mathrm{T}$. Pro malé $h_j > 0$ je $\Delta_{ij}(\mathbf{x}, \mathbf{h})$ standardní aproximace $\partial f_i(\mathbf{x}) / \partial x_j$ dopřednou diferencí. Matice $\boldsymbol{\Delta}(\mathbf{x}, \mathbf{h})$ s prvky $\Delta_{ij}(\mathbf{x}, \mathbf{h})$ je aproximací Jacobiovy matice $\mathbf{J}(\mathbf{x})$. Po nahrazení v (53) $\mathbf{J}(\mathbf{x}_k)$ pomocí $\boldsymbol{\Delta}(\mathbf{x}_k, \mathbf{h}_k)$, dostaneme diskretizovanou Newtonovu metodu [3]

$$\boldsymbol{\Delta}(\mathbf{x}_k, \mathbf{h}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + \mathbf{f}(\mathbf{x}_k) = \mathbf{o}. \tag{55}$$

3.3 Zobecněná metoda sečen

Zobecněnou metodu sečen [3] získáme, když j – tou složku vektoru $\mathbf{h}_k$ nahradíme

$$h_j^{(k)} = x_j^{(k-1)} - x_j^{(k)}. \tag{56}$$

Pro $n = 1$ přejde rovnice (56) do tvaru (13). Tato metoda potřebuje dvě dostatečně dobré počáteční aproximace $\mathbf{x}_0$ a $\mathbf{x}_1$. Řád iterace je stejný jako v jedné dimenzi, $m = 1{,}618$.

3.4 Zobecněná Steffensenova metoda

Zobecněnou Steffensenovu metodu [3] dostaneme, když v diskretizované Newtonově metodě položíme

$$h_j^{(k)} = f_j(\mathbf{x}_k). \tag{57}$$

V případě $n = 1$ bude iterační předpis (56) stejný jako (18). Řád iterace je opět stejný jako v jednodimenzionálním případě, $m = 2$.

3.5 Metoda prosté iterace

Dříve popsanou metodu prosté iterace lze zobecnit i pro soustavu nelineárních rovnic [3]. V mnoha aplikacích se můžeme setkat s nelineární soustavou

$$\mathbf{x} = \mathbf{a} + h\boldsymbol{\rho}(\mathbf{x}), \tag{58}$$

kde $\mathbf{a}$ je číselný vektor a h je malé kladné číslo. Pokud převedeme vše na jednu stranu a označíme soustavu $\mathbf{f}(\mathbf{x}) = \mathbf{x} - \mathbf{a} - h\boldsymbol{\rho}(\mathbf{x})$, můžeme soustavu $\mathbf{f}(\mathbf{x}) = \mathbf{o}$ vyřešit výše popsanými metodami, např. Newtonovou.

Pokud je však úloha ve tvaru (58) můžeme použít jiný, velmi jednoduchý postup. Označme $\mathbf{g}(\mathbf{x}) = \mathbf{a} + h\boldsymbol{\rho}(\mathbf{x})$ a úlohu můžeme zapsat ve tvaru

$$\mathbf{x}^* = \mathbf{g}(\mathbf{x}^*), \tag{59}$$

kde kořen $\mathbf{x}^*$ odpovídá pevnému bodu zobrazení $\mathbf{g}(\mathbf{x})$.

Úlohu (59) zkusíme vyřešit metodou prosté iterace: zvolíme počáteční aproximaci $\mathbf{x}_0$ a počítáme

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k), \quad k = 0, 1, \ldots \tag{60}$$

Aby posloupnost konvergovala, musí být splněna podmínka

$$\left\|\mathbf{J}_\mathbf{g}(\mathbf{x})\right\| \leq K < 1, \tag{61}$$

kde $\mathbf{J}_\mathbf{g}(\mathbf{x})$ značí Jacobiho matici soustavy $\mathbf{g}(\mathbf{x})$ a závorky $\|\cdot\|$ normu. Rychlost obecně lineární konvergence závisí na K.

Vraťme se k úloze (58). Pokud má funkce $\boldsymbol{\rho}(\mathbf{x})$ v okolí kořene $\mathbf{x}^*$ ohraničené parciální derivace, potom pro dostatečně malé h a pro $\mathbf{x}_0$ dostatečně blízké k $\mathbf{x}^*$ posloupnost postupných aproximací

$$\mathbf{x}_{k+1} = \mathbf{a} + h\boldsymbol{\rho}(\mathbf{x}_k), \quad k = 0, 1, \ldots \tag{62}$$

konverguje k $\mathbf{x}^*$.

II. PRAKTICKÁ ČÁST

4 IMPLEMENTACE METOD

Metody popsané v teoretické části práce a výpočty spojené s aplikacemi, které jsou uvedené v praktické části, jsem realizoval v prostředí Mathematica 7 for Students od společnosti Wolfram Research. Stručně popíšu software Mathematica a ukážu realizaci jedné metody. Soubory - notebooky s ostatními metodami jsou přiložené na CD.

4.1 Programový systém Mathematica

Po zhruba dvacetiletém vývoji je v současnosti (2010) poslední verze Mathematica 7 [16]. V systému lze provádět běžné výpočty, ale i modelování, simulace, vizualizace, vývoj a dokumentace. Po přidání dalších aplikačních knihoven lze možnosti rozšířit požadovaným směrem. Knihovny tvoří jak Wolfram Research, tak nezávislí výrobci. Některé aplikační knihovny od společnosti Wolfram Research:

- **Control System Professional.** Nabízí objektově orientované prostředí pro řešení běžných problémů v oblastech řízení a systémů.
- **Dynamic Visualizer**. Umožňuje tvořit prostorovou grafiku v reálném čase.
- **Fuzzy Logic.** Flexibilní prostředí pro práci s fuzzy systémy.
- **Mechanical Systems.** Analýza kinematiky a dynamiky systémů těles.
- **Neural Networks.** Učení a analýza neuronových sítí.
- **Structural Mechanics.** Experimentování a zpracování problémů elastických systémů a konečných prvků s úplnými symbolickými schopnostmi.
- **Time series.** Plně integrované prostředí pro časově závislou datovou analýzu.
- **Wavelet Explorer**. Analýza a zpracování signálů a obrazů.

Některé aplikační knihovny od jiných výrobců **Chyba! Nenalezen zdroj odkazů.**]:

- **Analog Insydes**. Inteligentní systém na symbolické návrhy analogových obvodů.
- **Derivatives Expert.** Inovační software k analýze cenných papírů a derivátů.
- **Geometrica.** Přesné rýsování a geometrie.
- **Operations Research.** Knihovna na řešení široké řady problémů v optimalizaci.
- **Schematic Solver**. Schematické zobrazení, symbolické řešení, zpracování a implementace analogových a digitálních systémů.
- **Tensors in Physics**. Výpočty tenzorů se stovkami či tisíci komponentů.
- **Mathematica Link for Excel.** Napojení Mathematica pro Excel.

4.2 Realizace Brentovy metody

Většina popsaných iteračních metod má velmi jednoduchý algoritmus provedení. Program běží ve smyčce, kde se v každém kroku přičte k aktuální aproximaci kořene spočtený iterační přírůstek a uloží se jako nová aproximace kořene, viz (3). Po dosažení požadované přesnosti se cyklus ukončí. Průběh vypadá např. jako na obrázku (Obr. 5).

Proto znázorním realizaci Brentovy metody, která si v každém kroku vybírá jednu ze třech jednodušších metod a algoritmus je zajímavější. Jak už bylo napsáno, Brentovu metodu používají systém Matlab, Mathematica, aj. při řešení nelineárních rovnic.

Níže je zápis zdrojového kódu v Mathematice. Na obrázcích (Obr. 12) až (Obr. 14) je algoritmus znázorněn ve vývojovém diagramu.

Zdrojový kód Brentovy metody:

```
1 fce[q_] := (q + 3)*((q - 1)^2)  (*vyhodnocovaná funkce f (q)=0*)
2 a = -4; (*startovací body a, b*)
3 b = 4/3; (*Funkce musí mít v a, b opačná znaménka*)
4 δ = 0.0001; (*přesnost*)
5 Plot[fce[x], {x, a, b}] (*zobrazení situace v grafu*)
```

Zadání inicializačních údajů: funkce, startovacích bodů a přesnosti. Funkce se zobrazí v grafu.

```
6 NeniMezi[c_, a_, b_] := If[a < b,
7   If[c < a || c > b, 1, 0],
8   If[c < b || c > a, 1, 0]
9 ];
```

Funkce *NeniMezi* vrací 1, pokud *c* není mezi *a* a *b*, jinak vrátí 0.

```
10 SetAttributes[Swap, HoldAll]
11 Swap[x_, y_] := (pom = y;
12   y = x;
13   x = pom;
14 )
```

Funkce *Swap* prohodí hodnoty *x* a *y*.

```
15 fa = fce[a]; (*vyhodnocení funkce v bodě a*)
16 fb = fce[b]; (*vyhodnocení funkce v bodě b*)
17 If[fa*fb ≥ 0, (*Pokud není splněna počáteční podmínka, tak stop*)
18 Print["Chyba! Funkční hodnoty v počátečních bodech musí mít opačná
 znaménka."],
19 If[Abs[fa] < Abs[fb], Swap[a, b]; Swap[fa, fb]];
```

```
20 (*Absolutní hodnota funkční hodnoty v bodě a musí být větší než v
 b.*)
21 c = a; (*Třetí bod c je potřeba pro konstrukci paraboly.*)
22 mflag = 1; (*Příznak použití metody bisekce v předchozím kroku.*)
```

Vyhodnocení funkcí, test počáteční podmínky, nachystání se na iterační cyklus, který následuje:

```
23 Do[
24 fc = fce[c];
25 If[fb == 0 || fs == 0 || Abs[b - a] < δ, Break[]];
```

Pokud jsme dostatečně blízko, tak se cyklus ukončí.

```
26 If[fa != fc && fb != fc,
```

Funkční hodnoty bodů *a,b* jsou různé od *c*, má smysl konstruovat parabolu.

```
27 s = (a*fb*fc)/((fa - fb)*(fa - fc)) +
28 (b*fa*fc)/((fb - fa)*(fb - fc)) + (c*fa*fb)/((fc - fa)*(fc - fb));,
```

Metoda inverzní kvadratické interpolace.

```
29 s = (fb*a - b*fa)/(fb - fa);
```

Jinak je použita metoda sečen.

```
30 ];
31 If[NeniMezi[s, (3 a + b)/4, b] == 1 ||
32 (mflag == 1 && Abs[s - b] >= Abs[b - c]/2) ||
33 (mflag == 0 && Abs[s - b] >= Abs[c - d]/2) ||
34 (mflag == 1 && Abs[b - c] < Abs[δ]) ||
35 (mflag == 0 && Abs[c - d] < Abs[δ]),
```

Podmínky, při jejichž splnění dá metoda bisekce lepší výsledek než předchozí dvě metody.

```
36 s = (a + b)/2;
```

Metoda půlení intervalu.

```
37 mflag = 1;, (*příznak - metoda použita*)
38 mflag = 0   (*metoda bisekce nepoužita*)
39 ];
40 fs = fce[s]; (*Vyhodnocení funkce v novém bodě.*)
41 d = c;   (*Paměť d: c v minulém kroku, kvůli vyhodnocení úspěšnosti
 metody.*)
42 c = b;
43 If[fa*fs < 0, (*Dále budeme pracovat v tom intervalu, kde je
 kořen.*)
44   b = s;
45   fb = fce[b];
```

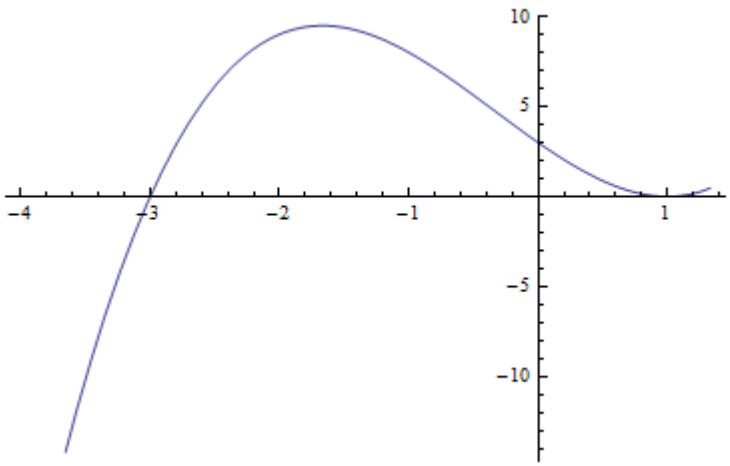
```
46 , a = s;
47 fa = fce[a];
48 ];
```

Zmenšení intervalu tak, aby v něm zůstal kořen.

```
49 If[Abs[fa] < Abs[fb], Swap[a, b]; Swap[fa, fb]];(*jako na začátku*)
50 Print[s // N] (*Výpis iterací.*)
51 , {15}] (*Konec Do, číslo znamená maximální počet cyklů.*)
52 ] (*konec If počáteční kontroly*)
```

Zajištění aby v *a* byla větší funkční hodnota než v *b* (v absolutní hodnotě), výpis výsledku a konec cyklu.

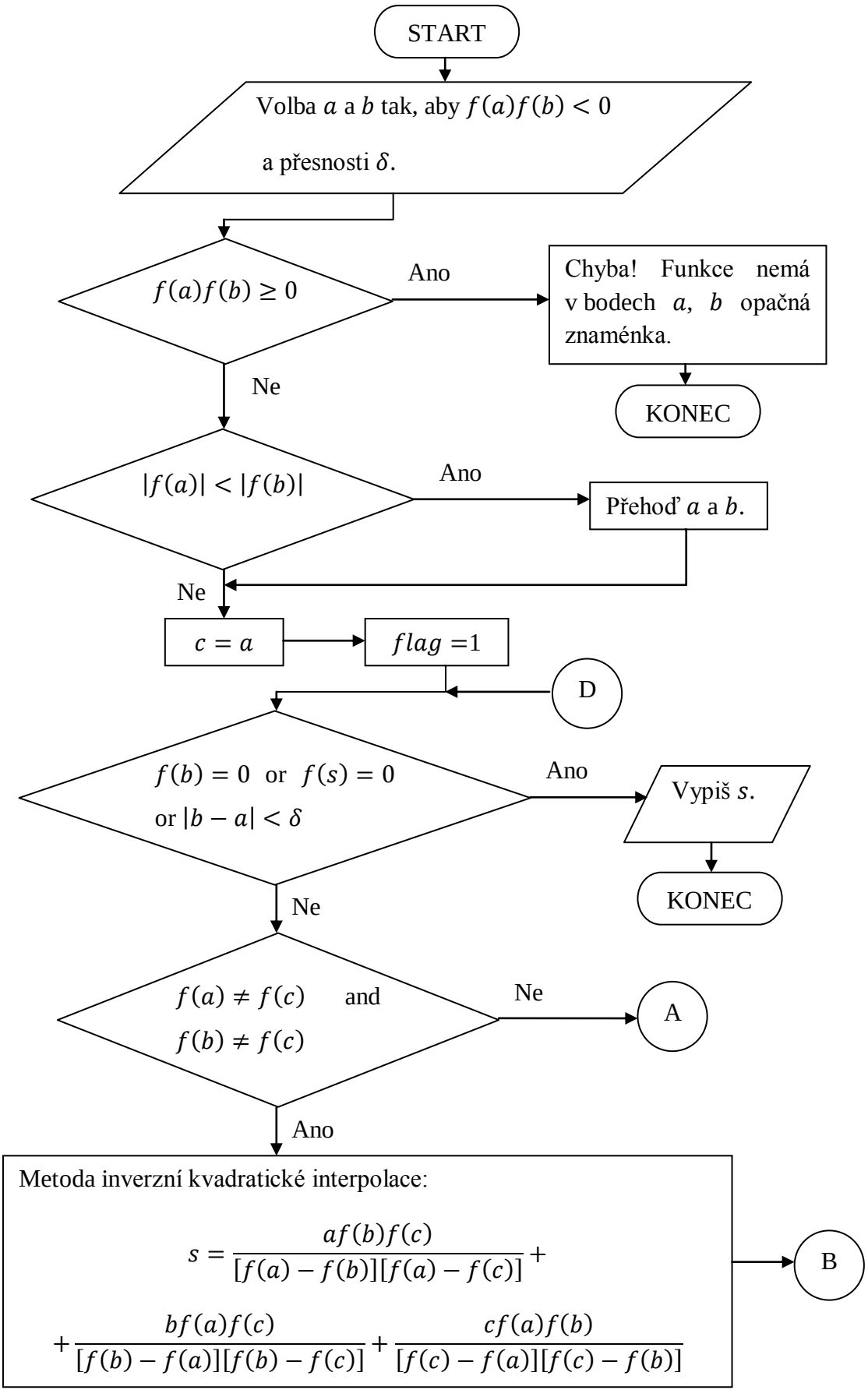
Pro naši zadanou funkci program vykreslí graf, viz obrázek (Obr. 11).



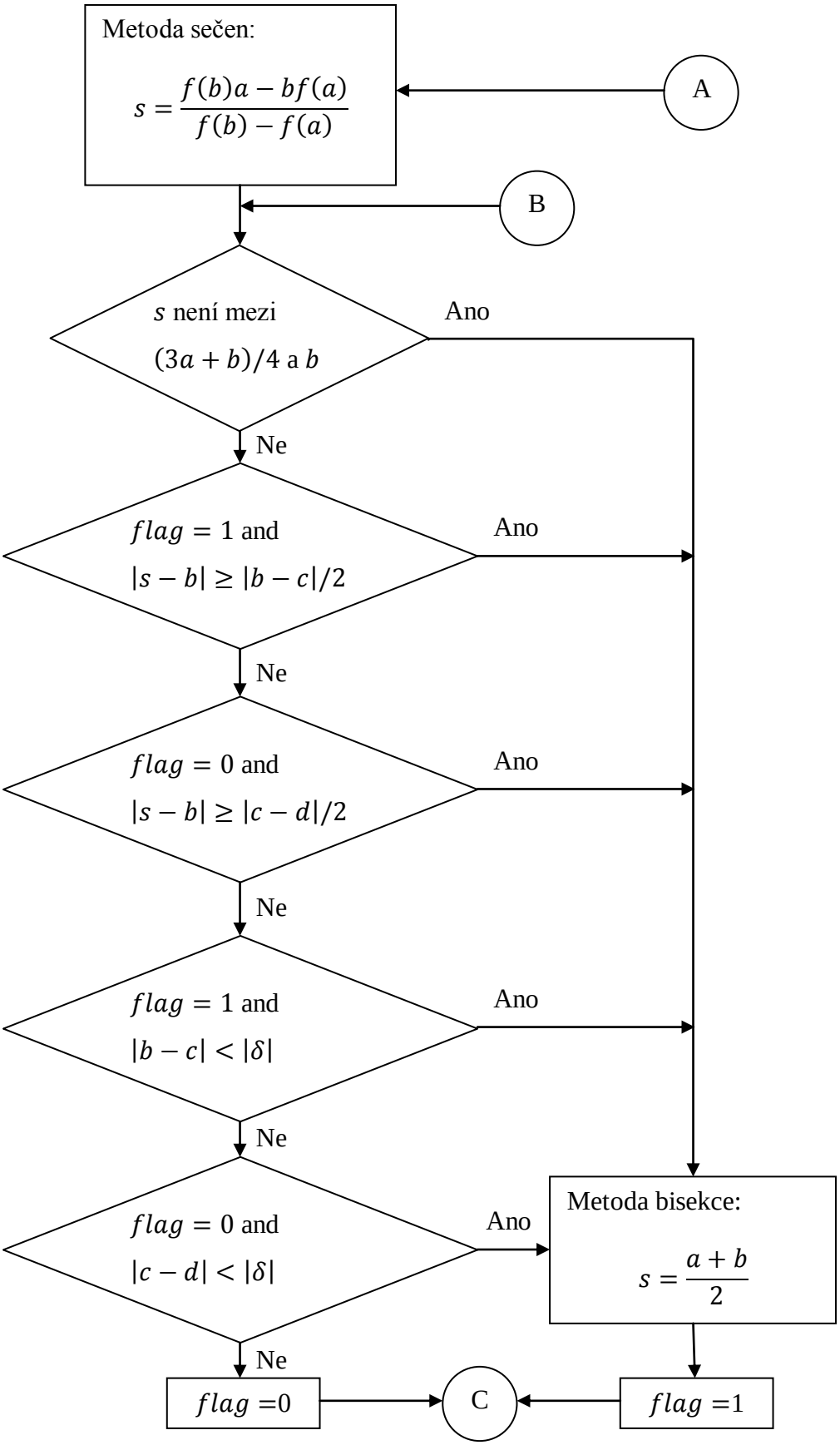
Obr. 11. Průběh vyhodnocované funkce v prostředí Mathematica.

A vypíše hodnoty:

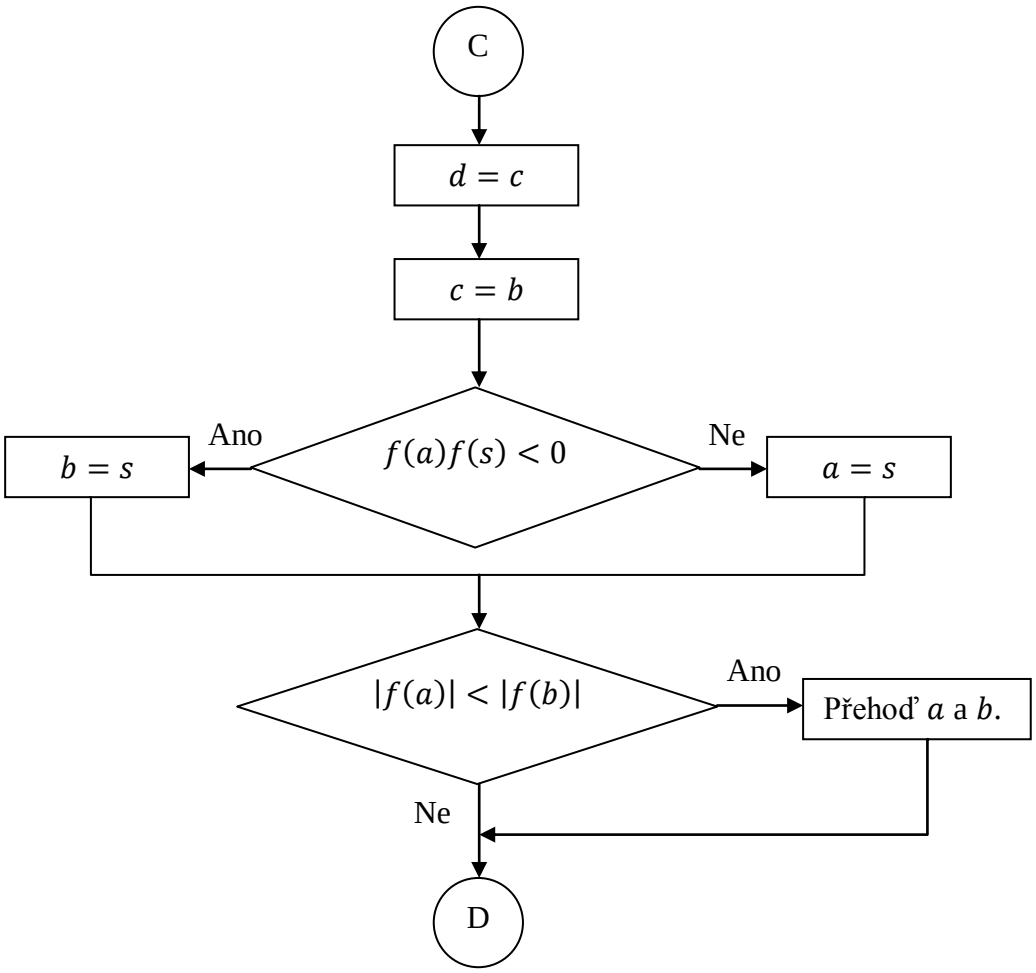
```
1 1.23256
2 1.14205
3 -1.42897
4 -2.71449
5 -3.35724
6 -3.03587
7 -2.99436
8 -2.9999
9 -3.
```



Obr. 12. Vývojový diagram Brentovy metody, část první.



Obr. 13. Vývojový diagram Brentovy metody, část druhá.



Obr. 14 Vývojový diagram Brentovy metody, část třetí.

4.3 Hledání nulových bodů funkcemi programu Mathematica

Při řešení nelineárních rovnic nemusíme postupovat podle popsaných nebo jiných metod, ale můžeme pohodlněji využít interní funkce nějakého programového systému. Mathematica poskytuje např. tyto funkce:

- **Solve**. Vyřeší rovnici či soustavu rovnic algebraickými vzorci, tzn. maximálně polynomiální rovnice čtvrtého řádu.
- **NSolve.** Vyřeší rovnici či soustavu rovnic numerickými metodami. Použít lze ovšem opět jen na polynomiální rovnice.
- **FindRoot.** Najde kořeny libovolných rovnic, je zde nutné ale zadat počáteční aproximaci.
- **DSolve.** Přesně vyřeší diferenciální rovnice.
- **NDSolve**. Řeší diferenciální rovnice numericky.

4.3.1 Funkce FindRoot

Pro nás bude tedy nejzajímavější příkaz *FindRoot*, protože rovnice nemusí být vždy algebraická, tj. polynomiální a řešením diferenciálních rovnic se tato práce nezabývá. Funkce FindRoot hledá kořeny defaultně Newtonovou metodou. Musíme ji proto zadat počáteční aproximaci kořene. Když zadáme dvě počáteční aproximace, je defaultně použita Brentova metoda. FindRoot dokáže řešit také soustavu rovnic a pokud je startovací bod zadán jako komplexní číslo, nalezneme i komplexní kořen. Metodu vyhledávání je možno změnit parametrem *Method*.

Příklad: Nalezněme funkcí FindRoot kořen rovnice $\sin x = 0$ metodou sečen z počátečních bodů $x_0 = -1$ a $x_1 = 3{,}5$. Zapíšeme:

```
1 FindRoot[Sin[x], {x, -1, 3.5}, Method -> "Secant"]
```

Funkce vrátí výsledek:

```
2 {x -> 3.14159}
```

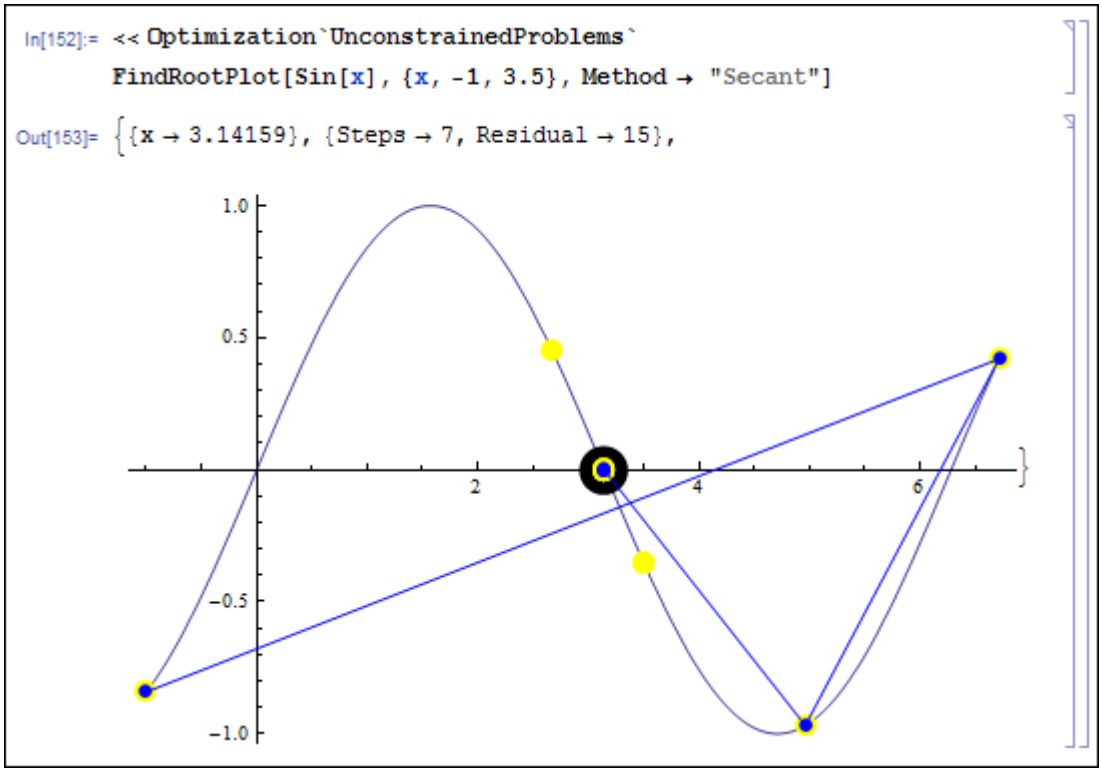
4.3.2 Funkce FindRootPlot

Pokud nás zajímá průběh iteračního procesu, můžeme si ho znázornit např. funkcí *FindRootPlot*, která je součástí souboru *Optimization`UnconstrainedProblems`*. Způsob zápisu:

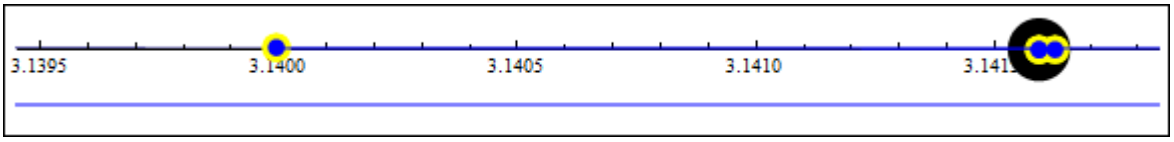
```
1 << Optimization`UnconstrainedProblems`
2 FindRootPlot[Sin[x], {x, -1, 3.5}, Method -> "Secant"]
```

Funkce FindRootPlot vrací ve vektorovém zápisu nalezený nulový bod, údaje o průběhu iteračního procesu a graf se znázorněným průběhem procesu, viz obrázek (Obr. 15). V našem případě lze vyčíst, že se nalezený kořen rovná $x^* = 3{,}14159$, iterační proces se skládal ze 7 kroků a zadaná funkce se vyhodnotila patnáctkrát. V grafu je znázorněn průběh funkce, velkým černým bodem je znázorněn nalezený kořen, modrými body jsou označeny jednotlivé aproximace kořene, ty jsou spojeny modrými úsečkami a žluté body značí vyhodnocení funkce.

Možná může někoho překvapit, že kroků má být 7, ale na obrázku (Obr. 15) se zdají být jenom 4. Je to tím, že další aproximace jsou blízko sebe u hodnoty 3 a aproximace se zpřesňuje v desetinných místech. Po přiblížení situace od sebe rozeznáme další body, viz obrázek (Obr. 16).



Obr. 15. Funkce FindRootPlot v Mathematice.



Obr. 16. Zpřesňování aproximace nulového bodu.

4.3.3 Funkce StepMonitor

Zajímají – li nás přesné hodnoty všech aproximací kořene, můžeme si je vypsat díky funkci *StepMonitor*. Zapíšeme:

```
1 FindRoot[Sin[x], {x, -1, 3.5}, Method -> "Secant",
2 StepMonitor :> Print["x = ", x]];
```

Vypíše se nám:

```
3 x = 6.71696
4 x = 4.96347
5 x = 3.14000
6 x = 3.14163
7 x = 3.14159
8 x = 3.14159
```

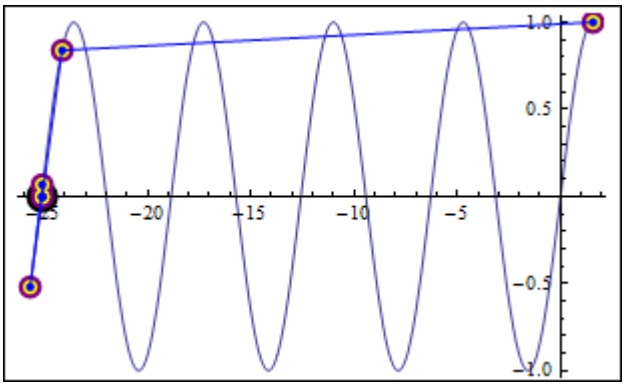
Dále můžeme zadat požadovanou přesnost, maximální počet iterací, aj. Tímto způsobem můžeme sledovat iterační procesy a srovnávat je pro různé rovnice a metody.

4.3.4 Robustnost funkce FindRoot

Jestliže funkce FindRoot používá Newtonovu metodu, tj. metodu tečen, je otázka co se stane v případě, že tečna neprotne osu x a tudíž nemůžeme pokračovat v iteracích. Zkusme to otestovat na jednoduchém příkladu:

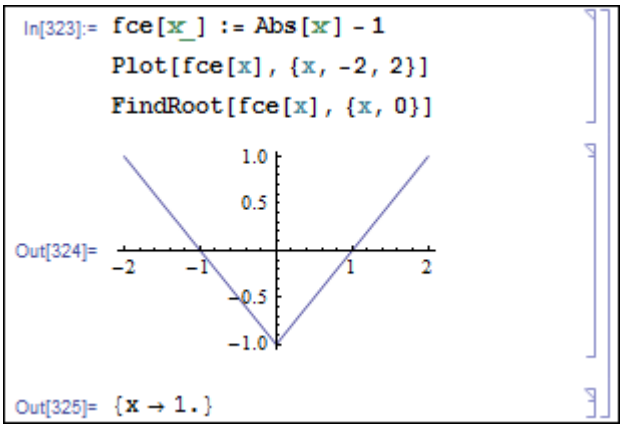
Nalezněme kořen rovnice $\sin x = 0$ z počátečního bodu $x_0 = \frac{\pi}{2}$.

Derivace funkce v počátečním bodu je rovna nule, a proto tečna sestrojená v tomto bodě bude rovnoběžná s osou x a neprotne ji. Funkce FindRoot je zřejmě ošetřena proti takovému případu a kořen najde. I když nutno přiznat, že má menší potíže a nenajde kořen nejbližší. Průběh a výsledek v prostředí Mathematica lze sledovat na obrázku (Obr. 17). Nalezený kořen: $x^* = -25{,}1327$.Vypadá to, že první krok nebyl proveden metodou tečen.



Obr. 17. Test funkce FindRoot, případ 1.

Další otázka je, co se stane, pokud tečnu v bodě sestrojit vůbec nelze. Zkusme vyřešit příklad: Naleznout kořen rovnice $|x| - 1 = 0$ s počátečním bodem $x_0 = 0$. Jak vidíme na obrázku (Obr. 18), i s tímto případem si funkce FindRoot poradí. Vypadá to, jako by byla použita jednostranná derivace zprava.



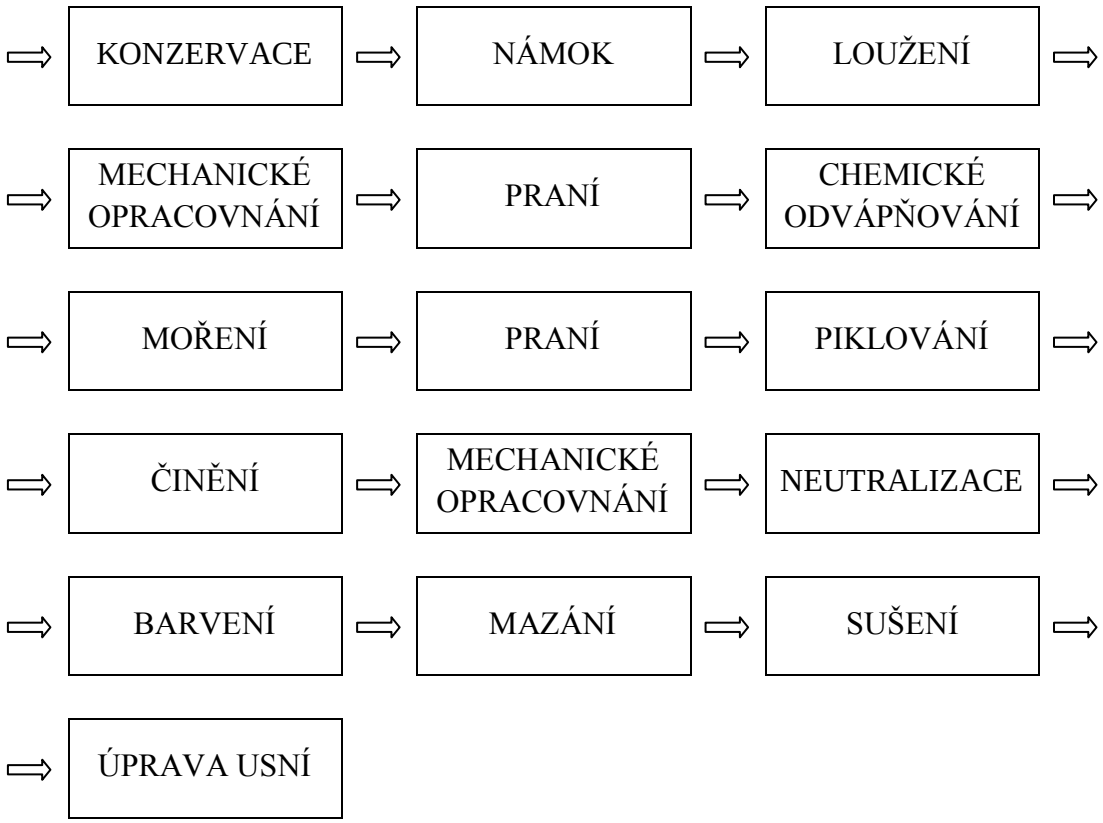
Obr. 18. Test funkce FindRoot, případ 2.

5 PŘÍKLADY APLIKACÍ VYUŽÍVAJÍCÍ NULOVÉ BODY NELINEÁRNÍCH ROVNIC

Zabývat se řešením nelineárních rovnic má svůj praktický význam. Tyto rovnice je třeba řešit v nejrůznějších inženýrských aplikacích. Uveďme několik příkladů.

5.1 Modelování pracích procesů

Zpracování kůže patří mezi nejstarší řemesla. Dnes máme díky matematice možnost procesy probíhající v kůži nebo usni modelovat a řízení výroby optimalizovat vzhledem k požadovaným parametrům. Operace při zpracování kůže jsou znázorněny na schématu (Obr. 19) [8].

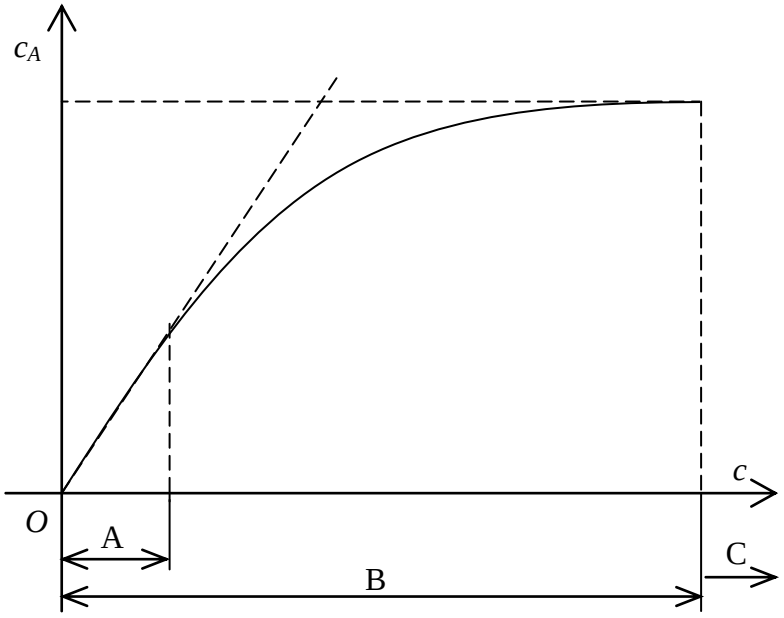


Obr. 19. Koželužské operace při zpracování kůže.

Zaměřme se na praní, které následuje po loužení a mechanickém opracování [8] [11]. Smyslem tohoto praní je odvápnit kůži nebo useň (dále jen pevná fáze). Odvápnění znamená snížení obsahu nežádoucího hydroxidu vápenatého v pevné fázi tak, aby mohli probíhat další koželužské operace. Praním pevné fáze vodou lze odstranit volně uložený hydroxid vápenatý a vyplaví se tak většina vápna z povrchu pevné fáze a z mezivláknitých prostorů. Po praní nastupuje operace chemického odvápňování, která odstraní hydroxid vápenatý chemicky vázaný na kolagenní vlákna.

5.1.1 Obecná sorpční izoterma

Při odvápňování pevné fáze vodou při praní se využívá rozdílných chemických potenciálů vápna v pevné fázi a prací lázni. Při optimalizaci pracího procesu musíme vědět, jak se vypíraná složka váže na pevnou fázi. To zjistíme stanovením sorpční izotermy, tj. závislostí rovnovážné koncentrace vypírané složky v pevné fázi c_A [kg·m^{-3}] na rovnovážné koncentraci složky v prací lázni c [kg·m^{-3}]. Rovnovážná koncentrace je koncentrace, která se nemění s časem při konstantních podmínkách experimentu, tj. vlastnosti systému jako celku (teplota, tlak, složení) se nemění. To je důvod, proč se dále nepočítá např. s teplotou. Obecná sorpční izoterma je znázorněna na obrázku (Obr. 20).



Obr. 20. Obecná sorpční izoterma.

Pro další postup je podstatné znát, ve které části sorpční izotermy se nachází stav vypírané složky. Podle obrázku (Obr. 20) to může být stav C nebo B. V oblasti stavu C je vypíraná složka volná, neváže se. V oblasti stavu B je vypíraná složka vázaná na pevnou fázi. V této oblasti můžeme vymezit zónu A, ve které je sorpční závislost prakticky lineární a konstanta úměrnosti (rovnovážná sorpční konstanta) nám charakterizuje sílu vazby vypírané složky na pevnou fázi. To nám do značné míry dovolí určit, jak je praní v této oblasti účinné.

5.1.2 Druhy praní

Prací proces se může uspořádat různými způsoby. Podle toho rozlišujeme různé druhy praní. Jsou to:

- **Lázňové praní**. Pevná fáze je ponořena do většího objemu prací kapaliny a to zpravidla jen v jednom cyklu, při němž kapalina do zařízení nepřitéká, ani z něho neodtéká.
- **Dekantační praní.** Od lázňového praní se liší tím, že se používají většinou menší objemy prací lázně, které se po dobu trvání praní několikrát vymění.
- **Průtočné praní.** Proces je uspořádán tak, že do pracího zařízení kontinuálně přitéká a odtéká prací kapalina. Pevná fáze zůstává během praní v pracím zařízení. Průtočné praní je ovlivněno množstvím kapaliny, která zůstává v pracím zařízení, tj. dynamickou zádrží.
- **Kontinuální praní.** Při tomto praní do pracího zařízení během pracího procesu kontinuálně, v různém uspořádání, přichází a odchází jak pevná fáze, tak kapalina.

Uspořádání pracího procesu má vliv na spotřebované množství vody nebo stupeň pracího procesu a délce trvání pracího procesu. Pro vyjádření spotřebované vody použijeme veličinu *námokové číslo Na* jako poměr mezi objemem lázně a objemem pevné fáze. *Stupeň pracího procesu y* se rovná podílu hmotnosti vyprané složky k počáteční hmotnosti této složky v pevné fázi.

Dále se zaměřme jen na lázňové praní při vypírání vázané složky, přičemž závislost rovnovážné koncentrace vypírané složky v pevné fázi na rovnovážné koncentraci složky v prací lázni má lineární průběh, tj. oblast A na obrázku (Obr. 20).

5.1.3 Lázňové praní

Model operace lázňového praní pevné fáze vodou lze vyjádřit difuzním modelem transportu vypíraného hydroxidu vápenatého uvnitř pevné fáze pomocí parciální diferenciální rovnice (63). Následující doplňující podmínky zajistí existenci a jednoznačnost řešení úlohy. Řešením modelu bude koncentrační pole uvnitř pevné fáze a v prací lázni. (Význam veličin je uveden v seznamu použitých symbolů a zkratek.)

$$D\frac{\partial^2 c}{\partial x^2} = \frac{\partial c}{\partial t} + \frac{\partial c_A}{\partial t} \qquad t > 0,\ \ 0 \le x \le b \tag{63}$$

$$c_A = \frac{Ac}{Bc + 1} \tag{64}$$

$$c(x, 0) = c_p \tag{65}$$

$$\frac{\partial c}{\partial x}(0,t) = 0 \tag{66}$$

$$c(b,t) = \varepsilon \cdot c_0(t) \tag{67}$$

$$\frac{\partial c}{\partial x}(b,t) = -\frac{V_0}{D \cdot S} \cdot \frac{\partial c_0}{\partial t}(t) \tag{68}$$

$$c_0(0) = 0 \tag{69}$$

Rovnice (63) je rovnicí pro difuzi vápenatých iontů z pevné fáze do vodní lázně. Koncentraci vázané složky c_A lze vyjádřit některým ze vztahů pro sorpční izotermu. V tomto případě vztah (64) vyjadřuje Langmuirovu sorpční izotermu. Počáteční podmínka (65) znamená, že na začátku praní je konstantní rozdělení koncentrace v pevné fázi. Okrajová podmínka (66) vyjadřuje, že koncentrační pole v pevné fázi je symetrické. Okrajová podmínka (67) značí dokonalé míchání lázně. Bilanční vztah (68) znamená, že rychlost sdílení hmoty prané složky na povrchu pevné fáze je rovna akumulaci této složky v lázni. Vztah (69) vyjadřuje, že pro praní použijeme čistou vodu vzhledem k vypírané složce v pevné fázi.

Při nízké koncentraci $c(x,t)$ vápenatých iontů, tj. $B \cdot c \ll 1$, má sorpční izoterma lineární tvar a vztah (64) se zjednoduší na tvar

$$c_A = K \cdot c, \tag{70}$$

kde $K \approx A$. Úloha se potom stane lineární a vztah (63) zapíšeme jako

$$\frac{\partial c}{\partial t} - \frac{D}{1+A} \cdot \frac{\partial^2 c}{\partial x^2} = 0. \tag{71}$$

Pro obecnější vyjádření a zjednodušení výpočtu jsou zavedeny bezrozměrné veličiny:

$$C = \frac{c}{c_p}, \qquad C_0 = \frac{c_0}{c_p}, \qquad F_0 = \frac{D \cdot t}{b^2(1+A)}, \qquad X = \frac{x}{b}, \qquad Na = \frac{V_0}{V} \tag{72}$$

Úlohu je možno řešit Laplaceovou transformací. Řešením je bezrozměrné koncentrační pole v pevné fázi:

$$C(X, F_0) = \frac{\varepsilon(A+1)}{\varepsilon(A+1) + Na} - \tag{73}$$

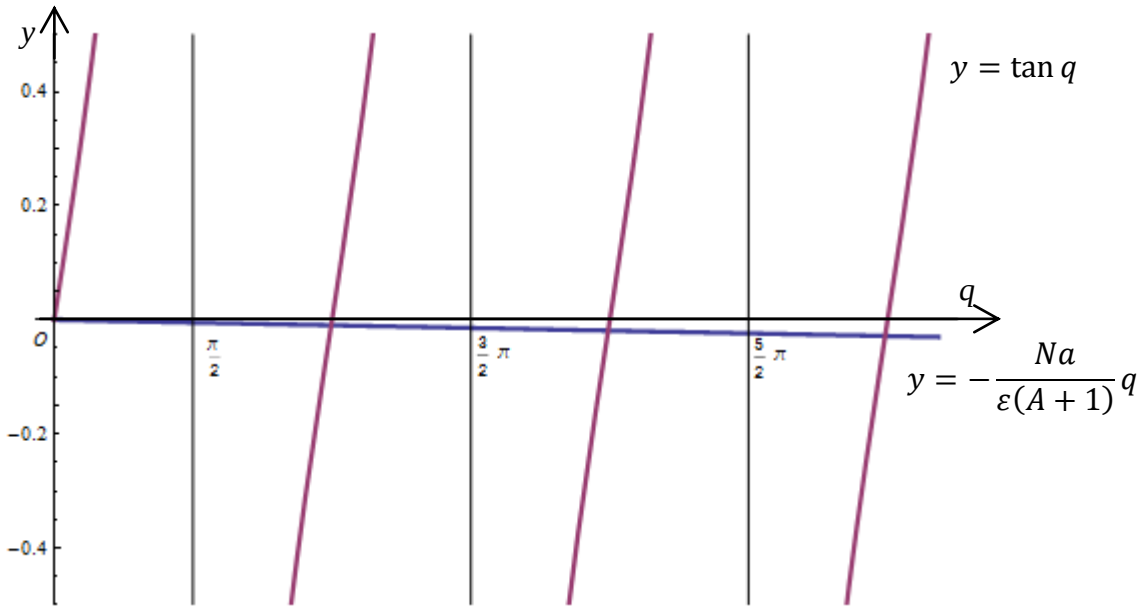
$$-2Na\sum_{n=1}^{\infty}\frac{\cos(Xq_n)\,\exp(-F_0q_n^2)}{\varepsilon(A+1)\cos q_n-\dfrac{\varepsilon(A+1)}{q_n}\sin q_n-Na\cdot q_n\sin q_n},$$

kde q_n je n-tý kladný kořen transcendentní rovnice

$$\tan q=-\frac{Na\cdot q}{\varepsilon(A+1)}. \tag{74}$$

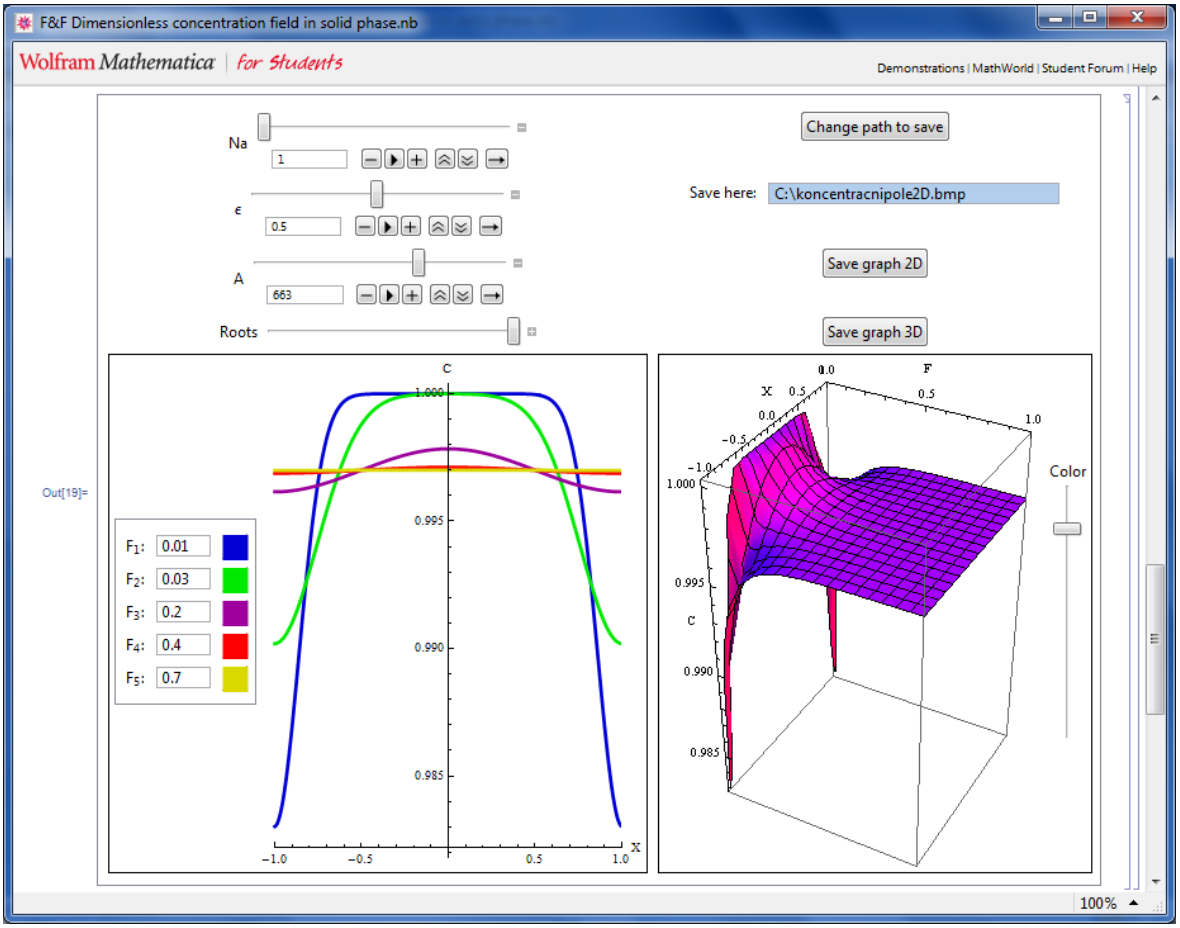
Tuto rovnici je možno řešit kteroukoli metodou pro řešení jedné nelineární rovnice popsanou v teoretické části práce. Pokud si do grafu vyneseme zvlášť levou a pravou stranu rovnice, vidíme, že kladné nulové body q_n budeme hledat v intervalech

$q_n\in\left(\frac{\pi}{2}+n\cdot\pi\,;\,\frac{\pi}{2}+(n+1)\cdot\pi\right)$ pro $n=\{0,1,2,\ldots\}$. Viz obrázek (Obr. 21).



Obr. 21. Informace o poloze kořenů grafickým znázorněním.

V systému Mathematica jsem vyčíslil kořeny rovnice (74) a dosadil je do vztahu (73). Výsledné bezrozměrné koncentrační pole vypírané složky v pevné fázi jsem znázornil ve 2D a 3D grafu. Oba grafy jsem vložil do panelu spolu s ovládacími prvky, kterými je možno dynamicky měnit parametry lázňového praní a pevné fáze. Podle zvolených parametrů se překreslují výsledné grafy. Dále je možno měnit počet vyčíslených kořenů, hodnoty časů ve 2D grafu, barvy grafů a grafy exportovat do souboru. Popsaný panel v okně Mathematica je na obrázku (Obr. 22).



Obr. 22. Vizualizace bezrozměrného koncentračního pole vypírané složky v pevné fázi v prostředí Mathematica.

5.1.4 Další typy nelineárních transcendentních rovnic při pracích procesech

Podobně se v modelech jiných druhů praní než lázňového také vyskytují rovnice, které je nutno řešit numericky. Uveďme pro stručnost jen rovnice (bez modelů) [7].

Průtočné praní s dynamickou zádrží:

$$\tan q = \frac{(A+1)Q_V b^2 - DV_0 \cdot q^2}{(A+1)DV\varepsilon \cdot q} \tag{75}$$

Průtočné praní bez dynamické zádrže:

$$\tan q = \frac{Q_V b^2}{DV\varepsilon \cdot q} \tag{76}$$

Význam veličin je uveden v seznamu použitých symbolů a zkratek. Také tyto rovnice můžeme řešit kteroukoli metodou popsanou v teoretické části práce nebo v Mathematice jednoduše funkcí FindRoot na příslušných intervalech, ve kterých leží kořeny.

5.2 Průtočný chemický reaktor s chlazením v plášti

Chemické reaktory jsou součástí mnoha technologií v chemickém i spotřebním průmyslu. Model průtočného chemického reaktoru s chlazením v plášti, ve kterém probíhá exotermická reakce podle schéma A $\overset{k_1}{\rightarrow}$ B $\overset{k_2}{\rightarrow}$ C (A, B, C symbolizují látky, → chemickou reakci, k rychlost reakce), s dokonale promíchávanou chladicí kapalinou a reakční směsí a dalšími běžnými zjednodušeními, lze popsat čtyřmi nelineárními diferenciálními rovnicemi (77) až (80).

$$\frac{dc_A}{dt} = -\left(\frac{Q_r}{V_r} + k_1\right)c_A + \frac{Q_r}{V_r}c_{Af} \tag{77}$$

$$\frac{dc_B}{dt} = -\left(\frac{Q_r}{V_r} + k_2\right)c_B + k_1 c_A + \frac{Q_r}{V_r}c_{Bf} \tag{78}$$

$$\frac{dT_r}{dt} = \frac{h_r}{\rho_r c_{pr}} + \frac{Q_r}{V_r}\left(T_{rf} - T_r\right) + \frac{A_h U}{V_r \rho_r c_{pr}}\left(T_c - T_r\right) \tag{79}$$

$$\frac{dT_c}{dt} = \frac{Q_c}{V_c}\left(T_{cf} - T_c\right) + \frac{A_h U}{V_c \rho_c c_{pc}}\left(T_r - T_c\right) \tag{80}$$

V nich t [s] znamená čas, c [mol m^{-3}] molární koncentrace, T [K] teplota, V [m^3] objem, ρ [kg m^{-3}] hustota, c_p [J kg^{-1} K^{-1}] tepelná kapacita, Q [m^3 s^{-1}] objemový průtok, A_h [m^2] plocha tepelné výměny a U [J m^{-2}s^{-1}K^{-1}] součinitel prostupu tepla. Dolní indexy vyjadřují: r reakční směs, c chladicí kapalina a f vstupní hodnoty.

Rychlost reakce a reakční teplo jsou vyjádřeny jako:

$$k_j = k_{j0}\exp\left(\frac{-E_j}{RT_r}\right), j = 1{,}2 \tag{81}$$

$$h_r = (-\Delta H_{r1})k_1 c_A + (-\Delta H_{r2})k_2 c_B, \tag{82}$$

kde jsou k_0 [s^{-1}] pre-exponenciální faktory, E [J] aktivační energie a $-\Delta H_r$ [J mol^{-1}] záporně brané reakční entalpie.

Uvažujme soustavu v ustáleném stavu. Potom jsou v rovnicích (77) až (80) derivace na levých stranách rovny nule a spolu s (81) a (82) dostáváme soustavu nelineárních transcendentních rovnic, kterou můžeme vyřešit numericky. Nejjednodušší je v tomto případě použít metodu prosté iterace. Pevný bod budeme hledat k teplotě T_r. Z (81) vyjádříme rychlosti k_1 a k_2 s počáteční aproximací T_r. Ty dosadíme do rovnic (77) a (78),

tím dostaneme koncentrace c_A a c_B. Po jejich dosazení do rovnice (80) získáme teplotu T_c a tu dosadíme do rovnice (79) a dostaneme další aproximaci teploty T_r. Tento cyklus stále opakujeme, dokud nedosáhneme požadované přesnosti. Soubor s realizovaným algoritmem v prostředí Mathematica je přiložen na CD.

5.3 Optimalizace výkonu fotovoltaického článku

Výkonová charakteristika fotovoltaického článku je jednou ze základních charakteristik, která slouží pro optimální nastavení střídače fotovoltaického systému, který přeměňuje elektrickou energii vyrobenou ve fotočláncích (DC) na elektrickou energii, která je poté následně dodávána do místa spotřeby (většinou AC) [15]. Snahou je, aby fotovoltaický systém vždy dodával do zátěže nejvyšší výkon. Nejvyšší výkon při daných podmínkách (teplota, intenzita osvětlení) odpovídá tzv. bodu *MPP* (Maximum Power Point) na výkonové charakteristice fotočlánku. Bod *MPP* je určen dvojicí napěťové a proudové souřadnice:

$$MPP = [U_m, I_m]; \;\; P_{max} = U_m \cdot I_m. \tag{83}$$

Při hledání polohy napěťové souřadnice *MPP* vyjdeme z napěťového modelu fotovoltaického článku:

$$I = I_L - I_0 \cdot \exp\left(\frac{U}{U_t}\right). \tag{84}$$

Při hledání polohy proudové souřadnice *MPP* vyjdeme z proudového modelu fotovoltaického článku:

$$U = U_t \cdot \ln\frac{I_L - I}{I_0}. \tag{85}$$

Veličiny v těchto modelech (84) a (85) mají význam:

- $I\ [A]$ proud protékající fotočlánkem.
- $I_L\ [A]$ fotoproud úměrný zářivému toku (intenzitě osvětlení).
- $I_0\ [A]$ proud v závěrném směru.
- $U\ [V]$ napětí na fotočlánku.
- $U_t\ [V]$ teplotní napětí.

Z rovnic (84) a (85) se dají odvodit vztahy pro U_m a I_m (86) a (87).

$$I_L = I_0 \cdot \exp\left(\frac{U_m}{U_t}\right) \cdot \left(1 + \frac{U_m}{U_t}\right) \tag{86}$$

$$\ln \frac{I_L - I_m}{I_0} = \frac{I_m}{I_L - I_m} \tag{87}$$

Z těchto dvou vztahů musíme vyčíslit U_m a I_m pro zjištění polohy *MPP*. Hledané veličiny ale nemůžeme explicitně vyjádřit, a proto musíme rovnice řešit numericky. Nejedná se o soustavu rovnic a každou rovnici řešíme zvlášť např. použitím některé metody popsané v teoretické části práce. Soubor s hledáním kořene Newtonovou metodou v Mathematice je přiložen na CD.

5.4 Další výskyty nelineárních rovnic

Další nelineární rovnice, které se musí řešit numericky, se vyskytují v nejrůznějších inženýrských aplikacích [12]. Jmenujme např.:

- Výpočet složité chemické rovnováhy.
- Řešení protiproudých separačních zařízení, jako jsou např. rektifikační či absorpční kolony.
- Výpočet stacionární simulace systému zařízení.
- Náhrada parabolických nebo eliptických rovnic metodou konečných diferencí.
- Výpočet stacionárních stavů dynamických modelů popsaných obyčejnými diferenciálními rovnicemi.
- Výpočty v modelech z různých odvětví, např. model pH neutralizace, model výparníku, model poškození betonu, atd.

ZÁVĚR

V teoretické části práce jsem popsal nejpoužívanější numerické metody pro řešení převážně jedné nelineární transcendentní rovnice. Uvedl jsem také zobecnění některých metod pro řešení soustavy rovnic. Vhodnost použití konkrétní metody je třeba zvažovat vzhledem ke konkrétním požadavkům a situaci. Metody, které vždy konvergují, a máme při jejich použití jistotu nalezení kořene, mají obecně nižší rychlost konvergence. Naopak metody s vyšší rychlostí konvergence nemusí konvergovat vždy a vyžadují přesnější počáteční aproximaci kořene. Možností je kombinovat nějakou vždy konvergující metodu s metodou rychlejší, jak to dělá např. Brentova metoda. Popsané metody jsem realizoval v systému Mathematica. Tyto soubory jsou přiloženy na CD. V praktické části práce jsem na ukázku uvedl implementaci Brentovy metody. Mathematica je vybavena, stejně jako jiné podobné systémy počítačové algebry, funkcemi pro řešení těchto rovnic. V praktické části práce je popsána nejvíce funkce FindRoot, jejíž základ tvoří Newtonova či Brentova metoda. Při jejím použití uživatel nemusí znát postup metody, jakým funkce hledá kořen, ale musí zadat počáteční aproximaci nulového bodu. Nejjednodušší a nejnázornější způsob jak zjistit přibližnou polohu kořene je znázornit si rovnici jako průběh funkce v kartézské soustavě souřadnic. V případě soustavy rovnic to však není vždy dovoleno a ani neexistuje žádná metoda, která by zaručeně vždy konvergovala ke kořeni. Proto je získání informace o poloze kořene ve vícedimenzionálním případě složitější. Jako aplikace využití nulových bodů nelineárních rovnic jsem uvedl řešení rovnice při výpočtu koncentračního pole vypírané složky v pevné fázi při lázňovém praní. Dále nelineární rovnice při jiných typech praní, soustavu rovnic, kterou je nutno řešit při výpočtu průtočného chemického reaktoru s chlazením v plášti. A rovnice, které se řeší při optimalizaci výkonu fotovoltaického článku.

ZÁVĚR V ANGLIČTINĚ

In the theoretical part, I described the most widely used numerical methods for solving nonlinear transcendental predominantly single equation. I also introduced a generalization of some methods for solving systems of equations. Suitability of a particular method should be considered in regard to the specific needs and situation. Methods, which always converge, and we have confidence in their use to find the root, generally have lower rates of convergence. By contrast methods with high convergence speed does not always converge and require more accurate initial approximation of roots. Option is to combine an always convergent method with a faster method, as it does the Brent's method. I realized described methods in the Mathematica system. These files are included on the CD. In the practical part I for the show said implementation of Brent's methods. Mathematica is equipped, as well as other similar systems computer algebra, functions for solving these equations. The practical part describes the most FindRoot function, which basis form Newton's method and Brent's. In its application the user need not know the method, which seeks the roots of functions, but must enter an initial approximation of the zero point. The simplest and the most illustrative way to determine the approximate location of the root is represent the equation as a function in Cartesian coordinates. In the case of system of equations that is not always allowed, and nor is there any method that would surely always converge to the root. Therefore to obtain information on the location of the root in multi-dimensional system is complex case. As an application uses the zeros of nonlinear equations I mentioned solutions of the equation for calculating the concentration field of the washing element of the solid phase in bath washing. Furthermore, nonlinear equations for other types of washing, the system of equations, which must be calculated in flow chemical reactor with cooling in the mantle and equations to calculate the optimization of performance of photovoltaic cell.

SEZNAM POUŽITÉ LITERATURY

[1] Banachova věta o pevném bodě In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6. 7. 2008, 1. 3. 2010 [cit. 2010-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Banachova_v%C4%9Bta_o_pevn%C3%A9m_bod%C4%9B>.

[2] Brent's method. In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 8 December 2004, last modified on 21 May 2010 [cit. 2010-05-23].

Dostupné z WWW: <http://en.wikipedia.org/wiki/Brent%27s_method>.

[3] ČERMÁK, Libor; HLAVIČKA, Rudolf Numerické metody: Řešení nelineárních rovnic. 2006 [cit. 2010-05-10]. Dostupné z WWW:

<http://mathonline.fme.vutbr.cz/UploadedFiles/244.pdf>.

[4] *ČVUT Praha - Fakulta jaderná a fyzikálně inženýrská: Katedra fyzikální elektroniky* [online]. [cit. 2010-05-21]. Řešení nelineárních rovnic. Dostupné z WWW: <http://kfe.fjfi.cvut.cz/~klimo/nm/l8.pdf>.

[5] DÁVID, Arnold. *Numerické metódy na osobnom počítači*. 1. vydanie. Bratislava: Alfa, 1988. 177 s.

[6] *ELKAN, spol. s r.o.: Mathematica* [online]. 2010 [cit. 2010-05-25]. Produkty. Dostupné z WWW: <http://www.mathematica.cz/produkty-all.php>.

[7] FIALKA, M.; CHARVÁTOVÁ, H. Modelling of extraction process in tannery by computer algebraic system. In *Topical Problems of Fluid Mechanics* 2007. Institute of Thermomechanics, Academy of Sciences of the Czech Republic, Prague, February 28 – March 2nd, 2007, 37 – 40.

[8] CHARVÁTOVÁ, H. *Modelování chemického odvápňování holiny.* Dizertační práce. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta technologická, 2007.

[9] JANÁČOVÁ, D. *Modelování extrakčních procesů*. Habilitační práce, VŠB – Technická univerzita Ostrava, Ostrava, 2002.

[10] KOLMOGOROV, A. N.; FORMIN, S. V. *Základy teorie funkcí a funkcionální analýzy*. Praha: SNTL, 1975.

[11] KOLOMAZNÍK, K. *Modelování zpracovatelských procesů.* Skriptum, Brno: VUT v Brně, určeno pro FT ve Zlíně, 1978.

[12] KUBÍČEK, Milan. *Numerické algoritmy řešení chemickoinženýrských úloh*. Praha: SNTL/ALFA, 1983. 356 s.

[13] VITÁSEK, Emil. *Numerické metody*. Praha: SNTL, 1987. 516 s.

[14] *VUT Fakulta strojního inženýrství, ústav matematiky* [online]. [cit. 2010-05-22]. Základy funkcionální analýzy.

Dostupné z WWW: <www.math.fme.vutbr.cz/download.aspx?id_file=1525>.

[15] Vyšší odborná škola a střední průmyslová škola Varnsdorf [online]. [cit. 2010-05-31]. Výkonová charakteristika FVČ. Dostupné z WWW: <http://www.vosvdf.cz/cmsb/index.php?p=1386>.

[16] *WOLRAM RESEARCH* [online]. 2010 [cit. 2010-05-25]. Dostupné z WWW: <http://www.wolfram.com>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

V [m^3]	Objem pevné fáze, tj. vzorku holiny.
V_0 [m^3]	Objem prací vody jakožto odvápňovacího činidla.
t [s]	Čas.
c [$kg \cdot m^{-3}$]	Objemová koncentrace $Ca(OH)_2$ v holině.
c_0 [$kg \cdot m^{-3}$]	Objemová koncentrace $Ca(OH)_2$ v lázni.
c_p [$kg \cdot m^{-3}$]	Počáteční koncentrace $Ca(OH)_2$ v holině.
D [$m^2 \cdot s^{-1}$]	Efektivní difuzní koeficient vymývané složky z pevné fáze.
x [m]	Souřadnice polohy.
b [m]	Poloviční tloušťka holiny.
ε [1]	Porozita (je poměr objemu pórů k objemu vzorku holiny).
N_a [1]	Námokové číslo (je podíl V_0 / V).
q_n [1]	n-tý kořen jisté transcendentní rovnice.
A [1]	Rovnovážná konstanta sorpce (z Langmuirovy izotermy).
K [1]	Přibližná hodnota rovnovážné sorpční konstanty A.
B [$m^3 \cdot kg^{-1}$]	Konstanta z Langmuirovy izotermy.
F_o [1]	Fourierovo kritérium (bezrozměrný čas).
C [1]	Bezrozměrná objemová koncentrace $Ca(OH)_2$ v holině.
C_0 [1]	Bezrozměrná objemová koncentrace $Ca(OH)_2$ v lázni.
X [1]	Bezrozměrná prostorová souřadnice.
Q_V [$m^3 \cdot s^{-1}$]	Objemový průtok prací kapaliny.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK