Kořeny nelineárních funkcí

Matematické algoritmy (K611MAG)

Jan Přikryl

9. přednáška 11MAG
pondělí 1. prosince 2014

verze:2014-12-01 10:19

Obsah

1 Nelineární rovnice **1**

1.1 Formulace úlohy 1

1.2 Existence a jednoznačnost 2

1.3 Podmíněnost řešení 4

1.4 Iterační metody a jejich konvergence 5

2 Iterační metody **7**

2.1 Metoda půlení intervalu neboli bisekce 7

2.2 Metoda postupných aproximací 9

2.3 Newtonova metoda 12

2.4 Metoda sečen 14

3 Dodatky **16**

3.1 Bezpečné metody 16

3.2 Numerický výpočet kořenů polynomu 17

3.3 Numerické řešení soustav nelineárních rovnic 17

1 Řešení nelineárních rovnic

1.1 Formulace úlohy

Pro detailnější obeznámení s pojmy, uváděnými níže, doporučuji i zde konzultovat knihu Michaela T. Heathe [1], případně nějakou z českých učebnic či mnoha skript o numerické matematice,

která v posledních letech vyšla – například [2], [3] (části tohoto skripta jsou dostupné i on-line). Mnohé ze zde použitých obrázků jsme převzali právě z [1].

Budeme se zde zabývat především numerickými metodami pro (přibližné) řešení **nelineární rovnice**

$$f(x) = 0, \tag{1}$$

kde $f : \mathbb{R} \to \mathbb{R}$ je reálná nelineární funkce jedné reálné proměnné. Řešit **lineární rovnice** tvaru $ax + b = c$, tj. $ax + b - c = 0$, jsme se naučili již na střední škole. Seznámili jsme se tam i s řešením některých nelineárních rovnic, například kvadratické rovnice $ax^2 + bx + c = 0$ nebo rovnice $\sin x = 0$. K řešení většiny nelineárních rovnic však potřebujeme použít některou vhodnou numerickou metodu.

Řešením nebo **kořenem** uvedené rovnice nebo také **nulovým bodem** funkce f nazýváme takové reálné číslo x^*, pro které platí $f(x^*) = 0$. Nelineární rovnice mohou mít právě jedno řešení (rovnice $x - \sin x = 0$), více řešení (rovnice $x^2 - 1 = 0$), nebo nemusí mít žádné řešení (rovnice $\sin x = 2$).

Budeme se také zabývat speciálním rovnicemi tvaru

$$x = g(x), \tag{2}$$

kde opět $g : \mathbb{R} \to \mathbb{R}$ a tedy $f(x) = x - g(x)$. Řešení takové rovnice se nazývá také **pevným bodem** funkce g.

Speciální situace nastává také, je-li uvažovaná funkce f polynom. Hledáme pak totiž často i jeho komplexní kořeny. Pro polynomy existují tedy i speciální numerické metody, jimiž se zde však nemůžeme extra zabývat. V praxi se setkáme i se soustavami nelineárních rovnic, jejichž řešením pak není číslo, ale vektor hodnot. Existují numerické metody i pro řešení takových soustav, z časových důvodů se jimi ale zde také zabývat nebudeme. Případné zájemce odkazujeme na Heathovu knihu [1] nebo na Míkovu učebnici [2] či skripta [3].

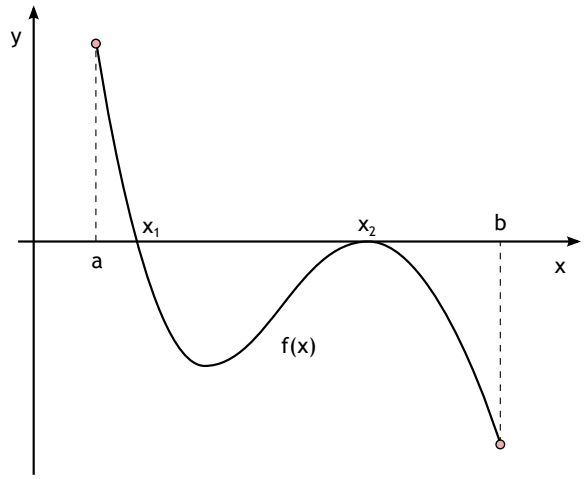
1.2 Existence a jednoznačnost

Existence a jednoznačnost řešení nelineárních rovnic je podstatně komplikovanější záležitost než je tomu u lineárních rovnic a jejich soustav. V mnoha případech je obtížné stanovit existenci nebo počet řešení nelineární rovnice. Zatímco u soustav lineárních rovnic musí být počet řešení roven nule, jedné nebo být nekonečný, nelineární rovnice mohou mít jakýkoli počet řešení, a to dokonce i pro jedinou rovnici.

Příklad 1 (Existence kořenů). Uvažujeme-li kořeny následujících rovnic na celém $\mathbb{R}$, pak rovnice

- $e^x + 1 = 0$ nemá žádné řešení
- $e^{-x} - x = 0$ má právě jedno řešení
- $x^2 - 4\sin x = 0$ má dvě řešení
- $x^3 + 6x^2 + 11x - 6 = 0$ má tři řešení
- $\sin x = 0$ má nekonečně mnoho řešení

Jakkoli je tedy obtížné získat jakákoli globální tvrzení o počtu řešení nelineární rovnice, máme přesto k dispozici některá užitečná lokální kritéria zaručující existenci aspoň jednoho kořene rovnice na daném intervalu. Jedno takové praktické kritérium je založeno na matematické větě, která pochází od B. Bolzana a říká:



Obrázek 1: Příklad nelineární funkce s kořeny x_1 a x_2 na intervalu $[a, b]$. Tento interval je pro danou funkci $f(x)$ uzávěrou kořene.

Věta 2 (Bolzanova věta)**.** *Nechť funkce f je spojitá na uzavřeném omezeném intervalu $[a, b]$ a nechť platí $f(a) \cdot f(b) < 0$ (tj. $f(a)$ a $f(b)$ mají opačná znaménka). Pak funkce f má na intervalu (a, b) alespoň jeden nulový bod.*

Takový interval $[a, b]$, v jehož koncových bodech má funkce f opačná znaménka, budeme nazývat **uzávěrou** řešení nelineární rovnice $f(x) = 0$. Jak uvidíme později, v řadě numerických metod pro řešení nelineárních rovnic hraje důležitou roli právě postupné zužování takové předem nalezené uzávěry. Jak ovšem počáteční uzávěru najít, je víceméně záležitostí pokusů a omylů. Jedna z možností je odhadnout nějak počáteční interval, na němž budeme chování funkce f zkoumat (i když to ještě nebude uzávěra kořene), a pak procházet tímto intervalem po nějakých vhodně volených krocích, postupně počítat hodnoty $f(x)$ a sledovat, kdy v nich dojde ke změně znaménka.

Poznamenejme ještě, že právě zmíněné kritérium udává pouze postačující, nikoli nutnou podmínku existence kořene. Nemělo by nás tedy od hledání kořene předem odrazovat to, že jsme nenašli jeho uzávěru. V řadě případů totiž ani pro daný kořen uzávěra neexistuje. Jako příklad stačí vzít triviální rovnici $x^2 = 0$ s jediným kořenem $x = 0$. Zde pro všechna x máme $x^2 \geq 0$ a ke změně znaménka tedy nemůže dojít.

Obraťme svoji pozornost nyní k rovnici (2). Zde lze k důkazu existence pevného bodu na daném intervalu využít opět klasické matematiky, konkrétně tzv. věty o kontrakci.

Definice 3 (Kontrakce)**.** Řekneme, že funkce $g : \mathbb{R} \to \mathbb{R}$ je na množině $S \subseteq \mathbb{R}$ **kontrakce**, pokud existuje konstanta $L \in \mathbb{R}$, $0 < L < 1$ taková, že pro všechna $x, y \in S$ platí

$$|g(x) - g(y)| \leq L|x - y|.$$

Věta 4 (O existenci pevného bodu)**.** *Jestliže je funkce g kontrakce na uzavřené množině $S \subseteq \mathbb{R}$ a $g(S) \subseteq S$, pak má g v S pevný bod, a to právě jeden.*

Pokud čtenáři vadí, že jsme výše uvedenou definici a větu formulovali pro obecnou množinu S, může si pod S představovat nějaký interval. Z uvedené věty ihned plyne, že pokud v rovnici (1) můžeme psát $f(x) = x - g(x)$, kde g je kontrakce na nějaké uzavřené množině S taková, že zobrazuje S do sebe samé, má rovnice $f(x) = 0$ na množině S právě jedno řešení, totiž pevný

bod funkce g. Brzy uvidíme, že tato skutečnost nám dává možnost odvodit některé numerické metody pro řešení nelineárních rovnic. Poznamenejme ještě, že pokud pro všechna $x \in S$ existuje derivace funkce g a platí $|g'(x)| < 1$, dá se ukázat, že g na S je kontrakce.

Z matematického či logického hlediska jsou naše úvahy o uzávěře spojité funkce či předpoklady věty o kontrakci pouze *postačující podmínky*, nikoli však podmínky nutné. Není tedy nikde psáno, že funkce f nemůže mít nulový bod v intervalu, který není uzávěrou, nebo že funkce g, která není kontrakce, nemůže mít pevný bod. V praxi tedy může být užitečné využít soudobých možností našeho softwarového vybavení a při řešení nelineárních rovnic si nejprve nechat vykreslit graf funkce f pro rovnici (1) nebo grafy $y = x$ a $y = g(x)$ pro rovnici (2).

Příklad 5 (Nesplněné postačující podmínky). Nejsou-li postačující podmínky pro existenci nulového nebo pevného bodu, neznamená to ještě, že takový bod neexistuje:

- funkce $f(x) = x^2$ má nulový bod $x = 0$ na intervalu $[-1, 1]$, který není uzávěra
- funkce $g(x) = \sin x$ má pevný bod $x = 0$, ale v okolí tohoto bodu to není kontrakce

Doposud jsme se soustředili převážně na existenci kořenů nelineárních rovnic a ne na jejich jednoznačnost, protože se obecně má za to, že nelineární rovnice mohou mít více než jedno řešení, přinejmenším globálně. Jednoznačnost kořene nás přesto může zajímat, alespoň lokálně, například na daném intervalu. Připomeňme si, že z lineární algebry víme, že soustava lineárních rovnic s regulární maticí má vždy právě jedno řešení. Pro nelineární funkce f platí podobné tvrzení o regularitě, přinejmenším lokálně. Pokud totiž funkce f má v daném bodě x^* nenulovou derivaci, pak existuje otevřený interval kolem tohoto bodu, v němž je funkce f ostře monotónní, tedy rostoucí nebo klesající. V takové situaci v okolí bodu x^* tedy může existovat nejvýše jeden kořen.

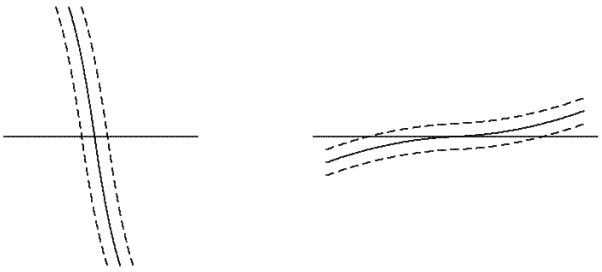
Pokud však v nějakém kořenu x^* má funkce f nulovou derivaci, má tento kořen jisté zvláštní vlastnosti, které ovlivňují jak podmíněnost řešené úlohy, tak také chování použité numerické metody. Nulový bod x^* funkce f, pro který platí zároveň $f(x^*) = 0$ a $f'(x^*) = 0$ se nazývá *násobný kořen* rovnice $f(x) = 0$. Geometricky to znamená, že graf funkce f má v tomto bodě vodorovnou tečnu splývající s osou x. Kořeny, které nejsou násobné, se nazývají *jednoduché*. Kořen x_1 z Obrázku1 je tedy jednoduchý, kořen x_2 je násobný. Pojem násobnosti lze pro hladké funkce f dále upřesnit. Pokud platí $f(x^*) = f'(x^*) = f''(x^*) = \cdots = f^{(m-1)}(x^*) = 0$, ale $f^{(m)} \neq 0$, řekneme, že násobnost kořene x^* je m.

Příklad 6 (Pevné body). Ověřte si následující tvrzení:

- funkce $g(x) = \sin x$ má jediný pevný bod $x = 0$
- funkce $g(x) = x^2$ má pevné body $x = 0$ a $x = 1$
- kořen $x = 0$ rovnice $\sin x = 0$ je jednoduchý
- kořen $x = 0$ rovnice $x^2 = 0$ je dvojnásobný
- kořen $x = 1$ rovnice $x^3 - 3x^2 + 3x - 1 = 0$ je trojnásobný

1.3 Podmíněnost řešení

Abychom mohli kvantitativně měřit citlivost řešení nelineárních rovnic na data (funkční hodnoty), musíme pracovat s absolutním číslem podmíněnosti, což je obdoba již zavedeného čísla



Obrázek 2: Podmíněnost kořenů nelineární rovnice $f(x) = 0$. Vlevo: dobře podmíněná úloha, vpravo: špatně podmíněná úloha.

podmíněnosti z minulé přednášky, kde ale místo relativních změn v čitateli i jmenovateli vystupují absolutní odchylky. Je to dáno tím, že hodnota funkce f v kořenu rovnice je rovna nule. Dá se ukázat, že pokud má funkce f v okolí kořene x^* derivaci, je pak toto číslo podmíněnosti přibližně

$$C_{\mathrm{p,abs}} \approx \frac{1}{|f'(x^*)|}.$$

Pokud je ve jmenovateli $f'(x^*) = 0$ (násobný kořen), klademe $C_{\mathrm{p,abs}} = \infty$. Z definice čísla podmíněnosti pak vyplývá, že pokud najdeme bod $\tilde{x}$ takový, že $|f(\tilde{x})| < \epsilon$, může odchylka $|\tilde{x} - x^*|$ tohoto bodu od kořene rovnice $f(x) = 0$ mít velikost $\epsilon/|f'(x^*)|$. Pro malé hodnoty $|f'(x^*)|$ tedy může být tato odchylka od kořene velká, i když funkční hodnota sama je malá.

Celou situaci ilustruje Obrázek 2. Čárkované křivky vyznačují oblast nejistoty kolem každé plně nakreslené křivky, takže nulový bod dané funkce může být kdekoli mezi body, v nichž čárkované křivky protínají vodorovnou osu. Malý interval nejistoty pro nulový bod na levém obrázku je dán tím, že daná křivka strmě roste (takže převrácená hodnota derivace je malá), kdežto velký interval nejistoty pro nulový bod na pravém obrázku plyne z pomalého růstu (a tedy velké převrácené hodnoty derivace). Všimněte si také toho, že šíře pásu nejistot kolem funkčních hodnot je na obou obrázcích stejná.

Máme-li násobný kořen x^*, je $f'(x^*) = 0$, takže číslo podmíněnosti násobného kořene je nekonečné. To dává smysl, protože nepatrná změna v f může způsobit, že z násobného kořene se stane více než jeden kořen nebo naopak násobný kořen zmizí. Stačí si k tomu nakreslit například funkci $f(x) = x^2$ a posunout ji o malé ϵ nahoru nebo dolů. ■ **příklad,obrázek** ■

Podmíněnost nelineární rovnice ovlivňuje náš pohled na přibližné řešení $\tilde{x}$: máme usilovat o to, aby $|f(\tilde{x})|$ byla malá, nebo spíše o to, aby bylo malé $|\tilde{x} - x^*|$, jakkoli přesné řešení x^* předem neznáme? Jak už to u numerických metod bývá, obě uvedené veličiny nejsou nutně malé současně, závisí to ještě na podmíněnosti. Tato skutečnost ovlivňuje volbu algoritmů numerických metod, o nichž budeme hovořit ve zbytku této přednášky. V každém případě je užitečné získat předem nějakou informaci o podmíněnosti řešené úlohy.

1.4 Iterační metody a jejich konvergence

Numerické metody pro řešení nelineárních rovnic jsou vesměs metody iterační. Iterace (z lat. *iterare*, opakovat) znamená postupné opakování určitého postupu, během kterého se postupně generuje posloupnost hodnot $x_0, x_1, \ldots, x_k, \ldots$ taková, že v našem případě (hledáme kořen x^* nelineární rovnice) postupně získávané hodnoty konvergují k hledanému řešení, $x_k \to x^*$ pro

$k \to \infty$. Při skutečném výpočtu samozřejmě nemůžeme jít s k do nekonečna a iterační postup zastavíme po určitém dostatečně velkém počtu kroků pomocí vhodně zvoleného zastavovacího kritéria. Získáme tak přibližnou hodnotu hledaného kořene. Termín **iterace** se v numerické matematice používá nejen k označení výše uvedeného postupu jako celku, ale nazývá se tak také každý jeho krok a nazývají se tak také postupně počítané hodnoty x_k, tedy aproximace hledaného kořene.

Abychom mohli porovnávat efektivitu iteračních metod, potřebujeme nějak charakterizovat jejich rychlost konvergence, tj. rychlost konvergence posloupnosti iterací x_k k hledanému kořenu rovnice. *Chyba*(nepřesnost) k-té iterace, kterou budeme označovat e_k, se obvykle definuje jako $e_k = x_k - x^*$, kde x_k je aproximace (přiblížení) hledaného řešení získaná v iteraci k a x^* je skutečné (přesné) řešení. Některé z používaných metod neprodukují přímo konkrétní přibližné řešení x_k, ale pouze interval, který s určitostí obsahuje přesné řešení, přičemž délka tohoto intervalu se během iteračního procesu postupně zmenšuje. U takové metody pak e_k definujeme jako délku tohoto intervalu po k-té iteraci. V obou případech pak řekneme, že daná iterační **metoda konverguje s rychlostí** r (také: metoda je řádu r), jestliže pro nějakou konečnou kladnou konstantu $C > 0$ platí

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^r} = C.$$

Speciálně se rozlišují následující případy:

- pokud $r = 1$ a $C < 1$, je konvergence *lineární*,
- pokud $r > 1$, je konvergence *superlineární*,
- pokud $r = 2$, je konvergence *kvadratická*,
- pokud $r = 3$, je konvergence *kubická*, atd.

Jeden z důvodů, proč rozlišujeme mezi lineární a superlineární konvergencí, je ten, že, asymptoticky pro velká k, lineárně konvergentní posloupnost získává po každé iteraci jistý stále stejný počet přesných číslic, kdežto superlineárně konvergující posloupnost v jednotlivých iteracích získává počet přesných číslic, který stále roste. Přesněji můžeme říci, že lineárně konvergentní posloupnost získává v každé iteraci $-\log(C)$ přesných číslic, kdežto superlineárně konvergující posloupnost má po každé iteraci r-krát více přesných číslic než měla před touto iterací. Speciálně pak platí, že u kvadraticky konvergentní metody se počet přesných číslic po každé iteraci zdvojnásobí (pro dostatečně velká k).

Příklad 7 (Rychlosti konvergence). Jestliže členy následujících posloupností představují velikosti chyb postupně generovaných iteračních aproximací, jsou rychlosti konvergence takové, jak je u jednotlivých posloupností uvedeno.

- $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, \ldots$ lineární, $C = 10^{-1}$
- $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, \ldots$ lineární, $C = 10^{-2}$
- $10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}, \ldots$ superlineární, ale ne kvadratická
- $10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}, \ldots$ kvadratická

V teorii numerických metod se dokazují věty o konvergenci, které nám umožňují říci, kdy pro danou rovnici ta či ona metoda konverguje a jak rychle. Nedávají nám ale explicitně pokyny

pro to, kdy máme iterační proces zastavit a prohlásit výsledné přibližné řešení za „dostatečně přesné“. Navrhnout vhodné zastavovací kritérium je poměrně složitá záležitost, a to z řady důvodů. Díky teorii můžeme v zásadě vědět, že se chyba $|e_k|$ postupně zmenšuje, ale protože neznáme přesné řešení, není tu možnost přímo zjistit, jak veliké $|e_k|$ je. Rozumnou náhražkou tu může sloužit relativní změna v postupných iteracích, tedy

$$\frac{|x_{k+1} - x_k|}{|x_k|}.$$

Pokud se tato veličina stane dostatečně malou, znamená to, že se přibližné hodnoty řešení už přestaly významně měnit a nemá tedy cenu pokračovat. Na druhé straně bychom chtěli mít jistotu, že jsme skutečně získali dobré přibližné řešení a že tedy aspoň je hodnota $f(x_k)$ přiměřeně malá. Jak už jsme si ale mohli povšimnout, obě dvě tyto uvedené veličiny nemusí být malé současně, roli tu hraje podmíněnost úlohy. Dále se zde projevuje také případná změna měřítka u proměnné x a funkce f. Ze všech těchto důvodů je vytvoření zcela spolehlivého zastavovacího kritéria velmi obtížné a musíme se také spoléhat na další informace, které o řešené úloze víme. U iteračních metod, které budeme vzápětí v této přednášce popisovat, proto zpravidla vynecháváme jakýkoli test na konvergenci a místo toho pouze naznačujeme jistý neurčený počet iterací s tím, že iterační proces je třeba ukončit poté, co se vyhoví určitému vhodnému kritériu, jehož volba je (bohužel) na uživateli.

2 Numerické metody řešení nelineárních rovnic

Budeme se zabývat numerickým (přibližným) řešením nelineární rovnice (1): pro danou spojitou funkci $f : \mathbb{R} \to \mathbb{R}$ hledáme bod $x^* \in \mathbb{R}$ takový, že $f(x^*) = 0$.

2.1 Metoda půlení intervalu neboli bisekce

V počítačové aritmetice s konečnou přesností nemusí existovat strojové číslo x^* takové, že $f(x^*)$ je přesně nula. Alternativní možnost je hledat nějaký velmi malý interval $[a, b]$, ve kterém f mění znaménko. Jak jsme již uvedli v odstavci 1.2, taková *uzávěra* zaručuje, že příslušná spojitá funkce musí někde uvnitř tohoto intervalu mít nulový bod. **Metoda půlení intervalu** neboli **metoda bisekce** začíná od nějaké počáteční uzávěry a postupně snižuje její velikost do té doby, až je řešení uzavřeno s požadovanou přesností (resp. tak, jak to aritmetika počítače dovolí). V každé iteraci se nejprve stanoví *střed* aktuálního intervalu a pro další iteraci se ponechá pouze jedna z polovin intervalu podle toho, jaké znaménko má funkční hodnota ve středu. Tato polovina pak tvoří opět (již kratší) uzávěr, s nímž vstupujeme do další iterace. Metodu bisekce formálně můžeme zapsat jako Algoritmus 1, v němž jako vstupní data figuruje funkce f, uzávěra $[a, b]$ a chybová tolerance Δ_{tol} pro délku výsledného intervalu obsahujícího kořen.

Uvedeme ještě pár poznámek k implementaci metody bisekce ve výše uvedeném algoritmu:

Především, zdá se, že nejpřirozenější vzorec pro výpočet středu intervalu $[a, b]$ by byl $m = (a + b)/2$. Jenže v počítačové aritmetice není v extrémních případech zaručeno, že takto počítaný bod m vůbec padne do intervalu $[a, b]$. Komu se to zdá divné, může si v aritmetice se dvěma desítkovými číslicemi zkusit podle tohoto vzorce spočítat střed intervalu $[0.67, 0.69]$ (vyjde $m = 0.7$). Kromě toho může u tohoto vzorce mezivýsledek $a + b$ překročit rozsah počítače i v situacích, kdy střed intervalu v rozsahu počítače leží. Jakkoli jde o extrémní případy, je na tomto jednoduchém příkladu vidět, že počítačová implementace algoritmů není jen pouhé

Algoritmus 1 Metoda půlení intervalu

Require: Funkce f, uzávěra $[a, b]$, chybová tolerance Δ_{tol}
Ensure: $x^* : f(x^*) \approx 0$
  while $(b - a) > \Delta_{\text{tol}}$ **do**
    $m \leftarrow a + \dfrac{b-a}{2}$
    if $\operatorname{sgn} f(a) = \operatorname{sgn} f(m)$ **then**
      $a \leftarrow m$
    else
      $b \leftarrow m$
    end if
  end while

přepisování vzorečků v nějakém vhodném programovacím jazyce. Vzorec použitý v Algoritmu 1 se uvedeným problémům vyhýbá.

Dále, pokud jde o testování toho, zda dvě hodnoty $f(x_1)$ a $f(x_2)$ mají stejné znaménko, je na počítači lepší používat funkci signum než matematicky ekvivalentní testování, zda součin $f(x_1) \cdot f(x_2)$ je kladný nebo záporný. Takový součin může totiž také překročit rozsah počítače směrem k nekonečnu a v okolí kořene směrem k nule. Poznamenejme pro pořádek, že je $\operatorname{sgn}(x) = 1$ pro $x \geq 0$ a $\operatorname{sign}(x) = -1$ pro $x < 0$.

Příklad 8 (Metoda bisekce). Metodu půlení intervalu ukážeme na příkladu hledání kořene rovnice

$$f(x) = x^2 - 4 \sin x = 0.$$

Jako počáteční uzávěru vezmeme interval $[a, b]$, kde $a = 1$ a $b = 3$. Záleží tu pouze na tom, aby se funkční hodnoty v těchto dvou bodech lišily ve znaménku. Vypočítáme hodnotu funkce ve středním bodě intervalu, tedy v $m = 2$ a zjistíme, že $f(m)$ má opačné znaménko než $f(a)$, takže si podržíme levou polovinu počátečního intervalu a položíme pro další krok $b = m$. Pak tento postup opakujeme tak dlouho, až se interval uzávěry zúží na požadovanou velikost. Následující tabulka ukazuje možnou posloupnost iterací.

a	$f(a)$	b	$f(b)$
1.000000	-2.365884	3.000000	8.435520
1.000000	-2.365884	2.000000	0.362810
1.500000	-1.739980	2.000000	0.362810
1.750000	-0.873444	2.000000	0.362810
1.875000	-0.300718	2.000000	0.362810
1.875000	-0.300718	1.937500	0.019849
1.906250	-0.143255	1.937500	0.019849
1.921875	-0.062406	1.937500	0.019849
1.929688	-0.021454	1.937500	0.019849
1.933594	-0.000846	1.937500	0.019849
1.933594	-0.000846	1.935547	0.009491
1.933594	-0.000846	1.934570	0.004320
1.933594	-0.000846	1.934082	0.001736

Interval, u kterého jsme iterace ukončili, má délku menší, než 0.0005, a můžeme tedy říci, že nalezený kořen je s touto přesností roven $x^* \approx 1.934$.

Na závěr ještě několik poznámek k metodě půlení intervalu.

- V metodě půlení se nikde nevyužívají velikosti funkčních hodnot, pouze jejich znaménka.
- Pokud výpočet začneme s uzávěrou spojité funkce, pak metoda konverguje vždy, ale dosti pomalu.
- V každé iteraci se délka uzávěry snižuje na polovinu, takže rychlost konvergence je *lineární*, s $r = 1$ a $C = 0.5$.
- V každé iteraci bisekce získáváme jednu další přesnou dvojkovou číslici v přibližném řešení.
- Pro daný počáteční interval $[a, b]$ je délka intervalu po k iteracích rovna $(b - a)/2^k$, takže k dosažení chybové tolerance ϵ_{tol} je zapotřebí zhruba

$$\log_2 \left(\frac{b-a}{\epsilon_{\text{tol}}} \right)$$

  iterací, nezávisle na vlastnostech použité funkce f.

2.2 Metoda postupných aproximací

Metoda postupných aproximací nebo také *metoda prosté iterace* slouží k hledání pevných bodů funkce g z rovnice (2). Připomeňme tedy, že pro funkci $g : \mathbb{R} \to \mathbb{R}$ se *pevným bodem* nazývá takové číslo x^* (pokud existuje), pro které platí

$$x^* = g(x^*).$$

Důvodem tohoto názvu je skutečnost, že x^* se po aplikaci funkce g nezmění. Zatímco u nelineární rovnice $f(x) = 0$ hledáme bod, v němž graf funkce f protíná osu x (tedy přímku $y = 0$), při hledání pevného bodu funkce g chceme najít bod, v němž graf funkce g protne diagonální přímku $y = x$. Úlohy na hledání pevného bodu dost často pocházejí přímo z praxe, ale pro nás zde mají význam také z toho důvodu, že řešení nelineární rovnice (1) lze zpravidla převést na hledání pevného bodu odpovídající nelineární funkce g, tedy na řešení rovnice (2). Metoda postupných aproximací (prosté iterace) pro řešení této rovnice je založena na opakovaném (iteračním) použití vzorce

$$x_{k+1} = g(x_k)$$

s vhodně zvoleným *počátečním přiblížením* (počáteční aproximací) x_0.

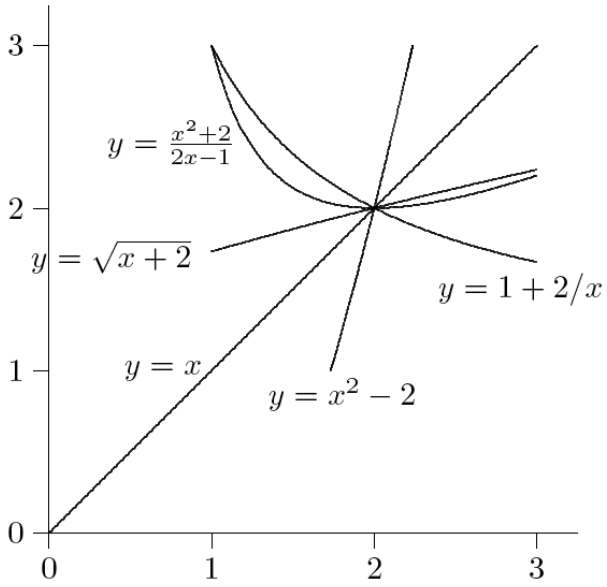
Chceme-li řešit rovnici $f(x) = 0$ metodou postupných aproximací, pak ji nejprve musíme převést na úlohu o pevném bodu pro nějakou vhodně vybranou funkci g. Takových možností bývá pro danou f více, ale ne všechny jsou stejně vhodné pro získání iteračního schématu k řešení výchozí rovnice. Výsledná iterační metoda se pro různé volby g může lišit nejen co do rychlosti konvergence, ale také v tom, zda vůbec konverguje či nikoli.

Příklad 9 (Úlohy na pevný bod). Nelineární rovnice

$$f(x) = x^2 - x - 2 = 0$$

má kořeny $x^* = 2$ a $x^* = -1$. Mezi ekvivalentní úlohy na hledání pevného bodu patří úlohy (2) s funkcemi (ověřte si to)

1. $g(x) = x^2 - 2$,



Obrázek 3: Pevný bod $(2, 2)$ nelineárních funkcí.

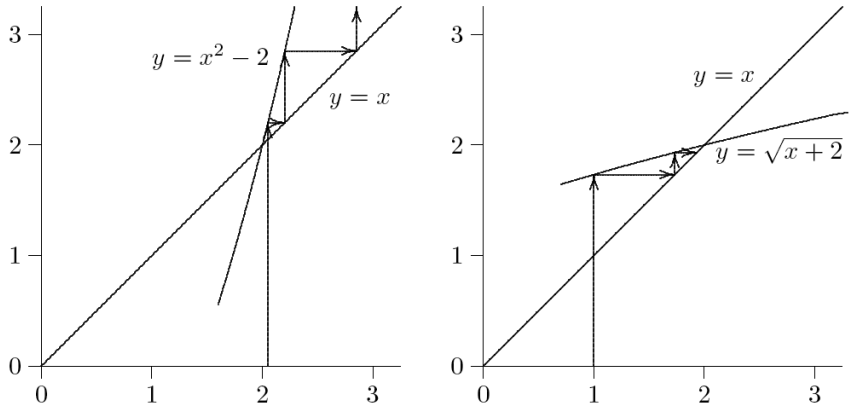
2. $g(x) = \sqrt{x+2}$ (ekvivalence pouze pro nezáporné pevné body, srv. (2)),
3. $g(x) = 1 + (2/x)$,
4. $g(x) = (x^2 + 2)/(2x - 1)$.

Na obrázku 3 je vykreslen průběh každé z těchto funkcí spolu s přímkou $y = x$. Všimněme si, že funkce g jsou konstruovány tak, že jejich grafy vesměs protínají přímku $y = x$ v pevném bodě $(2, 2)$.

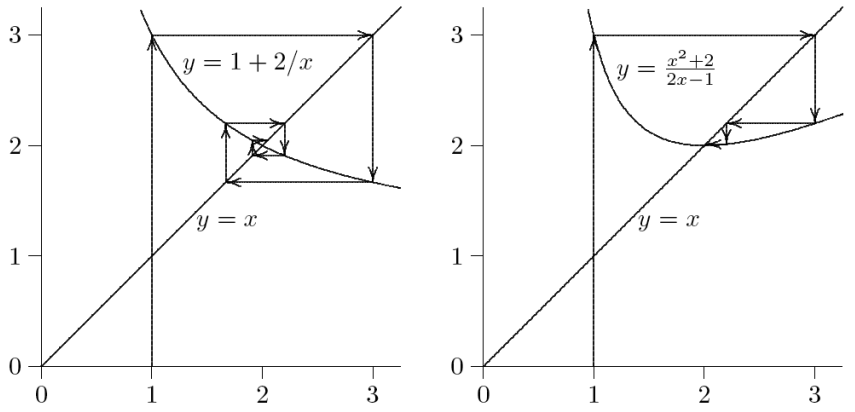
Průběh příslušných iteračních schémat metody postupných aproximací je graficky znázorněn na obrázcích 4 a 5. Šipka ve svislém směru odpovídá výpočtu hodnoty dané funkce v nějakém bodě a vodorovná šipka směřující k přímce $y = x$ vyznačuje, že se výsledek předchozího výpočtu hodnoty funkce g použije jeko argument pro příští výpočet funkční hodnoty. U první z uvedených funkcí vidíme, že i přes to, že počáteční bod je velmi blízko řešení, postupné aproximace divergují. U ostatních tří funkcí je vidět, že postupné iterace konvergují k pevnému bodu, i když byly odstartovány v bodě, který je od řešení relativně daleko. Zdá se přitom, že rychlosti konvergence pro tyto tři funkce se mohou lišit.

Jak lze z grafů funkcí na obrázcích 4 a 5 vidět, chování metody prosté iterace se může značně odlišovat, od divergence přes pomalou konvergenci k rychlé konvergenci. Nejjednodušší (i když ne nejobecnější) způsob, jak charakterizovat chování iteračního schématu $x_{k+1} = g(x_k)$ pro řešení úlohy na pevný bod tvaru $x = g(x)$, je pokusit se vzít v úvahu derivaci funkce g v hledaném řešení x^* za předpokladu, že funkce g je hladká a tato derivace existuje. Dá se ukázat, že pokud $x^* = g(x^*)$ a $|g'(x^*)| < 1$, pak iterační schéma metody postupných aproximací **lokálně konverguje**. To znamená, že existuje nějaký interval obsahující x^* takový, že metoda prosté iterace s funkcí g konverguje, pokud je odstartována z nějakého x_0, jež leží uvnitř tohoto intervalu. Říkáme také, že metoda konverguje pro dostatečně blízké počáteční přiblížení. Naproti tomu pokud $|g'(x^*)| > 1$, pak metoda prosté iterace diverguje pro jakékoli počáteční přiblížení kromě x^*.

Důkaz tohoto tvrzení je založen na větě o střední hodnotě funkce, ale z časových důvodů jej zde neuvádíme, jakkoli není složitý (viz [2], [3] nebo [1]). Plyne z něj ale také to, že pokud metoda



Obrázek 4: Metoda postupných aproximací pro první a druhou funkci g.



Obrázek 5: Metoda postupných aproximací pro třetí a čtvrtou funkci g.

konverguje, je její asymptotická rychlost konvergence lineární s konstantou $C = |g'(x^*)|$. Čím menší je tato konstanta, tím je konvergence rychlejší, a ideální by tedy pro danou rovnici (1) bylo najít ekvivalentní formulaci (2) s funkcí g, pro niž by platilo $g'(x^*) = 0$. V takovém případě se dá pomocí Taylorova rozvoje opět poměrně snadno ukázat, že konvergence je nejméně kvadratická. V příštím odstavci si popíšeme jeden systematický způsob takové volby funkce g pro rovnici $f(x) = 0$.

Příklad 10 (Konvergence metody postupných aproximací). Pro čtyři úlohy na pevný bod z předcházejícího příkladu máme následující výsledky:

1. $g'(x) = 2x$, takže $g'(2) = 4$ a metoda postupných aproximací tedy diverguje.
2. $g'(x) = 1/(2\sqrt{x+2})$, takže $g'(2) = 1/4$ a metoda postupných aproximací konverguje linárně s konstantou $C = 1/4$. Kladné znaménko derivace $g'(2)$ vede k tomu, že se iterace přibližují k pevnému bodu z jedné strany.
3. $g'(x) = -2/x^2$, takže $g'(2) = -1/2$ a metoda postupných aproximací konverguje lineárně s konstantou $C = 1/2$. Záporné znaménko derivace $g'(2)$ vede k tomu, že se iterace přibližují k pevnému bodu po spirále, střídavě vždy z opačné strany.
4. $g'(x) = (2x^2 - 2x - 4)/(2x - 1)^2$, takže $g'(2) = 0$ a metoda postupných aproximací konverguje kvadraticky.

2.3 Newtonova metoda

Metoda bisekce nepoužívá jiné vlastnosti funkčních hodnot než jejich znaménka, což vede k tomu, že konverguje vždy, ale pomalu. Pokud se využijí také velikosti funkčních hodnot, můžeme odvodit rychleji konvergující metody, které nám v každé iteraci budou dávat přesnější aproximaci kořene řešené rovnice. V první řadě se zde využívá aproximace funkce f vystupující v rovnici pomocí prvních dvou členů jejího Taylorova rozvoje, tedy

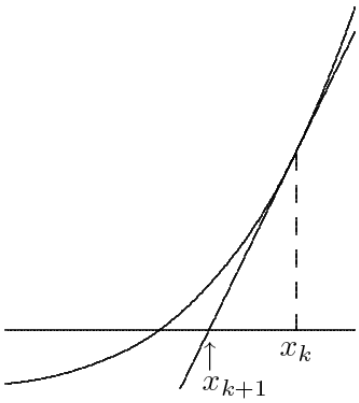
$$f(x+h) \approx f(x) + f'(x)h,$$

což je lineární funkce h, která aproximuje f v okolí bodu x. Nahradíme tudíž nelineární funkci f touto lineární funkcí, jejíž nulový bod v h se snadno vypočítá, je to $h = -f(x)/f'(x)$, pokud ovšem $f'(x) \neq 0$. Je jasné, že kořeny obou těchto funkcí nejsou obecně identické, takže popsaný postup musíme iteračně opakovat. To vede k iterační metodě, které se říká **Newtonova metoda** (nebo také *Newtonova-Raphsonova*), jejíž algoritmus uvádíme jako Algoritmus 2.

Algoritmus 2 Newtonova metoda

Require: Funkce f, počáteční aproximace x_0, chybová tolerance Δ_{tol}
Ensure: $x^* : f(x^*) \approx 0$
  $k \leftarrow 0$
  while $|f(x_k)| > \Delta_{\text{tol}}$ **do**
    $x_{k+1} \leftarrow x_k + f(x_k)/f'(x_k)$
    $k \leftarrow k + 1$
  end while

Ne obrázku 6 ukazujeme, že Newtonova metoda se dá interpretovat jako aproximace funkce f poblíž x_k tečnou ke grafu funkce vedenou v bodě $(x_k, f(x_k))$. Jako další aproximaci řešení pak bereme nulový bod této lineární tečné funkce a proces postupně opakujeme. Někdy se Newtonově metodě proto také říká *metoda tečen*.



Obrázek 6: Newtonova metoda pro řešení nelineární rovnice.

Příklad 11 (Newtonova metoda). Newtonovu metodu předvedeme opět na hledání kořene rovnice

$$f(x) = x^2 - 4\sin x = 0.$$

Derivace této funkce je

$$f'(x) = 2x - 4\cos x,$$

takže iterační schéma je dáno vzorcem

$$x_{k+1} = x_k - \frac{x_k^2 - 4\sin x_k}{2x_k - 4\cos x_k}.$$

Jako počáteční přiblížení zvolíme $x_0 = 3$ a postupně obdržíme posloupnost iterací, která je uvedena dále. Přitom $h_k = -f(x_k)/(x_k)$ označuje změnu x_k v každé iteraci. Iterační proces můžeme ukončit, když bude $|h_k|/|x_k|$ nebo $|f(x_k)|$, nebo obojí, menší než námi předepsaná tolerance.

k	x_k	$f(x_k)$	$f'(x_k)$	h_k
0	3.000000	8.435520	9.959970	-0.846942
1	2.153058	1.294772	6.505771	-0.199019
2	1.954039	0.108438	5.403795	-0.020067
3	1.933972	0.001152	5.288919	-0.000218
4	1.933754	0.000000	5.287670	0.000000

Na Newtonovu metodu se můžeme také dívat jako na speciální způsob převodu nelineární rovnice $f(x) = 0$ na úlohu o pevném bodě pro jistou funkci g, tedy $x = g(x)$, kde za funkci g volíme

$$g(x) = x - f(x)/f'(x).$$

a pevný bod hledáme metodou postupných aproximací. Abychom vyšetřili konvergenci metody, potřebujeme tedy nejprve znát derivaci funkce g, což je po úpravě

$$g'(x) = f(x)f''(x)/(f'(x))^2$$

(pokud $f'(x) \neq 0$). Je-li tedy x^* jednoduchý kořen, tj. $f(x^*) = 0$ a $f'(x^*) \neq 0$, pak $g'(x^*) = 0$. Newtonova metoda má tedy pro jednoduché kořeny asymptoticky kvadratickou rychlost konvergence, tedy $r = 2$.

Kvadratická rychlost konvergence Newtonovy metody znamená, že asymptoticky (v blízkosti kořene) se chyba metody po každé iteraci umocní na druhou. Jinak také můžeme říci, že se počet přesných (správných) číslic přibližného řešení po každé iteraci zdvojnásobí. Naproti tomu pro násobné kořeny je Newtonova metoda pouze lineárně (lokálně) konvergentní s konstantou $C = 1-(1/m)$, kde m je násobnost počítaného kořene. Opět ale musíme zdůraznit, že tyto úvahy o konvergenci platí pouze lokálně v nějakém větším nebo menším okolí hledaného kořene a že Newtonova metoda, která není odstartována dostatečně blízko ke kořeni, nemusí konvergovat vůbec. Jednoduchý příklad je situace, kdy někdy během iterací bude $f'(x_k)$ relativně malé (graf funkce f bude mít v bodě x_k téměř vodorovnou tečnu) a v důsledku toho bude následující iterace mít tendenci ležet někde daleko od posledního přiblížení.

Příklad 12 (Newtonova metoda pro násobný kořen). Následující dva příklady ukazují oba typy výše popsaného chování Newtonovy metody. První z nich ukazuje kvadratickou konvergenci k jednoduchému kořenu, druhý lineární konvergenci k násobnému kořenu. Násobnost kořene ve druhém z uvedených příkladů je 2, takže $C = 1/2$.

	$f(x) = x^2 - 1$	$f(x) = x^2 - 2x + 1$
k	x_k	x_k
0	2.0	2.0
1	1.25	1.5
2	1.025	1.25
3	1.0003	1.125
4	1.00000005	1.0625
5	1.0	1.03125

2.4 Metoda sečen

Jistou nevýhodou Newtonovy metody je, že za její kvadratickou konvergenci platíme tím, že v každém iteračním kroku musíme kromě funkční hodnoty počítat také hodnotu derivace. Výpočet hodnot derivace přitom může být nepohodlný nebo časově náročný, takže bychom mohli uvažovat o tom, že hodnoty derivací budeme nahrazovat diferenčními podíly vyplývajícími z definice derivace funkce, tedy bychom mohli pro vhodné dostatečně malé h klást například

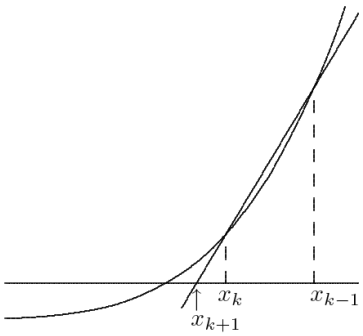
$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

To by ovšem znamenalo počítat v každé iteraci jednu funkční hodnotu navíc, a to jen proto, abychom získali přibližnou informaci o hodnotě derivace. Lepší je založit podobnou diferenční aproximaci derivace na funkčních hodnotách, které jsme už během iterací stejně vypočítali, a klást

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

Tento postup vede k *metodě sečen*, jejíž algoritmus uvádíme jako Algoritmus 3. Na obrázku 7 vidíme, že metoda sečen se dá interpretovat jako aproximování funkce f přímkou procházející předchozími dvěma iteracemi, tedy sečnou, přičemž za nové přiblížení bereme nulový bod této lineární funkce. Na rozdíl od Newtonovy metody zde ovšem potřebujeme dvě počáteční aproximace.

Algoritmus 3 Metoda sečen



Obrázek 7: Metoda sečen pro řešení nelineární rovnice.

$x_0, x_1 =$ počáteční aproximace
for $k = 0, 1, 2, \ldots$
$\quad x_{k+1} = x_k - f(x_k)(x_k - x_{k-1})/(f(x_k) - f(xk - +))$
end

Příklad 13 (Metoda sečen). Metodu sečen budeme ilustrovat opět na hledání kořene rovnice

$$f(x) = x^2 - 4\sin x = 0.$$

Za potřebná dvě počáteční přiblížení vezmeme $x_0 = 1$ a $x_1 = 3$, vypočítáme příslušné funkční hodnoty a za další přibližné řešení vezmeme průsečík přímky spojující tyto dvě funkční hodnoty s nulou. Celý postup pak opakujeme, přičemž použijeme toto nově získané přiblížení kořene a tu novější ze dvou předcházejích iterací, takže v každém iteračním kroku potřebujeme vypočítat pouze jednu novou funkční hodnotu. Posloupnost provedených iterací je uvedena v tabulce, kde h_k označuje změnu x_k v příslušné iteraci.

k	x_k	$f(x_k)$	h_k
0	1.000000	-2.365884	
1	3.000000	8.435520	-1.561930
2	1.438070	-1.896774	0.286735
3	1.724805	-0.977706	0.305029
4	2.029833	0.534305	-0.107789
5	1.922044	-0.061523	0.011130
6	1.933174	-0.003064	0.000583
7	1.933757	0.000019	-0.000004
8	1.933754	0.000000	0.000000

Protože každé nové přibližné řešení, které dává metoda sečen, závisí na *dvou* předchozích iteracích, je vyšetřování konvergence metody o něco složitější a detaily jsme nuceni zde vypustit. Uvádíme alespoň, že se dá dokázat, že chyby metody splňují pro jistou kladnou konstantu $c > 0$ vztah

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k| \cdot |e_{k-1}|} = c,$$

což znamená, že posloupnost iterací metodou sečen lokálně konverguje a rychlost konvergence je superlineární. Přesněji (viz [2], [3] a [1]) se dá ukázat, že asymptotická rychlost konvergence

metody sečen je[1]

$$r = \frac{1+\sqrt{5}}{2} \approx 1{,}618.$$

Stejně jako u Newtonovy metody je i u metody sečen ke konvergenci nutno iterace odstartovat dostatečně blízko kořene.

Porovnáme-li metodu sečen s Newtonovou metodou, vidíme, že metoda sečen má výhodu v tom, že v každé iteraci potřebuje vypočítat pouze jednu novou funkční hodnotu. Za nevýhodu bychom mohli považovat to, že vyžaduje dvě startovací hodnoty a že vůči Newtonově metodě konverguje pomaleji, i když stále superlineárně. Menší pracnost provedení jedné iterace vyváží u metody sečen zpravidla to, že k dosažení konečného výsledku musíme provést větší počet iterací. Dá se tedy říci, že nalezení přibližné hodnoty řešení nelineární rovnice metodou sečen je často méně pracné než použití Newtonovy metody.

3 Dodatky

3.1 Bezpečné metody

Rychle konvergující metody pro numerické řešení nelineárních rovnic jako jsou například Newtonova metoda či metoda sečen (další takové metody lze najít v literatuře [2, 3, 1]) nejsou bezpečné v tom smyslu, že pokud nejsou odstartovány dostatečně blízko kořene, nemusí konvergovat. Bezpečnou metodou v tomto smyslu je metoda půlení intervalu, která je ale pomalá a tedy nákladná. Jakou metodu tedy volit?

Řešením tohoto dilematu jsou hybridní metody, které jsou zahrnuty ve většině moderního matematického softwaru a které v sobě kombinují vlastnosti obou výše popsaných typů metod. Jejich algoritmy jsou ale ovšem složitější. Tyto metody mohou například pracovat s rychle konvergentní metodou a přitom docílit toho, že iterace zůstávají uvnitř počáteční uzávěry kořene. Pokud následující aproximace řešení rychlým algoritmem padne mimo interval uzávěry, vrátíme se a provedeme jednu iteraci bezpečnou metodou, například bisekcí. Pak se může zkusit opět použitá rychlá metoda, tentokrát ovšem už na menším intervalu a s větší nadějí na úspěch. ke konci výpočtu už by měly iterace běžet tou rychlou metodou. Uvedený postup je jen zřídka horší než použitá pomalá metoda, zpravidla je mnohem rychlejší.

Populární implementace výše popsaného hybridního postupu dnes pochází od Brenta (v literatuře také tedy Brentova metoda) a kombinuje v sobě bezpečí bisekce s rychlejší konvergencí tzv. **inverzní kvadratické interpolace** (více k tomu viz [1]). Díky tomu, že se zde vyhýbáme Newtonově metodě, nejsou k výpočtu zapotřebí hodnoty derivace. Soudobý kvalitní software musí při implementaci metody vzít v úvahu také to, že se její algoritmus realizuje v počítačové aritmetice, tedy např. ohlídat možná překročení rozsahu počítače nebo nepřiměřeně přísné požadavky na přesnost výsledku. Dobrou implementaci výše popsaného postupu představuje například funkce `fzero` v Matlabu.

Poznamenáváme ještě, že jakousi kombinací metody bisekce a metody sečen je metoda *regula falsi* (z lat., doslova pravidlo falše). Každý její krok začíná tím, že body x_k a x_{k-+} tvoří uzávěru hledaného kořene, ale místo aby se v každém kroku interval uzávěry půlil, vypočítá se nejprve x_{k+1} pomocí vzorce metody sečen. Průběh funkce se tedy na daném intervalu opět nahradí sečnou. Pak se z takto získaných tří bodů zachovají ty dva, v nichž má funkce f opačná znaménka, a postup se opakuje. Metoda regula falsi je další vždy konvergentní metodou, musíme ji

[1] Pokud vám to číslo připadá povědomé, připomínám, že je to hodnota zlatého řezu.

ovšem odstartovat z uzávěry kořene. Její konvergence je pouze lineární a může, ale nemusí být rychlejší než metoda půlení. Lze také ukázat, že v některých případech může jeden z krajních bodů uzávěry zůstávat během iterací trvale beze změny a ačkoli druhý bod konverguje ke kořenu rovnice, uzávěra se nemůže zmenšit pod jistou mez.

3.2 Numerický výpočet kořenů polynomu

Až dosud jsme se zabývali metodami pro nalezení jednoho nulového bodu obecné reálné funkce jedné reálné proměnné. Pokud je uvažovaná funkce polynom $p(x)$ stupně n, pak potřebujeme často najít všechny jeho nulové body, z nichž některé mohou být komplexní, i když polynom sám má reálné koeficienty. O kořenech polynomů nám algebraická teorie říká podrobnější informace než známe o nulových bodech obecných funkcí. Především je zde tzv. *základní věta algebry*, podle níž každý polynom stupně n má v komplexní rovině právě n nulových bodů (kořenů), pokud každý z nich počítáme tolikrát, kolik činí jeho násobnost. Dále se dá ukázat, že pokud má reálný polynom komplexní kořeny, vyskytují se tyto kořeny vždy ve dvojicích komplexně sdružených čísel, tedy jako $x \pm ßy$.

Pro hledání kořenů polynomů není nezbytné používat komplexní aritmetiku, leckdy lze počítat jejich reálné a imaginární části x a y odděleně. Pro výpočet kořenů polynomů existuje řada možností:

- Použijeme některou z popsaných obecných metod(např. Newtonovu metodu) a nalezneme jeden kořen x_1. Pak dále pracujeme s redukovaným polynomem $p(x)/(x-x_1)$, jehož stupeň je o jedničku nižší. Postup opakujeme tak dlouho, dokud nestanovíme všechny kořeny. Metoda se komplikuje, pokud narazíme na komplexní kořen.
- K danému polynomu sestavíme jeho *doprovodnou matici*, což je speciální matice mající vlastní čísla shodná s kořeny polynomu. Pak nějakou vhodnou numerickou metodou algebry stanovíme jako kořeny daného polynomu vlastní čísla této matice. Tento postup, který je použit ve funkci `roots` v Matlabu, je spolehlivý, ale není tak efektivní jako použití numerických metod odvozených speciálně pro výpočet kořenů polynomu.
- Použijeme některou ze speciálních metod pro výpočet nulových modů polynomů. Najdou se mezi nimi jak bezpečné metody, které izolují kořeny například ve sjednoceni disků v komplexní rovině (ty jsou ovšem podobně jako bisekce pouze lineárně konvergentní), tak rychle konvergující metody (i rychlejší než je Newtonova metoda). O těchto speciálních metodách se lze poučit například v [2, 3].

3.3 Numerické řešení soustav nelineárních rovnic

Řešení soustav nelineárních rovnic je obtížnější, než je tomu u jedné rovnice, a to z řady důvodů:

- Chování soustavy může být mnohem rozmanitější než chování jedné rovnice (a jejich kořenů). Teoretická analýza existence a počtu řešení je tak mnohem složitější.
- Konvenční metody používané pro jednu rovnici se leckdy dají víceméně přímočaře zobecnit i pro soustavy, ale u soustav není jednoduchý způsob, jak zobecnit pojem uzávěry řešení, takže zde není jednoduché sestrojit bezpečné, globálně konvergující metody. Určité možnosti zde ale existují, nicméně se vymykají možnostem tohoto textu a nenajdou se

ani v běžných učebnicích. Nicméně v Matlabu je pro řešení soustav nelineárních rovnic k dispozici vcelku spolehlivá funkce `fsolve`.

- Pracnost numerického řešení soustav nelineárních rovnic roste nelineárně s počtem neznámých. Tak například jeden iterační krok Newtonovy metody pro soustavu o n neznámých znamená obecně výpočet n^2 hodnot derivací a jedno řešení souatavy n lineárních rovnic o n neznámých, což je samo o sobě obecně řádově n^3 aritmetických operací. Také organizační struktura algoritmů je mnohem složitější.

Jak jsme uvedli již na začátku tohoto textu, studium problematiky soustav se zde vymyká našim časovým možnostem. Úvodní informace může zájemce najít v doporučené literatuře [2], [3], [1].

Reference

[1] Michael T. Heath. *Scientific computing: an introductory survey.* McGraw-Hill, Boston, 2 edition, 2002.

[2] Stanislav Míka. *Numerické metody algebry.* MVŠT. SNTL, Praha, 1982.

[3] Stanislav Míka and Marek Brandner. *Numerické metody I.* FAV ZČU, Plzeň, 2. edition, 2002.