UČEBNÍ TEXTY VYSOKÝCH ŠKOL

Vysoké učení technické v Brně Fakulta strojního inženýrství

Doc. RNDr. Libor Čermák, CSc.
RNDr. Rudolf Hlavička, CSc.

Numerické metody

AKADEMICKÉ NAKLADATELSTVÍ CERM, s.r.o. Brno

ISBN 978-80-214-5437-8

Obsah

Předmluva

Tato skripta jsou určena studentům prvního ročníku Fakulty strojního inženýrství VUT v Brně pro studium předmětu *Numerické metody I*. Skripta jsou věnována tradičním tématům numerické matematiky: zdrojům chyb a jejich šíření, řešení soustav lineárních rovnic, aproximaci funkcí, numerickému derivování a integrování, řešení nelineárních rovnic a minimalizaci funkcí.

Předpokládá se, že čtenář skript má základní znalosti z lineární algebry, z diferenciálního a integrálního počtu funkcí jedné proměnné a z programování. Funkce více proměnných se v tomto textu vyskytují jen v odstavcích 3.1.3, 4.3.3, 5.3 a 6.2 a k pochopení zde probírané látky postačí znalosti průběžně získávané v paralelně probíhajícím kurzu *MATEMATIKA II*, věnovaném převážně diferenciálnímu a integrálnímu počtu funkcí více proměnných.

Do skripta jsme ke každému tématu zařadili klasické metody, standardně uváděné v každém úvodním kurzu numerické matematiky, které slouží především k pochopení a ilustraci dané problematiky. Klasické metody jsou dnes již často překonány efektivnějšími postupy. Potíž s moderními, v současnosti používanými algoritmy, je však v tom, že bývají často poměrně komplikované, takže porozumět jim nebývá snadné. Přesto se alespoň o některých moderních algoritmech v tomto textu stručně zmiňujeme.

Většinu metod jsme se pokusili zdůvodnit, preciznímu dokazování jsme se však úmyslně vyhýbali. U některých tvrzení uvádíme literární zdroj, v němž lze najít odpovídající vysvětlení. Jde-li však o tvrzení natolik běžné, že ho lze najít prakticky v jakékoliv základní učebnici numerických metod, pak literární zdroj neuvádíme.

Řešené příklady jsme volili tak, aby pomáhaly pochopit a používat někdy poněkud těžko stravitelné formule a postupy.

Skripta obsahují celou řadu algoritmů, řešených příkladů a úloh k samostatnému procvičení. Většina úloh se dá jen těžko zvládnout bez využití počítače. Proto se předpokládá, že studenti si některé algoritmy sami v MATLABu naprogramují.

Při zpracování skript jsme vycházeli z osvědčených učebnic numerické matematiky, jakými jsou např. knihy [3], [22], ale také z modernějších knih [17], [7], [11] a [15]. Zvláště posledně citovaná kniha Molerova, která je volně ke stažení na Internetu, byla pro nás velkou inspirací. Pokud jde o české zdroje, nejvíce podnětů jsme čerpali z knih [19], [14], [8]. Moderní česky psaná monografie numerických metod v současnosti není k dispozici. Kromě výše uvedených knih lze však v češtině číst také [20], vybrané kapitoly věnované numerickým metodám v [21] a [23].

Doufáme, že se Vám, milý čtenáři, numerické metody zalíbí, a že je dokážete později efektivně využít při řešení konkrétních technických problémů. Mějte ale prosím pořád na zřeteli, že to, co se o numerických metodách v tomto textu dozvíte, je opravdu jen naprostý základ. Zbytek je už na Vás, zdrojů informací je dostatek, zejména díky Internetu. Základní orientace v numerických metodách Vám pak umožní kvalifikovaně používat nepřeberné množství programů, ať už volně dostupných na Internetu nebo programů komerčních. A to se může hodit.

Brno, listopad 2016

Libor Čermák
Rudolf Hlavička

1. Úvod do problematiky numerických metod

Při řešení problémů reálného světa se stále častěji setkáváme s potřebou popsat zkoumanou skutečnost pomocí věrohodného *matematického modelu* a ten pak uspokojivě vyřešit. Žijeme v době počítačů a tak je přirozené, že k realizaci matematického modelu počítač využijeme. Počítače umí pracovat velmi rychle s informacemi kódovanými pomocí čísel. A právě zde je místo pro *numerickou matematiku* (v angličtině *numerical analysis*) jakožto vědní disciplínu, která vyvíjí a analyzuje metody, jejichž technologickým jádrem jsou manipulace s čísly. V posledních letech se v anglicky psané literatuře místo termínu numerical analysis stále častěji používá termín *scientific computing* (odpovídající český termín nám bohužel není znám).

Když chceme metodami numerické matematiky vyřešit daný problém popsaný obecným matematickým modelem, musíme takový model nejdříve digitalizovat, to jest formulovat ho ve tvaru *numerické úlohy*, jejíž vstupní i výstupní data jsou čísla. *Numerická metoda* je postup řešení numerické úlohy. Přesný popis kroků realizujících numerickou metodu označujeme jako *algoritmus numerické metody*. Lze ho vyjádřit jako posloupnost akcí (proveditelných na počítači), které k danému (přesně specifikovanému konečnému) souboru vstupních čísel jednoznačně přiřadí odpovídající (přesně specifikovaný konečný) soubor výstupních čísel.

Příprava rozsáhlých souborů vstupních dat bývá označována jako *preprocessing*. Rozsah souboru výsledných údajů je často ohromný, pro člověka nestravitelný, a proto je třeba výsledky vhodně zpřístupnit tak, aby je zadavatel výpočtu byl vůbec schopen vyhodnotit. Metodám, které to provádějí, se říká *postprocessing*. Jednou z forem postprocessingu je vizualizace výsledků.

Jako příklad problému ze života uvažujme předpověď počasí. Pohyb vzduchu v atmosféře dovedeme alespoň přibližně popsat pomocí soustav parciálních diferenciálních rovnic a vhodných doplňujících podmínek. Metodami numerické matematiky dokážeme tyto rovnice přibližně řešit. Potřebná vstupní data se získávají pomocí družic a pozemních meteorologických stanovišť. Výsledky numerických výpočtů zpracované do animovaných meteorologických map pak sledujeme v televizní předpovědi počasí.

Při řešení reálných problémů téměř nikdy nezískáme přesné řešení, musíme se spokojit jen s řešením přibližným, které je zatíženo chybami. Naším cílem je organizovat výpočet tak, aby celková chyba byla co nejmenší.

Především se musíme vyvarovat hrubých *lidských chyb*, které vyplývají z nepochopení problému a z nepozornosti nebo nedbalosti člověka při jeho řešení.

Chyba matematického modelu. Při vytváření matematického modelu reálného problému provádíme vždy jisté idealizace. Rozdíl mezi řešením idealizovaného problému a řešením problému reálného nazýváme *chybou matematického modelu*. Do této kategorie chyb zahrnujeme také *chyby ve vstupních údajích*.

Příklad. Máme určit povrch zemského pláště. K výpočtu použijeme vzorec $S = 4\pi r^2$ pro povrch koule o poloměru r. Chyba modelu spočívá v předpokladu, že Země je koule.

Chyba numerické metody. Jestliže k řešení (numerické) úlohy použijeme numerickou metodu, která nám neposkytne přesné (teoretické) řešení dané úlohy, pak chybu, které se dopustíme, nazýváme *chybou numerické metody*. Důležitou součástí návrhu numerické

metody je *odhad chyby numerické metody.*

Příklad. Máme spočítat hodnotu funkce $\sin 1$ sečtením konečného počtu členů Taylorovy řady

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \cdots$$

pro $x = 1$. Je známo, že sečtením prvních tří členů řady se dopustíme chyby velikosti nejvýše $1/7!$, obecně sečtením prvních n členů se dopustíme chyby nejvýše $1/(2n+1)!$.

Zaokrouhlovací chyby. Při práci na počítači můžeme k reprezentaci čísel použít jen konečný počet cifer. Pracujeme proto s přibližnými hodnotami čísel, které dostaneme zaokrouhlením přesných hodnot. *Zaokrouhlovací chyby* vznikají už při vkládání dat do počítače, další pak vznikají při číselných výpočtech. Při špatně organizovaném výpočtu může dojít v důsledku nahromadění zaokrouhlovacích chyb k naprostému znehodnocení výsledku, viz příklad 1.7.

Příklad. Číslo π neumíme do počítače vložit přesně. Také výsledek operace, při níž číslo 2 dělíme číslem 3, nezobrazíme na standardním počítači pracujícím s binárními čísly přesně.

Je třeba mít na paměti, že při řešení reálného problému vystupují obvykle všechny chyby současně.

1.1. Chyby v numerických výpočtech

Absolutní a relativní chyba. Ve výpočtech jsme často nuceni nahradit přesné číslo x přibližným číslem $\tilde{x}$. Číslo $\tilde{x}$ potom nazýváme *aproximací čísla* x. Rozdíl $\tilde{x} - x = \Delta x$ nazýváme *absolutní chybou aproximace* $\tilde{x}$ a číslo

$$\frac{\Delta x}{x} = \frac{\tilde{x} - x}{x}, \qquad x \neq 0,$$

nazýváme *relativní chybou aproximace* $\tilde{x}$. Pro $|\Delta x| \leq \varepsilon$ se používá také symbolický zápis $\tilde{x} = x \pm \varepsilon$ a míní se tím, že $x - \varepsilon \leq \tilde{x} \leq x + \varepsilon$. Podobně se pro $|\Delta x/x| \leq \delta$ používá zápis $\tilde{x} = x(1 \pm \delta)$. Absolutní hodnota relativní chyby se často uvádí v procentech.

Nyní posoudíme chybu, které se dopustíme při výpočtu hodnoty $f(x_1, x_2, \ldots, x_n)$ funkce f, když přesné hodnoty x_i nahradíme přibližnými hodnotami $\tilde{x}_i = x_i + \Delta x_i$. Z Taylorova rozvoje $f(\tilde{\mathbf{x}})$ okolo bodu $\mathbf{x}$ dostaneme

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

Považujeme-li součiny chyb $\Delta x_i \Delta x_j$ za malé, máme pro absolutní chybu

$$|\Delta f(\mathbf{x})| := |f(\tilde{\mathbf{x}}) - f(\mathbf{x})| \doteq \left| \sum_{i=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_i} \Delta x_i \right| \leq \sum_{i=1}^{n} \left| \frac{\partial f(\mathbf{x})}{\partial x_i} \right| \cdot |\Delta x_i| \tag{1.1}$$

a pro chybu relativní

$$\left|\frac{\Delta f(\mathbf{x})}{f(\mathbf{x})}\right| \doteq \left|\sum_{i=1}^{n} \frac{x_i}{f(\mathbf{x})}\frac{\partial f(\mathbf{x})}{\partial x_i}\frac{\Delta x_i}{x_i}\right| \leq \sum_{i=1}^{n} \left|\frac{x_i}{f(\mathbf{x})}\frac{\partial f(\mathbf{x})}{\partial x_i}\right| \cdot \left|\frac{\Delta x_i}{x_i}\right| . \tag{1.2}$$

Při praktických odhadech se hodnota funkce f a hodnoty jejích derivací $\partial f/\partial x_i$ na pravých stranách přibližných nerovností (1.2) a (1.1) počítají v bodě $\tilde{\mathbf{x}}$.

Chyby základních aritmetických operací. Zvolíme-li $f(x, y) = x \pm y$, dostaneme pro absolutní a relativní chybu součtu a rozdílu

$$\Delta(x \pm y) \doteq \Delta x \pm \Delta y\,, \qquad \frac{\Delta(x \pm y)}{x \pm y} \doteq \frac{x}{x \pm y}\frac{\Delta x}{x} \pm \frac{y}{x \pm y}\frac{\Delta y}{y}\,. \tag{1.3}$$

Pro vyjádření chyby součinu volíme $f(x, y) = xy$ a obdržíme

$$\Delta(xy) \doteq y\Delta x + x\Delta y\,, \qquad \frac{\Delta(xy)}{xy} \doteq \frac{\Delta x}{x} + \frac{\Delta y}{y} \tag{1.4}$$

a pro chybu podílu dostaneme volbou $f(x, y) = x/y$

$$\Delta\left(\frac{x}{y}\right) \doteq \frac{1}{y}\Delta x - \frac{x}{y^2}\Delta y\,, \qquad \frac{\Delta(x/y)}{x/y} \doteq \frac{\Delta x}{x} - \frac{\Delta y}{y}\,. \tag{1.5}$$

Všimněte si, že relativní chyba součtu resp. rozdílu může být výrazně větší než relativní chyby operandů v případě, když $|x \pm y|$ je podstatně menší než $|x|$ nebo $|y|$. Při dělení malým číslem je (díky druhé mocnině y ve jmenovateli) významná chyba absolutní.

Platné dekadické cifry. Nechť $\tilde{x}$ je aproximace čísla x, kterou zapišme v mocninném dekadickém rozvoji jako

$$\tilde{x} = \pm\,[\,d_1 \cdot 10^e + d_2 \cdot 10^{e-1} + \cdots + d_k \cdot 10^{e+1-k} + d_{k+1} \cdot 10^{e-k} + \ldots\,]\,, \qquad d_1 \neq 0\,.$$

Řekneme, že *$k-$tá dekadická cifra d_k* aproximace $\tilde{x}$ *je platná*, jestliže

$$|\tilde{x} - x| \leq 5 \cdot 10^{e-k}, \tag{1.6}$$

tj. když se $\tilde{x}$ liší od x nejvýše o 5 jednotek řádu příslušného následující cifře. Platí-li nerovnost (1.6) pro $k \leq p$, ale pro $k = p + 1$ už neplatí, říkáme, že $\tilde{x}$ *má p platných cifer*. Číslo $\tilde{x} = \pm\, d_1{,}d_2 d_3 \ldots d_p \cdot 10^e$, které má všech p cifer platných, je *správně zaokrouhlenou hodnotou* čísla x.

Platná desetinná místa. Řekneme, že aproximace $\tilde{x}$ čísla x má *k-té desetinné místo platné*, jestliže

$$|\tilde{x} - x| \leq 5 \cdot 10^{-k-1}, \tag{1.7}$$

tj. když se $\tilde{x}$ liší od x nejvýše o 5 jednotek řádu příslušného následujícímu desetinnému místu. Platí-li nerovnost (1.7) pro $k \leq p$, ale pro $k = p + 1$ už neplatí, říkáme, že $\tilde{x}$ má *p platných desetinných míst*. Ve správně zaokrouhleném čísle je tedy každé desetinné místo platné.

V následující tabulce uvádíme několik příkladů:

x	$\tilde{x}$	platné cifry	platná desetinná místa
284	290	1	–
−45,8472	−45,798	3	1
100,002	99,9973	4	2
99,9973	100,002	5	2
−0,003728	−0,0041	1	3
$1{,}841 \cdot 10^{-6}$	$2{,}5 \cdot 10^{-6}$	0	5

Při odečítání dvou blízkých čísel dochází ke ztrátě platných cifer, jak o tom svědčí

Příklad 1.1. Je-li

$$x = 4{,}998949 \cdot 10^{1}, \quad \tilde{x} = 4{,}999 \cdot 10^{1}, \quad |\Delta x| = 5{,}10 \cdot 10^{-4}, \quad \left|\frac{\Delta x}{x}\right| \doteq 1{,}020 \cdot 10^{-5},$$

$$y = 5{,}001848 \cdot 10^{1}, \quad \tilde{y} = 5{,}002 \cdot 10^{1}, \quad |\Delta y| = 1{,}52 \cdot 10^{-3}, \quad \left|\frac{\Delta y}{y}\right| \doteq 3{,}039 \cdot 10^{-5},$$

pak pro rozdíly $z = y - x$, $\tilde{z} = \tilde{y} - \tilde{x}$ dostáváme

$$z = 2{,}899 \cdot 10^{-2}, \quad \tilde{z} = 3 \cdot 10^{-2}, \quad |\Delta z| = 1{,}01 \cdot 10^{-3}, \qquad \left|\frac{\Delta z}{z}\right| \doteq 3{,}484 \cdot 10^{-2},$$

takže $\tilde{z}$ má jen jednu platnou cifru, zatímco $\tilde{x}$ i $\tilde{y}$ mají čtyři platné cifry. □

Příklad 1.2. Nechť $x = 1{,}3262 \pm 5 \cdot 10^{-5}$, $y = -6{,}5347 \pm 5 \cdot 10^{-5}$, $z = 13{,}235 \pm 5 \cdot 10^{-4}$. Máme určit aproximaci funkční hodnoty $f = xy/z$, absolutní a relativní chybu a počet platných cifer výsledku.

Spočteme $\tilde{f} = \tilde{x}\tilde{y}/\tilde{z} = -6{,}548031\ldots \cdot 10^{-1}$. Podle (1.1) pak přibližně platí

$$\left|\frac{\Delta f}{\tilde{f}}\right| \le \left[\left|\frac{\tilde{y}}{\tilde{z}}\,\Delta x\right| + \left|\frac{\tilde{x}}{\tilde{z}}\,\Delta y\right| + \left|\frac{\tilde{x}\tilde{y}}{\tilde{z}^2}\,\Delta z\right|\right]\left|\frac{\tilde{x}\tilde{y}}{\tilde{z}}\right|^{-1} = \left|\frac{\Delta x}{\tilde{x}}\right| + \left|\frac{\Delta y}{\tilde{y}}\right| + \left|\frac{\Delta z}{\tilde{z}}\right| \doteq 8{,}31 \cdot 10^{-5}.$$

Odtud $|\Delta f| \doteq 8{,}31 \cdot 10^{-5} \cdot |\tilde{f}| \doteq 5{,}44 \cdot 10^{-5} < 5 \cdot 10^{-1-3}$, takže (se třemi platnými ciframi) $f = -0{,}6548 \pm 0{,}0001$. □

1.2. Reprezentace čísel v počítači

Reálná čísla jsou v počítačích reprezentována v *systému čísel s pohyblivou řádovou čárkou* (v angličtině *floating point numbers*). Základní myšlenka je podobná *semilogaritmickému zápisu* (v angličtině *scientific notation*), v němž např. číslo 245700 píšeme jako $2{,}457 \cdot 10^5$ a číslo 0,0005768 jako $5{,}768 \cdot 10^{-4}$. V tomto formátu se desetinná čárka *pohybuje* (v doslovném překladu plave) v závislosti na dekadickém exponentu. Formálně lze systém $\mathbb{F}$ *normalizovaných čísel pohyblivé řádové čárky* charakterizovat čtyřmi celými čísly:

β	základ číselné soustavy ($\beta \ge 2$),
p	přesnost ($p \ge 1$),
$[L\,, U]$	rozsah exponentu ($L < 0 < U$).

Každé číslo $x \in \mathbb{F}$ má tvar

$$x = \pm m \cdot \beta^e\,, \quad \text{kde} \quad m = d_1 + \frac{d_2}{\beta} + \frac{d_3}{\beta^2} + \cdots + \frac{d_p}{\beta^{p-1}}$$

je *normalizovaná mantisa*, $d_i \in \{0, 1, \ldots, \beta - 1\}$, $i = 1, 2, \ldots, p$, jsou cifry mantisy, p je počet cifer mantisy a $e \in \langle L, U\rangle$ je celočíselný *exponent*. Normalizace mantisy znamená, že pro $x \neq 0$ je prvá cifra mantisy nenulová, tj. platí $d_1 \geq 1$, takže $1 \leq m < \beta$. Když $x = 0$, pak je nulová mantisa i exponent, tj. $m = e = 0$.

Většina počítačů používá *binární aritmetiku*, kdy $\beta = 2$. Pro stručnější zápis binárních čísel se běžně používá *hexadecimální soustava*, v níž $\beta = 16$ (cifry 10 až 15 zapisujeme pomocí písmen A,B,C,D,E,F), a někdy rovněž *oktalová soustava*, kdy $\beta = 8$. Výsledky výpočtů se zpravidla uvádějí v běžné *dekadické soustavě*, tj. pro $\beta = 10$.

Množina $\mathbb{F}$ čísel pohyblivé řádové čárky je konečná, počet čísel v ní je

$$2(\beta - 1)\beta^{p-1}(U - L + 1) + 1\,,$$

neboť můžeme volit dvě znaménka, $\beta - 1$ možností pro první cifru mantisy, β možností pro zbývajících $p - 1$ cifer mantisy a $U - L + 1$ možných hodnot exponentu. Poslední jednička odpovídá číslu nula.

Nejmenší kladné číslo v $\mathbb{F}$ je číslo UFL $= \beta^L$ (podle anglického UnderFlow Level), které má první cifru mantisy rovnu jedné, zbývající cifry mantisy nulové a exponent nejmenší možný. Největší číslo v $\mathbb{F}$ je číslo OFL $= (\beta - \beta^{1-p})\beta^U$ (podle anglického OverFlow Level), které má všechny cifry mantisy rovné $\beta - 1$ a exponent největší možný.

Zaokrouhlování. Reálná čísla, která jsou přesně zobrazitelná v systému $\mathbb{F}$, se nazývají *strojová čísla*. Pokud dané reálné číslo $x \notin \mathbb{F}$, musíme ho aproximovat blízkým strojovým číslem, které značíme $\mathrm{fl}(x)$ (podle anglického floating). Standardní způsob je *zaokrouhlení*: $\mathrm{fl}(x)$ je strojové číslo nejbližší k x (když máme na výběr ze dvou možností, pak vybereme to strojové číslo, které má poslední cifru sudou).

Strojová přesnost. Číslo $\varepsilon_m = \beta^{1-p}$ se nazývá *strojové epsilon* nebo také *strojová přesnost* (anglicky machine epsilon, označované jako ε_{mach}). Význam čísla ε_m dokládá několik jeho charakteristických vlastností:

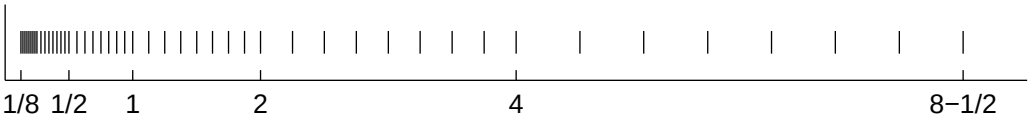
a) v intervalu $\langle \beta^e, \beta^{e+1}\rangle$ jsou strojová čísla rozmístěna rovnoměrně s krokem $\varepsilon_m \beta^e$;

b) největší možná relativní chyba, která vznikne při aproximaci reálného čísla v systému $\mathbb{F}$ pohyblivé řádové čárky, nepřesáhne $\frac{1}{2}\varepsilon_m$, tj. platí $|\mathrm{fl}(x) - x| \leq \frac{1}{2}\varepsilon_m |x|$;

c) ε_m je největší z kladných čísel ε, pro která $\mathrm{fl}(1 + \frac{1}{2}\varepsilon) = 1$.

Je dobré uvědomit si, že $\varepsilon_m \in \mathbb{F}$, jen když $1 - p \geq L$.

Příklad 1.3. Prozkoumejme, jaká čísla můžeme zobrazit v modelovém binárním systému $\mathbb{F}$ v případě, že mantisa má $p = 4$ cifry a exponent e je omezen zdola číslem $L = -3$ a shora číslem $U = 2$, tj. $-3 \leq e \leq 2$. Umístění kladných strojových čísel na číselné ose je patrné z obrázku 1.1: nejmenší z nich UFL $= 2^{-3} = 1/8$ a největší OFL$= (2 - 2^{-3}) \cdot 2^2 = 8 - 1/2$. Všimněte si, že v každém binárním intervalu $2^e \leq x \leq 2^{e+1}$ jsou čísla rozložena rovnoměrně



Obr. 1.1: Strojová čísla

s krokem $\varepsilon_m 2^e$. Tak třeba mezi 1 a 2, tj. pro $e = 0$, je vzdálenost dvou sousedních čísel rovna $\varepsilon_m = 1/8$. Systém $\mathbb{F}$ obsahuje 48 kladných čísel, 48 záporných čísel a nulu, tj. celkem 97 čísel. □

Standard IEEE. V počítačích vyvinutých po roce 1985 se reálná čísla zobrazují prakticky výhradně podle standardu IEEE, a to zpravidla v těchto přesnostech:

a) *Jednoduchá přesnost.* Použijí se 4 bajty, tj. 32 bitů, z toho 23 bitů pro mantisu, 8 bitů pro exponent a 1 bit pro znaménko mantisy. Protože mantisa je normalizovaná, pro $x \neq 0$ je $d_1 = 1$. Tato cifra se neukládá, takže počet cifer mantisy $p = 24$. Rozsah exponentu je $-126 \leq e \leq 127$. Zobrazit lze dekadická čísla s absolutní hodnotou v rozsahu

$$\mathrm{UFL} = 2^{-126} \doteq 1{,}2 \times 10^{-38} \quad \text{až} \quad \mathrm{OFL} = (2 - 2^{-23}) \times 2^{127} \doteq 3{,}4 \times 10^{38}$$

a nulu (má všechny bity nulové). Strojová přesnost $\varepsilon_m = 2^{-23} \doteq 1{,}2 \times 10^{-7}$. Říkáme, že *mantisa má zhruba 7 dekadických cifer přesnosti*.

b) *Dvojnásobná přesnost.* Použije se 8 bajtů, tj. 64 bitů, z toho 52 bitů pro mantisu, 11 bitů pro exponent a 1 bit pro znaménko mantisy. První bit mantisy se neukládá (pro $x \neq 0$ je $d_1 = 1$), takže mantisa má $p = 53$ cifer. Rozsah exponentu je $-1022 \leq e \leq 1023$. Zobrazit lze dekadická čísla s absolutní hodnotou v rozsahu

$$\mathrm{UFL} = 2^{-1022} \doteq 2{,}2 \times 10^{-308} \text{ až } \mathrm{OFL} = (2 - 2^{-52}) \times 2^{1023} \doteq 1{,}8 \times 10^{308}$$

a nulu (má všechny bity nulové). Strojová přesnost $\varepsilon_m = 2^{-52} \doteq 2{,}2 \times 10^{-16}$. Říkáme, že *mantisa má zhruba 16 dekadických cifer přesnosti*.

Podle IEEE standardu existuje také binární reprezentace INF pro výrazy typu $+\infty$ (třeba výsledek operace $1/0$), $-$INF pro výrazy typu $-\infty$ (třeba výsledek operace $-1/0$) a NAN (not a number) pro výrazy typu $0/0$, $\infty - \infty$ a $0 \times (\pm\infty)$ (třeba výsledek operace $1/0 - 2/0$). Systém $\mathbb{F}$ je na většině počítačů rozšířen o tzv. *subnormální čísla*, což jsou nenulová nenormalizovaná čísla s nejmenším možným exponentem $e = L$. Nejmenší kladné subnormální číslo $\mathrm{UFL}_s = \varepsilon_m \cdot \mathrm{UFL}$, tj. v jednoduché přesnosti $\mathrm{UFL}_s \doteq 1,4 \cdot 10^{-45}$ a ve dvojnásobné přesnosti $\mathrm{UFL}_s \doteq 4,9 \cdot 10^{-324}$.

Počítačová aritmetika. Nechť $x, y \in \mathbb{F}$ jsou strojová čísla, `op` je některá ze základních aritmetických operací $+,-,\times,/$ a `flop` je odpovídající operace prováděná počítačem v režimu pohyblivé řádové čárky podle IEEE standardu. Pak x `flop` $y = \mathrm{fl}(x$ `op` $y)$. To znamená, že výsledek aritmetické operace provedené v počítači je stejný, jako když operaci provedeme přesně a pak získaný výsledek vložíme do počítače.

Přetečení, podtečení. Jestliže je absolutní hodnota výsledku aritmetické operace větší než OFL, dochází k tzv. *přetečení*, a je-li naopak absolutní hodnota nenulového výsledku menší než UFL (resp. UFL_s na počítačích se subnormálními čísly), dochází k tzv. *podtečení*. Dojde-li při běhu programu k přetečení, systém vydá varování a výpočet přeruší. V případě podtečení situace není tak vážná: výsledek se nahradí nulou a výpočet pokračuje bez přerušení. Reakci programu na přetečení však může poučený programátor sám řídit.

1.3. Podmíněnost úloh a algoritmů

Korektní úlohy. Matematickou úlohu lze chápat jako zobrazení $y = f(x)$, které ke každému vstupnímu údaji x z množiny D vstupních dat přiřadí výsledek y z množiny R výstupních dat. Řekneme, že matematická úloha

$$y = f(x)\,, \qquad x \in D\,, \quad y \in R\,,$$

je *korektní*, když

1) ke každému vstupu $x \in D$ existuje jediné řešení $y \in R$,

2) toto řešení závisí spojitě na vstupních datech, tj. když $x \to a$, potom $f(x) \to f(a)$.

Velkou třídu nekorektních úloh tvoří nejednoznačně řešitelné úlohy. Nekorektní úlohy se nedají rozumně řešit a proto se jimi nebudeme vůbec zabývat.

Podmíněnost úloh. Budeme říkat, že korektní úloha je *dobře podmíněná*, jestliže malá změna ve vstupních datech vyvolá malou změnu řešení. Je-li $y + \Delta y$ resp. y řešení úlohy odpovídající vstupním datům $x + \Delta x$ resp. x, potom číslo

$$C_p = \frac{|\Delta y|/|y|}{|\Delta x|/|x|} = \frac{|\text{relativní chyba na výstupu}|}{|\text{relativní chyba na vstupu}|} \tag{1.8}$$

(kde místo absolutních hodnot jsou obecně normy, viz kap. 2) nazýváme *číslo podmíněnosti* úlohy $y = f(x)$. Je-li C_p malé číslo, je úloha dobře podmíněná, pro velká C_p je úloha špatně podmíněná. Místo o dobré nebo špatné podmíněnosti mluvíme někdy o malé nebo velké *citlivosti vzhledem ke vstupním datům*.

Příklad 1.4. Odhadněte číslo podmíněnosti úlohy: stanovit funkční hodnotu (diferencovatelné) funkce $y = f(x)$. Z (1.2) plyne, že

$$C_p \doteq \left| \frac{x f'(x)}{f(x)} \right| . \tag{1.9}$$

Konkrétně pro funkci $f(x) = \operatorname{tg} x$ dostaneme $C_p \doteq |2x/\sin 2x|$. Výpočet $\operatorname{tg} x$ je velmi citlivý pro x blízké celočíselnému násobku $\pi/2$. Třeba pro $x = 1{,}57079$ je $C_p \doteq 2{,}48 \cdot 10^5$. Výsledek můžeme ověřit také přímým výpočtem podle vzorce (1.8), pro $\Delta x = 10^{-9}$ dostaneme opět $C_p \doteq 2{,}48 \cdot 10^5$. □

Číslo podmíněnosti definované podle (1.8) se někdy označuje jako *relativní číslo podmíněnosti*. Většinou je to vhodné měřítko citlivosti, je-li však x nebo y rovno nule, použít

ho nelze. V takových případech lze zkusit *absolutní číslo podmíněnosti* definované jako $\bar{C}_p = |\Delta y| \,/|\Delta x|$.

Příklad 1.5. Posoudíme citlivost výpočtu funkční hodnoty $f(x) = x^2 - 1$. Pro kořeny $x_{1,2} = \pm 1$ není relativní číslo podmíněnosti definováno. Stejný závěr potvrzuje vyjádření $C_p \doteq |2x^2/(x^2-1)|$ odvozené podle (1.9). Absolutní číslo podmíněnosti je

$$\bar{C}_p = \left| \frac{f(x+\Delta x) - f(x)}{\Delta x} \right| \doteq |f'(x)| \,,$$

tj. pro $f(x) = x^2 - 1$ je $\bar{C}_p \doteq |2x|$ a speciálně pro $x = \pm 1$ dostaneme $\bar{C}_p \doteq 2$. □

Stabilita algoritmu. Při realizaci numerické metody na počítači vznikají zaokrouhlovací chyby, nejdříve ve vstupních datech a pak v průběhu výpočtu při provádění aritmetických operací. Chceme-li se při výpočtech vyvarovat nesmyslných výsledků, musíme si vybírat tzv. *stabilní algoritmy*, které jsou k šíření zaokrouhlovacích chyb málo citlivé. Aby byl algoritmus stabilní, musí být

1) *dobře podmíněný*, tj. málo citlivý na poruchy ve vstupních datech,

2) *numericky stabilní*, tj. málo citlivý na vliv zaokrouhlovacích chyb vznikajících během výpočtu.

Příklad 1.6. Kvadratická rovnice $x^2 - 2bx + c = 0$ má pro $b^2 > c$ dva různé reálné kořeny

$$x_{1,2} = b \pm d \,, \qquad (A_1)$$

kde $d = \sqrt{b^2 - c}$. Jestliže $|b| \doteq |d|$, pak se při výpočtu jednoho z kořenů budou odečítat dvě přibližně stejně velká čísla téhož znaménka, což vede, jak už víme, ke vzniku velké relativní chyby (viz (1.3) a příklad 1.1). Výpočet podle algoritmu (A_1) tedy obecně stabilní není. Existuje však snadná pomoc: protože $x_1 x_2 = c$, můžeme postupovat takto:

$$x_1 = \begin{cases} b + d \,, & \text{je-li } b \geq 0, \\ b - d \,, & \text{je-li } b < 0, \end{cases} \qquad x_2 = c/x_1. \qquad (A_2)$$

Algoritmus (A_2) nedostatek algoritmu (A_1) odstraňuje, tj. je stabilní. □

Příklad 1.7. Počítejme integrál

$$E_n = \int_0^1 x^n e^{x-1} \,\mathrm{d}x \qquad \text{pro } n = 1, 2, \ldots$$

Integrací per-partes dostaneme

$$\int_0^1 x^n e^{x-1} \,\mathrm{d}x = [x^n e^{x-1}]_0^1 - \int_0^1 n x^{n-1} e^{x-1} \,\mathrm{d}x \,,$$

nebo-li

$$E_n = 1 - n E_{n-1} \,. \qquad (\mathrm{F})$$

Protože $E_1 = 1/e$, můžeme při výpočtu E_n postupovat podle algoritmu

$$E_1 = 1/e\,, \qquad E_n = 1 - nE_{n-1}\,, \qquad n = 2, 3, \ldots \tag{A_1}$$

Výpočet jsme provedli na počítači v jednoduché přesnosti (tj. cca na 7 platných cifer), a pro $n = 12$ jsme dostali $E_{12} \doteq -4{,}31$. To je ale zcela nepřijatelný výsledek, neboť pro kladný integrand nemůžeme dostat zápornou hodnotu integrálu! Tento jev je způsoben tím, že při výpočtu E_n se chyba obsažená v E_{n-1} násobí $n-$krát, takže celková chyba roste podobně jako $n!$.

Algoritmus(A_1) je tedy nestabilní. Vzniká otázka, zda můžeme vypočítat E_{12} užitím rekurentní formule (F) tak, aby výsledek měl všech 7 cifer platných. Možné to je, musíme ale použít jiný algoritmus. Přepíšeme-li formuli (F) na tvar

$$E_{n-1} = \frac{1 - E_n}{n}\,,$$

bude se chyba vstupující do každého kroku dělit n. Z odhadu

$$E_n = \int_0^1 x^n e^{x-1}\,\mathrm{d}x \le \int_0^1 x^n\,\mathrm{d}x = \frac{1}{n+1}$$

vyplývá, že $E_n \to 0$ pro $n \to \infty$. Při výpočtu proto budeme postupovat podle algoritmu

$$E_N = 0\,, \qquad E_{n-1} = \frac{1 - E_n}{n}\,, \qquad n = N, N-1, \ldots\,, \tag{A_2}$$

kde N je dostatečně velké. Zvolíme-li $N = 20$, dostaneme $E_{12} \doteq 7{,}177325 \cdot 10^{-2}$, což je hodnota, která má 7 cifer platných.

Jestliže výpočet provádíme ve dvojnásobné přesnosti (cca 16 platných cifer), dostaneme obdobné výsledky. Rozdíl je pouze v tom, že výpočet algoritmem (A_1) se pokazí až pro větší n; pro $n = 20$ už ale vyjde nesmyslná hodnota $E_{20} \doteq -30{,}19$. Stabilním algoritmem (A_2) pro $N = 35$ dostaneme $E_{20} \doteq 4{,}554488407581805 \cdot 10^{-2}$ s 16-ti platnými ciframi. □

Další příklady věnované podmíněnosti problémů a zejména algoritmů uvedeme postupně v následujících kapitolách.

1.4. Cvičení

1.1. Řešte kvadratickou rovnici

$$\begin{array}{lrcl} a) & x^2 - 10{,}1\,x + 1 & = & 0, \\ b) & x^2 - 1\,000\,000{,}000\,001\,x + 1 & = & 0, \\ c) & x^2 - 1\,000\,000\,000{,}000\,000\,001\,x + 1 & = & 0, \end{array}$$

numericky stabilním algoritmem. Porovnejte s výpočtem pomocí klasického vzorce.

[Stabilní algoritmus dává přesné výsledky. Absolutní hodnota chyby klasického algoritmu roste: a) $|E| \doteq 3{,}6 \cdot 10^{-16}$, b) $|E| \doteq 7{,}6 \cdot 10^{-12}$, c) $|E| \doteq 10^{-9}$.]

1.2. Vysvětlete, proč se výrazy $f_1(x) = \sqrt{x}(\sqrt{x+1} - \sqrt{x})$ a $f_2(x) = \dfrac{\sqrt{x}}{\sqrt{x+1} + \sqrt{x}}$ chovají pro velká x odlišně, přestože matematicky je $f_1 = f_2$. Který z nich je numericky stabilní?

[f_2 je stabilní, ve vzorci f_1 dochází pro velká x k odečítání blízkých čísel a ke ztrátě platných cifer.]

1.3. Naprogramujte zobrazení binární reprezentace reálných čísel standardu IEEE pomocí řetězce znaků '0' a '1'. Oddělte bity znaménka, exponentu a mantisy. Jakým způsobem se zobrazí čísla $\frac{1}{3},\ \pi,\ -70{,}07$?

[Zobrazení je ve tvaru | znaménko | exponent | mantisa |.

Jednoduchá přesnost:

$\frac{1}{3} =$ | 0 | 01111101 | 01010101010101010101 |

$\pi =$ | 0 | 10000000 | 10010010000111111011011 |

$-70{,}07 =$ | 1 | 10000101 | 00011000010001111010111 |

Dvojnásobná přesnost:

$\frac{1}{3} =$ | 0 | 01111111101 | 0101010101010101010101010101010101010101010101010101 |

$\pi =$ | 0 | 10000000000 | 1001001000011111101101010001000100001011010001100 0 |

$-70{,}07 =$ | 1 | 10000000101 | 0001100001000111101011100001010001111010111000010100]

1.4. Převeďte následující čísla standardu IEEE zobrazená pomocí '0' a '1' na semilogaritmický tvar:

(a) jednoduchá přesnost: znaménko | 0 | exponent | 10011101 | mantisa | 00100110010110000000110 |

(b) jednoduchá přesnost: znaménko | 1 | exponent | 01110101 | mantisa | 00100110001101101001011 |

(c) dvojnásobná přesnost: znaménko | 1 | exponent | 01111101010 |

mantisa | 1010000110010000111100111111111111100101101001110010 0 |

[(a) $\underbrace{+}_{\text{znaménko}}\ \underbrace{(1 + 0/2 + 0/2^2 + 1/2^3 + 0/2^4 + \ldots)}_{\text{bity mantisy}} \cdot 2^e \doteq 1{,}149780988693237 \cdot 2^{30} \doteq 1{,}234568 \cdot 10^9,$

kde $e = \underbrace{1 + 0 \cdot 2 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 0 \cdot 2^6 + 1 \cdot 2^7}_{\text{bity exponentu (zprava)}} - \underbrace{127}_{U} = 30\,;$

(b) $\underbrace{-}_{\text{znaménko}}\ \underbrace{(1 + 0/2 + 0/2^2 + 1/2^3 + 0/2^4 + \ldots)}_{\text{bity mantisy}} \cdot 2^e \doteq -1{,}149270415306091 \cdot 2^{-10} \doteq -0{,}001122334,$

kde $e = \underbrace{1 + 0 \cdot 2 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 + 1 \cdot 2^7}_{\text{bity exponentu (zprava)}} - \underbrace{127}_{U} = -10\,;$

(c) postupujeme analogicky jako pro jednoduchou přesnost, nyní však $U = 1023$; tak dostaneme číslo $-1{,}6311180591104 \cdot 2^{-21} = -7{,}777777 \cdot 10^{-7}$.]

1.5. Ověřte na počítači, že výraz `abs(3*(4/3-1)-1)` se vyčíslí jako strojové epsilon.

1.6. Určete maximální chybu pro $y = x_1 x_2^2/\sqrt{x_3}$, kde $x_1 = 2 \pm 0{,}1$, $x_2 = 3 \pm 0{,}2$, $x_3 = 1 \pm 0{,}1$.

[$y = 18 \pm 4{,}2$, tj. $|\Delta y| \lesssim 4{,}2$ a $|\Delta y/y| \lesssim 0, 2\overline{33}$, což je přibližně 23%.]

1.7. Uvažujme tři rekurentní formule

$$(a)\ p_1 = 1,\ p_2 = \frac{1}{2} - \frac{\varepsilon}{2},\ p_{i+1} = \frac{1}{2} p_i$$

$$(b)\ q_1 = 1,\ q_2 = \frac{1}{2} - \frac{\varepsilon}{2},\ q_{i+1} = \frac{3}{2} q_i - \frac{1}{2} q_{i-1} \qquad \text{pro } i = 2, 3, \ldots .$$

$$(c)\ r_1 = 1,\ r_2 = \frac{1}{2} - \frac{\varepsilon}{2},\ r_{i+1} = \frac{7}{2} r_i - \frac{3}{2} r_{i-1}$$

Ověřte, že pro $\varepsilon = 0$ všechny tři formule generují členy posloupnosti $a_i = 1/2^i$. Napište program a počítejte a_i, p_i, q_i, r_i pro několik malých hodnot ε (např. $\varepsilon = 10^{-6},\ i = 1, 2, \ldots, 50$). Co pozorujete?

[$|p_i - 1/2^i| \to 0$, $|q_i - 1/2^i| \to \varepsilon$, $|r_i - 1/2^i| \to \infty$.]

2. Řešení soustav lineárních rovnic

Jednou z nejčastěji se vyskytujících úloh výpočetní praxe je úloha vyřešit soustavu lineárních rovnic. Takové soustavy bývají často velmi rozsáhlé, současná výpočetní technika umožňuje v přijatelných časech vyřešit soustavy s několika milióny neznámých. Metody řešení dělíme na přímé a iterační. *Přímé metody* jsou takové metody, které dodají v konečném počtu kroků přesné řešení za předpokladu, že výpočet probíhá bez zaokrouhlovacích chyb, tedy zcela přesně. *Iterační metody* poskytnou jen řešení přibližné. To ale vůbec nevadí, pokud je přibližné řešení dostatečně dobrou aproximací řešení přesného. Počet kroků iterační metody závisí na požadované přesnosti.

Budeme se tedy zabývat řešením soustavy lineárních rovnic

$$\begin{array}{ccccccccc} a_{11}\,x_1 & + & a_{12}\,x_2 & + & \cdots & + & a_{1n}\,x_n & = & b_1\,, \\ a_{21}\,x_1 & + & a_{22}\,x_2 & + & \cdots & + & a_{2n}\,x_n & = & b_2\,, \\ \vdots & & & & & & & \vdots & \\ a_{n1}\,x_1 & + & a_{n2}\,x_2 & + & \cdots & + & a_{nn}\,x_n & = & b_n\,. \end{array} \tag{2.1}$$

Soustavu (2.1) můžeme psát ve tvaru

$$\sum_{j=1}^{n} a_{ij} x_j = b_i\,, \qquad i = 1, 2, \ldots, n, \tag{2.2}$$

nebo v maticovém tvaru

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,, \tag{2.3}$$

kde

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Matici $\mathbf{A}$ nazýváme *maticí soustavy*, $\mathbf{b}$ je *vektor pravé strany* a $\mathbf{x}$ *vektor neznámých*.

Budeme předpokládat, že matice soustavy je regulární, takže řešená soustava má jediné řešení.

2.1. Přímé metody

2.1.1. Gaussova eliminační metoda

Základní přímou metodou řešení soustav lineárních rovnic je *Gaussova eliminační metoda*, stručně GEM. Skládá se ze dvou částí. V *přímém chodu* GEM se soustava (2.1) převede na ekvivalentní soustavu

$$\mathbf{U}\mathbf{x} = \mathbf{c}\,, \tag{2.4}$$

kde $\mathbf{U}$ je tzv. *horní trojúhelníková matice*, což je matice, která má pod hlavní diagonálou všechny prvky nulové, tj. $\mathbf{U} = \{u_{ij}\}_{i,j=1}^{n}$ a $u_{ij} = 0$ pro $i > j$,

$$\mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1,n-1} & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2,n-1} & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3,n-1} & u_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & u_{n-1,n-1} & u_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 & u_{nn} \end{pmatrix}.$$

Ve *zpětném chodu* se pak řeší soustava (2.4). Protože $\mathbf{A}$ je regulární, je také $\mathbf{U}$ regulární, což znamená, že diagonální prvky $u_{ii} \neq 0$, $i = 1, 2, \ldots, n$. Díky tomu vypočteme z poslední rovnice x_n, z předposlední x_{n-1} atd. až nakonec z první rovnice vypočteme x_1.

Přímý chod GEM. Pro usnadnění popisu přímého chodu GEM položíme $\mathbf{A}^{(0)} = \mathbf{A}$, $\mathbf{b}^{(0)} = \mathbf{b}$, prvky matice matice $\mathbf{A}^{(0)}$ označíme $a_{ij}^{(0)} \equiv a_{ij}$ a prvky vektoru $\mathbf{b}^{(0)}$ označíme $b_i^{(0)} \equiv b_i$. Přímý chod GEM popisuje následující

algoritmus GEMz (základní, bez výběru hlavního prvku):

for $k := 1$ **to** $n - 1$ **do**
begin
  $\mathbf{A}^{(k)} := \mathbf{A}^{(k-1)}$; $\mathbf{b}^{(k)} := \mathbf{b}^{(k-1)}$;
  for $i := k + 1$ **to** n **do**
  begin
    $m_{ik} := a_{ik}^{(k)} / a_{kk}^{(k)}$;
    for $j := k + 1$ **to** n **do** $a_{ij}^{(k)} := a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$;
    $b_i^{(k)} := b_i^{(k)} - m_{ik} b_k^{(k)}$;
  end
end

Přímý chod má $n-1$ kroků. V k-tém kroku se soustava rovnic $\mathbf{A}^{(k-1)}\mathbf{x} = \mathbf{b}^{(k-1)}$ transformuje na soustavu $\mathbf{A}^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$. Prvních k rovnic se už nemění. Tato skutečnost je v algoritmu GEMz vyjádřena příkazy $\mathbf{A}^{(k)} := \mathbf{A}^{(k-1)}$ a $\mathbf{b}^{(k)} := \mathbf{b}^{(k-1)}$. Smyslem transformace je vyloučit neznámou x_k z rovnic $i > k$, tj. vynulovat poddiagonální koeficienty v k-tém sloupci matice $\mathbf{A}^{(k)}$. Dosáhneme toho tak, že od i-té rovnice odečteme m_{ik} násobek k-té rovnice. *Multiplikátory* m_{ik} musejí zajistit, aby v pozici (i, k) matice $\mathbf{A}^{(k)}$ vznikla nula:

$$a_{ik}^{(k)} - m_{ik} a_{kk}^{(k)} = 0 \qquad \Longrightarrow \qquad m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}.$$

Číslo $a_{kk}^{(k)}$ se nazývá *hlavní prvek* nebo také *pivot*. Při výpočtu multiplikátoru m_{ik} může algoritmus GEMz zhavarovat v případě, že $a_{kk}^{(k)} = 0$. Tomuto problému bychom se mohli vyhnout, kdybychom k-tou rovnici prohodili s některou z dalších rovnic, která má u proměnné x_k nenulový koeficient. Postup založený na této myšlence se nazývá GEM s výběrem hlavního prvku. Podrobně se jím budeme zabývat v následujícím odstavci. GEMz je tedy algoritmus *GEM bez výběru hlavního prvku*.

V tomto odstavci budeme předpokládat, že $\mathbf{A}$ je taková matice soustavy, pro kterou jsou všechny hlavní prvky $a_{kk}^{(k)}$ nenulové.

Programování. Prvky matic $A^{(k)}$ uchováváme v dvourozměrném poli `A` a prvky vektorů $\mathbf{b}^{(k)}$ v jednorozměrném poli `b`. Příkazy $\mathbf{A}^{(k)} := \mathbf{A}^{(k-1)}$ a $\mathbf{b}^{(k)} := \mathbf{b}^{(k-1)}$ se proto ve skutečnosti neprovádějí. Protože v pozici (i,k) pole `A` vznikne nula, lze prvek `A(i,k)` využít pro uskladnění multiplikátoru m_{ik}.

LU rozklad. Po ukončení přímého chodu je horní trojúhelníková matice $\mathbf{U}$ v rovnici (2.4) určena diagonálními a naddiagonálními prvky matice $\mathbf{A}^{(n-1)}$, tj.

$$u_{ij} := \begin{cases} 0 & \text{pro } j = 1, 2, \ldots, i-1\,, \\ a_{ij}^{(n-1)} & \text{pro } j = i, i+1, \ldots, n\,, \end{cases} \qquad i = 1, 2, \ldots, n\,. \tag{2.5}$$

Vektor $\mathbf{c}$ v (2.4) je transformovanou pravou stranou $\mathbf{b}^{(n-1)}$, tj.

$$c_i := b_i^{(n-1)}\,, \qquad i = 1, 2, \ldots, n. \tag{2.6}$$

Multiplikátory m_{ij} z přímého chodu umístíme do dolní trojúhelníkové matice $\mathbf{L} = \{\ell_{ij}\}_{i,j=1}^{n}$, $\ell_{ij} = 0$ pro $j > i$,

$$\mathbf{L} = \begin{pmatrix} \ell_{11} & 0 & 0 & \cdots & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ell_{n-1,1} & \ell_{n-1,2} & \ell_{n-1,3} & \cdots & \ell_{n-1,n-1} & 0 \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & \ell_{n,n-1} & \ell_{nn} \end{pmatrix},$$

definované předpisem

$$\ell_{ik} := \begin{cases} 0 & \text{pro } i = 1, 2, \ldots, k-1\,, \\ 1 & \text{pro } i = k\,, \\ m_{ik} & \text{pro } i = k+1, k+2, \ldots, n\,, \end{cases} \qquad k = 1, 2, \ldots, n\,. \tag{2.7}$$

Dá se ukázat, že platí

$$\mathbf{A} = \mathbf{L}\mathbf{U}\,. \tag{2.8}$$

Vyjádření matice $\mathbf{A}$ jako součinu dolní trojúhelníkové matice $\mathbf{L}$ a horní trojúhelníkové matice $\mathbf{U}$ se nazývá *LU rozklad matice* $\mathbf{A}$. Ten je možné použít k pozdějšímu řešení soustavy rovnic se stejnou maticí soustavy $\mathbf{A}$, avšak s jinou pravou stranou. To je užitečné zejména při řešení posloupnosti úloh $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$, kdy se nová pravá strana $\mathbf{b}_i$ může sestavit až poté, co se vyřešily předchozí soustavy $\mathbf{A}\mathbf{x}_k = \mathbf{b}_k$ pro $k < i$.

Ukažme si, jak lze soustavu $\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}$ efektivně vyřešit. Když si označíme $\mathbf{U}\mathbf{x} = \mathbf{y}$, vidíme, že $\mathbf{y}$ je řešení soustavy $\mathbf{L}\mathbf{y} = \mathbf{b}$. Určíme tedy nejdříve $\mathbf{y}$ jako řešení soustavy $\mathbf{L}\mathbf{y} = \mathbf{b}$ a pak $\mathbf{x}$ jako řešení soustavy $\mathbf{U}\mathbf{x} = \mathbf{y}$, tj.

$$\mathbf{L}\mathbf{y} = \mathbf{b}\,, \qquad \mathbf{U}\mathbf{x} = \mathbf{y}\,. \tag{2.9}$$

Zřejmě $\mathbf{y} = \mathbf{b}^{(n-1)}$ je transformovaná pravá strana získaná algoritmem GEMz.

Soustavu $\mathbf{L}\mathbf{y} = \mathbf{b}$ vyřešíme snadno, z první rovnice vypočítáme y_1, ze druhé rovnice y_2 atd. až nakonec z poslední rovnice vypočítáme y_n. Soustavu $\mathbf{U}\mathbf{x} = \mathbf{y}$ řešíme pozpátku, tj. z poslední rovnice vypočteme x_n, z předposlední x_{n-1} atd. až nakonec z první rovnice vypočteme x_1.

Při řešení soustav rovnic bývá *LU* rozklad označován také jako eliminace nebo přímý chod a výpočet řešení podle (2.9) bývá označován jako zpětný chod.

Kdy lze algoritmus GEMz použít? Jak jsme již uvedli, slabým místem algoritmu GEMz může být výpočet multiplikátoru m_{ik}, neboť obecně nelze vyloučit, že v průběhu eliminace vznikne $a_{kk}^{(k)} = 0$. V aplikacích se však poměrně často řeší soustavy rovnic, pro které nulový pivot v algoritmu GEMz vzniknout nemůže. Abychom takové soustavy mohli popsat, zavedeme si několik nových pojmů.

Řekneme, že matice $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{n}$ je *ryze diagonálně dominantní*, jestliže

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|\,, \qquad i = 1, 2, \ldots, n\,, \tag{2.10}$$

nebo-li slovy, v každém řádku je absolutní hodnota diagonálního prvku větší než součet absolutních hodnot zbývajících prvků tohoto řádku. I když matice $\mathbf{A}$ soustavy rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$ diagonálně dominantní není, lze někdy vhodným přeskládáním rovnic docílit toho, že matice $\hat{\mathbf{A}}$ takto vzniklé ekvivalentní soustavy rovnic $\hat{\mathbf{A}}\mathbf{x} = \hat{\mathbf{b}}$ už diagonálně dominantní je.

V aplikacích se také poměrně často setkáváme s tzv. pozitivně definitními maticemi. Takové matice lze specifikovat pomocí řady navzájem ekvivalentních definic. Jednu z nich si teď uvedeme: řekneme, že matice $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{n}$ je *pozitivně definitní*, jestliže je symetrická a pro každý nenulový sloupcový vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ platí

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \sum_{i,j=1}^{n} x_i a_{ij} x_j > 0\,. \tag{2.11}$$

Ověřit přímo tuto podmínku není snadné. Je-li však $\mathbf{A}$ regulární, pak z (2.11) okamžitě plyne, že $\mathbf{A}^T\mathbf{A}$ je pozitivně definitní. (Dokažte to!) Vynásobíme-li tedy soustavu rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$ zleva maticí $\mathbf{A}^T$, dostaneme ekvivalentní soustavu $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ s pozitivně definitní maticí soustavy. Tento postup se však pro praktické řešení soustav rovnic nehodí (operace $\mathbf{A}^T\mathbf{A}$ vyžaduje velký objem výpočtů, u iteračních metod se navíc významně zhoršuje rychlost konvergence).

Při řešení konkrétních praktických úloh bývá obvykle už předem známo (z povahy řešeného problému a ze způsobu jeho diskretizace), zda matice vznikajících soustav lineárních rovnic jsou (resp. nejsou) pozitivně definitní. Uveďme si však přesto alespoň jednu často uváděnou (nutnou a postačující) podmínku pozitivní definitnosti, známou jako

Sylvesterovo kritérium. Čtvercová symetrická matice $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{n}$ je pozitivně definitní, právě když jsou kladné determinanty všech hlavních rohových submatic $\{a_{ij}\}_{i,j=1}^{k}$,

$k = 1, 2, \ldots, n$, tj. když platí

$$a_{11} > 0\,, \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} > 0\,, \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} > 0\,, \ \ldots\ , \ \begin{vmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \ldots & a_{nn} \end{vmatrix} > 0\,. \qquad \square$$

Dá se ukázat, že *algoritmus GEMz lze použít pro řešení soustav, jejichž matice je buďto ryze diagonálně dominantní nebo pozitivně definitní.* Úspěšné použití algoritmu GEMz lze zaručit také pro další typy matic, které se při řešení praktických úloh často vyskytují (viz např. [4], [13]).

Výpočtová náročnost GEM. Přímý chod GEM vyžaduje $\frac{1}{3}n^3 + O(n^2)$ operací násobicích (tj. násobení nebo dělení) a $\frac{1}{3}n^3 + O(n^2)$ operací sčítacích (tj. sčítání nebo odečítání). Symbolem $O(n^2)$ jsme přitom vyjádřili řádově méně významný počet operací řádu n^2 (tvaru $\alpha_2 n^2 + \alpha_1 n + \alpha_0$, kde α_2, α_1, α_0 jsou čísla nezávislá na n). Člen $\frac{1}{3}n^3$ souvisí s transformací matice soustavy. Počet operací souvisejících s transformací pravé strany je o řád nižší a je tedy zahrnut do členu $O(n^2)$.

Zpětný chod GEM je výpočetně podstatně méně náročný. Řešení soustavy rovnic s trojúhelníkovou maticí vyžaduje $\frac{1}{2}n^2 + O(n)$ operací násobicích a $\frac{1}{2}n^2 + O(n)$ operací sčítacích. Přitom $O(n)$ reprezentuje počet operací řádu n (tvaru $\alpha_1 n + \alpha_0$, kde α_1, α_0 jsou čísla nezávislá na n). GEM zpětný chod, tj. výpočet $\mathbf{x}$ ze soustavy (2.4), proto vyžaduje $\frac{1}{2}n^2 + O(n)$ operací a LU zpětný chod, tj. výpočet $\mathbf{y}$ a $\mathbf{x}$ ze soustav (2.9), vyžaduje dvojnásobný počet operací, tj. $n^2 + O(n)$.

Pro velký počet rovnic, tj. pro velké n, proto můžeme tvrdit, že eliminace vyžaduje přibližně $\frac{1}{3}n^3$ operací a GEM resp. LU zpětný chod přibližně $\frac{1}{2}n^2$ resp. n^2 operací (násobicích a stejně tak sčítacích).

Choleského rozklad. Pozitivně definitní matici $\mathbf{A}$ lze vyjádřit ve tvaru

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T, \tag{2.12}$$

kde $\mathbf{L}$ je dolní trojúhelníková matice, jejíž nenulové prvky jsou postupně pro $k = 1, 2, \ldots, n$ určeny předpisem

$$\ell_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2}\,, \tag{2.13}$$

$$\ell_{ik} = \frac{1}{\ell_{kk}} \left(a_{ik} - \sum_{j=1}^{k-1} \ell_{ij}\ell_{kj} \right), \qquad i = k+1, k+2, \ldots, n\,.$$

Soustavu rovnic řešíme podle (2.9) pro $\mathbf{U} = \mathbf{L}^T$. Vyjádření matice $\mathbf{A}$ ve tvaru (2.12) se nazývá *Choleského rozklad* matice $\mathbf{A}$. Choleského rozklad vyžaduje přibližně poloviční výpočtové náklady oproti obecnému *LU* rozkladu, tedy přibližně $\frac{1}{6}n^3$ operací násobicích a zhruba stejný počet operací sčítacích (výpočet odmocnin nemá na celkový počet operací podstatný vliv). Choleského algoritmus (2.13) lze použít k efektivnímu posouzení pozitivní definitnosti matice $\mathbf{A}$: je-li $\mathbf{A}$ symetrická a platí-li $a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2 > 0$ pro $k = 1, 2, \ldots, n$, pak $\mathbf{A}$ je pozitivně definitní. V MATLABu lze pro Choleského rozklad použít funkci `chol`.

2.1.2. Výběr hlavního prvku

Začneme příkladem.

Příklad 2.1. Máme vyřešit soustavu rovnic

$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2{,}099 & 6 \\ 5 & -1{,}1 & 4{,}8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 3{,}901 \\ 5{,}9 \end{pmatrix}$$

na hypotetickém počítači, který pracuje v dekadické soustavě s pětimístnou mantisou. Přesné řešení je

$$\mathbf{x} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}.$$

V prvním kroku eliminujeme poddiagonální prvky v prvním sloupci a dostaneme

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0{,}001 & 6 \\ 0 & 2{,}4 & 4{,}8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6{,}001 \\ 2{,}4 \end{pmatrix}.$$

Prvek v pozici $(2, 2)$ je ve srovnání s ostatními prvky matice malý. Přesto pokračujme v eliminaci. V dalším kroku je třeba ke třetímu řádku přičíst řádek druhý násobený 2400:

$$(4{,}8 + 6 \cdot 2400) \cdot x_3 = 2{,}4 + 6{,}001 \cdot 2400\,.$$

Na levé straně je koeficient $4{,}8 + 6 \cdot 2400 = 14404{,}8$ zaokrouhlen na 14405. Na pravé straně výsledek násobení $6{,}001 \cdot 2400 = 14402{,}4$ nelze zobrazit přesně, musí být zaokrouhlen na 14402. K tomu se pak přičte 2,4 a znovu dojde k zaokrouhlení. Poslední rovnice tak nabude tvaru

$$14405\, x_3 = 14404\,.$$

Zpětný chod začne výpočtem

$$x_3 = \frac{14404}{14\,405} \doteq 0{,}99993\,.$$

Přesný výsledek je $x_3 = 1$. Zdá se, že chyba není nijak vážná. Bohužel, x_2 je třeba určit z rovnice

$$-0{,}001\, x_2 + 6 \cdot 0{,}99993 = 6{,}001,$$

což dává, po zaokrouhlení $6 \cdot 0{,}99993 \doteq 5{,}9996$,

$$x_2 = \frac{0{,}0014}{-0{,}001} = -1{,}4.$$

Nakonec vypočteme x_1 z první rovnice

$$10x_1 - 7 \cdot (-1{,}4) = 7$$

a dostaneme $x_1 = -0{,}28$. Místo přesného řešení $\mathbf{x}$ jsme dostali přibližné řešení

$$\tilde{\mathbf{x}} = \begin{pmatrix} -0{,}28 \\ -1{,}4 \\ 0{,}99993 \end{pmatrix}.$$

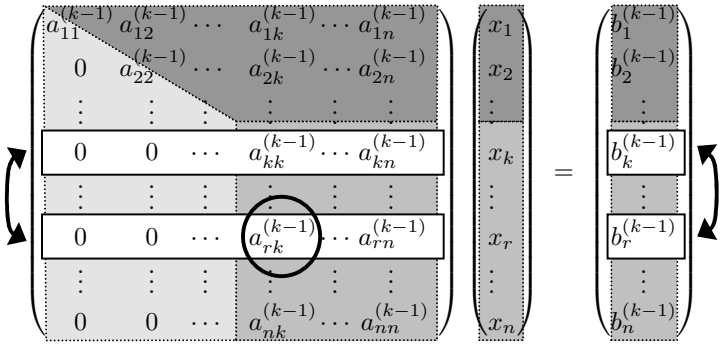
Kde vznikl problém? Nedošlo k žádnému hromadění chyb způsobenému prováděním tisíců operací. Matice soustavy není blízká matici singulární. Potíž je jinde, působí ji malý pivot ve druhém kroku eliminace. Tím vznikne multiplikátor -2400 a v důsledku toho má poslední rovnice koeficienty zhruba 1000 krát větší než koeficienty původní rovnice. Zaokrouhlovací chyby, které jsou malé vzhledem k těmto velkým koeficientům, jsou nepřijatelné pro koeficienty původní matice a také pro samotné řešení.

Snadno se prověří, že když druhou a třetí rovnici prohodíme, nevzniknou žádné velké multiplikátory a výsledek je zcela přesný. Ukazuje se, že to platí obecně: jestliže jsou absolutní hodnoty multiplikátorů menší nebo nejvýše rovny 1, pak je numericky spočtené řešení vyhovující. □

Částečný výběr hlavního prvku je modifikace GEM zajišťující, aby absolutní hodnota multiplikátorů byla menší nebo rovna jedné. V k-tém kroku eliminace se jako pivot vybírá prvek s největší absolutní hodnotou v zatím neeliminované části k-tého sloupce matice $\mathbf{A}^{(k-1)}$, tj. mezi prvky $a_{ik}^{(k-1)}$ pro $i \geq k$. Nechť tedy r je takový řádkový index, pro který

$$|a_{rk}^{(k-1)}| = \max_{k \leq i \leq n} |a_{ik}^{(k-1)}|. \tag{2.14}$$

Pak prohodíme k-tou a r-tou rovnici. Ze soustavy rovnic $\mathbf{A}^{(k-1)}\mathbf{x} = \mathbf{b}^{(k-1)}$ tak dostaneme soustavu $\mathbf{A}^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$, přičemž $\mathbf{A}^{(k)}$ získáme prohozením k-tého a r-tého řádku matice $\mathbf{A}^{(k-1)}$ a podobně $\mathbf{b}^{(k)}$ získáme prohozením k-tého a r-tého prvku vektoru $\mathbf{b}^{(k-1)}$. Poddiagonální prvky v k-tém sloupci matice $\mathbf{A}^{(k)}$ eliminujeme stejně jako v algoritmu GEMz. Řešení soustavy lineárních rovnic s částečným výběrem hlavních prvků v MATLABu dostaneme pomocí příkazu `x=A\b`.



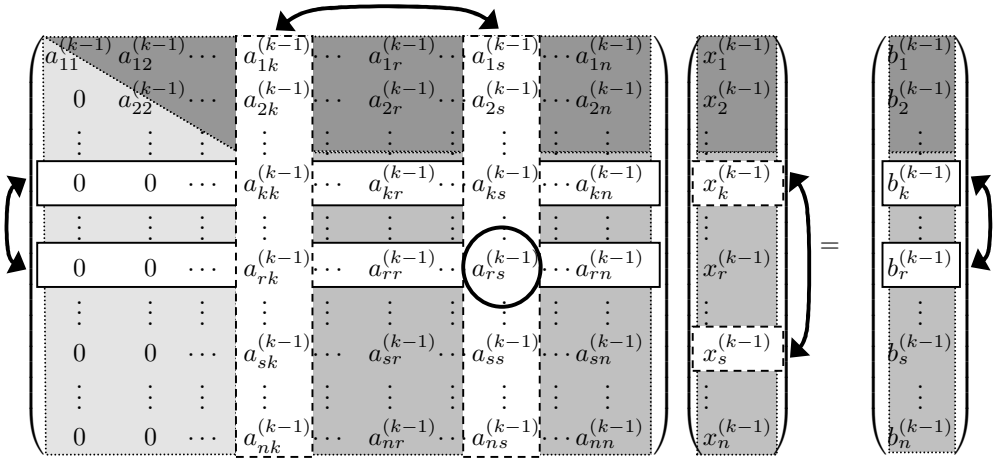
Obr. 2.1: GEM s částečným výběrem hlavního prvku (v kroužku)

Úplný výběr hlavního prvku je postup, který může absolutní hodnoty multiplikátorů zmenšit ještě výrazněji. Dociluje se toho tím, že v k-tém kroku eliminace se jako pivot vybírá prvek s největší absolutní hodnotou v dosud neeliminované části matice $\mathbf{A}^{(k-1)}$, tj.

v řádcích $i \geq k$ a sloupcích $j \geq k$. Nechť tedy r je řádkový a s sloupcový index vybraný tak, že

$$|a_{rs}^{(k-1)}| = \max_{k \leq i,j \leq n} |a_{ij}^{(k-1)}| \,. \tag{2.15}$$

Pak prohodíme k-tou a r-tou rovnici a k-tou a s-tou neznámou. Ze soustavy rovnic



Obr. 2.2: GEM s úplným výběrem hlavního prvku (v kroužku)

$\mathbf{A}^{(k-1)}\mathbf{x}^{(k-1)} = \mathbf{b}^{(k-1)}$ dostaneme soustavu $\mathbf{A}^{(k)}\mathbf{x}^{(k)} = \mathbf{b}^{(k)}$, přičemž $\mathbf{A}^{(k)}$ získáme prohozením k-tého a r-tého řádku a k-tého a s-tého sloupce matice $\mathbf{A}^{(k-1)}$, $\mathbf{b}^{(k)}$ získáme prohozením k-tého a r-tého prvku vektoru $\mathbf{b}^{(k-1)}$ a $\mathbf{x}^{(k)}$ získáme prohozením k-tého a s-tého prvku vektoru $\mathbf{x}^{(k-1)}$. Přitom $\mathbf{x}^{(0)} = \mathbf{x}$ je původní vektor neznámých. Prohazování proměnných registrujeme ve vektoru $\mathbf{p} = (p_1, p_2, \ldots, p_n)^T$. Na počátku položíme $p_i = i$ a při každém prohození proměnných prohodíme také odpovídající prvky vektoru $\mathbf{p}$. Po ukončení přímého chodu je i-tá složka vektoru $\mathbf{x}^{(n-1)}$ rovna p_i-té složce původního vektoru $\mathbf{x}$. Ve zpětném chodu vypočteme $\mathbf{x}^{(n-1)}$ a pomocí vektoru $\mathbf{p}$ určíme $\mathbf{x}$.

Částečný nebo úplný výběr hlavního prvku? Většinou se používá jen částečný výběr hlavního prvku. Praxe i teorie potvrzuje, že i částečný výběr hlavních prvků stačí k tomu, aby zaokrouhlovací chyby zůstaly dostatečně malé a neznehodnotily výsledné řešení. Dalším důvodem, proč částečnému výběru hlavních prvků dáváme přednost, je to, že jeho realizace vyžaduje výrazně menší počet operací než výběr úplný.

LU rozklad s částečným výběrem hlavního prvku je standardní rutina dostupná v každé knihovně programů pro numerické řešení úloh lineární algebry (v MATLABu viz funkce `lu`). Vstupním parametrem je matice $\mathbf{A}$. Výstupní parametry jsou tři: dolní trojúhelníková matice $\mathbf{L}$ (s jedničkami na hlavní diagonále), horní trojúhelníková matice $\mathbf{U}$ a tzv. permutační matice $\mathbf{P}$, přičemž

$$\mathbf{LU} = \mathbf{PA} \,. \tag{2.16}$$

Přitom *permutační matice* je taková matice, která vznikne z jednotkové matice nějakým proházením jejích řádků. Následuje popis algoritmu pro LU rozklad matice $\mathbf{A}$ s částečným výběrem hlavních prvků.

Algoritmus LUp (s částečným výběrem hlavního prvku)

1) Položíme $\mathbf{A}^{(0)} = \mathbf{A}$, $\mathbf{P}^{(0)} = \mathbf{I}$.

2) Postupně pro $k = 1, 2, \ldots, n-1$ provádíme

   2a) Určíme index r pivotního řádku, viz (2.14).

   2b) Prohodíme řádky k a r v matici $\mathbf{A}^{(k-1)}$ a takto získanou matici označíme jako $\mathbf{A}^{(k)}$. Prohodíme rovněž řádky k a r v matici $\mathbf{P}^{(k-1)}$ a takto získanou matici označíme jako $\mathbf{P}^{(k)}$.

   2c) Upravíme matici $\mathbf{A}^{(k)}$ tak, že eliminujeme poddiagonální prvky v k-tém sloupci, tj. postupně pro $i = k+1, k+2, \ldots, n$ počítáme

$$\begin{aligned} m_{ik} &:= a_{ik}^{(k)} / a_{kk}^{(k)} \,, \\ a_{ij}^{(k)} &:= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \qquad \text{pro } j = k+1, k+2, \ldots, n, \\ a_{ik}^{(k)} &:= m_{ik} \,. \end{aligned}$$

   Do anulovaných pozic (i,k) matice $\mathbf{A}^{(k)}$ tedy ukládáme multiplikátory m_{ik}.

3) Dolní trojúhelníkovou matici $\mathbf{L}$ sestrojíme tak, že do hlavní diagonály dáme jedničky a poddiagonální prvky převezmeme z výsledné matice $\mathbf{A}^{(n-1)}$. Horní trojúhelníkovou matici $\mathbf{U}$ dostaneme z diagonálních a naddiagonálních prvků výsledné matice $\mathbf{A}^{(n-1)}$. Permutační matice $\mathbf{P}$ je rovna výsledné permutační matici $\mathbf{P}^{(n-1)}$.

Po získání matic $\mathbf{L}$, $\mathbf{U}$ a $\mathbf{P}$ vyřešíme už soustavu rovnic $\mathbf{Ax} = \mathbf{b}$ snadno. Zřejmě

$$\mathbf{PAx} = \mathbf{Pb} \qquad \Longrightarrow \qquad \mathbf{LUx} = \mathbf{Pb} \,.$$

Proto nejdříve určíme vektor $\mathbf{z} = \mathbf{Pb}$, tj. prvky vektoru $\mathbf{b}$ pravé strany proházíme stejně, jako jsme prohazovali řádky matic $\mathbf{A}^{(k-1)}$ a $\mathbf{P}^{(k-1)}$ v algoritmu LUp. Označíme-li $\mathbf{Ux} = \mathbf{y}$, vidíme, že $\mathbf{y}$ je řešení soustavy $\mathbf{Ly} = \mathbf{z}$. Vyřešením této soustavy dostaneme $\mathbf{y}$. Zbývá ještě vyřešit soustavu $\mathbf{Ux} = \mathbf{y}$ a řešení $\mathbf{x}$ je nalezeno. Shrneme-li to, počítáme postupně $\mathbf{z}$, $\mathbf{y}$ a $\mathbf{x}$ z rovnic

$$\mathbf{z} = \mathbf{Pb} \,, \qquad \mathbf{Ly} = \mathbf{z} \,, \qquad \mathbf{Ux} = \mathbf{y} \,. \tag{2.17}$$

Uchovávat historii prohazování řádků v matici $\mathbf{P}$ je zřejmé plýtvání paměťovým místem, vždyť z celkového počtu n^2 prvků matice $\mathbf{P}$ je jich pouze n nenulových. Proto místo s maticí $\mathbf{P}$ stačí pracovat jen s *vektorem řádkových permutací* $\mathbf{p}$.

Na začátku položíme $\mathbf{p}^{(0)} = (1, 2, \ldots, n)^T$ a ve zbytku algoritmu pak prohazujeme prvky vektoru $\mathbf{p}^{(k-1)}$. Matice $\mathbf{P}$ byla zapotřebí jen pro sestavení vektoru $\mathbf{z}$. To ale dokážeme s pomocí vektoru $\mathbf{p}$ také: do i-té složky vektoru $\mathbf{z}$ vložíme p_i-tou složku vektoru $\mathbf{b}$, postupně pro $i = 1, 2, \ldots, n$.

Příklad 2.2. Provedeme *LU* rozklad matice

$$\mathbf{A} = \begin{pmatrix} -0{,}4 & -0{,}95 & -0{,}4 & -7{,}34 \\ 0{,}5 & -0{,}3 & 2{,}15 & -2{,}45 \\ -2 & 4 & 1 & -3 \\ -1 & 5{,}5 & 2{,}5 & 3{,}5 \end{pmatrix} .$$

V prvním sloupci najdeme jako pivota číslo -2 ve třetím řádku. Proto prohodíme první a třetí řádek. Pak eliminujeme poddiagonální prvky v prvním sloupci a na jejich místa zapíšeme použité multiplikátory. Prohození řádků vyznačíme také v permutačním vektoru. Tak dostaneme

$$\mathbf{A}^{(1)} = \begin{pmatrix} -2 & 4 & 1 & -3 \\ -0{,}25 & 0{,}7 & 2{,}4 & -3{,}2 \\ 0{,}2 & -1{,}75 & -0{,}6 & -6{,}74 \\ 0{,}5 & 3{,}5 & 2 & 5 \end{pmatrix}, \qquad \mathbf{p}^{(1)} = \begin{pmatrix} 3 \\ 2 \\ 1 \\ 4 \end{pmatrix}$$

První řádek se už měnit nebude. Ve druhém kroku najdeme pivota ve druhém sloupci. Je to číslo 3,5 ve čtvrtém řádku. Proto prohodíme druhý a čtvrtý řádek jak v matici $\mathbf{A}^{(1)}$ tak ve vektoru $\mathbf{p}^{(1)}$. Pak eliminujeme prvky v pozicích (3,2) a (4,2) a na jejich místa zapíšeme použité multiplikátory. Výsledkem je

$$\mathbf{A}^{(2)} = \begin{pmatrix} -2 & 4 & 1 & -3 \\ 0{,}5 & 3{,}5 & 2 & 5 \\ 0{,}2 & -0{,}5 & 0{,}4 & -4{,}24 \\ -0{,}25 & 0{,}2 & 2 & -4{,}2 \end{pmatrix}, \qquad \mathbf{p}^{(2)} = \begin{pmatrix} 3 \\ 4 \\ 1 \\ 2 \end{pmatrix}.$$

První dva řádky už zůstanou bez změny. Pivot ve třetím sloupci je číslo 2 ve čtvrtém řádku. Prohodíme tedy třetí a čtvrtý řádek v matici $\mathbf{A}^{(2)}$ i ve vektoru $\mathbf{p}^{(2)}$. Pak eliminujeme prvek v pozici (4,3) a na jeho místo vložíme použitý multiplikátor. Tak dostaneme

$$\mathbf{A}^{(3)} = \begin{pmatrix} -2 & 4 & 1 & -3 \\ 0{,}5 & 3{,}5 & 2 & 5 \\ -0{,}25 & 0{,}2 & 2 & -4{,}2 \\ 0{,}2 & -0{,}5 & 0{,}2 & -3{,}4 \end{pmatrix}, \qquad \mathbf{p}^{(3)} = \begin{pmatrix} 3 \\ 4 \\ 2 \\ 1 \end{pmatrix}.$$

Dostali jsme tedy

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0{,}5 & 1 & 0 & 0 \\ -0{,}25 & 0{,}2 & 1 & 0 \\ 0{,}2 & -0{,}5 & 0{,}2 & 1 \end{pmatrix}, \qquad \mathbf{U} = \begin{pmatrix} -2 & 4 & 1 & -3 \\ 0 & 3{,}5 & 2 & 5 \\ 0 & 0 & 2 & -4{,}2 \\ 0 & 0 & 0 & -3{,}4 \end{pmatrix}.$$

Permutační vektor $\mathbf{p} = \mathbf{p}^{(3)}$. Pokud bychom chtěli vytvořit permutační matici $\mathbf{P}$, stačí vzít jednotkovou matici a přeuspořádat ji tak, že původně p_i-tý řádek se stane řádkem i-tým. Když to provedeme, dostaneme pro permutační vektor

$$\mathbf{p} = \begin{pmatrix} 3 \\ 4 \\ 2 \\ 1 \end{pmatrix} \qquad \text{permutační matici} \qquad \mathbf{P} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Snadno se ověří, že $\mathbf{LU} = \mathbf{PA}$.

Ukažme si ještě řešení soustavy rovnic pro volenou pravou stranu. Zvolme třeba

$$\mathbf{b} = \begin{pmatrix} -13{,}14 \\ 2{,}15 \\ 9 \\ 27{,}5 \end{pmatrix}, \qquad \text{pak} \qquad \mathbf{z} = \begin{pmatrix} 9 \\ 27{,}5 \\ 2{,}15 \\ -13{,}14 \end{pmatrix},$$

a dále řešením soustav $\mathbf{Ly} = \mathbf{z}$ a pak $\mathbf{Ux} = \mathbf{y}$ dostaneme

$$\mathbf{y} = \begin{pmatrix} 9 \\ 23 \\ -0{,}2 \\ -3{,}4 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} 3 \\ 4 \\ 2 \\ 1 \end{pmatrix}.$$

Výpočet determinantu. Je známo, že determinant $\det(\mathbf{A})$ matice $\mathbf{A}$

a) se nezmění, když k řádku matice $\mathbf{A}$ přičteme násobek jiného řádku matice $\mathbf{A}$;

b) změní znaménko, když prohodíme dva jeho (různé) řádky nebo sloupce.

Provádíme-li tedy GEM podle algoritmu GEMz, tj. bez výběru hlavních prvků, pak $\det(\mathbf{A}) = \det(\mathbf{U}) = u_{11}u_{22}\cdots u_{nn}$ dostaneme jako součin diagonálních prvků matice $\mathbf{U}$. Pokud provádíme částečný nebo úplný výběr hlavních prvků, pak stačí, když si poznamenáme, třeba do proměnné q, celkový počet prohození řádků (pro částečný výběr) nebo řádků i sloupců (pro úplný výběr). Je-li q sudé, pak $\det(\mathbf{A}) = \det(\mathbf{U})$, a je-li q liché, je $\det(\mathbf{A}) = -\det(\mathbf{U})$. Platí tedy

$$\det(\mathbf{A}) = (-1)^q\, u_{11}u_{22}\cdots u_{nn}\,. \tag{2.18}$$

Řešení soustavy rovnic s více pravými stranami nepředstavuje žádný problém, místo s jednou pravou stranou pracujeme současně s m pravými stranami. Vzorce (2.9) a (2.17) zůstávají v platnosti, jediný rozdíl je v tom, že teď $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m)$ je matice pravých stran, takže také $\mathbf{y}$, $\mathbf{x}$ a případně $\mathbf{z}$ jsou matice téhož typu jako $\mathbf{b}$. Zejména tedy i-tý sloupec matice $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ je řešení příslušné i-té pravé straně. Případ úplného výběru hlavních prvků zde rozebírat nebudeme.

Pokud jde o počet operací, tak pro každou pravou stranu je třeba započítat přibližně n^2 operací násobicích a stejný počet operací sčítacích, takže celkem jde o mn^2 operací. Je-li počet m pravých stran malý ve srovnání s počtem n rovnic, jsou náklady na provedení LU rozkladu, tj. přibližně $\frac{1}{3}n^3$ operací, výrazně převažující.

Výpočet matice inverzní lze provést tak, že řešíme soustavu rovnic $\mathbf{Ax} = \mathbf{b}$ s n pravými stranami pro $\mathbf{b} = \mathbf{I}$, kde $\mathbf{I}$ je jednotková matice. Když úlohu řešíme buďto bez výběru hlavních prvků nebo s částečným výběrem hlavních prvků, pak dostaneme $\mathbf{x} = \mathbf{A}^{-1}$, tj. matici inverzní. Počet operací potřebný pro řešení trojúhelníkových soustav vyžaduje přibližně n^3 operací. Protože LU rozklad potřebuje přibližně $\frac{1}{3}n^3$ operací, je to celkem $\frac{4}{3}n^3$ operací. Dá se ukázat, že při důmyslně organizovaném výpočtu lze počet potřebných operací snížit o jednu třetinu, tj. na přibližně n^3 operací, viz např. [3].

Řídká matice soustavy. Řekneme, že *matice je řídká*, když počet jejích nenulových prvků je výrazně menší než počet všech jejích prvků. Takové matice vznikají při řešení řady významných aplikací. Častý je případ, kdy počet nenulových koeficientů v rovnici nepřevýší malé číslo m, které vůbec nezávisí na počtu n rovnic, takže s růstem n dostáváme stále řidší matice soustav. Řídké matice jsou v paměti počítače účelně reprezentovány jen pomocí svých nenulových koeficientů. Pro řešení soustav s řídkými maticemi existují velice efektivní algoritmy. Jedním z hlavních cílů, které tyto algoritmy sledují, je provádění eliminačních kroků v takovém pořadí, aby vznikalo co nejméně nových nenulových koeficientů.

Pásová matice soustavy. Speciálním případem řídké matice je *pásová matice*, která má nenulové koeficienty jen v pásu okolo hlavní diagonály. Přesněji, matice $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{n}$, pro kterou existují celá nezáporná čísla p, q taková, že

$$a_{ij} = 0 \qquad \text{když} \quad i > j + p \qquad \text{nebo} \quad j > i + q\,,$$

se nazývá pásová matice s pásem o šířce $w = p+q+1$ (má p poddiagonál a q naddiagonál). Číslo $s = \max(p, q)$ se nazývá *poloviční šířka pásu*. Pro matici s poloviční šířkou pásu s tedy zřejmě platí

$$a_{ij} = 0 \qquad \text{pro } |i - j| > s.$$

Při eliminaci pásových matic mohou nenulové koeficienty vznikat jen uvnitř pásu. Toho lze využít a docílit znatelnou úsporu jak v reprezentaci matice soustavy v paměti počítače, tak v počtu potřebných operací. Pro matici s poloviční šířkou pásu $s = p = q$ potřebuje LU rozklad bez výběru hlavních prvků $ns(s+\delta)+O(s^3)$ operací a zpětný chod $n(2s+\delta)+O(s^2)$ operací, kde $\delta = 0$ pro operace sčítací a $\delta = 1$ pro operace násobicí.

Soustava rovnic s třídiagonální maticí. V případě $s = 1$ hovoříme o *třídiagonální matici*. Algoritmus GEM pro řešení soustavy rovnic s třídiagonální maticí je velmi jednoduchý. Bez výběru hlavních prvků pro soustavu

$$\begin{pmatrix} a_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ b_1 & a_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & b_2 & a_3 & c_3 & \dots & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & & b_{n-2} & a_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & & 0 & b_{n-1} & a_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

v přímém chodu počítáme

$$b_i := b_i/a_i, \qquad a_{i+1} := a_{i+1} - b_i c_i, \qquad d_{i+1} := d_{i+1} - b_i d_i\,, \quad i = 1, 2, \dots, n-1\,,$$

a ve zpětném chodu určíme

$$x_n := d_n/a_n \qquad \text{a dále} \qquad x_i := (d_i - c_i x_{i+1})/a_i\,, \quad i = n-1, n-2, \dots, 1.$$

Transformovaná matice soustavy obsahuje koeficienty LU rozkladu původní matice.

2.1.3. Vliv zaokrouhlovacích chyb

Při řešení soustav lineárních rovnic téměř vždy působí zaokrouhlovací chyby. Jejich vliv prozkoumáme v následujícím příkladu.

Příklad 2.3. Předpokládejme, že na hypotetickém počítači s třímístnou mantisou máme vyřešit soustavu

$$\begin{pmatrix} 3{,}96 & 1{,}01 \\ 1 & 0{,}25 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5{,}03 \\ 1{,}25 \end{pmatrix}.$$

Přibližné řešení budeme značit $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)^T$. Protože $3{,}96 > 1$, nemusíme rovnice prohazovat a multiplikátor $m_{21} = 1/3{,}96 \doteq 0{,}253$. Od druhé rovnice odečítáme m_{21}-násobek první rovnice, tj.

$$(0{,}25 - 0{,}253 \cdot 1{,}01) \cdot x_2 = 1{,}25 - 0{,}253 \cdot 5{,}03\,.$$

Při zaokrouhlování na 3 platné cifry dostaneme $1{,}01 \cdot 0{,}253 \doteq 0{,}256$ a $5{,}03 \cdot 0{,}253 \doteq 1{,}27$, takže

$$\tilde{x}_2 = \frac{-0{,}02}{-0{,}006} \doteq 3{,}33\,.$$

Z první rovnice pak vypočteme $\tilde{x}_1 = (5{,}03 - 1{,}01 \cdot 3{,}33)/3{,}96 \doteq 0{,}422$. Dostali jsme tedy přibližné řešení

$$\tilde{\mathbf{x}} = \begin{pmatrix} 0{,}422 \\ 3{,}33 \end{pmatrix}.$$

Spočteme-li (přesně) reziduum $\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$, obdržíme

$$\mathbf{r} = \begin{pmatrix} 5{,}03 - 3,96 \cdot 0{,}422 - 1{,}01 \cdot 3{,}33 \\ 1{,}25 - 1 \cdot 0{,}422 - 0{,}25 \cdot 3{,}33 \end{pmatrix} = \begin{pmatrix} -0{,}00442 \\ -0{,}00450 \end{pmatrix}.$$

To by nás mohlo svádět k domněnce, že získané řešení $\tilde{\mathbf{x}}$ je prakticky přesné. Tak tomu ale není, přesné řešení je

$$\mathbf{x} = \begin{pmatrix} 0{,}25 \\ 4 \end{pmatrix},$$

takže $\tilde{x}_1$ a $\tilde{x}_2$ nemají ani jednu cifru platnou! Pro zajímavost uvádíme, že pro $p = 4, 6, \ldots$ cifer mantisy dostaneme přesné řešení a pro $p = 5, 7, \ldots$ řešení, jehož složky mají $p - 3$ platných cifer. □

Přestože příklad 2.3 je značně umělý, je jednoduchou ilustrací obecně platného tvrzení:

Gaussova eliminace s částečným výběrem hlavních prvků zaručuje vznik malých reziduí.

K tomuto tvrzení je třeba připojit několik vysvětlujících poznámek. Slovem zaručuje se míní, že lze dokázat přesnou větu, která (za splnění jistých technických předpokladů týkajících se výpočtů v pohyblivé řádové čárce) uvádí nerovnosti omezující velikost jednotlivých složek rezidua. Dále slovem malých míníme v řádu zaokrouhlovacích chyb *relativně* vzhledem ke třem veličinám: k velikosti prvků původní matice koeficientů $\mathbf{A}$, k velikosti prvků matic $\mathbf{A}^{(k)}$ vznikajících v průběhu eliminace a k velikosti prvků řešení $\tilde{\mathbf{x}}$. Konečně je třeba dodat, že i když je reziduum malé, tvrzení neříká nic o velikosti chyby $\tilde{\mathbf{x}} - \mathbf{x}$.

K posouzení vztahu mezi velikostí rezidua a velikostí chyby se používá veličina známá jako číslo podmíněnosti matice.

2.1.4. Podmíněnost

Abychom mohli určit podmíněnost úlohy najít řešení $\mathbf{x}$ soustavy lineárních rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$, potřebujeme posoudit, jak moc se změní řešení $\mathbf{x}$, když trochu změníme data, tj. matici soustavy $\mathbf{A}$ a vektor pravé strany $\mathbf{b}$. Zatímco k měření velikosti čísel používáme absolutní hodnotu, podobný nástroj pro měření velikosti vektorů a matic si teprve musíme zavést.

Norma vektoru je nezáporné číslo, které reprezentuje jeho velikost. Třída vektorových norem, známá jako l_p, závisí na parametru $1 \leq p \leq \infty$:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

Nejčastěji se používá $p = 1$, $p = 2$ nebo limitní případ pro $p \to \infty$:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|\,, \qquad \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}, \qquad \|\mathbf{x}\|_\infty = \max_i |x_i|\,.$$

l_1-norma je známa také jako *Manhattan* norma, neboť odpovídá vzdálenosti mezi dvěma místy v pravoúhlé mříži městských ulic. l_2-norma je běžná Euclidova vzdálenost neboli *délka vektoru*. l_∞-norma je známa také jako *Čebyševova* norma. V MATLABu lze vektorové normy $\|\cdot\|_p$ určit pomocí funkce `norm`.

Konkrétní hodnota p je často nepodstatná, normu pak jednoduše značíme $\|\mathbf{x}\|$. Vektorová norma splňuje následující základní vlastnosti, typické pro pojem vzdálenosti:

1. $\|\mathbf{x}\| > 0\,,$ když $\mathbf{x} \neq \mathbf{o}$ a $\|\mathbf{o}\| = 0,$
2. $\|c\,\mathbf{x}\| = |c| \cdot \|\mathbf{x}\|$ pro každé číslo $c,$
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ (trojúhelníková nerovnost).

Symbolem $\mathbf{o}$ jsme přitom označili *nulový vektor*, tj. vektor, jehož všechny složky jsou rovny nule.

Číslo podmíněnosti matice. Násobením vektoru $\mathbf{x}$ maticí $\mathbf{A}$ zleva dostaneme nový vektor $\mathbf{A}\mathbf{x}$, který může mít úplně jinou normu než $\mathbf{x}$. Tato změna normy přímo souvisí s citlivostí, kterou chceme měřit. Rozsah možných změn lze vyjádřit pomocí dvou čísel:

$$M = \max_{\mathbf{x}\neq\mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}\,, \qquad m = \min_{\mathbf{x}\neq\mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}\,, \tag{2.19}$$

kde maximum resp. minimum se uvažuje pro všechny nenulové vektory $\mathbf{x}$. Poměr M/m se nazývá *číslo podmíněnosti* matice $\mathbf{A}$:

$$\kappa(\mathbf{A}) = \frac{M}{m} = \left(\max_{\mathbf{x}\neq\mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \right) \cdot \left(\min_{\mathbf{x}\neq\mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \right)^{-1}. \tag{2.20}$$

Konkrétní hodnota $\kappa(\mathbf{A})$ závisí na použité vektorové normě. Protože nás však obvykle zajímá jen řádová velikost vhodně spočteného odhadu čísla podmíněnosti, nebývá použitý typ normy většinou podstatný.

Uvažujme systém rovnic

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

a druhý systém rovnic, který dostaneme, když změníme pravou stranu:

$$\mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}\,, \qquad \text{nebo-li} \qquad \mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}.$$

$\Delta\mathbf{b} = \tilde{\mathbf{b}} - \mathbf{b}$ můžeme považovat za chybu pravé strany $\mathbf{b}$ a $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$ za odpovídající chybu řešení $\mathbf{x}$. Protože $\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}$, z definice M a m okamžitě plyne

$$\|\mathbf{b}\| \le M\|\mathbf{x}\|\,, \qquad \|\Delta\mathbf{b}\| \ge m\|\Delta\mathbf{x}\|\,,$$

takže pro $m \neq 0$,

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \kappa(\mathbf{A})\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} = \kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}\,, \qquad \text{kde} \qquad \mathbf{r} = \mathbf{b} - \mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) \tag{2.21}$$

je reziduum. Číslo podmíněnosti tedy působí jako zesilovač velikosti relativní chyby: relativní změna $\|\Delta\mathbf{b}\|/\|\mathbf{b}\|$ pravé strany vyvolá $\kappa(\mathbf{A})$ krát větší relativní změnu $\|\Delta\mathbf{x}\|/\|\mathbf{x}\|$ řešení. Dá se ukázat, že změny v matici koeficientů mají na změny řešení obdobný vliv.

Nerovnost (2.21) také potvrzuje zkušenost, kterou jsme udělali v příkladu 2.3, totiž že malé reziduum neznamená malou relativní chybu. Na pravé straně nerovnosti (2.21) je totiž norma rezidua $\|\mathbf{r}\|$ násobena číslem podmíněnosti $\kappa(\mathbf{A})$ matice $\mathbf{A}$, takže i když je reziduum malé, může být přesto relativní chyba řešení velká, když $\kappa(\mathbf{A})$ je velké.

Všimněme si některých vlastností čísla podmíněnosti.

1. Protože $M \ge m$, je $\kappa(\mathbf{A}) \ge 1$.
2. Pro jednotkovou matici $\mathbf{I}$ je $\mathbf{I}\mathbf{x} = \mathbf{x}$, takže $\kappa(\mathbf{I}) = 1$.
3. Je-li $\mathbf{A}$ singulární, je $m = 0$ a tedy $\kappa(\mathbf{A}) = \infty$.
4. Když matici $\mathbf{A}$ vynásobíme číslem c, pak M i m se násobí $|c|$ a $\kappa(c\mathbf{A}) = \kappa(\mathbf{A})$.
5. Pro diagonální matici $\mathbf{D}$ je $\kappa(\mathbf{D}) = \max_i |d_{ii}| \,/\, \min_i |d_{ii}|$ (dokažte!).

Z posledních dvou vlastností plyne, že $\kappa(\mathbf{A})$ je lepším měřítkem blízkosti matice $\mathbf{A}$ k matici singulární než její determinant $\det(\mathbf{A})$. Jako extrémní příklad uvažujme diagonální matici řádu 100, která má na diagonále čísla 0,1. Pak $\det(\mathbf{A}) = 10^{-100}$, což lze považovat za velmi malé číslo, avšak $\kappa(\mathbf{A}) = 1$. Soustava rovnic s takovou maticí se chová spíše jako soustava s jednotkovou maticí než jako soustava s maticí téměř singulární. V MATLABu lze číslo podmíněnosti matice určit pomocí funkce `cond`.

Norma matice. Číslo M je známo jako *norma matice*. Označujeme ji stejně jako normu vektoru:

$$\|\mathbf{A}\| = \max_{\mathbf{x}\neq\mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|\,. \tag{2.22}$$

Snadno se ověří, že $\|\mathbf{A}^{-1}\| = 1/m$, takže ekvivalentní vyjádření čísla podmíněnosti je

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \,. \tag{2.23}$$

Znovu je třeba připomenout, že konkrétní hodnota $\kappa(\mathbf{A})$ závisí na použité vektorové normě. Snadno se spočtou maticové normy odpovídající vektorovým normám l_1 a l_∞. Platí

$$\|\mathbf{A}\|_1 = \max_j \sum_i |a_{ij}| \qquad \text{(maximum sloupcových součtů)}\,,$$

$$\|\mathbf{A}\|_\infty = \max_i \sum_j |a_{ij}| \qquad \text{(maximum řádkových součtů)}\,.$$

Často používanou maticovou normu příslušnou k nejznámější vektorové l_2-normě už snadno spočítat nelze. Platí totiž

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{max}(\mathbf{A}^T\mathbf{A})}, \quad \text{kde } \lambda_{max}(\mathbf{A}^T\mathbf{A}) \text{ je největší vlastní číslo matice } \mathbf{A}^T\mathbf{A},$$

což je největší kořen polynomu $p(x) = \det(\mathbf{A}^T\mathbf{A} - x\cdot\mathbf{I})$. Je-li matice $\mathbf{A}$ řádu n, pak $p(x)$ je polynom stupně n a jeho kořeny pro $n > 4$ umíme najít jen numericky, viz kapitola 5. Poznamenejme, že v MATLABu lze vlastní čísla matic určit pomocí funkce `eig`.

Více o normách matic. Maticová norma definovaná vztahem (2.22) se nazývá *přirozená* norma. Někdy také říkáme, že je to norma *přidružená* k vektorové normě, pomocí níž je definována. Například maticová $\|\cdot\|_\infty$ norma je přidružená k vektorové ℓ_∞-normě.

Všimněte si, že z definice (2.22) přirozené maticové normy okamžitě plyne

$$\|\mathbf{A}\mathbf{x}\| \le \|\mathbf{A}\| \cdot \|\mathbf{x}\| \,. \tag{2.24}$$

Jestliže vektorová a maticová norma splňují vztah (2.24), říkáme, že jsou *souhlasné*.

Existují i jiné než přirozené maticové normy. Jednou z nich je *Frobeniova* norma

$$\|\mathbf{A}\|_F = \left(\sum_{i,j=1}^{n} a_{ij}^2 \right)^{1/2}, \tag{2.25}$$

o které lze dokázat, že je to norma souhlasná s Euklidovou l_2-normou, tj. že platí

$$\|\mathbf{A}\mathbf{x}\|_2 \le \|\mathbf{A}\|_F \cdot \|\mathbf{x}\|_2 \,.$$

V MATLABu lze maticové normy $\|\cdot\|_1$, $\|\cdot\|_2$,$\|\cdot\|_\infty$ a $\|\cdot\|_F$ určit pomocí funkce `norm`.

Maticové normy, které jsme si doposud definovali, mají následující význačné vlastnosti:

1. $\|\mathbf{A}\| > 0\,,$ když $\mathbf{A} \neq \mathbf{O}$ a $\|\mathbf{O}\| = 0,$
2. $\|c\mathbf{A}\| = |c| \cdot \|\mathbf{A}\|$ pro každé číslo c,
3. $\|\mathbf{A} + \mathbf{B}\| \le \|\mathbf{A}\| + \|\mathbf{B}\|$ (trojúhelníková nerovnost),
4. $\|\mathbf{A}\mathbf{B}\| \le \|\mathbf{A}\| \cdot \|\mathbf{B}\|$ (multiplikativnost),
5. $\|\mathbf{A}\mathbf{x}\| \le \|\mathbf{A}\| \cdot \|\mathbf{x}\|$ pro každý vektor $\mathbf{x}$ (souhlasnost).

Symbolem $\mathbf{O}$ jsme si přitom označili *nulovou matici*, tj. matici, jejíž všechny prvky jsou rovny nule. Poznamenejme, že obecně je norma čtvercové matice definována jako reálná funkce splňující jen první čtyři z výše uvedených podmínek.

Příklad 2.4. Soustava rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$, kde

$$\mathbf{A} = \begin{pmatrix} 1 & 10 \\ 10 & 101 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 11 \\ 111 \end{pmatrix}, \qquad \text{má řešení} \qquad \mathbf{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Použijeme l_∞-normu a spočteme $\|\mathbf{b}\|_\infty = 111$, $\|\mathbf{x}\|_\infty = 1$. Když pravou stranu změníme na

$$\tilde{\mathbf{b}} = \begin{pmatrix} 11{,}11 \\ 110{,}89 \end{pmatrix}, \qquad \text{dostaneme řešení} \qquad \tilde{\mathbf{x}} = \begin{pmatrix} 13{,}21 \\ -0{,}21 \end{pmatrix}.$$

Označíme-li $\Delta\mathbf{b} = \tilde{\mathbf{b}} - \mathbf{b}$, $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$, pak $\|\Delta\mathbf{b}\|_\infty = 0{,}11$ a $\|\Delta\mathbf{x}\|_\infty = 12{,}21$. Vidíme, že poměrně malá změna pravé strany zcela změnila řešení. Relativní změny jsou

$$\frac{\|\Delta\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty} = 9{,}\overline{909} \cdot 10^{-4}, \qquad \frac{\|\Delta\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} = 12{,}21\,.$$

Podle (2.21) můžeme odhadnout číslo podmíněnosti

$$\kappa(\mathbf{A}) \geq \frac{12{,}21}{9{,}\overline{909} \cdot 10^{-4}} = 12321.$$

Ve skutečnosti je $\mathbf{b}$ a $\Delta\mathbf{b}$ zvoleno tak, že $\kappa(\mathbf{A}) = 12321$. To se snadno ověří, neboť

$$\mathbf{A}^{-1} = \begin{pmatrix} 101 & -10 \\ -10 & 1 \end{pmatrix}, \text{ takže } \|\mathbf{A}^{-1}\|_\infty = 111 = \|\mathbf{A}\|_\infty \text{ a } \kappa(\mathbf{A}) = 111^2 = 12321.$$

Ukažme si ještě, že vztah (2.21) platí jako rovnost:

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} = \frac{12{,}21}{1} = 111^2\,\frac{0{,}11}{111} = \kappa(\mathbf{A})\frac{\|\Delta\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty} = 111 \cdot \|\Delta\mathbf{b}\|_\infty = 111 \cdot \|\mathbf{r}\|_\infty\,,$$

kde $\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$ je reziduum. □

K určení $\kappa(\mathbf{A})$ potřebujeme znát $\|\mathbf{A}^{-1}\|$. Avšak výpočet $\mathbf{A}^{-1}$ vyžaduje přibližně třikrát tolik práce jako celé řešení soustavy rovnic. Naštěstí přesnou hodnotu $\kappa(\mathbf{A})$ obvykle nepotřebujeme, vystačíme s dostatečně dobrým odhadem $\kappa(\mathbf{A})$. Spolehlivé a poměrně velmi rychlé odhady čísla podmíněnosti matic patří v současné době ke standardnímu vybavení programů pro řešení soustav lineárních rovnic. Jestliže program zjistí, že číslo podmíněnosti je příliš velké, vydá varování nebo dokonce výpočet přeruší.

Shrnutí. *Soustava lineárních rovnic je dobře (špatně) podmíněná*, právě když je matice soustavy dobře (špatně) podmíněná. Podmíněnost matice soustavy $\mathbf{A}$ měříme pomocí čísla podmíněnosti $\kappa(\mathbf{A}) \geq 1$. Je-li číslo $\kappa(\mathbf{A})$ malé, je matice $\mathbf{A}$ dobře podmíněná. V opačném případě, tj. když $\kappa(\mathbf{A}) \gg 1$, je matice $\mathbf{A}$ špatně podmíněná. Špatně podmíněnou soustavu rovnic lze obvykle jen velmi obtížně řešit. Pomoci může výpočet s vícemístnou mantisou

(je vhodné použít dvojnásobnou nebo ještě větší přesnost). Existují však výjimky: je-li například $\mathbf{A}$ diagonální matice, ve které $a_{ii} = 10^i$, pak je $\kappa(\mathbf{A}) = 10^{n-1}$, což je pro velké n velké číslo, a přesto řešení $x_i = 10^{-i}b_i$ získáme bez problémů pro libovolně velký počet rovnic.

Předpokládejme, že matice soustavy je dobře podmíněná. Pak je GEM s částečným (nebo úplným) výběrem hlavního prvku dobře podmíněný algoritmus: protože velikost multiplikátorů nepřesahuje jedničku, vznikající zaokrouhlovací chyby se dalším výpočtem nezesilují. Když naopak výběr hlavních prvků neprovádíme, můžeme dostat multiplikátory, jejichž absolutní hodnota je větší než jedna, což má za následek zvětšování dříve vzniklých zaokrouhlovacích chyb. GEM bez výběru hlavních prvků je tedy obecně špatně podmíněný algoritmus. Výjimku z tohoto pravidla představuje řešení soustav se speciální maticí soustavy, např. když je matice soustavy ostře diagonálně dominantní nebo pozitivně definitní, pak je i GEM bez výběru hlavních prvků dobře podmíněný algoritmus.

2.2. Iterační metody

Mnoho praktických problémů vyžaduje řešení rozsáhlých soustav lineárních rovnic $\mathbf{Ax} = \mathbf{b}$, v nichž matice $\mathbf{A}$ je naštěstí řídká, tj. má relativně málo nenulových prvků. Standardní eliminační metody studované v předchozí kapitole 2.1 nejsou pro řešení takových soustav vhodné, neboť v průběhu eliminace dochází postupně k zaplňování původně nenulových pozic v matici soustavy, což vede k velkým nárokům na počet aritmetických operací a klade také vysoké nároky na paměť počítače.

To je důvod, proč se pro řešení takových soustav používají *iterační metody*. Zvolí se počáteční vektor $\mathbf{x}_0$ a generuje se posloupnost vektorů $\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2 \ldots$, která konverguje k hledanému řešení $\mathbf{x}$. Společným rysem všech iteračních metod je fakt, že každý jednotlivý iterační krok $\mathbf{x}_k \to \mathbf{x}_{k+1}$ vyžaduje objem výpočtů srovnatelný s násobením matice $\mathbf{A}$ vektorem, což je pro řídké matice objem nevelký (pokud je v každém řádku matice $\mathbf{A}$ řádu n nejvýše m nenulových prvků, jde o nm operací násobení a sčítání). Přijatelný objem výpočtů lze proto dosáhnout i pro poměrně velký počet iterací.

Na obhajobu přímých metod je však třeba dodat, že pro soustavy s řídkými maticemi existují také velmi efektivní algoritmy eliminačního typu. Přesto, pro extrémně rozsáhlé soustavy rovnic se speciální strukturou matice soustavy jsou vhodně zvolené iterační metody efektivnější a jsou často jedinou prakticky realizovatelnou metodou řešení.

Konvergence. Většina klasických iteračních metod vychází z rozkladu matice soustavy $\mathbf{A} = \mathbf{M} - \mathbf{N}$, kde $\mathbf{M}$ je regulární matice. Pak je posloupnost $\mathbf{x}_k$ definována předpisem

$$\mathbf{M}\mathbf{x}_{k+1} = \mathbf{N}\mathbf{x}_k + \mathbf{b}\,, \tag{2.26}$$

přičemž počáteční aproximace $\mathbf{x}_0$ je daná.

Řekneme, že *iterační metoda konverguje*, a píšeme $\mathbf{x}_k \to \mathbf{x}$, když číselná posloupnost $\|\mathbf{x}_k - \mathbf{x}\| \to 0$. Označme $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}$ chybu v k-té iteraci. Protože $\mathbf{Mx} = \mathbf{Nx} + \mathbf{b}$, dostaneme

$$\mathbf{M}(\mathbf{x}_{k+1} - \mathbf{x}) = \mathbf{N}(\mathbf{x}_k - \mathbf{x}) \qquad \text{nebo-li} \qquad \mathbf{e}_{k+1} = \mathbf{M}^{-1}\mathbf{N}\mathbf{e}_k\ .$$

Označíme-li $\mathbf{T} = \mathbf{M}^{-1}\mathbf{N}$, pak pomocí (2.24) dostáváme

$$\|\mathbf{e}_{k+1}\| \le \|\mathbf{T}\| \cdot \|\mathbf{e}_k\| \le \|\mathbf{T}\|^2 \cdot \|\mathbf{e}_{k-1}\| \le \cdots \le \|\mathbf{T}\|^{k+1} \cdot \|\mathbf{e}_0\| \,.$$

Konvergence iterační metody z libovolného startovacího vektoru proto jistě nastane, když

$$\|\mathbf{T}\| < 1 \,, \qquad \text{kde } \mathbf{T} = \mathbf{M}^{-1}\mathbf{N} \text{ je } \textit{iterační matice}. \tag{2.27}$$

Podmínka (2.27) se neověřuje snadno, a proto si u konkrétních metod uvedeme jiné postačující podmínky konvergence.

Kritéria pro ukončení iterací. Jde o to, jak rozhodnout, zda $\mathbf{x}_{k+1}$ je už dostatečně dobrá aproximace řešení $\mathbf{x}$. Řešení $\mathbf{x}$ neznáme, takže se bez něj musíme obejít. Nabízí se zkoumat velikost změny $\mathbf{x}_{k+1} - \mathbf{x}_k$ nebo velikost rezidua $\mathbf{r}_{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}$. Postupuje se tak, že uživatel zadá malé kladné číslo ε jako požadovanou přesnost a v každém kroku metody se testuje, zda je už splněna např. jedna z následujících podmínek

1. $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \le \varepsilon \|\mathbf{x}_k\| \,,$
2. $\|\mathbf{r}_{k+1}\| \le \varepsilon (\|\mathbf{A}\| \cdot \|\mathbf{x}_{k+1}\| + \|\mathbf{b}\|) \,,$
3. $\|\mathbf{r}_{k+1}\| \le \varepsilon \|\mathbf{r}_0\| \,.$

Je-li podmínka na ukončení iterací splněna, výpočet přerušíme a $\mathbf{x}_{k+1}$ považujeme za přibližnou hodnotu řešení $\mathbf{x}$.

Jacobiova metoda. Předpokládejme, že $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, kde $\mathbf{D}$ je diagonální matice, která má stejnou diagonálu jako $\mathbf{A}$, a kde $\mathbf{L}$ resp. $\mathbf{U}$ je ryze dolní resp. horní trojúhelníková část $\mathbf{A}$, tj.

$$\mathbf{D} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix},$$

$$\mathbf{L} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ a_{21} & 0 & & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \qquad \mathbf{U} = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & & 0 & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix}.$$

Nejjednodušší rozklad $\mathbf{A}$ dostaneme pro $\mathbf{M} = \mathbf{D}$ a $\mathbf{N} = -(\mathbf{L} + \mathbf{U})$. Metoda (2.27) je pak tvaru

$$\mathbf{D}\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}_k \tag{2.28}$$

a je známa jako *Jacobiova metoda*. Soustava (2.28) s diagonální maticí se řeší snadno. Zapíšeme-li (2.28) po složkách (složky vektoru $\mathbf{x}_k$ jsou značeny $x_i^{(k)}$, podobně pro $\mathbf{x}_{k+1}$), dostaneme

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \ne i}}^{n} a_{ij}\, x_j^{(k)} \right), \qquad i = 1, 2, \ldots, n \,.$$

Analýzou vlastností iterační matice $\mathbf{T} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$ lze dokázat, že

Jacobiova metoda konverguje, když $\mathbf{A}$ *je ryze diagonálně dominantní.*

Gaussova-Seidelova metoda. Všimněte si, že Jacobiova metoda používá $\mathbf{x}_k$ k výpočtu všech složek $\mathbf{x}_{k+1}$. Protože (alespoň na sériových počítačích) prvky vektoru $\mathbf{x}_{k+1}$ počítáme postupně jeden za druhým, vznikl přirozený nápad využít ihned ty složky $\mathbf{x}_{k+1}$, které jsou už k dispozici. Tak dostáváme *Gaussovu-Seidelovu metodu*:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)} \right), \qquad i = 1, 2, \ldots, n\,.$$

Vyjádříme-li tuto metodu v maticovém tvaru, máme

$$(\mathbf{D} + \mathbf{L})\mathbf{x}_{k+1} = \mathbf{b} - \mathbf{U}\mathbf{x}_k\,.$$

Je dokázáno, že

Gaussova-Seidelova metoda konverguje, když $\mathbf{A}$ *je ryze diagonálně dominantní nebo pozitivně definitní.*

Poznámky. Následuje několik poznatků o vzájemném vztahu Jacobiovy a Gaussovy-Seidelovy metody.

1. Konvergence Gaussovy-Seidelovy metody je pro mnohé matice $\mathbf{A}$ rychlejší než konvergence Jacobiovy metody. Tak je tomu třeba v případě, když $\mathbf{A}$ je ryze diagonálně dominantní.

2. Existují matice, pro které Gaussova-Seidelova metoda konverguje a Jacobiova metoda nekonverguje a naopak, pro které konverguje Jacobiova metoda a Gaussova-Seidelova metoda nekonverguje.

3. Jacobiova metoda umožňuje paralelní výpočet (všechny složky $x_i^{(k+1)}$ mohou být počítány současně, každá na jiném procesoru), zatímco Gaussova-Seidelova metoda je ze své podstaty sekvenční ($x_i^{(k+1)}$ lze vypočítat až po té, co byly spočteny všechny složky $x_j^{(k+1)}$ pro $j < i$). Pro speciální typy matic $\mathbf{A}$ jsou však vypracovány postupy umožňující paralelizovat i Gaussovu-Seidelovu metodu.

Relaxační metody. Bezprostředně poté, co jsme základní metodou (Jacobiovou nebo Gaussovou-Seidelovou) spočetli i-tou složku $x_i^{(k+1)}$, provedeme její modifikaci

$$x_i^{(k+1)} := (1 - \omega)x_i^{(k)} + \omega x_i^{(k+1)},$$

kde $\omega > 0$ je tzv. *relaxační parametr*. Volíme ho tak, abychom vylepšili konvergenci základní metody. Pro $\omega = 1$ dostáváme původní metodu. Zvolíme-li $\omega < 1$, hovoříme o *dolní relaxaci*, v případě $\omega > 1$ jde o *horní relaxaci*. Efektivní volba relaxačního parametru ω závisí na zvolené základní metodě a na matici soustavy $\mathbf{A}$.

Praktické zkušenosti potvrzují, že dolní relaxace může zajistit konvergenci v případě, když základní metoda nekonverguje. Vhodnou volbou relaxačního parametru lze rychlost

konvergence původní metody podstatně zrychlit. Pro zvolenou metodu a speciální tvar matice $\mathbf{A}$ jsou známy vzorce pro optimální hodnotu ω_{opt} relaxačního parametru. Tyto vzorce však mají význam spíše teoretický, neboť výpočet podle nich je příliš náročný. Proto se pracuje s proměnným relaxačním faktorem, v k-té iteraci s ω_k, a jeho hodnota se v každé iteraci zpřesňuje tak, aby se postupně blížila k optimálnímu ω_{opt}. Konkrétní metody lze najít ve specializované literatuře.

Relaxace Jacobiovy metody. Dá se ukázat, že když konverguje Jacobiova metoda, tak konverguje také relaxovaná Jacobiova metoda pro $0 < \omega \leq 1$.

Relaxace Gaussovy-Seidelovy metody je v literatuře známa jako *SOR metoda* (podle anglického Successive Over Relaxation). O konvergenci SOR metody máme zejména následující poznatky:

1. Pokud SOR metoda konverguje, pak je $0 < \omega < 2$.
2. SOR metoda konverguje, když $\mathbf{A}$ je ryze diagonálně dominantní a $0 < \omega \leq 1$.
3. SOR metoda konverguje, když $\mathbf{A}$ je pozitivně definitní a $0 < \omega < 2$.

SOR metoda zapsaná po složkách je tvaru

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)} \right) + (1-\omega) x_i^{(k)}\,, \qquad i = 1, 2, \ldots, n\,,$$

a SOR maticově je

$$(\omega^{-1}\mathbf{D} + \mathbf{L})\mathbf{x}_{k+1} = \mathbf{b} - ((1-\omega^{-1})\mathbf{D} + \mathbf{U})\mathbf{x}_k.$$

Když ve složkovém zápisu SOR metody na pravé straně místo $x_j^{(k+1)}$ píšeme $x_j^{(k)}$, dostaneme relaxaci Jacobiovy metody označovanou jako JOR metoda, maticově

$$\omega^{-1}\mathbf{D}\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{L} + (1-\omega^{-1})\mathbf{D} + \mathbf{U})\mathbf{x}_k\,.$$

Příklad 2.5. Velké řídké matice vznikají při numerickém řešení parciálních diferenciálních rovnic. MATLAB má ve své galerii příklady takových matic. Příkazem

```
K = gallery('poisson',n)
```

vygenerujeme matici, jejíž strukturu nyní popíšeme. [1]

[1] Parciální diferenciální rovnice jsou nezbytné pro modelování technických problémů. Při řešení Dirichletovy úlohy pro Poissonovu rovnici na čtverci $\Omega = (0, l) \times (0, l)$,

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y) \quad \text{v } \Omega \quad \text{a} \quad u(x,y) = g(x,y) \text{ na hranici } \partial\Omega\,,$$

(funkce f a g jsou dané, neznámá je funkce u) metodou sítí vzniká soustava lineárních rovnic s maticí soustavy $\mathbf{K}$ uvažovanou v tomto příkladu 2.5.

Matice $\mathbf{K}$ je blokově třídiagonální, pro devět rovnic vypadá takto

$$\left(\begin{array}{rrr|rrr|rrr} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{array}\right).$$

Obecně je $\mathbf{K}$ složena z čtvercových submatic řádu n (tzv. bloků) $\mathbf{B}$, $-\mathbf{I}$, $\mathbf{O}$, které jsou ve čtvercové matici řádu n^2 rozmístěny ve třech diagonálách

$$\mathbf{K} = \left(\begin{array}{c|c|c|c|c|c} \mathbf{B} & -\mathbf{I} & \mathbf{O} & \cdots & \mathbf{O} & \mathbf{O} \\ \hline -\mathbf{I} & \mathbf{B} & -\mathbf{I} & \cdots & \mathbf{O} & \mathbf{O} \\ \hline \mathbf{O} & -\mathbf{I} & \mathbf{B} & \cdots & \mathbf{O} & \mathbf{O} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline \mathbf{O} & \mathbf{O} & \mathbf{O} & \cdots & \mathbf{B} & -\mathbf{I} \\ \hline \mathbf{O} & \mathbf{O} & \mathbf{O} & \cdots & -\mathbf{I} & \mathbf{B} \end{array}\right).$$

Matice $\mathbf{I}$ je jednotková matice, $\mathbf{O}$ je čtvercová matice s nulovými prvky a $\mathbf{B}$ je třídiagonální matice

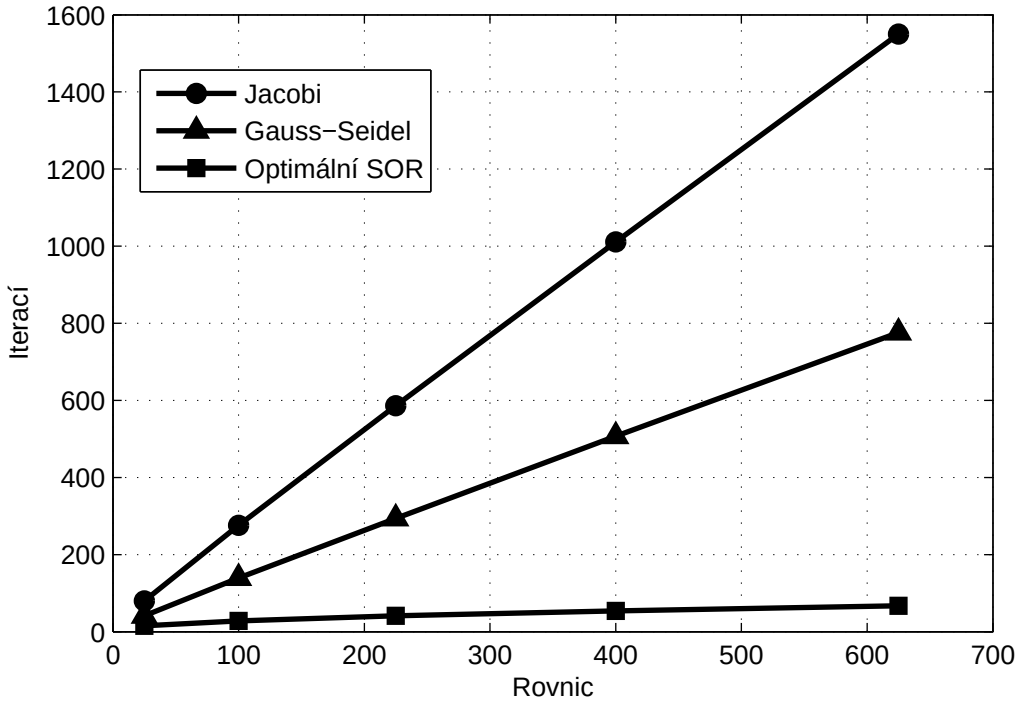
$$\mathbf{B} = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 4 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 4 \end{pmatrix}.$$

Na matici $\mathbf{K}$ se často testuje účinnost numerických metod pro řešení soustav lineárních rovnic s řídkou maticí. Na obrázku 2.3 je vidět závislost počtu iterací (potřebných pro dosažení zvolené přesnosti) na počtu rovnic n^2. Při metodě SOR bylo zvoleno optimální ω.

Poznámka. Iterační metody, se kterými jsme se seznámili, bývají označovány jako *klasické iterační metody*. V současnosti se používají už jen zřídka. Do učebního textu jsme je zařadili hlavně proto, že jsou poměrně jednoduché a přitom na nich lze dobře ukázat, jak iterační metody fungují.

Na druhé straně metody, které se skutečně používají, jsou poměrně složité k pochopení, takže je v tomto základním kurzu nelze dost dobře vysvětlit. Spokojíme se tedy s konstatováním, že existuje značné množství výkonných iteračních metod, viz např. [13]. Tak třeba pro soustavy s pozitivně definitní maticí patří mezi nejpopulárnější *metoda sdružených gradientů* (stručně CG podle anglického *conjugate gradient*), v MATLABu

viz funkce `pcg`. Základní verze této metody je stručně popsána v kapitole 6.2, viz také cvičení 6.7. Pro soustavy s nesymetrickou maticí je situace komplikovanější, jednoznačný favorit mezi metodami pro jejich řešení neexistuje. Jednou z mnoha používaných metod je *zobecněná metoda minimálních reziduí* (stručně GMRES podle anglického *generalized minimal residual*), v MATLABu viz funkce `gmres`.



Obr. 2.3: Srovnání klasických iteračních metod

2.3. Cvičení

2.1. Metodou GEMz a GEM s částečným výběrem hlavního prvku (GEMpp) řešte rovnice

$$a)\quad \begin{array}{rcrcrcl} x & + & 2y & + & 3z & = & 6 \\ 2x & + & 4y & + & 5z & = & 11 \\ 7x & + & 8y & + & 9y & = & 24 \end{array} \qquad b)\quad \begin{array}{rcrcl} 10^{-15}x & + & y & = & 1+10^{-15} \\ x & + & 10^{11}y & = & 1+10^{11} \end{array}$$

[a) GEMz selže, GEMpp najde řešení $(1\ 1\ 1)^T$; b) GEMz spočte $(0{,}888\ 1)^T$, GEMpp dává $(1\ 1)^T$.]

2.2. Odvoďte (pro $n = 3$) vzorce (2.13).

$$[\mathbf{L}\mathbf{L}^T = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix} = \begin{pmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{pmatrix},$$

$\mathbf{L}\mathbf{L}^T = \mathbf{A} \quad \Rightarrow \quad l_{11} = \sqrt{a_{11}}, \quad l_{21} = a_{21}/l_{11}, \quad l_{31} = a_{31}/l_{11}, \quad l_{22} = \sqrt{a_{22} - l_{21}^2}, \quad \ldots]$

2.3. Pomocí Choleského rozkladu řešte soustavy rovnic $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$, $i = 1, 2, 3$, kde

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 3 \end{pmatrix}, \quad \mathbf{b}_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad \mathbf{b}_2 = \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix}.$$

$$[\ \mathbf{L} \doteq \begin{pmatrix} 1{,}414214 & 0 & 0 \\ 0{,}707107 & 1{,}224745 & 0 \\ 1{,}414214 & 0{,}816497 & 0{,}577350 \end{pmatrix}, \quad \mathbf{x}_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.\ ]$$

2.4. Řešte soustavy rovnic $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$, $i = 1, 2, 3$, kde

$$\mathbf{A} = \begin{pmatrix} 1 & -1 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & 2 \end{pmatrix}, \quad \mathbf{b}_1 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{b}_2 = \begin{pmatrix} 1 \\ 2 \\ 7 \end{pmatrix}, \quad \mathbf{b}_3 = 2\mathbf{x}_2 - \mathbf{x}_1.$$

[Proveďte LU rozklad $\mathbf{PA} = \mathbf{LU}$, kde $\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ 0{,}5 & 1 & 0 \\ 0{,}5 & -0{,}\overline{33} & 1 \end{pmatrix}$, $\mathbf{U} = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 1{,}5 & 1{,}5 \\ 0 & 0 & 1 \end{pmatrix}$,
$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$. Nejdříve získáme $\mathbf{x}_1$, $\mathbf{x}_2$, sestavíme $\mathbf{b}_3$ a opět LU rozkladem spočteme $\mathbf{x}_3$:
$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$, $\mathbf{x}_2 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}$, $\mathbf{b}_3 = \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}$, $\mathbf{x}_3 = \begin{pmatrix} 1 \\ 1{,}\overline{66} \\ 0{,}\overline{66} \end{pmatrix}$.]

2.5. Spočtěte číslo podmíněnosti matice soustavy $\mathbf{A}$ ze cvičení 2.1 b).
[$\kappa(\mathbf{A}) \doteq 10^{22}$ v normě $\|\cdot\|_\infty$.]

2.6. Spočtěte determinant matice $\mathbf{A}$ ze cvičení 2.4 použitím LU rozkladu.

[$|\mathbf{A}| = (-1)^2|\mathbf{U}| = 2 \cdot 1{,}5 \cdot 1 = 3$.]

2.7. Spočtěte inverzní matici k matici $\mathbf{A}$ ze cvičení 2.4.

$$[\ \mathbf{A}^{-1} = \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0{,}\overline{33} & 0{,}\overline{33} \\ 1 & -0{,}\overline{66} & 0{,}\overline{33} \end{pmatrix}.]$$

2.8. Naprogramujte metodu GEMz a upravte ji tak, aby bylo možné volitelně zapnout/vypnout částečný výběr hlavního prvku (GEMpp).

2.9. Naprogramujte výpočet vektorových a maticových norem.

2.10. Naprogramujte řešení soustav s třídiagonální maticí. Prvky matice držte v paměti počítače v úsporném formátu např. jako tři vektory (jednorozměrná pole). Jak vypadá transformovaná matice $\widetilde{\mathbf{A}} := \mathbf{LU}$ a matice $\mathbf{L}$ a $\mathbf{U}$ získané LU rozkladem matice

$$\mathbf{A} = \begin{pmatrix} 100 & 12 & 0 & 0 \\ 21 & 100 & 23 & 0 \\ 0 & 32 & 100 & 34 \\ 0 & 0 & 43 & 100 \end{pmatrix}?$$

$$[\ \widetilde{\mathbf{A}} \doteq \begin{pmatrix} 100 & 12 & 0 & 0 \\ 0{,}21 & 97{,}48 & 23 & 0 \\ 0 & 0{,}3283 & 92{,}4497 & 34 \\ 0 & 0 & 0{,}4651 & 84{,}186 \end{pmatrix},$$

$$\mathbf{L} \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0{,}21 & 1 & 0 & 0 \\ 0 & 0{,}3283 & 1 & 0 \\ 0 & 0 & 0{,}4651 & 1 \end{pmatrix}, \quad \mathbf{U} \doteq \begin{pmatrix} 100 & 12 & 0 & 0 \\ 0 & 97{,}48 & 23 & 0 \\ 0 & 0 & 92{,}4497 & 34 \\ 0 & 0 & 0 & 84{,}186 \end{pmatrix}.]$$

2.11. Naprogramujte (a) Jacobiovu a (b) Gauss-Seidelovu iterační metodu. Jak vypadá vektor $\mathbf{x}_5$ pro soustavu rovnic

$$\begin{pmatrix} -1 & 9 & 4 \\ -4 & -4 & 15 \\ 33 & -3 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 29 \\ 33 \\ 24 \end{pmatrix},$$

když nejdříve rovnice soustavy vhodně přerovnáte (tak, aby matice soustavy byla ryze diagonálně dominantní) a když jako počáteční aproximaci zvolíte vektor $\mathbf{x}_0 = (0,0,0)^T$?

[(a) $\mathbf{x}_5 \doteq (0{,}995405; 2{,}007110; 2{,}986472)^T$, (b) $\mathbf{x}_5 \doteq (1{,}000144; 1{,}999887; 3{,}000008)^T$.]

2.12. Ověřte, že pro soustavu rovnic

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 6 \end{pmatrix}$$

platí: (a) matici soustavy nelze žádným přerovnáním řádků upravit tak, aby byla ryze diagonálně dominantní; (b) Jacobiova metoda nekonverguje (použijte počítač); (c) matice soustavy je pozitivně definitní (použijte Sylvesterovo kriterium); (d) Gaussova-Seidelova metoda konverguje (použijte počítač).

2.13. Ověřte, že pro soustavu rovnic

$$\begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ 7 \end{pmatrix}$$

platí: (a) matici soustavy nelze žádným přerovnáním řádků upravit tak, aby byla ryze diagonálně dominantní; (b) Jacobiova metoda nekonverguje (použijte počítač); (c) matice soustavy není pozitivně definitní (použijte Sylvesterovo kriterium); (d) Gaussova-Seidelova metoda nekonverguje (použijte počítač).

2.14. Upravte soustavu ze cvičení 2.13 na tvar $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ s pozitivně definitní maticí soustavy. Jako počáteční aproximaci zvolte vektor $\mathbf{x}_0 = (0,0,0)^T$ a řešte Gaussovou-Seidelovou metodou. Použijte počítač a pozorujte pomalou konvergenci k řešení. Kolik iterací je třeba k dosažení přesnosti $\varepsilon = 10^{-6}$, použijeme-li k ukončení výpočtu podmínku $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_\infty \le \varepsilon\|\mathbf{x}_k\|_\infty$? Konverguje Jacobiova metoda?

[$\mathbf{A}^T\mathbf{A} = \begin{pmatrix} 6 & 7 & 10 \\ 7 & 9 & 12 \\ 10 & 12 & 17 \end{pmatrix}$, $\mathbf{A}^T\mathbf{b} = \begin{pmatrix} 23 \\ 28 \\ 39 \end{pmatrix}$, $k+1 = 681$. Jacobiova metoda nekonverguje.]

2.15. Naprogramujte Jacobiovu relaxační metodu (stručně JOR) a Gaussovu-Seidelovu relaxační metodu SOR. Pracujte se soustavami

$$(i) \quad \begin{pmatrix} 10 & 3 \\ 2 & 20 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10 \\ 2 \end{pmatrix}, \qquad (ii) \quad \begin{pmatrix} 3 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix},$$

počáteční iteraci zvolte $\mathbf{x}_0 = (0,0)^T$. (a) Jak vypadají vektory $\mathbf{x}_5$ vznikající při řešení soustavy (i) metodou JOR a SOR, když použijeme relaxační parametr $\omega = 0{,}5$? (b) Experimentálně najděte přibližnou optimální hodnotu ω_{opt} pro metodu SOR a soustavu (ii). Pro nalezené ω_{opt}, a také pro $\omega = 0{,}5$, určete nejmenší počet kroků, při kterém je splněna podmínka $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_\infty \le 10^{-6}\|\mathbf{x}_k\|_\infty$. Návod: pomocí programu měňte ω od 1 s krokem 0,01 do 1,99 a určete případ, kdy vyjde k nejmenší.

[(a) JOR: $\mathbf{x}_5 \doteq (0{,}9592; 0{,}0166)^T$, SOR: $\mathbf{x}_5 \doteq (0{,}9640; 0{,}0083)^T$;
(b) pro $\omega_{opt} \doteq 1{,}15$ je $k+1 = 10$, pro $\omega = 0{,}5$ je $k+1 = 50$.]

3. Aproximace funkcí

Aproximovat funkci $f(x)$ znamená nahradit ji funkcí $\varphi(x)$, která je k $f(x)$ v jistém smyslu blízká. Píšeme $\varphi(x) \approx f(x)$. Budeme se zabývat dvěma základními typy aproximace, a to interpolací a metodou nejmenších čtverců.

Interpolace je taková aproximace, při níž $\varphi(x)$ nabývá v zadaných bodech x_i předepsaných hodnot $y_i = f(x_i)$. Někdy navíc žádáme, aby funkce φ a f měly v bodech x_i také stejné derivace. Interpolaci je věnován odstavec 3.1.

Metoda nejmenších čtverců je taková aproximace, při níž $\varphi(x)$ prokládáme mezi zadanými body $[x_i, y_i]$ tak, aby vzdálenost funkcí f a φ byla v jistém smyslu minimální. Je přitom charakteristické, že funkce φ body $[x_i, y_i]$ neprochází. Metoda nejmenších čtverců je vyložena v odstavci 3.2.

Aproximaci $\varphi(x)$ použijeme k přibližnému výpočtu hodnot funkce $f(x)$, třeba při vykreslování $\varphi \approx f$. Je žádoucí, aby výpočet $\varphi(x)$ byl jednoduchý. Proto se φ často hledá ve tvaru polynomu.

Obecně, $\varphi(x)$ se používá k řešení úloh, v nichž vystupuje funkce f, kterou je účelné nebo dokonce nezbytné nahradit její vhodnou aproximací φ. Jako příklad uveďme výpočet derivace nebo určitého integrálu: $f'(x)$ nahradíme pomocí $\varphi'(x)$ a $\int_a^b f(x)\,\mathrm{d}x$ nahradíme pomocí $\int_a^b \varphi(x)\,\mathrm{d}x$.

3.1. Interpolace

Interpolační funkci $\varphi(x)$ vybíráme z vhodné třídy funkcí. Omezíme se na dva nejběžnější případy:

a) $\varphi(x)$ je polynom;

b) $\varphi(x)$ je po částech polynom, na každém subintervalu obecně jiný.

3.1.1. Interpolace polynomem

Předpokládejme, že jsou dány navzájem různé body

$$x_0, x_1, \ldots, x_n\,, \qquad x_i \neq x_j \quad \text{pro} \quad i \neq j\,,$$

říkáme jim také *uzly interpolace*, a v každém z nich je předepsána hodnota y_i. Hledáme *interpolační polynom* $P_n(x)$ stupně nejvýše n, který splňuje *interpolační podmínky*

$$P_n(x_i) = y_i\,, \qquad i = 0, 1, \ldots, n\,. \tag{3.1}$$

Existenci interpolačního polynomu dokážeme tak, že ho zkonstruujeme.

Lagrangeův tvar interpolačního polynomu má vyjádření

$$P_n(x) = y_0\ell_0(x) + y_1\ell_1(x) + \cdots + y_n\ell_n(x) = \sum_{i=0}^{n} y_i\ell_i(x)\,, \tag{3.2}$$

kde $\ell_i(x)$ jsou tzv. *fundamentální polynomy* definované předpisem

$$\ell_i(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}\,. \tag{3.3}$$

Snadno nahlédneme, že

$$\ell_i(x_k) = \begin{cases} 1 & \text{pro } k=i, \\ 0 & \text{pro } k\neq i, \end{cases} \qquad i,k = 0,1,\dots,n\,, \tag{3.4}$$

takže interpolační podmínky $P_n(x_k) = \sum_{i=0}^{n} y_i\ell_i(x_k) = y_k$, $k = 0,1,\dots,n$, jsou splněny.

Interpolační polynom je daty $[x_i, y_i]$, $i = 0,1,\dots,n$, určen jednoznačně. Skutečně, jsou-li P a Q interpolační polynomy splňující interpolační podmínky $P(x_i) = Q(x_i) = y_i$, pak polynom $P-Q$ je roven nule v uzlech $x_0, x_1, \dots, x_n$. Avšak polynom stupně nejvýše n nemůže mít více než n kořenů, pokud se nerovná identicky nule. V našem případě proto nutně $P-Q=0$ a tedy $P=Q$. Tím je jednoznačnost interpolačního polynomu dokázána.

Příklad 3.1. Určíme interpolační polynom pro data předepsaná tabulkou

x_i	-1	1	2	3
y_i	-6	-2	-3	2

Nejdříve získáme fundamentální polynomy

$$\begin{aligned}
\ell_0(x) &= \frac{(x-1)(x-2)(x-3)}{(-1-1)(-1-2)(-1-3)} = -\frac{1}{24}\,(x^3-6x^2+11x-6)\,,\\
\ell_1(x) &= \frac{(x+1)(x-2)(x-3)}{(1+1)(1-2)(1-3)} = \frac{1}{4}\,(x^3-4x^2+x+6)\,,\\
\ell_2(x) &= \frac{(x+1)(x-1)(x-3)}{(2+1)(2-1)(2-3)} = -\frac{1}{3}\,(x^3-3x^2-x+3)\,,\\
\ell_3(x) &= \frac{(x+1)(x-1)(x-2)}{(3+1)(3-1)(3-2)} = \frac{1}{8}\,(x^3-2x^2-x+2)
\end{aligned}$$

a pak sestavíme interpolační polynom

$$P_3(x) = -6\cdot\ell_0(x) - 2\cdot\ell_1(x) - 3\cdot\ell_2(x) + 2\cdot\ell_3(x) = x^3-3x^2+x-1\,. \qquad \square$$

Hlavní předností Lagrangeova tvaru interpolačního polynomu je jeho elegantní forma. Používá se proto zejména v teoretických úvahách. Pro praktické použití však ideální není. Upozorněme na dva jeho hlavní nedostatky.

(a) Přidáme-li další uzel x_{n+1}, musíme přepočítat všechny fundamentální polynomy.

(b) Počet operací potřebných k výpočtu hodnoty $P_n(\bar{x})$ je poměrně značný, vyžaduje $O(n^2)$ operací.

Oba tyto nedostatky odstraňuje efektivnější zápis Lagrangeova interpolačního polynomu, viz cvičení 3.1, formule (3.36), nebo barycentrická interpolační formule (3.37), více viz [2]. Další možností je použít

Newtonův tvar interpolačního polynomu

$$P_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \cdots + a_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})\,. \quad (3.5)$$

Přidání dalšího uzlu x_{n+1} je snadné, k $P_n(x)$ stačí přičíst jeden další člen, neboť

$$P_{n+1}(x) = P_n(x) + a_{n+1}(x-x_0)(x-x_1)\cdots(x-x_n)\,.$$

Nedostatek (a) Lagrangeova tvaru interpolačního polynomu jsme tedy překonali. Hodnotu $z = P_n(\bar{x})$ určíme podobně jako v *Hornerově schématu*, tj. postupem:

$$z := a_n \text{ a pak pro } i = n-1, n-2, \ldots, 0 \text{ počítej } z := z(\bar{x} - x_i) + a_i\,. \quad (3.6)$$

Koeficienty a_i lze vypočítat přímo z interpolačních podmínek (3.1) a dosáhnout tak významné úspory v počtu operací. Existuje však ještě lepší způsob a ten si teď uvedeme.

Nejdříve definujeme *poměrné diference*:

$$\begin{aligned}
&P[x_i] := y_i\,,\\
&P[x_{i-1}, x_i] := (P[x_i] - P[x_{i-1}])/(x_i - x_{i-1})\,,\\
&P[x_{i-2}, x_{i-1}, x_i] := (P[x_{i-1}, x_i] - P[x_{i-2}, x_{i-1}])/(x_i - x_{i-2}), \qquad (3.7)\\
&\text{a dále, pro } 3 \le k \le n\text{:}\\
&P[x_{i-k}, x_{i-k+1}, \ldots, x_{i-1}, x_i] := (P[x_{i-k+1}, \ldots, x_i] - P[x_{i-k}, \ldots, x_{i-1}])/(x_i - x_{i-k})\,.
\end{aligned}$$

Dá se ukázat, že

$$a_i = P[x_0, x_1, \ldots, x_i]\,, \quad (3.8)$$

takže *Newtonův tvar interpolačního polynomu* je

$$\begin{aligned}
P_n(x) = {} & P[x_0] + P[x_0, x_1](x-x_0) + P[x_0, x_1, x_2](x-x_0)(x-x_1) + \ldots\\
& + P[x_0, x_1, \ldots, x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})\,.
\end{aligned} \quad (3.9)$$

Označíme-li $P_{ik} = P[x_{i-k}, \ldots, x_i]$, pak $a_i = P_{ii}$. Následuje

algoritmus výpočtu poměrných diferencí

Pro $i = 0, 1, \ldots, n$ prováděj:

  $P_{i0} := y_i$

  Pro $k = 1, 2, \ldots, i$ proveď:

    $P_{ik} := (P_{i,k-1} - P_{i-1,k-1})/(x_i - x_{i-k})$

  konec cyklu k,

konec cyklu i.

$$
\begin{array}{c|ccccccc}
x_0 & P_{00} & & & & & \\
x_1 & P_{10} & P_{11} & & & & \\
x_2 & P_{20} & P_{21} & P_{22} & & & \\
x_3 & P_{30} & P_{31} & P_{32} & P_{33} & & \\
\vdots & \vdots & \vdots & \vdots & & \ddots & \\
x_n & P_{n0} & P_{n1} & P_{n2} & \dots & P_{n,n-1} & P_{nn}
\end{array}
$$

Výpočet lze přehledně zaznamenat do tabulky, kterou vyplňujeme po řádcích. Výpočet koeficientů $a_i = P_{ii}$ vyžaduje $O(n^2)$ operací, avšak výpočet $P_n(\bar{x})$ podle (3.6) vyžaduje jen $O(n)$ operací. Dosáhli jsme tedy řádově nižšího počtu operací, než kolik je jich třeba pro výpočet $P_n(\bar{x})$ z Lagrangeova interpolačního polynomu. To znamená, že také nedostatek (b) Lagrangeova interpolačního polynomu je uspokojivě vyřešen.

Příklad 3.2. Sestavíme Newtonův interpolační polynom pro data z příkladu 3.1. Průběh výpočtu zaznamenáváme do tabulky. Dostaneme

$$
\begin{array}{r|cccc}
x_i & P_{i0} & P_{i1} & P_{i2} & P_{i3} \\
\hline
-1 & -6 & & & \\
1 & -2 & 2 & & \\
2 & -3 & -1 & -1 & \\
3 & 2 & 5 & 3 & 1
\end{array}
\qquad
\begin{array}{lcl}
\Longrightarrow & a_0 & = -6 \\
\Longrightarrow & a_1 & = 2 \\
\Longrightarrow & a_2 & = -1 \\
\Longrightarrow & a_3 & = 1\,,
\end{array}
$$

takže $P_3(x) = -6 + 2 \cdot (x+1) + (-1) \cdot (x+1)(x-1) + 1 \cdot (x+1)(x-1)(x-2)$.
V bodě $\bar{x} = 0{,}5$ podle (3.6) vypočteme

$$P_3(0{,}5) = ((1 \cdot (0{,}5-2) - 1) \cdot (0{,}5-1) + 2) \cdot (0{,}5+1) - 6 = -1{,}125\,.$$

Když přidáme další uzel $x_4 = 0$ a v něm předepíšeme hodnotu $y_4 = 2$, stačí dopočítat jeden řádek tabulky. Dostaneme

$$
\begin{array}{r|ccccc}
x_4 & P_{40} & P_{41} & P_{42} & P_{43} & P_{44} \\
\hline
0 & 2 & 0 & 2{,}5 & 0{,}5 & -0{,}5
\end{array}
\qquad \Longrightarrow \quad a_4 = -0{,}5\,,
$$

a tedy $P_4(x) = P_3(x) + (-0{,}5) \cdot (x+1)(x-1)(x-2)(x-3)$. □

Chyba aproximace interpolačním polynomem. Nejdříve zavedeme následující

Označení. Symbolem $C\langle a,b\rangle$ budeme značit množinu všech funkcí, které jsou v intervalu $\langle a,b\rangle$ spojité, a symbolem $C^k\langle a,b\rangle$ pak množinu všech funkcí, které jsou v intervalu $\langle a,b\rangle$ spojité spolu se svými derivacemi až do řádu k včetně. Pro $k=0$ zřejmě $C^0\langle a,b\rangle \equiv C\langle a,b\rangle$.

Předpokládejme, že čísla y_i nejsou libovolná, ale že $y_i = f(x_i)$ jsou hodnoty funkce f v uzlech interpolace. Pak nás jistě bude zajímat chyba

$$E_n(\bar{x}) := f(\bar{x}) - P_n(\bar{x})$$

ve zvoleném bodě $\bar{x}$. Pro $\bar{x} = x_i$ je $E_n(x_i) = 0$. Jaká je ale chyba mimo uzly interpolace?

Nechť tedy $\bar{x}$ je libovolný bod, $\langle a, b\rangle$ je nějaký interval obsahující všechny uzly x_i interpolace a také zkoumaný bod $\bar{x}$, a nechť $f \in C^{n+1}\langle a, b\rangle$. Pak pro chybu $E_n(\bar{x})$ platí

$$E_n(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(\bar{x} - x_0)(\bar{x} - x_1)\dots(\bar{x} - x_n)\,, \tag{3.10}$$

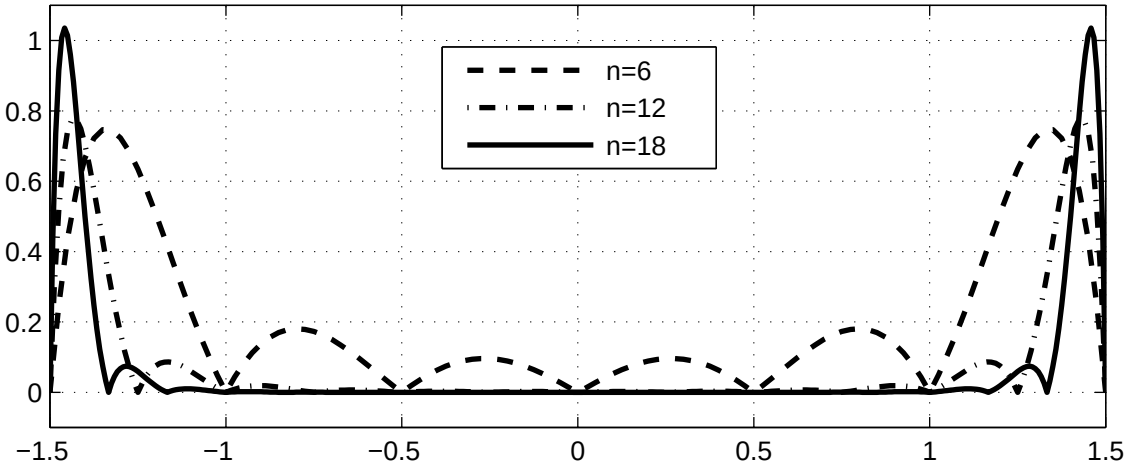
kde $\xi = \xi(\bar{x})$ je (blíže neurčený) bod z intervalu $\langle a, b\rangle$. Zápisem $\xi = \xi(\bar{x})$ přitom chceme zdůraznit, že poloha bodu ξ závisí nejen na funkci f a na interpolantu P_n, ale také na zvoleném bodu $\bar{x}$.

Poznámky. Abychom zjednodušili výklad, budeme předpokládat, že $x_0 < x_1 < \cdots < x_n$.

1) Jestliže M_{n+1} je taková konstanta, že $|f^{(n+1)}(x)| \le M_{n+1}$ pro každé $x \in \langle a, b\rangle$, pak

$$|E_n(\bar{x})| \le \frac{M_{n+1}}{(n+1)!} \max_{x\in\langle a,\,b\rangle} |\omega_{n+1}(x)|\,, \tag{3.11}$$

kde $\omega_{n+1}(x) = (x - x_0)(x - x_1)\dots(x - x_n)$. Odhad (3.11) je však obvykle příliš pesimistický.



Obr. 3.1: Graf funkce $|\omega_{n+1}(x)|$

2) Jestliže má funkce $f(x)$ derivace všech řádů ohraničené stejnou konstantou, pak pro dostatečně velké n je chyba libovolně malá.

Příklad 3.3. Pro $f(x) = \sin x$ lze vzít $M_{n+1} = 1$, proto

$$|E_n(x)| \le \frac{(b-a)^{n+1}}{(n+1)!}\,. \quad \text{Dá se dokázat, že} \quad \frac{(b-a)^{n+1}}{(n+1)!} \to 0 \quad \text{pro} \quad n \to \infty\,,$$

takže $P_n(x) \to f(x)$ pro každé x z libovolného konečného intervalu $\langle a, b\rangle$. □

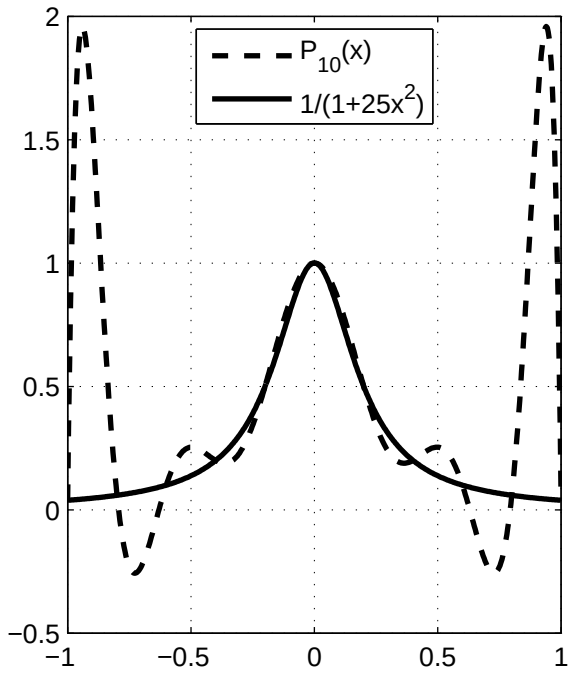
3) Jestliže interpolační polynom používáme k výpočtu hodnot interpolované funkce vně intervalu $\langle x_0, x_n\rangle$, říkáme, že provádíme *extrapolaci*. V tomto případě může být chyba aproximace velká, neboť hodnota $|\omega_{n+1}(x)|$ rychle roste, když se x vzdaluje od x_0 doleva nebo od x_n doprava.

4) $\omega_{n+1}(x)$ může nabývat velkých hodnot také uvnitř intervalu $\langle x_0, x_n\rangle$, zejména když jsou uzly x_i rozmístěny rovnoměrně, tj. když $x_i = x_0 + ih$, kde h je pevně zvolený krok. Na obr. 3.1 vidíme, že ve středu intervalu $\langle x_0, x_n\rangle$ nabývá $|\omega_{n+1}(x)|$ nejmenších hodnot, v blízkosti středů krajních intervalů, zejména intervalů $\langle x_0, x_1\rangle$ a $\langle x_{n-1}, x_n\rangle$, je však hodnota $|\omega_{n+1}(x)|$ značná. Toto chování polynomu $\omega_{n+1}(x)$ se promítne i do průběhu interpolantu $P_n(x)$.

Příklad 3.4. Sestrojíme interpolační polynom Rungeovy funkce

$$f(x) = \frac{1}{1 + 25x^2}$$

na rovnoměrném dělení intervalu $\langle -1, 1\rangle$.



Obr. 3.2: Interpolační polynom Rungeovy funkce

Jde o známý příklad, na kterém se demonstruje, že pro rostoucí počet dílků chyba interpolace neomezeně roste. □

Vhodným rozmístěním uzlů lze chybu $|\omega_{n+1}(x)|$ minimalizovat, viz cvičení 3.6. Tuto možnost však obvykle nemáme, neboť uzly jsou pevně dány. Proto *používání interpolačních polynomů vysokých stupňů obecně nelze doporučit.*

Hermitova interpolace. Doposud jsme se zabývali *Lagrangeovou interpolací.* Jejím charakteristickým rysem je to, že interpolační polynom $P_n(x)$ je určen zadanými hodnotami $P_n(x_i) = y_i$ v uzlech x_i. Pokud interpolační polynom určují navíc také předepsané derivace, hovoříme o *Hermitově interpolaci.*

Předpokládejme tedy, že v každém uzlu x_i je zadáno $\alpha_i + 1$ čísel $y_i^{(0)}, y_i^{(1)}, \ldots, y_i^{(\alpha_i)}$. Označme $\alpha = n + \sum_{i=0}^{n} \alpha_i$. Pak *Hermitovým interpolačním polynomem* $P_\alpha(x)$ nazveme polynom stupně nejvýše α, který splňuje interpolační podmínky

$$\frac{\mathrm{d}^j}{\mathrm{d}x^j} P_\alpha(x_i) = y_i^{(j)}\,, \qquad j = 0, 1, \ldots, \alpha_i\,, \quad i = 0, 1, \ldots, n\,. \tag{3.12}$$

Nultou derivací přitom rozumíme funkční hodnotu. Je dokázáno, že existuje jediný takový polynom. Jestliže

$$y_i^{(j)} = \frac{\mathrm{d}^j}{\mathrm{d}x^j} f(x_i)\,, \qquad j = 0, 1, \ldots, \alpha_i\,, \quad i = 0, 1, \ldots, n\,,$$

říkáme, že $P_\alpha(x)$ je Hermitův interpolační polynom funkce $f(x)$.

Nechť $\langle a, b\rangle$ je interval obsahující uzly interpolace. Jestliže $f \in C^{\alpha+1}\langle a, b\rangle$, pak pro chybu Hermitovy interpolace v bodě $\bar{x} \in \langle a, b\rangle$ platí

$$f(\bar{x}) - P_\alpha(\bar{x}) = \frac{f^{(\alpha+1)}(\xi)}{(\alpha+1)!}(\bar{x} - x_0)^{\alpha_0+1}(\bar{x} - x_1)^{\alpha_1+1} \ldots (\bar{x} - x_n)^{\alpha_n+1}\,, \tag{3.13}$$

kde $\xi = \xi(\bar{x})$ je (nějaký blíže neurčený) bod z intervalu $\langle a, b\rangle$.

Použití Hermitova polynomu vysokého stupně obecně nelze doporučit, protože (stejně jako u Lagrangeovy interpolace) může být chyba interpolace mezi uzly značná.

Vzorec pro určení koeficientů Hermitova interpolačního polynomu je poměrně komplikovaný, lze ho najít např. v [22], zde ho neuvádíme. Místo toho si na příkladu ukážeme, jak lze Hermitův polynom určit přímo z interpolačních podmínek.

Příklad 3.5. Sestrojíme Hermitův interpolační polynom pro data podle tabulky

x_i	y_i	y_i'	y_i''
-1	2	-4	12
1	2	4	

Tedy $x_0 = -1$, $\alpha_0 = 2$, $y_0^{(0)} = 2$, $y_0^{(1)} = -4$, $y_0^{(2)} = 12$,
$x_1 = 1$, $\alpha_1 = 1$, $y_1^{(0)} = 2$, $y_1^{(1)} = 4$.

Protože je předepsáno celkem 5 podmínek, Hermitův polynom navrhneme jako polynom stupně $\alpha = 4$. Abychom si ušetřili práci, zapíšeme ho ve tvaru mocninného rozvoje okolo toho bodu, v němž je předepsán největší počet podmínek, v našem případě tedy okolo $x_0 = -1$. Pak

$$P_4(x) = a + b(x+1) + c(x+1)^2 + d(x+1)^3 + e(x+1)^4\,.$$

Koeficienty a, b, c získáme snadno. Z podmínky $P_4(-1) = 2$ okamžitě dostaneme $a = 2$. Podobně z podmínky $P_4'(-1) = -4$ obdržíme $b = -4$, a protože $P_4''(-1) = 2c$, z podmínky $P_4''(-1) = 12$ dostaneme $c = 6$. Dále

$$\begin{aligned} P_4(1) &= 2 - 4 \cdot 2 + 6 \cdot 2^2 + d \cdot 2^3 + e \cdot 2^4 = 2 \qquad &\Longrightarrow \qquad 8d + 16e &= -16\,, \\ P_4'(1) &= -4 + 2 \cdot 6 \cdot 2 + 3 \cdot d \cdot 2^2 + 4 \cdot e \cdot 2^3 = 4 \qquad &\Longrightarrow \qquad 12d + 32e &= -16\,. \end{aligned}$$

Tuto soustavu vyřešíme a dostaneme $d = -4$, $e = 1$. Tedy

$$P_4(x) = 2 - 4(x+1) + 6(x+1)^2 - 4(x+1)^3 + (x+1)^4 = x^4 - 1\,. \qquad \square$$

3.1.2. Interpolační splajny

Jestliže chceme interpolovat funkci $f(x)$ na poměrně dlouhém intervalu $\langle a, b\rangle$, musíme žádat splnění interpolačních podmínek v dostatečně velkém počtu bodů. Pokud bude interpolantem polynom, musí být vysokého stupně a to, jak víme, obvykle vede k velkým chybám mezi uzly. Tudy proto cesta nevede. Lepší je rozdělit interval $\langle a, b\rangle$ na řadu menších subintervalů a na každém z nich sestrojit interpolační polynom nižšího stupně.

Předpokládejme, že

$$a = x_0 < x_1 < \cdots < x_{i-1} < x_i < x_{i+1} < \cdots < x_{n-1} < x_n = b \tag{3.14}$$

je *dělení* intervalu $\langle a, b\rangle$. V každém uzlu x_i je předepsána hodnota y_i interpolantu. Délku i-tého intervalu $\langle x_{i-1}, x_i\rangle$ označíme h_i a délku nejdelšího intervalu h, tj.

$$h_i = x_i - x_{i-1}\,, \qquad i = 1, 2, \ldots, n\,, \qquad\qquad h = \max_{1\le i\le n} h_i\,. \tag{3.15}$$

Hledaný po částech polynomický interpolant budeme značit $S(x)$ a nazveme ho *interpolačním splajnem*. Na každém intervalu $\langle x_{i-1}, x_i\rangle$ je $S(x)$ polynom, jehož příslušnost k i-tému intervalu vyznačíme indexem i, tj.

$S(x)$ je na intervalu $\langle x_{i-1}, x_i\rangle$ polynom $S_i(x)$.

K vyjádření polynomu $S_i(x)$ s výhodou použijeme *lokální proměnnou*

$s = x - x_{i-1}$.

Budeme také používat první poměrnou diferenci

$$\delta_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} = \frac{y_i - y_{i-1}}{h_i}\,.$$

Lineární interpolační splajn (dále jen lineární splajn) je to nejjednodušší, co nás napadne: každé dva sousední body $[x_{i-1}, y_{i-1}]$ a $[x_i, y_i]$ spojíme úsečkou. Zřejmě

$$S_i(x) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}) = y_{i-1} + s\delta_i \tag{3.16}$$

je lineární interpolační polynom procházející body $[x_{i-1}, y_{i-1}]$ a $[x_i, y_i]$. Lineární splajn $S(x)$ je spojitá funkce, derivace $S'(x)$ je však ve vnitřních uzlech obecně nespojitá.

Jestliže $y_i = f(x_i)$, $i = 0, 1, \ldots, n$, a $f \in C^2\langle a, b\rangle$, pak pro chybu interpolace platí

$$|f(x) - S(x)| \le Ch^2\,, \tag{3.17}$$

kde $x \in \langle a, b\rangle$ je libovolné a C je konstanta nezávislá na h.

Pro dostatečně mnoho uzlů lze učinit chybu libovolně malou. Například při vykreslování na obrazovku monitoru s rozlišením 1920 x 1080 bodů jistě stačí použít 1920 interpolačních uzlů k získání kvalitního grafu interpolované funkce. Možná bychom byli spokojeni, i kdybychom zvolili méně uzlů, avšak při postupném snižování počtu uzlů by nutně nastal okamžik, kdy by nás již začaly rušit ostré hrany grafu $S(x)$ v interpolačních

uzlech. Pokud bychom současně vykreslovali také funkci $f(x)$, pak by nám začaly vadit také viditelné odchylky interpolantu $S(x)$ od interpolované funkce $f(x)$ mezi uzly interpolace.

Přesnější interpolant bychom mohli sestrojit tak, že bychom na intervalech $\langle x_0, x_k\rangle$, $\langle x_k, x_{2k}\rangle, \ldots$ aproximovali $f(x)$ pomocí interpolačních polynomů stupně (nejvýše) $k > 1$. Chyba interpolace by v tom případě byla úměrná h^{k+1}, derivace v uzlech $x_k, x_{2k}, \ldots$ by však zůstaly nespojité. Velké k ale nemá smysl používat, jinak bychom zase mohli dostat velké chyby mezi uzly interpolace a byli bychom zpět v situaci, které jsme se právě interpolací po částech chtěli vyhnout.

Velmi populární je aproximace po částech kubickým polynomem, která je nejen spojitá, ale má také spojité první nebo dokonce i druhé derivace. Popisu takových aproximací se budeme věnovat v následujících odstavcích.

Hermitův kubický interpolační splajn (dále jen Hermitův kubický splajn) hledáme jako funkci $S(x)$, která

a) je v intervalu $\langle a, b\rangle$ spojitá spolu se svou první derivací, tj. $S \in C^1\langle a, b\rangle$,

b) splňuje interpolační podmínky

$$S(x_i) = y_i\,, \qquad S'(x_i) = d_i\,, \qquad i = 0, 1, \ldots, n\,, \tag{3.18}$$

kde y_i, d_i jsou předepsané funkční hodnoty a derivace,

c) je na každém intervalu $\langle x_{i-1}, x_i\rangle$, $i = 1, 2, \ldots, n$, polynom třetího stupně.

$S_i(x)$ je tedy kubický Hermitův polynom jednoznačně určený podmínkami

$$\begin{aligned} S_i(x_{i-1}) &= y_{i-1}\,, \quad & S_i'(x_{i-1}) &= d_{i-1}\,,\\ S_i(x_i) &= y_i\,, & S_i'(x_i) &= d_i\,. \end{aligned}$$

Snadno ověříme, že tyto podmínky jsou splněny pro

$$S_i(x) = y_{i-1} + sd_{i-1} + s^2\,\frac{3\delta_i - 2d_{i-1} - d_i}{h_i} + s^3\,\frac{d_{i-1} - 2\delta_i + d_i}{h_i^2}\,. \tag{3.19}$$

Funkce $S(x)$ je spojitá spolu se svou první derivací, druhá derivace už obecně spojitá není.

Jestliže $y_i = f(x_i)$, $d_i = f'(x_i)$, $i = 0, 1, \ldots, n$, a $f \in C^4\langle a, b\rangle$, pak pro chybu interpolace platí

$$|f(x) - S(x)| \le Ch^4\,, \tag{3.20}$$

kde $x \in \langle a, b\rangle$ je libovolné a C je konstanta nezávislá na h.

Pokud derivace d_i nejsou k dispozici, musíme je vypočítat pomocí vhodně zvolených dodatečných podmínek.

Kubický interpolační splajn (dále jen kubický splajn). Směrnice d_i ve vnitřních uzlech můžeme určit tak, že požadujeme, aby splajn $S \in C^2\langle a, b\rangle$, tj. aby platilo

$$S_i''(x_i) = S_{i+1}''(x_i)\,, \qquad i = 1, 2, \ldots, n-1\,. \tag{3.21}$$

Derivováním (3.19) dostaneme

$$S_i''(x) = \frac{(6h_i - 12s)\delta_i + (6s - 4h_i)d_{i-1} + (6s - 2h_i)d_i}{h_i^2}\,.$$

Pro $x = x_i$ je $s = h_i$, takže

$$S_i''(x_i) = \frac{-6\delta_i + 2d_{i-1} + 4d_i}{h_i}\,.$$

Pro $x = x_{i-1}$ je $s = 0$ a

$$S_i''(x_{i-1}) = \frac{6\delta_i - 4d_{i-1} - 2d_i}{h_i}\,.$$

Když v posledním vzorci zvětšíme index i o jedničku, dostaneme

$$S_{i+1}''(x_i) = \frac{6\delta_{i+1} - 4d_i - 2d_{i+1}}{h_{i+1}}\,.$$

Dosazením do (3.21) tak dostaneme rovnice

$$h_{i+1}d_{i-1} + 2(h_{i+1} + h_i)d_i + h_i d_{i+1} = 3(h_{i+1}\delta_i + h_i\delta_{i+1})\,, \qquad i = 1, 2, \ldots, n-1\,. \quad (3.22)$$

Jestliže předepíšeme *okrajové podmínky*

$$S'(a) = d_a\,, \quad S'(b) = d_b\,, \quad (3.23)$$

pak v soustavě (3.22) dosadíme v první rovnici $d_0 := d_a$ a člen $h_2 d_a$ převedeme na pravou stranu, a v poslední rovnici dosadíme $d_n := d_b$ a člen $h_{n-1}d_b$ převedeme na pravou stranu. Soustavu pak řešíme a získáme zbývající směrnice $d_i, i = 1, 2, \ldots, n-1$. Matice soustavy je třídiagonální, diagonálně dominantní, takže soustavu lze snadno vyřešit GEM upravenou pro soustavy s třídiagonální maticí. V MATLABu lze pro výpočet kubického splajnu použít funkci `spline`.

Jestliže $y_i = f(x_i)$, $i = 0, 1, \ldots, n$, $d_0 = f'(x_0)$, $d_n = f'(x_n)$, a když $f \in C^4\langle a, b\rangle$, pak pro chybu interpolace opět platí (3.20).

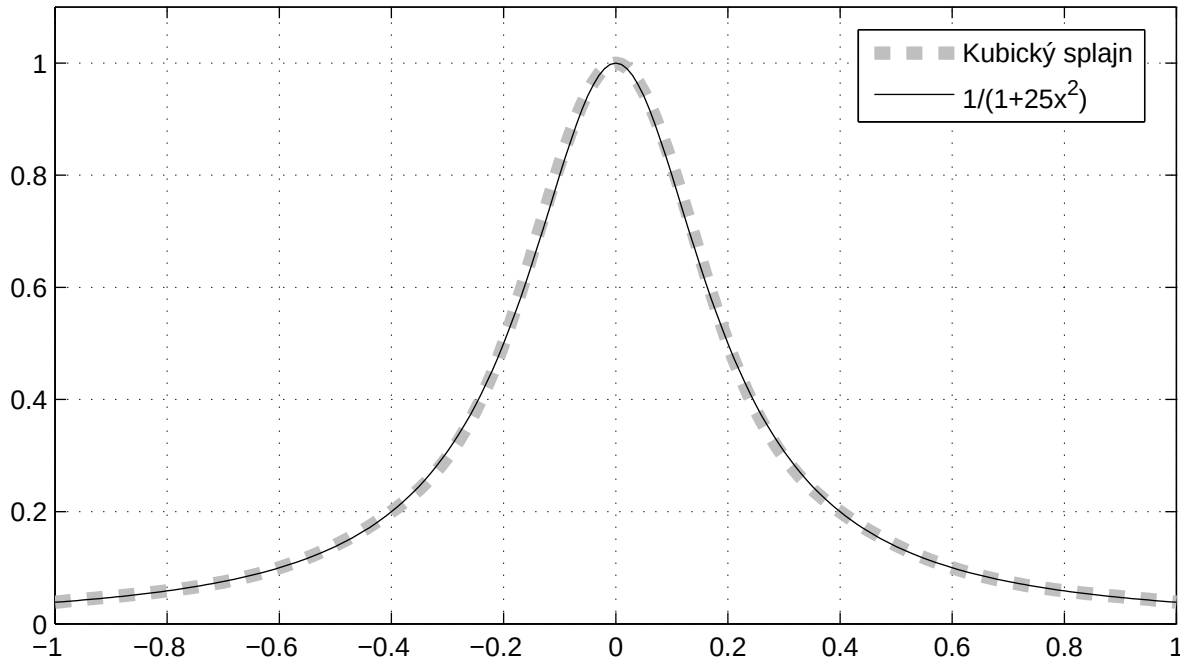
Obrázek 3.3 potvrzuje, že pomocí kubického splajnu lze pro data stejná jako v příkladu 3.4 dostat zcela vyhovující aproximaci Rungeovy funkce.

Kubický splajn má pozoruhodnou *extremální vlastnost*, kterou si teď popíšeme. Označíme

$$V = \{v \in C^2\langle a, b\rangle \,|\, v(x_i) = y_i, i = 0, 1, \ldots, n, v'(x_0) = d_0, v'(x_n) = d_n\}$$

množinu všech funkcí, které mají v intervalu $\langle a, b\rangle$ spojitou druhou derivaci, procházejí zadanými body $[x_i, y_i]$, $i = 0, 1, \ldots, n$, a v krajních bodech $a = x_0$ a $x_n = b$ jejich derivace nabývají předepsaných hodnot d_0 a d_n. Pak $\int_a^b [v''(x)]^2\,\mathrm{d}x$ nabývá na množině funkcí V své nejmenší hodnoty pro kubický splajn $S(x)$, tj. platí

$$\int_a^b [S''(x)]^2\,\mathrm{d}x = \min_{v \in V} \int_a^b [v''(x)]^2\,\mathrm{d}x\,.$$



Obr. 3.3: Aproximace Rungeovy funkce kubickým splajnem

Tato vlastnost má zajímavou interpretaci v mechanice. Je totiž známo, že ohybová energie homogenního izotropního prutu, jehož střednicová čára má rovnici $y = v(x)$, $x \in \langle a, b\rangle$, má přibližně hodnotu $E(v) = c\int_a^b [v''(x)]^2\,\mathrm{d}x$, kde c je vhodná konstanta. A dále je také známo, že prut, který je donucen procházet pevnými interpolačními body $[x_i, y_i]$, zaujme pozici s minimální energií. Extremální vlastnost tedy tvrdí, že kubický splajn aproximuje střednicovou čáru takového prutu.

Jestliže směrnice d_a a d_b v krajních bodech intervalu $\langle a, b\rangle$ neznáme, osvědčil se postup, označovaný v anglicky psané literatuře jako *not a knot*. Myšlenka je jednoduchá: požadujeme, aby splajn byl jednoduchým polynomem třetího stupně na prvních dvou intervalech, tj. pro $x_0 \le x \le x_2$, a na posledních dvou intervalech, tj. pro $x_{n-2} \le x \le x_n$. V uzlech x_1 a x_{n-1} tedy už nedochází k napojování dvou různých polynomů, tj. uzel x_1 a x_{n-1} už není knot, česky *uzel splajnu*, odtud název postupu not a knot.

Polynomy $S_1(x)$ a $S_2(x)$ mají v bodě x_1 společnou funkční hodnotu y_1, stejnou první derivaci d_1 a podle (3.21) také stejnou druhou derivaci. Aby oba polynomy byly totožné stačí, když budou mít v bodě x_1 také stejnou třetí derivaci. Stejnou úvahu lze provést v bodě x_{n-1}. Dostáváme tak okrajové podmínky

$$S_1'''(x_1) = S_2'''(x_1)\,, \qquad S_{n-1}'''(x_{n-1}) = S_n'''(x_{n-1})\,. \tag{3.24}$$

Když pomocí (3.19) vyjádříme podmínku $S_1'''(x_1) = S_2'''(x_1)$ a upravíme ji pomocí první rovnice soustavy (3.22), dostaneme rovnici

$$h_2 d_0 + (h_2 + h_1)d_1 = [(3h_1 + 2h_2)h_2\delta_1 + h_1^2\delta_2]/(h_1 + h_2)\,. \tag{3.25}$$

Podobně zpracujeme také podmínku $S_{n-1}'''(x_{n-1}) = S_n'''(x_{n-1})$ a dostaneme rovnici

$$(h_n + h_{n-1})d_{n-1} + h_{n-1}d_n = [h_n^2\delta_{n-1} + (2h_{n-1} + 3h_n)h_{n-1}\delta_n]/(h_{n-1} + h_n)\,. \tag{3.26}$$

Neznámé $d_0, d_1, \ldots, d_n$ pak dostaneme jako řešení soustavy rovnic, z nichž první je rovnice (3.25), pak následuje $n-1$ rovnic (3.22) a nakonec přijde ještě rovnice (3.26). Použijeme opět GEM upravenou pro soustavy s třídiagonální maticí.

Je-li aproximovaná funkce periodická s periodou $b-a$, je přirozené požadovat, aby aproximující splajn byl rovněž periodický s toutéž periodou. Je-li tedy S *periodický kubický splajn* s periodou $b-a$, pak na intervalu $\langle b, b+h_1\rangle$ nabývá stejných hodnot jako na intervalu $\langle a, a+h_1\rangle$. Jestliže označíme $h_{n+1} = h_1$, $x_{n+1} = x_n + h_{n+1}$, $S(x_{n+1}) = y_{n+1}$ a $S'(x_{n+1}) = d_{n+1}$, pak musí platit $y_n = y_0$, $d_n = d_0$, $y_{n+1} = y_1$ a $d_{n+1} = d_1$. Spojitost S'' v bodě x_n znamená, že rovnice (3.22) platí také pro $i = n$. Periodický kubický splajn je tedy určen, pokud vypočteme $d_1, d_2, \ldots, d_n$ z rovnic (3.22) pro $i = 1, 2, \ldots, n$ s tím, že v první z nich, tj. pro $i = 1$, místo d_0 píšeme d_n, v poslední z nich , tj. pro $i = n$, místo d_{n+1} píšeme d_1 a položíme $h_{n+1} = h_1$, $\delta_{n+1} = \delta_1$.

Shrnutí. Kubický interpolační splajn $S(x)$ je funkce, která

a) je v intervalu $\langle a, b\rangle$ spojitá spolu se svou první a druhou derivací, tj. $S \in C^2\langle a, b\rangle$,

b) splňuje interpolační podmínky $S(x_i) = y_i$, $i = 0, 1, \ldots, n$, kde y_i jsou předepsané funkční hodnoty,

c) je na každém intervalu $\langle x_{i-1}, x_i\rangle$ polynom třetího stupně,

d) splňuje okrajové podmínky (3.23) nebo (3.24) nebo je periodická s periodou $b-a$.

Příklad 3.6. Oblouk $[x(t), y(t)] \equiv [\cos t, \sin t]$, $t \in \langle 0, \pi/2\rangle$, budeme aproximovat křivkou $[S^x(t), S^y(t)]$, kde $S^x(t)$ resp. $S^y(t)$ jsou kubické splajny funkcí $\cos t$ resp. $\sin t$ na intervalu $\langle 0, \pi/2\rangle$. Interval $\langle 0, \pi/2\rangle$ rozdělíme na n stejných dílků, takže $t_i = \pi i/(2n)$. Označíme $d_i^x = [S^x]'(t_i)$, $d_i^y = [S^y]'(t_i)$. Protože $x'(t) = -\sin t$, $y'(t) = \cos t$, položíme

$$d_0^x = -\sin 0 = 0\,, \quad d_n^x = -\sin(\pi/2) = -1\,, \quad d_0^y = \cos 0 = 1\,, \quad d_n^y = \cos(\pi/2) = 0\,.$$

V dalším budeme uvažovat $n = 3$. Soustavu rovnic (3.22) sestavíme zvlášť pro x-ovou a zvlášť pro y-ovou složku. Matice obou soustav je stejná, pravé strany jsou však různé. Na dalším řádku je uvedena soustava rovnic se dvěma pravými stranami a její řešení:

$$\tfrac{1}{6}\pi \begin{pmatrix} 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ d_1^x & d_1^y \\ d_2^x & d_2^y \\ -1 & 0 \end{pmatrix} = 3 \begin{pmatrix} -\frac{1}{2} & \frac{1}{2}\sqrt{3} \\ -\frac{1}{2}\sqrt{3} & \frac{1}{2} \end{pmatrix} \implies \begin{array}{l} d_1^y = -d_2^x \doteq 0{,}865537, \\ d_2^y = -d_1^x \doteq 0{,}499813. \end{array}$$

$S_i^x(t)$ a $S_i^y(t)$ určíme podle (3.19). Například pro $t \in \langle \pi/6, \pi/3\rangle$ dostaneme

$$\begin{aligned} S_2^x(t) &\doteq 0{,}8660 - 0{,}4998(t - \tfrac{1}{6}\pi) - 0{,}4431(t - \tfrac{1}{6}\pi)^2 + 0{,}1195(t - \tfrac{1}{6}\pi)^3, \\ S_2^y(t) &\doteq 0{,}5000 + 0{,}8655(t - \tfrac{1}{6}\pi) - 0{,}2554(t - \tfrac{1}{6}\pi)^2 - 0{,}1195(t - \tfrac{1}{6}\pi)^3. \end{aligned}$$ □

Konstrukce kubického interpolačního splajnu užitím druhých derivací. Snadno ověříme, že kubický polynom

$$S_i(x) = y_{i-1} + s\,\frac{6\delta_i - 2h_iM_{i-1} - h_iM_i}{6} + s^2\,\frac{M_{i-1}}{2} + s^3\,\frac{M_i - M_{i-1}}{6h_i}\,. \tag{3.27}$$

splňuje podmínky

$$S_i(x_{i-1}) = y_{i-1}\,, \quad S_i''(x_{i-1}) = M_{i-1}\,,$$
$$S_i(x_i) = y_i\,, \qquad S_i''(x_i) = M_i\,.$$

Funkce $S(x)$, která je na každém intervalu $\langle x_{i-1}, x_i\rangle$ definována předpisem (3.27), proto zřejmě splňuje podmínky $S(x_i) = y_i$, $S''(x_i) = M_i$, $i = 0, 1, \ldots, n$. $S(x)$ je tedy na intervalu $\langle a, b\rangle$ spojitá a má v něm spojitou druhou derivaci. Abychom dostali kubický splajn, musí mít $S(x)$ v $\langle a, b\rangle$ spojitou také první derivaci. Protože nespojitost $S'(x)$ může nastat jedině ve vnitřních uzlech, stačí požadovat

$$S_i'(x_i) = S_{i+1}'(x_i)\,, \qquad i = 1, 2, \ldots, n-1\,. \tag{3.28}$$

Vyjádříme-li (3.28) pomocí (3.27), dostaneme rovnice

$$h_i M_{i-1} + 2(h_i + h_{i+1})M_i + h_{i+1}M_{i+1} = 6(\delta_{i+1} - \delta_i)\,, \qquad i = 1, 2, \ldots, n-1\,. \tag{3.29}$$

Když zvolíme okrajové podmínky

$$S''(a) = M_a\,, \qquad S''(b) = M_b\,, \tag{3.30}$$

dosadíme je do (3.29), soustavu rovnic vyřešíme a získáme M_i, $i = 1, 2, \ldots, n-1$. Všimněte si, že matice soustavy (3.29) je symetrická. Je samozřejmě možné uvažovat také jiné typy okrajových podmínek, např. (3.23) nebo (3.24).

Kubický splajn s vlastností $S''(a) = S''(b) = 0$ se nazývá *přirozený kubický splajn*. Je známo, že přirozený kubický splajn aproximuje průhyb prostě podepřeného nosníku procházejícího body $[x_i, y_i]$. M_i mají význam ohybových momentů v uzlech $[x_i, y_i]$.

3.1.3. Interpolace funkcí více proměnných

Omezíme se na případ, kdy f je funkce dvou proměnných definovaná v oblasti Ω.

Interpolace po částech lineární. Předpokládejme, že Ω je mnohoúhelník. Oblast Ω *triangulujeme*, tj. vyjádříme ji jako sjednocení trojúhelníků $T_1, T_2, \ldots, T_m$, z nichž každé dva různé buďto nemají žádný společný bod nebo mají společný vrchol popřípadě mají společnou stranu. Množinu $\mathcal{T} = \{T_k\}_{k=1}^m$ všech takových trojúhelníků nazýváme *triangulací* oblasti Ω. Vrcholy trojúhelníků triangulace označíme $P_1 = [x_1, y_1]$, $P_2 = [x_2, y_2]$, $\ldots$, $P_n = [x_n, y_n]$ a nazveme je *uzly triangulace*. Předpokládejme, že v každém uzlu P_i je předepsána hodnota $f_i = f(x_i, y_i)$ interpolované funkce f.

Po částech lineárním interpolantem funkce f na oblasti Ω rozumíme funkci S, která je v Ω spojitá, splňuje interpolační podmínky $S(x_i, y_i) = f_i$, $i = 1, 2, \ldots, n$, a která je na každém trojúhelníku $T_k \in \mathcal{T}$ lineární. Na T_k je tedy $z = S(x, y) \equiv S_k(x, y)$ rovnice roviny určené hodnotami funkce f ve vrcholech T_k. Připomeňme, že rovnice roviny procházející body $[x_a, y_a, z_a]$, $[x_b, y_b, z_b]$ a $[x_c, y_c, z_c]$ může být vyjádřena ve tvaru

$$\begin{vmatrix} x - x_a & y - y_a & z - z_a \\ x_b - x_a & y_b - y_a & z_b - z_a \\ x_c - x_a & y_c - y_a & z_c - z_a \end{vmatrix} = 0\,.$$

Vypočítat hodnotu $z = S(x, y)$ pro $(x, y) \in \Omega$ je snadné: určíme trojúhelník T_k, v němž bod $[x, y]$ leží, a vypočteme $z = S_k(x, y)$.

Chyba interpolace je tím menší, čím jemnější triangulaci zvolíme. Když $f \in C^2(\Omega)$ (tj. když f je v Ω spojitá spolu se svými prvními a druhými parciálními derivacemi), pak

$$|f(x, y) - S(x, y)| \leq Ch^2\,,$$

kde h je nejdelší strana trojúhelníků triangulace a C je konstanta nezávislá na h.

Interpolace po částech bilineární. Předpokládejme, že $\Omega = \langle a, b\rangle \times \langle c, d\rangle$ je obdélník. Pomocí dělení $a = x_0 < x_1 < \cdots < x_n = b$, $c = y_0 < y_1 < \cdots < y_m = d$ rozložíme výchozí obdélník Ω na menší obdélníky $R_{ij} = \{(x, y) \,|\, x_{i-1} \leq x \leq x_i, y_{j-1} \leq y \leq y_j\}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. Předpokládejme, že v uzlech $[x_i, y_j]$ jsou předepsány hodnoty $f_{ij} = f(x_i, y_j)$ funkce f.

Na obdélníku R_{ij} definujeme funkci

$$S_{ij}(x, y) = f_{i-1,j-1}\frac{x_i - x}{x_i - x_{i-1}}\,\frac{y_j - y}{y_j - y_{j-1}} + f_{i,j-1}\,\frac{x - x_{i-1}}{x_i - x_{i-1}}\,\frac{y_j - y}{y_j - y_{j-1}} +$$
$$+ f_{i-1,j}\,\frac{x_i - x}{x_i - x_{i-1}}\,\frac{y - y_{j-1}}{y_j - y_{j-1}} + f_{ij}\,\frac{x - x_{i-1}}{x_i - x_{i-1}}\,\frac{y - y_{j-1}}{y_j - y_{j-1}}\,.$$

Funkce S_{ij} je *bilineární*, tj. pro pevné $x = C$ je $S_{ij}(C, y)$ lineární funkce proměnné y a pro pevné $y = D$ je $S_{ij}(x, D)$ lineární funkce proměnné x. Funkce S_{ij} je interpolant funkce f na obdélníku R_{ij}, tj. S_{ij} nabývá ve vrcholech $[x_{i-1}, y_{j-1}]$, $[x_i, y_{j-1}]$, $[x_i, y_j]$ a $[x_{i-1}, y_j]$ stejných hodnot jako funkce f, jak se snadno přesvědčíme.

Na Ω definujeme funkci S předpisem $S(x, y) = S_{ij}(x, y)$ pro $(x, y) \in R_{ij}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. Protože $S(x_i, y_j) = f_{ij}$, $i = 0, 1, \ldots, n$, $j = 0, 1, \ldots, m$, řekneme, že S je *po částech bilineární interpolant* funkce f na obdélníku Ω. Snadno ověříme, že S je v Ω spojitá (stačí si uvědomit, že S_{ij}, a tedy také S, je na každé straně obdélníka R_{ij} lineární funkce jednoznačně určena pomocí hodnot funkce f v koncových bodech této strany). Když $f \in C^2(\Omega)$, pak pro chybu interpolace platí odhad

$$|S(x, y) - f(x, y)| \leq C(h^2 + k^2)\,, \quad \text{kde } h = \max_{1\leq i\leq n}(x_i - x_{i-1})\,,\ k = \max_{1\leq j\leq m}(y_j - y_{j-1})$$

a C je konstanta nezávislá na h, k.

3.2. Metoda nejmenších čtverců

označuje postup pro přibližné řešení přeurčených nebo nepřesně zadaných soustav rovnic, založený na minimalizaci kvadrátů jejich reziduí.

Prokládání dat křivkami je významná skupina úloh, které lze metodou nejmenších čtverců řešit (v anglicky psané literatuře se pro tyto aplikace používá označení *curve fitting*). Popišme si, o co v takových úlohách jde.

Nechť t je nezávisle proměnná, například čas, a $y(t)$ je neznámá funkce proměnné t, kterou chceme aproximovat. Předpokládejme, že jsme provedli m *pozorování*, při nichž byly hodnoty y přibližně změřeny pro určité (navzájem různé) hodnoty t, takže

$$y_i \approx y(t_i)\,, \qquad i = 1, 2, \ldots, m\,,$$

kde symbol $\approx$ vyjadřuje přibližnou rovnost. Naším záměrem je modelovat $y(t)$ lineární kombinací n *bázových funkcí* pro nějaké $n \leq m$:

$$y(t) \approx x_1\varphi_1(t) + x_2\varphi_2(t) + \cdots + x_n\varphi_n(t) =: R_n(t)\,.$$

Funkce $R_n(t)$ se ve statistice nazývá *lineární regresní funkce*. Bázové funkce navrhujeme podle očekávaného průběhu neznámé funkce $y(t)$, určit se mají *parametry* $x_1, x_2, \ldots, x_n$, a to tak, aby

$$y_i \approx R_n(t_i), \quad i = 1, 2, \ldots, m, \qquad \text{maticově} \qquad \mathbf{y} \approx \mathbf{A}\mathbf{x},$$

kde $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$ jsou naměřená data, $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ je vektor neznámých parametrů a $\mathbf{A}$ je tak zvaná *návrhová matice*,

$$\mathbf{A} = \begin{pmatrix} \varphi_1(t_1) & \varphi_2(t_1) & \ldots & \varphi_n(t_1) \\ \varphi_1(t_2) & \varphi_2(t_2) & \ldots & \varphi_n(t_2) \\ \vdots & \vdots & & \vdots \\ \varphi_1(t_m) & \varphi_2(t_m) & \ldots & \varphi_n(t_m) \end{pmatrix} \equiv (\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n)\,.$$

Vektor $\boldsymbol{\varphi}_i = (\varphi_i(t_1), \varphi_i(t_2), \ldots \varphi_i(t_m))^T$, $i = 1, 2, \ldots, n$, je i-tý sloupec matice $\mathbf{A}$.

Rezidua jsou rozdíly mezi pozorováními y_i a modelovanými hodnotami $R_n(t_i)$:

$$r_i = y_i - R_n(t_i) = y_i - \sum_{j=1}^{n} \varphi_j(t_i)x_j \equiv y_i - \sum_{j=1}^{n} a_{ij}x_j\,, \qquad i = 1, 2, \ldots, m\,,$$

kde $a_{ij} = \varphi_j(t_i)$. V maticovém zápisu

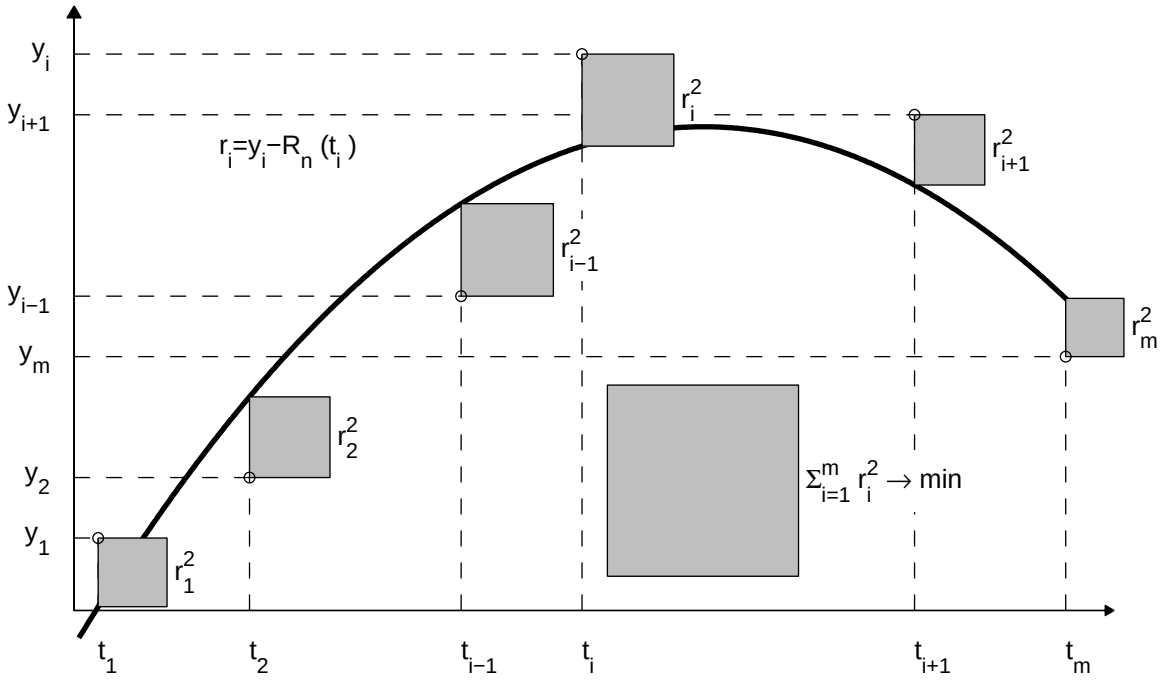
$$\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x}\,. \tag{3.31}$$

Parametry x_i chceme určit tak, aby rezidua byla co nejmenší. Metodu nejmenších čtverců dostaneme, když minimalizujeme součet čtverců reziduí:

$$\|\mathbf{r}\|_2^2 = \sum_{i=1}^{m} r_i^2 \to \min\,. \tag{3.32}$$

Někdy se používá také *vážená metoda nejmenších čtverců*: když jsou některá pozorování významnější než ostatní, můžeme jednotlivým pozorováním přisoudit váhy $w_i > 0$ a minimalizovat součet vážených čtverců reziduí

$$\|\mathbf{r}\|_{2,w}^2 = \sum_{i=1}^{m} w_i r_i^2 \to \min\,.$$



Obr. 3.4: Princip metody nejmenších čtverců

Je-li například chyba i-tého pozorování přibližně rovna e_i, zvolíme $w_i = 1/e_i$.

Normální soustava rovnic. Označme $F(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \equiv \|\mathbf{r}\|_2^2$. Řešení minimalizační úlohy (3.32) musí splňovat nutnou podmínku pro extrém:

$$\frac{\partial F(\mathbf{x})}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n} a_{ij} x_j \right)^2 = 0\,, \qquad k = 1, 2, \ldots, n\,.$$

Když provedeme naznačené derivování, dostaneme

$$\frac{\partial F(\mathbf{x})}{\partial x_k} = 2 \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n} a_{ij} x_j \right) (-a_{ik}) = 0,$$

a odtud

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ik} a_{ij} \right) x_j = \sum_{i=1}^{m} a_{ik} y_i\,, \qquad k = 1, 2, \ldots, n\,,$$

což lze zapsat maticově jako

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{y}\,. \tag{3.33}$$

Soustava lineárních rovnic (3.33) je známa jako *normální soustava rovnic*. Když jsou sloupce matice $\mathbf{A}$ lineárně nezávislé, je matice $\mathbf{G} := \mathbf{A}^T\mathbf{A}$ pozitivně definitní, takže řešení $\mathbf{x}^*$ normální soustavy rovnic minimalizuje $F(\mathbf{x}) \equiv \|\mathbf{r}\|_2^2$ a je tedy řešením úlohy (3.32):

$$\|\mathbf{y} - \mathbf{A}\mathbf{x}^*\|_2^2 = \min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \quad \text{nebo-li} \quad \mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2\,.$$

Vyjádříme-li normální soustavu rovnic pomocí vektorů $\boldsymbol{\varphi}_i$, dostaneme

$$\begin{pmatrix} (\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_1) & (\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2) & \dots & (\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_n) \\ (\boldsymbol{\varphi}_2,\boldsymbol{\varphi}_1) & (\boldsymbol{\varphi}_2,\boldsymbol{\varphi}_2) & \dots & (\boldsymbol{\varphi}_2,\boldsymbol{\varphi}_n) \\ \vdots & \vdots & \dots & \vdots \\ (\boldsymbol{\varphi}_n,\boldsymbol{\varphi}_1) & (\boldsymbol{\varphi}_n,\boldsymbol{\varphi}_2) & & (\boldsymbol{\varphi}_n,\boldsymbol{\varphi}_n) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} (\boldsymbol{\varphi}_1,\mathbf{y}) \\ (\boldsymbol{\varphi}_2,\mathbf{y}) \\ \vdots \\ (\boldsymbol{\varphi}_n,\mathbf{y}) \end{pmatrix}, \tag{3.34}$$

kde

$$(\boldsymbol{\varphi}_k,\boldsymbol{\varphi}_j) = \sum_{i=1}^{m} \varphi_k(t_i)\varphi_j(t_i) \qquad \text{a} \qquad (\boldsymbol{\varphi}_k,\mathbf{y}) = \sum_{i=1}^{m} \varphi_k(t_i)y_i$$

jsou skalární součiny vektorů $\boldsymbol{\varphi}_k,\boldsymbol{\varphi}_j$ a $\boldsymbol{\varphi}_k,\mathbf{y}$. Matice $\mathbf{G}$ soustavy (3.34) se nazývá *Gramova matice* soustavy vektorů $\{\boldsymbol{\varphi}_j\}_{j=1}^n$.

Při návrhu aproximace $R_n(t)$ bychom měli vybírat funkce $\varphi_i(t)$ tak, aby sloupce $\boldsymbol{\varphi}_i$ matice $\mathbf{A}$ byly lineárně nezávislé. V opačném případě, jak se dá ukázat, má úloha (3.32) nekonečně mnoho řešení, což je zřejmě nežádoucí.

Uveďme si dva významné speciální případy, pro které jsou sloupce matice $\mathbf{A}$ lineárně nezávislé (důkaz lze najít např. v [1]):

a) pro $n = N+1$ volíme $\varphi_j(t) = t^{j-1}$, $j = 1,2,\dots,N+1$;

b) pro $n = 2N+1$ volíme

$$\varphi_1(t) = 1, \quad \varphi_{2k}(t) = \cos\frac{k\pi}{L}t, \quad \varphi_{2k+1}(t) = \sin\frac{k\pi}{L}t, \quad k = 1,2,\dots,N\,,$$

a časy pozorování t_i vybíráme z intervalu$\langle c, c+2L)$, kde $L > 0$, c libovolné.

Aproximace $R_n(t)$ je v případě a) algebraický polynom stupně N a v případě b) trigonometrický polynom stupně N.

Když je $m = n$ a matice $\mathbf{A}$ je regulární, pak $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{y}$ a $\mathbf{r} = \mathbf{o}$, tj. $R_n(t_i) = y_i$, $i = 1,2,\dots,m$. Pokud jsou však naměřená data y_i zatížena chybami, pak není účelné, aby funkce $R_n(t)$ tyto chyby kopírovala. Naopak, aby $R_n(t)$ věrohodně vystihovala (rekonstruovala) neznámou funkci $y(t)$, je žádoucí, aby $R_n(t)$ naměřená data *vyrovnávala (vyhlazovala)*. To je ale možné jen tehdy, když počet pozorování m je výrazně větší než počet n návrhových parametrů, tj. pro $m \gg n$.

Příklad 3.7. Pro data předepsaná tabulkou

t_i	0	0,5	1	1,5	2	2,5	3
y_i	3,57	2,99	2,62	2,33	2,22	2,10	2,05

určíme aproximaci $R_2(t) = x_1 + x_2e^{-t}$ metodou nejmenších čtverců. Zřejmě $\varphi_1(t) = 1$ a $\varphi_2(t) = e^{-t}$. Normální soustava rovnic je tvaru

$$\begin{pmatrix} \sum\limits_{i=1}^{7} 1\cdot 1 & \sum\limits_{i=1}^{7} 1\cdot e^{-t_i} \\ \sum\limits_{i=1}^{7} e^{-t_i}\cdot 1 & \sum\limits_{i=1}^{7} e^{-t_i}\cdot e^{-t_i} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sum\limits_{i=1}^{7} 1\cdot y_i \\ \sum\limits_{i=1}^{7} e^{-t_i}\cdot y_i \end{pmatrix}.$$

Vypočteme-li příslušné sumy, dostaneme soustavu (zobrazujeme nejvýše 4 desetinná místa)

$$\begin{pmatrix} 7 & 2{,}4647 \\ 2{,}4647 & 1{,}5805 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 17{,}88 \\ 7{,}4422 \end{pmatrix},$$

jejíž řešení je $x_1 \doteq 1{,}9879$ a $x_2 \doteq 1{,}6087$. Hledaná aproximace $R_2(t) \doteq 1{,}99 + 1{,}61e^{-t}$ a $\|\mathbf{r}\| \doteq 0{,}0651$. □

Příklad 3.8. Daty z předchozího příkladu proložíme postupně polynomy prvního, druhého a třetího stupně. Za bázové volíme funkce $\varphi_j(t) = t^{j-1}$, kde $j = 1, 2, \ldots, n$ a $n = 2, 3, 4$.

a) *Polynom prvního stupně.* Pro $\varphi_1(t) = 1$ a $\varphi_2(t) = t$ dostaneme normální soustavu rovnic

$$\begin{pmatrix} 7 & 10{,}5 \\ 10{,}5 & 22{,}75 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 17{,}88 \\ 23{,}45 \end{pmatrix},$$

která má řešení $x_1 \doteq 3{,}28$ a $x_2 \doteq -0{,}48$, takže $R_2(t) \doteq 3{,}28 - 0{,}48t$ a $\|\mathbf{r}\|_2 \doteq 0{,}4756$. Aproximace lineárním polynomem tedy není vhodná, neboť je málo přesná.

b) *Polynom druhého stupně.* Normální soustava rovnic

$$\begin{pmatrix} 7 & 10{,}5 & 22{,}75 \\ 10{,}5 & 22{,}75 & 55{,}125 \\ 22{,}75 & 55{,}125 & 142{,}1875 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 17{,}88 \\ 23{,}45 \\ 49{,}0625 \end{pmatrix}$$

má řešení $x_1 \doteq 3{,}53$, $x_2 \doteq -1{,}09$ a $x_3 \doteq 0{,}20$, takže $R_3(t) \doteq 3{,}53 - 1{,}09t + 0{,}2t^2$ a $\|\mathbf{r}\|_2 \doteq 0{,}1006$. Velikost rezidua se zmenšila, je však stále větší než v příkladu 3.7.

c) *Polynom třetího stupně.* Normální soustava rovnic (zobrazujeme nejvýše 4 desetinná místa)

$$\begin{pmatrix} 7 & 10{,}5 & 22{,}75 & 55{,}125 \\ 10{,}5 & 22{,}75 & 55{,}125 & 142{,}1875 \\ 22{,}75 & 55{,}125 & 142{,}1875 & 381{,}2813 \\ 55{,}125 & 142{,}1875 & 381{,}2813 & 1049{,}5469 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 17{,}88 \\ 23{,}45 \\ 49{,}0625 \\ 116{,}78 \end{pmatrix}$$

má řešení $x_1 \doteq 3{,}57$, $x_2 \doteq -1{,}35$, $x_3 \doteq 0{,}43$ a $x_4 \doteq -0{,}05$, takže
$R_4(t) \doteq 3{,}57 - 1{,}35t + 0{,}43t^2 - 0{,}05t^3$ a $\|\mathbf{r}\|_2 \doteq 0{,}0360$.

Pokud bychom stupeň polynomu dále zvyšovali, zjistili bychom, že polynom $R_7(t)$ šestého stupně prochází všemi body $[t_i, y_i]$, takže je to interpolační polynom.

Všimněte si ještě prvků Gramových matic: s rostoucím řádem největší koeficient prudce roste. Spolu s ním prudce roste také číslo podmíněnosti Gramových matic. Do následující tabulky jsme zaznamenali čísla podmíněnosti $\kappa_2(\mathbf{G})$ pro $n = 2, 3, \ldots, 7$ (spočtená v MATLABu pomocí maticové $\|\cdot\|_2$ normy)

n	2	3	4	5	6	7
$\kappa_2(\mathbf{G})$	16	$4{,}27 \cdot 10^2$	$1{,}91 \cdot 10^4$	$1{,}20 \cdot 10^6$	$1{,}17 \cdot 10^8$	$2{,}31 \cdot 10^{10}$

Pokud bychom tabulku dat z příkladu 3.7 rozšířili o další sloupce, mohli bychom teoreticky v regresní funkci $\sum_{j=1}^{n} x_j t^{j-1}$ zvětšovat n (až do počtu m sloupců). Prakticky to však možné není, neboť pro velké n by číslo podmíněnosti Gramovy matice bylo neúnosně velké. Regresní funkce $R_n(t)$ s bázovými funkcemi $\varphi_j(t) = t^{j-1}$, $j = 1, 2, \ldots, n$, se proto pro větší n nepoužívají. Pokud potřebujeme polynomickou regresní funkci vyššího stupně, měli bychom použít tzv. *ortogonální polynomy*, viz např. [22]. □

Řešení přeurčených soustav lineárních rovnic $\mathbf{Ax} \approx \mathbf{b}$ *ve smyslu metody nejmenších čtverců* znamená najít přibližné řešení, které minimalizuje délku rezidua $\|\mathbf{b} - \mathbf{Ax}\|_2$. Jsou-li sloupce matice soustavy $\mathbf{A}$ lineárně nezávislé, lze přibližné řešení lze určit jako řešení normální soustavy rovnic $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$.

Pro velké soustavy rovnic to však není dobrá cesta, neboť číslo podmíněnosti Grammovy matice $\mathbf{A}^T\mathbf{A}$ může být značně velké. Existují lepší postupy založené na tzv. *QR rozkladu* matice $\mathbf{A}$ nebo na tzv. *pseudoinverzi* matice $\mathbf{A}$, viz např. [22], [15]. Tyto metody se použití Gramovy matice vyhýbají, jsou stabilní a vypořádají se i s případem, když jsou sloupce matice $\mathbf{A}$ lineárně závislé. V MATLABu lze použít příkaz `x=A\b`, který je založen na QR rozkladu, nebo příkaz `x=pinv(A)*b`, který používá pseudoinverzi.

3.3. Cvičení

3.1. Lagrangeovy fundamentální polynomy $\ell_i(x)$ lze vyjádřit ve tvaru

$$\ell_i(x) = \frac{\omega_{n+1}(x)}{x - x_i} w_i, \quad \text{kde} \quad w_i = \frac{1}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}\,.$$

Lagrangeův interpolační polynom (3.2) proto můžeme zapsat ve tvaru

$$P_n(x) = \omega_{n+1}(x) \sum_{i=0}^{n} y_i \frac{w_i}{x - x_i}. \tag{3.35}$$

Protože interpolační polynom P_n interpoluje funkci $f(x) = 1$ přesně, platí

$$1 = \omega_{n+1}(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i}\,. \tag{3.36}$$

Dělíme-li (3.35) výrazem (3.36), člen $\omega_{n+1}(x)$ se zkrátí a dostaneme *barycentrickou interpolační formuli*

$$P_n(x) = \frac{\sum\limits_{i=0}^{n} \dfrac{w_i}{x - x_i} y_i}{\sum\limits_{i=0}^{n} \dfrac{w_i}{x - x_i}}. \tag{3.37}$$

Kolik operací vyžaduje výpočet koeficientů $\{w_i\}_{i=0}^n$ a kolik výpočet $P_n(x)$ podle (3.35) nebo (3.37)? Kolik operací vyžaduje přidání dalšího uzlu x_{n+1}, tj. modifikace $\{w_i\}_{i=0}^n$ a výpočet w_{n+1}?

3.2. Pro rovnoměrné dělení nedává interpolační polynom nejlepší výsledky. Přesto se však často používá. Platí odhad

$$|\omega_{n+1}(\bar{x})| \leq \frac{h^{n+1}}{4}\, n! \quad \forall \bar{x} \in \langle a, b \rangle\,, \quad \text{kde} \quad h = \frac{b-a}{n}, \quad x_i = a + ih\,, \quad i = 0, 1, \ldots, n\,. \tag{3.38}$$

Porovnejte s obrázkem 3.1. Ověřte experimentálně pomocí jednoduchého programu. Volte různá x, a, b, n.

3.3. Pomocí (3.38) můžeme interpolační chybu (3.11) odhadnout takto:

$$|E_n(\bar{x})| \le M_{n+1}\frac{h^{n+1}}{4(n+1)} \quad \forall \bar{x} \in \langle a,b\rangle\,. \tag{3.39}$$

Jak velké n musíme volit, aby pro $f(x) = \sin x + \cos x$, $\langle a,b\rangle = \langle 0,\pi\rangle$, chyba interpolace klesla pod 10^{-8}?

[$n \ge 11$.]

3.4. Symbolicky (např. pomocí MAPLE) odhadněte M_{11} pro Rungeovu funkci $f(x) = 1/(1+25x^2)$ na intervalu $I = \langle -1,1\rangle$. Pomocí vzorce (3.39) odhadněte chybu interpolačního polynomu P_{10} Rungeovy funkce pro rovnoměrné dělení na I. Porovnejte s obrázkem 3.2.

[$M_{11} \doteq 1{,}77 \cdot 10^{15}$, takto získaný odhad $|E_{10}(\bar{x})| \le 8{,}24 \cdot 10^5$ chyby interpolace je silně nadhodnocen.]

3.5. Napište program pro výpočet hodnot interpolačního polynomu a doplňte ho o možnost grafického posouzení kvality aproximace.

3.6. Vhodnou volbou interpolačních uzlů lze dosáhnout lepší aproximace. Ověřte si to tak, že sestrojíte interpolant nad tzv. Čebyševovými uzly, které pro interval $\langle -1,1\rangle$ jsou $x_i = \cos\dfrac{2i+1}{2n+2}\,\pi$, $i = 0,1,\ldots,n$.

(a) Vykreslete graf Čebyševova interpolantu Rungeovy funkce na intervalu $\langle -1,1\rangle$ pro $n = 10, 20, 30$.
(b) Pomocí transformace $x = \frac{1}{2}[a+b+(b-a)\xi]$, $\xi \in \langle -1,1\rangle$, určete Čebyševovy uzly na libovolném intervalu $\langle a,b\rangle$. Napište program včetně grafického výstupu.

3.7. Ověřte, že Čebyševovy uzly na intervalu $\langle -1,1\rangle$ jsou x-ové souřadnice bodů rovnoměrně rozmístěných na polokružnici $x^2 + y^2 = 1$, $y \ge 0$.

[Body $[x_i, y_i]$, kde $x_i = \cos\varphi_i$, $y_i = \sin\varphi_i$ a $\varphi_i = \dfrac{2i+1}{2n+2}\,\pi$, leží na polokružnici $x^2 + y^2 = 1$, $y \ge 0$, přičemž $\varphi_0 = \dfrac{1}{2}\dfrac{\pi}{n+1}$ a $\varphi_i - \varphi_{i-1} = \dfrac{\pi}{n+1}$, $i = 1,2,\ldots,n$.]

3.8. Naprogramujte Newtonův tvar interpolačního polynomu včetně efektivního výpočtu funkčních hodnot a možnosti přidávání uzlů. Najděte Newtonův tvar interpolačního polynomu pro
(a) $f(x) = \sin x$, interval $\langle 1,3\rangle$, $n = 4$, rovnoměrné dělení.
(b) $f(x) = \cos x$, interval $\langle 1,3\rangle$, $n = 3$, vzestupně uspořádané Čebyševovy uzly.

[(a) $P_4 = 0{,}841471 + 0{,}312048(x-1) - 0{,}488443(x-1)(x-1{,}5) + 0{,}028792(x-1)(x-1{,}5)(x-2) +$
$+0{,}036338(x-1)(x-1{,}5)(x-2)(x-2{,}5)$.
(b) $P_3 = 0{,}474746 - 0{,}963144(x-1{,}07612) + 0{,}058075(x-1{,}07612)(x-1{,}617317) +$
$+0{,}144128(x-1{,}07612)(x-1{,}617317)(x-2{,}382683)$.]

3.9. Pro chybu interpolačního polynomu s Čebyševovými uzly platí odhad

$$|E_n(\bar{x})| \le M_{n+1}\frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} \quad \forall \bar{x} \in \langle a,b\rangle\,. \tag{3.40}$$

Pomocí (3.39) a (3.40) odhadněte chyby interpolačních polynomů ze cvičení 3.8 (a), (b).

[(a) $|E_4(\bar{x})| \le 0{,}0015625$, (b) $|E_3(\bar{x})| \le 0{,}0052083$.]

3.10. Jak velký počet n Čebyševových uzlů musíme zvolit, aby pro $f(x) = \sin x + \cos x$, $\langle a,b\rangle = \langle 0,\pi\rangle$, chyba interpolace klesla pod 10^{-8}?

[$n \ge 10$.]

3.11. Napište program, který sestaví soustavu rovnic pro obecný Hermitův interpolační polynom. Soustavu lineárních rovnic potom řešte algoritmem GEMpp. Jaký polynom dostanete pro data

x_i	y_i	y_i'	y_i''	y_i'''
0	-1	1		
1	1	1	0	1

[$P_5(x) = x + \frac{1}{2}(x-1)^3 - \frac{14}{3}(x-1)^4 - \frac{23}{6}(x-1)^5 = -3{,}8\overline{33}x^5 + 14{,}5x^4 - 19{,}5x^3 + 9{,}8\overline{33}x^2 + x - 1.$]

3.12. Určete kubický splajn pro data z příkladu 3.1. Směrnice v krajních bodech volte $d_0 = 1$ a $d_3 = 0$.

[$S(x) = \begin{cases} -6 + (x+1) + 2{,}\overline{63}(x+1)^2 - 1{,}06\overline{81}(x+1)^3 & \text{pro } x \in \langle -1, 1\rangle, \\ -2 - 1{,}\overline{27}(x-1) - 3{,}7\overline{72}(x-1)^2 + 4{,}0\overline{45}(x-1)^3 & \text{pro } x \in \langle 1, 2\rangle, \\ -3 + 3{,}3\overline{18}(x-2) + 8{,}3\overline{63}(x-2)^2 - 6{,}68\overline{18}(x-2)^3 & \text{pro } x \in \langle 2, 3\rangle. \end{cases}$]

3.13. Určete přirozený kubický splajn pro data z příkladu 3.1. Výpočet proveďte pomocí momentů M_i.

[$S(x) = \begin{cases} -6 + 3{,}565217(x+1) - 0{,}391304(x+1)^3 & \text{pro } x \in \langle -1, 1\rangle, \\ -2 - 1{,}130435(x-1) - 2{,}347826(x-1)^2 + 2{,}478261(x-1)^3 & \text{pro } x \in \langle 1, 2\rangle, \\ -3 + 1{,}608696(x-2) + 5{,}086957(x-2)^2 - 1{,}695652(x-2)^3 & \text{pro } x \in \langle 2, 3\rangle. \end{cases}$]

3.14. Odvoďte not a knot podmínky pomocí momentů M_i.

[$\dfrac{M_1 - M_0}{h_1} = \dfrac{M_2 - M_1}{h_2}$ a $\dfrac{M_{n-1} - M_{n-2}}{h_{n-1}} = \dfrac{M_n - M_{n-1}}{h_n}$.]

3.15. (a) Určete kubický splajn s podmínkami not a knot pro data

x_i	-2	-1	0	1	2
y_i	-1	0	1	0	-1

(b) Ověřte, že kubický splajn S pro data z příkladu 3.1 a okrajové podmínky not a knot splývá s interpolačním polynomem P_3.

[(a) $S(x) = \begin{cases} -1 + 1{,}5(x+2)^2 - 0{,}5(x+2)^3 & \text{pro } x \in \langle -2, 0\rangle, \\ 1 - 1{,}5x^2 + 0{,}5x^3 & \text{pro } x \in \langle 0, 2\rangle\,. \end{cases}$]

3.16. Odvoďte podmínky (3.30) pomocí (3.19).

[$4d_0 + 2d_1 = 6\delta_1 - M_a$ a $2d_{n-1} + 4d_n = 6\delta_n - h_n M_b\,.$]

3.17. Odvoďte podmínky (3.23) pomocí (3.27).

[$2h_1 M_0 + h_1 M_1 = 6(\delta_1 - d_a)$ a $h_n M_{n-1} + 2h_n M_n = 6(d_b - \delta_n)\,.$]

3.18. Napište program, který body $[x_i, y_i]$, $i = 0, 1, \ldots, n$, $[x_n, y_n] = [x_0, y_0]$, proloží uzavřenou křivku $[S^x(t), S^y(t)]$. Návod. Sestavte periodický kubický splajn $S^x(t)$ procházející body $[t_i, x_i]$ a periodický kubický splajn $S^y(t)$ procházející body $[t_i, y_i]$. Zvolte buďto

(a) $\quad t_i = \dfrac{i}{n}, \quad i = 0, 1, \ldots, n,$

nebo

(b) $\quad t_0 = 0, \quad t_i = t_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = 1, 2, \ldots, n.$

Pro $[x_i, y_i] = \left[3\cos\dfrac{2\pi}{n}i, 2\sin\dfrac{2\pi}{n}i\right]$, $i = 0, 1, \ldots, n$, vykreslete $[S^x(t), S^y(t)]$, $t \in \langle 0, t_n\rangle$ pro $n = 8, 16, 32$. Pro $n = 8$ spočtěte derivace $d_7^x = [S^x]'(t_7)$, $d_7^y = [S^y]'(t_7)$.

[(a) $d_7^x = 13{,}2983, d_7^y = 8{,}8656$, (b) $d_7^x = 0{,}8540, d_7^y = 0{,}6234$.]

3.19. Napište program, který pro uzly $x_0 < x_1 < \cdots < x_n$ sestrojí posloupnost $\varphi_0, \varphi_1, \ldots, \varphi_{n+2}$ tzv. *kardinálních splajnů*. Kardinální splajny jsou kubické splajny: $\varphi_0, \varphi_1, \ldots, \varphi_n$ jsou určené podmínkami

$$\varphi_k(x_i) = \begin{cases} 1 & \text{pro } i = k, \\ 0 & \text{pro } i \neq k, \end{cases} \qquad k, i = 0, 1, \ldots, n\,, \qquad \varphi_k'(x_0) = \varphi_k'(x_n) = 0\,, \quad k = 0, 1, \ldots, n\,,$$

a φ_{n+1}, φ_{n+2} jsou určené podmínkami

$$\begin{array}{llll} \varphi_{n+1}(x_i) = 0, & i = 0, 1, \ldots, n\,, & \varphi_{n+1}'(x_0) = 1, & \varphi_{n+1}'(x_n) = 0, \\ \varphi_{n+2}(x_i) = 0, & i = 0, 1, \ldots, n\,, & \varphi_{n+2}'(x_0) = 0, & \varphi_{n+2}'(x_n) = 1\,. \end{array}$$

Zkonstruujte kardinální splajny pro posloupnost uzlů $\{x_i\}_{i=0}^{10} = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$. Jaké jsou hodnoty $\varphi_5(-4{,}5)$ a $\varphi_{11}(0{,}5)$?

[$\varphi_5(-4{,}5) = 1{,}79 \cdot 10^{-3}$, $\varphi_{11}(0{,}5) = -2{,}19 \cdot 10^{-4}$.]

3.20. Nechť $\varphi_0, \varphi_1, \ldots, \varphi_{n+2}$ jsou kardinální splajny na dělení $a = x_0 < x_1 < \cdots < x_n = b$. Ověřte, že

$$S(x) = \sum_{k=0}^{n} y_k \varphi_k(x) + d_a \varphi_{n+1}(x) + d_b \varphi_{n+2}(x)$$

je kubický splajn splňující podmínky $S(x_i) = y_i$, $i = 0, 1, \ldots, n$, $S'(a) = d_a$, $S'(b) = d_b$. Protože pro derivace $d_i = S'(x_i)$, $i = 1, 2, \ldots, n-1$, ve vnitřních uzlech platí (zdůvodněte!)

$$\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix} = \begin{pmatrix} \varphi_0'(x_1) & \varphi_1'(x_1) & \ldots & \varphi_n'(x_1) & \varphi_{n+1}'(x_1) & \varphi_{n+2}'(x_1) \\ \varphi_0'(x_2) & \varphi_1'(x_2) & \ldots & \varphi_n'(x_2) & \varphi_{n+1}'(x_2) & \varphi_{n+2}'(x_2) \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ \varphi_0'(x_{n-2}) & \varphi_1'(x_{n-2}) & \ldots & \varphi_n'(x_{n-2}) & \varphi_{n+1}'(x_{n-2}) & \varphi_{n+2}'(x_{n-2}) \\ \varphi_0'(x_{n-1}) & \varphi_1'(x_{n-1}) & \ldots & \varphi_n'(x_{n-1}) & \varphi_{n+1}'(x_{n-1}) & \varphi_{n+2}'(x_{n-1}) \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \\ d_a \\ d_b \end{pmatrix},$$

je podstatná informace o kardinálních splajnech obsažena v matici jejich prvních derivací.

Napište program, který pomocí kardinálních splajnů počítá kubické splajny $[S^x(t), S^y(t)]$ aproximující parametricky zadanou rovinnou křivku $[x(t), y(t)]$, $t \in \langle 0, 1\rangle$. Na intervalu $\langle 0, 1\rangle$ použijte rovnoměrné dělení $t_i = i/n$, $i = 0, 1, \ldots, n$, kde $n \geq 1$ je volitelné. Matici derivací kardinálních splajnů $\{\varphi_k'(t_i)\}$, $k = 0, 1, \ldots, n+2$, $i = 1, 2, \ldots, n$, stačí spočítat jen jednou a pak ji použít pro jakoukoliv křivku popsanou na dělení $\{t_i\}_{i=0}^n$!

3.21. Rovnoměrným dělením intervalu $I = \langle -1, 2\rangle$ na 5 stejných dílů dostanete hodnoty t_i. Hodnoty y_i spočtěte jako $y_i = f(t_i)$, kde $f(t) = e^{-t^2}$. Vyrovnejte takto vzniklou tabulku metodou nejmenších čtverců pomocí bázových funkcí (a) $1, \sin t, \cos t$, (b) $1, \sin t, \cos t, \sin 2t, \cos 2t$, (c) kvadratickým a (d) kubickým polynomem. Porovnejte hodnoty norem reziduí $\|\mathbf{r}\|$.

[(a) $R_3(t) = 0{,}139765 + 0{,}027624 \sin t + 0{,}686403 \cos t$, $\|\mathbf{r}\| = 0{,}308041$,
(b) $R_5(t) = 0{,}271575 + 0{,}076233 \sin t + 0{,}415631 \cos t - 0{,}056804 \sin 2t + 0{,}288253 \cos 2t$, $\|\mathbf{r}\| = 0{,}039053$,
(c) $R_3(t) = 0{,}779604 + 0{,}043244t - 0{,}248728t^2$, $\|\mathbf{r}\| = 0{,}383744$,
(d) $R_4(t) = 0{,}946472 - 0{,}184071t - 0{,}567990t^2 + 0{,}212842t^3$, $\|\mathbf{r}\| = 0{,}101482$.]

3.22. Řešte přeurčenou soustavu rovnic

$$\begin{array}{rcrcr} x & + & 2y & = & 6 \\ 2x & + & 4y & = & 11 \\ 7x & + & 8y & = & 24 \end{array}$$

[$x = 0{,}5\overline{33}$, $y = 2{,}5\overline{33}$, $\|\mathbf{r}\| \doteq 0{,}4472$.]

4. Numerický výpočet derivace a integrálu

Společným východiskem obou postupů je náhrada funkce $f(x)$ vhodnou aproximací $\varphi(x)$, která je pak derivována nebo integrována. Jsou-li hodnoty $y_i \approx f(x_i)$ získány měřením, je vhodné data nejdříve vyrovnat, tj. φ získáme pomocí metody nejmenších čtverců. Pokud je funkce $f(x)$ zadána přesnými hodnotami $y_i = f(x_i)$ ve velkém počtu uzlů x_i, pak je účelné určit φ jako po částech polynomický interpolant. Když je uzlů jen pár, lze jako φ použít Lagrangeův popř. Hermitův polynom nevysokého stupně.

4.1. Numerické derivování

Přibližný výpočet derivace $f'(x)$ má smysl například tehdy, když

a) pro dané x můžeme získat odpovídající hodnotu $y = f(x)$, avšak explicitní vyjádření funkce $f(x)$ k dispozici nemáme a proto vzorec pro $f'(x)$ neumíme napsat;

b) funkce $f(x)$ je tak složitá, že výpočet její derivace je příliš pracný;

c) hodnoty funkce $f(x)$ známe jen v několika tabulkových bodech.

V takových případech je účelné nahradit funkci $f(x)$ vhodnou aproximací $\varphi(x)$ a hodnotu derivace $\varphi'(x)$ považovat za přibližnou hodnotu derivace $f'(x)$. Podobně postupujeme, když potřebujeme přibližně určit vyšší derivace: $f^{(k)}(x)$ nahradíme pomocí $\varphi^{(k)}(x)$.

V dalším si uvedeme několik často používaných formulí založených na derivování Lagrangeova interpolačního polynomu $P_n(x)$, tj. když $f'(x)$ aproximujeme pomocí $P_n'(x)$.

Chyba aproximace v uzlovém bodě. Nechť $f \in C^{n+1}\langle a, b\rangle$, kde a je nejmenší a b je největší z uzlů interpolace. Pak pro chybu $f'(x_s) - P_n'(x_s)$ v některém z uzlů x_s platí

$$f'(x_s) - P_n'(x_s) = \frac{f^{(n+1)}(\xi_s)}{(n+1)!}\omega_{n+1}'(x_s)\,, \tag{4.1}$$

kde $\omega_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$ a ξ_s je nějaký bod z intervalu (a, b).

Přehled užitečných vzorců. Uvažme případ, kdy uzly x_i jsou *ekvidistantní* s krokem h, tj. $x_i = x_0 + ih$, $i = 1, 2, \ldots, n$. Abychom docílili jednotného zápisu, označíme uzel x_s, v němž počítáme přibližnou hodnotu derivace, vždy jako x. Také ostatní uzly nečíslujeme, ale vyjadřujeme je pomocí x jako $x + h$, $x - h$ apod. Příslušný vzorec je platný jen pro funkce, které mají potřebný počet spojitých derivací. Bod ξ leží vždy mezi nejmenším a největším uzlem použitým ve vzorci. Pomocí vztahu (4.1) tak dostaneme:

$$n = 1: \qquad f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(\xi)\,, \tag{4.2a}$$

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{1}{2}hf''(\xi)\,, \tag{4.2b}$$

$$n = 2: \qquad f'(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h} + \frac{1}{3}h^2 f'''(\xi)\,, \tag{4.3a}$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6}h^2 f'''(\xi)\,, \tag{4.3b}$$

$$f'(x) = \frac{3f(x) - 4f(x-h) + f(x-2h)}{2h} + \frac{1}{3}h^2 f'''(\xi)\,. \tag{4.3c}$$

Uveďme ještě nejznámější formuli $f''(x_1) \doteq P_2''(x_1)$ pro výpočet druhé derivace. Rovnost

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}h^2 f^{(4)}(\xi)\,. \tag{4.4}$$

ověříme užitím Taylorova rozvoje $f(x \pm h)$ okolo x.

Formule ze vzorce (4.2a) je známa jako *první diference vpřed (dopředná diference)* a formule ze vzorce (4.2b) jako *první diference vzad (zpětná diference)*. Formule ze vzorce (4.3b) bývá označována jako *první centrální diference* a formule ze vzorce (4.4) jako *druhá centrální diference*.

Numerický výpočet parciální derivace nepředstavuje žádný nový problém: derivujeme-li podle proměnné x_i, ostatních proměnných $x_j \neq x_i$ si nevšímáme a některou z výše uvedených formulí aplikujeme jen na x_i. Tak třeba pomocí dopředné diference (4.2a) dostaneme

$$\frac{\partial f(x_1, x_2)}{\partial x_2} \approx \frac{f(x_1, x_2 + h) - f(x_1, x_2)}{h}\,.$$

Podmíněnost numerického výpočtu derivace. Ve vzorcích (4.2) – (4.4) jsme uvedli vždy formuli (jako první sčítanec na pravé straně) a její *diskretizační chybu* (jako druhý sčítanec). Při numerickém výpočtu derivace hrají významnou roli také zaokrouhlovací chyby, a to jak v hodnotách funkce f (tj. ve vstupních datech), tak při vyhodnocení formule (tj. při výpočtu). Ukážeme si to pro formuli ze vzorce (4.2a).

Ve skutečnosti za přibližnou hodnotu derivace $f'(x)$ považujeme výraz

$$\tilde{f}'(x) := \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h} = \frac{[f(x+h) + \varepsilon_1] - [f(x) + \varepsilon_0]}{h} = f'(x) + \frac{1}{2}hf''(\xi) + \frac{\varepsilon_1 - \varepsilon_0}{h}\,,$$

kde ε_1 resp. ε_0 je zaokrouhlovací chyba, které se dopustíme při výpočtu $f(x+h)$ resp. $f(x)$. Tedy

$$f'(x) = \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h} + E_d + E_r\,,$$

kde $E_d := -\frac{1}{2}hf''(\xi)$ je *diskretizační chyba* a $E_r := -(\varepsilon_1 - \varepsilon_0)/h$ je *chyba zaokrouhlovací*. Chování obou chyb je pro $h \to 0$ diametrálně odlišné: zatímco $|E_d| \to 0$, $|E_r| \to \infty$. Pro malá h se tedy zřejmě jedná o špatně podmíněnou úlohu: malé změny $\varepsilon_0, \varepsilon_1$ ve vstupních datech vyvolají velkou změnu E_r a následně také výsledku $\tilde{f}'(x)$.

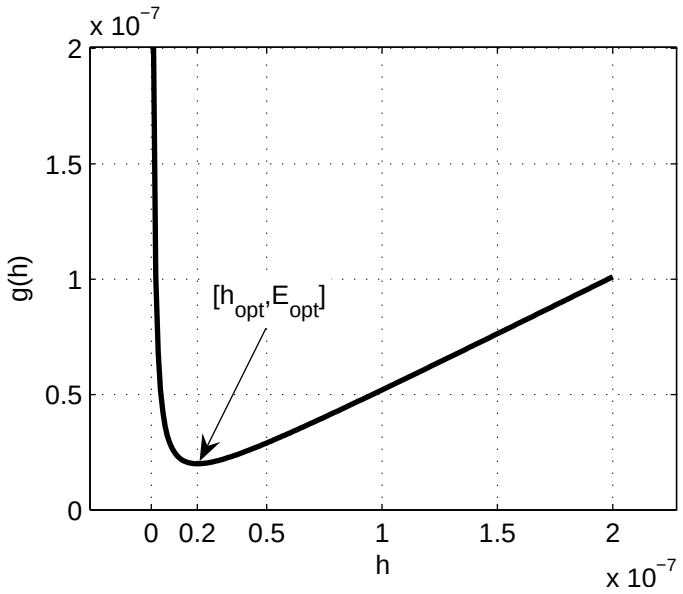
Když pro jednoduchost zanedbáme zaokrouhlovací chyby vznikající při vyčíslení formule $[\tilde{f}(x+h) - \tilde{f}(x)]/h$, dostáváme pro celkovou chybu $E = E_d + E_r$ odhad

$$|E| \leq |E_d| + |E_r| \leq \frac{1}{2}hM_2 + 2\frac{\varepsilon}{h} \equiv g(h)\,,$$

kde $M_2 \geq |f''(\xi)|$ a $\varepsilon \geq \max(|\varepsilon_1|, |\varepsilon_2)|)$. Minimalizací funkce $g(h)$ obdržíme optimální délku kroku

$$h_{opt} = 2\sqrt{\frac{\varepsilon}{M_2}}\,, \quad \text{pro kterou} \quad |E_{opt}| = g(h_{opt}) = 2\sqrt{\varepsilon M_2}\,.$$

Předpokládejme, že hodnoty $f(x)$ i $f(x+h)$ dokážeme vypočítat s relativní chybou rovnou přibližně číslu δ, takže $\varepsilon \approx M_0\delta$, kde $M_0 \approx \max(|f(x_0)|, |f(x_0+h)|)$. Pro $M_0 \approx M_2$ je $h_{opt} \approx 2\sqrt{\delta}$ a $|E_{opt}| \approx 2M_0\sqrt{\delta}$. Počítáme-li tedy např. ve dvojnásobné přesnosti a pokud $\delta \approx 10^{-16}$, pak $h_{opt} \approx 2 \cdot 10^{-8}$. Jestliže navíc $|f'(x)| \approx M_0$, pak $|E_{opt}| \approx 2|f'(x)|\sqrt{\delta}$, a to znamená, že velikost relativní chyby derivace $\tilde{f}'(x)$ je řádově rovna druhé odmocnině velikosti relativní chyby funkčních hodnot. To nás opravňuje k tvrzení: *při přibližném výpočtu derivace formulí dopředné (nebo zpětné) diference dochází při optimální volbě kroku ke ztrátě přibližně poloviny platných cifer.*



Obr. 4.1: Chyba numerické derivace: pro $g(h) = \frac{1}{2}h + 2 \cdot 10^{-16}/h$ je $h_{opt} = 2 \cdot 10^{-8} = E_{opt}$

Podobné chování vykazují i ostatní formule numerického derivování, tj. *pro krok h blízký h_{opt} je numerický výpočet derivace špatně podmíněná úloha*: nepatrné zmenšení kroku vyvolá značný nárust chyby, viz obr. 4.1.

4.2. Richardsonova extrapolace

Přibližný výpočet derivace lze efektivně zpřesnit technikou známou jako *Richardsonova extrapolace*. Je to univerzální postup umožňující pomocí základní metody nižší přesnosti vytvářet metody vyšší přesnosti. Ukažme si, jak se to dělá.

Předpokládejme, že základní metoda je reprezentována funkcí $F(h)$ parametru h. Metodou F umíme vypočítat hodnotu $F(h)$ pro malá $h > 0$. Naším cílem je co nejpřesněji aproximovat hodnotu $F(0)$, kterou však přímo z formule F určit neumíme.

Předpokládejme, že funkce $F(h)$ může být zapsána ve tvaru mocninného rozvoje

$$F(h) = a_0 + a_1 h^2 + a_2 h^4 + a_3 h^6 + \dots \tag{4.5}$$

Konkrétní formule F je zkoumána v příkladu 4.1, viz (4.14), v příkladu 4.5 a ve cvičení 4.2, viz (4.30). Pro malé h je $F(h)$ jistě dobrou aproximací $F(0) = a_0$. Pokusíme se najít lepší aproximaci a_0. Začneme tím, že vypočteme $F(\frac{h}{2})$. Podle (4.5) platí

$$F\left(\frac{h}{2}\right) = a_0 + a_1 \left(\frac{h}{2}\right)^2 + a_2 \left(\frac{h}{2}\right)^4 + a_3 \left(\frac{h}{2}\right)^6 + \dots \tag{4.6}$$

Největší chybu ve výrazu $a_0 - F(h)$ i $a_0 - F(\frac{h}{2})$ představuje člen obsahující druhou mocninu h. Zbavíme se ho tak, že od čtyřnásobku rovnice (4.6) odečteme rovnici (4.5) a výsledek dělíme třemi. Tak dostaneme

$$F_2(h) := \frac{4F(\frac{h}{2}) - F(h)}{3} = a_0 + a_2^{(2)} h^4 + a_3^{(2)} h^6 + \dots \tag{4.7}$$

Snadno ověříme, že $|a_j^{(2)}| < |a_j|$, $j = 2, 3, \dots$. $F_2(h)$ je proto lepší aproximace a_0 než $F(h)$ neboť $a_0 - F_2(h)$ začíná až čtvrtou mocninou h. Dostali jsme tedy metodu F_2, která je (pro dosti malá h) lepší než původní metoda F. Protože $F_2(h) \approx F(0)$ je spočtena pomocí hodnot funkce F pro h a $\frac{h}{2}$, představuje $F_2(h)$ *extrapolaci funkce F do nuly* (ověřte, že $F_2(h) = P_1(0)$, kde $P_1(t)$ je lineární interpolační polynom procházející body $[(\frac{h}{2})^2, F(\frac{h}{2})]$ a $[h^2, F(h)]$).

Podobným postupem odstraníme z $F_2(h)$ člen obsahující čtvrtou mocninu h a získáme ještě lepší aproximaci $F(0)$. Nejprve vypočteme $F_2(\frac{h}{2})$. Podle (4.7) platí

$$F_2\left(\frac{h}{2}\right) = a_0 + a_2^{(2)} \left(\frac{h}{2}\right)^4 + a_3^{(2)} \left(\frac{h}{2}\right)^6 + \dots \tag{4.8}$$

Rovnici (4.8) násobíme 16, odečteme (4.7) a výsledek dělíme 15. Tak dostaneme metodu F_3, která je pro zvolené h definována předpisem

$$F_3(h) := \frac{16F_2(\frac{h}{2}) - F_2(h)}{15} = a_0 + a_3^{(3)} h^6 + \dots \tag{4.9}$$

přičemž $|a_j^{(3)}| < |a_j^{(2)}| < |a_j|$, $j = 3, 4, \dots$. Všimněte si, abychom mohli vypočítat $F_2(\frac{h}{2})$, musíme nejdříve určit $F(\frac{h}{4})$.

Takto můžeme pokračovat a získávat stále lepší metody, pro které

$$F_{i+1}(h) = \frac{4^i F_i(\frac{h}{2}) - F_i(h)}{4^i - 1} = a_0 + a_{i+1}^{(i+1)} h^{2i+2} + \dots, \qquad i = 1, 2, \dots \tag{4.10}$$

a kde $F_1(h) = F(h)$. Pro koeficienty rozvoje přitom platí $|a_j^{(i+1)}| < |a_j|$, $j = i+1, i+2, \dots$. Protože $F_i(h) - F(0) = a_i^{(i)} h^{2i} + \dots$, řekneme, že $F_i(h)$ je aproximace $F(0)$ řádu h^{2i}.

Výpočet lze přehledně uspořádat do tabulky

$$
\begin{array}{lllllc}
F_1(h) & & & & & \\
F_1(\frac{h}{2}) & F_2(h) & & & & \\
F_1(\frac{h}{4}) & F_2(\frac{h}{2}) & F_3(h) & & & \\
F_1(\frac{h}{8}) & F_2(\frac{h}{4}) & F_3(\frac{h}{2}) & F_4(h) & & \\
F_1(\frac{h}{16}) & F_2(\frac{h}{8}) & F_3(\frac{h}{4}) & F_4(\frac{h}{2}) & F_5(h) & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
\quad\Longleftrightarrow\quad
\begin{array}{lllllc}
T_{00} & & & & & \\
T_{10} & T_{11} & & & & \\
T_{20} & T_{21} & T_{22} & & & \\
T_{30} & T_{31} & T_{32} & T_{33} & & \\
T_{40} & T_{41} & T_{42} & T_{43} & T_{44} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Tab. 4.1 Richardsonova extrapolace, $T_{si} = F_{i+1}(h/2^{s-i})$

Tabulku vyplňujeme po řádcích. Prvky tabulky označíme jako T_{si}, kde $s = 0, 1, \ldots$ je řádkový index a $i = 0, 1, \ldots s$ je index sloupcový. Prvek T_{s0} v prvním sloupci tabulky vypočteme pomocí základní metody $F = F_1$,

$$T_{s0} = F(h/2^s)\,, \qquad s = 0, 1, \ldots\,, \tag{4.11}$$

a další prvky v tomto řádku počítáme ve shodě s (4.10) podle předpisu

$$T_{si} := \frac{4^i T_{s,i-1} - T_{s-1,i-1}}{4^i - 1} = T_{s,i-1} + \frac{T_{s,i-1} - T_{s-1,i-1}}{4^i - 1}\,, \quad i = 1, 2, \ldots, s\,. \tag{4.12}$$

Výpočet ukončíme a T_{si} považujeme za dostatečně přesnou aproximaci $F(0)$, pokud

$$|T_{si} - T_{s,i-1}| < \max(\varepsilon_r |T_{si}|, \varepsilon_a)\,, \tag{4.13}$$

kde ε_r je požadovaná relativní přesnost a ε_a požadovaná přesnost absolutní.

Příklad 4.1. Richardsonovu extrapolaci použijeme pro zpřesnění výpočtu derivace podle formule (4.3b). Jestliže má funkce f dostatečný počet spojitých derivací, pak

$$F(h) := \frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{f^{(3)}(x)}{3!}h^2 + \frac{f^{(5)}(x)}{5!}h^4 + \ldots\,, \tag{4.14}$$

takže $F(h)$ je tvaru (4.5).

Počítejme derivaci funkce $f(x) = \cos x$ pro $x = 1$. Zvolíme např. $h = 0{,}8$ a výpočet ukončíme, když bude splněna podmínka (4.13) pro $\varepsilon_r = \varepsilon_a = 10^{-5}$. V následující tabulce značíme $h_s = h/2^s$, prvky T_{s0} počítáme ze vztahu

$$T_{s0} = \frac{\cos(1 + h_s) - \cos(1 - h_s)}{2h_s}\,,$$

prvky T_{s1} a T_{s2} v dalších sloupcích počítáme podle (4.12). Čísla v tabulce jsou zaokrouhlena na 6 cifer. Protože $|T_{32} - T_{31}| < 10^{-5}$, považujeme $T_{32} = -0{,}841471$ za přibližnou hodnotu $f'(1)$. Přesná hodnota $f'(1) = -\sin(1) \doteq -0{,}84147098$, takže T_{32} má všechny

s	h_s	T_{s0}	T_{s1}	T_{s2}
0	0,8	−0,754543		
1	0,4	−0,819211	−0,840766	
2	0,2	−0,835872	−0,841426	−0,841470
3	0,1	−0,840069	−0,841468	$\boxed{-0{,}841471}$

cifry platné. □

Poznámka. Richardsonovu extrapolaci lze aplikovat na základní metodu F také v případě, když má funkce $F(h)$ obecný rozvoj

$$F(h) = a_0 + a_1 h^{p_1} + a_2 h^{p_2} + a_3 h^{p_3} + \dots \tag{4.5'}$$

kde $1 \le p_1 < p_2 < p_3 < \dots$ jsou přirozená čísla. Přesnější metodu F_{i+1} v tom případě definujeme předpisem

$$F_{i+1}(h) = \frac{2^{p_i} F_i(\frac{h}{2}) - F_i(h)}{2^{p_i} - 1} = a_0 + a_{i+1}^{(i+1)} h^{p_{i+1}} + \dots\,, \qquad i = 1, 2, \dots\,, \tag{4.10'}$$

a T_{si} počítáme podle

$$T_{si} := \frac{2^{p_i} T_{s,i-1} - T_{s-1,i-1}}{2^{p_i} - 1} = T_{s,i-1} + \frac{T_{s,i-1} - T_{s-1,i-1}}{2^{p_i} - 1}\,, \quad i = 1, 2, \dots, s\,. \tag{4.12'}$$

Protože $F_i(h) - F(0) = a_i^{(i)} h^{p_i} + \dots$, řekneme, že$F_i(h)$ je aproximace $F(0)$ řádu h^{p_i}. Pro $p_i = 2i$ dostaneme dříve uvažovaný případ, viz (4.5), (4.10) a (4.12). □

Příklad 4.2. Richardsonovou extrapolací zpřesníme výpočet derivace podle formule (4.2a). Z Taylorovy věty dostaneme

$$F(h) := \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{f^{(2)}(x)}{2!} h + \frac{f^{(3)}(x)}{3!} h^2 + \dots\,, \tag{4.15}$$

což odpovídá (4.5') pro $p_i = i$. Počítat budeme stejnou úlohu jako v příkladu 4.1. Tentokrát požadovanou přesnost dosáhneme až pro T_{44}. Richardsonova extrapolace je méně

s	h_s	T_{s0}	T_{s1}	T_{s2}	T_{s3}	T_{s4}
0	0,8	−0,959381				
1	0,4	−0,925838	−0,892295			
2	0,2	−0,889723	−0,853608	−0,840712		
3	0,1	−0,867062	−0,844401	−0,841332	−0,841421	
4	0,05	−0,854625	−0,842188	−0,841451	−0,841468	$\boxed{-0{,}841471}$

účinná: zatímco pro formuli (4.3b) je T_{32} aproximace řádu h^6, pro formuli (4.2a) je T_{44} aproximace řádu h^5 a k dosažení požadované přesnosti bylo třeba zvolit menší h_s. □

4.3. Numerické integrování

Cílem tohoto odstavce je přibližný výpočet integrálu $I(f) := \int_a^b f(x)\,\mathrm{d}x$. Existuje několik důvodů, proč tento integrál nepočítáme přesně, například

a) integrál $I(f)$ neumíme spočítat analytickými metodami;

b) analytický výpočet je příliš pracný;

c) funkce $f(x)$ je dána jen tabulkou.

Za přibližnou hodnotu integrálu $I(f)$ považujeme integrál $Q(f) := I(\varphi)$, kde $\varphi(x)$ je vhodná aproximace funkce $f(x)$. Předpis $Q(f)$ pro přibližný výpočet integrálu se nazývá *kvadraturní formule*. Rozdíl $I(f) - Q(f)$ označíme $R(f)$ a nazveme *(diskretizační) chybou kvadraturní formule*, tedy

$$I(f) = Q(f) + R(f)\,.$$

Řekneme, že kvadraturní formule je *řádu* r, když integruje přesně polynomy stupně r a polynomy stupně $r+1$ už přesně neintegruje, tj. když $R(x^k) = 0$ pro $k = 0, 1, \ldots, r$, a $R(x^{r+1}) \neq 0$.

Řád formule stačí ověřit na intervalu $\langle a, b\rangle = \langle 0, 1\rangle$. Skutečně, pomocí transformace $x = a+t(b-a)$ dostaneme $\int_a^b x^k\,\mathrm{d}x = (b-a)\int_0^1 g(t)\,\mathrm{d}t$, kde $g(t) = (a+t(b-a))^k$ je polynom stupně k v proměnné t. Proto když formule integruje přesně polynomy stupně $k \leq r$ na intervalu $\langle 0, 1\rangle$, integruje přesně také polynomy stupně r na intervalu $\langle a, b\rangle$.

4.3.1. Základní formule

dostaneme integrací interpolačního polynomu.

Obdélníková formule. Když $P_0(x) = f(\frac{1}{2}(a+b))$ je polynom stupně 0, tj. konstanta rovná hodnotě funkce f ve středu $\frac{1}{2}(a+b)$ intervalu $\langle a, b\rangle$, pak odpovídající formule

$$Q_M(f) := \int_a^b P_0(x)\,\mathrm{d}x = (b-a) f\left(\frac{a+b}{2}\right)\,. \tag{4.16}$$

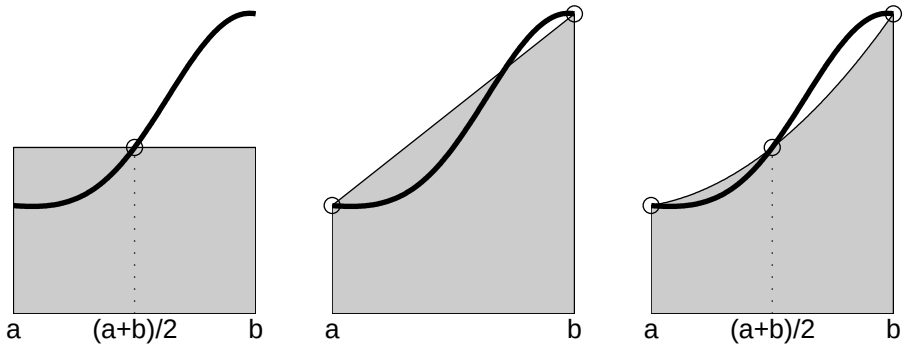
Název formule vyjadřuje skutečnost, že pro $f(\frac{1}{2}(a+b)) > 0$ je $Q_M(f)$ obsah obdélníka o stranách délky $b-a$ a $f(\frac{1}{2}(a+b))$. Obdélníková formule (4.16) se v anglicky psané literatuře označuje jako midpoint rule, odtud index M. Obdélníková formule je řádu 1: na intervalu $\langle 0, 1\rangle$ je $Q_M(1) = 1 = I(1)$, $Q_M(x) = \frac{1}{2} = I(x)$ a $Q_M(x^2) = \frac{1}{4} \neq \frac{1}{3} = I(x^2)$. Pro chybu $R_M(f)$ obdélníkové formule lze za předpokladu, že $f \in C^2\langle a, b\rangle$, odvodit

$$R_M(f) = \frac{1}{24} f''(\xi)(b-a)^3\,, \quad \text{kde } \xi \in (a, b) \tag{4.17}$$

je nějaký (blíže neurčený) bod intervalu (a, b).

Lichoběžníková formule. Jako $P_1(x)$ označíme lineární polynom procházející body $[a, f(a)]$ a $[b, f(b)]$. Integrací $P_1(x)$ na intervalu $\langle a, b\rangle$ obdržíme

$$Q_T(f) := \int_a^b P_1(x)\,\mathrm{d}x = \frac{b-a}{2}[f(a) + f(b)]\,. \tag{4.18}$$



Obr. 4.2: Obdélníková, lichoběžníková a Simpsonova formule

Název formule vyjadřuje skutečnost, že pro $f(a) > 0$, $f(b) > 0$ je $Q_T(f)$ obsah lichoběžníka, jehož rovnoběžné strany mají délky $f(a)$, $f(b)$ a jehož výška je rovna $b-a$. Index T je první písmeno anglického slůvka trapezoid, česky lichoběžník. Lichoběžníková formule je řádu 1: lineární polynom integruje přesně, kvadratický nikoliv. Když $f \in C^2\langle a,b\rangle$, pak pro chybu lichoběžníkové formule platí

$$R_T(f) = -\frac{1}{12} f''(\xi)(b-a)^3\,, \quad \text{kde } \xi \in (a,b). \tag{4.19}$$

Všimněte si:

a) Pokud se druhá derivace $f''(x)$ funkce $f(x)$ na intervalu $\langle a,b\rangle$ příliš nemění, pak je absolutní hodnota $|R_T(f)|$ chyby lichoběžníkové fomule přibližně dvakrát větší než absolutní hodnota $|R_M(f)|$ chyby formule obdélníkové.

b) Pokud druhá derivace $f''(x)$ funkce $f(x)$ nemění na intervalu $\langle a,b\rangle$ znaménko, tj. jeli funkce $f(x)$ pořád konvexní nebo konkávní, pak znaménko chyby lichoběžníkové formule je opačné než znaménko chyby formule obdélníkové. Za těchto okolností přesná hodnota $I(f)$ integrálu leží v intervalu, jehož krajní body jsou hodnoty $Q_M(f)$ a $Q_T(f)$.

Simpsonova formule. Integrací kvadratického interpolačního polynomu $P_2(x)$ procházejícího body $[a, f(a)]$, $[\frac{1}{2}(a+b), f(\frac{1}{2}(a+b))]$ a $[b, f(b)]$ dostaneme

$$Q_S(f) = \int_a^b P_2(x)\,\mathrm{d}x = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]\,. \tag{4.20}$$

Simpsonova formule je řádu 3. Ověřte! (Stačí porovnat $Q_S(x^k)$ a $I(x^k)$ na intervalu $\langle 0,1\rangle$ postupně pro $k = 0,1,2,3,4$.) Když $f \in C^4\langle a,b\rangle$, pro chybu platí

$$R_S(f) = -\frac{1}{90} f^{(4)}(\xi)\left(\frac{b-a}{2}\right)^5\,, \quad \text{kde } \xi \in (a,b)\,. \tag{4.21}$$

Booleova formule vznikne integrací interpolačního polynomu $P_4(x)$, jehož uzly jsou, kromě koncových bodů a, b, také střed $\frac{1}{2}(a+b)$ intervalu $\langle a,b\rangle$ a body $a + \frac{1}{4}(b-a)$

a $b - \frac{1}{4}(b-a)$ ležící v jedné čtvrtině a ve třech čtvrtinách intervalu $\langle a, b\rangle$. Booleova formule

$$Q_B(f) = \frac{b-a}{90}\left[7f(a) + 32f\left(\frac{3a+b}{4}\right) + 12f\left(\frac{a+b}{2}\right) + 32f\left(\frac{a+3b}{4}\right) + 7f(b)\right]$$

je řádu 5 (ověřte!). Pokud $f \in C^6\langle a, b\rangle$, pro chybu platí

$$R_B(f) = -\frac{8}{945}f^{(6)}(\xi)\left(\frac{b-a}{4}\right)^7, \quad \text{kde } \xi \in (a, b)\,.$$

4.3.2. Složené formule

Abychom dostali dostatečně přesnou aproximaci integrálu $I(f)$, rozdělíme interval $\langle a, b\rangle$ na kratší podintervaly a na každém z nich použijeme některou ze základních formulí. Omezíme se na případ, kdy základní formule na podintervalech jsou vždy stejné, a to buďto obdélníkové nebo lichoběžníkové nebo Simpsonovy (sestavení složené Booleovy formule ponecháváme čtenáři jako cvičení). Budeme uvažovat *rovnoměrné (= ekvidistantní) dělení*

$$a = x_0 < x_1 < \cdots < x_n = b, \text{ kde } x_i = a + ih,\ h = (b-a)/n \text{ a } i = 0, 1, \ldots, n. \quad (4.22)$$

Složenou formuli na dělení (4.22) budeme značit pomocí horního indexu n. Délka h se nazývá *krok dělení*.

Složenou obdélníkovou formuli dostaneme součtem jednoduchých obdélníkových formulí $hf(x_{i-1} + \frac{1}{2}h)$ na podintervalech $\langle x_{i-1}, x_i\rangle$. Výsledkem je složená formule

$$Q_M^n(f) := h\left[f\left(x_0 + \frac{1}{2}h\right) + f\left(x_1 + \frac{1}{2}h\right) + \cdots + f\left(x_{n-1} + \frac{1}{2}h\right)\right]. \quad (4.23)$$

Chybu $R_M^n(f)$ dostaneme jako součet dílčích chyb $\frac{1}{24}f''(\xi_i)h^3$ na $\langle x_{i-1}, x_i\rangle$:

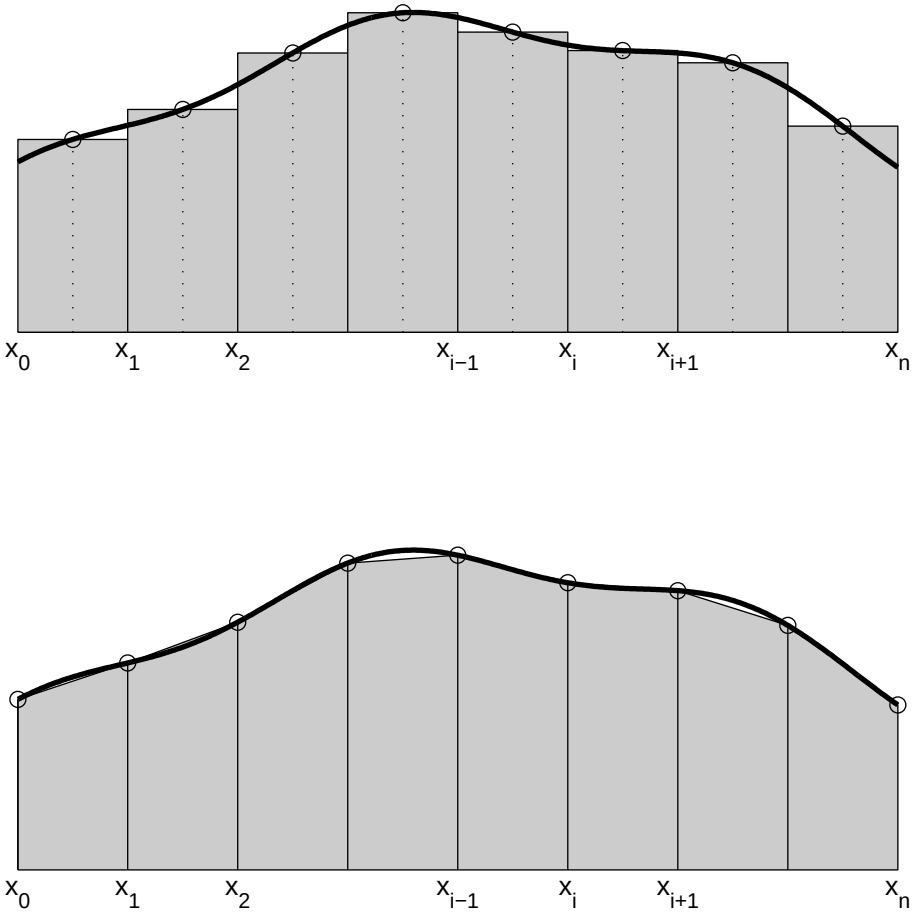
$$\begin{aligned} R_M^n(f) =& \frac{1}{24}h^3 f''(\xi_1) + \frac{1}{24}h^3 f''(\xi_2) + \cdots + \frac{1}{24}h^3 f''(\xi_n) = \\ & \frac{1}{24}h^2\frac{b-a}{n}[f''(\xi_1) + f''(\xi_2) + \cdots + f''(\xi_n)]\,. \end{aligned}$$

Ze spojitosti f'' plyne, že aritmetický průměr $[f''(\xi_1) + f''(\xi_2) + \cdots + f''(\xi_n)]/n$ druhých derivací je roven druhé derivaci$f''(\xi)$ v nějakém bodě $\xi \in (a, b)$. Pro chybu $R_M^n(f)$ složené obdélníkové formule tedy platí

$$R_M^n(f) = \frac{b-a}{24}f''(\xi)h^2\,, \qquad \text{kde } \xi \in (a, b)\,. \quad (4.24)$$

Složená lichoběžníková formule vznikne součtem jednoduchých lichoběžníkových formulí $\frac{1}{2}h[f(x_{i-1}) + f(x_i)]$ na podintervalech $\langle x_{i-1}, x_i\rangle$. Výsledkem je formule

$$Q_T^n(f) := h\left[\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right]. \quad (4.25)$$



Obr. 4.3: Složená obdélníková a lichoběžníková formule

Chybu $R_T^n(f)$ dostaneme jako součet chyb $-\frac{1}{12}f''(\xi_i)h^3$ na podintervalech $\langle x_{i-1}, x_i\rangle$. Po jednoduché úpravě obdržíme

$$R_T^n(f) = -\frac{b-a}{12}f''(\xi)h^2\,, \qquad \text{kde } \xi \in (a,b)\,. \tag{4.26}$$

V MATLABu lze použít funkci `trapz`.

Složenou Simpsonovu formuli dostaneme pro sudý počet n dílků tak, že sečteme jednoduché Simpsonovy formule na intervalech $\langle x_0, x_2\rangle$, $\langle x_2, x_4\rangle$, …, $\langle x_{n-2}, x_n\rangle$ délky $2h$:

$$\begin{aligned} Q_S^n(f) :=& \frac{2h}{6}[f(x_0)+4f(x_1)+f(x_2)] + \frac{2h}{6}[f(x_2)+4f(x_3)+f(x_4)] + \cdots + \\ & \frac{2h}{6}[f(x_{n-4})+4f(x_{n-3})+f(x_{n-2})] + \frac{2h}{6}[f(x_{n-2})+4f(x_{n-1})+f(x_n)]\,. \end{aligned}$$

Odtud tedy

$$Q_S^n(f) := \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+\cdots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)]\,. \tag{4.27}$$

Chyba $R_S^n(f)$ je součtem chyb na podintervalech:

$$R_S^n(f) = -\frac{1}{90}h^5 f^{(4)}(\xi_2) - \frac{1}{90}h^5 f^{(4)}(\xi_4) - \ldots - \frac{1}{90}h^5 f^{(4)}(\xi_n) =$$
$$-\frac{1}{90}h^4\frac{b-a}{2}\left\{\frac{2}{n}\left[f^{(4)}(\xi_2) + f^{(4)}(\xi_4) + \cdots + f^{(4)}(\xi_n)\right]\right\}.$$

Když aritmetický průměr čtvrtých derivací (tj. výraz ve složené závorce) nahradíme členem $f^{(4)}(\xi)$, dostaneme

$$R_S^n(f) = -\frac{b-a}{180}f^{(4)}(\xi)h^4\,, \qquad \text{kde } \xi \in (a,b)\,. \tag{4.28}$$

Jak dosáhnout požadované přesnosti. Vyvstává otázka jak zajistit, aby chyba, které se při numerickém výpočtu integrálu dopustíme, byla menší než zadaná tolerance ε. Ve vzorcích (4.24), (4.26) a (4.28) bohužel vystupuje blíže neurčené číslo ξ, o němž víme jen to, že leží někde v intervalu (a,b). Když označíme

$$M_2 = \max_{x\in\langle a,b\rangle}|f''(x)|, \qquad M_4 = \max_{x\in\langle a,b\rangle}|f^{(4)}(x)|\,,$$

pak chyby můžeme odhadnout podle vztahu

$$|R_M^n(f)| \le \frac{b-a}{24}M_2h^2 \qquad \text{pro složenou obdélníkovou formuli,} \tag{4.24'}$$

$$|R_T^n(f)| \le \frac{b-a}{12}M_2h^2 \qquad \text{pro složenou lichoběžníkovou formuli,} \tag{4.26'}$$

$$|R_S^n(f)| \le \frac{b-a}{180}M_4h^4 \qquad \text{pro složenou Simpsonovu formuli.} \tag{4.28'}$$

Počet dílků $n = (b-a)/h$ pak lze určit tak, aby $|R_M^n(f)| \le \varepsilon$ popř. $|R_T^n(f)| \le \varepsilon$ nebo $|R_S^n(f)| \le \varepsilon$. Takto stanovený počet dílků je však zpravidla přehnaně velký. To je důsledek toho, že jsme v odhadech chyb nahradili derivaci v blíže neurčeném bodě maximem této derivace na celém intervalu $\langle a,b\rangle$.

Příklad 4.3. Určeme počet dílků n tak, abychom vypočítali $\int_0^{\pi/2} e^x\cos x\,\mathrm{d}x$ s chybou nejvýše 10^{-4} pomocí složené obdélníkové, lichoběžníkové a Simpsonovy formule. Přesná hodnota $\int_0^{\pi/2} e^x\cos x\,\mathrm{d}x = \frac{1}{2}e^x(\cos x + \sin x)\big|_0^{\pi/2} \doteq 1{,}905239$.

Pro $f(x) = e^x\cos x$ je $f''(x) = -2e^x\sin x$ a tedy $M_2 \le 2e^{\pi/2}$. V případě obdélníkové formule pomocí (4.24') odvodíme:

$$|R_M^n(f)| \le \frac{\pi/2}{24}\,2\,e^{\pi/2}\left(\frac{\pi/2}{n}\right)^2 \le 10^{-4} \qquad \Longrightarrow \qquad n \ge 125\,.$$

Provedeme-li výpočet s tímto n podle (4.23), dostaneme $Q_M^{125}(f) \doteq 1{,}905277$, takže skutečná chyba je asi $3{,}8\cdot 10^{-5}$. Cestou pokusů se ukázalo, že pro dosažení požadované tolerance stačí vzít $n = 78$.

V případě lichoběžníkové formule pomocí (4.26') odvodíme:

$$|R_T^n(f)| \le \frac{\pi/2}{12}\, 2\, e^{\pi/2} \left(\frac{\pi/2}{n}\right)^2 \le 10^{-4} \qquad \Longrightarrow \qquad n \ge 177\,.$$

Výpočtem podle (4.25) dostaneme $Q_T^{177}(f) \doteq 1{,}905201$, tj. skutečná chyba je asi $3{,}8 \cdot 10^{-5}$. Experimentálně lze ověřit, že požadovanou přesnost lze dosáhnout už při $n = 110$.

Všimněte si, že $Q_T^{177}(f) < I(f) < Q_M^{125}(f)$. To není náhoda. Na $\langle 0, \pi/2\rangle$ je $f''(x) < 0$, tj. f je konkávní, a proto $Q_T^{n_1}(f) < I(f) < Q_M^{n_2}(f)$ pro libovolné počty n_1, n_2 dílků.

Pro odhad počtu dílků Simpsonovy formule vypočteme $f^{(4)}(x) = -4e^x \cos x$ a určíme $M_4 \le 4e^{\pi/2}$. Podle (4.28') tak odvodíme:

$$|R_S^n(f)| \le \frac{\pi/2}{180}\, 4\, e^{\pi/2} \left(\frac{\pi/2}{n}\right)^4 \le 10^{-4} \qquad \Longrightarrow \qquad n \ge 12\,.$$

Výpočtem podle (4.27) dostaneme $Q_S^{12}(f) \doteq 1{,}905226$, tj. skutečná chyba je asi $1{,}3 \cdot 10^{-5}$. Požadovaná přesnost je ve skutečnosti dosažena už pro $n = 8$. □

Rombergova integrace je založena na Richardsonově extrapolaci složené lichoběžníkové formule. Když má totiž funkce f dostatečný počet spojitých derivací, pak je známo, že pro $F(h) := Q_T^n(f)$, kde $h = (b-a)/n$, platí rozvoj (4.5), v němž $a_0 = I(f)$, tj.

$$Q_T^n(f) = I(f) + a_1 h^2 + a_2 h^4 + a_3 h^6 + \dots$$

Přibližný výpočet integrálu $I(f)$ lze proto provést podle vzorců (4.11) – (4.13). Označíme-li $n_s = 2^s n$ a $h_s = 2^{-s} h$, $s = 0, 1, \dots$, pak $T_{s0} = Q_T^{n_s}(f)$ je složená lichoběžníková formule pro n_s dílků. Při jejím výpočtu s výhodou využijeme hodnot funkce f, které jsme už dříve vypočetli na hrubších děleních. Dá se ukázat, že $T_{s1} = Q_S^{n_s}(f)$ je složená Simpsonova formule a $T_{s2} = Q_B^{n_s}(f)$ je složená Booleova formule. Obecně platí, že T_{si} je kvadraturní formule řádu $2i+1$ s krokem h_s.

Příklad 4.5. Rombergovou metodou vypočteme $\int_0^{\pi/2} e^x \cos x \,\mathrm{d}x$ s přesností 10^{-4}. Výsledek je zaznamenán v následující tabulce. Protože $|I(f) - T_{22}| \doteq 2{,}7 \cdot 10^{-6}$, $T_{22} = 1{,}90524$ má

s	n_s	T_{s0}	T_{s1}	T_{s2}
0	2	1,61076		
1	4	1,83082	1,90418	
2	8	1,88659	1,90517	1,90524

všechny cifry platné. □

4.3.3. Doplňující poznatky

Obecný tvar kvadraturní formule je

$$Q(f) = w_0 f(x_0) + w_1 f(x_1) + \dots + w_n f(x_n)\,. \tag{4.29}$$

Body $x_0, x_1, \ldots, x_n$ se nazývají *uzly* a čísla $w_0, w_1, \ldots, w_n$ *koeficienty* (někdy také *váhy*) kvadraturní formule.

Gaussovy kvadraturní formule vybírají uzly a koeficienty tak, aby formule měla maximální řád $r = 2n + 1$ (dá se ukázat, že vyšší řád formule (4.29) mít nemůže). Gaussovy formule se běžně uvádějí pro interval $\langle -1, 1\rangle$. To však není žádné omezení, neboť substitucí

$$x = \frac{a+b}{2} + \frac{b-a}{2}\,t$$

lze výpočet integrálu převést z intervalu $\langle a, b\rangle$ na interval $\langle -1, 1\rangle$. Uzly a koeficienty Gaussových formulí najdeme v každé učebnici numerické matematiky. My zde uvedeme jen první tři formule pro $n = 0, 1, 2$ (pro výpočet integrálu na intervalu $\langle -1, 1\rangle$):

$$\begin{aligned}
&Q_{G0}(f) = 2f(0), && R_{G0}(f) = \frac{1}{3}f''(\xi)\,,\\
&Q_{G1}(f) = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right), && R_{G1}(f) = \frac{1}{135}f^{(4)}(\xi)\,,\\
&Q_{G2}(f) = \frac{5}{9}f(-\sqrt{0,6}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{0,6}), && R_{G2}(f) = \frac{1}{15\,750}f^{(6)}(\xi)\,.
\end{aligned}$$

Ve vzorcích pro chybu je ξ nějaký (blíže neurčený) bod z intervalu $(-1, 1)$, pro každou formuli obecně jiný. Všimněte si, že $Q_{G0}(f)$ je obdélníková formule. Formule $Q_{G1}(f)$ integruje přesně polynomy stupně 3 stejně jako Simpsonova formule. Zatímco Simpsonova formule $Q_S(f)$ je tříbodová, Gaussova formule $Q_{G1}(f)$ je jen dvoubodová. Srovnáním tvaru zbytku $R_S(f)$ Simpsonovy formule a $R_{G1}(f)$ Gaussovy formule lze usuzovat, že Gaussova formule je přibližně o 50% přesnější. Tříbodová formule $Q_{G2}(f)$ integruje přesně polynomy stupně 5.

Adaptivní integrace je založena na nerovnoměrném dělení intervalu integrace $\langle a, b\rangle$: v místech, kde je integrovaná funkce dostatečně hladká a mění se pomalu, použijeme dělení hrubší, a v místech, kde je výpočet integrálu obtížný, použijeme dělení jemnější.

Vysvětleme si, jak se to dá prakticky udělat. Integrál $I(a, b) := \int_a^b f(x)\,\mathrm{d}x$ počítáme dvěma kvadraturními formulemi: základní formulí $Q_1(a, b)$ a přesnější formulí $Q_2(a, b)$. Když si představíme, že formule Q_2 je zcela přesná, můžeme chybu $I(a, b) - Q_1(a, b)$ aproximovat výrazem $Q_2(a, b) - Q_1(a, b)$. Proto, je-li $|Q_2(a, b) - Q_1(a, b)| \le \varepsilon$, kde ε je zadaná přesnost, považujeme $Q(a, b, \varepsilon) := Q_2(a, b)$ za přibližnou hodnotu integrálu $I(a, b)$. V opačném případě, tj. pro $|Q_2(a, b) - Q_1(a, b)| > \varepsilon$, interval $\langle a, b\rangle$ rozdělíme na dva stejně dlouhé intervaly $\langle a, c\rangle$ a $\langle c, b\rangle$, kde $c = (a + b)/2$, na těchto intervalech spočteme nezávisle na sobě přibližné hodnoty $Q_{ac} := Q(a, c, \hat{\varepsilon})$ a $Q_{cb} := Q(c, b, \hat{\varepsilon})$ integrálů $I(a, c)$ a $I(c, b)$, a nakonec položíme $Q(a, b, \varepsilon) := Q_{ac} + Q_{cb}$. Algoritmus je rekurzivní: výpočet $Q(a, c, \hat{\varepsilon})$ a $Q(c, b, \hat{\varepsilon})$ na dceřiných intervalech $\langle a, c\rangle$ a $\langle c, b\rangle$ probíhá analogicky jako výpočet $Q(a, b, \varepsilon)$ na rodičovském intervalu $\langle a, b\rangle$.

Šikovnou implementaci adaptivní integrace dostaneme, když základní formule Q_1 je složená Simpsonova formule Q_S^4 a přesnější formule Q_2 je Booleova formule $Q_B \equiv Q_B^4$. Obě formule používají stejné uzly: krajní body a, b, střed $c = \frac{1}{2}(a + b)$ a body $d = a + \frac{1}{4}(b - a)$ v jedné čtvrtině a $e = b - \frac{1}{4}(b - a)$ ve třech čtvrtinách intervalu $\langle a, b\rangle$. Na dceřiném

intervalu proto stačí dopočítat jen dvě hodnoty $f(d)$ a $f(e)$, neboť zbývající tři hodnoty $f(a)$, $f(b)$ a $f(c)$ se převezmou z rodičovského intervalu.

Protože délka dceřiného intervalu je rovna polovině délky intervalu rodičovského, zdá se přirozené volit $\hat{\varepsilon} = \frac{1}{2}\varepsilon$. Teoretická analýza i praktické zkušenosti však potvrdily, že stačí uvažovat $\hat{\varepsilon} = \varepsilon$. Podrobný popis tohoto algoritmu může čtenář najít např. v [15] nebo [11], viz také funkce `integral` nebo `quadgk` v MATLABu. Hruhý popis zaznamenává následující

algoritmus ADAPT:

function $Q(f, a, b, \varepsilon)$;
  $I_1 := Q_1(f, a, b)$; { Simpsonova formule Q_S^4 }
  $I_2 := Q_2(f, a, b)$; { Booleova formule Q_B^4 }
  $c := (a + b)/2$; { střed intervalu $\langle a, b\rangle$ }
  if abs $(I_2 - I_1) < \varepsilon$ **then** { je dosažena požadovaná přesnost ? }
    $Q := I_2$ { ano, hodnota I_2 se akceptuje }
  else { ne, rekurzivní volání funkce Q na dceřin- }
    $Q := Q(f, a, c, \varepsilon) + Q(f, c, b, \varepsilon)$; { ných subintervalech $\langle a, c\rangle$ a $\langle c, b\rangle$ }

Kvůli jednoduchosti jsme do algoritmu ADAPT nezahrnuli přenos funkčních hodnot $f(a)$, $f(d)$, $f(c)$, $f(e)$ a $f(b)$ (používají se při vyhodnocení formulí Q_1 a Q_2) z rodičovského intervalu $\langle a, b\rangle$ do dceřiných intervalů $\langle a, c\rangle$ a $\langle c, b\rangle$.

Podmíněnost numerického výpočtu integrálu. Ukážeme, že výpočet podle kvadraturní formule (4.29) s kladnými koeficienty w_i je dobře podmíněná úloha. Pro jednoduchost se omezíme jen na prozkoumání vlivu nepřesností ve funkčních hodnotách, tj. nebudeme uvažovat zaokrouhlovací chyby vznikající při provádění aritmetických operací v průběhu vyhodnocování formule.

Předpokládejme tedy, že místo přesných hodnot $f(x_i)$ dosadíme do formule přibližné hodnoty $\tilde{f}(x_i) = f(x_i) + \varepsilon_i$. Pak

$$\tilde{Q}(f) := \sum_{i=0}^{n} w_i \tilde{f}(x_i) = \sum_{i=0}^{n} w_i f(x_i) + \sum_{i=0}^{n} w_i \varepsilon_i = Q(f) + \sum_{i=0}^{n} w_i \varepsilon_i \,.$$

V dalším využijeme toho, že každá smysluplná formule integruje přesně konstantní funkci. Pak ale $\sum_{i=0}^{n} w_i = Q(1) = \int_a^b 1 \,\mathrm{d}x = b - a$. Když označíme $\varepsilon = \max_i |\varepsilon_i|$ velikost maximální chyby ve funkčních hodnotách, dostaneme pro chybu kvadraturní formule odhad

$$|\tilde{Q}(f) - Q(f)| = \Big| \sum_{i=0}^{n} w_i \varepsilon_i \Big| \le \sum_{i=0}^{n} w_i |\varepsilon_i| \le \varepsilon \sum_{i=0}^{n} w_i = (b - a)\varepsilon \,.$$

Odtud je vidět, že když jsou chyby ε_i malé, je chyba kvadraturní formule také malá.

Numerický výpočet integrálu funkce dvou proměnných. V odstavci 3.1.3 jsme uvedli dva příklady, jak funkci f v oblasti Ω aproximovat interpolantem S. Integrací funkce S přes oblast Ω pak dostaneme složenou kvadraturní formuli $Q(f) = \int_\Omega S(x, y) \,\mathrm{d}x\,\mathrm{d}y$ aproximující $I(f) = \int_\Omega f(x, y) \,\mathrm{d}x\,\mathrm{d}y$.

V případě po částech lineární interpolace

$$Q(f) = \int_\Omega S(x,y)\,\mathrm{d}x\,\mathrm{d}y = \sum_{k=1}^{m} \int_{T_k} S_k(x,y)\,\mathrm{d}x\,\mathrm{d}y\,,$$

kde S_k je lineární polynom jednoznačně určený hodnotami funkce f ve vrcholech trojúhelníka T_k a $\Omega = T_1 \cup T_2 \cup \cdots \cup T_m$. Snadným výpočtem dostaneme

$$\int_{T_k} S_k(x,y)\,\mathrm{d}x\,\mathrm{d}y = \tfrac{1}{3}|T_k| \cdot [f(A_k) + f(B_k) + f(C_k)]\,,$$

kde $|T_k|$ je obsah trojúhelníka T_k a $f(A_k)$, $f(B_k)$, $f(C_k)$ jsou hodnoty funkce f ve vrcholech A_k, B_k, C_k trojúhelníka T_k. Pro $f \in C^2(\Omega)$ je $|I(f)-Q(f)| \le Ch^2$, kde h je nejdelší strana trojúhelníků T_k a C je konstanta nezávislá na h. Vidíme tedy, že chyba je libovolně malá, když zvolíme dostatečně jemnou triangulaci oblasti Ω.

V případě bilineární interpolace postupujeme podobně:

$$Q(f) = \int_\Omega S(x,y)\,\mathrm{d}x\,\mathrm{d}y = \sum_{i=1}^{n}\sum_{j=1}^{m} \int_{R_{ij}} S_{ij}(x,y)\,\mathrm{d}x\,\mathrm{d}y\,,$$

kde S_{ij} je bilineární polynom jednoznačně určený hodnotami funkce f ve vrcholech čtyřúhelníka R_{ij} a $\Omega = R_{11} \cup R_{12} \cup \cdots \cup R_{nm}$. Na obdélníku R_{ij} pak snadno odvodíme

$$\int_{R_{ij}} S_{ij}(x,y)\,\mathrm{d}x\,\mathrm{d}y = \tfrac{1}{4}|R_{ij}| \cdot (f_{i-1,j-1} + f_{i,j-1} + f_{i-1,j} + f_{ij})\,,$$

kde $|R_{ij}| = (x_i - x_{i-1})(y_j - y_{j-1})$ je obsah obdélníka R_{ij}. Pokud $f \in C^2(\Omega)$, pak pro chybu platí $|I(f)-Q(f)| \le C(h^2+k^2)$, kde $h = \max_i(x_i - x_{i-1})$, $k = \max_j(y_j - y_{j-1})$ a kde C je konstanta nezávislá na h, k. Chybu zřejmě opět učiníme libovolně malou, pokud obdélník Ω rozdělíme dostatečně jemně.

4.4. Cvičení

4.1. Naprogramujte Richardsonovu extrapolaci z příkladu 4.1. Spočtěte tímto programem derivaci $f'(1{,}5)$ pro funkce (a) $x\sin x$, (b) $\frac{\sin x}{x}$. Zvolte startovací krok $h_0 = 1$ a tolerance $\varepsilon_r = \varepsilon_a = 10^{-5}$.

[(a) $T_{00} = 0{,}628234$ $T_{11} = 1{,}093421$ $T_{22} = 1{,}103579$ $T_{32} = 1{,}103600$ $\boxed{T_{33} = 1{,}103601}$;
(b) $T_{00} = -0{,}359731$ $T_{11} = -0{,}395853$ $T_{22} = -0{,}396173$
$T_{31} = -0{,}396172$ $\boxed{T_{32} = -0{,}396173}$.]

4.2. Ověřte, že pro druhou centrální diferenci (4.4) platí rozvoj (4.6), konkrétně

$$F(h) := \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = f''(x) + \frac{1}{4!}h^2 f^{(4)}(x) + \frac{1}{6!}h^4 f^{(6)}(x) + \dots\,. \tag{4.30}$$

Naprogramujte Richardsonovu extrapolaci (4.11)-(4.13) pro $F(h)$ podle (4.30). Spočtěte tímto programem druhou derivaci $f''(0{,}5)$ pro funkce (a) $x\cos x$, (b) $\frac{\mathrm{tg}\,x}{x}$. Zvolte startovací krok $h_0 = 0{,}8$ a tolerance $\varepsilon_r = \varepsilon_a = 10^{-5}$.

[(a) $T_{00} = -1{,}275680$ $T_{11} = -1{,}396717$ $T_{22} = -1{,}397641$ $T_{31} = -1{,}397639$ $\boxed{T_{32} = -1{,}397642}$;
(b) $T_{00} = 2{,}526186$ $T_{11} = 0{,}977209$ $T_{22} = 1{,}198335$ $T_{33} = 1{,}190579$
$T_{42} = 1{,}190647$ $\boxed{T_{43} = 1{,}190646}$.]

4.3. Pomocí vzorců (4.24'), (4.26') a (4.28') odhadněte potřebné počty dílků n pro dosažení přesnosti $\varepsilon = 10^{-6}$ pro integrál $\int_0^5 xe^{-x^2}\,\mathrm{d}x$. Maxima M_2 a M_4 odhadněte graficky.

[$M_2 \approx 2$, $M_4 \approx 16{,}4$; $n = 4565$, pro složenou lichoběžníkovou formuli, $n = 3228$ pro složenou obdélníkovou formuli, $n = 130$ pro složenou Simpsonovu formuli.]

4.4. Naprogramujte Rombergovu integraci. Spočtěte tímto programem určité integrály (a) $\int_0^2 e^{-x^2}\,\mathrm{d}x$, (b) $\int_0^{\pi/2} \frac{\sin x}{x}\,\mathrm{d}x$. Začněte s jedním dílkem, tj. $n_0 = 1$, a zvolte tolerance $\varepsilon_a = \varepsilon_r = 10^{-6}$.

[(a) $T_{00} = 1{,}018316$ $T_{11} = 0{,}829944$ $T_{22} = 0{,}885270$ $T_{33} = 0{,}882032$
$T_{41} = 0{,}882080$ $\boxed{T_{42} = 0{,}882081}$;
(b) $T_{00} = 1{,}285398$ $T_{11} = 1{,}371275$ $T_{22} = 1{,}370761$ $T_{32} = 1{,}370762$ $\boxed{T_{33} = 1{,}370762}$.]

4.5. Pomocí vzorců (4.17) a (4.19) dokažte, že formule $Q_\square(f) = \frac{2}{3}Q_M(f) + \frac{1}{3}Q_T(f)$ je nejméně řádu 2. Dále pak ukažte, že $Q_\square(f)$ je ve skutečnosti Simpsonova formule řádu 3.

4.6. Ověřte, že Gaussova formule Q_{G1} je řádu 3 a Q_{G2} je řádu 5. (Návod: prověřte, že $\int_{-1}^1 x^i\,\mathrm{d}x = Q_{G1}(x^i)$ pro $i = 0, 1, 2, 3$, a že $\int_0^1 x^i\,\mathrm{d}x = Q_{G2}(x^i)$ pro $i = 0, 1, 2, 3, 4, 5$.)

4.7. Ověřte, že když v Rombergově metodě zvolíme $T_{00} = Q_T(f)$, dostaneme $T_{11} = Q_S(f)$ a $T_{22} = Q_B(f)$.

4.8. Naprogramujte algoritmus ADAPT včetně přenosu již vypočtených funkčních hodnot z mateřského intervalu do dceřiných intervalů. Pro funkci $f(x) = \sin x^2$ a přesnost $\varepsilon = 10^{-6}$ spočtěte (a) $\int_0^\pi f(x)\,\mathrm{d}x$, (b) délku křivky $[x, f(x)]$, $x \in \langle 0, \pi\rangle$, tj. $\int_0^\pi \sqrt{1 + (f'(x))^2}\,\mathrm{d}x$. Určete počet `pf` vyhodnocení integrandu.

[(a) $I \doteq 0{,}772652$, `pf` $= 89$, (b) $I \doteq 7{,}562235$, `pf` $= 153$.]

4.9. Číslo π lze definovat jako obsah jednotkového kruhu $\{[x, y];\, x^2 + y^2 \le 1\}$. Nakreslete si a ověřte, že

$$(i)\quad \pi = 4\int_{-\sqrt{2}/2}^{\sqrt{2}/2} \sqrt{1 - x^2}\,\mathrm{d}x - 2\,, \qquad (ii)\quad \pi = 4\iint_\Omega \mathrm{d}x\,\mathrm{d}y\,,$$

kde $\Omega = \{[x, y];\, x^2 + y^2 \le 1,\ |x| \le y\}$. (a) Určete π pomocí vzorce (i), integrál počítejte algoritmem ADAPT s přesností $\varepsilon = 10^{-6}$. Kolik dělicích bodů algoritmus ADAPT potřebuje? (b) Určete π pomocí vzorce (ii). Dvojný integrál počítejte složenou kvadraturní formulí. Oblast Ω triangulujte jednoduchým způsobem: $\Omega = \bigcup_{k=1}^m T_k$, kde T_k je trojúhelník s vrcholy $[0, 0]$, $[x_k, f(x_k)]$, $[x_{k+1}, f(x_{k+1})]$, $f(x) = \sqrt{1 - x^2}$ a $x_k = -\sqrt{2}/2 + \sqrt{2}(k - 1)/m$, $k = 1, 2, \ldots, m + 1$. (nakreslete!) Obsah $|T|$ trojúhelníka T s vrcholy $[x_a, y_a]$, $[x_b, y_b]$, $[x_c, y_c]$ se počítá podle vzorce

$$|T| = \tfrac{1}{2}\,|d|\,, \qquad \text{kde} \qquad d = \begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix}.$$

Kolik trojúhelníků musíte zvolit k dosažení stejné přesnosti $\varepsilon = 10^{-6}$? Který z obou algoritmů je vhodnější?

[(a) 25, (b) 1155.]

5. Řešení nelineárních rovnic

Kořeny nelineární rovnice $f(x) = 0$ obecně neumíme vyjádřit explicitním vzorcem. K řešení nelineární rovnice proto používáme iterační metody: z jedné nebo několika počátečních aproximací hledaného kořene x^* generujeme posloupnost $x_0, x_1, x_2, \ldots$, která ke kořenu x^* konverguje. Pro některé metody stačí, když zadáme interval $\langle a, b\rangle$, který obsahuje hledaný kořen. Jiné metody vyžadují, aby počáteční aproximace byla k hledanému kořenu dosti blízko; na oplátku takové metody konvergují mnohem rychleji. Často proto začínáme s hrubou, avšak spolehlivou metodou, a teprve když jsme dostatečně blízko kořene, přejdeme na jemnější, rychleji konvergující metodu.

Abychom naše úvahy zjednodušili, omezíme se na problém určení reálného *jednoduchého kořene* x^* rovnice $f(x) = 0$, tj. předpokládáme, že $f'(x^*) \neq 0$. Budeme také automaticky předpokládat, že funkce $f(x)$ je spojitá a má tolik spojitých derivací, kolik je jich v dané situaci zapotřebí.

5.1. Určení počáteční aproximace

Počáteční aproximaci kořenů rovnice $f(x) = 0$ můžeme zjistit z grafu funkce $f(x)$: ručně, nebo raději pomocí vhodného programu na počítači, vykreslíme funkci $f(x)$ a vyhledáme její průsečíky s osou x.

Jinou možností je sestavení tabulky $[x_i, f(x_i)]$ pro nějaké dělení

$$a = x_0 < x_1 \cdots < x_{i-1} < x_i < \cdots < x_n = b$$

zvoleného intervalu $\langle a, b\rangle$. Když ve dvou sousedních bodech tabulky nabývá funkce $f(x)$ hodnot s opačným znaménkem, tj. když $f(x_{i-1})f(x_i) < 0$, pak mezi body x_{i-1} a x_i leží reálný kořen rovnice $f(x) = 0$.

Příklad 5.1. Získáme hrubý odhad kořenů rovnice $f(x) = 0$, kde

$$f(x) = 4\sin x - x^3 - 1\,.$$

Z obrázku 5.1 zjistíme, že existují tři kořeny: $x_1^* \in (-2, -1)$, $x_2^* \in (-1, 0)$ a $x_3^* \in (1, 2)$. □

Na principu znaménkových změn je založena

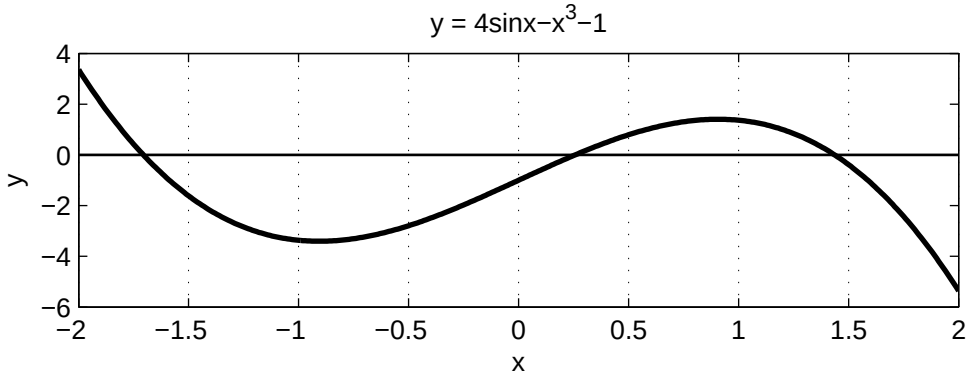
Metoda bisekce známá také jako *metoda půlení intervalů*. Předpokládejme, že funkce $f(x)$ má v koncových bodech intervalu (a_0, b_0) opačná znaménka, tj. platí $f(a_0)f(b_0) < 0$. Sestrojíme posloupnost intervalů $(a_1, b_1) \supset (a_2, b_2) \supset (a_3, b_3) \supset \ldots$, které obsahují kořen. Intervaly (a_{k+1}, b_{k+1}), $k = 0, 1, \ldots$, určíme rekurzivně způsobem, který si nyní popíšeme.

Střed intervalu (a_k, b_k) je bod $x_{k+1} = \frac{1}{2}(a_k + b_k)$. Když $f(x_{k+1}) = 0$, pak $x_{k+1} = x^*$ je kořen a dál nepokračujeme. Pokud $f(x_{k+1}) \neq 0$, položíme

$$(a_{k+1}, b_{k+1}) = \begin{cases} (a_k, x_{k+1})\,, & \text{když} \quad f(a_k)f(x_{k+1}) < 0\,, \\ (x_{k+1}, b_k)\,, & \text{když} \quad f(a_k)f(x_{k+1}) > 0\,. \end{cases} \tag{5.1}$$

Z konstrukce (a_{k+1}, b_{k+1}) okamžitě plyne $f(a_{k+1})f(b_{k+1}) < 0$, takže každý interval (a_k, b_k) obsahuje kořen. Po k krocích je kořen v intervalu $I_k := (a_k, b_k)$ délky

$$|I_k| = b_k - a_k = 2^{-1}(b_{k-1} - a_{k-1}) = \cdots = 2^{-k}(b_0 - a_0)\,.$$



Obr. 5.1: Graf funkce $4\sin x - x^3 - 1$

Střed x_{k+1} intervalu (a_k, b_k) aproximuje kořen x^* s chybou

$$|x_{k+1} - x^*| \leq \tfrac{1}{2}(b_k - a_k) = 2^{-k-1}(b_0 - a_0)\,. \tag{5.2}$$

Pro $k \to \infty$ zřejmě $|I_k| \to 0$ a $x_k \to x^*$.

Příklad 5.2. Metodu bisekce aplikujeme na rovnici z příkladu 5.1. Jako počáteční zvolíme interval $(a_0, b_0) = (1, 2)$. Připomeňme, že $f(1) > 0$, $f(2) < 0$. Proto také $f(a_k) \geq 0$, $f(b_k) \leq 0$ pro každé k. Posloupnost intervalů zaznamenáváme do tabulky. Po pěti krocích má interval (a_5, b_5) délku $2^{-5} = 0{,}03125$ a $x_6 = 1{,}421875$ aproximuje kořen s chybou nepřesahující $2^{-6} = 0{,}015625$. □

k	a_k	b_k	x_{k+1}	$f(x_{k+1})$
0	1	2	1,5	< 0
1	1	1,5	1,25	> 0
2	1,25	1,5	1,375	> 0
3	1,375	1,5	1,4375	< 0
4	1,375	1,4375	1,40625	> 0
5	1,40625	1,4375	1,421875	

Metoda bisekce konverguje pomalu: protože $10^{-1} \doteq 2^{-3{,}32}$, zpřesnění o jednu dekadickou cifru vyžaduje v průměru 3,32 kroku. Všimněte si, že rychlost konvergence vyjádřená vztahem (5.2) vůbec nezávisí na funkci $f(x)$. To proto, že jsme využívali pouze znaménka funkčních hodnot. Když tyto hodnoty (a případně také hodnoty derivací $f(x)$) využijeme efektivněji, můžeme dosáhnout podstatně rychlejší konvergence. Takové zpřesňující metody však konvergují pouze tehdy, když pro ně zvolíme dostatečně dobrou počáteční aproximaci. Vhodná počáteční aproximace bývá často určena právě metodou bisekce.

5.2. Zpřesňující metody

Snad nejznámější mezi nimi je

Newtonova metoda nebo-li *metoda tečen*. Jak je u iteračních metod zvykem, vyjdeme z počáteční aproximace x_0 a postupně počítáme $x_1, x_2, \ldots$ způsobem, který si teď vysvětlíme.

Předpokládejme, že známe x_k a máme určit lepší aproximaci x_{k+1}. Uděláme to tak, že 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

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

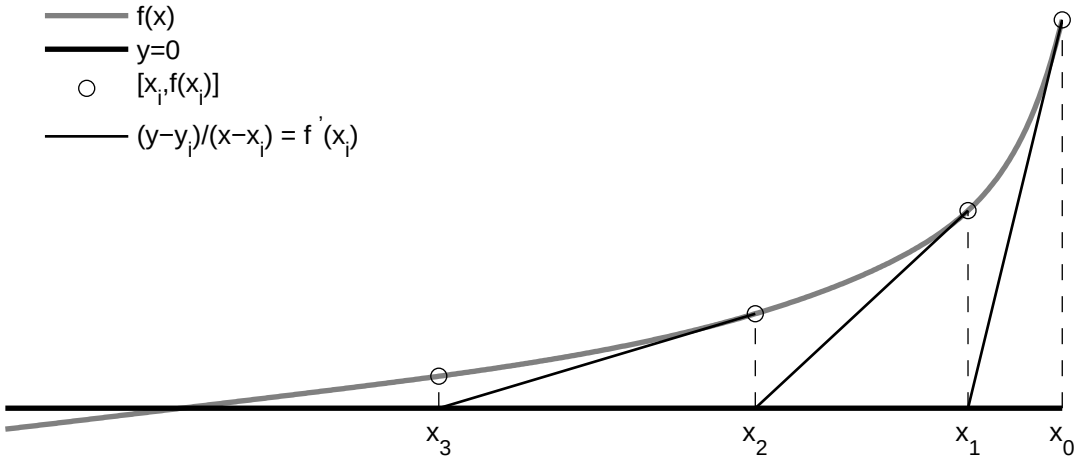
tedy dosadíme $y := 0$, vypočteme x a položíme $x_{k+1} := x$. Tak dostaneme předpis

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}\,. \tag{5.3}$$

Výpočet ukončíme a x_{k+1} považujeme za dostatečně přesnou aproximaci kořene, pokud

$$|x_{k+1} - x_k| \le \varepsilon\,, \quad \text{případně} \quad |x_{k+1} - x_k| \le \varepsilon|x_k| \quad \text{nebo} \quad |f(x_{k+1})| \le \varepsilon\,, \tag{5.4}$$

kde ε je požadovaná přesnost. Tím sice není zaručeno, že také $|x_{k+1} - x^*| \le \varepsilon$, je to ale obvyklý způsob, pomocí něhož iterace ukončíme. Tato tzv. *stop kritéria* jsou vhodná i pro další metody, které v tomto odstavci uvedeme.



Obr. 5.2: Newtonova metoda

Příklad 5.3. Newtonovou metodou určíme kladný kořen rovnice z příkladu 5.1. Zvolíme $x_0 = 2$. Výpočet ukončíme, když $|f(x_k)| < 10^{-5}$. Poslední sloupec vyžaduje znalost přes-

k	x_k	$f(x_k)$	$f'(x_k)$	$f(x_k)/f'(x_k)$	$x_k - x^*$
0	2	−5,362810	−13,66459	0,392460	0,563550
1	1,607540	−1,156877	−7,899490	0,146450	0,171089
2	1,461090	−0,143158	−5,966406	0,023994	0,024640
3	1,437096	−0,003653	−5,662524	0,000645	0,000646
4	1,436451	−0,000003			0,000000

ného řešení. To získáme provedením ještě jednoho kroku Newtonovy metody. Dá se ukázat, že $x^* \doteq x_5 = 1{,}43645032$ má všechny cifry platné. Požadovaná přesnost byla tedy dosažena ve čtvrtém kroku, $x_4 \doteq 1{,}43645$ má všechny cifry platné. □

Konvergence Newtonovy metody. Nechť $e_k = x_k - x^*$ je chyba v k-tém kroku. Ukážeme si, jak souvisí s chybou e_{k+1} v kroku následujícím. Z Taylorova rozvoje $f(x^*)$ okolo x_k dostaneme

$$0 = f(x^*) = f(x_k) + (x^* - x_k)f'(x_k) + \frac{1}{2}(x^* - x_k)^2 f''(\xi)\,,$$

kde ξ je nějaký blíže neurčený bod intervalu, jehož krajní body jsou x_k a x^*. Když rovnici dělíme $f'(x_k)$, dostaneme

$$-\frac{1}{2}(x^* - x_k)^2 \frac{f''(\xi)}{f'(x_k)} = \frac{f(x_k)}{f'(x_k)} + (x^* - x_k) = x^* - \left[x_k - \frac{f(x_k)}{f'(x_k)}\right] = x^* - x_{k+1}\,,$$

takže máme

$$e_{k+1} = \frac{1}{2}\,\frac{f''(\xi)}{f'(x_k)}\,e_k^2\,, \tag{5.5}$$

a když $x_k \to x^*$, pak

$$\frac{e_{k+1}}{e_k^2} \longrightarrow C\,, \qquad \text{kde} \quad C = \frac{1}{2}\frac{f''(x^*)}{f'(x^*)}\,.$$

Protože chyba e_{k+1} je úměrná druhé mocnině chyby e_k, říkáme, že Newtonova metoda *konverguje kvadraticky* nebo také, že je *druhého řádu*. Uveďme si přesnější definici:

Nechť $x_0, x_1, x_2, \ldots$ je posloupnost, která konverguje k x^* a $e_k = x_k - x^*$. Když existuje číslo p a konstanta $C \neq 0$ taková, že

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = C\,, \tag{5.6}$$

pak p se nazývá řád konvergence posloupnosti a C je chybová konstanta. Speciálně říkáme, že

	lineární,		$p = 1 \quad a \quad C < 1\,,$
konvergence je	*superlineární,*	*když*	$p > 1\,,$
	kvadratická,		$p = 2\,.$

Řekneme, že daná metoda je řádu p, jestliže všechny konvergentní posloupnosti získané touto metodou mají řád konvergence větší nebo rovný p a nejméně jedna z těchto posloupností má řád konvergence rovný přesně p.

V blízkosti kořene platí: čím vyšší řád p, tím rychlejší konvergence, neboť

$$|e_{k+1}| \approx C|e_k|^p\,,$$

takže když $|e_k|$ je malé, pak $|e_{k+1}|$ je tím menší, čím je p větší.

Víme už, že když Newtonova metoda konverguje, pak rychlost konvergence $x_k \to x^*$ je alespoň kvadratická (pro některé funkce f může být i vyšší). Zbývá ještě zodpovědět otázku, za jakých podmínek je zaručeno, že konvergence vůbec nastane. Ukažme si to.

Předpokládejme, že v nějakém okolí I kořene platí

$$\frac{1}{2}\left|\frac{f''(y)}{f'(x)}\right| \le m \qquad \text{pro všechna} \qquad x, y \in I\,.$$

Když $x_k \in I$, pak z (5.5) plyne $|e_{k+1}| \le m|e_k|^2$ nebo-li $|me_{k+1}| \le |me_k|^2$. Opakováním této úvahy dostaneme

$$|me_{k+1}| \le |me_k|^2 \le |me_{k-1}|^4 \le |me_{k-2}|^8 \le |me_{k-3}|^{16} \le \cdots \le |me_0|^r\,, \text{kde } r = 2^{k+1}.$$

Když platí $|me_0| < 1$, pak jistě $|e_{k+1}| \to 0$ a tedy $x_{k+1} \to x^*$. Dokázali jsme tedy, že

Newtonova metoda vždy konverguje za předpokladu, že počáteční aproximaci zvolíme dostatečně blízko ke kořenu.

Dobrou počáteční aproximaci x_0 můžeme získat např. metodou bisekce. Vhodným spojením metody bisekce a Newtonovy metody lze sestrojit *kombinovanou metodu*, která vždy konverguje, viz např. procedura `rtsafe` v [18]. V blízkosti kořene se přitom uplatní jen Newtonova metoda, takže konvergence je rychlá.

Pomocí náčrtku snadno ověříme, že Newtonova metoda konverguje, když jsou splněny tzv. *Fourierovy podmínky*:

a) $f \in C^2\langle a, b\rangle$ a přitom $f(a)f(b) < 0$;

b) f' a f'' nemění na intervalu $\langle a, b\rangle$ znaménko a $f'(x) \neq 0$ pro každé $x \in \langle a, b\rangle$;

c) jako x_0 volíme ten z bodů a, b, v němž je $f(x_0)f''(x_0) > 0$.

Praktický význam však Fourierovy podmínky nemají, neboť pro velké $b - a$ obvykle tyto podmínky buďto neplatí nebo je neumíme snadno ověřit.

Metoda sečen. V každém kroku Newtonovy metody musíme počítat hodnotu $f(x_k)$ a derivaci $f'(x_k)$. Když vzorec pro výpočet derivace nemáme k dispozici, nebo když náklady spojené s výpočtem derivace jsou vysoké, můžeme derivaci aproximovat podílem

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

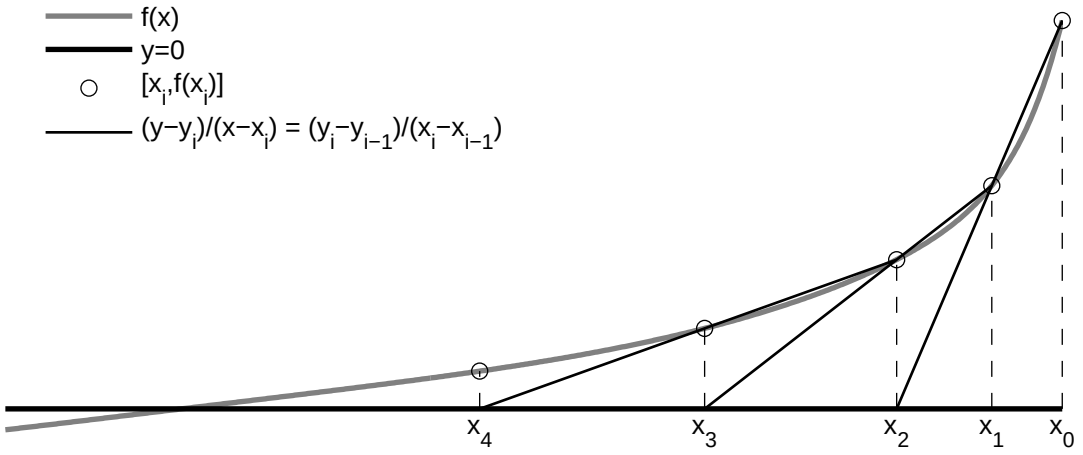
Tak dostaneme *metodu sečen*: zadáme dvě počáteční aproximace x_0, x_1 a počítáme $x_2, x_3, \ldots$ podle předpisu

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)\,. \tag{5.7}$$

Název metody vychází z její geometrické interpretace: x_{k+1} je x-ová souřadnice průsečíku přímky procházející body $[x_{k-1}, f(x_{k-1})]$ a $[x_k, f(x_k)]$ s osou x:

$$y = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}\,(x - x_k) = 0 \qquad \Longrightarrow \qquad x = x_{k+1}\,.$$

Protože tato přímka protíná graf funkce f, je to sečna, odtud metoda sečen.



Obr. 5.3: Metoda sečen

Všimněte si, že v každém kroku vyčíslujeme hodnotu funkce jen jednou: vypočteme $f(x_k)$, hodnotu $f(x_{k-1})$ převezmeme z předchozího kroku.

Dá se odvodit, že rychlost konvergence metody sečen je řádu $p = \frac{1}{2}(1+\sqrt{5}) \approx 1{,}618$, tedy poněkud nižší než u Newtonovy metody. Číslo $\tau = (\sqrt{5}-1)/2 \approx 0{,}618$ je tzv. poměr zlatého řezu. Toto magické číslo se znovu objeví v odstavci 6.1, kde si o něm řekneme trochu víc (viz poznámka o zlatém řezu).

Příklad 5.4. Metodou sečen určíme kladný kořen rovnice z příkladu 5.1. Zvolíme $x_0 = 1$, $x_1 = 2$. Výpočet ukončíme, když bude $|f(x_k)| < 10^{-5}$. Až do čtvrtého kroku (výpočet x_5) je konvergence poměrně pomalá. Teprve v posledních dvou krocích se plně uplatnila rychlá konvergence metody sečen. □

k	x_k	$f(x_k)$	$x_k - x^*$
0	1	1,365884	−0,436450
1	2	−5,362810	0,563550
2	1,202994	0,991513	−0,233456
3	1,327357	0,543420	−0,109094
4	1,478177	−0,246970	0,041726
5	1,431051	0,030349	−0,005400
6	1,436208	0,001370	−0,000242
7	1,436452	−0,000008	0,000001

Metoda sečen zaručeně konverguje, pokud zvolíme startovací hodnoty x_0 a x_1 dostatečně blízko ke kořenu x^*. To lze zajistit např. metodou bisekce. Další metodou, jak získat dobré startovací aproximace, je varianta metody sečen známá jako

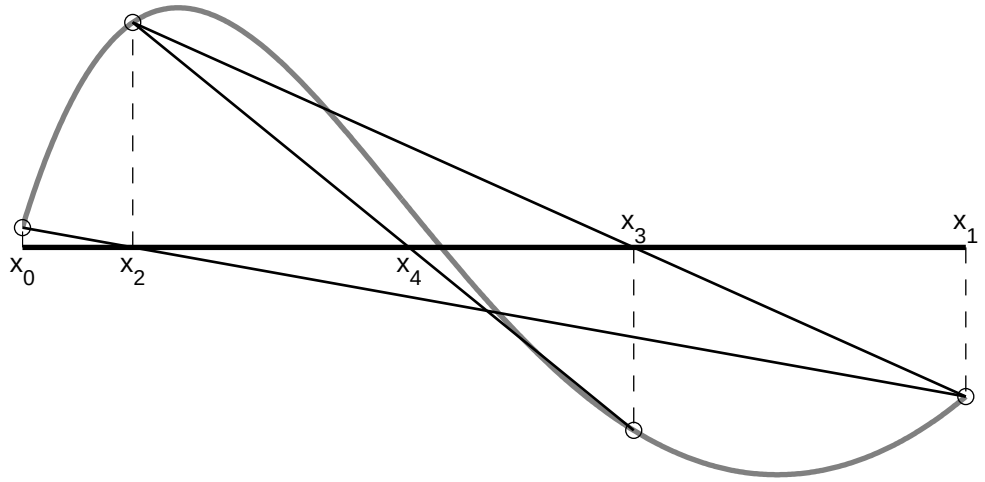
Metoda regula falsi. Počáteční aproximace x_0 a x_1 se volí tak, aby $f(x_0)f(x_1) < 0$. Nová aproximace x_{k+1} se opět získá jako průsečík sečny s osou x. Sečna však tentokrát spojuje bod $[x_k, f(x_k)]$ s bodem $[x_\ell, f(x_\ell)]$, kde ℓ je největší index, pro který $f(x_k)f(x_\ell) < 0$. Výpočet tedy probíhá podle vzorce

$$x_{k+1} = x_k - \frac{x_k - x_\ell}{f(x_k) - f(x_\ell)} f(x_k)\,, \qquad k = 1, 2, \ldots. \tag{5.8}$$

Přitom pro $k = 1$ je $\ell = 0$, a po výpočtu x_{k+1} určíme index ℓ takto:

když $f(x_{k+1})f(x_\ell) > 0$, pak $\ell = k$, v opačném případě se ℓ nemění.

Výhodou metody regula falsi je to, že podobně jako metoda bisekce vždy konverguje: interval I_k, jehož koncové body jsou x_k a x_ℓ, obsahuje kořen. Na rozdíl od metody bisekce však délka intervalu I_k nekonverguje k nule. Rychlost konvergence metody regula falsi je jen lineární. Metodu regula falsi (podobně jako metodu bisekce) proto používáme pouze pro získání dobré počáteční aproximace, pak přecházíme na rychlejší metodu.



Obr. 5.4: Regula falsi

Příklad 5.5. Metodou regula falsi určíme kladný kořen rovnice z příkladu 5.1. Zvolíme $x_0 = 1$, $x_1 = 2$. Z tabulky je vidět, že počínaje druhým krokem je $x_\ell = 2$. Dále je zřejmé, že do čtvrtého kroku je přesnost metody regula falsi srovnatelná s přesností metody sečen, viz příklad 5.4. V následujících krocích je už ale patrná lineární konvergence, podmínka $|f(x_k)| < 10^{-5}$ je splněna až pro x_{16}. Všimněte si: délka intervalů $I_k = (x_k, 2)$, $k \geq 2$, konverguje k číslu $x - x^* \doteq 0,563550$. □

k	ℓ	x_ℓ	x_k	$f(x_k)$	$x_k - x^*$
0			1	1,365884	−0,436450
1	0	1	2	−5,362810	0,563550
2	1	2	1,202994	0,991513	−0,233456
3	1	2	1,327357	0,543420	−0,109094
4	1	2	1,389245	0,253012	−0,047205
5	1	2	1,416762	0,108896	−0,019688
6	1	2	1,428369	0,045283	−0,008081
7	1	2	1,433156	0,018561	−0,003295
⋮					
15	1	2	1,436448	0,000014	−0,000002
16	1	2	1,436449	0,000006	−0,000001

Steffensenova metoda se řídí předpisem

$$x_{k+1} = x_k - \frac{f(x_k)}{d_k}, \qquad \text{kde} \quad d_k = \frac{f(x_k + f(x_k)) - f(x_k)}{f(x_k)} \tag{5.9}$$

je speciálně spočtená aproximace $f'(x_k)$ připomínající dopřednou diferenci:

$$f'(x_k) \approx d_k = \frac{f(x_k + h_k) - f(x_k)}{h_k}\,, \qquad \text{kde} \quad h_k = f(x_k)\,.$$

V každém kroku se funkce f vyhodnocuje dvakrát: kromě $h_k = f(x_k)$ se počítá ještě také $f(x_k + h_k)$. Oproti metodě sečen je tu jedno vyhodnocení funkce navíc. Na druhé straně lze ukázat, že rychlost konvergence Steffensenovy metody je stejná jako u Newtonovy metody, tedy kvadratická.

Metoda inverzní kvadratické interpolace. Metoda sečen používá dva předchozí body k získání dalšího, proč tedy nepoužít tři?

Body $[x_{k-2}, f(x_{k-2})]$, $[x_{k-1}, f(x_{k-1})]$ a $[x_k, f(x_k)]$ můžeme proložit parabolu $P_2(x)$ a hledat její průsečík s osou x. Za další aproximaci x_{k+1} pak zvolíme ten z kořenů polynomu $P_2(x)$, který je blíž k předchozí aproximaci x_k. Na tomto principu je založena *Müllerova metoda*. Potíž je v tom, že parabola nemusí x-ovou osu protnout, neboť kvadratická funkce $P_2(x)$ nemusí mít reálné kořeny. Výpočet je proto třeba provádět v komplexní aritmetice, a to i v případě, že rovnice $f(x) = 0$ má jen reálné kořeny.

Místo paraboly v proměnné x můžeme třemi body proložit parabolu $Q_2(y)$ v proměnné y, určenou interpolačními podmínkami

$$Q_2(f(x_{k-2})) = x_{k-2}\,, \quad Q_2(f(x_{k-1})) = x_{k-1}\,, \quad Q_2(f(x_k)) = x_k\,.$$

Jsou-li hodnoty $f(x_{k-2})$, $f(x_{k-1})$ a $f(x_k)$ navzájem různé, parabola $Q_2(y)$ existuje a protíná osu x v jediném bodě. Klademe tedy $x_{k+1} = Q_2(0)$. Tato metoda je známa jako *metoda inverzní kvadratické interpolace*. Její konvergence je superlineární řádu $p \approx 1,839$, viz [7].

Brentova metoda. Metoda inverzní kvadratické interpolace spolu s metodou sečen a metodou bisekce jsou základem populární *Brentovy metody*, viz např. [15], dále také program `zbrent` v [18] nebo funkce `fzero` v MATLABu.

Předností Brentovy metody je to, že nepoužívá derivace funkce f, je spolehlivá, tj. zaručeně konverguje ke kořenu, a po několika počátečních krocích se chyba rychle zmenšuje, neboť rychlost konvergence je superlineární.

Startovací body x_0 a x_1 je třeba zvolit tak, aby $f(x_0)f(x_1) < 0$. Aproximace x_2 se určí metodou sečen. Nechť (a_1, b_1) je interval, jehož koncové body jsou x_0 a x_1. Pak zřejmě $x_2 \in (a_1, b_1)$. Další aproximaci x_3 budeme hledat v kratším intervalu $(a_2, b_2) \subset (a_1, b_1)$, jehož jeden koncový bod je x_2 a druhý je ten z bodů a_1, b_1, v němž má funkce f opačné znaménko než v x_2, takže $f(a_2)f(b_2) < 0$ a (a_2, b_2) obsahuje kořen.

Při výpočtu $x_3, x_4, \ldots$ Brentova metoda používá jednu ze tří základních metod tak, aby nová aproximace $x_{k+1} \in (a_k, b_k)$. Dále se vybere interval $(a_{k+1}, b_{k+1}) \subset (a_k, b_k)$ obsahující kořen. Jedním z jeho koncových bodů je x_{k+1}, druhým je ten z bodů a_k, b_k, v němž má funkce f znaménko opačné než v x_{k+1}. Při výpočtu x_{k+1} se přednostně použije metoda inverzní kvadratické interpolace, pokud takto získaná aproximace není dostatečně dobrá, zkusí se metoda sečen, a když ani ta nezabere, použije se jako záchrana metoda bisekce. Podrobnější popis Brentovy metody je uveden např. v [15], [18].

Příklad 5.6. Budeme hledat kladný kořen rovnice z příkladu 5.1 a porovnáme jednotlivé metody podle počtu `pk` kroků a počtu `pf` vyhodnocení funkce f (u Newtonovy metody

do `pf` zahrneme také počet vyhodnocení derivace f'). Pro výpočet Brentovou metodou jsme použili upravený program `fzerotx`, viz [15].

Výpočet jsme zahájili takto: v metodě bisekce počáteční interval $(a_0, b_0) = (1, 2)$, v Newtonově a Steffensenově metodě $x_0 = 2$, v ostatních metodách $x_0 = 1$ a $x_1 = 2$. Použili jsme stop kritérium $|f(x_k)| < \varepsilon$. V tabulce jsou uvedeny hodnoty `pk/pf` pro několik tolerancí ε.

ε	10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}
bisekce	9/11	19/21	29/31	39/41	49/51
regula falsi	10/12	17/19	25/27	33/35	40/42
sečny	6/8	7/9	8/10	8/10	9/11
Newton	4/8	5/10	5/10	6/12	6/12
Steffensen	4/8	5/10	6/12	6/12	7/14
Brent	6/7	7/8	7/8	8/9	8/9

Nejmenší `pk` má Newtonova metoda, nejmenší `pf` Brentova metoda. Z výpisu o průběhu výpočtu Brentovou metodou vyplývá, že se ani jednou nepoužila bisekce, proto tak skvělý výsledek. □

Poznámka (*O metodě prosté iterace*). Předpokládejme, že funkce $g \in C\langle a, b\rangle$ splňuje tyto dvě podmínky:

(α) $g(x) \in \langle a, b\rangle \qquad \forall x \in \langle a, b\rangle$,

(β) existuje číslo q, $0 \le q < 1$, takové, že $|g(x) - g(y)| \le q|x - y| \qquad \forall x, y \in \langle a, b\rangle$.

Pak rovnice $x = g(x)$ má v $\langle a, b\rangle$ jediné řešení x^* a *posloupnost postupných aproximací* $x_{k+1} = g(x_k)$, $k = 0, 1, \ldots$, konverguje k x^* pro každé $x_0 \in \langle a, b\rangle$. Bod $x^* = g(x^*)$ se nazývá *pevný bod* funkce g (zobrazuje x^* na sebe). Následuje náčrt důkazu.

1) *Existence.* Z podmínky (α) plyne $g(a) \ge a$, $g(b) \le b$, odtud $(a - g(a)) \cdot (b - g(b)) \le 0$, takže v $\langle a, b\rangle$ leží kořen rovnice $x - g(x) = 0$.

2) *Jednoznačnost.* Nechť pro $x^*, y^* \in \langle a, b\rangle$ platí $x^* = g(x^*)$, $y^* = g(y^*)$. Podle (β) $|x^* - y^*| = |g(x^*) - g(y^*)| \le q|x^* - y^*|$, což je možné jedině když $x^* = y^*$.

3) *Konvergence.* Podle (β) je $|x_k - x^*| = |g(x_{k-1}) - g(x^*)| \le q|x_{k-1} - x^*|$. Opakováním této úvahy dostaneme nakonec $|x_k - x^*| \le q^k|x_0 - x^*| \to 0$ pro $k \to \infty$, takže $x_k \to x^*$.

Místo podmínky (β) můžeme pro $g \in C^1\langle a, b\rangle$ použít silnější podmínku

(β') $|g'(x)| \le q < 1 \qquad \forall x \in \langle a, b\rangle$.

Podle věty o střední hodnotě totiž $g(x) - g(y) = g'(\xi)(x - y)$, kde ξ leží mezi x a y, takže pro $x, y \in \langle a, b\rangle$ podle (β') je $|g(x) - g(y)| = |g'(\xi)| \cdot |x - y| \le q|x - y|$, tj. platí ($\beta$).

Všimněte si, že pro $\langle a, b\rangle = \langle x^* - \delta, x^* + \delta\rangle$ je platnost podmínky (α) důsledkem platnosti podmínky (β): $|g(x) - x^*| = |g(x) - g(x^*)| \le q|x - x^*| < |x - x^*|$, tj. když $|x - x^*| \le \delta$, pak také $|g(x) - x^*| \le \delta$.

Přibližný výpočet kořene x^* rovnice $x = g(x)$ podle formule $x_{k+1} = g(x_k)$ se nazývá *metoda prosté iterace* nebo také *metoda postupných aproximací*. Podmínky (α) a (β) nebo (β') jsou postačující pro konvergenci této metody.

Vhodnou úpravou rovnice $f(x) = 0$ na tvar $x = g(x)$ můžeme dostat řadu různých konkrétních metod. Tak třeba pro $g(x) = x - f(x)/f'(x)$ dostaneme Newtonovu metodu.

Rychlost konvergence posloupnosti postupných aproximací $\{x_k\}_{k=0}^{\infty}$ závisí na chování funkce g v bodě x^*. Jsou-li splněny podmínky (α) a (β) nebo (β'), a má-li g dostatečný počet spojitých derivací, dají se dokázat následující tvrzení.

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

Pro Newtonovu metodu $g'(x) = f(x)f''(x)/(f'(x))^2$, tj. $g'(x^*) = 0$, takže konvergence $x_k \to x^*$ je řádu alespoň dva (což potvrzuje nám již známý výsledek).

Dá se také dokázat, že když $|g'(x^*)| > 1$, pak pro $x_0 \neq x^*$ posloupnost postupných aproximací k x^* konvergovat nemůže. □

Příklad 5.7. Nelineární rovnice $f(x) = x^2 - 2x - 3 = 0$ má kořeny $x^* = -1$ a $x^* = 3$. Prozkoumáme konvergenci ke kořenu $x^* = 3$ pro několik iteračních funkcí g.

1. $g(x) = (x^2 - 3)/2$, $g'(x) = x$, $|g'(3)| = 3$, pro $x_0 \neq 3$ konvergence nenastane.
2. $g(x) = \sqrt{2x + 3}$, $g'(x) = 1/\sqrt{2x + 3}$, $|g'(3)| = 1/3$, lineární konvergence nastane např. pro libovolné x_0 z intervalu $\langle 2, 4\rangle$, neboť v něm $|g'(x)| \leq 1/\sqrt{7}$.
3. $g(x) = 2 + 3/x$, $g'(x) = -3/x^2$, $|g'(3)| = 1/3$, lineární konvergence nastane např. pro libovolné x_0 z intervalu $\langle 2, 4\rangle$, neboť v něm $|g'(x)| \leq 3/4$.
4. $g(x) = (x^2+3)/(2x-2)$, $g'(x) = 2(x^2-2x-3)/(2x-2)^2$, $g'(3) = 0$, $g''(3) = 1/2$, kvadratická konvergence nastane např. pro x_0 z intervalu $\langle 2{,}5; 3{,}5\rangle$, v němž $|g'(x)| < 0{,}39$, jak snadno zjistíme. Ověřte, že tato iterační funkce odpovídá Newtonově metodě. □

Poznámka (*O násobných kořenech*). Řekneme, že kořen x^* rovnice $f(x) = 0$ má *násobnost* q, jestliže funkce $g(x) = f(x)/(x - x^*)^q$ je v bodě x^* definována a kořen v něm už nemá, tj. když $0 < |g(x^*)| < \infty$. Jestliže má funkce $f(x)$ v okolí kořene x^* spojité derivace až do řádu q včetně, pak $f^{(j)}(x^*) = 0$, $j = 0, 1, \ldots, q-1$.

Některé z doposud uvedených metod lze použít také pro nalezení násobných kořenů, konvergence však bývá pomalejší. Tak třeba Newtonova metoda konverguje jen lineárně s chybovou konstantou $C = (q-1)/q$.

Když očekáváme, že rovnice $f(x) = 0$ může mít násobné kořeny, je vhodné využít toho, že funkce $u(x) = f(x)/f'(x)$ má pouze jednoduché kořeny. Místo rovnice $f(x) = 0$ tedy řešíme rovnici $u(x) = 0$. □

Poznámka (*O dosažitelné přesnosti*). Nechť x_k je aproximace jednoduchého kořene rovnice $f(x) = 0$. Pomocí věty o střední hodnotě dostaneme

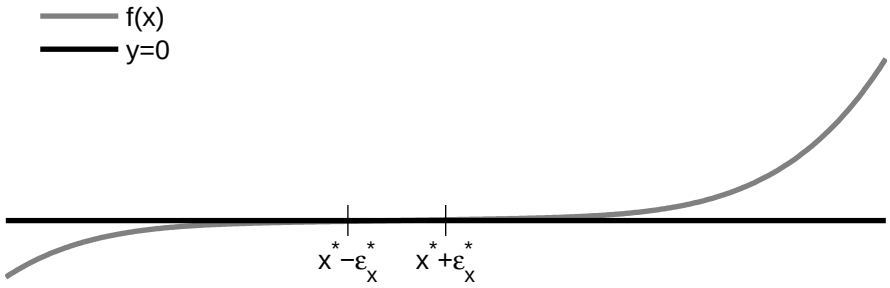
$$f(x_k) = f(x_k) - f(x^*) = f'(\xi)(x_k - x^*)\,,$$

kde ξ je nějaký bod ležící mezi x_k a x^*. Předpokládejme, že při výpočtech pracujeme jen s přibližnými hodnotami $\tilde{f}(x_k) = f(x_k) + \delta_k$, přičemž $|\delta_k| \leq \delta$. Pak nejlepší výsledek,

kterého můžeme dosáhnout, je $\tilde{f}(x_k) = 0$. V tom případě $|f(x_k)| \leq \delta$, takže

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

pokud se f' v blízkosti kořene příliš nemění. Vypočítat x^* s menší chybou než ε_x^* nelze. Proto se ε_x^* nazývá *dosažitelná přesnost kořene* x^*. Všimněte si: když je velikost směrnice $|f'(x^*)|$ v kořenu x^* malá, je dosažitelná přesnost ε_x^* velká, viz obr. 5.5. V takovém případě je výpočet kořene x^* *špatně podmíněný problém*: malá změna f vyvolá velkou změnu x^*.



Obr. 5.5: Dosažitelná přesnost kořene

Podobná úvaha pro kořen násobnosti q dává dosažitelnou přesnost

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

Exponent $1/q$ je příčinou toho, že výpočet násobného kořene je obecně špatně podmíněná úloha. Tak třeba pro $f(x) = x^q$ je $x^* = 0$ kořen násobnosti q a $\varepsilon_x^* = \delta^{1/q}$. Pro $q = 15$ a $\delta = 10^{-15}$ dostaneme $\varepsilon_x^* = 0{,}1$! □

Poznámka (*O kořenech polynomů*). Polynom $p_n(x)$ stupně n má n obecně komlexních kořenů. Pro výpočet jednoduchých reálných kořenů funkce $f(x) = p_n(x)$ lze použít libovolnou z dosud uvedených metod. O tom, jak se vypořádat s případnými násobnými kořeny, pojednává výše uvedená poznámka. Pro výpočet komplexních kořenů lze použít např. Newtonovu metodu, v níž jako počáteční aproximaci volíme komplexní číslo.

Pokud nás zajímají všechny kořeny polynomu, tak po nalezení reálného kořene x^* polynom $p_n(x)$ dělíme členem $x - x^*$. Tak dostaneme polynom $p_{n-1}(x) = p_n(x)/(x - x^*)$ stupně $n - 1$ a dále hledáme jeho kořeny. Když je x^* komplexní kořen, pak je kořenem také komplexně sdružené číslo $\bar{x}^*$. V tom případě dělíme $p_n(x)$ kvadratickým polynomem $(x - x^*)(x - \bar{x}^*)$, jehož koeficienty jsou reálná čísla. Tak dostaneme polynom $p_{n-2}(x)$ stupně $n - 2$ s reálnými koeficienty a pokračujeme hledáním jeho kořenů.

Pro výpočet kořenů polynomů jsou navrženy také speciální, velmi efektivní metody, o nichž lze získat informace např. v [22]. □

5.3. Soustavy nelineárních rovnic

Mnohé z metod určených pro řešení jedné nelineární rovnice lze zobecnit na řešení soustav nelineárních rovnic. Bohužel to neplatí pro metodu bisekce ani pro metodu regula

falsi. A co je ještě horší: pro soustavy nelineárních rovnic není známa žádná univerzální metoda, která by dokázala spolehlivě určit dostatečně dobrou počáteční aproximaci řešení. Uspokojivou počáteční aproximaci proto musíme odhadnout. Pomoci nám může znalost konkrétního problému, který na řešení nelineární soustavy vede. Odhad řešení lze někdy získat také na základě pomocných výpočtů provedených za zjednodušujících předpokladů, například tak, že nelineární problém aproximujeme vhodným problémem lineárním.

Uvažujme tedy soustavu n nelineárních rovnic o n neznámých

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n) = 0\,, \\ &f_2(x_1, x_2, \ldots, x_n) = 0\,, \\ &\vdots \\ &f_n(x_1, x_2, \ldots, x_n) = 0\,, \end{aligned} \qquad \text{nebo-li} \qquad \mathbf{f}(\mathbf{x}) = \mathbf{o}\,, \tag{5.10}$$

kde

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

Řešením soustavy (5.10) je každý číselný vektor $\mathbf{x}^*$, pro který $\mathbf{f}(\mathbf{x}^*) = \mathbf{o}$.

V tomto odstavci budeme automaticky předpokládat, že funkce $\mathbf{f}(\mathbf{x})$ je spojitá a má tolik spojitých derivací, kolik je jich v dané situaci zapotřebí.

Newtonova metoda a její modifikace. Pro $n = 1$ lze Newtonovu metodu odvodit také z Taylorova rozvoje

$$0 = f(x^*) = f(x_k) + f'(x_k)(x^* - x_k) + \text{chyba} \doteq f(x_k) + f'(x_k)(x^* - x_k)$$

tak, že přibližnou rovnost nahradíme rovností a místo x^* píšeme x_{k+1}, tj. počítáme

$$f'(x_k)(x_{k+1} - x_k) = -f(x_k)\,, \qquad k = 0, 1, \ldots.$$

Podobně lze pomocí Taylorovy formule v n dimenzích odvodit Newtonovu metodu pro soustavu (5.10),

$$\mathbf{f}'(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)\,, \tag{5.11}$$

kde $\mathbf{f}'(\mathbf{x})$ je *Jacobiho matice* funkce $\mathbf{f}(\mathbf{x})$, tj.

$$\mathbf{f}'(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \dfrac{\partial f_2(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial f_n(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_n(\mathbf{x})}{\partial x_n} \end{pmatrix}.$$

Výpočet organizujeme tak, že nejdříve vyřešíme soustavu lineárních rovnic

$$\mathbf{f}'(\mathbf{x}_k)\,\mathbf{d}_k = -\mathbf{f}(\mathbf{x}_k) \qquad \text{a pak určíme} \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k\,. \tag{5.12}$$

Když je matice $\mathbf{f}'(\mathbf{x}_k)$ regulární, můžeme lineární soustavu rovnic vyřešit metodami popsanými v kapitole 2 (je-li $\mathbf{f}'(\mathbf{x}_k)$ singulární, je třeba metodu vhodně modifikovat, popř. výpočet jako neúspěšný ukončit). Pro velké n a řídkou matici $\mathbf{f}'(\mathbf{x})$ lze efektivně použít iterační metody ($\mathbf{x}_k$ je dobrá počáteční aproximace, navíc $\mathbf{x}_{k+1}$ není třeba počítat příliš přesně, neboť je to jen mezivýsledek na cestě k nalezení $\mathbf{x}^*$).

Newtonova metoda konverguje, pokud je počáteční aproximace $\mathbf{x}_0$ dostatečně blízko kořene $\mathbf{x}^*$. Rychlost konvergence je kvadratická, tj. existuje okolí $O(\mathbf{x}^*)$ bodu $\mathbf{x}^*$ a konstanta C taková, že

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \le C\|\mathbf{x}_k - \mathbf{x}^*\|^2 \qquad \forall\, \mathbf{x}_k \in O(\mathbf{x}^*)\,.$$

Pro ukončení iterací použijeme některé ze stop kriterií

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon\,, \quad \|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon\|\mathbf{x}_k\| \quad \text{nebo} \quad \|\mathbf{f}(\mathbf{x}_{k+1})\| \le \varepsilon\,, \tag{5.13}$$

kde ε je zadaná přesnost. Tato kritéria jsou obecně použitelná také pro další metody, které si v této kapitole uvedeme.

V každém kroku Newtonovy metody je třeba řešit soustavu lineárních rovnic (5.11), což pro velké n představuje značný objem výpočtů. Navíc je třeba v každém kroku vypočítat n^2 složek $\partial f_i(\mathbf{x}_k)/\partial x_j$ matice $\mathbf{f}'(\mathbf{x}_k)$. To může být také velmi obtížné v případě, že parciální derivace nejsou určeny jednoduchými vzorci. Proto se někdy postupuje tak, že se $\mathbf{f}'(\mathbf{x}_k)$ přepočítává jen občas, např. každých m kroků, tj. $\mathbf{x}_{k+1}$ počítáme podle

$$\mathbf{f}'(\mathbf{x}_p)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)\,, \quad k = p, p+1, \ldots, p+m-1\,, \quad p = 0, m, 2m, \ldots.$$

V takovém případě je účelné rozložit matici $\mathbf{f}'(\mathbf{x}_p)$ pomocí LU rozkladu na součin dolní trojúhelníkové matice $\mathbf{L}_p$ a horní trojúhelníkové matice $\mathbf{U}_p$,

$$\mathbf{f}'(\mathbf{x}_p) = \mathbf{L}_p\mathbf{U}_p\,.$$

Tato náročná operace se provede jen pro $k = 0, m, 2m, \ldots$. Aproximaci $\mathbf{x}_{k+1}$ pak dostaneme řešením dvou soustav lineárních rovnic s trojúhelníkovými maticemi,

$$\mathbf{L}_p\mathbf{y} = -\mathbf{f}(\mathbf{x}_k)\,, \qquad \mathbf{U}_p\mathbf{d}_k = \mathbf{y}\,, \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k\,.$$

Pro $k \notin \{0, m, 2m, \ldots\}$ tedy provádíme jen laciné zpětné chody. V propracovaných algoritmech se přepočet Jacobiho matice nevolí staticky, tj. každých m kroků, ale dynamicky, tj. pro $p = 0 < p_1 < p_2 < \ldots$, a to podle rychlosti poklesu $\|\mathbf{f}(\mathbf{x}_k)\|$.

Parciální derivace v Jacobiho matici $\mathbf{f}'(\mathbf{x})$ se často aproximují pomocí diferenčních podílů,

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

kde $h_j \neq 0$ jsou vhodně zvolené parametry, $\mathbf{h} = (h_1, h_2, \ldots, h_n)^T$. 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 $\mathbf{\Delta}(\mathbf{x}, \mathbf{h})$ s prvky $\Delta_{ij}(\mathbf{x}, \mathbf{h})$ je aproximací Jacobiho matice $\mathbf{f}'(\mathbf{x})$. Když tedy v (5.11) nahradíme $\mathbf{f}'(\mathbf{x}_k)$ pomocí $\mathbf{\Delta}(\mathbf{x}_k, \mathbf{h}_k)$, dostaneme *diskretizovanou Newtonovu metodu*

$$\mathbf{\Delta}(\mathbf{x}_k, \mathbf{h}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)\,. \tag{5.14}$$

Zobecněnou metodu sečen dostaneme, když za j-tou složku $h_j^{(k)}$ vektoru $\mathbf{h}_k$ dosadíme $h_j^{(k)} = x_j^{(k-1)} - x_j^{(k)}$ (pro $n = 1$ je pak rovnice (5.14) totožná s předpisem (5.7)) a *zobecněnou Steffensenovu metodu* obdržíme, když položíme $h_j^{(k)} = f_j(\mathbf{x}_k)$ (pro $n = 1$ je pak rovnice (5.14) totožná s předpisem(5.9)). Řád konvergence obou metod je stejný jako v jedné dimenzi, tj. 1,618 pro metodu sečen a 2 pro Steffensenovu metodu. Metoda sečen potřebuje dvě dostatečně dobré počáteční aproximace $\mathbf{x}_0$ a $\mathbf{x}_1$.

Příklad 5.8. Newtonovou metodou určíme kořeny soustavy rovnic

$$f(x, y) = x^3 - xy^2 - 1 = 0\,,$$
$$g(x, y) = y^3 - 2x^2y + 2 = 0\,.$$

Podle (5.12) určíme $\mathbf{x}_{k+1} \equiv (x_{k+1}, y_{k+1})^T$ takto:

$$\begin{pmatrix} f_x(x_k, y_k) & f_y(x_k, y_k) \\ g_x(x_k, y_k) & g_y(x_k, y_k) \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} -f(x_k, y_k) \\ -g(x_k, y_k) \end{pmatrix}, \quad \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k + a_k \\ y_k + b_k \end{pmatrix}.$$

Soustavu rovnic je výhodné vyřešit pomocí Crammerova pravidla, tj.

$$a_k = -\frac{D_x}{D}\,, \qquad b_k = -\frac{D_y}{D}\,,$$

kde

$$D = f_x g_y - f_y g_x\,, \qquad D_x = f g_y - f_y g\,, \qquad D_y = f_x g - f g_x\,,$$

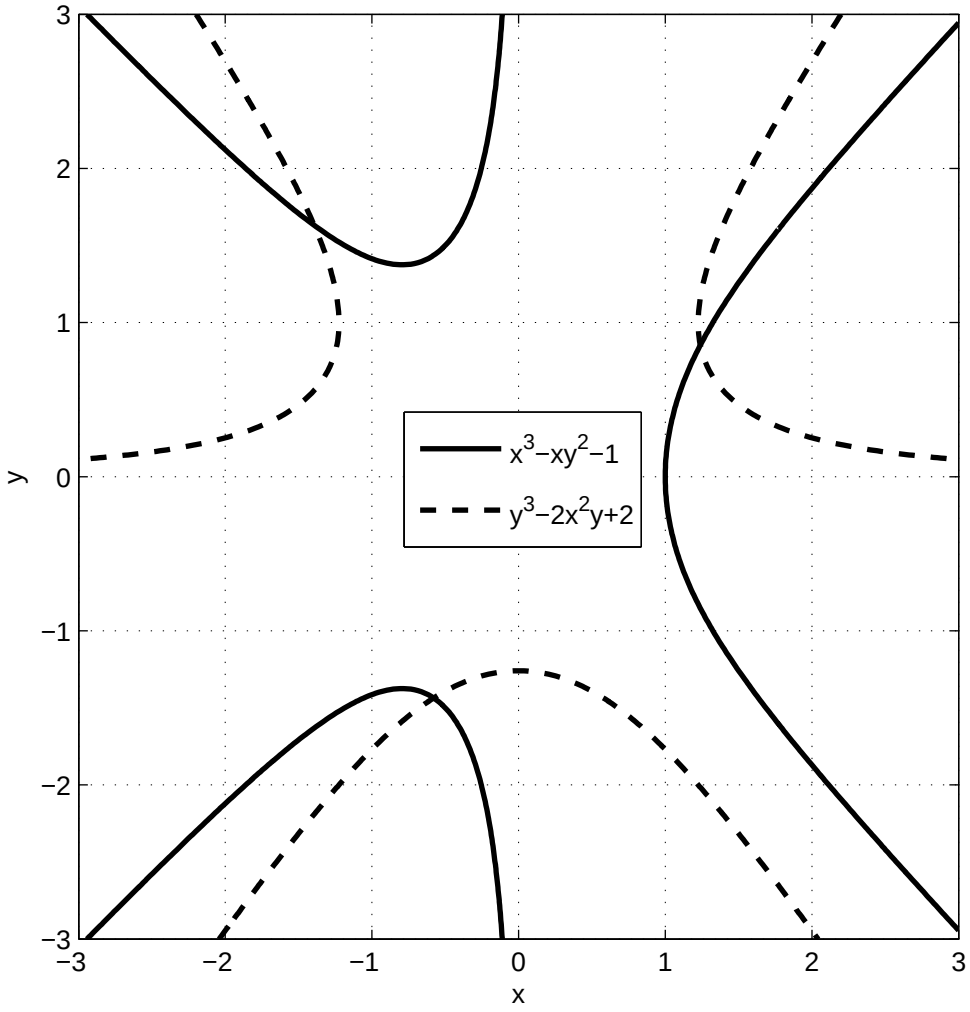
přičemž hodnoty všech funkcí se počítají v bodě $[x_k, y_k]$.

Z obrázku 5.6 zjistíme, že soustava má celkem tři kořeny. Vybereme si třeba ten, který leží ve čtverci $\Omega := \{[x, y] \mid -2 \leq x \leq -1, 1 \leq y \leq 2\}$, a jako počáteční aproximaci zvolíme $x_0 = -1$, $y_0 = 1$. Výpočet ukončíme, když

$$\|\mathbf{f}(\mathbf{x}_{k+1})\|_\infty = \max\{|f(x_{k+1}, y_{k+1})|\,; |g(x_{k+1}, y_{k+1})|\} < 10^{-5}\,.$$

Výpočet je zaznamenán v následující tabulce (f_k a g_k označuje hodnotu v bodě $[x_k, y_k]$).

k	x_k	y_k	f_k	g_k	$x_k - x^*$	$y_k - y^*$
0	−1	1	−1	1	0,394069	−0,631182
1	−1,5	2	1,625	1	−0,105931	0,368818
2	−1,379562	1,673966	0,240186	0,318968	0,014507	0,042784
3	−1,392137	1,629879	0,000193	0,012219	0,001932	−0,001303
4	−1,394072	1,631182	−0,000005	−0,000018	−0,000002	0,000000
5	−1,394069	1,631182	−0,000000	−0,000000	0,000000	0,000000



Obr. 5.6: Soustava dvou nelineárních rovnic

Všimněte si, že v blízkosti kořene je konvergence velmi rychlá. Přesné řešení jsme aproximovali pomocí $\mathbf{x}_5$. $x_5 = -1{,}39407$ a $y_5 = 1{,}63118$ mají všechny cifry platné. □

Zvýšení spolehlivosti metod Newtonova typu. Newtonova metoda a její varianty nemusejí konvergovat, když startujeme daleko od kořene. Je však možné přijmout jistá opatření, která oblast konvergence těchto metod podstatně rozšíří.

Nejjednodušší je použít *tlumenou Newtonovu metodu*, ve které se Newtonův (nebo aproximovaný Newtonův) krok $\mathbf{d}_k$ počítá jako obvykle, ale pak se jako další aproximace bere $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k$, kde λ_k je číselný parametr. Daleko od kořene bývá krok $\mathbf{d}_k$ nespolehlivý, mnohdy příliš velký, a tak se můžeme pokusit vybrat λ_k tak, aby $\mathbf{x}_{k+1}$ byla lepší aproximace $\mathbf{x}^*$ než $\mathbf{x}_k$. Jedním ze způsobů, jak toho dosáhnout, je sledovat $\|\mathbf{f}(\mathbf{x}_k)\|_2$ a zajistit, aby v každé iteraci délka vektoru $\mathbf{f}(\mathbf{x}_k)$ dostatečně poklesla. Parametr λ_k je také možné určit minimalizací funkce $\varphi(\lambda) = \|\mathbf{f}(\mathbf{x}_k + \lambda \mathbf{d}_k)\|_2$ (minimalizaci se věnuje následující kapitola). Ať už parametr λ_k volíme jakkoliv, v blízkosti kořene vždy stačí brát $\lambda_k = 1$ a dosáhnout tak řádu konvergence netlumené metody.

Poněkud komplikovanější, avšak mnohem spolehlivější, je *Newtonova metoda s lokálně omezeným krokem*, ve které $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$ a krok $\mathbf{d}_k$ dostaneme minimalizací funkce $\varphi(\mathbf{d}) = \|\mathbf{f}(\mathbf{x}_k) + \mathbf{f}'(\mathbf{x}_k)\mathbf{d}\|_2$ na oblasti $\|\mathbf{d}\|_2 \leq \Delta_k$, kde Δ_k je vhodně volený parametr. Pro $\varphi(\mathbf{d}_k) = 0$ dostaneme standardní Newtonovu metodu (5.12).

Podrobnější (a mnohé další) informace k tomuto tématu čtenář najde ve specializované literatuře, např. v [16].

Metoda prosté iterace. V mnoha významných aplikacích se řeší nelineární soustava

$$\mathbf{x} = \mathbf{a} + h\boldsymbol{\varphi}(\mathbf{x})\,, \tag{5.15}$$

kde $\mathbf{a}$ je číselný vektor a h je malé kladné číslo. Položíme-li $\mathbf{f}(\mathbf{x}) = \mathbf{x} - \mathbf{a} - h\boldsymbol{\varphi}(\mathbf{x})$, můžeme soustavu $\mathbf{f}(\mathbf{x}) = \mathbf{o}$ vyřešit Newtonovou metodou nebo některou z jejích modifikací. V případě úlohy (5.15) se však přímo nabízí jiný, velmi jednoduchý postup, který si teď popíšeme. Označíme-li $\mathbf{g}(\mathbf{x}) = \mathbf{a} + h\boldsymbol{\varphi}(\mathbf{x})$, dostáváme úlohu

$$\text{určit } \mathbf{x}^* \text{ splňující} \qquad \mathbf{x}^* = \mathbf{g}(\mathbf{x}^*)\,. \tag{5.16}$$

Bod $\mathbf{x}^*$ se nazývá *pevný bod* zobrazení $\mathbf{g}(\mathbf{x})$.

Úlohu (5.16) se pokusíme vyřešit metodou prosté iterace: zvolíme počáteční aproximaci $\mathbf{x}_0$ a počítáme

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k)\,, \qquad k = 0, 1, \ldots \tag{5.17}$$

a doufáme, že takto generovaná *posloupnost postupných aproximací* $\mathbf{x}_k$ konverguje k $\mathbf{x}^*$. Postačující podmínky konvergence udává následující

Věta (*O konvergenci metody prosté iterace*). *Nechť Ω je uzavřená oblast, ve které má funkce $\mathbf{g}$ spojité první parciální derivace a splňuje tyto dvě podmínky:*

$$\begin{aligned} &(\alpha) \qquad \mathbf{g}(\mathbf{x}) \in \Omega \qquad \forall \mathbf{x} \in \Omega\,, \\ &(\beta') \qquad \|\mathbf{g}'(\mathbf{x})\| \leq q < 1 \qquad \forall \mathbf{x} \in \Omega\,. \end{aligned} \tag{5.18}$$

Pak v Ω existuje jediný pevný bod $\mathbf{x}^*$ zobrazení $\mathbf{g}$ a posloupnost postupných aproximací získaná předpisem (5.17) k němu konverguje pro libovolnou počáteční aproximaci $\mathbf{x}_0 \in \Omega$. Přitom

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq q\|\mathbf{x}_k - \mathbf{x}^*\|\,,$$

tj. rychlost obecně jen lineární konvergence závisí na q. □

Vraťme se zpět k úloze (5.15). Když má funkce $\boldsymbol{\varphi}(\mathbf{x})$ v okolí kořene $\mathbf{x}^*$ ohraničené parciální derivace, pak pro dostatečně malé h a pro $\mathbf{x}_0$ dosti blízké k $\mathbf{x}^*$ posloupnost postupných aproximací

$$\mathbf{x}_{k+1} = \mathbf{a} + h\boldsymbol{\varphi}(\mathbf{x}_k)\,, \qquad k = 0, 1, \ldots$$

konverguje k $\mathbf{x}^*$. (Snadno se ověří, že v tom případě lze aplikovat předchozí větu, když zvolíme $O(\mathbf{x}^*) = \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}^*\| < \delta\}$, kde $\delta > 0$ je dostatečně malé číslo.)

Příklad 5.9. Metodou prosté iterace určíme řešení soustavy rovnic

$$x = 0{,}2 + 0{,}1(-xy^2 + 3x)\,,$$
$$y = 0{,}6 + 0{,}1(-x^2y^3 - 2y)$$

v oblasti $\Omega = \{[x,y]\,|\,0 \le x \le 1, 0 \le y \le 1]\}$. Nejdříve ověříme, že funkce

$$g_1(x,y) = 0{,}2 + 0{,}1(-xy^2 + 3x)\,, \qquad g_2(x,y) = 0{,}6 + 0{,}1(-x^2y^3 - 2y)$$

splňují podmínky (5.18).

Protože pro $[x,y] \in \Omega$ platí

$$0 \le 0{,}2 + 0{,}1(-xy^2 + 3x) \le 1\,, \qquad 0 \le 0{,}6 + 0{,}1(-x^2y^3 - 2y) \le 1\,,$$

tj. $[g_1(x,y), g_2(x,y)] \in \Omega$, podmínka a) je splněna.

Abychom ověřili podmínku b), vyjádříme si Jacobiho matici

$$\mathbf{g}'(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial g_1(x,y)}{\partial x} & \dfrac{\partial g_1(x,y)}{\partial y} \\ \dfrac{\partial g_2(x,y)}{\partial x} & \dfrac{\partial g_2(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} -0{,}1y^2 + 0{,}3 & -0{,}2xy \\ -0{,}2xy^3 & -0{,}3x^2y^2 - 0{,}2 \end{pmatrix}$$

a odhadneme např. v $\|\cdot\|_\infty$ normě

$$\|\mathbf{g}'(\mathbf{x})\|_\infty = \max\{|-0{,}1y^2 + 0{,}3| + |-0{,}2xy|\,;|-0{,}2xy^3| + |-0{,}3x^2y^2 - 0{,}2|\}\,.$$

Zřejmě

$$\|\mathbf{g}'(\mathbf{x})\|_\infty \le \max\{0{,}3 + 0{,}2\,;\,0{,}2 + 0{,}5\} = 0{,}7 \qquad \text{pro } \mathbf{x} = [x,y] \in \Omega\,,$$

takže podmínka b) je splněna pro $q = 0{,}7$.

Výpočet zahájíme třeba z počáteční aproximace $x_0 = y_0 = 0$ a pro ukončení iterací použijeme stop kritérium

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_\infty = \max\{|x_{k+1} - x_k|\,;|y_{k+1} - y_k|\} < 10^{-5}\,.$$

Výpočet je zaznamenán v následující tabulce.

k	x_k	y_k	$x_k - x_{k-1}$	$y_k - y_{k-1}$	$x_k - x^*$	$y_k - y^*$
0	0	0			−0,275892	−0,499211
1	0,2	0,6	0,2	0,6	−0,075892	0,100789
2	0,252800	0,479136	0,052800	−0,120864	−0,023092	−0,020075
3	0,270036	0,503470	0,017236	0,024334	−0,005856	0,004259
⋮						
8	0,275882	0,499209	0,000025	−0,000009	−0,000010	−0,000001
9	0,275889	0,499211	0,000007	0,000002	−0,000003	0,000000

Přesné řešení jsme aproximovali pomocí $\mathbf{x}_{15}$. $x_9 = 0{,}27589$ a $y_9 = 0{,}49921$ mají všechny cifry platné. □

Poznámka. Když $\mathbf{g}(\mathbf{x}) = \mathbf{T}\mathbf{x} + \mathbf{c}$ je lineární funkce, pak $\mathbf{g}'(\mathbf{x}) = \mathbf{T}$. Pokud $\|\mathbf{T}\| < 1$, pak jsou postačující podmínky (5.18) splněny pro každé $\mathbf{x}$, takže $\mathbf{x}_k \to \mathbf{x}^*$ pro libovolnou počáteční aproximaci $\mathbf{x}_0$. Tento výsledek jsme dokázali v kapitole 2, viz (2.27).

5.4. Cvičení

5.1. K jakému číslu konverguje posloupnost iterací, definovaná takto

$$x_0 = 1, \qquad x_{k+1} = \frac{1}{2}x_k + \frac{1}{x_k}.$$

Zdůvodněte konvergenci!

[$\lim_{k\to\infty} x_k = \sqrt{2}$, iterační předpis je Newtonova metoda pro rovnici $x^2 - 2 = 0$.]

5.2. Metodou sečen a tečen spočtěte řešení dále uvedených rovnic na 6 platných desetinných cifer přesně. Porovnejte počet iterací nutných k dosažení požadované přesnosti. (a) $x - e^{-x} = 0$, $x_0 = 0{,}5$, $x_1 = 0{,}6$; (b) $x - \cos x = 0$, $x_0 = 0{,}5$, $x_1 = 0{,}6$; (c) $x^3 + 4x^2 - 10 = 0$, $x_0 = 1{,}5$, $x_1 = 1{,}6$. (x_1 použijte jen v metodě sečen.)

[(a) 0,567143 (b) 0,739085 (c) 1,365230.]

5.3. (a) Naprogramujte hledání znaménkové změny. Vstupem je funkce $f(x)$, interval $\langle a, b\rangle$ a přirozené číslo n. Program rozdělí interval $\langle a, b\rangle$ na n stejných dílků. V dělicích bodech x_i, $i = 0, 1, \ldots, n$, spočte $f(x_i)$ a vrátí seznam subintervalů (x_k, x_{k+1}), v nichž platí $f(x_k)f(x_{k+1}) < 0$.
(b) Naprogramujte metodu půlení intervalu. Vstupem bude funkce $f(x)$, interval $\langle a, b\rangle$ a požadovaná délka $d = |b_k - a_k|$ výsledného intervalu $\langle a_k, b_k\rangle$.
(c) Propojte (a) s (b); dostanete nástroj pro hledání kvalitních počátečních aproximací.
(d) Naprogramujte metodu sečen a tečen včetně testování požadované přesnosti ε. Počáteční aproximaci určete pomocí (c) jako $x_0 = (a_k + b_k)/2$.

5.4. Jaké výsledky přinese program ze cvičení 5.3 pro vstupní data:
(a) $f(x) = x^3 + 5x^2 - 10$, $\langle a, b\rangle = \langle -5, 3\rangle$, $n = 5$, $d = 0{,}1$, $\varepsilon = 10^{-6}$?
(b) $f(x) = x^5 + 2x^4 - x^3 - 2x^2 + 0{,}1$, $\langle a, b\rangle = \langle -3, 3\rangle$, $n = 5$, $d = 0{,}1$, $\varepsilon = 10^{-6}$?
(c) $f(x) = x^5 + 2x^4 - x^3 - 2x^2 + 0{,}1$, $\langle a, b\rangle = \langle -3, 3\rangle$, $n = 10$, $d = 0{,}1$, $\varepsilon = 10^{-6}$?

[(a) Najde všechny kořeny $x_1^* \doteq -4{,}507903$, $x_2^* \doteq -1{,}755640$, $x_3^* \doteq 1{,}263543$;
(b) najde jen kořeny $x_1^* \doteq -2{,}008176$, $x_2^* \doteq -0{,}945472$, $x_3^* \doteq 0{,}982479$;
(c) najde všechny kořeny $x_1^* \doteq -2{,}008176$, $x_2^* \doteq -0{,}945472$, $x_3^* \doteq 0{,}982479$, $x_4^* \doteq -0{,}246397$, $x_5^* \doteq 0{,}217566$.]

5.5. Ověřte, že Newtonova metoda pro $f(x) = (x-1)^2$ konverguje k $x^* = 1$ jen lineárně.

[$x_{k+1} - 1 = \frac{1}{2}(x_k - 1)$.]

5.6. Hledejte kořen rovnice $f(x) := x^2 + \ln x - 10/x = 0$ na intervalu $I = \langle 1, 4\rangle$ metodou prosté iterace. Zkoumejte na počítači konvergenci pro iterační funkce (a) $g(x) = e^{10/x - x^2}$, (b) $g(x) = 10/(x^2 + \ln x)$, (c) $g(x) = \sqrt{10/x - \ln x}$, (d) $g(x) = \sqrt[3]{10 - x\ln x}$. Počáteční iteraci x_0 zvolte vždy ve středu intervalu, tj. $x_0 = 2{,}5$. Ověřte, že $f(x) = 0 \iff x = g(x)$ a odhadněte $q = \max_{x\in I} |g'(x)|$. Jsou výsledky v souladu s větou o konvergenci?

[(a) nekonverguje, $q \doteq 9{,}72 \cdot 10^4$; (b) nekonverguje, $q \doteq 30$; (c) konverguje, $q \doteq 1{,}74$; (d) konverguje rychleji, $q \doteq 0{,}57$.]

5.7. Newtonovou metodou řešte soustavy rovnic na 6 desetinných cifer přesně.

$$\text{(a)}\quad \begin{aligned} 2\cos(xy) &= 1 \\ 2\sin(x+y) &= 1 \end{aligned} \qquad \text{(b)}\quad \begin{aligned} 2\cos(xy) - \sin x &= 0 \\ 2x\sin y - 3y\sin x &= -1 \end{aligned} \qquad \text{(c)}\quad \begin{aligned} x^2y - e^{xy} &= -2 \\ \ln(x+1)y - y^2/x &= 0 \end{aligned}$$

Jako starovací hodnoty použijte (a) $x_0 = 2{,}5$, $y_0 = 0{,}25$; (b) $x_0 = 1$, $y_0 = 1$; (c) $x_0 = 1$, $y_0 = 1$.
Blízkost kořene ověřte graficky.

[(a) $x^* \doteq 2{,}125254$, $y^* \doteq 0{,}492740$; (b) $x^* \doteq 1{,}023402$, $y^* \doteq 1{,}103856$; (c) $x^* \doteq 1{,}256558$, $y^* \doteq 1{,}022638$.]

6. Optimalizace

Optimalizační úlohy se zabývají výběrem nejlepších řešení z dané množiny možných řešení. Matematicky můžeme optimalizační úlohu formulovat jako nalezení prvku $\mathbf{x}^* \in M$ takového, že pro libovolný prvek $\mathbf{x} \in M$ platí

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \qquad \forall \mathbf{x} \in M\,, \tag{6.1}$$

kde $f : M \mapsto \mathbb{R}$ je minimalizovaná (někdy se také říká *účelová* nebo *cílová* nebo *kriteriální*) *funkce* a M je *množina přípustných řešení*. Jestliže přípustným řešením může být každý bod $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ n-rozměrného Euklidova prostoru $\mathbb{R}^n$, tj. $M = \mathbb{R}^n$, hovoříme o *nepodmíněné optimalizaci*. O funkci f budeme předpokládat, že je spojitá (případně i se svými prvními a dalšími derivacemi).

Optimalizační úloha (6.1) se nazývá *úlohou globální optimalizace*. My se v této kapitole omezíme na jednodušší *úlohu lokální optimalizace* spočívající v nalezení *lokálního minima*, tj. prvku $\mathbf{x}^* \in M$ takového, že platí

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \qquad \forall \mathbf{x} \in M \cap O(\mathbf{x}^*)\,, \tag{6.2}$$

kde $O(\mathbf{x}^*)$ je nějaké okolí bodu $\mathbf{x}^*$.

Podrobněji si všimneme dvou speciálních úloh: v odstavci 6.1 se seznámíme s metodami pro minimalizaci funkce jedné proměnné na intervalu $\langle a, b\rangle$ a v odstavci 6.2 se budeme věnovat metodám nepodmíněné minimalizace funkce více proměnných.

Poznámka. Určení maxima funkce $g(\mathbf{x})$ můžeme převést na úlohu určení minima funkce $f(\mathbf{x}) = -g(\mathbf{x})$. □

6.1. Jednorozměrná minimalizace

V tomto odstavci uvedeme metody pro přibližné určení bodu x^* lokálního minima funkce $f(x)$ na intervalu $\langle a, b\rangle$. Jestliže je funkce f na intervalu $\langle a, b\rangle$ *unimodální*, tj. když má v intervalu $\langle a, b\rangle$ jediné minimum, pak ho (dále uvedenými metodami přibližně) najdeme. Pokud však má funkce f na intervalu $\langle a, b\rangle$ více lokálních minim, najdeme jedno z nich.

Intervalovou bisekci použít nemůžeme: i když známe $f(a)$, $f(b)$ a $f((a + b)/2)$, nedokážeme rozhodnout, ve které polovině intervalu $\langle a, b\rangle$ minimum leží.

Použít lze intervalovou trisekci. Nechť $h = (b - a)/3$, takže $u = a + h$ a $v = b - h$ dělí interval na tři stejné části. Předpokládejme, že $f(u) < f(v)$. Pak minimum jistě leží vlevo od v, takže b nahradíme pomocí v. Tím se délka intervalu (obsahujícího minimum) zkrátí na dvě třetiny své původní délky. Bod u se však stane středem nového intervalu a nebude proto v dalším kroku využitelný. Funkci f tedy musíme vyhodnocovat v každém kroku dvakrát. To je neefektivní.

Metoda zlatého řezu je založena na šikovnějším výběru dělicích bodů u a v. Nechť $h = \varrho(b - a)$, kde ϱ je číslo o něco větší než $1/3$, jehož přesnou hodnotu teprve určíme. Pak body $u = a + h$ a $v = b - h$ dělí interval $\langle a, b\rangle$ na tři nestejné části. V prvním kroku vyhodnotíme $f(u)$ a $f(v)$. Předpokládejme, že $f(u) < f(v)$. Pak víme, že minimum je

mezi a a v. Nahradíme b pomocí v a proces opakujeme. Když zvolíme správnou hodnotu ϱ, bod u bude ve správné pozici použitelné v příštím kroku. Po prvním kroku se tak funkce f bude vyhodnocovat v každém kroku už jen jednou.

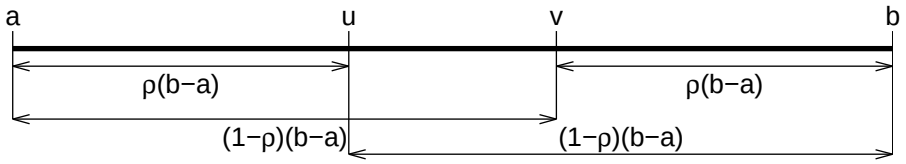
Jak tedy zvolit ϱ? Tak, aby bod u hrál v redukovaném intervalu $\langle a, v\rangle$ stejnou roli jako bod v v původním intervalu $\langle a, b\rangle$, tj. aby poměr délky intervalu $\langle a, u\rangle$ k délce intervalu $\langle a, v\rangle$ byl stejný jako poměr délky intervalu $\langle a, v\rangle$ k délce intervalu $\langle a, b\rangle$,

$$\frac{u-a}{v-a} = \frac{v-a}{b-a} \qquad \Longleftrightarrow \qquad \frac{\varrho}{1-\varrho} = \frac{1-\varrho}{1} \qquad \Longleftrightarrow \qquad \varrho^2 - 3\varrho + 1 = 0\,.$$

Vyhovující řešení je

$$\varrho = (3-\sqrt{5})/2 \approx 0{,}382, \qquad \text{kde} \quad \tau = 1-\varrho = (\sqrt{5}-1)/2 \approx 0{,}618$$

je číslo známé jako *poměr zlatého řezu*.



Obr. 6.1: Zlatý řez

Poznámka (*O zlatém řezu*). Říkáme, že bod dělí interval v poměru zlatého řezu, když dva nově vzniklé subintervaly mají tuto vlastnost: poměr délky kratšího subintervalu k délce delšího subintervalu je stejný jako poměr délky delšího subintervalu k délce celého intervalu. Z výše uvedené konstrukce je zřejmé, že bod u (ale také bod v) dělí interval $\langle a, b\rangle$ v poměru zlatého řezu.

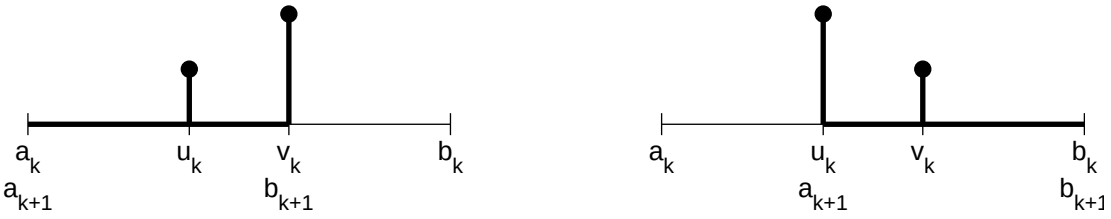
Připomeňme si, že s číslem τ jsme se setkali již v kapitole 5.2, kde jsme uvedli, že rychlost konvergence metody sečen $p = 1 + \tau \doteq 1{,}618$. □

Zatím jsme předpokládali, že $f(u) < f(v)$. V opačném případě, tj. když $f(u) \geq f(v)$, leží minimum v intervalu $\langle u, b\rangle$, takže a nahradíme pomocí u. Snadno ověříme, že v redukovaném intervalu $\langle u, b\rangle$ bude mít v stejnou roli jako mělo u v původním intervalu $\langle a, b\rangle$, takže hodnotu funkce f na redukovaném intervalu opět stačí počítat jen jednou.

Délka redukovaného intervalu je τ-krát menší než délka původního intervalu. Z výchozího intervalu$\langle a_0, b_0\rangle = \langle a, b\rangle$ tak postupně sestrojíme intervaly $\langle a_1, b_1\rangle \supset \langle a_2, b_2\rangle \supset \ldots$, které obsahují minimum a jejichž délka je v každém kroku redukována faktorem τ. Na výchozím intervalu $\langle a_0, b_0\rangle$ určíme $u_0 = a + \varrho(b-a)$, $v_0 = b - \varrho(b-a)$ a vypočteme $f(u_0)$, $f(v_0)$. Interval $\langle a_{k+1}, b_{k+1}\rangle$, $k = 0, 1, \ldots$, dostaneme pomocí a_k, b_k, u_k, v_k a již dříve vypočtených hodnot $f(u_k)$, $f(v_k)$ takto:

1) když $f(u_k) < f(v_k)$, pak
$$a_{k+1} := a_k, \quad b_{k+1} := v_k, \quad v_{k+1} := u_k, \quad u_{k+1} := a_{k+1} + b_{k+1} - v_{k+1}$$
a vypočteme $f(u_{k+1})$;
2) v opačném případě, tj. když $f(u_k) \geq f(v_k)$, provedeme
$$a_{k+1} := u_k, \quad b_{k+1} := b_k, \quad u_{k+1} := v_k, \quad v_{k+1} := a_{k+1} + b_{k+1} - u_{k+1}$$

a vypočteme $f(v_{k+1})$.



Obr. 6.2: Metoda zlatého řezu

Po k krocích leží minimum v intervalu $I_k := \langle a_k, b_k \rangle$ délky

$$|I_k| = b_k - a_k = \tau(b_{k-1} - a_{k-1}) = \cdots = \tau^k (b_0 - a_0)\,.$$

Střed x_{k+1} intervalu $\langle a_k, b_k \rangle$ aproximuje minimum x^* s chybou

$$|x_{k+1} - x^*| \leq \tfrac{1}{2}(b_k - a_k) = \tfrac{1}{2}\tau^k (b_0 - a_0)\,. \tag{6.3}$$

Pro $k \to \infty$ zřejmě $|I_k| \to 0$ a $x_k \to x^*$.

Konvergence metody zlatého řezu je poměrně pomalá. Proto je v blízkosti minima účelné přejít na rychleji konvergentní metodu. Jednou z možností je metoda kvadratické interpolace, s níž se také seznámíme. Předtím ale

Příklad 6.1. Metodou zlatého řezu určíme minimum funkce $f(x) = x^4 - 3x^3 + x + 7$. Jako počáteční zvolíme interval $\langle a_0, b_0 \rangle = \langle 1, 3 \rangle$. Výpočet provedeme s přesností $\varepsilon = 10^{-3}$, tj. když $b_k - a_k < 2\varepsilon$, položíme $x_{k+1} = (a_k + b_k)/2$. Výpočet je zaznamenán v následující tabulce. Podtržením jsou vyznačeny ty vnitřní body, v nichž se počítá hodnota účelové funkce.

k	a_k	u_k	v_k	b_k	$f(u_k)$		$f(v_k)$
0	1,0000	<u>1,7639</u>	<u>2,2361</u>	3,0000	1,979900	$\geq$	0,695048
1	1,7639	2,2361	<u>2,5279</u>	3,0000	0,695048	$<$	1,901312
2	1,7639	<u>2,0557</u>	2,2361	2,5279	0,852324	$\geq$	0,695048
3	2,0557	2,2361	<u>2,3475</u>	2,5279	0,695048	$<$	0,906510
4	2,0557	<u>2,1672</u>	2,2361	2,3475	0,690296	$<$	0,695048
5	2,0557	<u>2,1246</u>	2,1672	2,2361	0,729249	$\geq$	0,690296
$\vdots$							
14	2,1973	2,1982	<u>2,1988</u>	2,1997	0,681572	$<$	0,681575
15	2,1973			2,1988			

Požadovaná přesnost je dosažena pro $k = 15$, takže (po zaokrouhlení na 3 desetinné cifry) $x_{16} \doteq 2{,}198$. Hodnota účelové funkce se počítá celkem 16-krát. Protože přesná hodnota $x^* \doteq 2{,}198266$, má $x_{16} := 2{,}198$ všechny cifry platné. □

Metoda kvadratické interpolace. Předpokládejme, že minimum leží v intervalu $\langle a_k, b_k\rangle$, a že v nějakém jeho vnitřním bodě c_k hodnota funkce f nepřesáhne hodnoty $f(a_k)$, $f(b_k)$ v krajních bodech a_k, b_k, tj. že

$$\text{pro} \quad a_k < c_k < b_k \quad \text{platí} \quad f(c_k) \le \min\{f(a_k); f(b_k)\}\,. \tag{6.4}$$

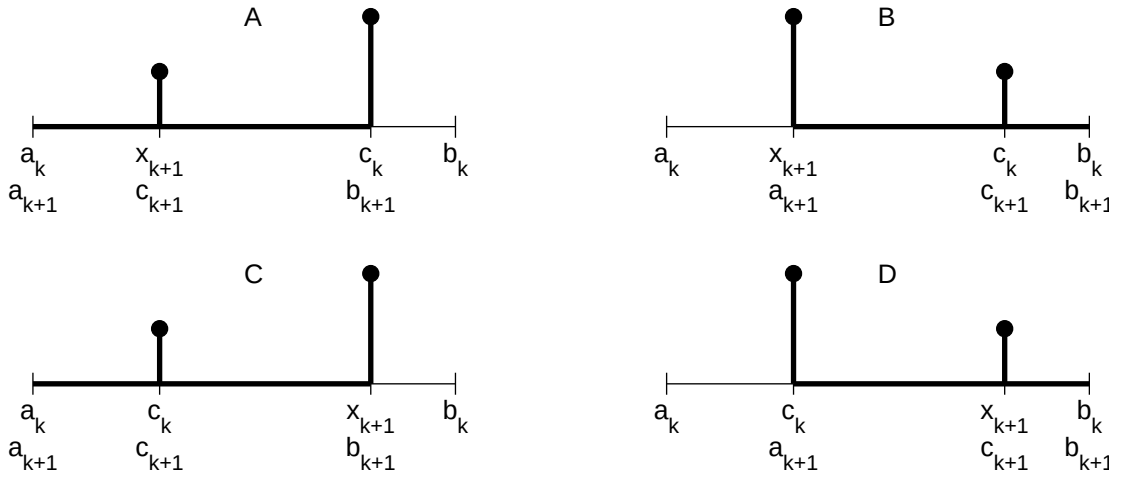
Body $[a_k, f(a_k)]$, $[c_k, f(c_k)]$ a $[b_k, f(b_k)]$ proložíme parabolu $P_2(x)$ (kvadratický interpolační polynom) a bod x_{k+1} jejího minima považujeme za další aproximaci x^*. Vzorec pro výpočet x_{k+1} dostaneme řešením lineární rovnice $P_2'(x_{k+1}) = 0$. Dá se ukázat, že

$$x_{k+1} = c_k - \frac{1}{2}\frac{(c_k - a_k)^2[f(c_k) - f(b_k)] - (c_k - b_k)^2[f(c_k) - f(a_k)]}{(c_k - a_k)[f(c_k) - f(b_k)] - (c_k - b_k)[f(c_k) - f(a_k)]}\,. \tag{6.5}$$

Z (6.4) plyne, že $x_{k+1} \in (a_k, b_k)$. Když náhodou $x_{k+1} = c_k$, vložíme do x_{k+1} jiný vnitřní bod intervalu $\langle a_k, b_k\rangle$. S tímto slabým místem metody kvadratické interpolace (a s dalšími, zde nezmíněnými nedostatky) se úspěšně vyrovnává Brentova metoda. Stručná zmínka o ní je uvedena v následujícím textu.

Z bodů a_k, b_k, c_k a x_{k+1} pak vybereme nový interval (a_{k+1}, b_{k+1}) obsahující minimum a bod c_{k+1} splňující podmínku (6.4), tentokrát pro index $k+1$. Postupujeme podle následujících pravidel:

$$\begin{aligned}
&\text{A}: \quad x_{k+1} < c_k \quad \text{a} \quad f(x_{k+1}) < f(c_k) \quad \Longrightarrow \quad a_{k+1} = a_k, \quad c_{k+1} = x_{k+1}, \quad b_{k+1} = c_k,\\
&\text{B}: \quad x_{k+1} < c_k \quad \text{a} \quad f(x_{k+1}) \ge f(c_k) \quad \Longrightarrow \quad a_{k+1} = x_{k+1}, \quad c_{k+1} = c_k, \quad b_{k+1} = b_k,\\
&\text{C}: \quad c_k < x_{k+1} \quad \text{a} \quad f(c_k) < f(x_{k+1}) \quad \Longrightarrow \quad a_{k+1} = a_k, \quad c_{k+1} = c_k, \quad b_{k+1} = x_{k+1},\\
&\text{D}: \quad c_k < x_{k+1} \quad \text{a} \quad f(c_k) \ge f(x_{k+1}) \quad \Longrightarrow \quad a_{k+1} = c_k, \quad c_{k+1} = x_{k+1}, \quad b_{k+1} = b_k.
\end{aligned}$$



Obr. 6.3: Metoda kvadratické interpolace

Výpočet ukončíme a x_{k+1} považujeme za dostatečně dobrou aproximaci minima x^*, když je splněno některé ze stop kriterií

$$|x_{k+1} - x_k| < \varepsilon\,,\ |x_{k+1} - x_k| < \varepsilon|x_k|\,,\ |f(x_k) - f(x_{k+1})| < \varepsilon\,,\ |f(x_k) - f(x_{k+1})| < \varepsilon|f(x_k)|\,,$$

kde ε je předepsaná tolerance.

Výpočet x_{k+1}, $k = 0, 1, \ldots$, vyžaduje $k+3$ vyhodnocení účelové funkce: $f(a_0)$, $f(c_0)$, $f(b_0)$ a dále $f(x_i)$, $i = 1, 2, \ldots, k$.

Pokud metoda kvadratické interpolace konverguje, pak je rychlost její konvergence superlineární řádu $p \approx 1{,}324$, viz [7].

Příklad 6.2. Minimum funkce $f(x) = x^4 - 3x^3 + x + 7$ určíme metodou kvadratické interpolace. Výpočet ukončíme, když $|x_{k+1} - x_k| < 10^{-5}$. Výpočet je zaznamenán v následující tabulce. V posledním sloupci je uvedeno, který z případů A,B,C,D nastává, a podtržením

$a_0 = \underline{2{,}000000}$	$x_1 = \underline{2{,}204918}$	$<$	$c_0 = \underline{2{,}500000}$	$b_0 = 3{,}000000$	$f(x_1) < f(c_0)$	$\Longrightarrow$ A
$a_1 = 2{,}000000$	$x_2 = \underline{2{,}180689}$	$<$	$c_1 = \underline{2{,}204918}$	$b_1 = \underline{2{,}500000}$	$f(x_2) \geq f(c_1)$	$\Longrightarrow$ B
$a_2 = \underline{2{,}180689}$	$x_3 = \underline{2{,}197322}$	$<$	$c_2 = \underline{2{,}204918}$	$b_2 = 2{,}500000$	$f(x_3) < f(c_2)$	$\Longrightarrow$ A
$a_3 = 2{,}180689$	$c_3 = \underline{2{,}197322}$	$<$	$x_4 = \underline{2{,}198232}$	$b_3 = \underline{2{,}204918}$	$f(c_3) \geq f(x_4)$	$\Longrightarrow$ D
$a_4 = 2{,}197322$	$c_4 = \underline{2{,}198232}$	$<$	$x_5 = \underline{2{,}198264}$	$b_4 = \underline{2{,}204918}$	$f(c_4) \geq f(x_5)$	$\Longrightarrow$ D
$a_5 = 2{,}198232$	$c_5 = 2{,}198264$	$<$	$x_6 = 2{,}198265$	$b_5 = 2{,}204918$		

jsou (postupně zleva doprava) vyznačeny body $a_{k+1} < c_{k+1} < b_{k+1}$. Z tabulky je zřejmé, že požadovaná přesnost byla dosažena pro $x_6 \doteq 2{,}19827$ (všechny cifry jsou platné). Hodnota účelové funkce se počítala celkem 8-krát. □

Brentova metoda je kombinovaná metoda, která v sobě spojuje spolehlivost metody zlatého řezu a rychlou konvergenci metody kvadratické interpolace. Popis Brentovy metody lze najít např. v [18], viz funkce `brent`, nebo v [15], viz funkce `fmintx`. Brentův algoritmus je také základem funkce `fminbnd` pro jednorozměrnou minimalizaci v MATLABu.

6.2. Minimalizace funkce více proměnných

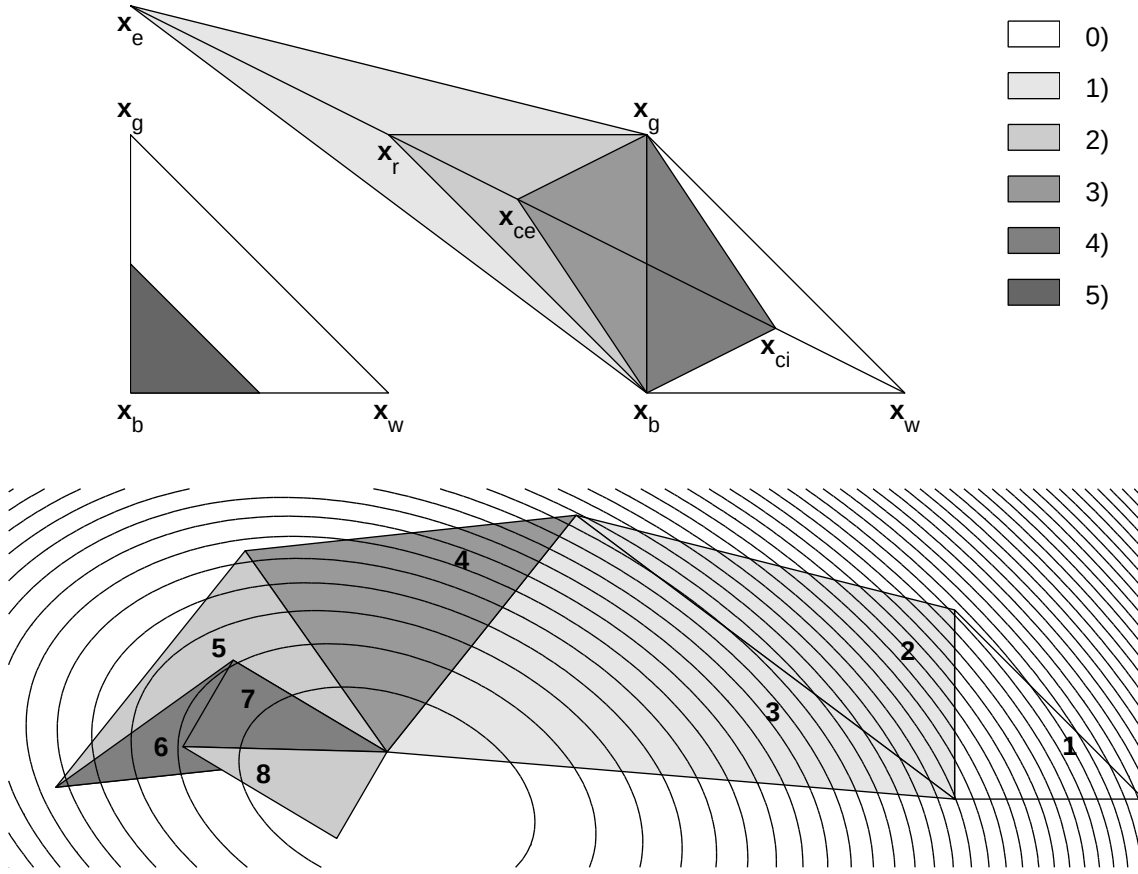
Nelderova-Meadova metoda známá také jako *metoda simplexů* je populární metoda nepoužívající derivace účelové funkce. Patří mezi tzv. *komparativní metody*, což jsou metody, které hledají minimum účelové funkce f porovnáváním jejích hodnot v určitých vybraných bodech prostoru $\mathbb{R}^n$. V případě metody simplexů jsou vybranými body vrcholy simplexu (pro $n = 2$ trojúhelníka, pro $n = 3$ čtyřstěnu).

Hlavní myšlenka jednoho kroku metody je jednoduchá: mezi vrcholy $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$ simplexu vybereme nejhorší vrchol $\mathbf{x}_w$ (v angličtině worst), v němž účelová funkce nabývá největší hodnotu, a nahradíme ho lepším vrcholem $\hat{\mathbf{x}}$, v němž je hodnota účelové funkce menší.

Vrchol $\hat{\mathbf{x}}$ hledáme na polopřímce, která vychází z nejhoršího vrcholu $\mathbf{x}_w$ a prochází těžištěm $\bar{\mathbf{x}}$ zbývajících vrcholů. Nejlepší z nich označíme $\mathbf{x}_b$ (v angličtině best), tj. $\mathbf{x}_b$ je ten z vrcholů $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$, v němž účelová funkce nabývá nejmenší hodnotu.

První pokus, jak vybrat $\hat{\mathbf{x}}$, označujeme jako *reflexi*: bod $\mathbf{x}_r = \bar{\mathbf{x}} + (\bar{\mathbf{x}} - \mathbf{x}_w)$, je obraz bodu $\mathbf{x}_w$ ve středové souměrnosti se středem $\bar{\mathbf{x}}$. Když je $f(\mathbf{x}_r) < f(\mathbf{x}_b)$, pak to znamená, že pokles hodnot na polopřímce $\mathbf{x}_w\bar{\mathbf{x}}$ je značný, a proto zkusíme postoupit po této polopřímce ještě dál, do bodu $\mathbf{x}_e = \bar{\mathbf{x}} + 2(\bar{\mathbf{x}} - \mathbf{x}_w)$. Výběr bodu $\mathbf{x}_e$ bývá označován jako *expanze*. Když $f(\mathbf{x}_e) < f(\mathbf{x}_b)$, pak $\hat{\mathbf{x}} = \mathbf{x}_e$, tj. nejhorší vrchol $\mathbf{x}_w$ nahradíme bodem $\mathbf{x}_e$.

Jestliže $f(\mathbf{x}_e) \geq f(\mathbf{x}_b)$, zkusíme použít alespoň bod $\mathbf{x}_r$. Podmínkou pro jeho zařazení do simplexu je splnění podmínky $f(\mathbf{x}_r) < f(\mathbf{x}_g)$ pro některý vrchol $\mathbf{x}_g$ jiný než nejhorší, tj. pro $\mathbf{x}_g \neq \mathbf{x}_w$ (index g připomíná anglické slůvko good). Pokud taková podmínka platí, bereme $\hat{\mathbf{x}} = \mathbf{x}_r$ místo původního $\mathbf{x}_w$.



Obr. 6.4: Nelderova-Meadova metoda: 0) originální trojúhelník, 1) expanze, 2) reflexe, 3) vnější kontrakce, 4) vnitřní kontrakce, 5) redukce

Když nevyhovuje $\mathbf{x}_e$ ani $\mathbf{x}_r$, zkusíme najít bod $\hat{\mathbf{x}}$ na úsečce s koncovými body $\mathbf{x}_w$, $\mathbf{x}_r$ tak, aby $f(\hat{\mathbf{x}}) < \min\{f(\mathbf{x}_w); f(\mathbf{x}_r)\}$. Konkrétně postupujeme takto:

a) pokud $f(\mathbf{x}_r) < f(\mathbf{x}_w)$, zkusíme bod $\mathbf{x}_{ce} = \frac{1}{2}(\bar{\mathbf{x}} + \mathbf{x}_r)$ (leží blíže k bodu $\mathbf{x}_r$), a když $f(\mathbf{x}_{ce}) < f(\mathbf{x}_r)$, pak $\hat{\mathbf{x}} = \mathbf{x}_{ce}$, takže provedeme $\mathbf{x}_w := \mathbf{x}_{ce}$.

b) jestliže $f(\mathbf{x}_r) \geq f(\mathbf{x}_w)$, zkusíme bod $\mathbf{x}_{ci} = \frac{1}{2}(\bar{\mathbf{x}} + \mathbf{x}_w)$ (leží blíže k bodu $\mathbf{x}_w$), a pokud $f(\mathbf{x}_{ci}) < f(\mathbf{x}_w)$, pak $\hat{\mathbf{x}} = \mathbf{x}_{ci}$, tj. provedeme $\mathbf{x}_w := \mathbf{x}_{ci}$.

Výběr bodu $\mathbf{x}_{ce}$ resp. $\mathbf{x}_{ci}$ označujeme jako *kontrakci* (v dolním indexu: písmeno c připomíná anglické slovo contraction, písmeno e anglické slovo external ($\mathbf{x}_{ce}$ leží vně původního simplexu) a písmeno i připomíná anglické slovo internal ($\mathbf{x}_{ci}$ leží uvnitř původního simplexu)).

Když nevyhovuje $\mathbf{x}_e$, $\mathbf{x}_r$, $\mathbf{x}_{ce}$ ani $\mathbf{x}_{ci}$, usoudíme, že vrchol $\mathbf{x}_b$, v němž účelová funkce nabývá své nejmenší hodnoty, je blízko minima. Proto provedeme *redukci simplexu*: vrchol

$\mathbf{x}_b$ v simplexu zůstane a zbývající vrcholy $\mathbf{x}_i$ se posunou do středů úseček $\mathbf{x}_b\mathbf{x}_i$. Simplex se tedy stáhne k nejlepšímu vrcholu $\mathbf{x}_b$.

Transformaci simplexu, představující jeden krok Nelderovy-Meadovy metody, popíšeme v pěti bodech takto:

1) expanze :	$f(\mathbf{x}_r) < f(\mathbf{x}_b)$ a navíc $f(\mathbf{x}_e) < f(\mathbf{x}_b)$	$\Longrightarrow$	$\mathbf{x}_w := \mathbf{x}_e$
2) reflexe :	$f(\mathbf{x}_r) < f(\mathbf{x}_g)$ pro nějaký vrchol $\mathbf{x}_g \neq \mathbf{x}_w$	$\Longrightarrow$	$\mathbf{x}_w := \mathbf{x}_r$
3) vnější kontrakce :	$f(\mathbf{x}_r) < f(\mathbf{x}_w)$ a navíc $f(\mathbf{x}_{ce}) < f(\mathbf{x}_r)$	$\Longrightarrow$	$\mathbf{x}_w := \mathbf{x}_{ce}$
4) vnitřní kontrakce :	$f(\mathbf{x}_r) \geq f(\mathbf{x}_w)$ a navíc $f(\mathbf{x}_{ci}) < f(\mathbf{x}_w)$	$\Longrightarrow$	$\mathbf{x}_w := \mathbf{x}_{ci}$
5) redukce :	$\mathbf{x}_i := \frac{1}{2}(\mathbf{x}_b + \mathbf{x}_i)$ pro všechna $\mathbf{x}_i \neq \mathbf{x}_b$		

Body 1 až 5 procházíme postupně shora dolů. Když některá z podmínek v bodech 1 až 4 není splněna, přejdeme na následující bod. Když splněna je, provedeme náhradu $\mathbf{x}_w$ podle příkazu za šipkou a transformace je hotova. Není-li splněna podmínka v žádném z bodů 1 až 4, provedeme redukci podle bodu 5.

Na začátku výpočtu je dána počáteční aproximace $\mathbf{x}_0 = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})^T$ a malé číslo δ. Další vrcholy $\mathbf{x}_i$ *startovacího simplexu* odvodíme z vrcholu $\mathbf{x}_0$ tak, že k jeho i-té složce $x_i^{(0)}$ přičteme číslo δ, tj. $\mathbf{x}_i = (x_1^{(0)}, \ldots, x_i^{(0)} + \delta, \ldots, x_n^{(0)})^T$, $i = 1, 2, \ldots, n$.

Simplex opakovaně transformujeme. Výpočet ukončíme a vrchol $\mathbf{x}_b$ považujeme za dostatečně dobrou aproximaci minima $\mathbf{x}^*$, pokud jsou vrcholy simplexu navzájem dosti blízko a funkční hodnoty v nich se málo liší, tj. když pro zadané tolerance ε_1, ε_2 platí

$$\|\mathbf{x}_i - \mathbf{x}_b\| < \varepsilon_1 \quad \text{a současně} \quad |f(\mathbf{x}_i) - f(\mathbf{x}_b)| < \varepsilon_2 \quad \text{pro každý vrchol } \mathbf{x}_i \neq \mathbf{x}_b\,. \tag{6.6}$$

Nelderova-Meadova metoda je *heuristická metoda* (heuristický nebo-li objevovací je takový postup, který je založen nejenom na logickém uvažování a zkušenostech, ale také na pozorování a experimentování). Metoda je vhodná pro minimalizaci funkcí menšího počtu proměnných, řekněme pro $n \leq 10$. Přestože o konvergenci metody je toho známo jen velmi málo, praxe hovoří v její prospěch: metoda je až překvapivě úspěšná. Proto je považována za metodu spolehlivou nebo-li *robustní*. Podstatnou nevýhodou metody je to, že je pomalá, zejména v blízkosti minima. Další minus představuje velký objem výpočtů. Přesto nejde o mrtvou metodu, což je zřejmé např. z toho, že je implementována v MATLABu jako funkce `fminsearch`.

Příklad 6.3. Funkce $f(x, y) = 70\,[(x-2)^4 + (x-2y)^2]$ nabývá minima pro $x^* = 2$, $y^* = 1$. Výpočet provedeme metodou Neldera-Meada, v níž zvolíme $\mathbf{x}_0 = (2{,}1; 0{,}7)^T$ a $\delta = 0{,}1$. Výpočty jsou prováděny přesně, do tabulky 6.1 však (kvůli úspoře místa) zapisujeme hodnoty zaokrouhlené na dvě desetinná místa. Pro každý bod zapisujeme ve sloupci pod sebou x-ovou souřadnici, y-ovou souřadnici a funkční hodnotu. V rámečku uvádíme nově určený lepší bod $\hat{\mathbf{x}}$ (a funkční hodnotu v něm). Pro $\varepsilon_1 = 10^{-4}$ a $\varepsilon_2 = 10^{-8}$ výpočet končí po 47 krocích a $\mathbf{x}_b \doteq (1{,}999953; 0{,}999976)^T$. □

Metoda největšího spádu je základní minimalizační metoda, která používá derivace účelové funkce. Takové *metody* se nazývají *gradientní*.

Začneme tím, že si vysvětlíme obecný princip *spádové metody*. Předpokládejme tedy, že jsme v bodu $\mathbf{x}_k$ a chceme se dostat blíže k minimu. Zvolíme směr $\mathbf{d}_k$, v němž funkce f

krok	typ	$\mathbf{x}_w$	$\mathbf{x}_g$	$\mathbf{x}_b$	$\bar{\mathbf{x}}$	$\mathbf{x}_r$	$\mathbf{x}_e$	$\mathbf{x}_{ce}$	$\mathbf{x}_{ci}$
1	expanze	2,20 0,70 44,91	2,10 0,70 34,31	2,10 0,80 17,51	2,10 0,75 –	2,00 0,80 11,20	1,90 0,85 2,81		
2	reflexe	2,10 0,70 34,31	2,10 0,80 17,51	1,90 0,85 2,81	2,00 0,82 –	1,90 0,95 0,01	1,80 1,08 8,69		
3	vnější kontrakce	2,10 0,80 17,51	1,90 0,85 2,81	1,90 0,95 0,01	1,90 0,90 –	1,70 1,00 6,87		1,80 0,95 0,81	
4	vnitřní kontrakce	1,90 0,85 2,81	1,80 0,95 0,81	1,90 0,95 0,01	1,85 0,95 –	1,80 1,05 6,41			1,88 0,90 0,41

Tab. 6.1: Příklad 6.3

klesá, a na polopřímce $\mathbf{x}_k + \lambda \mathbf{d}_k$, $\lambda \geq 0$, vybereme bod

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k \,, \tag{6.7}$$

v němž $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$. *Směrový vektor* $\mathbf{d}_k$ nazýváme *spádový* (ve směru $\mathbf{d}_k$ hodnota účelové funkce f padá dolů), odtud spádová metoda. Číslo λ_k se nazývá *parametr délky kroku* (je-li $\mathbf{d}_k$ jednotkový vektor, tj. když $\|\mathbf{d}_k\|_2 = 1$, pak $\lambda_k = \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2$ je vzdálenost bodů $\mathbf{x}_k$ a $\mathbf{x}_{k+1}$, tj. λ_k je délka kroku). λ_k dostaneme minimalizací funkce $\varphi(\lambda) = f(\mathbf{x}_k + \lambda \mathbf{d}_k)$ pro $\lambda \geq 0$. Minimum λ_k funkce $\varphi(\lambda)$ určíme přibližně pomocí několika málo kroků vhodné metody jednorozměrné minimalizace (přesná minimalizace je zbytečný přepych, neboť prostřednictvím λ_k určujeme jen mezivýsledek $\mathbf{x}_{k+1}$ na cestě k minimu $\mathbf{x}^*$). Určení λ_k tedy vyjádříme zápisem

$$\lambda_k \doteq \operatorname*{argmin}_{\lambda > 0} \varphi(\lambda) \,, \qquad \text{kde} \quad \varphi(\lambda) = f(\mathbf{x}_k + \lambda \mathbf{d}_k) \,. \tag{6.8}$$

Nyní se věnujme už vlastní metodě největšího spádu. Je známo, že funkce $f(\mathbf{x})$ nejrychleji klesá ve směru záporného gradientu. Označíme-li tedy gradient jako

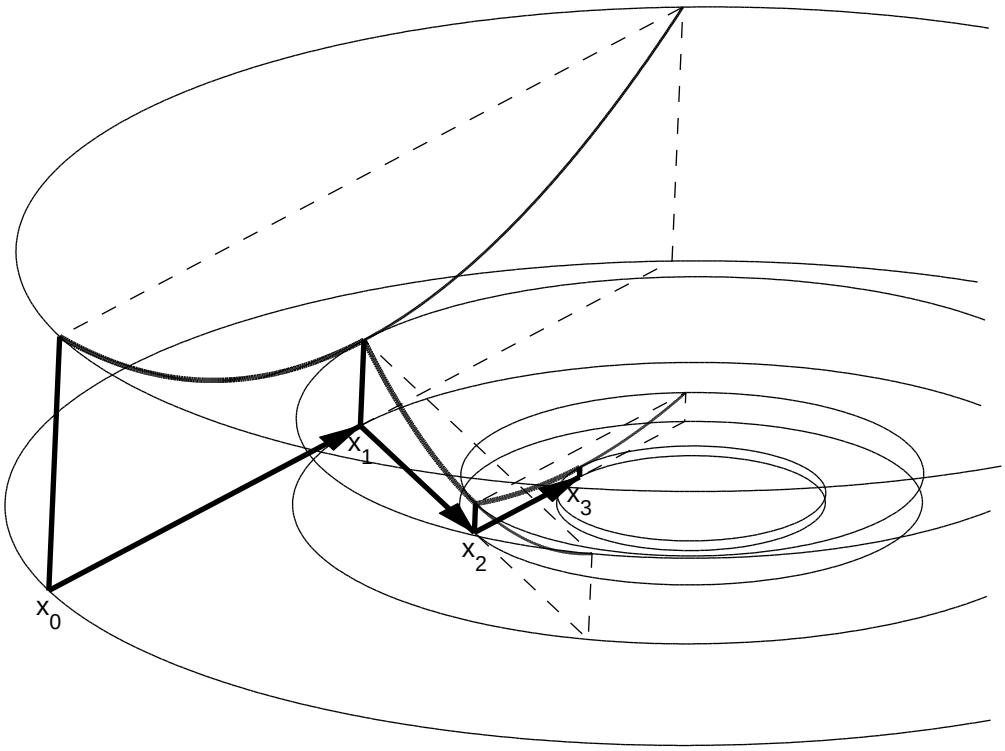
$$\mathbf{g}(\mathbf{x}) \equiv \nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right)^T ,$$

pak v metodě největšího spádu volíme jako spádový vektor

$$\mathbf{d}_k = -\mathbf{g}(\mathbf{x}_k) \,. \tag{6.9}$$

Výpočet ukončíme a $\mathbf{x}_{k+1}$ považujeme za uspokojivou aproximaci minima $\mathbf{x}^*$, když

$$\|\mathbf{g}(\mathbf{x}_{k+1})\| < \varepsilon \,, \tag{6.10}$$



Obr. 6.5: Princip spádových metod

kde ε je předepsaná tolerance (připomeňme, že v minimu $\mathbf{g}(\mathbf{x}^*) = \mathbf{o}$).

V počáteční fázi výpočtu, když jsme od minima ještě dosti daleko, dochází obvykle k poměrně rychlému poklesu hodnot účelové funkce. Zato v blízkosti minima je konvergence pomalá, jen lineární, tj. platí $\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq C\|\mathbf{x}_k - \mathbf{x}^*\|$, kde konstanta C je sice menší než jedna, ale často jen nepatrně.

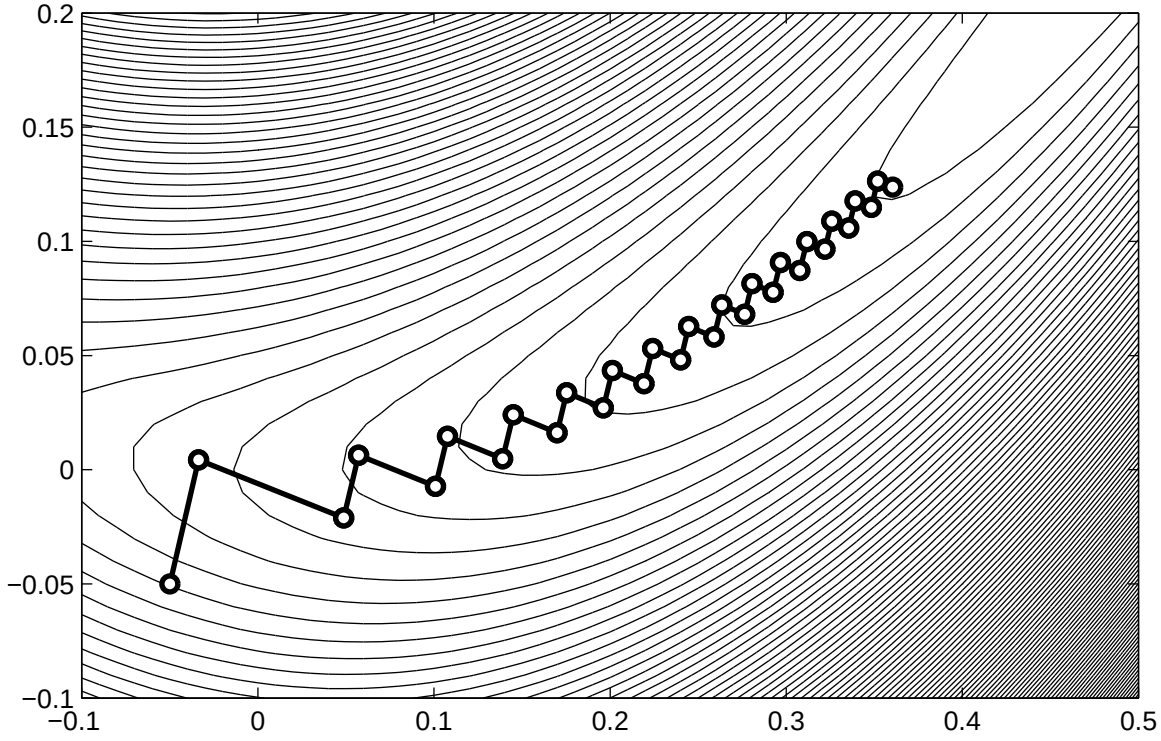
Cik-cak efekt. Pokud λ_k vypočteme přesně, má funkce $\varphi(\lambda)$ v bodě λ_k minimum, a proto

$$0 = \varphi'(\lambda_k) = \sum_{i=1}^{n} \frac{\partial f(\mathbf{x}_{k+1})}{\partial x_i} d_i^{(k)} = -\sum_{i=1}^{n} d_i^{(k+1)} d_i^{(k)} = -\mathbf{d}_{k+1}^T \mathbf{d}_k \,,$$

tj. směrové vektory $\mathbf{d}_{k+1}$ a $\mathbf{d}_k$ jsou navzájem kolmé. Ukážeme si, že právě tato vlastnost může být příčinou velmi pomalé konvergence.

Pro jednoduchost předpokládejme, že minimalizujeme funkci dvou proměnných. V tom případě si můžeme vypomoci jednoduchou představou: nacházíme se v terénu a chceme najít nejnižší bod, tj. dno nějaké prohlubně. Předpokládejme, že prohlubeň má tvar *podlouhlé zahnuté rokle*. Metoda největšího spádu nás nasměruje z počátečního stanoviště $\mathbf{x}_0$ dolů kolmo k vrstevnici $f(\mathbf{x}) = f(\mathbf{x}_0)$. Sestupujeme tak dlouho, dokud terén klesá. V nejnižším místě je další stanoviště $\mathbf{x}_1$. Zde se zastavíme, otočíme se o 90 stupňů a sestupujeme znovu dolů (kolmo k vrstevnici $f(\mathbf{x}) = f(\mathbf{x}_1)$) do dalšího stanoviště $\mathbf{x}_2$ atd. Je zřejmé, že vzdálenosti mezi jednotlivými stanovišti se budou postupně zkracovat. V blízkosti minima bude naše putování působit směšně: místo abychom do něj došli několika málo kroky, budeme se k němu spíše plížit než blížit po trase tvořené čím dál

tím kratšími navzájem kolmými úseky. Tento jev bývá označován jako *cik-cak efekt*. V takovém extrémním případě metoda největšího spádu selhává, neboť počet kroků potřebný k dosažení přijatelné aproximace minima je neúnosně velký. Obdobná situace nastane i v případě, když jednorozměrnou minimalizaci provádíme jen přibližně. □



Obr. 6.6: Cik-cak efekt

Příklad 6.4. Funkci $f(x,y) = x^2 + 2y^2 + xy - x$ minimalizujeme metodou největšího spádu. Jako počáteční aproximaci zvolíme $x_0 = 1$, $y_0 = 2$. Podrobně provedeme první dva kroky. Nejdříve určíme

$$\mathbf{g}(x,y) = \begin{pmatrix} 2x + y - 1 \\ x + 4y \end{pmatrix} \quad \text{a odtud} \quad \mathbf{d}(x,y) = \begin{pmatrix} -2x - y + 1 \\ -x - 4y \end{pmatrix}.$$

Pro

$$\mathbf{x}_0 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ tak dostaneme } \mathbf{d}_0 = \begin{pmatrix} -3 \\ -9 \end{pmatrix}, \text{ takže } \mathbf{x}_1 = \mathbf{x}_0 + \lambda_0 \mathbf{d}_0 = \begin{pmatrix} 1 - 3\lambda_0 \\ 2 - 9\lambda_0 \end{pmatrix}.$$

Funkce $\varphi(\lambda)$ je tedy tvaru

$$\begin{aligned} \varphi(\lambda) = f(\mathbf{x}_0 + \lambda \mathbf{d}_0) &= (1 - 3\lambda)^2 + 2(2 - 9\lambda)^2 + (1 - 3\lambda)(2 - 9\lambda) - (1 - 3\lambda) = \\ &= 198\lambda^2 - 90\lambda + 10\,. \end{aligned}$$

Minimalizaci funkce $\varphi(\lambda)$ umíme provést přesně:

$$\text{z podmínky } \varphi'(\lambda_0) = 396\lambda_0 - 90 = 0 \text{ dostaneme } \lambda_0 = \frac{5}{22}\,, \text{ takže}$$

$$\mathbf{x}_1 = \begin{pmatrix} 1-3\cdot 5/22 \\ 2-9\cdot 5/22 \end{pmatrix} = \frac{1}{22}\begin{pmatrix} 7 \\ -1 \end{pmatrix} \doteq \begin{pmatrix} 0{,}318 \\ -0{,}045 \end{pmatrix}.$$

Další krok provádíme podobně. Nejdříve určíme $\mathbf{d}_1$ a pak zapíšeme hledaný tvar pro $\mathbf{x}_2$:

$$\mathbf{d}_1 = \begin{pmatrix} -2\cdot 7/22-(-1/22)+1 \\ -7/22-4\cdot(-1/22) \end{pmatrix} = \frac{1}{22}\begin{pmatrix} 9 \\ -3 \end{pmatrix}, \quad \mathbf{x}_2 = \mathbf{x}_1+\lambda_1\mathbf{d}_1 = \frac{1}{22}\begin{pmatrix} 7+9\lambda_1 \\ -1-3\lambda_1 \end{pmatrix}.$$

Odtud pro $\varphi(\lambda) = f(\mathbf{x}_1+\lambda\mathbf{d}_1)$ po úpravě dostaneme

$$\varphi(\lambda) = \frac{1}{22^2}(72\lambda^2-90\lambda-110) \text{ a z podmínky } \varphi'(\lambda_1)=0 \text{ máme } \lambda_1 = \frac{5}{8}.$$

Dosazením za λ_1 do $\mathbf{x}_2$ nakonec obdržíme

$$\mathbf{x}_2 = \frac{1}{176}\begin{pmatrix} 101 \\ -23 \end{pmatrix} \doteq \begin{pmatrix} 0{,}574 \\ -0{,}131 \end{pmatrix}.$$

V tomto ilustračním příkladu metoda největšího spádu rychle konverguje k přesnému řešení $x^*=4/7$, $y^*=-1/7$: už hodnoty x_6, y_6 jsou na 6 desetinných míst přesné. □

Poznámka (*O spádových metodách pro řešení soustav lineárních rovnic*). Když $\mathbf{A}$ je pozitivně definitní matice, pak pro funkci $\mathbf{f}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}-\mathbf{x}^T\mathbf{b}$ je $\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x}-\mathbf{b}$ a $\mathbf{g}'(\mathbf{x}) = \mathbf{A}$, takže jediné minimum $\mathbf{x}^*$ funkce $\mathbf{f}(\mathbf{x})$ je řešením soustavy lineárních rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$. Pro spádový vektor $\mathbf{d}_k$ počítáme $\mathbf{x}_{k+1} = \mathbf{x}_k+\lambda_k\mathbf{d}_k$, přičemž délku λ_k kroku vyjádříme z podmínky $0{=}\varphi'(\lambda_k) = \frac{\mathrm{d}}{\mathrm{d}\lambda}f(\mathbf{x}_k+\lambda\mathbf{d}_k)\big|_{\lambda=\lambda_k} = \mathbf{d}_k^T\mathbf{A}\mathbf{d}_k\lambda_k - \mathbf{d}_k^T\mathbf{r}_k$, kde $\mathbf{r}_k = \mathbf{b}-\mathbf{A}\mathbf{x}_k$. (Ověřte!) Tak dostaneme metodu: $\mathbf{x}_0$ dáno, $\mathbf{r}_0 = \mathbf{b}-\mathbf{A}\mathbf{x}_0$, pro $k=0,1,\ldots$ počítáme

$$\mathbf{v}_k = \mathbf{A}\mathbf{d}_k\,, \quad \lambda_k = \frac{\mathbf{d}_k^T\mathbf{r}_k}{\mathbf{d}_k^T\mathbf{v}_k}\,, \quad \mathbf{x}_{k+1} = \mathbf{x}_k+\lambda_k\mathbf{d}_k\,, \quad \mathbf{r}_{k+1} = \mathbf{r}_k-\lambda_k\mathbf{v}_k\,. \tag{6.11}$$

(Vzorec pro $\mathbf{r}_{k+1}$ obdržíme tak, že do $\mathbf{r}_{k+1} = \mathbf{b}-\mathbf{A}\mathbf{x}_{k+1}$ dosadíme $\mathbf{x}_{k+1} = \mathbf{x}_k+\lambda_k\mathbf{d}_k$.) Výpočet ukončíme, když je reziduum $\mathbf{r}_k$ dostatečně malé, tj. když $\|\mathbf{r}_{k+1}\| < \varepsilon$. V metodě největšího spádu $\mathbf{d}_k = -\mathbf{g}(\mathbf{x}_k) = \mathbf{r}_k$. V praxi se však tato metoda nepoužívá, neboť obvykle konverguje příliš pomalu. Existuje lepší volba: zvolíme-li $\mathbf{d}_0 = \mathbf{r}_0$, a počítáme-li $\mathbf{d}_{k+1}$ pro $k\geq 0$ ze vzorce $\mathbf{d}_{k+1} = \mathbf{r}_{k+1}+\beta_k\mathbf{d}_k$, v němž $\beta_k = (\mathbf{r}_{k+1}^T\mathbf{r}_{k+1})/(\mathbf{r}_k^T\mathbf{r}_k)$, dostaneme *metodu sdružených gradientů*, která konverguje podstatně rychleji, viz [13], [22], [7].

Newtonova metoda je gradientní metoda, která se pokouší najít minimum jako řešení $\mathbf{x}^*$ soustavy nelineárních rovnic $\mathbf{g}(\mathbf{x}) = \mathbf{o}$ Newtonovou metodou. Takovým řešením ale může být každý stacionární bod funkce f (bod $\mathbf{x}^*$, ve kterém pro funkci f platí $\nabla f(\mathbf{x}^*) = \mathbf{o}$, se nazývá *stacionární bod* funkce f), tedy také maximum nebo sedlový bod (obdoba inflexního bodu pro funkci jedné proměnné). Budeme tedy potřebovat Jacobiovu matici

$$\mathbf{g}'(\mathbf{x}) \equiv \mathbf{H}(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{pmatrix},$$

která se v tomto případě nazývá *Hessova matice* funkce f. Výpočet provádíme podle vzorců (5.12), tj. najdeme řešení $\mathbf{d}_k$ lineární soustavy rovnic

$$\mathbf{H}(\mathbf{x}_k)\,\mathbf{d}_k = -\mathbf{g}(\mathbf{x}_k) \qquad \text{a pak určíme} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k\,. \tag{6.12}$$

Všimněte si, že vzorec (6.12) pro $\mathbf{x}_{k+1}$ je formálně stejný jako vzorec (6.7), když v něm $\lambda_k = 1$. Pro ukončení výpočtu použijeme stop kritérium (6.10). Směrový vektor

$$\mathbf{d}_k = -\mathbf{H}^{-1}(\mathbf{x}_k)\mathbf{g}(\mathbf{x}_k) \tag{6.13}$$

obecně není spádový. Dá se však ukázat, že v blízkosti minima vektor $\mathbf{d}_k$ spádový je.

Z odstavce 5.3 už víme, že Newtonova metoda konverguje, když je počáteční aproximace $\mathbf{x}_0$ dostatečně blízko řešení $\mathbf{x}^*$, a že v tom případě je konvergence rychlá, totiž kvadratická. Na druhé straně, když $\mathbf{x}_0$ není dosti blízko $\mathbf{x}^*$, Newtonova metoda vůbec konvergovat nemusí. Proto se nabízí možnost začít výpočet metodou největšího spádu a dokončit ho Newtonovou metodou. Tento poznatek je východiskem pro odvození řady efektivních metod. Mezi takové patří například

Kvazinewtonovské metody. Výpočet probíhá podobně jako v metodě největšího spádu. Směrový vektor $\mathbf{d}_k$ je však tvaru $\mathbf{d}_k = -\mathbf{B}_k\mathbf{g}(\mathbf{x}_k)$. Matice $\mathbf{B}_k$ se vybírají tak, aby vektor $\mathbf{d}_k$ byl spádový. Pro $k = 0$ je přitom $\mathbf{B}_0 = \mathbf{I}$ jednotková matice, takže $\mathbf{d}_0 = -\mathbf{g}(\mathbf{x}_0)$ je směr metody největšího spádu. Čím víc se $\mathbf{x}_k$ blíží minimu, tím je $\mathbf{B}_k$ lepší aproximací $\mathbf{H}^{-1}(\mathbf{x}_k)$, takže $\mathbf{d}_k$ se blíží směrovému vektoru $-\mathbf{H}^{-1}(\mathbf{x}_k)\mathbf{g}(\mathbf{x}_k)$ Newtonovy metody. To je jen velmi hrubý popis, detaily viz např. [16], [22].

Příklad 6.5. Pro testování kvality minimalizačních metod se používá mezi jinými také tzv. banánová funkce $f(x,y) = 100(y - x^2)^2 + (1 - x)^2$. Průběh funkce f navozuje představu hluboké rokle se strmými stěnami, jejíž zakřivené dno o rovnici $y = x^2$ (vzdáleně připomínající banán) se jen velmi mírně svažuje k minimu v bodě $x^* = y^* = 1$.

Minimum hledejme Newtonovou metodou, tj. soustavu rovnic

$$\frac{\partial f(x,y)}{\partial x} = -400x(y - x^2) - 2(1 - x) = 0\,, \qquad \frac{\partial f(x,y)}{\partial y} = 200(y - x^2) = 0$$

řešíme Newtonovou metodou: v každém kroku vypočteme a_k, b_k ze soustavy rovnic

$$\begin{pmatrix} 1200x_k^2 - 400y_k + 2 & -400x_k \\ -400x_k & 200 \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} 400x_k(y_k - x_k^2) + 2(1 - x_k) \\ -200(y_k - x_k^2) \end{pmatrix}$$

a potom určíme další aproximaci

$$x_{k+1} = x_k + a_k\,, \qquad y_{k+1} = y_k + b_k\,.$$

Pro numerický experiment jsme zvolili počáteční aproximaci $x_0 = 3$, $y_0 = 2$. Výpočet potvrdil velmi rychlou konvergenci Newtonovy metody, neboť už v pátém kroku jsme dostali aproximaci, která měla 16 platných cifer. To je skvělé!

Pro srovnání jsme tutéž úlohu řešili také metodou největšího spádu. Jednorozměrnou minimalizaci (6.8) jsme provedli přibližně metodou zlatého řezu na intervalu $\langle 0, 1\rangle$ s přesností 10^{-4}. Pro $\varepsilon = 10^{-3}$ byla podmínka (6.10) splněna až pro $k = 1324$, kdy jsme dostali aproximaci $(1{,}001; 1{,}001)^T$. To je neuspokojivé.

To metoda Neldera-Meada si vedla lépe. Pro výpočet jsme vybrali parametry $\delta = 0{,}2$, $\varepsilon_1 = 10^{-2}$ a $\varepsilon_2 = 10^{-4}$. Podmínka (6.6) byla splněna po 57 krocích, kdy jsme dostali aproximaci $(1{,}001; 1{,}001)^T$. To je přijatelné. □

Poznámka. Řešení $\mathbf{x}^*$ soustavy n nelineárních rovnic $\mathbf{f}(\mathbf{x}) = \mathbf{o}$ můžeme získat jako bod, v němž funkce $h(\mathbf{x}) = \sum_{i=1}^{n} [f_i(\mathbf{x})]^2$ (tj. součet čtverců reziduií) nabývá svého globálního minima (neboť $0 = h(\mathbf{x}^*) = \min_{\mathbf{x}\in\mathbf{R}^n} h(\mathbf{x})$, právě když $\mathbf{f}(\mathbf{x}^*) = \mathbf{o}$). Minimalizaci $h(\mathbf{x})$ lze provádět pomocí algoritmů pro řešení nelineární úlohy nejmenších čtverců, viz např. [16].

6.3. Cvičení

6.1. Naprogramujte metodu zlatého řezu a spočtěte minima funkcí $f(x)$ s přesností $\varepsilon = 10^{-6}$.
(a) $f(x) = x^4 - 3x^3 + 6x^2 - 4x$ na intervalu $\langle 0, 1\rangle$
(b) $f(x) = x^4 - 2x^3 + 1{,}5x^2 - \sin x + 2$ na intervalu $\langle 0, 2\rangle$
(c) $f(x) = x^2 \sin(x^2 + x - 1) - x$ na intervalu $\langle 0, 1\rangle$

[(a) $x^* \doteq 0{,}459211$ (b) $x^* \doteq 0{,}844759$ (c) $x^* \doteq 0{,}640572$.]

6.2. Naprogramujte metodu kvadratické interpolace a spočtěte minima funkcí $f(x)$ s přesností $\varepsilon = 10^{-6}$.
(a) $f(x) = \cos(x^2 + x) \sin x$ na intervalu $\langle 1, 2\rangle$
(b) $f(x) = x \sin x - \cos x + x$ na intervalu $\langle -1, 1\rangle$
(c) $f(x) = -\cos[(x^2 - x + 1)/(x + 1)]$ na intervalu $\langle 0, 2\rangle$

[(a) $x^* \doteq 1{,}357381$ (b) $x^* \doteq -0{,}344607$ (c) $x^* \doteq 0{,}732051$.]

6.3. Naprogramujte metodu Neldera-Meada a spočtěte minima funkcí $f(x, y)$ s přesností $\varepsilon_1 = 10^{-6}$ a $\varepsilon_2 = 10^{-8}$.
(a) $f(x, y) = x^2 + 5y^2 + 2xy - 3x - y$ pro $\mathbf{x}_0 = (2{,}1; 0{,}7)^T$
(b) $f(x, y) = x^2 + y^4 + x - \sin(xy)$ pro $\mathbf{x}_0 = (0; 0)^T$
(c) $f(x, y) = (x - y + xy)^2 + x^2 - xy + y^2 + x$ pro $\mathbf{x}_0 = (0; 0)^T$

[(a) $\mathbf{x}^* \doteq (1{,}75; -0{,}25)^T$ (b) $\mathbf{x}^* \doteq (-0{,}754039; -0{,}556309)^T$ (c) $\mathbf{x}^* \doteq (-0{,}675251; -0{,}385880)^T$.]

6.4. Naprogramujte metodu největšího spádu a spočtěte minima funkcí $f(x, y)$ s přesností $\varepsilon = 0{,}1$.
(a) $f(x, y) = x^4 + y^4 + x^2 - 2yx + y^2 - x$ pro $\mathbf{x}_0 = (0; 0)^T$
(b) $f(x, y) = (x + y)^4 + x^2 + y^2 - x$ pro $\mathbf{x}_0 = (1; 1)^T$

[(a) $\mathbf{x}^* \doteq (0{,}6; 0{,}4)^T$ (b) $\mathbf{x}^* \doteq (0{,}4; -0{,}1)^T$.]

6.5. Naprogramujte Newtonovu metodu a spočtěte minima funkcí ze cvičení 6.4 s přesností $\varepsilon = 10^{-6}$. Výsledek z cvičení 6.4 použijte jako počáteční aproximaci.

[(a) $\mathbf{x}^* \doteq (0{,}561992; 0{,}416984)^T$ (b) $\mathbf{x}^* \doteq (0{,}420582; -0{,}079418)^T$.]

6.6. Naprogramujte metodu největšího spádu pro řešení soustav lineárních rovnic, viz (6.11), kde položíte $\mathbf{d}_k = \mathbf{r}_k$. Zvolte $\varepsilon = 10^{-5}$ a iterační proces ukončete podmínkou $\|\mathbf{r}_k\|_2 < \varepsilon\|\mathbf{b}\|_2$. (a) Kolik iterací se vykoná při řešení soustavy z cvičení 2.12? (b) Kolik iterací se vykoná při řešení soustavy $\mathbf{K}\mathbf{x} = \mathbf{b}$, kde $\mathbf{K}$ je matice řádu n^2 popsaná v příkladě 2.5 a $\mathbf{b}$ je vektor, který má všechny složky $b_i = (n + 1)^{-2}$? Volte $n = 5, 10, 15, 20, 25$ a porovnejte s obrázkem 2.3!

[(a) 77 (b) $74, 254, 584, 1000, 1566$.]

6.7. Naprogramujte metodu sdružených gradientů pro řešení soustav lineárních rovnic, viz (6.11), kde položíte $\mathbf{d}_0 = \mathbf{r}_0$ a $\mathbf{d}_{k+1}$ pro $k \geq 0$ počítáte ze vzorce $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$, v němž $\beta_k = (\mathbf{r}_{k+1}^T \mathbf{r}_{k+1})/(\mathbf{r}_k^T \mathbf{r}_k)$. Zvolte $\varepsilon = 10^{-5}$ a iterační proces ukončete podmínkou $\|\mathbf{r}_k\|_2 < \varepsilon\|\mathbf{b}\|_2$. (a) Kolik iterací se vykoná při řešení soustavy z cvičení 2.12? (b) Kolik iterací se vykoná při řešení soustavy $\mathbf{K}\mathbf{x} = \mathbf{b}$, kde $\mathbf{K}$ je matice řádu n^2 popsaná v příkladě 2.5 a $\mathbf{b}$ je vektor, který má všechny složky $b_i = (n + 1)^{-2}$? Volte $n = 5, 10, 15, 20, 25$ a porovnejte s obrázkem 2.3!

[(a) 3 (b) $5, 14, 22, 29, 36$.]

Literatura

[1] I.S. Berezin, N.P. Židkov: *Čislennyje metody I,II,* Nauka, Moskva, 1962.

[2] J. P. Berrut, L. N. Trefethen: *Barycentric Lagrange Interpolation,* SIAM Rev., Vol. 46, No. 3, pp. 501–517, 2004.

[3] G. Dahlquist, G. Å Bjőrk: *Numerical Methods,* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[4] M. Fiedler: *Speciální matice a jejich použití v numerické matematice,* SNTL, Praha, 1981.

[5] D. Hanselman, B. Littlefield: *Mastering MATLAB 7,* Pearson Prentice Hall, Upper Saddle River, NJ, 2005.

[6] G. Hämmerlin, K. H. Hoffmann: *Numerical Mathematics,* Springer-Verlag, Berlin, 1991.

[7] M. T. Heath: *Scientific Computing. An Introductory Survey,* McGraw-Hill, New York, 2002.

[8] I. Horová, J. Zelinka: *Numerické metody,* učební text Masarykovy univerzity, Brno, 2004.

[9] J. Kobza: *Splajny,* učební text Palackého univerzity, Olomouc, 1993.

[10] J. Klapka, J. Dvořák, P. Popela: *Metody operačního výzkumu,* učební text, FSI VUT Brno, 2001.

[11] J. H. Mathews, K. D. Fink: *Numerical Methods Using MATLAB,* Pearson Prentice Hall, New Jersey, 2004.

[12] MATLAB: *Mathematics,* Version 7, The MathWorks, Inc., Natick, 2004.

[13] G. Meurant: *Computer Solution of Large Linear Systems,* Elsevier, Amsterodam, 1999.

[14] S. Míka: *Numerické metody algebry,* SNTL, Praha, 1985.

[15] C. B. Moler: *Numerical Computing with MATLAB,* Siam, Philadelphia, 2004. `http://www.mathworks.com/moler`.

[16] J. Nocedal, S. J. Wright: *Numerical Optimization, Springer Series in Operations Research,* Springer, Berlin, 1999.

[17] A. Quarteroni, R. Sacco, F. Saleri: *Numerical Mathematics,* Springer, Berlin, 2000.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling: *Numerical Recipes in Pascal, The Art of Scientific Computing,* Cambridge University Press, Cambridge, 1990.

[19] P. Přikryl: *Numerické metody matematické analýzy,* SNTL, Praha, 1985.

[20] A. R. Ralston: *Základy numerické matematiky,* Academia, Praha, 1973.

[21] K. Rektorys: *Přehled užité matematiky I,II,* Prometheus, Praha, 1995.

[22] J. Stoer, R. Bulirsch: *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1993.

[23] E. Vitásek: *Numerické metody,* SNTL, Praha, 1987.

[24] W. Y. Yang, W. Cao, T. S. Chung, J. Morris: *Applied Numerical Methods Using Matlab,* John Willey & Sons, New Jersey, 2005.