UČEBNÍ TEXTY VYSOKÝCH ŠKOL

Vysoké učení technické v Brně Fakulta strojního inženýrství

Doc. RNDr. Libor Čermák, CSc.

Numerické metody

Ústav matematiky FSI VUT v Brně

Obsah

Předmluva

Skripta *Numerické metody* jsou věnována tradičním tématům numerické matematiky: zdroje chyb a jejich šíření, řešení soustav lineárních rovnic, metoda nejmenších čtverců, aproximace funkcí, numerické derivování a integrování, řešení nelineárních rovnic, výpočet vlastních čísel a vlastních vektorů, řešení počátečních a okrajových úloh pro obyčejné diferenciální rovnice a řešení parciálních diferenciálních rovnic. ř Do skripta jsou ke každému tématu zařazeny 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. Některé z nich jsou přesto do textu zařazeny.

Při zpracování skript jsem vycházel z osvědčených učebnic numerické matematiky, jakými jsou např. knihy [10], [55], [42], [21], [37], [31], [58] a z vynikajících monografií, mezi nimi zejména [26], [50], [51], [29], [18], [5], [61].

Pokud jde o české zdroje, nejvíce podnětů jsem čerpal z knih [41], [34], [22], [58], [59] a ze skript [36], [9] a [35].

Moderní česky psaná monografie numerických metod v současnosti není k dispozici. Kromě výše uvedených knih lze však v češtině číst také [43], vybrané kapitoly věnované numerickým metodám v [44] a [58].

Brno, červen 2020 Libor Čermák

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 $\tilde{x} = \pm\, d_1{,}d_2d_3 \ldots d_p \cdot 10^e$, 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 stabilita 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 stabilitě algoritmů uvedeme postupně v následujících kapitolách.

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 (stručně SLR)

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

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 SLR 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 rovnici (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

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ m_{21} & 1 & 0 & \cdots & 0 & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n-1,1} & m_{n-1,2} & m_{n-1,3} & \ldots & 1 & 0 \\ m_{n1} & m_{n2} & m_{n3} & \cdots & m_{n,n-1} & 1 \end{pmatrix}. \tag{2.7}$$

Pak

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

Důkaz. Nechť

$$\mathbf{M}_k = \begin{pmatrix} 1 & & 0 & 0 & 0 & 0 & \ldots & 0 \\ & \ddots & & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & & 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & -m_{k+1,k} & 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & -m_{k+2,k} & 0 & 1 & & 0 \\ \vdots & \ldots & \vdots & \vdots & \vdots & & \ddots & \\ 0 & \ldots & 0 & -m_{nk} & 0 & 0 & \ldots & 1 \end{pmatrix} = \mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T,$$

kde $\mathbf{I}$ je jednotková matice, $\mathbf{m}_k = (0, \ldots, 0, m_{k+1,k}, \ldots, m_{nk})^T$, $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$, $\mathbf{e}_k$ je k-tý sloupec jednotkové matice. Protože $\mathbf{e}_k^T \mathbf{m}_k = 0$, dostaneme

$$(\mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T)(\mathbf{I} + \mathbf{m}_k \mathbf{e}_k^T) = \mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T + \mathbf{m}_k \mathbf{e}_k^T + \mathbf{m}_k \mathbf{e}_k^T \mathbf{m}_k \mathbf{e}_k^T = \mathbf{I}.$$

Vidíme tedy, že pro matici $\mathbf{L}_k$ inverzní k matici $\mathbf{M}_k$ platí

$$\mathbf{L}_k = \mathbf{M}_k^{-1} = \mathbf{I} + \mathbf{m}_k \mathbf{e}_k^T .$$

k-tý krok LU rozkladu, ve kterém se nulují poddiagonální prvky v k-tém sloupci, lze vyjádřit zápisem $\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{A}^{(k-1)}$, $k = 1, 2, \ldots, n-1$, přičemž $\mathbf{A}^{(0)} = \mathbf{A}$. Proto

$$\mathbf{M}_{n-1}\mathbf{M}_{n-2} \ldots \mathbf{M}_1 \mathbf{A} = \mathbf{U}$$

je horní trojúhelníková matice. Odtud

$$\mathbf{A} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1} \ldots \mathbf{M}_{n-1}^{-1}\mathbf{U} = \mathbf{L}\mathbf{U}, \quad \text{kde} \quad \mathbf{L} = \mathbf{L}_1\mathbf{L}_2 \ldots \mathbf{L}_{n-1}. \qquad \square$$

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 $k < i$. V MATLABu lze pro LU rozklad použít funkci `lu`.

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. [17], [33]).

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

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

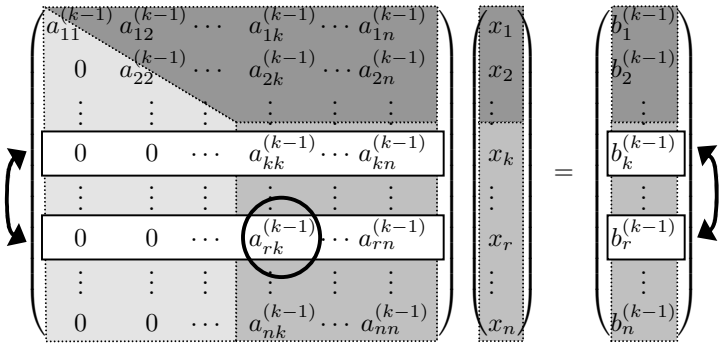
Kde vznikl problém? Nedošlo k žádnému hromadění chyb způsobenému prováděním tisíců operací. Matice soustavy není blízká matici singulární. Potíž je jinde, působí ji malý pivot ve druhém kroku eliminace. Tím vznikne multiplikátor -2400 a v důsledku toho má poslední rovnice koeficienty zhruba 1000 krát 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 $\mathbf{b}^{(k-1)}$, viz Obr. 2.1. Poddiagonální prvky v k-tém sloupci matice $\mathbf{A}^{(k)}$ eliminujeme stejně jako v algoritmu GEMz. Řešení SLR s částečným výběrem hlavních prvků v MATLABu dostaneme pomocí příkazu `x=A\b`. Použít lze také funkci `linsolve`.



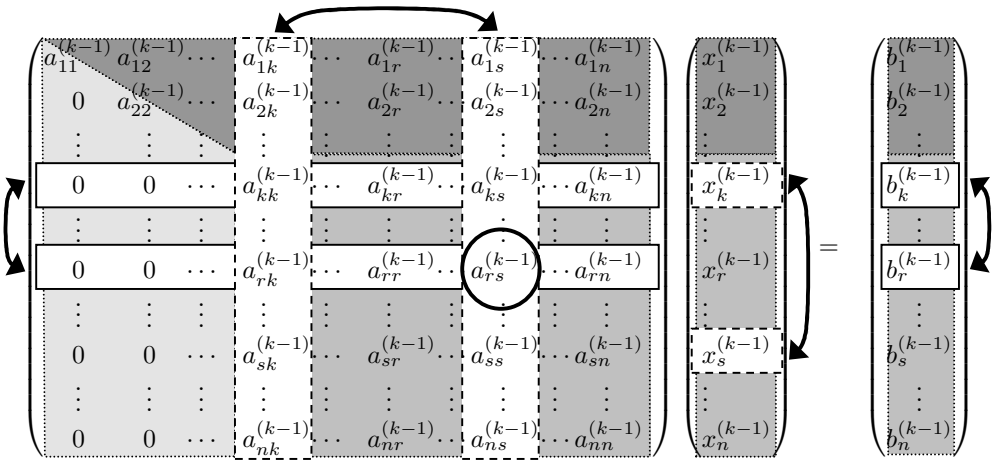
Obr. 2.1: GEM s částečným výběrem hlavního prvku (v kroužku)

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, viz Obr. 2.2. Ze soustavy rovnic $\mathbf{A}^{(k-1)}\mathbf{x}^{(k-1)} = \mathbf{b}^{(k-1)}$ 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.



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

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 $\mathbf{P}^{(n-1)}$.

Důkaz. k-tý krok LU rozkladu, ve kterém se nulují poddiagonální prvky v k-tém sloupci, lze vyjádřit zápisem $\mathbf{A}^{(k)} = \mathbf{M}_k\mathbf{P}_k\mathbf{A}^{(k-1)}$, $k = 1, 2, \ldots, n-1$. Proto

$$(\mathbf{M}_{n-1}\mathbf{P}_{n-1})(\mathbf{M}_{n-2}\mathbf{P}_{n-2})\ldots(\mathbf{M}_3\mathbf{P}_3)(\mathbf{M}_2\mathbf{P}_2)(\mathbf{M}_1\mathbf{P}_1)\mathbf{A} = \mathbf{U}$$

je horní trojúhelníková matice. Protože $\mathbf{P}_k^{-1} = \mathbf{P}_k$, $\mathbf{M}_k^{-1} = \mathbf{L}_k$, dostaneme

$$\mathbf{P}_{n-1}\mathbf{P}_{n-2}\ldots\mathbf{P}_3\mathbf{P}_2\mathbf{P}_1\mathbf{A} =$$

$$(\mathbf{P}_{n-1}(\mathbf{P}_{n-2}(\ldots(\mathbf{P}_3(\mathbf{P}_2(\mathbf{P}_1\mathbf{P}_1\mathbf{L}_1)\mathbf{P}_2\mathbf{L}_2)\mathbf{P}_3\mathbf{L}_3)\ldots)\mathbf{P}_{n-2}\mathbf{L}_{n-2})\mathbf{P}_{n-1}\mathbf{L}_{n-1})\mathbf{U}.$$

Označíme-li

$$\mathbf{P} = \mathbf{P}_{n-1}\mathbf{P}_{n-2}\ldots\mathbf{P}_3\mathbf{P}_2\mathbf{P}_1,$$

$$\mathbf{L} = (\mathbf{P}_{n-1}(\mathbf{P}_{n-2}\ldots(\mathbf{P}_3(\mathbf{P}_2(\mathbf{P}_1\mathbf{P}_1\mathbf{L}_1)\mathbf{P}_2\mathbf{L}_2)\mathbf{P}_3\mathbf{L}_3)\ldots\mathbf{P}_{n-2}\mathbf{L}_{n-2})\mathbf{P}_{n-1}\mathbf{L}_{n-1}),$$

pak $\mathbf{PA} = \mathbf{LU}$. □

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

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

V MATLABu lze pro výpočet determinantu použít funkci `det`.

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 $mn^2 + O(n)$ operací. Je-li počet m pravých stran malý ve srovnání s počtem n rovnic, jsou náklady na provedení LU rozkladu 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. Vzhledem ke speciální pravé straně $\mathbf{b} = \mathbf{I}$ lze jak přímý tak zpětný chod GEM provést pomocí $\frac{1}{2}n^3 + O(n^2)$ operací, takže celkem potřebujeme $n^3 + O(n^2)$ operací.

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 $s = p = q = O(1)$ potřebuje LU rozklad bez výběru hlavních prvků $ns^2+O(1)$ operací a zpětný chod $ns+O(1)$ operací. Symbol $O(1)$ přitom reprezentuje číslo nezávislé na n.

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

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

Norma matice je nezáporné číslo, které reprezentuje velikost matice. Normu matice $\mathbf{A}$ značíme $\|\mathbf{A}\|$. Norma definovaná předpisem

$$\|\mathbf{A}\| = \max_{\mathbf{x} \neq \mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\| \tag{2.19}$$

se nazývá *přirozená* maticová norma *přidružená* k vektorové normě, pomocí níž je definována. Přirozená maticová norma přidružená k vektorové ℓ_p-normě

$$\|\mathbf{A}\|_p = \max_{\mathbf{x} \neq \mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \max_{\|\mathbf{x}\|_p=1} \|\mathbf{A}\mathbf{x}\|_p .$$

Není těžké ověřit, že

$$\|\mathbf{A}\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| \qquad \text{(maximum sloupcových součtů)}\,,$$

$$\|\mathbf{A}\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| \qquad \text{(maximum řádkových součtů)}\,.$$

Přirozená maticová norma přidružená k vektorové ℓ_2-normě je známa jako *spektrální norma*. Přitom

$$\|\mathbf{A}\|_2 = \sigma_{max}(\mathbf{A}) = \sqrt{\lambda_{max}(\mathbf{A}^T\mathbf{A})},$$

kde $\sigma_{max}(\mathbf{A})$ je největší singulární číslo matice $\mathbf{A}$, viz (3.12), a $\lambda_{max}(\mathbf{A}^T\mathbf{A})$ je největší vlastní číslo matice $\mathbf{A}^T\mathbf{A}$, viz (3.13). Pro symetrickou matici

$$\|\mathbf{A}\|_2 = \varrho(\mathbf{A}) := \max_{1\le i\le n} |\lambda_i(\mathbf{A})|,$$

kde $\lambda_i(\mathbf{A})$, $i = 1, 2, \ldots, n$, jsou vlastní čísla matice $\mathbf{A}$, viz kapitola 7.1.5. Číslo $\varrho(\mathbf{A})$ je známo jako *spektrální poloměr* matice $\mathbf{A}$. Pro pozitivně definitní matici

$$\|\mathbf{A}\|_2 = \lambda_{max}(\mathbf{A}).$$

Jestliže vektorová norma $\|\cdot\|_V$ a maticová norma $\|\cdot\|_M$ splňují vztah

$$\|\mathbf{Ax}\|_V \le \|\mathbf{A}\|_M \cdot \|\mathbf{x}\|_V, \tag{2.20}$$

říkáme, že maticová norma $\|\cdot\|_M$ je s vektorovou normou $\|\cdot\|_V$ *souhlasná*. Snadno nahlédneme, že přirozená maticová norma je s přidruženou maticovou normou souhlasná.

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

$$\|\mathbf{A}\|_F = \left(\sum_{i,j=1}^{n} a_{ij}^2 \right)^{1/2}, \tag{2.21}$$

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

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

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. $\|\mathbf{AB}\| \le \|\mathbf{A}\| \cdot \|\mathbf{B}\|$ pro čtvercové matice $\mathbf{A}$, $\mathbf{B}$ (submultiplikativnost).

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

Číslo podmíněnosti. Pro posouzení podmíněnosti úlohy „najít řešení $\mathbf{x}$ soustavy lineárních rovnic $\mathbf{Ax} = \mathbf{b}$“ hraje klíčovou roli tzv. *číslo podmíněnosti* $\kappa(\mathbf{A})$ matice $\mathbf{A}$,

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|. \tag{2.22}$$

Konkrétní hodnota čísla podmíněnosti závisí na použité vektorové normě. Pro maticovou p-normu budeme značit $\kappa_p(\mathbf{A}) = \|\mathbf{A}\|_p \cdot \|\mathbf{A}^{-1}\|_p$, podobně $\kappa_F(\mathbf{A}) = \|\mathbf{A}\|_F \cdot \|\mathbf{A}^{-1}\|_F$.

Všimněme si některých vlastností čísla podmíněnosti.

1. $\kappa_p(\mathbf{A}) \geq 1$.
2. Pro jednotkovou matici $\kappa_p(\mathbf{I}) = 1$.
3. Je-li $\mathbf{A}$ singulární, pak $\kappa_p(\mathbf{A}) = \infty$.
4. Když matici $\mathbf{A}$ vynásobíme číslem $c \neq 0$, pak $\kappa_p(c\mathbf{A}) = \kappa_p(\mathbf{A})$.
5. Pro diagonální matici $\mathbf{D} = \operatorname{diag}\{d_1, d_2, \ldots, d_n\}$ je $\kappa_p(\mathbf{D}) = \max\limits_i |d_i| \,/\, \min\limits_i |d_i|$.

Poznamenejme, že vlastnosti 1–4 platí pro každou přirozenou maticovou normu. Z vlastností 4 a 5 plyne, že $\kappa_p(\mathbf{A})$ 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 $\kappa_p(\mathbf{A}) = 1$. Soustava rovnic s takovou maticí se chová spíše jako soustava s jednotkovou maticí než jako soustava s maticí téměř singulární. V MATLABu lze čísla podmíněnosti matice $\kappa_1(\mathbf{A})$, $\kappa_2(\mathbf{A})$, $\kappa_\infty(\mathbf{A})$ a $\kappa_F(\mathbf{A})$ určit pomocí funkce `cond`.

K analýze podmíněnosti SLR budeme potřebovat následující

Lemma 2.1. Nechť $\|\mathbf{X}\| < 1$. Pak $\|(\mathbf{I} - \mathbf{X})^{-1}\| \leq \dfrac{1}{1 - \|\mathbf{X}\|}$.

Důkaz. Pro matici $\mathbf{X} = \{x_{k\ell}\}_{k\ell=1}^n$ na základě ekvivalence norem $\|\cdot\|$ a $\max_{1\leq k,\ell\leq n} |x_{k\ell}|$ platí $|x_{k\ell}| \leq c\|\mathbf{X}\|$, $1 \leq k, \ell \leq n$, kde c je konstanta. Tedy $|(\mathbf{X}^i)_{k\ell}| \leq c\|\mathbf{X}^i\| \leq c\|\mathbf{X}\|^i$, což znamená, že každá složka řady $\sum_{i=1}^\infty \mathbf{X}^i$ je majorizována konvergentní geometrickou řadou $c\sum_{i=1}^\infty \|\mathbf{X}\|^i$. Proto $\mathbf{S}_n = \sum_{i=1}^n \mathbf{X}^i$ konverguje k matici $\mathbf{S} = \sum_{i=1}^\infty \mathbf{X}^i$. Dále

$$(\mathbf{I} - \mathbf{X})(\mathbf{I} + \mathbf{X} + \mathbf{X}^2 + \cdots + \mathbf{X}^n) = \mathbf{I} - \mathbf{X}^{n+1} \to \mathbf{I},$$

neboť $\|\mathbf{X}^{n+1}\| \leq \|\mathbf{X}\|^{n+1} \to 0$. Tedy

$$(\mathbf{I} - \mathbf{X})\mathbf{S} = \mathbf{I} \implies \mathbf{S} = \sum_{i=1}^\infty \mathbf{X}^i = (\mathbf{I} - \mathbf{X})^{-1}.$$

Konečně

$$\left\|(\mathbf{I} - \mathbf{X})^{-1}\right\| = \left\|\sum_{i=1}^\infty \mathbf{X}^i\right\| \leq \sum_{i=1}^\infty \left\|\mathbf{X}^i\right\| \leq \sum_{i=1}^\infty \|\mathbf{X}\|^i = \frac{1}{1 - \|\mathbf{X}\|}. \qquad \square$$

Analýza podmíněnosti. Nechť $\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$ je pozměněná matice soustavy a $\tilde{\mathbf{b}} = \mathbf{b} + \tilde{\mathbf{b}}$ je pozměněná pravá strana. Řešení $\tilde{\mathbf{x}}$ soustavy $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ vyjádřeme ve tvaru $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$. Z rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$, $(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$ dostaneme $(\Delta\mathbf{A})\mathbf{x} + (\mathbf{A} + \Delta\mathbf{A})\Delta\mathbf{x} = \Delta\mathbf{b}$. Odtud

$$\begin{aligned}\Delta\mathbf{x} =& (\mathbf{A} + \Delta\mathbf{A})^{-1}(-(\Delta\mathbf{A})\mathbf{x} + \Delta\mathbf{b}) = \left[\mathbf{A}(\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A})\right]^{-1}(-(\Delta\mathbf{A})\mathbf{x} + \Delta\mathbf{b}) = \\ & (\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A})^{-1}\mathbf{A}^{-1}(-(\Delta\mathbf{A})\mathbf{x} + \Delta\mathbf{b}).\end{aligned}$$

Předpokládejme, že porucha $\Delta\mathbf{A}$ je tak malá, že $\|\mathbf{A}^{-1}\Delta\mathbf{A}\| \le \|\mathbf{A}^{-1}\| \cdot \|\Delta\mathbf{A}\| < 1$. Užitím lemmatu 2.1 pak postupně dostaneme

$$\begin{aligned}\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le& \|(\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A})^{-1}\| \cdot \|\mathbf{A}^{-1}\| \cdot \left(\|\Delta\mathbf{A}\| + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{x}\|}\right) \le \\ & \frac{\|\mathbf{A}^{-1}\|}{1 - \|\mathbf{A}^{-1}\| \cdot \|\Delta\mathbf{A}\|} \cdot \left(\|\Delta\mathbf{A}\| + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{x}\|}\right) \le \\ & \frac{\|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|}{1 - \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\| \cdot \dfrac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}} \left(\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{A}\| \cdot \|\mathbf{x}\|}\right).\end{aligned}$$

Celkem tedy

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \frac{\kappa(\mathbf{A})}{1 - \kappa(\mathbf{A})\dfrac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}} \left(\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}\right). \tag{2.23}$$

Číslo podmíněnosti $\kappa(\mathbf{A}) \ge 1$ tedy působí jako zesilovač relativních chyb $\|\Delta\mathbf{A}\|/\|\mathbf{A}\|$ a $\|\Delta\mathbf{b}\|/\|\mathbf{b}\|$. Z (2.23) pro $\Delta\mathbf{A} = \mathbf{O}$ dostaneme

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \kappa(\mathbf{A})\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} = \kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}, \tag{2.24}$$

kde $\mathbf{r} = \mathbf{b} - \mathbf{A}(\tilde{\mathbf{x}})$ je reziduum ($\Delta\mathbf{b} = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b} = -\mathbf{r}$, takže $\|\Delta\mathbf{b}\| = \|\mathbf{r}\|$). Nerovnost (2.24) potvrzuje zkušenost, kterou jsme udělali v příkladu 2.3, totiž že malé reziduum nezaručuje malou relativní chybu. Na pravé straně nerovnosti (2.24) 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.

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 (2.24) odhadneme

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

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

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

Ukažme si ještě, že vztah (2.24) platí jako rovnost:

$$\frac{\|\Delta\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} = \frac{12{,}21}{1} = 111^2\,\frac{0{,}11}{111} = \kappa_\infty(\mathbf{A})\frac{\|\Delta\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty} = 111 \cdot \|\mathbf{r}\|_\infty. \qquad \square$$

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 SLR. Jestliže program zjistí, že číslo podmíněnosti je příliš velké, vydá varování nebo dokonce výpočet přeruší. V MATLABu lze k rychlému posouzení podmíněnosti matice použít funkci `rcond`.

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 $\kappa(\mathbf{A})$. Je-li číslo $\kappa(\mathbf{A})$ malé (velké), je matice $\mathbf{A}$ je dobře (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 stabilní 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 nestabilní 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 stabilní algoritmus.

2.2. Iterační metody

Mnoho praktických problémů vyžaduje řešení rozsáhlých SLR, jejichž matice soustav jsou řídké, tj. mají 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 běžných 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 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.

2.2.1. Klasické iterační metody.

vycházejí z rozkladu matice soustavy $\mathbf{A} = \mathbf{M} - \mathbf{N}$, kde $\mathbf{M}$ je regulární matice. Posloupnost $\mathbf{x}_k$ je pak definována předpisem

$$\mathbf{M}\mathbf{x}_{k+1} = \mathbf{N}\mathbf{x}_k + \mathbf{b}, \tag{2.25}$$

přičemž počáteční aproximace $\mathbf{x}_0$ je daná. Matici $\mathbf{M}$, definující konkrétní metodu, volíme tak, aby řešení SLR (2.25) „nebylo příliš nákladné“ a přitom aby konvergence $\mathbf{x}_k \to \mathbf{x}^*$ byla „rychlá“.

Konvergence. Metodu (2.25) lze ekvivalentně zapsat ve tvaru

$$\mathbf{x}_{k+1} = \mathbf{T}\mathbf{x}_k + \mathbf{c}, \tag{2.26}$$

kde $\mathbf{T} = \mathbf{M}^{-1}\mathbf{N}$, $\mathbf{c} = \mathbf{M}^{-1}\mathbf{b}$. Metoda (2.26) je známa jako *metoda prosté iterace*. Je-li $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ přesné řešení, pak $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$ je chyba v k-tém kroku metody. Konvergence $\mathbf{x}_k \to \mathbf{x}^*$ metody (2.26) nastane, právě když $\mathbf{e}_k \to \mathbf{o}$. Odečteme-li od rovnice (2.26) rovnici $\mathbf{x}^* = \mathbf{T}\mathbf{x}^* + \mathbf{c}$, dostaneme

$$\mathbf{e}_{k+1} = \mathbf{T}\mathbf{e}_k = \mathbf{T}^2\mathbf{e}_{k-1} = \cdots = \mathbf{T}^{k+1}\mathbf{e}_0. \tag{2.27}$$

Proto

$$\mathbf{e}_k \to \mathbf{o} \iff \mathbf{T}^k \to \mathbf{O}. \tag{2.28}$$

Matice $\mathbf{A}$ s vlastností $\mathbf{A}^k \to \mathbf{O}$ pro $k \to \infty$ se nazývá *konvergentní matice*, anglicky *convergent matrix*. Nutnou a postačující podmínku pro to, aby matice $\mathbf{A}$ byla konvergentní, odvodíme pomocí Jordanova rozkladu matice $\mathbf{A}$.

Jordanův rozklad. Ke každé čtvercové matici $\mathbf{A}$ existuje regulární matice $\mathbf{X}$ taková, že

$$\mathbf{X}^{-1}\mathbf{A}\mathbf{X} = \mathbf{J},$$

kde

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_1 & & & \\ & \mathbf{J}_2 & & \\ & & \ddots & \\ & & & \mathbf{J}_t \end{pmatrix},$$

je blokově diagonální matice, známá jako *Jordanův kanonický tvar matice* $\mathbf{A}$, a

$$\mathbf{J}_i = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}, \qquad i = 1, 2, \ldots, t,$$

jsou *Jordanovy bloky*. Mimo hlavní diagonálu a první naddiagonálu jsou prvky matic $\mathbf{J}_i$ nulové. Důkaz existence Jordanova rozkladu viz [53].

Protože horní trojúhelníková matice $\mathbf{A}$ je s maticí $\mathbf{T}$ podobná, diagonální prvky λ_i Jordanových bloků $\mathbf{J}_i$ jsou vlastní čísla matice $\mathbf{A}$, viz kapitola 7.1.5.

Věta (o konvergentní matici). Matice $\mathbf{A}$ je konvergentní, právě když spektrální poloměr $\varrho(\mathbf{A})$ matice $\mathbf{A}$ je menší než 1, tj.

$$\mathbf{A}^k \to \mathbf{O} \ \text{ pro } k \to \infty \quad \Longleftrightarrow \quad \varrho(\mathbf{A}) < 1. \tag{2.29}$$

Důkaz. Snadno ověříme, že

$$\mathbf{A}^k = \mathbf{X}\mathbf{J}^k\mathbf{X}^{-1} = \mathbf{X} \begin{pmatrix} \mathbf{J}_1^k & & & \\ & \mathbf{J}_2^k & & \\ & & \ddots & \\ & & & \mathbf{J}_t^k \end{pmatrix} \mathbf{X}^{-1},$$

takže $\mathbf{A}^k \to \mathbf{O}$, právě když $\mathbf{J}_i^k \to \mathbf{O}$, $i = 1, 2, \ldots, t$. Matici $\mathbf{J}_i$ lze zapsat ve tvaru

$$\mathbf{J}_i = \lambda_i \mathbf{I}_i + \mathbf{N}_i,$$

kde $\mathbf{I}_i$ je jednotková matice řádu n_i a

$$\mathbf{N}_i = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix}.$$

Není těžké ověřit, že pro $1 \le k < n_i$ je $\mathbf{N}_i^k$ horní trojúhelníková s jedničkami v k-té naddiagonále a že pro $k \ge n_i$ je matice $\mathbf{N}_i^k = \mathbf{O}$ nulová. Proto

$$\mathbf{J}_i^k = (\lambda_i \mathbf{I}_i + \mathbf{N}_i)^k = \lambda_i^k \mathbf{I}_i + \sum_{j=1}^{n_i - 1} \binom{k}{j} \lambda_i^{k-j} \mathbf{N}_i^j \quad \text{pro } k \ge n_i.$$

(a) Jestliže $\mathbf{J}_i^k \to \mathbf{O}$, pak nutně $|\lambda_i| < 1$.

(b) Nechť $|\lambda_i| < 1$. Protože řada

$$\sum_{k=1}^{\infty} \binom{k}{j} |\lambda_i^{k-j}| \text{ konverguje, } \binom{k}{j} |\lambda_i^{k-j}| \to 0 \text{ pro } k \to \infty.$$

Odtud $\mathbf{J}_i^k \to \mathbf{O}$ pro $k \to \infty$. □

Dokázali jsme tedy nutnou a postačující podmínku konvergence:

metoda prosté iterace (2.26) *konverguje, právě když* $\varrho(\mathbf{T}) < 1$. (2.30)

Je-li $\mathbf{x}$ vlastní vektor matice $\mathbf{A}$ a λ je odpovídající vlastní číslo, tj. když $\mathbf{Ax} = \lambda\mathbf{x}$, pak

$$|\lambda|\|\mathbf{x}\| = \|\mathbf{Ax}\| \le \|\mathbf{A}\| \cdot \|\mathbf{x}\|, \quad \text{odtud } \varrho(\mathbf{A}) \le \|\mathbf{A}\|. \tag{2.31}$$

Můžeme tedy vyslovit postačující podmínku konvergence:

jestliže $\|\mathbf{T}\| < 1$, *pak metoda prosté iterace* (2.26) *konverguje.* (2.32)

Pro dále uvedené konkrétní iterační metody vyslovíme postačující podmínky konvergence v jiné, snadněji ověřitelné formě.

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 $\mathbf{r}_{k+1} = \mathbf{b} - \mathbf{Ax}_{k+1}$. Postupuje se tak, že uživatel zadá malé kladné číslo ε jako požadovanou přesnost a v každém kroku metody se testuje, zda je splněna některá 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 (2.32) je pak tvaru

$$\mathbf{D}\mathbf{x}_{k+1} = \mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}_k \tag{2.33}$$

a je známa jako *Jacobiova metoda*. Soustava (2.33) s diagonální maticí se řeší snadno. Zapíšeme-li (2.33) po složkách (složky vektoru $\mathbf{x}_k$ jsou značeny $x_i^{(k)}$, podobně pro $\mathbf{x}_{k+1}$), dostaneme

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

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

$$(\mathbf{D} + \mathbf{L})\mathbf{x}_{k+1} = \mathbf{b} - \mathbf{U}\mathbf{x}_k\,. \tag{2.34}$$

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 Jacobiho metody. Dá se ukázat, že když konverguje Jacobiho metoda, tak konverguje také relaxovaná Jacobiho 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

$$(\mathbf{D} + \omega\mathbf{L})\mathbf{x}_{k+1} = [(1-\omega)\mathbf{D} - \omega\mathbf{U}]\,\mathbf{x}_k + \omega\mathbf{b}. \tag{2.35}$$

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

$$\mathbf{D}\mathbf{x}_{k+1} = [(1-\omega)\mathbf{D} - \omega(\mathbf{L} + \mathbf{U})]\,\mathbf{x}_k + \omega\mathbf{b}. \tag{2.36}$$

Symetrická horní relaxace je v literatuře známa jako SSOR metoda (podle anglického „Symmetric Successive Overrelaxation“). Jeden krok metody je definován pomocí dvou

půlkroků, z nichž první je SOR krok a druhý je zpětný SOR krok:

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

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

Maticový zápis SSOR metody je tvaru

$$\begin{aligned}(\mathbf{D}+\omega\mathbf{L})\mathbf{x}_{k+1/2} &= [(1-\omega)\mathbf{D} - \omega\mathbf{U}]\mathbf{x}_k + \omega\mathbf{b},\\ (\mathbf{D}+\omega\mathbf{U})\mathbf{x}_{k+1} &= [(1-\omega)\mathbf{D} - \omega\mathbf{L}]\mathbf{x}_{k+1/2} + \omega\mathbf{b}.\end{aligned} \tag{2.37}$$

Vyloučením $\mathbf{x}_{k+1/2}$ dostaneme předpis

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

kde

$$\mathbf{M} = \frac{\omega}{2-\omega}\left(\frac{1}{\omega}\mathbf{D} + \mathbf{L}\right)\mathbf{D}^{-1}\left(\frac{1}{\omega}\mathbf{D} + \mathbf{U}\right),$$

$$\mathbf{N} = \frac{\omega}{2-\omega}\left(\frac{1}{\omega}\mathbf{D} + \mathbf{L}\right)\mathbf{D}^{-1}\left(\frac{1-\omega}{\omega}\mathbf{D} - \mathbf{L}\right)\left(\frac{1}{\omega}\mathbf{D} + \mathbf{L}\right)^{-1}\left(\frac{1-\omega}{\omega}\mathbf{D} - \mathbf{U}\right).$$

Postačující podmínky konvergence jsou stejné jako pro SOR metodu: $\mathbf{A}$ ryze diagonálně dominantní a $0 < \omega \leq 1$ nebo $\mathbf{A}$ pozitivně definitní a $0 < \omega < 2$. Je-li $\mathbf{A}$ symetrická, pak $\mathbf{U} = \mathbf{L}^T$, takže

$$\mathbf{M} = \frac{1}{\omega(2-\omega)}\left(\mathbf{D} + \omega\mathbf{L}\right)\mathbf{D}^{-1}\left(\mathbf{D} + \omega\mathbf{L}^T\right). \tag{2.38}$$

Tato matice se někdy používá jako předpodmiňovací v metodě sdružených gradientů. Je-li $\omega = 1$, dostaneme symetrickou Gaussovu-Seidelovu metodu, stručně SGS metodu.

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] Matice $\mathbf{K}$ je blokově třídiagonální,

[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 SLR s maticí soustavy $\mathbf{K}$ uvažovanou v tomto příkladu 2.5.

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

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

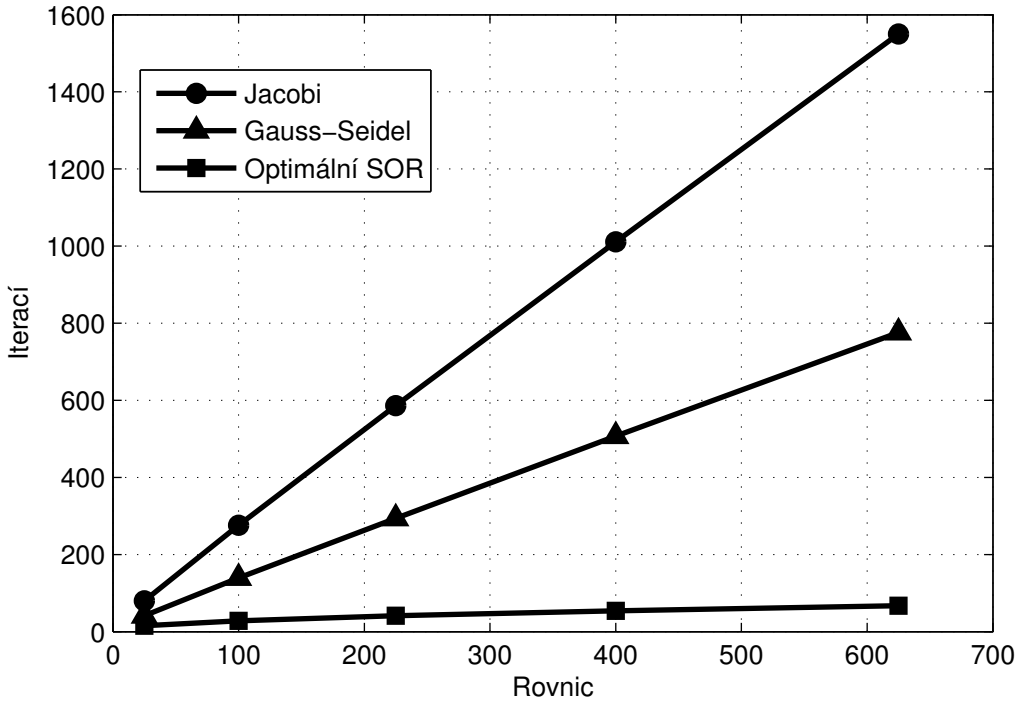
Matice $\mathbf{I}$ je jednotková matice, $\mathbf{O}$ je čtvercová matice s nulovými prvky a $\mathbf{B}$ je třídiagonální matice

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

Na matici $\mathbf{K}$ se často testuje účinnost numerických metod pro řešení SLR 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.

2.2.2. Další iterační metody.

Klasické iterační metody se současnosti 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. V této kapitole uvedeme dvě typické, v aplikacích standardně používané, iterační metody: *zobecněnou metodu minimálních reziduí* pro SLR s libovolnou regulární maticí soustavy a *metodu sdružených gradientů* pro SLR s pozitivně definitní maticí soustavy. Řadu dalších metod lze najít třeba v [33].



Obr. 2.3: Srovnání klasických iteračních metod

2.2.2.1. Zobecněná metoda minimálních reziduí

Máme najít řešení $\mathbf{x}^*$ soustavy lineárních rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$, kde $\mathbf{A}$ je obecná regulární matice. Začneme tím, že si vysvětlíme, jak pracuje

Metoda minimálních reziduí. Přibližné řešení $\mathbf{x}_{k+1}$ hledáme ve tvaru

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k \,, \tag{2.39}$$

kde $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k \neq \mathbf{o}$ je reziduum (je-li $\mathbf{r}_k = \mathbf{o}$, pak $\mathbf{x}_k = \mathbf{x}^*$) a α_k je koeficient určený tak, aby délka $\|\mathbf{r}_{k+1}\|_2$ vektoru nového rezidua $\mathbf{r}_{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}$ byla minimální.

V dalším budeme používat označení $(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}$ pro skalární součin vektorů $\mathbf{u}$ a $\mathbf{v}$, takže $\|\mathbf{u}\|_2 = (\mathbf{u}, \mathbf{u})^{1/2}$. Dolní index $_2$ v $\|\mathbf{u}\|_2$ budeme vynechávat, tj. $\|\mathbf{u}\| \equiv \|\mathbf{u}\|_2$.

Koeficient α_k dostaneme minimalizací funkce

$$\varphi(\alpha) = \|\mathbf{b} - \mathbf{A}(\mathbf{x}_k + \alpha\mathbf{r}_k)\|^2 = \|\mathbf{r}_k - \alpha\mathbf{A}\mathbf{r}_k\|^2 = \|\mathbf{r}_k\|^2 - 2\alpha(\mathbf{r}_k, \mathbf{A}\mathbf{r}_k) + \alpha^2\|\mathbf{A}\mathbf{r}_k\|^2.$$

Protože

$$\varphi'(\alpha) = -2(\mathbf{r}_k, \mathbf{A}\mathbf{r}_k) + 2\alpha\|\mathbf{A}\mathbf{r}_k\|^2 \,, \qquad \varphi''(\alpha) = 2\|\mathbf{A}\mathbf{r}_k\|^2 \,,$$

$\varphi(\alpha)$ nabývá svého minima pro

$$\alpha_k = \frac{(\mathbf{r}_k, \mathbf{A}\mathbf{r}_k)}{(\mathbf{A}\mathbf{r}_k, \mathbf{A}\mathbf{r}_k)} \,. \tag{2.40}$$

Metoda minimálních rezidií se někdy označuje zkratkou MR (podle anglického „minimal residual“). Je to jednokroková metoda: k určení $\mathbf{x}_{k+1}$ stačí znát jen $\mathbf{x}_k$ a $\mathbf{r}_k$ z bezprostředně

předchozího kroku. Dá se ukázat, že metoda MR konverguje, tj. $\mathbf{x}_k \to \mathbf{x}^*$, například tehdy, když $\mathbf{A}$ je pozitivně definitní. Metodu MR proto zobecníme tak, aby konvergence nastala pro každou regulární matici.

V dalším budeme pracovat s pojmem Krylovova podprostoru. Nejdříve připomeňme, že $\mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m)$ je vektorový prostor generovaný vektory $\mathbf{v}_1$, $\mathbf{v}_2, \ldots,$ $\mathbf{v}_m$, tj.

$$\mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m) = \{\mathbf{v} \,|\, \mathbf{v} = \sum_{i=1}^{m} \alpha_i \mathbf{v}_i, \alpha_i \in \mathbb{R}, i = 1, 2, \ldots, m\}.$$

Prostor

$$\mathcal{K}_m(\mathbf{C}, \mathbf{v}) = \mathrm{span}(\mathbf{v}, \mathbf{C}\mathbf{v}, \ldots, \mathbf{C}^{m-1}\mathbf{v})$$

je tzv. *Krylovův podprostor* (pro dané přirozené číslo m, matici $\mathbf{C}$ a vektor $\mathbf{v}$). $\mathcal{K}_m(\mathbf{C}, \mathbf{v})$ je podprostor $\mathbb{R}^n$, kde n je řád matice $\mathbf{C}$, dimenze $\mathcal{K}_m(\mathbf{C}, \mathbf{v})$ je nejvýše rovna menšímu z čísel m a n.

Zobecněná metoda minimálních reziduí (anglicky „generalized minimal residual“, zkratka GMRES). Přibližné řešení $\mathbf{x}_m$ soustavy $\mathbf{A}\mathbf{x} = \mathbf{b}$ hledejme ve tvaru

$$\mathbf{x}_m = \mathbf{x}_0 + \boldsymbol{\delta}_m \,,$$

kde $\boldsymbol{\delta}_m \in \mathcal{K}_m \equiv \mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$ vybereme tak, aby norma $\|\mathbf{r}_m\|$ rezidua $\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ byla minimální, tj. aby

$$\|\mathbf{r}_m\|^2 = \|\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}_m\|^2 \leq \|\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}\|^2 \quad \forall \boldsymbol{\delta} \in \mathcal{K}_m \,. \tag{2.41}$$

Předpokládejme, že $\mathcal{K}_m$ je dimenze m. Protože přirozená báze $\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\}$ prostoru $\mathcal{K}_m$ je špatně podmíněná, sestrojíme v něm ortonormální bázi $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ a $\mathbf{x}_m$ hledáme ve tvaru

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m \mathbf{y}_m \,, \quad \text{kde } \mathbf{V}_m = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m) \text{ a } \mathbf{y}_m \in \mathbb{R}^m \,, \tag{2.42}$$

přičemž $\mathbf{y}_m$ určíme v souladu s (2.41) tak, aby

$$\|\mathbf{r}_m\|^2 = \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m\|^2 \leq \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}\|^2 \quad \forall \mathbf{y} \in \mathbb{R}^m \,. \tag{2.43}$$

Navržená metoda je m-kroková, neboť k sestrojení $\mathbf{x}_m$ potřebujeme vektory $\{\mathbf{v}_k\}_{k=1}^m$.

Sestrojení ortonormální báze v $\mathcal{K}_m(\mathbf{A}, \mathbf{v})$. Nejdříve uvedeme základní verzi Gramovy-Schmidtovy ortonormalizace jako

Algoritmus AGS (Arnoldi-Gram-Schmidt)

1. $\mathbf{v}_1 := \mathbf{v}/\|\mathbf{v}\|$
2. **for** $k := 1, 2, \ldots, m$ **do**
3. **for** $i := 1, 2, \ldots, k$ **do** $h_{ik} := (\mathbf{A}\mathbf{v}_k, \mathbf{v}_i)$ **end**
4. $\mathbf{w}_k := \mathbf{A}\mathbf{v}_k - \sum_{i=1}^{k} h_{ik}\mathbf{v}_i$
5. $h_{k+1,k} := \|\mathbf{w}_k\|$

6. **if** $h_{k+1,k} = 0$ **then** stop
7. $\mathbf{v}_{k+1} := \mathbf{w}_k / h_{k+1,k}$
8. **end**

Tvrzení 1. Předpokládejme, že algoritmus AGS neskončí dříve než v m-tém kroku. Pak vektory $\mathbf{v}_1$, $\mathbf{v}_2, \ldots, \mathbf{v}_m$ tvoří ortonormální bázi Krylovova podprostoru $\mathcal{K}_m \equiv \mathcal{K}_m(\mathbf{A}, \mathbf{v})$.

Důkaz. Vektory $\mathbf{v}_k$, $k = 1, 2, \ldots, m$, jsou ortonormální (podle řádků 3 a 4 je $(\mathbf{w}_k, \mathbf{v}_j) = 0$, $j = 1, 2, \ldots, k$, a podle řádku 7 je $\|\mathbf{v}_{k+1}\| = \mathbf{w}_k / h_{k+1,k} = 1$, $k = 2, 3, \ldots, m$). Zbývá dokázat, že $\mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k) = \mathcal{K}_m$.

a) Nejdříve dokážeme, že $\mathbf{v}_j = q_{j-1}(\mathbf{A})\mathbf{v}_1$, kde $q_{j-1}(t) = \sum_{i=0}^{j-1} a_i^{(j-1)} t^{j-1-i}$ je polynom stupně $j-1$ s koeficientem $a_0^{(j-1)} \neq 0$. Důkaz provedeme indukcí. Pro $j = 1$ je $\mathbf{v}_1 = q_0(\mathbf{A})\mathbf{v}_1$ pro $q_0(t) = a_0^{(0)} = 1$. Předpokládejme, že tvrzení platí pro $j \leq k$. Pak

$$h_{k+1,k}\mathbf{v}_{k+1} = \mathbf{w}_k = \mathbf{A}\mathbf{v}_k - \sum_{i=1}^{k} h_{ik}\mathbf{v}_i = \mathbf{A}q_{k-1}(\mathbf{A})\mathbf{v}_1 - \sum_{i=1}^{k} h_{ik} q_{i-1}(\mathbf{A})\mathbf{v}_1 \,,$$

takže $\mathbf{v}_{k+1} = q_k(\mathbf{A})\mathbf{v}_1$, kde $q_k(t)$ je polynom stupně k a $a_0^{(k)} = a_0^{(k-1)} / h_{k+1,k} \neq 0$. Dokázali jsme tedy, že $\mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m) \subseteqq \mathcal{K}_m$.

b) Dokážeme, že $\mathbf{A}^{j-1}\mathbf{v}_1 \in \mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_j)$. Pro $j = 1$ tvrzení platí. Předpokládejme, že tvrzení platí pro $j \leq k$. Pak z rovnice

$$h_{k+1,k}\mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_k - \sum_{i=1}^{k} h_{ik}\mathbf{v}_i = a_0^{(k-1)}\mathbf{A}^k\mathbf{v}_1 + \sum_{i=1}^{k-1} a_i^{(k-1)}\mathbf{A}^{k-i}\mathbf{v}_1 - \sum_{i=1}^{k} h_{ik}\mathbf{v}_i,$$

a protože $a_0^{(k-1)} \neq 0$, tvrzení platí i pro $j = k+1$. Proto $\mathcal{K}_m \subseteqq \mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m)$.

Dokázali jsme tedy, že $\mathcal{K}_m = \mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m)$ a důkaz je hotov. □

Tvrzení 2. Označme $\mathbf{V}_m$ matici typu (n, m), jejíž sloupce jsou vektory $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$, a dále nechť $\bar{\mathbf{H}}_m$ je horní Hessenbergova matice typu $(m+1, m)$, jejíž nenulové prvky h_{ik} jsou definovány algoritmem AGS, tj.

$$\bar{\mathbf{H}}_m = \{\bar{h}_{ik}\} = \begin{cases} h_{ik}\,, & i = 1, 2, \ldots, k+1, \\ 0\,, & i = k+2, k+3, \ldots, m+1, \end{cases} \qquad k = 1, 2, \ldots, m\,.$$

Dále nechť $\mathbf{H}_m$ je matice, kterou dostaneme z $\bar{\mathbf{H}}_m$ vypuštěním jejího posledního řádku. Pak

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_m\mathbf{H}_m + \mathbf{w}_m\mathbf{e}_m^T = \tag{2.44}$$

$$= \mathbf{V}_{m+1}\bar{\mathbf{H}}_m\,, \tag{2.45}$$

$$\mathbf{V}_m^T\mathbf{A}\mathbf{V}_m = \mathbf{H}_m\,. \tag{2.46}$$

Přitom $\mathbf{e}_m$ je $m-$tý sloupec jednotkové matice řádu m.

Důkaz. (2.45) a (2.44) plyne z řádků 4, 5 a 7 algoritmu AGS, podle kterých

$$\mathbf{A}\mathbf{v}_k = \sum_{i=1}^{k+1} h_{ik}\mathbf{v}_i\,, \quad k = 1, 2, \ldots, m\,, \qquad \mathbf{A}\mathbf{v}_m = \sum_{i=1}^{m} h_{im}\mathbf{v}_i + \mathbf{w}_m\,. \tag{2.47}$$

(2.46) dostaneme z (2.44), neboť $\mathbf{V}_m^T\mathbf{w}_m = h_{m+1,m}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m)^T\mathbf{v}_{m+1} = \mathbf{o}$. □

Praktickou implementaci algoritmu AGM představuje

Algoritmus AGSM (Arnoldi-Gram-Schmidt modified)

1. $\mathbf{v}_1 := \mathbf{v}/\|\mathbf{v}\|$
2. **for** $k := 1, 2, \ldots, m$ **do**
3. $\quad\mathbf{w}_k := \mathbf{A}\mathbf{v}_k$
4. $\quad$**for** $i := 1, 2, \ldots, k$ **do**
5. $\quad\quad h_{ik} := (\mathbf{w}_k, \mathbf{v}_i)$
6. $\quad\quad\mathbf{w}_k := \mathbf{w}_k - h_{ik}\mathbf{v}_i$
7. $\quad$**end**
8. $\quad h_{k+1,k} := \|\mathbf{w}_k\|$
9. $\quad$**if** $h_{k+1,k} = 0$ **then** stop
10. $\quad\mathbf{v}_{k+1} := \mathbf{w}_k/h_{k+1,k}$
11. **end**

Protože $(\mathbf{A}\mathbf{v}_k - \sum_{j=1}^{i-1} h_{jk}\mathbf{v}_j, \mathbf{v}_i) = (\mathbf{A}\mathbf{v}_k, \mathbf{v}_i) = h_{ik}$, jsou algoritmy AGS a AGSM ekvivalentní. Při reálném výpočtu je však algoritmus AGSM stabilnější.

Algoritmus metody GMRES. Ze vztahu (2.43) plyne, že vektor $\mathbf{y}_m$ minimalizuje funkci

$$\varphi(\mathbf{y}) = \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}\|\,.$$

To ale znamená, že $\mathbf{y}_m$ je řešením přeurčené soustavy n lineárních rovnic o m neznámých

$$\mathbf{A}\mathbf{V}_m\mathbf{y} \cong \mathbf{r}_0$$

ve smyslu metody nejmenších čtverců, viz kapitola 3. Protože sloupce matice $\mathbf{V}_m$ jsou lineárně nezávislé, jsou lineárně nezávislé také sloupce matice $\mathbf{A}\mathbf{V}_m$, a proto je $\mathbf{y}_m$ určeno jednoznačně, viz kapitola 3.

V dalším popíšeme efektivní výpočet vektoru $\mathbf{y}_m$. Využijeme následující vlastnost: jestliže matice $\mathbf{Q}$ splňuje podmínku $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$, pak $\|\mathbf{v}\| = \|\mathbf{Q}\mathbf{v}\|$. Skutečně,

$$\|\mathbf{Q}\mathbf{v}\|^2 = (\mathbf{Q}\mathbf{v})^T\mathbf{Q}\mathbf{v} = \mathbf{v}^T\mathbf{Q}^T\mathbf{Q}\mathbf{v} = \mathbf{v}^T\mathbf{v} = \|\mathbf{v}\|^2\,.$$

Označíme-li $\|\mathbf{r}_0\| = \beta$, pak $\mathbf{v}_1 = \beta^{-1}\mathbf{r}_0$, a tedy

$$\mathbf{r}_0 = \beta\mathbf{v}_1 = \mathbf{V}_{m+1}\beta\bar{\mathbf{e}}_1\,,$$

kde $\bar{\mathbf{e}}_1$ je první sloupec jednotkové matice řádu $m+1$. Odtud a pomocí (2.45) dostaneme

$$\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y} = \mathbf{V}_{m+1}\beta\bar{\mathbf{e}}_1 - \mathbf{V}_{m+1}\bar{\mathbf{H}}_m\mathbf{y} = \mathbf{V}_{m+1}(\beta\bar{\mathbf{e}}_1 - \bar{\mathbf{H}}_m\mathbf{y})\,,$$

a protože vzhledem k ortonormalitě báze $\{\mathbf{v}_i\}_{i=1}^{m+1}$ je $\mathbf{V}_{m+1}^T\mathbf{V}_{m+1} = \mathbf{I}$, obdržíme

$$\varphi(\mathbf{y}) = \|\mathbf{V}_{m+1}(\beta\bar{\mathbf{e}}_1 - \bar{\mathbf{H}}_m\mathbf{y})\| = \|\bar{\mathbf{H}}_m\mathbf{y} - \beta\bar{\mathbf{e}}_1\|\,.$$

Proto $\mathbf{y}_m$ získáme řešením přeurčené soustavy $m+1$ rovnic o m neznámých

$$\bar{\mathbf{H}}_m \mathbf{y} \cong \beta \bar{\mathbf{e}}_1 \tag{2.48}$$

metodou nejmenších čtverců. Standardní postup je založen na tzv. QR rozkladu matice

$$\bar{\mathbf{H}}_m = \mathbf{Q}_{m+1} \bar{\mathbf{R}}_m \,,$$

kde $\mathbf{Q}_{m+1}$ je ortonormální matice řádu $m+1$ a $\bar{\mathbf{R}}_m$ je horní trojúhelníková matice,

$$\bar{\mathbf{R}}_m = \{\bar{r}_{ik}\}\,, \qquad \bar{r}_{ik} = \begin{cases} r_{ik}\,, & i = 1, 2, \ldots, k, \\ 0\,, & i = k+1, k+2, \ldots, m+1, \end{cases} \qquad k = 1, 2, \ldots, m\,.$$

Metodám QR rozkladu je věnována kapitola 3.4, zde budeme předpokládat, že matice $\mathbf{Q}_{m+1}$ a $\bar{\mathbf{R}}_m$ umíme určit. Pak

$$\varphi(\mathbf{y}) = \|\mathbf{Q}_{m+1}\bar{\mathbf{R}}_m \mathbf{y} - \beta\bar{\mathbf{e}}_1\| = \|\mathbf{Q}_{m+1}(\bar{\mathbf{R}}_m \mathbf{y} - \beta \mathbf{Q}_{m+1}^T \bar{\mathbf{e}}_1)\| = \|\bar{\mathbf{R}}_m \mathbf{y} - \bar{\mathbf{d}}_m\|\,,$$

kde $\bar{\mathbf{d}}_m = \beta \mathbf{Q}_{m+1}^T \bar{\mathbf{e}}_1$. Nechť

$$\bar{\mathbf{R}}_m = \begin{pmatrix} \mathbf{R}_m \\ \mathbf{O} \end{pmatrix}, \qquad \bar{\mathbf{d}}_m = \begin{pmatrix} \mathbf{d}_m \\ \gamma_m \end{pmatrix},$$

kde $\mathbf{R}_m$ je matice řádu m, $\mathbf{O}$ je nulová matice typu $(1, m)$, $\mathbf{d}_m$ vektor typu $(m, 1)$ a γ_m je skalár. Protože

$$\varphi(\mathbf{y}) = \left\| \begin{pmatrix} \mathbf{R}_m \\ \mathbf{O} \end{pmatrix} \mathbf{y} - \begin{pmatrix} \mathbf{d}_m \\ \gamma_m \end{pmatrix} \right\| = \left\| \begin{pmatrix} \mathbf{R}_m \mathbf{y} - \mathbf{d}_m \\ -\gamma_m \end{pmatrix} \right\|,$$

$\varphi(\mathbf{y})$ nabude své minimální hodnoty $\varphi(\mathbf{y}_m)$, právě když

$$\mathbf{R}_m \mathbf{y}_m = \mathbf{d}_m. \tag{2.49}$$

Pak $\varphi(\mathbf{y}_m) = |\gamma_m|$. Protože podle (2.43) je $\varphi(\mathbf{y}_m) = \|\mathbf{r}_m\|$, dostali jsme

$$\|\mathbf{r}_m\| = |\gamma_m|\,. \tag{2.50}$$

Je-li $\gamma_m = 0$, pak $\mathbf{x}_m = \mathbf{x}^*$. Nyní již můžeme uvést

Algoritmus GMRES(m).

1. zvolíme $\mathbf{x}_0$, ε, m, $nrmax$
2. **for** $nr := 1, 2, \ldots, nrmax$ **do**
3. $\quad \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta := \|\mathbf{r}_0\|$, $\mathbf{v}_1 := \mathbf{r}_0/\beta$
4. $\quad$ **for** $k := 1, 2, \ldots, m$ **do**
5. $\quad\quad \mathbf{w}_k := \mathbf{A}\mathbf{v}_k$
6. $\quad\quad$ **for** $i := 1, 2, \ldots, k$ **do**
7. $\quad\quad\quad h_{ik} := (\mathbf{w}_k, \mathbf{v}_i)$
8. $\quad\quad\quad \mathbf{w}_k := \mathbf{w}_k - h_{ik}\mathbf{v}_i$

9. **end**
10. $h_{k+1,k} := \|\mathbf{w}_k\|$, sestavíme $\bar{\mathbf{H}}_k$, vypočteme $\mathbf{R}_k$, $\mathbf{d}_k$, γ_k
11. **if** $|\gamma_k| < \varepsilon$ **or** $k = m$
12. $\mathbf{y}_k = \mathbf{R}_k^{-1}\mathbf{d}_k$, $\mathbf{x} := \mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k$, **if** $|\gamma_k| < \varepsilon$, stop, **end**
13. **else**
14. $\mathbf{v}_{k+1} := \mathbf{w}_k / h_{k+1,k}$, sestavíme $\mathbf{V}_{k+1}$
15. **end**
16. **end**
17. $\mathbf{x}_0 := \mathbf{x}$
18. **end**
19. print(`'GMRES(m) nekonverguje'`)

K algoritmu GMRES(m) připojíme několik poznámek.

1. Všimněte si, že algoritmus GMRES(1) je ekvivalentní s algoritmem MR. GMRES(m) je zřejmě m-kroková metoda.

2. Označme jako GMRES algoritmus GMRES(n). GMRES lze považovat za metodu minimalizační, přibližné řešení $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k$ je definováno takto:

   určit $\mathbf{x}_k \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ splňující $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\| \le \|\mathbf{b} - \mathbf{A}\mathbf{x}\|$ $\forall \mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.

   Pokud bychom počítali přesně, pak v řádku 17 dostaneme přesné řešení $\mathbf{x} = \mathbf{x}^*$ pro nějaké $k \le n$.

   Důkaz. Je-li $h_{k+1,k} = 0$ pro $k \le n$, pak z (2.48) plyne $\mathbf{H}_k\mathbf{y}_k = \beta\mathbf{e}_1$, kde $\mathbf{e}_1$ je první sloupec jednotkové matice řádu k, a dále z (2.44) plyne $\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k$, neboť $\mathbf{w}_k = \mathbf{o}$. Proto

   $$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k = \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k) = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k\mathbf{y}_k =$$

   $$\mathbf{r}_0 - \mathbf{V}_k\mathbf{H}_k\mathbf{y}_k = \mathbf{r}_0 - \mathbf{V}_k\beta\mathbf{e}_1 = \mathbf{r}_0 - \beta\mathbf{v}_1 = \mathbf{o}\,,$$

   takže $\mathbf{x}_k = \mathbf{A}^{-1}\mathbf{b} = \mathbf{x}^*$. Pokud $h_{k+1,k} \neq 0$, $k = 1, 2, \ldots, n-1$, pak $h_{n+1,n} = 0$. □

3. O metodě GMRES se mluví také jako o metodě kosoúhlé projekce (anglicky „oblique projection“), čímž se míní, že $\mathbf{x}_m$ určujeme tak, aby

   $$\mathbf{x}_m - \mathbf{x}_0 \in \mathcal{K}_m \qquad \text{a} \qquad \mathbf{r}_m \perp \mathcal{L}_m := \mathbf{A}\mathcal{K}_m\,.$$

   Snadno ověříme, že pak $\mathbf{r}_m$ minimalizuje $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$ v $\mathbf{x}_0 + \mathcal{K}_m$.

   Důkaz. Označíme-li $\psi(\mathbf{y}) = \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}\|^2$, pak uvedené tvrzení je důsledkem vztahů

   $$\mathbf{o} = \nabla\psi(\mathbf{y}_m) = -2(\mathbf{A}\mathbf{V}_m)^T\mathbf{r}_m\,, \qquad \nabla^2\psi(\mathbf{y}_m) = 2(\mathbf{A}\mathbf{V}_m)^T\mathbf{A}\mathbf{V}_m\,.$$

4. Pro velké m je třeba v paměti počítače uchovávat matice $\mathbf{V}_m$ a $\bar{\mathbf{H}}_m$, které jsou pro velké n značně rozsáhlé. Proto se obvykle volí m v řádu desítek. V takovém případě se ale může stát, že konvergence metody GMRES(m) je velmi pomalá. Proto bývá omezen maximální počet tzv. restartů, viz parametr *nrmax* na řádku 1.

5. $\mathbf{R}_k$, $\mathbf{d}_k$ a γ_k počítáme Givensovým QR algoritmem, viz kapitola 3.4.2, v němž využijeme to, že matice $\bar{\mathbf{H}}_k$ má pod hlavní diagonálou nenulové jen prvky $h_{i+1,i}$, $i = 1, 2, \ldots, k$.

6. Rychlost konvergence můžeme podstatně zvýšit, když místo rovnice $\mathbf{Ax} = \mathbf{b}$ řešíme rovnici $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$, kde $\mathbf{M}$ je tzv. předpodmiňovací matice. Za $\mathbf{M}$ lze zvolit například neúplný LU rozklad matice $\mathbf{A}$, viz [46]. Inverzní matici $\mathbf{M}^{-1}$ ve skutečnosti nepočítáme. Algoritmus GMRES(m) se změní pouze v řádcích 3 a 5 takto:
   3. vypočteme $\mathbf{z}_0$ jako řešení $\mathbf{Mz}_0 = \mathbf{b} - \mathbf{Ax}_0$, $\beta := \|\mathbf{z}_0\|$, $\mathbf{v}_1 := \mathbf{z}_0/\beta$
   5. vypočteme $\mathbf{w}_k$ jako řešení $\mathbf{Mw}_k = \mathbf{Av}_k$

7. V MATLABu je metoda GMRES(m) implementována jako funkce `gmres` a neúplný LU rozklad jako funkce `ilu`.

2.2.2.2. Metoda sdružených gradientů

je jednou ze základních metod pro řešení soustav lineárních rovnic $\mathbf{Ax} = \mathbf{b}$ s pozitivně definitní maticí soustavy. Předpokládejme tedy, že matice $\mathbf{A}$ je pozitivně definitní. To nám umožňuje definovat skalární součin $(\mathbf{u}, \mathbf{v})_\mathbf{A} = \mathbf{u}^T\mathbf{Av}$ a tzv. *energetickou normu*

$$\|\mathbf{u}\|_\mathbf{A} = (\mathbf{u}, \mathbf{u})_\mathbf{A}^{1/2} = \sqrt{\mathbf{u}^T\mathbf{Au}}.$$

Přesné řešení soustavy lineárních rovnic $\mathbf{Ax} = \mathbf{b}$ budeme značit $\mathbf{x}^*$.

K lepšímu pochopení principu, z něhož metoda sdružených gradientů vychází, poslouží jednoduchá

Metoda největšího spádu. Přibližné řešení $\mathbf{x}_{k+1}$ hledáme ve tvaru

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{r}_k\,, \tag{2.39}$$

kde $\mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k \neq \mathbf{o}$ je reziduum a α_k je koeficient určený tak, aby energetická norma $\|\mathbf{e}_{k+1}\|_\mathbf{A}$ chyby $\mathbf{e}_{k+1} = \mathbf{x}_{k+1} - \mathbf{x}^*$ byla minimální. Koeficient α_k dostaneme minimalizací funkce

$$\varphi(\alpha) = \tfrac{1}{2}\|\mathbf{x}_k + \alpha\mathbf{r}_k - \mathbf{x}^*\|_\mathbf{A}^2 = \tfrac{1}{2}(\mathbf{x}_k + \alpha\mathbf{r}_k - \mathbf{x}^*)^T\mathbf{A}(\mathbf{x}_k + \alpha\mathbf{r}_k - \mathbf{x}^*)\,.$$

S přihlédnutím k symetrii matice $\mathbf{A}$ obdržíme

$$\varphi'(\alpha) = \mathbf{r}_k^T\mathbf{A}(\mathbf{x}_k + \alpha\mathbf{r}_k - \mathbf{x}^*) = \mathbf{r}_k^T(\mathbf{Ax}_k + \alpha\mathbf{Ar}_k - \mathbf{b}) = \mathbf{r}_k^T(\alpha\mathbf{Ar}_k - \mathbf{r}_k)\,,$$
$$\varphi''(\alpha) = \mathbf{r}_k^T\mathbf{Ar}_k\,.$$

Z nutné podmínky $\varphi'(\alpha_k) = 0$ pro extrém funkce $\varphi(\alpha)$ určíme

$$\alpha_k = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{r}_k, \mathbf{Ar}_k)}\,,$$

a protože matice $\mathbf{A}$ je pozitivně definitní, je α_k hledané minimum funkce φ.

Rovnici $\varphi'(\alpha_k) = 0$, ze které jsme určili α_k (a tedy také $\mathbf{x}_{k+1}$), lze ekvivalentně zapsat ve tvaru

$$\mathbf{r}_k^T(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}) = -\mathbf{r}_k^T\mathbf{r}_{k+1} = 0\,,$$

tj. $\mathbf{x}_{k+1}$ jsme určili tak, aby reziduum $\mathbf{r}_{k+1}$ bylo ortogonální k předchozímu reziduu $\mathbf{r}_k$.

Všimněte si, že přibližné řešení $\mathbf{x}_{k+1}$ je minimem funkce $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{A}}^2$ na polopřímce $\mathbf{x}_k + \alpha\mathbf{r}_k$, $\alpha \geq 0$. Protože $\mathbf{r}_k = -\nabla\psi(\mathbf{x}_k)$, minimum $\mathbf{x}_{k+1}$ hledáme ve směru největšího spádu funkce ψ, což vysvětluje název metody.

Zobecněním metody největšího spádu je

Metoda ortogonální projekce, stručně FOM podle anglického „full ortogonalization method“, viz např. [46], která počítá $\mathbf{x}_m$ tak, aby

$$\mathbf{x}_m - \mathbf{x}_0 \in \mathcal{K}_m \qquad \text{a} \qquad \mathbf{r}_m \perp \mathcal{K}_m\,.$$

Přibližné řešení hledáme stejně jako v metodě GMRES ve tvaru

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m\,, \tag{2.42}$$

zvolíme $\mathbf{v}_1 = \beta^{-1}\mathbf{r}_0$, kde $\beta = \|\mathbf{r}_0\|$, $\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_m$ vypočteme algoritmem AGSM a $\mathbf{y}_m$ určíme tak, aby

$$\mathbf{o} = \mathbf{V}_m^T\mathbf{r}_m = \mathbf{V}_m^T(\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m)) = \mathbf{V}_m^T\mathbf{r}_0 - \mathbf{V}_m^T\mathbf{A}\mathbf{V}_m\mathbf{y}_m = \beta\mathbf{e}_1 - \mathbf{H}_m\mathbf{y}_m\,.$$

Využili jsme toho, že $\mathbf{V}_m^T\mathbf{r}_0 = \mathbf{V}_m^T\beta\mathbf{v}_1 = \beta\mathbf{e}_1$, kde $\mathbf{e}_1$ je první sloupec jednotkové matice řádu m, a dále také vztahu (2.46). Vektor $\mathbf{y}_m$ je proto řešením soustavy m lineárních rovnic

$$\mathbf{H}_m\mathbf{y}_m = \beta\mathbf{e}_1\,. \tag{2.51}$$

Je-li $h_{m,m+1} = 0$, pak $\mathbf{x}_m = \mathbf{x}^*$. Skutečně, pomocí (2.42), (2.44) a (2.51) dostaneme

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_m\| = \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m\| = \|\mathbf{r}_0 - \mathbf{V}_m\mathbf{H}_m\mathbf{y}_m\| = \|\mathbf{r}_0 - \mathbf{V}_m\beta\mathbf{e}_1\| = 0.$$

Pro pozitivně definitní matici $\mathbf{A}$ je rovnice $\mathbf{V}_m^T\mathbf{r}_m = \mathbf{o}$ ekvivalentním vyjádřením toho, že $\mathbf{x}_m$ minimalizuje funkci $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{A}}^2$ na množině $\mathbf{x}_0 + \mathcal{K}_m$ nebo-li že $\mathbf{y}_m$ minimalizuje funkci $\varphi(\mathbf{y}) = \frac{1}{2}\|\mathbf{x}_0 + \mathbf{V}_m\mathbf{y} - \mathbf{x}^*\|_{\mathbf{A}}$ v $\mathbb{R}^m$. Skutečně, $\nabla\varphi(\mathbf{y}) = \mathbf{V}_m^T(\mathbf{A}\mathbf{x} - \mathbf{b})$, kde $\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}$, $\nabla^2\varphi(\mathbf{y}) = \mathbf{V}_m^T\mathbf{A}\mathbf{V}_m$, takže pro minimum $\mathbf{y}_m$ funkce $\varphi(\mathbf{y})$ platí $\mathbf{o} = \nabla\varphi(\mathbf{y}_m) = -\mathbf{V}_m^T\mathbf{r}_m$.

FOM lze použít i v případě, že matice $\mathbf{A}$ není pozitivně definitní: $\mathbf{y}_m$ zase určíme z (2.51), nepůjde však už o minimum funkce $\psi(\mathbf{x})$.

Pro pozitivně definitní matici $\mathbf{A}$ lze m-krokovou FOM metodu přeformulovat na jednokrokovou metodu známou jako *metoda sdružených gradientů*. V následujícím textu ukážeme, jak to lze provést. Předpokládejme tedy, že matice $\mathbf{A}$ je pozitivně definitní. Podle (2.46) je $\mathbf{H}_m$ symetrická, a protože je horní Hessenbergova, je třídiagonální,

$$\mathbf{H}_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ldots & \ldots & \ldots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}, \tag{2.52}$$

tj. $h_{kk} = \alpha_k$, $h_{k,k+1} = h_{k+1,k} = \beta_k$. Pomocí LU rozkladu dostaneme

$$\mathbf{H}_m = \mathbf{L}_m\mathbf{U}_m = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \dots & \dots & \dots & \\ & & \lambda_{m-1} & 1 & \\ & & & \lambda_m & 1 \end{pmatrix} \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & \dots & \dots & \dots & \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{pmatrix}, \tag{2.53}$$

kde

$$\begin{aligned} &\eta_1 = \alpha_1\,, \\ &\lambda_k = \beta_k/\eta_{k-1}\,, \qquad \eta_k = \alpha_k - \lambda_k\beta_k\,, \quad k = 2, 3, \dots, m\,. \end{aligned} \tag{2.54}$$

Z (2.42) a (2.51) – (2.53) plyne

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{H}_m^{-1}(\beta\mathbf{e}_1) = \mathbf{x}_0 + \mathbf{V}_m\mathbf{U}_m^{-1}\mathbf{L}_m^{-1}(\beta\mathbf{e}_1)\,.$$

Označíme-li

$$\mathbf{P}_m = \mathbf{V}_m\mathbf{U}_m^{-1} \quad \text{a} \quad \mathbf{z}_m = \mathbf{L}_m^{-1}(\beta\mathbf{e}_1)\,, \tag{2.55}$$

pak

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{P}_m\mathbf{z}_m\,. \tag{2.56}$$

Z rovnice $\mathbf{P}_m\mathbf{U}_m = \mathbf{V}_m$ odvodíme pro sloupce $\mathbf{p}_i$ matice $\mathbf{P}_m = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m)$ rekurzi

$$\begin{aligned} &\mathbf{p}_1 = \eta_1^{-1}\mathbf{v}_1\,, \\ &\mathbf{p}_k = \eta_k^{-1}(\mathbf{v}_k - \beta_k\mathbf{p}_{k-1})\,, \quad k = 2, 3, \dots, m\,. \end{aligned} \tag{2.57}$$

Protože $\operatorname{span}\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m\} = \operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$, $\{\mathbf{p}_i\}_{i=1}^m$ je báze v $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$. Označíme-li $\mathbf{z}_m = (\zeta_1, \zeta_2, \dots, \zeta_m)^T$, pak z rovnice $\mathbf{L}_m\mathbf{z}_m = \beta\mathbf{e}_1$ dostaneme

$$\begin{aligned} &\zeta_1 = \beta\,, \\ &\zeta_k = -\lambda_k\zeta_{k-1}\,, \quad k = 2, 3, \dots, m\,. \end{aligned} \tag{2.58}$$

Ze vztahů (2.56) – (2.58) plyne, že pro výpočet $\mathbf{x}_m$ máme rekurentní předpis

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \zeta_k\mathbf{p}_k\,, \qquad k = 1, 2, \dots, m\,. \tag{2.59}$$

Vlastnosti formulí (2.57) – (2.59).

1. Reziduum $\mathbf{r}_k$ lze zapsat ve tvaru

   $$\mathbf{r}_k = \sigma_k\mathbf{v}_{k+1}, \quad k = 0, 1, \dots, m-1, \tag{2.60}$$

   kde σ_k je jisté číslo. To však znamená, že rezidua $\{\mathbf{r}_k\}_{k=0}^{m-1}$ jsou navzájem ortogonální (neboť vektory $\{\mathbf{v}_k\}_{k=1}^m$ jsou ortonormální).

Důkaz. Pro $k = 0$ je $\mathbf{r}_0 = \beta\mathbf{v}_1$, takže $\sigma_0 = \beta$. Pro $k > 0$ pomocí vztahů (2.42), (2.44) a (2.51), v nichž za m dosadíme k, dostaneme

$$\begin{aligned}\mathbf{r}_k &= \mathbf{b} - \mathbf{A}\mathbf{x}_k = \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k) = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k\mathbf{y}_k = \\ &= \mathbf{V}_k(\beta\mathbf{e}_1) - (\mathbf{V}_k\mathbf{H}_k + \mathbf{w}_k\mathbf{e}_k^T)\mathbf{y}_k = \\ &= \mathbf{V}_k(\beta\mathbf{e}_1 - \mathbf{H}_k\mathbf{y}_k) - \mathbf{w}_k\mathbf{e}_k^T\mathbf{y}_k = -\mathbf{w}_k\mathbf{e}_k^T\mathbf{y}_k\,.\end{aligned}$$

Protože podle AGSM je $\mathbf{w}_k = h_{k+1,k}\mathbf{v}_{k+1} = \beta_{k+1}\mathbf{v}_{k+1}$, odvodili jsme, že

$$\mathbf{r}_k = \sigma_k\mathbf{v}_{k+1}, \quad \text{kde} \quad \sigma_k = -\beta_{k+1}\mathbf{e}_k^T\mathbf{y}_k, \qquad k = 1, 2, \ldots, m-1. \qquad \square$$

2. Vektory $\{\mathbf{p}_i\}_{i=1}^m$ jsou *sdružené vzhledem k matici* $\mathbf{A}$, jinými slovy jsou *$A-$ortogonální*, tj. platí $(\mathbf{p}_i, \mathbf{p}_j)_\mathbf{A} = 0$ pro $i \neq j$. $\{\mathbf{p}_i\}_{i=1}^m$ je tedy $A-$ortogonální báze v $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$.

   Důkaz. Pro $m > 0$ pomocí (2.55), (2.46) a (2.53) zjistíme, že matice

   $$\mathbf{P}_m^T\mathbf{A}\mathbf{P}_m = \mathbf{U}_m^{-T}\mathbf{V}_m^T\mathbf{A}\mathbf{V}_m\mathbf{U}_m^{-1} = \mathbf{U}_m^{-T}\mathbf{H}_m\mathbf{U}_m^{-1} = \mathbf{U}_m^{-T}\mathbf{L}_m$$

   je dolní trojúhelníková ($\mathbf{U}_m^{-1}$ je horní trojúhelníková, takže $\mathbf{U}_m^{-T}$ je dolní trojúhelníková a $\mathbf{U}_m^{-T}\mathbf{L}_m$ je proto také dolní trojúhelníková), a protože je symetrická, musí být diagonální. Pro $\mathbf{p}_i \neq \mathbf{o}$ jsou diagonální prvky $\mathbf{p}_i^T\mathbf{A}\mathbf{p}_i \neq \mathbf{o}$, neboť $\mathbf{A}$ je pozitivně definitní. $\square$

Metoda sdružených gradientů. Vektory $\mathbf{p}_k$ očíslujeme od nuly, takže podle (2.59)

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \zeta_{k+1}\mathbf{p}_k, \quad k = 0, 1, \ldots, m-1. \tag{2.61'}$$

Pomocí (2.57) a (2.60) dostaneme

$$\eta_1\sigma_0\mathbf{p}_0 = \mathbf{r}_0, \quad \eta_{k+2}\sigma_{k+1}\mathbf{p}_{k+1} = \mathbf{r}_{k+1} - \frac{\beta_{k+2}\sigma_{k+1}}{\eta_{k+1}\sigma_k}\eta_{k+1}\sigma_k\mathbf{p}_k, \quad k = 0, 1, \ldots, m-2.$$

Označíme-li $\tilde{\mathbf{p}}_k = \eta_{k+1}\sigma_k\mathbf{p}_k$, $\tilde{\beta}_k = -\beta_{k+2}\sigma_{k+1}/(\eta_{k+1}\sigma_k)$, $k = 0, 1, \ldots, m-2$, pak

$$\tilde{\mathbf{p}}_0 = \mathbf{r}_0, \qquad \tilde{\mathbf{p}}_{k+1} = \mathbf{r}_{k+1} + \tilde{\beta}_k\tilde{\mathbf{p}}_k, \quad k = 0, 1, \ldots, m-2. \tag{2.62'}$$

Vektory $\{\tilde{\mathbf{p}}_k\}_{k=0}^{m-1}$ jsou $A-$ortogonální. Dosadíme-li do (2.61') $\mathbf{p}_k = \tilde{\mathbf{p}}_k/(\eta_{k+1}\sigma_k)$, dostaneme

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\tilde{\mathbf{p}}_k, \quad k = 0, 1, \ldots, m-1, \tag{2.61"}$$

kde $\alpha_k = \zeta_{k+1}/(\eta_{k+1}\sigma_k)$. Jestliže $\tilde{\mathbf{p}}_k$ označíme zase jako $\mathbf{p}_k$ a místo $\tilde{\beta}_k$ píšeme β_k, pak z (2.61") a (2.62') dostaneme

$$\mathbf{x}_0 \text{ dané}, \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k, \qquad k = 0, 1, \ldots, m-1, \tag{2.61}$$

$$\mathbf{p}_0 = \mathbf{r}_0, \quad \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k, \quad k = 0, 1, \ldots, m-1. \tag{2.62}$$

Následuje odvození vzorců pro výpočet koeficientů α_k a β_k, při kterém využijeme jen vztahy (2.61), (2.62), ortogonalitu reziduí $\{\mathbf{r}_k\}_{k=0}^m$ a $A-$ortogonalitu vektorů $\{\mathbf{p}_k\}_{k=0}^m$. Z (2.61) dostaneme

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k \,. \tag{2.63}$$

Jestliže rezidua mají být ortogonální, musí platit $(\mathbf{r}_k, \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k) = 0$, takže

$$\alpha_k = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{r}_k)} \,.$$

Pomocí (2.63) a $A-$ortogonality vektorů $\mathbf{p}_{k-1}$, $\mathbf{p}_k$ dostaneme

$$(\mathbf{A}\mathbf{p}_k, \mathbf{r}_k) = (\mathbf{A}\mathbf{p}_k, \mathbf{p}_k - \beta_{k-1}\mathbf{p}_{k-1}) = (\mathbf{A}\mathbf{p}_k, \mathbf{p}_k) \,,$$

takže koeficient

$$\alpha_k = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)} \,. \tag{2.64}$$

Jestliže využijeme $A-$ortogonalitu $\mathbf{p}_{k+1}$ a $\mathbf{p}_k$, tj. $(\mathbf{r}_{k+1} + \beta_k \mathbf{p}_k, \mathbf{A}\mathbf{p}_k) = 0$, dostaneme

$$\beta_k = -\frac{(\mathbf{r}_{k+1}, \mathbf{A}\mathbf{p}_k)}{(\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)} \,.$$

Tento vztah dále upravíme pomocí (2.63). Protože

$$\mathbf{A}\mathbf{p}_k = -\frac{1}{\alpha_k}(\mathbf{r}_{k+1} - \mathbf{r}_k) \,,$$

platí

$$\beta_k = \frac{1}{\alpha_k} \frac{(\mathbf{r}_{k+1}, \mathbf{r}_{k+1} - \mathbf{r}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)} = \frac{(\mathbf{r}_{k+1}, \mathbf{r}_{k+1})}{(\mathbf{r}_k, \mathbf{r}_k)} \,. \tag{2.65}$$

Vzorce (2.61) – (2.65) můžeme zapsat jako

Algoritmus CG (conjugate gradient)

1. zvolíme $\mathbf{x}_0$, ε
2. $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$
3. **for** $k := 0, 1, \ldots$ **until** *konvergence* **do**
4. $\quad \alpha_k := (\mathbf{r}_k, \mathbf{r}_k)/(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)$
5. $\quad \mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
6. $\quad \mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
7. $\quad \beta_k := (\mathbf{r}_{k+1}, \mathbf{r}_{k+1})/(\mathbf{r}_k, \mathbf{r}_k)$
8. $\quad \mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
9. **end**

K algoritmu CG připojíme dvě poznámky.

1. Metodu CG lze považovat za metodu minimalizační, aproximace $\mathbf{x}_k$ je definivána takto:

   určit $\mathbf{x}_k \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ splňující $\|\mathbf{x}_k - \mathbf{x}^*\|_{\mathbf{A}} \leq \|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{A}}\ \forall \mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.

   Při teoreticky přesném provádění operací výpočet skončí nalezením přesného řešení $\mathbf{x}_k = \mathbf{x}^*$ po nejvýše n krocích, tj. pro $k \leq n$. Skutečně, metoda sdružených gradientů je pro pozitivně definitní matice ekvivalentní s metodou ortogonální projekce, pro kterou jsme toto tvrzení dokázali.

2. Pro chybu $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$ platí, viz [46],

   $$\|\mathbf{e}_k\|_{\mathbf{A}} \leq 2 \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^k \|\mathbf{e}_0\|_{\mathbf{A}}\,, \tag{2.66}$$

   kde $\kappa_2(\mathbf{A})$ je spektrální číslo podmíněnosti matice $\mathbf{A}$, tj.

   $$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2 = \lambda_{max}(\mathbf{A})/\lambda_{min}(\mathbf{A}),$$

   přičemž $\lambda_{max}(\mathbf{A})$ resp. $\lambda_{min}(\mathbf{A})$ je největší resp. nejmenší vlastní číslo matice $\mathbf{A}$.

Rychlost konvergence metody CG lze podstatně urychlit pomocí techniky známé jako **Předpodmínění**. Místo úlohy $\mathbf{Ax} = \mathbf{b}$ řešíme úlohu

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P}^{-T}\mathbf{P}^T\mathbf{x} = \mathbf{P}^{-1}\mathbf{b}$$

tj. úlohu

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}\,, \qquad \text{kde} \quad \tilde{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}^{-T}, \quad \tilde{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b}\,, \quad \tilde{\mathbf{x}} = \mathbf{P}^T\mathbf{x}\,, \tag{2.67}$$

přičemž matici $\mathbf{P}$ volíme tak, aby číslo podmíněnosti $\kappa_2(\tilde{\mathbf{A}})$ bylo pokud možno výrazně menší než číslo podmíněnosti $\kappa_2(\mathbf{A})$.

Algoritmus PCG (preconditioned conjugate gradient) dostaneme z algoritmu CG tak, že v něm opatříme všechny neskalární proměnné vlnkou a pak za vlnkované proměnné podle (2.67) dosadíme $\tilde{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}^{-T}$, $\tilde{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b}$, $\tilde{\mathbf{x}}_k = \mathbf{P}^T\mathbf{x}_k$, $\tilde{\mathbf{r}}_k = \tilde{\mathbf{b}} - \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k = \mathbf{P}^{-1}\mathbf{r}_k$ a $\tilde{\mathbf{p}}_k := \mathbf{P}^T\mathbf{p}_k$. Pomocí *předpodmiňovací matice*

$$\mathbf{M} = \mathbf{P}\mathbf{P}^T \tag{2.68}$$

po úpravě dostaneme

Algoritmus PCG (preconditioned conjugate gradient)

1. zvolíme $\mathbf{x}_0$, ε
2. $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$, $\mathbf{p}_0 := \mathbf{z}_0$
3. **for** $k := 0, 1, \ldots$ **until** *konvergence* **do**
4. $\quad \alpha_k := (\mathbf{r}_k, \mathbf{z}_k)/(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)$
5. $\quad \mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$

6. $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
7. $\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$
8. $\beta_k := (\mathbf{r}_{k+1}, \mathbf{z}_{k+1})/(\mathbf{r}_k, \mathbf{z}_k)$
9. $\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$
10. **end**

Vektory $\mathbf{z}_k = \mathbf{M}^{-1}\mathbf{r}_k$, $k = 0, 1, \ldots$, získáme řešením rovnic $\mathbf{M}\mathbf{z}_k = \mathbf{r}_k$. Následuje přehled základních vlastností PCG metody.

1. Vektory $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_k$ jsou navzájem $\mathbf{M}^{-1}-$ortogonální, tj. platí

   $$(\mathbf{r}_i, \mathbf{M}^{-1}\mathbf{r}_j) = 0 \qquad \text{pro } i \neq j.$$

   Vektory $\mathbf{r}_i$ a $\mathbf{r}_j$ jsou tedy sdružené vzhledem k matici $\mathbf{M}^{-1}$.

2. Vektory $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k$ jsou navzájem $A-$ortogonální, tj. platí

   $$(\mathbf{p}_i, \mathbf{A}\mathbf{p}_j) = 0 \qquad \text{pro } i \neq j.$$

   Vektory $\mathbf{p}_i$ a $\mathbf{p}_j$ jsou tedy *sdružené* vzhledem k matici $\mathbf{A}$.

3. Metodu PCG lze považovat za metodu minimalizační: přibližné řešení $\mathbf{x}_k$ minimalizuje $\|\mathbf{x} - \mathbf{x}^*\|_{\mathbf{A}}$ na množině $\mathbf{x}_0 + \mathcal{K}_k(\mathbf{K}, \mathbf{z}_0)$, kde

   $$\mathbf{K} = \mathbf{M}^{-1}\mathbf{A}, \quad \mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0 \quad \text{a} \qquad \mathcal{K}_k(\mathbf{K}, \mathbf{z}_0) = \operatorname{span}(\mathbf{z}_0, \mathbf{K}\mathbf{z}_0, \ldots, \mathbf{K}^{k-1}\mathbf{z}_0)\,.$$

   Při teoreticky přesném provádění operací výpočet skončí nalezením přesného řešení $\mathbf{x}_k = \mathbf{x}^*$ po nejvýše n krocích, tj. pro $k \leq n$.

4. Pro chybu $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$ platí

   $$\|\mathbf{e}_k\|_{\mathbf{A}} \leq 2 \left(\frac{\sqrt{\kappa_2(\mathbf{K})} - 1}{\sqrt{\kappa_2(\mathbf{K})} + 1} \right)^k \|\mathbf{e}_0\|_{\mathbf{A}}\,,$$

   kde $\kappa_2(\mathbf{K})$ je spektrální číslo podmíněnosti matice $\mathbf{K}$, tj.

   $$\kappa_2(\mathbf{K}) = \|\mathbf{K}\|_2 \cdot \|\mathbf{K}^{-1}\|_2 = \max_{1 \leq i \leq n} \lambda_i(\mathbf{K}) / \min_{1 \leq i \leq n} \lambda_i(\mathbf{K})\,.$$

   Vhodnou volbou předpodmiňovací matice $\mathbf{M}$ lze číslo podmíněnosti matice $\mathbf{K}$ zmenšit (vzhledem k číslu podmíněnosti nepředpodmíněné matice $\mathbf{K} = \mathbf{A}$) a tím urychlit konvergenci PCG metody oproti metodě CG.

5. V MATLABu je metoda PCG implementována jako funkce `pcg`.

Volba předpodmiňovací matice. V literatuře, viz např. [33], [30], [4], lze najít řadu vhodných předpodmiňovacích matic. Jejich společným rysem jsou tyto vlastnosti:

a) $\mathbf{M}$ je pozitivně definitní;

b) $\mathbf{M}$ je dobrá aproximace $\mathbf{A}$;

c) sestavení matice $\mathbf{M}$ nevyžaduje velký objem výpočtů;

d) řešení soustav rovnic $\mathbf{M}\mathbf{z}_k = \mathbf{r}_k$ nevyžaduje velký objem výpočtů.

V dalším si uvedeme tři možnosti, jak lze matici $\mathbf{M}$ vybrat.

1. Zvolíme-li $\mathbf{M} = \mathbf{D}$, kde $\mathbf{D} = \mathrm{diag}\{a_{11}, a_{22}, \ldots, a_{nn}\}$, dostáváme zrychlení Jacobiho metody. Tato volba není příliš kvalitní, neboť $\mathbf{M}$ není dostatečně dobrou aproximací matice $\mathbf{A}$.

2. Lepší možnost představuje matice

$$\mathbf{M} = \frac{1}{\omega(2-\omega)}(\mathbf{D} + \omega\mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \omega\mathbf{L})^T,$$

kterou jsme poznali při popisu SSOR metody, viz (2.38). Řešení $\mathbf{z}_k$ SLR $\mathbf{M}\mathbf{z}_k = \mathbf{r}_k$ dostaneme řešením dvou SLR

$$(\mathbf{D} + \omega\mathbf{L})\mathbf{y} = \mathbf{r}_k\,, \qquad (\mathbf{D} + \omega\mathbf{L})^T\mathbf{z}_k = \omega(2-\omega)\mathbf{D}\mathbf{y}$$

s dolní trojúhelníkovou maticí $\mathbf{D} + \omega\mathbf{L}$ a horní trojúhelníkovou maticí $(\mathbf{D} + \omega\mathbf{L})^T$.

3. Volíme $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, kde $\mathbf{L}$ je dolní trojúhelníková matice získaná tzv. *neúplným Choleského* rozkladem matice $\mathbf{A}$ (takové metody se označují zkratkou IC podle anglického „incomplete Cholesky“). Algoritmus označovaný jako IC(0) (podle anglického *incomplete Cholesky with zero-fill*) postupuje stejně jako standardní Choleského rozklad s tím rozdílem, že pro $a_{ij} = 0$ klademe $\ell_{ij} = 0$.

Algoritmus IC(0).

Postupně pro $k = 1, 2, \ldots, n$ vypočteme

$$\ell_{kk} = \left(a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2\right)^{1/2},$$

$$\ell_{ik} = \begin{cases} \dfrac{1}{\ell_{kk}}\left(a_{ik} - \displaystyle\sum_{j=1}^{k-1} \ell_{ij}\ell_{kj}\right) & \text{pro } a_{ik} \neq 0, \\ 0 & \text{pro } a_{ik} = 0, \end{cases} \qquad i = k+1, k+2, \ldots, n\,.$$

Algoritmus IC(0) může zhavarovat, pokud $a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2 \leq 0$. Tato situace nenastane například tehdy, když $\mathbf{A}$ je ryze diagonálně dominantní nebo když mimodiagonální prvky $\mathbf{A}$ jsou nekladné, viz [33]. Níže uvedený algoritmus IC-MJ podle Jenningse a Malika nezhavaruje pro žádnou pozitivně definitní matici, viz [33].

Algoritmus IC-MJ.

1. **for** $k := 1$ **to** n **do** $d_k := a_{kk}$ **end**
2. **for** $k := 1$ **to** n **do**
3. $\quad d_k := d_k - \sum_{j=1}^{k-1} \ell_{kj}^2$
4. $\quad$ **for** $i := k+1$ **to** n **do**
5. $\quad\quad \ell_{ik} := a_{ik} - \sum_{j=1}^{k-1} \ell_{ij}\ell_{kj}$
6. $\quad\quad$ **if** $a_{ik} = 0$
7. $\quad\quad\quad d_k := d_k + |\ell_{ik}|$, $d_i := d_i + |\ell_{ik}|$, $\ell_{ik} := 0$
8. $\quad\quad$ **end**
9. $\quad$ **end**
10. $\quad \ell_{kk} := \sqrt{d_k}$
11. $\quad$ **for** $i := k+1$ **to** n **do** $\ell_{ik} := \ell_{ik}/\ell_{kk}$ **end**
12. **end**

Pokud v řádku 7 vypustíme příkazy $d_k := d_k + |\ell_{ik}|$, $d_i := d_i + |\ell_{ik}|$, dostaneme algoritmus IC(0). Řadu dalších algoritmů založených na principu neúplného Choleského rozkladu lze najít v [33] a [46]. Řešení soustavy rovnic $\mathbf{L}\mathbf{L}^T\mathbf{z}_k = \mathbf{r}_k$ je snadné: nejdříve určíme $\mathbf{y}$ jako řešení soustavy $\mathbf{L}\mathbf{y} = \mathbf{r}_k$ a pak vypočteme $\mathbf{z}_k$ jako řešení soustavy $\mathbf{L}^T\mathbf{z}_k = \mathbf{y}$. V MATLABu je neúplný Choleského rozklad implementován jako funkce `ichol`.

3. Metoda nejmenších čtverců

Úlohu metody nejmenších čtverců (stručně MNČ) lze formulovat velmi obecně, viz [25]. Stručně takovou formulaci uvedeme.

Nechť X je separabilní Hilbertův prostor s ortonormální bází $\{\varphi_i\}_{i=1}^{\infty}$, X_k je k-rozměrný podprostor s bází $\{\varphi_i\}_{i=1}^{k}$ a $f \in X$ je daný prvek. Úloha MNČ zní:

$$\text{určit } s_k^* \in X_k \text{ splňující:} \qquad \|f - s_k^*\| \le \|f - s\| \qquad \forall s \in X_k. \tag{3.1}$$

Řešení s_k^* je $k-$tý částečný součet Fourierovy řady prvku f vzhledem k systému $\{\varphi_i\}_{i=1}^{\infty}$, tj. $s_k^* = \sum_{i=1}^{k}(f,\varphi_i)\varphi_i$. Navíc $f = \lim_{k\to\infty} s_k^* \equiv \sum_{i=1}^{\infty}(f,\varphi_i)\varphi_i$, viz [25], [62].

V této kapitole se zaměříme na MNČ pro přibližné řešení soustav lineárních rovnic, a to v případech, kdy tyto soustavy řešení nemají, tj. když jde o tzv. *přeurčené* SLR, ale také v případech, kdy SLR mají nekonečně mnoho řešení, tj. když jde o tzv. *nedourčené* SLR. V kapitole 4.2 pak ukážeme použití MNČ při aproximaci funkcí.

3.1. Formulace

Zabývejme se řešením soustavy m lineárních rovnic s n neznámými

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,. \tag{3.2}$$

Matice soustavy $\mathbf{A}$ je tedy typu (m,n), vektor $\mathbf{b}$ pravé strany je typu $(m,1)$ a vektor řešení $\mathbf{x}$ je typu $(n,1)$. Nechť $\mathbf{A}' = (\mathbf{A},\mathbf{b})$ je rozšířená matice soustavy a $r = h(\mathbf{A})$ je hodnost matice $\mathbf{A}$. Pak z Frobeniovy věty plyne, že soustava lineárních rovnic má řešení, právě když $h(\mathbf{A}) = h(\mathbf{A}')$. Je-li navíc $r = n$, má SLR jediné řešení, zatímco v případě $r < n$ existuje nekonečně mnoho řešení $\mathbf{x}_0^* + N(\mathbf{A})$, kde $\mathbf{x}_0^*$ je nějaké (partikulární) řešení SLR (3.2) a

$$N(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{o}\} \tag{3.3}$$

je podprostor dimenze $n - r$ známý jako *jádro matice* $\mathbf{A}$.

Pokud $h(\mathbf{A}) < h(\mathbf{A}')$, SLR (3.2) řešení nemá. Můžeme však minimalizovat rozdíl mezi levou a pravou stranou SLR, tedy minimalizovat nějakou normu rezidua $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ jako funkci $\mathbf{x}$. Zvolíme-li euklidovskou normu (tj. délku vektoru), dostaneme metodu nejmenších čtverců: za řešení považujeme vektor $\mathbf{x}^*$, který minimalizuje součet čtverců rozdílů navzájem si odpovídajících složek na levé a pravé straně SLR. Abychom vyjádřili ztrátu přesné rovnosti, budeme úlohu „řešit SLR (3.2) ve smyslu metody nejmenších čtverců“ zapisovat ve tvaru

$$\mathbf{A}\mathbf{x} \cong \mathbf{b} \tag{3.4}$$

a budeme tím rozumět tento problém:

$$\text{určit } \mathbf{x}^* \in \mathbf{R}^n \text{ splňující:} \qquad \|\mathbf{b} - \mathbf{A}\mathbf{x}^*\| \le \|\mathbf{b} - \mathbf{A}\mathbf{x}\| \qquad \forall \mathbf{x} \in \mathbf{R}^n\,. \tag{3.4'}$$

Řešení $\mathbf{x}^*$ úlohy (3.4') budeme nazývat řešením SLR (3.2) ve smyslu metody nejmenších čtverců, stručně MNČ řešením SLR (3.2). Všimněte si, že každé „klasické“ řešení SLR

(3.2) je vždy také jejím MNČ řešením, tj. je řešením úlohy (3.4'). Opak obecně neplatí: jestliže pro MNČ řešení $\mathbf{x}^*$ je $\|\mathbf{b}-\mathbf{A}\mathbf{x}^*\|>0$, pak nemůže platit $\mathbf{A}\mathbf{x}^*=\mathbf{b}$.

Vektorové normy použité ve formulaci (3.4'), a všude dále v této kapitole, jsou euklidovské, tj. $\|\mathbf{v}\|\equiv\|\mathbf{v}\|_2=(\mathbf{v}^T\mathbf{v})^{1/2}$. Také maticové normy, které budeme v této kapitole používat, jsou euklidovké, tj. $\|\mathbf{A}\|\equiv\|\mathbf{A}\|_2=\max_{\mathbf{x}\neq\mathbf{o}}\|\mathbf{A}\mathbf{x}\|_2/\|\mathbf{x}\|_2$.

Existence a jednoznačnost. V dalším budeme používat označení

$$R(\mathbf{A})=\{\mathbf{y}\in\mathbb{R}^m\,|\,\mathbf{y}=\mathbf{A}\mathbf{x},\,\mathbf{x}\in\mathbb{R}^n\}$$

pro *obor hodnot matice* $\mathbf{A}$. Jestliže $\mathrm{span}(\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_r)=\{\sum_{i=1}^r x_i\mathbf{v}_i\,|\,x_1,x_2,\ldots,x_r\in\mathbb{R}\}$ je vektorový prostor generovaný vektory $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_r$, pak $R(\mathbf{A})=\mathrm{span}(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n)$, kde $\mathbf{A}=(\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n)$. Proto se také používá zápis $\mathrm{span}(\mathbf{A})\equiv R(\mathbf{A})\equiv\{\mathbf{A}\mathbf{x}\,|\,\mathbf{x}\in\mathbb{R}^n\}$.

Ukážeme, že úloha

$$\text{určit } \mathbf{y}^*\in R(\mathbf{A}) \text{ splňující:}\qquad \|\mathbf{b}-\mathbf{y}^*\|\le\|\mathbf{b}-\mathbf{y}\|\qquad\forall\mathbf{y}\in R(\mathbf{A}) \tag{3.5}$$

má jediné řešení. Všimněte si, že úloha (3.5) je speciálním případem obecné úlohy (3.1).

Důkaz. Nechť $r=h(\mathbf{A})$ je hodnost matice $\mathbf{A}$ a $\{\mathbf{v}_i\}_{i=1}^r$ je báze v $R(\mathbf{A})$. Pak každý vektor $\mathbf{y}\in R(\mathbf{A})$ lze zapsat ve tvaru $\mathbf{y}=\mathbf{V}\mathbf{x}$, kde $\mathbf{V}=(\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_r)$ a $\mathbf{x}\in\mathbb{R}^r$. Ukážeme, že funkce

$$\varphi(\mathbf{x})=\|\mathbf{b}-\mathbf{V}\mathbf{x}\|^2=(\mathbf{b}-\mathbf{V}\mathbf{x})^T(\mathbf{b}-\mathbf{V}\mathbf{x})=\mathbf{b}^T\mathbf{b}-2\mathbf{x}^T\mathbf{V}^T\mathbf{b}+\mathbf{x}^T\mathbf{V}^T\mathbf{V}\mathbf{x}$$

má jediné minimum. Spočteme gradient $\nabla\varphi(\mathbf{x})$ a Hessovu matici $\nabla^2\varphi(\mathbf{x})$:

$$\nabla\varphi(\mathbf{x})=2\mathbf{V}^T\mathbf{V}\mathbf{x}-2\mathbf{V}^T\mathbf{b}\,,\qquad\nabla^2\varphi(\mathbf{x})=2\mathbf{V}^T\mathbf{V}\,.$$

Protože matice $\mathbf{V}^T\mathbf{V}$ je pozitivně definitní, má soustava rovnic

$$\mathbf{V}^T\mathbf{V}\mathbf{x}=\mathbf{V}^T\mathbf{b} \tag{3.6}$$

jediné řešení $\mathbf{x}_V^*=(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{b}$, které je minimem funkce $\varphi(\mathbf{x})$ v $\mathbb{R}^r$. Proto $\mathbf{y}^*=\mathbf{V}\mathbf{x}_V^*$ je jediné minimum funkce $f(\mathbf{y})=\|\mathbf{b}-\mathbf{y}\|$ na $R(\mathbf{A})$. □

Řešení

$$\mathbf{y}^*=\mathbf{P}_A\mathbf{b},\qquad\mathbf{P}_A=\mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T,$$

úlohy (3.5) je *ortogonální projekce* vektoru pravé strany $\mathbf{b}$ do prostoru $R(\mathbf{A})$ generovaného sloupci matice $\mathbf{A}$. Skutečně, $\mathbf{P}_A\mathbf{b}\in R(\mathbf{A})$ a přitom vektor $\mathbf{r}^*=\mathbf{b}-\mathbf{P}_A\mathbf{b}$ je kolmý k $R(\mathbf{A})$:

$$\mathbf{V}^T\mathbf{r}^*=\mathbf{V}^T(\mathbf{b}-\mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{b})=\mathbf{o}.$$

Jen jako zajímavost si uveďme, že když báze $\{\mathbf{v}_i\}_{i=1}^r$ je ortonormální, pak $\mathbf{V}^T\mathbf{V}=\mathbf{I}$, takže pro $\mathbf{y}^*=\mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{b}=\mathbf{V}\mathbf{V}^T\mathbf{b}=\sum_{i=1}^r(\mathbf{b},\mathbf{v}_i)\mathbf{v}_i$ dostáváme vyjádření ve stejném tvaru, jaký jsme uvedli pro řešení obecného problému (3.1)

Z (3.5) plyne

$$\operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n}\|\mathbf{b}-\mathbf{A}\mathbf{x}\|=\{\mathbf{x}\,|\,\mathbf{A}\mathbf{x}=\mathbf{y}^*\}.$$

Odtud a z (3.4') plyne, že $\mathbf{x}^*$ je řešením soustavy lineárních rovnic

$$\mathbf{A}\mathbf{x} = \mathbf{P}_A\mathbf{b}\,. \tag{3.7}$$

Protože $\mathbf{P}_A\mathbf{b} \in R(\mathbf{A})$, soustava (3.7) má vždy řešení.

Jestliže jsou sloupce matice $\mathbf{A}$ lineárně nezávislé (říkáme také, že matice $\mathbf{A}$ má *plnou sloupcovou hodnost*), soustava (3.7) má jediné řešení. Můžeme ho určit z rovnic (3.6) pro $\mathbf{V} = \mathbf{A}$. Soustava rovnic

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{3.6'}$$

se nazývá *normální soustava rovnic*. Její řešení $\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$.

Pokud má matice $\mathbf{A}$ jen $r = h(\mathbf{A}) < n$ lineárně nezávislých sloupců (říkáme také, že matice $\mathbf{A}$ má *neúplnou sloupcovou hodnost*), má soustava (3.7) nekonečně mnoho řešení. Je-li $\mathbf{x}_0^*$ jedno (partikulární) řešení této soustavy, pak všechna řešení $\mathbf{x}^* = \mathbf{x}_0^* + N(\mathbf{A})$. Reziduum $\mathbf{r}^* = \mathbf{b} - \mathbf{A}\mathbf{x}^* = \mathbf{b} - \mathbf{P}_A\mathbf{b}$ je však pro všechna řešení $\mathbf{x}^*$ stejné.

Shrnutí. Řešení $\mathbf{x}^*$ soustavy lineárních rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$ metodou nejmenších čtverců vždy existuje. Nechť $r = h(\mathbf{A})$ je hodnost matice $\mathbf{A}$ a n je počet jejích sloupců. Pro $r = n$ existuje jediné řešení, zatímco v případě $r < n$ existuje nekonečně mnoho řešení, všechna se stejným reziduem.

Podmíněnost. Při řešení soustav lineárních rovnic s regulární maticí jsme podmíněnost měřili pomocí čísla podmíněnosti definovaného předpisem $\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$. V metodě nejmenších čtverců pracujeme s obdélníkovou maticí a pro tu není matice inverzní definována. Pro každou matici $\mathbf{A}$ však existuje jednoznačně určená *pseudoinverzní matice* označovaná jako $\mathbf{A}^+$, která se v mnoha ohledech chová podobně jako matice inverzní. Obecnou definici uvedeme později, viz kapitola 3.3, prozatím vystačíme s konstatováním, že pro obdélníkovou matici s lineárně nezávislými sloupci $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, speciálně pro regulární matici $\mathbf{A}^+ = \mathbf{A}^{-1}$.

Nyní již můžeme definovat číslo podmíněnosti $\kappa(\mathbf{A})$ matice $\mathbf{A}$ s lineárně nezávislými sloupci předpisem

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^+\|\,.$$

Má-li matice $\mathbf{A}$ neúplnou sloupcovou hodnost, klademe $\kappa(\mathbf{A}) = \infty$. Číslo podmíněnosti je tedy měřítkem blízkosti matice ke ztrátě (zde sloupcové) hodnosti (anglicky „rank deficiency“), více viz kapitola 3.3.

Při posouzení podmíněnosti úlohy „najít řešení soustavy lineárních rovnic ve smyslu metody nejmenších čtverců“ se omezíme na případ přeurčené soustavy s úplnou sloupcovou hodností, tj. $m > n = h(\mathbf{A})$. V tom případě z normálních rovnic $\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{o}$ plyne $(\mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}) = (\mathbf{P}_A\mathbf{b})^T(\mathbf{b} - \mathbf{P}_A\mathbf{b}) = 0$, takže pro úhel θ vektorů $\mathbf{b}$ a $\mathbf{P}_A\mathbf{b}$ platí

$$|\cos\theta| = \frac{|(\mathbf{P}_A\mathbf{b}, \mathbf{b})|}{\|\mathbf{P}_A\mathbf{b}\| \cdot \|\mathbf{b}\|} = \frac{|(\mathbf{P}_A\mathbf{b}, \mathbf{b} - (\mathbf{b} - \mathbf{P}_A\mathbf{b}))|}{\|\mathbf{P}_A\mathbf{b}\| \cdot \|\mathbf{b}\|} = \frac{\|\mathbf{P}_A\mathbf{b}\|}{\|\mathbf{b}\|}\,.$$

Pro jednoduchost se omezíme na malou změnu $\Delta\mathbf{b}$ pravé strany a její vliv na změnu $\Delta\mathbf{x}$ řešení normálních rovnic, tj. zabývejme se úlohou

$$\mathbf{A}^T\mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{A}^T(\mathbf{b} + \Delta\mathbf{b})\,.$$

Protože $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$, máme $\mathbf{A}^T\mathbf{A}\Delta\mathbf{x} = \mathbf{A}^T\Delta\mathbf{b}$, takže $\Delta\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\Delta\mathbf{b} = \mathbf{A}^+\Delta\mathbf{b}$, a tedy

$$\|\Delta\mathbf{x}\| \leq \|\mathbf{A}^+\| \cdot \|\Delta\mathbf{b}\| .$$

Podělíme-li obě strany $\|\mathbf{x}\|$, dostaneme

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}^+\| \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{x}\|} = \kappa(\mathbf{A}) \frac{\|\mathbf{b}\|}{\|\mathbf{A}\| \cdot \|\mathbf{x}\|} \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \leq \kappa(\mathbf{A}) \frac{\|\mathbf{b}\|}{\|\mathbf{P}_A\mathbf{b}\|} \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} = \frac{\kappa(\mathbf{A})}{|\cos\theta|} \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} .$$

Číslo podmíněnosti úlohy $\mathbf{A}\mathbf{x} \cong \mathbf{b}$ vyvolané změnou pravé strany závisí nejen na $\kappa(\mathbf{A})$, ale také na úhlu θ mezi vektorem $\mathbf{b}$ a jeho ortogonální projekcí $\mathbf{P}_A\mathbf{b}$ do $R(\mathbf{A})$. Číslo podmíněnosti je přibližně rovno $\kappa(\mathbf{A})$, když je reziduum $\mathbf{r} = \mathbf{b} - \mathbf{P}_A\mathbf{b}$ malé, tj. když $\cos\theta \approx 1$, avšak číslo podmíněnosti může být libovolně horší než $\kappa(\mathbf{A})$, když je reziduum velké, tj. když $\cos\theta \approx 0$.

Změníme-li matici soustavy $\mathbf{A}$ o $\Delta\mathbf{A}$, tj. když řešíme normální rovnice

$$(\mathbf{A} + \Delta\mathbf{A})^T(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = (\mathbf{A} + \Delta\mathbf{A})^T\mathbf{b} ,$$

pak lze pro malé $\Delta\mathbf{A}$ odvodit „přibližnou nerovnost“

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lesssim [\kappa^2(\mathbf{A})|\mathrm{tg}\,\theta| + \kappa(\mathbf{A})] \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|},$$

viz [21]. Tady je závislost na úhlu θ ještě výraznější: zatímco pro malé reziduum je podmíněnost přibližně $\kappa(\mathbf{A})$, pro velké reziduum je podstatně větší, přibližně $\kappa^2(\mathbf{A})|\mathrm{tg}\,\theta|$.

Shrnutí. Ukázali jsme, že podmíněnost úlohy „najít řešení přeurčené soustavy rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$ metodou nejmenších čtverců“ závisí jak na čísle podmíněnosti $\kappa(\mathbf{A})$ matice $\mathbf{A}$, tak na úhlu θ, který svírá vektor $\mathbf{b}$ se svou ortogonální projekcí $\mathbf{P}_A\mathbf{b}$ do $R(\mathbf{A})$: podmíněnost je tím horší, čím je $\kappa(\mathbf{A})$ a $|\mathrm{tg}(\theta)|$ větší.

3.2. Použití QR transformace

Minimalizace normy rezidua založená na řešení normálních rovnic má jeden velký nedostatek: číslo podmíněnosti matice $\mathbf{A}^T\mathbf{A}$ je kvadrátem čísla podmíněnosti matice $\mathbf{A}$ (viz (3.15) v kapitole 3.3), tj. pro velké $\kappa(\mathbf{A})$ je řešení normálních rovnic špatně podmíněný problém. Proto se běžně používá jiný postup, založený na tzv. QR transformaci. Ten si nyní popíšeme.

Matici $\mathbf{A}$ typu (m, n) lze vyjádřit ve tvaru součinu ortonormální matice $\mathbf{Q}$ řádu m a horní trojúhelníkové matice $\mathbf{R}$ typu (m, n), tj. $\mathbf{A} = \mathbf{Q}\mathbf{R}$. Připomeňme, že ortonormální matice je regulární a platí pro ni $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, kde $\mathbf{I}$ je jednotková matice. Horní trojúhelníkovou maticí se pak rozumí matice s nulovými prvky pod hlavní diagonálou, tj. pro $\mathbf{R} = \{r_{ij}\}$ je $r_{ij} = 0$ když $i > j$. Vyjádření $\mathbf{A} = \mathbf{Q}\mathbf{R}$ se nazývá QR *rozklad* matice $\mathbf{A}$ a konstrukce QR rozkladu se nazývá QR *transformace* nebo také QR *algoritmus*.

Konkrétním QR algoritmům je věnována kapitola 3.4. Předpokládejme tedy, že matice $\mathbf{Q}$ a $\mathbf{R}$ máme k dispozici. Protože

$$\|\mathbf{r}\|^2 = \mathbf{r}^T\mathbf{r} = \mathbf{r}^T\mathbf{Q}\mathbf{Q}^T\mathbf{r} = [\mathbf{Q}^T\mathbf{r}]^T\mathbf{Q}^T\mathbf{r} = \|\mathbf{Q}^T\mathbf{r}\|^2 ,$$

je minimalizace $\|\mathbf{r}\|$ ekvivalentní minimalizaci $\|\mathbf{Q}^T\mathbf{r}\|$. Úpravou obdržíme

$$\|\mathbf{Q}^T\mathbf{r}\| = \|\mathbf{Q}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\| = \|\mathbf{Q}^T(\mathbf{b} - \mathbf{Q}\mathbf{R}\mathbf{x})\| = \|\mathbf{R}\mathbf{x} - \mathbf{Q}^T\mathbf{b}\|\,.$$

Označme

$$\mathbf{R} = \begin{pmatrix}\mathbf{R}_1\\ \mathbf{R}_2\end{pmatrix}, \qquad \mathbf{d} = \mathbf{Q}^T\mathbf{b} = \begin{pmatrix}\mathbf{d}_1\\ \mathbf{d}_2\end{pmatrix}, \tag{3.8}$$

kde $\mathbf{R}_1$ je čtvercová matice řádu n , $\mathbf{R}_2$ je nulová matice typu $(m-n, n)$, $\mathbf{d}_1$ je sloupcový vektor typu $(n, 1)$ a $\mathbf{d}_2$ je sloupcový vektor typu $(m-n, 1)$. Pak

$$\|\mathbf{Q}^T\mathbf{r}\| = \left\|\begin{matrix}\mathbf{R}_1\mathbf{x} - \mathbf{d}_1\\ \mathbf{R}_2\mathbf{x} - \mathbf{d}_2\end{matrix}\right\| = \left\|\begin{matrix}\mathbf{R}_1\mathbf{x} - \mathbf{d}_1\\ -\mathbf{d}_2\end{matrix}\right\|$$

nabývá minima pro $\mathbf{x}^*$ splňující

$$\mathbf{R}_1\mathbf{x}^* = \mathbf{d}_1\,. \tag{3.9}$$

Přitom

$$\min\|\mathbf{r}\| = \min\|\mathbf{Q}^T\mathbf{r}\| = \|\mathbf{d}_2\|. \tag{3.10}$$

Jestliže matice $\mathbf{A}$ má plnou sloupcovou hodnost, tj. $h(\mathbf{A}) = n$, pak matice $\mathbf{R}_1$ je regulární a rovnice (3.9) má jediné řešení. Je-li $\mathbf{d}_2 = \mathbf{o}$, je $\mathbf{x}^*$ klasické řešení, tj. $\mathbf{A}\mathbf{x}^* = \mathbf{b}$.

Ve zbytku této kapitoly prozkoumáne případ, kdy matice $\mathbf{A}$ nemá plnou sloupcovou hodnost. Nechť tedy $r = h(\mathbf{A}) < n$ je počet lineárně nezávislých sloupců matice $\mathbf{A}$. Jejich přeskládáním lze jistě docílit toho, aby lineárně nezávislých bylo prvních r sloupců. Nechť je tedy $\mathbf{P}$ permutační matice taková, že matice $\mathbf{AP}$ má prvních r sloupců lineárně nezávislých, a nechť $\mathbf{AP} = \mathbf{QR}$ je QR rozklad matice $\mathbf{AP}$. Několik QR algoritmů, které kromě matic $\mathbf{Q}$ a $\mathbf{R}$ dodají také permutační matici $\mathbf{P}$, je uvedeno v kapitole 3.4. Předpokládejme proto, že matice $\mathbf{Q}$, $\mathbf{R}$ a $\mathbf{P}$ s vlastností $\mathbf{AP} = \mathbf{QR}$ už máme k dispozici.

Protože permutační matice je ortonormální, podobně jako dříve odvodíme

$$\|\mathbf{Q}^T\mathbf{r}\| = \|\mathbf{Q}^T(\mathbf{b} - \mathbf{A}\mathbf{P}\mathbf{P}^T\mathbf{x})\| = \|\mathbf{Q}^T(\mathbf{b} - \mathbf{Q}\mathbf{R}\mathbf{P}^T\mathbf{x})\| = \|\mathbf{R}\mathbf{P}^T\mathbf{x} - \mathbf{Q}^T\mathbf{b}\|\,.$$

Matici $\mathbf{R}$ a vektor $\mathbf{d} = \mathbf{Q}^T\mathbf{b}$ opět vyjádříme ve tvaru (3.8), kde však nyní $\mathbf{R}_1$ je obdélníková matice typu (r, n) hodnosti r, $\mathbf{R}_2$ je nulová matice typu $(m-r, n)$, $\mathbf{d}_1$ je sloupcový vektor typu $(r, 1)$ a $\mathbf{d}_2$ je sloupcový vektor typu $(m-r, 1)$. Optimální řešení $\mathbf{x}^*$ dostaneme tak, že místo rovnice (3.9) vyřešíme rovnici

$$\mathbf{R}_1\mathbf{P}^T\mathbf{x}^* = \mathbf{d}_1\,.$$

Pro příslušné minimální reziduum opět platí (3.10).

Označíme-li $\mathbf{u}^* = \mathbf{P}^T\mathbf{x}^*$, stačí určit $\mathbf{u}^*$ jako řešení rovnice

$$\mathbf{R}_1\mathbf{u}^* = \mathbf{d}_1 \tag{3.9'}$$

a položit $\mathbf{x}^* = \mathbf{P}\mathbf{u}^*$. Zaveďme si označení

$$\mathbf{R}_1 = (\mathbf{S}, \mathbf{T}), \quad \mathbf{u} = \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix},$$

kde $\mathbf{S}$ je regulární matice řádu r (s nulovými prvky pod hlavní diagonálou), $\mathbf{T}$ je matice typu $(r, n-r)$, $\mathbf{y}$ je sloupcový vektor typu $(r, 1)$ a $\mathbf{z}$ je sloupcový vektor typu $(n-r, 1)$. Každé řešení rovnice (3.9'), tj. rovnice

$$\mathbf{S}\mathbf{y} + \mathbf{T}\mathbf{z} = \mathbf{d}_1\,,$$

dostaneme tak, že zvolíme $\mathbf{z}$ a dopočítáme $\mathbf{y} = \mathbf{S}^{-1}\mathbf{d}_1 - \mathbf{S}^{-1}\mathbf{T}\mathbf{z}$.

Zvolíme-li $\mathbf{z} = \mathbf{o}$, dostaneme řešení

$$\mathbf{u}^* = \begin{pmatrix} \mathbf{S}^{-1}\mathbf{d}_1 \\ \mathbf{o} \end{pmatrix}, \qquad \mathbf{x}^* = \mathbf{P}\mathbf{u}^*,$$

které má nejvýše r nenulových prvků. Toto řešení bývá označováno jako *základní řešení* (v angličtině „basic solution"). V MATLABu dostaneme základní řešení, když použijeme příkaz `x=A\b`.

Jinou konkrétní volbou je řešení $\mathbf{u}^*$, které má minimální euklidovskou normu $\|\mathbf{u}^*\|$ (anglicky „minimum-norm solution"). V tom případě má také $\mathbf{x}^* = \mathbf{P}\mathbf{u}^*$ minimální normu, neboť permutační matice $\mathbf{P}$ je ortonormální, a proto $\|\mathbf{P}\mathbf{u}^*\| = \|\mathbf{u}^*\|$. Ukažme si, jak můžeme řešení s minimální euklidovskou normou určit. Využijeme toho, že $\mathbf{u}$ je rezidum přeurčené soustavy n rovnic

$$\begin{pmatrix} \mathbf{S}^{-1}\mathbf{T} \\ -\mathbf{I} \end{pmatrix} \mathbf{z} = \begin{pmatrix} \mathbf{S}^{-1}\mathbf{d}_1 \\ \mathbf{o} \end{pmatrix}, \tag{3.11}$$

kde $\mathbf{I}$ je jednotková matice řádu $n-r$ a $\mathbf{o}$ je nulový vektor typu $(n-r, 1)$. Protože sloupce matice soustavy (3.11) jsou lineárně nezávislé, existuje jediné řešení $\mathbf{z}^*$ minimalizující euklidovskou normu rezidua $\mathbf{u}$ soustavy rovnic (3.11). $\mathbf{z}^*$ určíme opět metodou nejmenších čtverců. Položíme-li pak $\mathbf{y}^* = \mathbf{S}^{-1}\mathbf{d}_1 - \mathbf{S}^{-1}\mathbf{T}\mathbf{z}^*$, je

$$\mathbf{u}^* = \begin{pmatrix} \mathbf{y}^* \\ \mathbf{z}^* \end{pmatrix} \quad \text{a} \quad \mathbf{x}^* = \mathbf{P}\mathbf{u}^*$$

hledané řešení soustavy rovnic $\mathbf{A}\mathbf{x} = \mathbf{b}$ s minimální euklidovskou normou.

3.3. Použití singulárního rozkladu

Řešení $\mathbf{x}^*$ soustavy lineárních rovnic $\mathbf{A}\mathbf{x} \cong \mathbf{b}$ metodou nejmenších čtverců s minimální euklidovskou normou lze zapsat ve tvaru $\mathbf{x}^* = \mathbf{A}^+\mathbf{b}$, kde $\mathbf{A}^+$ je tzv. pseudoinverze matice $\mathbf{A}$. V následujícím textu nejdříve zavedeme pojem singulárního rozkladu matice a pomocí něj budeme definovat pseudoinverzní matici.

Singulární rozklad matice. Každou matici $\mathbf{A}$ typu (m, n) lze vyjádřit ve tvaru

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

kde $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)$ je ortonormální matice řádu m, $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ ortonormální matice řádu n a $\mathbf{\Sigma}$ je diagonální matice typu (m, n),

$$\mathbf{\Sigma} \equiv \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)\,, \qquad p = \min(m, n)\,,$$

a

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0, \quad \sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_p = 0,$$

kde $h(\mathbf{A}) = r$ je hodnost matice $\mathbf{A}$, viz např. [55].

Vyjádření $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ se nazývá *singulární rozklad matice* $\mathbf{A}$, σ_i jsou *singulární čísla* a $\mathbf{u}_i$ resp. $\mathbf{v}_i$ jsou $i-$tý *levý singulární vektor* resp. $i-$tý *pravý singulární vektor*,

$$\mathbf{u}_i^T\mathbf{A} = \sigma_i\mathbf{v}_i^T\,, \qquad \mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i\,, \qquad i = 1, 2, \ldots, p.$$

Singulární čísla jsou určena jednoznačně, viz [11]. Singulární vektory jednoznačně určeny nejsou, jak plyne z vyjádření $\mathbf{A} = \sum_{i=1}^{p} \sigma_i\mathbf{u}_i\mathbf{v}_i^T$. Singulární rozklad má řadu významných aplikací, z nichž si některé teď uvedeme.

Vlastní čísla a vektory. Protože

$$\mathbf{A}\mathbf{A}^T\mathbf{U} = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\,, \qquad \mathbf{A}^T\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\,,$$

znamená to, že matice $\mathbf{A}\mathbf{A}^T$ i $\mathbf{A}^T\mathbf{A}$ mají vlastní čísla σ_i^2, $i = 1, 2, \ldots, p$, zbývající vlastní čísla těchto matic jsou rovna nule. K vlastnímu číslu σ_i^2 přísluší vlastní vektor $\mathbf{u}_i$ matice $\mathbf{A}\mathbf{A}^T$ a vlastní vektor $\mathbf{v}_i$ matice $\mathbf{A}^T\mathbf{A}$.

Nechť λ je vlastní číslo a $\mathbf{x}$ je odpovídající vlastní vektor symetrické matice $\mathbf{A}$, $\|\mathbf{x}\| = 1$. Pak lze ukázat, že λ je reálné číslo, $|\lambda| = \sigma$ je singulární číslo, pro $\lambda > 0$ je $\mathbf{x} = \mathbf{u} = \mathbf{v}$ nebo $\mathbf{x} = -\mathbf{u} = -\mathbf{v}$, pro $\lambda < 0$ je $\mathbf{x} = \mathbf{u} = -\mathbf{v}$ nebo $\mathbf{x} = -\mathbf{u} = \mathbf{v}$. Je-li $\mathbf{A}$ pozitivně definitní matice, pak $\lambda = \sigma$ a $\mathbf{x} = \mathbf{u} = \mathbf{v}$.

Spektrální maticová norma je rovna největšímu singulárnímu číslu matice,

$$\|\mathbf{A}\|_2 = \sigma_{max}(\mathbf{A}) \equiv \sigma_1\,. \tag{3.12}$$

Skutečně,

$$\sup_{\|\mathbf{x}\|\neq\mathbf{o}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} = \sup_{\|\mathbf{x}\|\neq\mathbf{o}} \frac{\|\mathbf{\Sigma}\mathbf{V}^T\mathbf{x}\|}{\|\mathbf{x}\|} = \sup_{\|\mathbf{y}\|\neq\mathbf{o}} \frac{\|\mathbf{\Sigma}\mathbf{y}\|}{\|\mathbf{V}\mathbf{y}\|} = \sup_{\|\mathbf{y}\|\neq\mathbf{o}} \frac{\|\mathbf{\Sigma}\mathbf{y}\|}{\|\mathbf{y}\|} = \sigma_1\,.$$

Protože σ_1^2 je největší vlastní číslo matice $\mathbf{A}^T\mathbf{A}$, platí také

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{max}(\mathbf{A}^T\mathbf{A})}. \tag{3.13}$$

Nejlepší aproximace maticí nižší hodnosti. Nechť

$$\mathbf{A}_q = \sum_{i=1}^{q} \sigma_i\mathbf{u}_i\mathbf{v}_i^T, \qquad q \leq r.$$

V [30] je dokázáno, že $\mathbf{A}_q$ má hodnost q a je nejlepší aproximací $\mathbf{A}$ mezi všemi maticemi hodnosti q v následujícím smyslu:

$$\min_{h(\mathbf{B})=q} \|\mathbf{A}-\mathbf{B}\|_2 = \|\mathbf{A}-\mathbf{A}_q\|_2 = \sigma_{q+1} \qquad \text{pro } q < r.$$

Nejmenší nenulové singulární číslo σ_r lze proto považovat za kvantitativní vyjádření ztráty hodnosti (v angličtině „rank deficiency“): σ_r je vzdálenost matice $\mathbf{A}$ od „nejbližší matice nižší hodnosti“. Je-li $\mathbf{A}$ regulární řádu n, pak σ_n vyjadřuje vzdálenost od „nejbližší singulární matice“, tj. σ_r je měřítkem ztráty regularity.

Hodnost matice $\mathbf{A}$ v MATLABu dostaneme pomocí funkce `rank(A)` jako počet nenulových singulárních čísel matice $\mathbf{A}$.

Jádro a obor hodnot matice. Snadno se ověří, že obor hodnot $R(\mathbf{A}) = \{\mathbf{Ax} \,|\, \mathbf{x} \in \mathbb{R}^n\}$ a jádro $N(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{Ax} = \mathbf{o}\}$, jsou určeny takto:

$$R(\mathbf{A}) = \operatorname{span}(\mathbf{u}_1, \ldots, \mathbf{u}_r)\,, \qquad N(\mathbf{A}) = \operatorname{span}(\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n)\,.$$

V MATLABu dostaneme singulární rozklad matice $\mathbf{A}$ pomocí příkazu `[U,S,V]=svd(A)`. Funkce `svd` je základem dalších funkcí `orth` a `null` pro výpočet ortonormální báze v $R(\mathbf{A})$ a v $N(\mathbf{A})$.

Pseudoinverzní matice je definována předpisem

$$\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T\,, \qquad \text{kde} \qquad \mathbf{\Sigma}^+ = \operatorname{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)\,.$$

Pro $h(\mathbf{A}) = n$ je $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. Skutečně, protože

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2)\mathbf{V}^T$$

je regulární,

$$(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T = \mathbf{V}\operatorname{diag}(\sigma_1^{-2}, \ldots, \sigma_n^{-2})\mathbf{V}^T\mathbf{V}\operatorname{diag}(\sigma_1, \ldots, \sigma_n)\mathbf{U}^T = \mathbf{A}^+\,.$$

Je-li $\mathbf{A}$ regulární, pak zřejmě $\mathbf{A}^+ = \mathbf{A}^{-1}$.

Snadno se ověří, že $\mathbf{A}^+\mathbf{A} = \mathbf{I}^{n\times n}$ pro $h(\mathbf{A}) = n$ a $\mathbf{A}\mathbf{A}^+ = \mathbf{I}^{m\times m}$ pro $h(\mathbf{A}) = m$.

Pseudoinverzní matice $\mathbf{A}^+$ se často definuje jako jediná matice $\mathbf{X}$ typu (n, m), která splňuje čtyři *Moorovy-Penrousovy* podmínky

(i) $\mathbf{AXA} = \mathbf{A}$ (iii) $(\mathbf{AX})^T = \mathbf{AX}$

(ii) $\mathbf{XAX} = \mathbf{X}$ (iv) $(\mathbf{XA})^T = \mathbf{XA}$

viz např. [30].

Číslo podmíněnosti matice. Číslo podmíněnosti $\kappa_2(\mathbf{A})$ matice $\mathbf{A}$ (v dalším stručně označované jako $\kappa(\mathbf{A})$) definujeme jako podíl největšího a nejmenšího singulárního čísla matice $\mathbf{A}$,

$$\kappa(\mathbf{A}) = \sigma_{max}(\mathbf{A})/\sigma_{min}(\mathbf{A})\,.$$

Má-li matice $\mathbf{A}$ plnou sloupcovou hodnost, tj. když $h(\mathbf{A}) = n$, pak

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^+\|\,, \tag{3.14}$$

jak jsme uvedli v kapitole 3.1. Když má matice $\mathbf{A}$ neúplnou sloupcovou hodnost, tj. když $h(\mathbf{A}) < n$, pak $\sigma_{min}(\mathbf{A}) = 0$, takže $\kappa(\mathbf{A}) = \infty$.

V kapitole 3.2 jsme také uvedli, že

$$\kappa(\mathbf{A}^T\mathbf{A}) = \kappa^2(\mathbf{A})\,. \tag{3.15}$$

To je ale přímý důsledek toho, že singulární čísla matice $\mathbf{A}^T\mathbf{A}$ jsou kvadráty singulárních čísel matice $\mathbf{A}$.

Řešení soustavy lineárních rovnic s minimální euklidovskou normou. Pomocí singulárního rozkladu dostaneme

$$\begin{aligned}\|\mathbf{r}\|^2 =& \|\mathbf{U}^T\mathbf{r}\|^2 = \|\mathbf{U}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\|^2 = \|\mathbf{U}^T(\mathbf{b} - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{x})\|^2 = \|\mathbf{\Sigma}\mathbf{V}^T\mathbf{x} - \mathbf{U}^T\mathbf{b}\|^2 = \\ =& \sum_{i=1}^{r}(\sigma_i\alpha_i - \mathbf{u}_i^T\mathbf{b})^2 + \sum_{i=r+1}^{m}(\mathbf{u}_i^T\mathbf{b})^2\,,\end{aligned}$$

kde $\boldsymbol{\alpha} = \mathbf{V}^T\mathbf{x}$. Minimální reziduum dostaneme, když $\alpha_i^* = \mathbf{u}_i^T\mathbf{b}/\sigma_i$ pro $i = 1, 2, \ldots, r$. Jestliže zvolíme $\alpha_i^* = 0$ pro $i = r{+}1, r{+}2, \ldots, n$, pak také euklidovská norma $\|\mathbf{x}^*\| = \|\boldsymbol{\alpha}^*\|$ MNČ řešení $\mathbf{x}^*$ je minimální. Přitom

$$\mathbf{x}^* = \sum_{i=1}^{r}\alpha_i^*\mathbf{v}_i = \sum_{i=1}^{r}\mathbf{u}_i^T\mathbf{b}/\sigma_i\mathbf{v}_i = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T\mathbf{b} = \mathbf{A}^+\mathbf{b}\,.$$

Existují spolehlivé stabilní algoritmy pro výpočet singulárního rozkladu, viz např. [30], a jejich kvalitní implementace. V MATLABu vypočteme singulární rozklad pomocí funkce `pinv`, příkaz `x=pinv(A)*b` dodá MNČ řešení s minimální euklidovskou normou.

3.4. QR algoritmy

Úvodem připomeňme, že QR rozklad není jednoznačný. Skutečně, nechť $\mathbf{A} = \mathbf{Q}\mathbf{R}$, kde $\mathbf{Q}$ je ortonormální a $\mathbf{R}$ je horní trojúhelníková. Nechť $\mathbf{D}$ je diagonální matice, která má na diagonále jedničky nebo mínus jedničky. Pak $\mathbf{A} = \mathbf{Q}\mathbf{R} = \mathbf{Q}\mathbf{D}\mathbf{D}\mathbf{R} = \bar{\mathbf{Q}}\bar{\mathbf{R}}$, kde $\bar{\mathbf{Q}} = \mathbf{Q}\mathbf{D}$ je ortonormální a $\bar{\mathbf{D}} = \mathbf{D}\mathbf{R}$ je horní trojúhelníková.

V této kapitole uvedeme tři často používané QR algoritmy založené na

- Householderových reflexích
- Givensových rovinných rotacích
- Gramově – Schmidtově ortogonalizaci

Obecně nejlepší je Householderův algoritmus, který proto rozebíráme poměrně podrobně. Zbývající dva algoritmy jsou pro některé typy matic také velmi užitečné. Uvádíme je mimo jiné proto, že Givensovy matice rovinných rotací i Gramův – Schmidův ortonormalizační proces jsou významné nástroje numerické matematiky, které nacházejí uplatnění nejen v metodě nejmenších čtverců.

3.4.1. Householderův QR algoritmus

je založen na použití *Householderových reflektorů*, což jsou matice tvaru

$$\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T, \tag{3.16}$$

kde $\mathbf{w}$ je vektor jednotkové délky, tj. $\|\mathbf{w}\| = 1$. Protože $\mathbf{H}^2 = \mathbf{I}$, je $\mathbf{H} = \mathbf{H}^{-1} = \mathbf{H}^T$, takže $\mathbf{H}$ ortonormální.

Geometricky je $\mathbf{Hx}$ zrcadlový obraz $\mathbf{x}$ vzhledem k nadrovině $\mathrm{span}(\mathbf{w})^\perp$ kolmé k $\mathbf{w}$. To je zřejmé z toho, že ortogonální projekce vektorů $\mathbf{x}$ a $\mathbf{Hx}$ do nadroviny $\mathrm{span}(\mathbf{w})^\perp$ jsou totožné: ortogonální projekce $\mathbf{P}(\mathbf{x}) = \mathbf{x} - (\mathbf{w}^T\mathbf{x})\mathbf{w}$ a ortogonální projekce

$$\mathbf{P}(\mathbf{Hx}) = (\mathbf{I} - 2\mathbf{w}\mathbf{w}^T)\mathbf{x} - (\mathbf{w}^T(\mathbf{I} - 2\mathbf{w}\mathbf{w}^T)\mathbf{x})\mathbf{w} = \mathbf{x} - 2(\mathbf{w}^T\mathbf{x})\mathbf{w} + (\mathbf{w}^T\mathbf{x})\mathbf{w} = \mathbf{P}(\mathbf{x}).$$

Pro daný vektor $\mathbf{x} \neq \mathbf{o}$ se vektor $\mathbf{w}$ Householderovy transformace (3.16) zvolí tak, aby

$$\mathbf{Hx} = \alpha\mathbf{e}_1\,,$$

kde α je skalár a $\mathbf{e}_1$ je první sloupec jednotkové matice. Transformace $\mathbf{Hx}$ tedy anuluje všechny složky $\mathbf{x}$ s výjimkou prvé. V tom je podstata Householderovy transformace! Z rovnice $(\mathbf{I} - 2\mathbf{w}\mathbf{w}^T)\mathbf{x} = \alpha\mathbf{e}_1$ plyne

$$(2\mathbf{w}^T\mathbf{x})\mathbf{w} = \mathbf{x} - \alpha\mathbf{e}_1\,.$$

To ale znamená, že $\mathbf{w}$ je násobkem vektoru $\mathbf{x} - \alpha\mathbf{e}_1$, takže

$$\mathbf{w} = \pm\frac{\mathbf{x} - \alpha\mathbf{e}_1}{\|\mathbf{x} - \alpha\mathbf{e}_1\|}\,.$$

Protože $\mathbf{H}$ je ortonormální, $|\alpha| = \|\mathbf{Hx}\| = \|\mathbf{x}\|$, tj. $\alpha = \pm\|\mathbf{x}\|$. Abychom se vyhnuli tomu, že vektor $\mathbf{x} - \alpha\mathbf{e}_1$ bude malý, zvolíme $\alpha = -\mathrm{sign}(x_1)\|\mathbf{x}\|$ a

$$\mathbf{w} = \frac{\mathbf{x} + \beta\mathbf{e}_1}{\|\mathbf{x} + \beta\mathbf{e}_1\|}\,, \qquad \text{kde} \quad \beta = -\alpha = \mathrm{sign}(x_1)\|\mathbf{x}\|\,.$$

Všimněte si, že když chceme vypočítat $\mathbf{Hx}$, nemusíme matici $\mathbf{H}$ vůbec sestavovat, neboť

$$\mathbf{Hx} = (\mathbf{I} - 2\mathbf{w}\mathbf{w}^T)\mathbf{x} = \mathbf{x} - \lambda\mathbf{w}\,, \qquad \text{kde} \quad \lambda = 2\mathbf{w}^T\mathbf{x}\,.$$

Věnujme se nyní už popisu vlastního Householderova QR algoritmu. V prvním kroku anulujeme poddiagonální prvky prvního sloupce $\mathbf{a}_1 = (a_{11}, a_{21}, \ldots, a_{m1})^T$ matice $\mathbf{A}$. Pokud $a_{21} = a_{31} = \cdots = a_{m1} = 0$, položíme $\mathbf{H}_1 = \mathbf{I}$, $\mathbf{A}_1 = \mathbf{H}_1\mathbf{A} = \mathbf{A}$ a první krok je hotov.

Je-li však alespoň jeden z prvků $a_{21}, a_{31}, \ldots, a_{m1}$ nenulový, anulujeme poddiagonální prvky prvního sloupce matice $\mathbf{A}$ pomocí Householderovy matice

$$\mathbf{H}_1 = \mathbf{I} - 2\mathbf{w}_1\mathbf{w}_1^T, \qquad \text{kde} \quad \mathbf{w}_1 = \frac{\mathbf{a}_1 + \beta_1\mathbf{e}_1}{\|\mathbf{a}_1 + \beta_1\mathbf{e}_1\|}, \quad \beta_1 = \text{sign}(a_{11})\|\mathbf{a}_1\|.$$

Pak v matici $\mathbf{A}_1 = \mathbf{H}_1\mathbf{A}$ je $a_{11}^{(1)} = -\beta_1$ a zbývající prvky prvního sloupce jsou nulové.

Předpokládejme, že matice $\mathbf{A}$ byla transformována v $k-1$ krocích na tvar

$$\mathbf{A}_{k-1} = \mathbf{H}_{k-1}\mathbf{H}_{k-2}\cdots\mathbf{H}_1\mathbf{A} = \begin{pmatrix} a_{11}^{(k-1)} & a_{12}^{(k-1)} & a_{13}^{(k-1)} & \cdots & \cdots & \cdots & a_{1n} \\ 0 & a_{22}^{(k-1)} & a_{23}^{(k-1)} & \cdots & \cdots & \cdots & a_{2n}^{(k-1)} \\ 0 & 0 & a_{33}^{(k-1)} & \cdots & \cdots & \cdots & a_{3n}^{(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ 0 & 0 & 0 & \cdots & a_{k+1,k}^{(k-1)} & \cdots & a_{k+1,n}^{(k-1)} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{mk}^{(k-1)} & \cdots & a_{mn}^{(k-1)} \end{pmatrix}.$$

Tato matice je horní trojúhelníková až do sloupce $k-1$. V $k-$tém kroku transformujeme matici $\mathbf{A}_{k-1}$ na matici

$$\mathbf{A}_k = \mathbf{H}_k\mathbf{A}_{k-1}$$

tak, aby $\mathbf{A}_k$ měla měla nulové poddiagonální prvky v prvních k sloupcích. Pokud $\mathbf{A}_{k-1}$ už v k-tém sloupci pod diagonálou samé nuly má, položíme $\mathbf{H}_k = \mathbf{I}$ a $k-$tý krok je hotov. Je-li však některý z prvků $a_{k+1,k}, a_{k+2,k}, \ldots, a_{m,k}$ nenulový, zvolíme za $\mathbf{H}_k$ Householderovu matici

$$\mathbf{H}_k = \mathbf{I} - 2\mathbf{w}_k\mathbf{w}_k^T,$$

v němž

$$\mathbf{w}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|} \tag{3.17}$$

a složky vektoru $\mathbf{z}_k$ jsou

$$z_i^{(k)} = \begin{cases} 0 & \text{pro } i < k, \\ a_{kk}^{(k-1)} + \beta_k & \text{pro } i = k, \\ a_{ik}^{(k-1)} & \text{pro } i > k, \end{cases} \qquad \beta_k = \text{sign}\left(a_{kk}^{(k-1)}\right)\left(\sum_{i=k}^{m}\left[a_{ik}^{(k-1)}\right]^2\right)^{1/2}. \tag{3.18}$$

Není obtížné prověřit, že matice $\mathbf{A}_{k-1}$ a $\mathbf{A}_k$ mají prvních $k-1$ řádků a sloupců stejných. Nuly pod diagonálou v prvních $k-1$ sloupcích tedy zůstávají zachovány. V $k-$tém sloupci matice $\mathbf{A}_k$ dostaneme na diagonále $-\beta_k$ a pod diagonálou samé nuly.

Nechť $q = \min(m-1, n)$. Pak po provedení q kroků dostaneme horní trojúhelníkovou matici

$$\mathbf{R} = \mathbf{H}_q\mathbf{H}_{q-1}\cdots\mathbf{H}_1\mathbf{A} \equiv \mathbf{H}\mathbf{A}, \qquad \text{kde} \quad \mathbf{H} = \mathbf{H}_q\mathbf{H}_{q-1}\cdots\mathbf{H}_1,$$

a odtud

$$\mathbf{A} = \mathbf{QR}\,, \qquad \text{kde} \quad \mathbf{Q} = \mathbf{H}^T = \mathbf{H}_1\mathbf{H}_2\cdots\mathbf{H}_q\,.$$

Matici $\mathbf{H}$ můžeme počítat tak, že položíme $\mathbf{H} = \mathbf{I}$ a v k-tém kroku provedeme $\mathbf{H} := \mathbf{H}_k\mathbf{H}$. Jestliže má matice $\mathbf{A}$ plnou hodnost, tj. když $h(\mathbf{A}) \equiv r = p$, kde $p = \min(m, n)$, pak také $h(\mathbf{R}) = p$ a matice $\mathbf{R}$ má na hlavní diagonále nenulové prvky. Jestliže však $\mathbf{A}$ plnou hodnost nemá, tj. když $r < p$, pak také $h(\mathbf{R}) = r$, a to znamená, že na hlavní diagonále matice $\mathbf{R}$ jsou i nulové prvky. Vhodným přeuspořádáním sloupců matice $\mathbf{A}$ lze docílit toho, že diagonální prvky r_{ii}, $i = 1, 2, \ldots, r$, budou nenulové, a že řádky $r+1, r+2, \ldots, m$ matice $\mathbf{R}$ budou nulové.

Pivotování po sloupcích. Začneme tím, že si připravíme permutační matici $\mathbf{P} = \mathbf{I}$, kde $\mathbf{I}$ je jednotková matice řádu n, do níž budeme zaznamenávat provedená prohození sloupců. V $k-$tém kroku pak určíme index s *pivotního sloupce* tak, že

$$\gamma_s = \max_{k\le j\le n} \gamma_j\,, \qquad \text{kde} \quad \gamma_j = \sum_{i=k}^{m} a_{ij}^2\,. \tag{3.19}$$

Jestliže $\gamma_s = 0$, je $h(\mathbf{A}) = k-1$ a QR algoritmus končí. Pokud ale $\gamma_s > 0$, prohodíme $k-$tý a $s-$tý sloupec matic $\mathbf{A}_{k-1}$ a $\mathbf{P}$ a teprve pak provedeme Householderovu transformaci. Tak nakonec dostaneme

$$\mathbf{AP} = \mathbf{QR}\,. \tag{3.20}$$

Čísla γ_j se nemusejí počítat v každém kroku znovu, využijeme-li vlastnost:

$$\text{pokud} \quad \mathbf{Qv} = \begin{pmatrix} \alpha \\ \mathbf{w} \end{pmatrix}\,, \quad \text{pak} \quad \|\mathbf{w}\|^2 = \|\mathbf{v}\|^2 - \alpha^2\,,$$

která platí pro každou ortonormální matici $\mathbf{Q}$. Nyní již můžeme uvést

Algoritmus HQR (Householder QR)

1. $\mathbf{H} := \mathbf{I}$, $\mathbf{p} := (1, 2, \ldots, n)^T$, $p := \min(m, n)$, $r := p$
2. **for** $j := 1, 2, \ldots, n$ **do** $\gamma_j := \sum_{i=1}^{m} a_{ij}^2$ **end**
3. **for** $k := 1, 2, \ldots, p$ **do**
4. určíme s a γ_s tak, že $\gamma_s = \max_{k\le j\le n} \gamma_j$
5. **if** $\gamma_s = 0$, $r := k - 1$, **goto** 13, **end**
6. **if** $s \neq k$, prohodíme $\gamma_k \leftrightarrows \gamma_s$, $\mathbf{a}_k \leftrightarrows \mathbf{a}_s$ a $p_k \leftrightarrows p_s$, **end**
7. **if** $k = m$ **goto** 13
8. vypočteme $\mathbf{w}_{k[k:m]}$ podle (3.17) – (3.18)
9. $\mathbf{A}_{[k:m,k:n]} := \mathbf{A}_{[k:m,k:n]} - 2\mathbf{w}_{k[k:m]}(\mathbf{w}_{k[k:m]}^T\mathbf{A}_{[k:m,k:n]})$
10. $\mathbf{H}_{[k:m,1:m]} := \mathbf{H}_{[k:m,1:m]} - 2\mathbf{w}_{k[k:m]}(\mathbf{w}_{k[k:m]}^T\mathbf{H}_{[k:m,1:m]})$
11. **for** $j := k + 1, k + 2, \ldots, n$ **do** $\gamma_j := \gamma_j - a_{kj}^2$ **end**
12. **end**
13. $\mathbf{R} := \mathbf{A}$, $\mathbf{Q} := \mathbf{H}^T$, $\mathbf{P} := (\mathbf{e}_{p_1}, \mathbf{e}_{p_2}, \ldots, \mathbf{e}_{p_n})$

K algoritmu HQR připojíme několik poznámek.

1. V řádku 1 je $\mathbf{I}$ jednotková matice řádu m.

2. Označili jsme $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$.

3. $\mathbf{A}_{[k:m,k:n]}$ je submatice $\mathbf{A}$ tvořená řádky $k, k+1, \ldots, m$ a sloupci $k, k+1, \ldots, n$, podobně $\mathbf{H}_{[k:m,1:m]}$ resp. $\mathbf{w}_{k[k:m]}$ je submatice $\mathbf{H}$ resp. subvektor $\mathbf{w}_k$.

4. V řádku 8 při výpočtu $\mathbf{w}_k$ využijeme v (3.18) toho, že $\beta_k = \mathrm{sign}(a_{kk})\gamma_k^{1/2}$.

5. V řádku 13 $\mathbf{e}_{p_j}$ je p_j-tý sloupec jednotkové matice řádu n.

6. $r = h(\mathbf{A})$ je hodnost matice $\mathbf{A}$.

Redukovaný Householderův QR rozklad. Nechť $n < m$, $\mathbf{Q}_1$ je matice tvořená prvními n sloupci matice $\mathbf{Q}$ a $\mathbf{R}_1$ je čtvercová matice tvořená prvními n řádky matice $\mathbf{R}$, pak

$$\mathbf{AP} = \mathbf{Q}_1\mathbf{R}_1\,. \tag{3.21}$$

Rozklad (3.21) bývá v anglicky psané literatuře označován jako „economy size QR decomposition“, tj. ekonomický (tedy úsporný, redukovaný) QR rozklad. Toto pojmenování se používá také třeba v MATLABu: příkazem `[Q,R,P]=qr(A)` dostaneme úplný rozklad (3.20), zatímco příkazem `[Q1,R1,P]=qr(A,0)` dostaneme redukovaný rozklad (3.21).

HQRe algoritmus (Householder QR economy)

dostaneme z HQR algoritmu náhradou řádků 1, 10 a 13,

1. $\mathbf{p} := (1, 2, \ldots, n)^T$, $p := n$, $r := n$
10. $\mathbf{q}_k := \mathbf{e}_k$, **for** $j := k, k-1, \ldots, 1$ **do** $\mathbf{q}_{k[j:m]} := \mathbf{q}_{k[j:m]} - (2\mathbf{w}_{j[j:m]}^T\mathbf{q}_{k[j:m]})\mathbf{w}_{j[j:m]}$ **end**
13. $\mathbf{R}_1$ je prvních n řádků matice $\mathbf{A}$, $\mathbf{Q}_1 = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n)$

přičemž $\mathbf{e}_k$ je $k-$tý sloupec jednotkové matice řádu m. Rozdíl oproti algoritmu HQR spočívá v tom, že v modifikovaném řádku 10 vytváříme $k-$tý sloupec $\mathbf{q}_k$ matice $\mathbf{Q}_1$. Generují se tedy jen ty ortonormální vektory, které jsou v $\mathbf{Q}_1$! Abychom pochopili řádek 10, musíme si uvědomit, že

$$\mathbf{H}_n\mathbf{H}_{n-1}\cdots\mathbf{H}_1\mathbf{AP} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)\mathbf{R}_1,$$

kde $\mathbf{e}_j$, $j = 1, 2, \ldots, n$, je $j-$tý sloupec jednotkové matice řádu m, takže

$$\mathbf{AP} = \mathbf{H}_1\mathbf{H}_2\cdots\mathbf{H}_n(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)\mathbf{R}_1 \equiv \mathbf{Q}_1\mathbf{R}_1,$$

a že přitom

$$\mathbf{H}_1\mathbf{H}_2\cdots\mathbf{H}_n\mathbf{e}_k = \mathbf{H}_1\mathbf{H}_2\cdots\mathbf{H}_k\mathbf{e}_k\,, \qquad k = 1, 2, \ldots, n.$$

K výpočtu $\mathbf{q}_k$ podle řádku 10 potřebujeme mít k dispozici vektory $\mathbf{w}_{j[j:m]}$, $j = 1, 2, \ldots, k$. K úschově $\mathbf{w}_{j[j+1:m]}$ lze využít poddiagonální pozice matice $\mathbf{A}$, v nichž by jinak byly nuly, $\mathbf{w}_{j[j:j]}$ je třeba uložit zvlášť.

Pokud QR algoritmus používáme jen k nalezení MNČ řešení SLR, v algoritmu HQR změníme řádky 1,10 a 13,

1. $\mathbf{d} := \mathbf{b}$, $\mathbf{p} := (1, 2, \ldots, n)^T$, $p := \min(m, n)$, $r := p$
10. $\quad \mathbf{d}_{[k:m]} := \mathbf{d}_{[k:m]} - (2\mathbf{w}_{k[k:m]}^T \mathbf{d}_{[k:m]})\mathbf{w}_{k[k:m]}$
13. $\mathbf{R}_1 := \mathbf{A}_{[1:r,1:n]}$, $\mathbf{d}_1 := \mathbf{d}_{[1:r]}$, $\mathbf{P} := (\mathbf{e}_{p_1}, \mathbf{e}_{p_2}, \ldots, \mathbf{e}_{p_n})$

a MNČ řešení soustavy $\mathbf{Ax} = \mathbf{b}$ dostaneme jako klasické řešení soustavy $\mathbf{R}_1\mathbf{P}^T\mathbf{x} = \mathbf{d}_1$, viz kapitola 3.2.

Závěr. Householderův QR algoritmus je považován mezi QR algoritmy za nejlepší. Používá ho také funkce `qr` v MATLABu. Pro speciální typy matic však může být některý z dalších známých QR algoritmů efektivnější.

3.4.2. Givensův QR algoritmus

Všimněte si, že matice

$$\mathbf{G} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

je ortonormální, když $c^2 + s^2 = 1$. Tato matice se nazývá *matice rovinné rotace*, protože čísla c a s jsou kosinus a sinus nějakého úhlu, o který se vektor $\mathbf{x}$ rovinně otočí, jestliže ho maticí $\mathbf{G}$ vynásobíme. Buď $\mathbf{x} = (x_1, x_2)^T \neq \mathbf{o}$. Jestliže položíme $c = x_1/\|\mathbf{x}\|$, $s = x_2/\|\mathbf{x}\|$, splňují čísla c a s potřebnou rovnost a platí

$$\mathbf{Gx} = \begin{pmatrix} x_1^2/\|\mathbf{x}\| + x_2^2/\|\mathbf{x}\| \\ -x_1x_2/\|\mathbf{x}\| + x_1x_2/\|\mathbf{x}\| \end{pmatrix} = \begin{pmatrix} \|\mathbf{x}\| \\ 0 \end{pmatrix}.$$

Právě v tom je smysl použití matice rovinné rotace, totiž anulovat druhou složku vektoru.

Buď $\mathbf{a}_1$ první sloupec matice $\mathbf{A}$. Chceme nejdříve vynulovat jeho druhý prvek čili prvek a_{21} matice. Je-li již $a_{21} = 0$, neprovádíme nic. V opačném případě pro $d = \sqrt{a_{11}^2 + a_{21}^2}$ položíme

$$\mathbf{G}_{21} = \begin{pmatrix} a_{11}/d & a_{21}/d & 0 & \ldots & 0 \\ -a_{21}/d & a_{11}/d & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}$$

a provedeme součin $\mathbf{G}_{21}\mathbf{A}$. Protože $\mathbf{G}_{21}\mathbf{a}_1$ je vektor, jehož druhá složka je nulová, anulovali jsme prvek a_{21}.

Jako další budeme anulovat třetí prvek prvního sloupce matice $\mathbf{G}_{21}\mathbf{A}$. Její prvky označíme zase jen a_{ij}, abychom nekomplikovali označení. Je-li $a_{31} \neq 0$, položíme nyní

$d = \sqrt{a_{11}^2 + a_{31}^2}$ a zavedeme další ortonormální matici

$$\mathbf{G}_{31} = \begin{pmatrix} a_{11}/d & 0 & a_{31}/d & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ -a_{31}/d & 0 & a_{11}/d & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}.$$

Po vynásobení touto maticí bude v poloze $(3,1)$ nulový prvek, a jak snadno zjistíme, v předchozím kroku anulovaný prvek v poloze $(2,1)$ zůstane nulový. Tak pokračujeme, až vynulujeme všechny prvky prvního sloupce.

Podobně anulujeme prvky pod diagonálou v dalších sloupcích, obecně prvek $a_{ik} \neq 0$ anulujeme vynásobením maticí $\mathbf{G}_{ik}$, kterou dostaneme z jednotkové matice $\mathbf{I}$ řádu m takto:

$$\mathbf{G}_{ik} := \mathbf{I}\,, \quad \text{pak v } \mathbf{G}_{ik} = \{g_{j\ell}\}_{j,\ell=1}^{m} \quad \text{změníme} \qquad \begin{aligned} g_{kk} &:= \ \ c_{ik}, \quad g_{ki} := s_{ik}, \\ g_{ik} &:= -s_{ik}, \quad g_{ii} := c_{ik}, \end{aligned}$$

přičemž $c_{ik} = a_{kk}/d$, $s_{ik} = a_{ik}/d$ a $d = \sqrt{a_{kk}^2 + a_{ik}^2}$. Důležité je, že jednou anulované pozice zůstavají nulové. Pokud $a_{ik} = 0$, klademe $\mathbf{G}_{ik} = \mathbf{I}$.

Matice $\mathbf{G}_{ik}$ jsou známé jako *Givensovy matice* rovinných rotací. Označíme-li

$$\mathbf{G}_k = \mathbf{G}_{mk}\mathbf{G}_{m-1,k}\dots\mathbf{G}_{k+1,k}, \quad k = 1,2,\dots,q = \min(m-1,n),$$

můžeme psát

$$\mathbf{G}_q\mathbf{G}_{q-1}\dots\mathbf{G}_1\mathbf{A} = \mathbf{R}\,.$$

Přitom $\mathbf{R}$ je horní trojúhelníková matice a $\mathbf{G} = \mathbf{G}_q\mathbf{G}_{q-1}\dots\mathbf{G}_1$ je matice ortonormální. Označíme-li $\mathbf{Q} = \mathbf{G}^T$, pak z rovnosti $\mathbf{GA} = \mathbf{R}$ plyne $\mathbf{A} = \mathbf{G}^T\mathbf{R} = \mathbf{QR}$.

Pivotování. Při anulování prvků k-tého sloupce je vhodné provádět pivotování: zvýšíme tím odolnost QR algoritmu vůči zaokrouhlovacím chybám a zajistíme, aby $r_{kk} \neq 0$ pro $k = 1,2,\dots,r$, kde $r = h(\mathbf{A})$ je hodnost matice $\mathbf{A}$. K dosažení tohoto cíle použijeme sloupcové pivotování. Postupujeme analogicky jako při pivotování v Householderově metodě, viz (3.19), (3.20).

Závěr. Givensův QR algoritmus je výhodný v případě, když matice $\mathbf{A}$ má pod hlavní diagonálou málo nenulových prvků v předem známých pozicích (i,j) a neprovádí se pivotování. Pak totiž stačí použít jen „cílené“ Givensovy rotace $\mathbf{G}_{ij}$. Je-li však $\mathbf{A}$ plná matice, dáme přednost Householderově QR metodě.

3.4.3. Gramův-Schmidtův QR algoritmus

Nechť $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ jsou lineárně nezávislé vektory. Naším cílem je setrojit ortonormální vektory $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ tak, aby $\operatorname{span}(\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n) = \operatorname{span}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$. $\mathbf{q}_1$ dostaneme normalizací $\mathbf{a}_1$, tj. $\mathbf{q}_1 = \mathbf{a}_1/\|\mathbf{a}_1\|$. Předpokládejme, že jsme už sestrojili ortonormální

vektory $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{k-1}$ s vlastností $\text{span}(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{k-1}) = \text{span}(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{k-1})$. Nejdříve najdeme vektor $\hat{\mathbf{q}} = \mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j$, který je ortogonální ke každému z vektorů $\mathbf{q}_i$, $i = 1, 2, \ldots, k-1$. Z podmínky $(\hat{\mathbf{q}}, \mathbf{q}_i) = 0$ dostaneme $r_{ik} = (\mathbf{a}_k, \mathbf{q}_i)$. $\mathbf{q}_k$ obdržíme normalizací $\hat{\mathbf{q}}$, tj. $\mathbf{q}_k = \hat{\mathbf{q}}/\|\hat{\mathbf{q}}\|$. To je

Klasický GSQR algoritmus (Gram-Schmidt QR)

1. $r_{11} := \|\mathbf{a}_1\|$, $\mathbf{q}_1 := \mathbf{a}_1/r_{11}$
2. **for** $k := 1, 2, \ldots, n$ **do**
3. **for** $i := 1, 2, \ldots, k-1$ **do** $r_{ik} := (\mathbf{a}_k, \mathbf{q}_i)$
4. $\hat{\mathbf{q}} := \mathbf{a}_k - \sum_{i=1}^{k-1} r_{ik}\mathbf{q}_i$
5. $r_{kk} := \|\hat{\mathbf{q}}\|$, $\mathbf{q}_k := \hat{\mathbf{q}}/r_{kk}$
6. **end**

Z řádků 4 a 5 plyne

$$\mathbf{a}_k = \sum_{i=1}^{k} r_{ik}\mathbf{q}_i, \quad k = 1, 2, \ldots, n.$$

Jestliže $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$, $\mathbf{Q}_1 = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n)$ a jestliže $\mathbf{R}_1$ je horní trojúhelníková matice řádu n, jejíž naddiagonální prvky r_{ik} jsou definovány GSQR algoritmem, pak zřejmě $\mathbf{A} = \mathbf{Q}_1\mathbf{R}_1$ je redukovaný QR rozklad matice $\mathbf{A}$.

GSQR je standardní Gramův – Schmidtův algoritmus. Existují alternativní formulace, které mají lepší numerické vlastnosti. Nejznámější z nich je modifikovaný Gramův-Schmidtův algoritmus.

GSMQR algoritmus (Gram-Schmidt modified QR)

1. $r_{11} := \|\mathbf{a}_1\|$, $\mathbf{q}_1 := \mathbf{a}_1/r_{11}$
2. **for** $k := 1, 2, \ldots, n$ **do**
3. $\hat{\mathbf{q}} := \mathbf{a}_k$
4. **for** $i := 1, 2, \ldots, k-1$ **do**
5. $r_{ik} := (\hat{\mathbf{q}}, \mathbf{q}_i)$
6. $\hat{\mathbf{q}} := \hat{\mathbf{q}} - r_{ik}\mathbf{q}_i$
7. **end**
8. $r_{kk} := \|\hat{\mathbf{q}}\|$
9. $\mathbf{q}_k := \hat{\mathbf{q}}/r_{kk}$
10. **end**

Podstatou modifikace je využití vztahu $r_{ik} = (\mathbf{a}_k, \mathbf{q}_i) = (\mathbf{a}_k - \sum_{j=1}^{i-1} r_{jk}\mathbf{q}_j, \mathbf{q}_i)$.

Pokud matice $\mathbf{A}$ nemá plnou sloupcovou hodnost, můžeme pomocí sloupcového pivotování algoritmus upravit tak, aby $r_{kk} \neq 0$ pro $k = 1, 2, \ldots, r$, kde $r = h(\mathbf{A})$ je hodnost matice $\mathbf{A}$.

4. 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 4.1.

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 4.2.

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 $\int_a^b \varphi(x)\,\mathrm{d}x$.

4.1. Interpolace

Interpolační funkci $\varphi(x)$ vybíráme z vhodné třídy funkcí. Omezíme se na nejběžnější případy, kdy φ je algebraický polynom (kapitola 4.1.1), po částech algebraický polynom nebo-li splajn (kapitola 4.1.2) a trigonometrický polynom (kapitola 4.1.3). V kapitole 4.1.4 se pak stručně zmíníme o interpolaci funkcí více proměnných.

4.1.1. Polynomická interpolace

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*

$$P_n(x_i) = y_i\,, \qquad i = 0, 1, \ldots, n\,. \tag{4.1}$$

Jestliže

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \tag{4.2}$$

pak lze koeficienty a_i, $i = 0, 1, \ldots, n$, získat řešením SLR

$$\mathbf{V}\mathbf{a} = \mathbf{y}, \tag{4.3}$$

kde

$$\mathbf{V} = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Matice $\mathbf{V}$ je známa jako *Vandermondova matice. Vandermodův determinant*

$$\det\mathbf{V} = \prod_{0 \le i < j \le n} (x_j - x_i)$$

je pro navzájem různé uzly nenulový. SLR (4.3) má tedy jediné řešení. Výpočet koeficientů a_i, $i = 0, 1, \ldots, n$, je ale poměrně pracný, vyžaduje totiž $O(n^3)$ operací. Hodnotu $P_n(\bar{x})$ polynomu P_n v bodu $\bar{x}$ určíme *Hornerovým schématem* pomocí $O(n)$ operací.

Lagrangeův tvar interpolačního polynomu má vyjádření

$$P_n(x) = y_0\ell_0(x) + y_1\ell_1(x) + \cdots + y_n\ell_n(x) = \sum_{i=0}^{n} y_i\ell_i(x)\,, \tag{4.4}$$

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

$$\ell_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}\,. \tag{4.5}$$

Snadno nahlédneme, že

$$\ell_i(x_k) = \begin{cases} 1 & \text{pro } k = i, \\ 0 & \text{pro } k \neq i, \end{cases} \qquad i, k = 0, 1, \ldots, n\,, \tag{4.6}$$

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

Interpolační polynom je daty $[x_i, y_i]$, $i = 0, 1, \ldots, n$, 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 $x_0, x_1, \ldots, x_n$. 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 4.1. Určíme interpolační polynom pro data předepsaná tabulkou

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

Nejdříve získáme fundamentální polynomy

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

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.

Uvedené nedostatky lze překonat pomocí barycentrické interpolační formule. Další možností je použití Newtonova interpolačního polynomu.

Barycentrická interpolační formule. Lagrangeovy fundamentální polynomy $\ell_i(x)$ lze vyjádřit ve tvaru

$$\ell_i(x) = \frac{\omega_{n+1}(x)}{x - x_i} w_i,$$

kde

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

Lagrangeův interpolační polynom (4.4) proto můžeme zapsat ve tvaru

$$P_n(x) = \omega_{n+1}(x) \sum_{i=0}^{n} y_i \frac{w_i}{x - x_i}. \tag{4.7}$$

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

$$1 = \omega_{n+1}(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i}\,. \tag{4.8}$$

Dělíme-li (4.7) výrazem (4.8), člen $\omega_{n+1}(x)$ se zkrátí a dostaneme *barycentrickou interpolační formuli*

$$P_n(x) = \frac{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i} y_i}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i}}, \qquad x \neq x_i, \quad i = 0, 1, \ldots, n. \tag{4.9}$$

Výpočet číselných koeficientů w_i, $i = 0, 1, \ldots, n$, vyžaduje $O(n^2)$ operací. Výpočet $P_n(\bar{x})$ provedeme pomocí $O(n)$ operací. Přidání dalšího uzlu x_{n+1} znamená přepočítání koeficientů w_i, $i = 0, 1, \ldots, n$, což si vyžádá pouhých $O(n)$ operací.

Newtonův tvar interpolačního polynomu.

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}) \,. \tag{4.10}$$

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

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

Hodnotu $z = P_n(\bar{x})$ spočteme zobecněným Hornerovým schématem:

$$z := a_n \text{ a pak pro } i = n-1, n-2, \ldots, 0 \text{ počítej } z := z(\bar{x} - x_i) + a_i \,. \tag{4.11}$$

Koeficienty a_i lze vypočítat přímo z interpolačních podmínek (4.1). Příslušná SLR

$$\mathbf{K}\mathbf{a} = \mathbf{y}$$

má dolní trojúhelníkovou matici soustavy,

$$\mathbf{K} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & 0 & \cdots & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & 0 \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{i=0}^{n-1}(x_n - x_i) \end{pmatrix},$$

takže $\mathbf{a}$ získáme pomocí $O(n^2)$ operací.

Koeficienty Newtonova interpolačního polynomu lze efektivně spočítat pomocí tzv. *poměrných diferencí*.

Poměrné diference. Začneme tím, že z bodů $\{(x_i, y_i)\}_{i=0}^{n}$ vybereme libovolných $k+1$ navzájem různých bodů $\{(x_{i_j}, y_{i_j})\}_{j=0}^{k}$, $0 \leq k \leq n$, a zkonstruujeme interpolační polynom $P_{i_0 i_1 \ldots i_k}$ stupně k splňující interpolační podmínky

$$P_{i_0 i_1 \ldots i_k}(x_{i_j}) = y_{i_j}, \qquad j = 0, 1, \ldots, k. \tag{4.12}$$

Polynom $P_{i_0 i_1 \ldots i_k}$ lze určit rekurzí

$$P_{i_0}(x) = y_{i_0}, \tag{4.13a}$$

$$P_{i_0 i_1 \ldots i_k}(x) = \frac{(x - x_{i_0}) P_{i_1 \ldots i_k}(x) - (x - x_{i_k}) P_{i_0 i_1 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}. \tag{4.13b}$$

Skutečně, snadno ověříme, že $P_{i_0 i_1 \dots i_k}$ splňuje podmínky (4.12).

Všimněte si, že oba polynomy $P_{i_0 i_1 \dots i_{k-1}}$ a $P_{i_0 i_1 \dots i_{k-1} i_k}$ mají kořeny x_{i_0}, x_{i_1}, …, $x_{i_{k-1}}$. Proto existuje jednoznačně definovaný koeficient $a_{i_0 i_1 \dots i_k}$ takový, že

$$P_{i_0 i_1 \dots i_k}(x) = P_{i_0 i_1 \dots i_{k-1}}(x) + a_{i_0 i_1 \dots i_k}(x - x_{i_0})(x - x_{i_1}) \dots (x - x_{i_{k-1}}). \tag{4.14}$$

Označíme-li $a_{i_0} = P_{i_0}(x)$, dostaneme

$$P_{i_0 i_1 \dots i_k}(x) = a_{i_0} + a_{i_0 i_1}(x - x_{i_0}) + \dots + a_{i_0 i_1 \dots i_k}(x - x_{i_0})(x - x_{i_1}) \dots (x - x_{i_{k-1}}). \tag{4.15}$$

Srovnáním koeficientů u x^k na levé a pravé straně výrazu (4.13b) obdržíme

$$a_{i_0 i_1 \dots i_k} = \frac{a_{i_1 i_2 \dots i_k} - a_{i_0 i_1 \dots i_{k-1}}}{x_{i_k} - x_{i_0}}. \tag{4.16}$$

Koeficient $a_{i_0 i_1 \dots i_k}$ je znám jako *poměrná diference*. Při obvyklém značení

$$P[x_{i_0}, x_{i_1}, \dots, x_{i_k}] = a_{i_0 i_1 \dots i_k}$$

tak dostaneme rekurenci

$$\begin{aligned} &P[x_{i_0}] = y_{i_0}, \\ &P[x_{i_0}, x_{i_1}, \dots, x_{i_k}] = \frac{P[x_{i_1}, x_{i_2}, \dots, x_{i_k}] - P[x_{i_0}, x_{i_1}, \dots, x_{i_{k-1}}]}{x_{i_k} - x_{i_0}}. \end{aligned} \tag{4.17}$$

Jestliže v (4.15) zvolíme $k = n$, $i_j = j$, $j = 0, 1, \dots, n$, dostaneme pro $P_n = P_{01\dots n}$ vyjádření

$$P_n(x) = P[x_0] + P[x_0, x_1](x - x_0) + \dots + P[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}), \tag{4.18}$$

známé jako *Newtonův tvar interpolačního polynomu*. Přitom

$$P[x_i] = y_i, \tag{4.19}$$

$$P[x_{i-k}, x_{i-k+1}, \dots, x_{i-1}, x_i] = \frac{P[x_{i-k+1}, \dots, x_i] - P[x_{i-k}, \dots, x_{i-1}]}{x_i - x_{i-k}}. \tag{4.20}$$

Označíme-li $P_{ik} = P[x_{i-k}, x_{i-k+1}, \dots, x_i]$, pak

$$P_n(x) = a_0 + a_1(x - x_0) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}), \tag{4.21}$$

kde $a_i = P_{ii}$, $i = 0, 1, \dots, n$. Tyto koeficienty počítáme rekurzí

$$\left.\begin{aligned} &P_{i0} = y_i, \\ &P_{ik} = \frac{P_{i,k-1} - P_{i-1,k-1}}{x_i - x_{i-k}}, \quad k = 1, 2, \dots, i, \end{aligned}\right\} \quad i = 0, 1, \dots, n. \tag{4.22}$$

Algoritmus výpočtu poměrných diferencí

pro $i = 0, 1, \dots, n$ prováděj :

  $P_{i0} := y_i$

  pro $k = 1, 2, \dots, i$ proveď :

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

  konec cyklu k ,

konec cyklu i .

Výpočet lze přehledně zaznamenat do tabulky

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

kterou vyplňujeme po řádcích. Určení poměrných diferencí P_{ii}, $i = 0, 1, \dots, n$, vyžaduje $O(n^2)$ operací.

Příklad 4.2. Sestavíme Newtonův interpolační polynom pro data z příkladu 4.1. 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 (4.11) 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, $I[x_0, \ldots, x_n, \bar{x}]$ je nejmenší interval obsahující uzly interpolace a bod $\bar{x}$, a nechť $f \in C^{n+1}\langle a, b\rangle$, kde $\langle a, b\rangle \supseteq I[x_0, \ldots, x_n, \bar{x}]$. Pak pro chybu $E_n(\bar{x})$ platí

$$E_n(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(\bar{x}), \tag{4.23}$$

kde $\omega_{n+1}(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$ a $\xi = \xi(\bar{x})$ je (blíže neurčený) bod z intervalu $I[x_0, \ldots, x_n, \bar{x}]$. 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 $\bar{x}$.

Důkaz. Hledejme konstantu K takovou, pro kterou má funkce

$$F(x) = f(x) - P_n(x) - K\omega_{n+1}(x)$$

kořen $\bar{x}$, tj. platí $F(\bar{x}) = 0$. Jestliže taková konstanta existuje, pak $F(x)$ má $n + 2$ kořenů $x_0, x_1, \ldots, x_n, \bar{x}$ v $I[x_0, \ldots, x_n, \bar{x}]$. Opakovaným použitím Rolleovy věty[2] zjistíme, že $F'(x)$ má v uvažovaném intervalu alespoň $n + 1$ kořenů, $F''(x)$ alespoň n kořenů až konečně $F^{(n+1)}(x)$ má v $I[x_0, \ldots, x_n, \bar{x}]$ alespoň jeden kořen ξ. Protože $P^{(n+1)}(x) \equiv 0$,

$$F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - K(n+1)!\,, \quad \text{takže } K = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

$F(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) - \dfrac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) = 0$, tj. (4.23) platí. □

Poměrnou diferenci $f[x_0, x_1, \ldots, x_k]$ funkce f definujeme pomocí předpisů (4.19) a (4.20), v nichž místo y_i píšeme f_i, kde $f_i = f(x_i)$, a místo P píšeme f, tj.

$$f[x_i] = f_i, \quad f[x_{i-k}, x_{i-k+1}, \ldots, x_{i-1}, x_i] = \frac{f[x_{i-k+1}, \ldots, x_i] - f[x_{i-k}, \ldots, x_{i-1}]}{x_i - x_{i-k}}.$$

Zvolíme-li $x_{n+1} = \bar{x}$, $f_{n+1} = f(\bar{x})$, pak z Newtonovy formule (4.18) plyne

$$f(\bar{x}) = P_{n+1}(\bar{x}) = P_n(\bar{x}) + f[x_0, x_1, \ldots, x_n, \bar{x}]\omega_{n+1}(\bar{x}).$$

Porovnáme-li tento vztah s (4.23), dostaneme

$$f[x_0, x_1, \ldots, x_n, \bar{x}] = \frac{f^{(n+1)}(\xi)}{(n+1)!} \quad \text{pro nějaké } \xi \in I[x_0, \ldots, x_n, \bar{x}]. \tag{4.24}$$

Chybu interpolace tedy můžeme zapsat také ve tvaru

$$E_n(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) = f[x_0, x_1, \ldots, x_n, \bar{x}]\omega_{n+1}(\bar{x}). \tag{4.25}$$

Z (4.24) také plyne

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!} \quad \text{pro nějaké } \xi \in I[x_0, x_1, \ldots, x_n]. \tag{4.26}$$

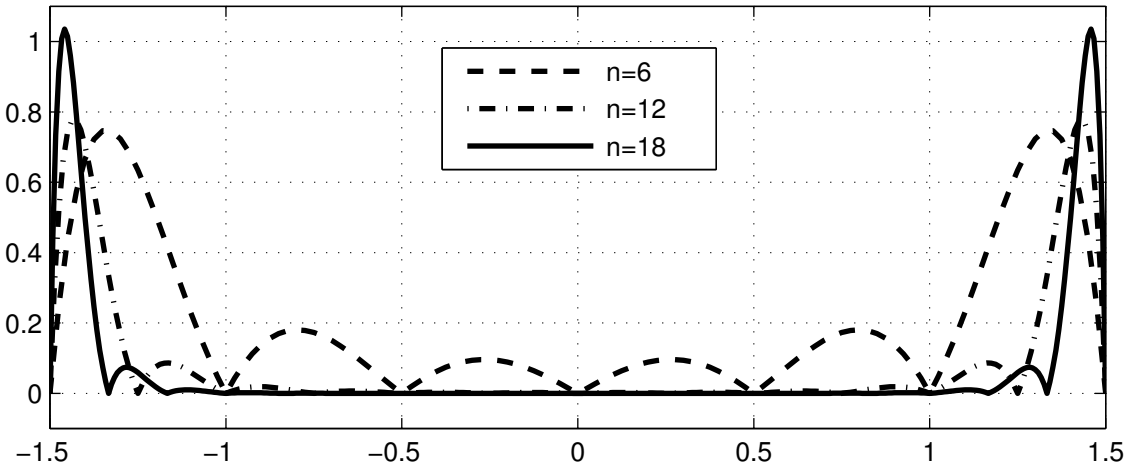
Následuje několik poznámek.

[2]Rolleova věta. Jestliže $f \in C^1\langle a, b\rangle$ a $f(a) = f(b)$, pak existuje alespoň jedno $c \in (a, b)$ takové, že $f'(c) = 0$.

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

$$|E_n(\bar{x})| \le \frac{M_{n+1}}{(n+1)!} \max_{x\in\langle a,\,b\rangle} |\omega_{n+1}(x)|. \tag{4.27}$$

Odhad (4.27) je však obvykle příliš pesimistický.



Obr. 4.1: 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 4.3. Pro $f(x) = \sin x$ lze vzít $M_{n+1} = 1$, proto

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

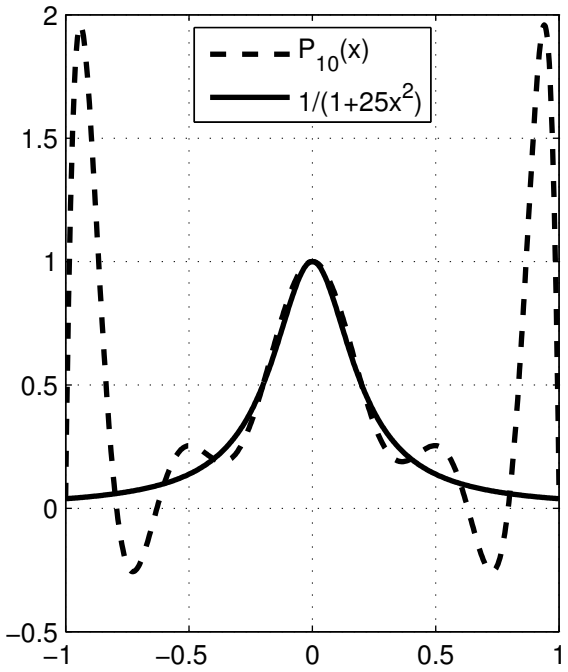
takže $P_n(x) \to f(x)$ pro každé x z libovolného konečného intervalu $\langle a, b\rangle$. □

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. 4.1 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 4.4. Sestrojíme interpolační polynom Rungeovy funkce

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



Obr. 4.2: Interpolační polynom Rungeovy funkce

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

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

Proto

používání interpolačních polynomů vysokých stupňů obecně nelze doporučit.

Vhodnou volbou interpolačních uzlů však můžeme chybu interpolace zmenšit. Na intervalu $\langle a, b\rangle$ je optimální zvolit

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\xi_i, \quad i = 0, 1, \ldots, n,$$

kde tzv. *Čebyševovy uzly*

$$\xi_i = \cos\frac{2i+1}{2n+2}\,\pi, \quad i = 0, 1, \ldots, n,$$

jsou kořeny *Čebyševova polynomu* $T_{n+1}(\xi) = \cos((n+1)\arccos\xi)$. Pomocí trigonometrického vztahu

$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}$$

dostaneme rekurentní předpis

$$T_0(\xi) = 1,\ T_1(\xi) = \xi,\ T_{n+1}(\xi) = 2\xi T_n(\xi) - T_{n-1}(\xi), \quad n = 1, 2, \ldots \tag{4.28}$$

Odtud $\omega_{n+1}(\xi) = \prod_{i=0}^{n}(\xi - \xi_i) = 2^{-n}T_{n+1}(\xi)$, takže

$$\max_{x\in\langle a,b\rangle} |\omega_{n+1}(x)| \le \frac{1}{2^n}\left(\frac{b-a}{2}\right)^{n+1},$$

takže podle (4.27)

$$|E_n(\bar{x})| \le 2\left(\frac{b-a}{4}\right)^{n+1}\frac{M_{n+1}}{(n+1)!}.$$

Hermitova interpolace. Nechť $x_0, x_1, \ldots, x_n$ jsou navzájem různé body. 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* nazveme polynom stupně nejvýše α, který splňuje interpolační podmínky

$$\frac{\mathrm{d}^j}{\mathrm{dx}^j} P_\alpha(x_i) = y_i^{(j)}, \qquad j = 0, 1, \ldots, \alpha_i, \quad i = 0, 1, \ldots, n. \tag{4.29}$$

Nultou derivací přitom rozumíme funkční hodnotu a značíme $y_i^{(0)} = y_i$, $i = 0, 1, \ldots, n$. Ukažme si, že existuje jediný Hermitův polynom splňující podmínky (4.29). Důkaz provedeme ve dvou krocích.

(a) *Jednoznačnost* dokážeme sporem. Předpokládejme, že existují dva polynomy P_α a Q_α splňující interpolační podmínky (4.29). Pak $F_\alpha = P_\alpha - Q_\alpha$ je polynom stupně α, který má $\alpha + 1$ kořenů. To je ale možné jedině tehdy, když F_α je identicky roven nule neboli když $P_\alpha = Q_\alpha$.

(b) *Existence.* Stačí ukázat, že matice soustavy rovnic (4.29) je regulární. Důkaz provedeme sporem. Kdyby matice soustavy (4.29) regulární nebyla, pak bychom pro nulovou pravou stranu soustavy (4.29) dostali nekonečně mnoho různých řešení, což je spor s jednoznačností. □

Jestliže

$$y_i^{(j)} = \frac{\mathrm{d}^j}{\mathrm{dx}^j} f(x_i), \qquad j = 0, 1, \ldots, \alpha_i, \quad i = 0, 1, \ldots, n, \tag{4.30}$$

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

Nechť $\bar{x}$ je libovolný bod, $I[x_0, \ldots, x_n, \bar{x}]$ je nejmenší interval obsahující uzly interpolace a nechť $f \in C^{n+1}\langle a, b\rangle$, kde $\langle a, b\rangle \supseteq I[x_0, \ldots, x_n, \bar{x}]$. Pak pro chybu Hermitovy interpolace v bodě $\bar{x}$ platí

$$f(\bar{x}) - P_\alpha(\bar{x}) = \frac{f^{(\alpha+1)}(\xi)}{(\alpha+1)!}(\bar{x} - x_0)^{\alpha_0+1}(\bar{x} - x_1)^{\alpha_1+1}\ldots(\bar{x} - x_n)^{\alpha_n+1}, \tag{4.31}$$

kde $\xi = \xi(\bar{x})$ je nějaký bod z intervalu $I[x_0, \ldots, x_n, \bar{x}]$. Důkaz lze provést stejně jako pro Lagrangeovu interpolaci, viz důkaz (4.23).

Hermitův interpolační polynom $P_\alpha(x) = \sum_{k=0}^{\alpha} a_k x^k$ lze určit řešením soustavy $\alpha + 1$ rovnic (4.29) pro neznámé koeficienty $\{a_k\}_{k=0}^{\alpha}$.

Příklad 4.5. 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

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

Další možností je vyjádření Hermitova interpolačního polynomu pomocí zobecněných poměrných diferencí.

Newtonův tvar Hermitova interpolačního polynomu. Pomocí uzlů x_0,x_1,...,x_n a odpovídajících hodnot $f_0 = f(x_0), f_1 = f(x_1), \ldots, f_n = f(x_n)$, definujeme posloupnost bodů $[X_0, F_0]$, $[X_1, F_1]$, ..., $[X_\alpha, F_\alpha]$ takto:

$$\begin{aligned} &([X_0,F_0],[X_1,F_1],\ldots,[X_\alpha,F_\alpha]) :=\\ &(\underbrace{[x_0,f_0],\ldots,[x_0,f_0]}_{(\alpha_0+1)\text{-krát}},\underbrace{[x_1,f_1],\ldots,[x_1,f_1]}_{(\alpha_1+1)\text{-krát}},\ldots,\underbrace{[x_n,f_n]\ldots,[x_n,f_n]}_{(\alpha_n+1)\text{-krát}}). \end{aligned}$$

Jestliže $\beta_0 = 0$, $\beta_i = \beta_{i-1} + (\alpha_{i-1} + 1)$, $i = 1, 2, \ldots, n$, pak

$$[X_k, F_k] = [x_i, f_i] \quad \text{pro} \quad k = \beta_i, \beta_i + 1, \ldots, \beta_i + \alpha_i, \quad i = 0, 1, \ldots, n.$$

Nechť $\tilde{X}_k(t)$, $k = 0, 1, \ldots, \alpha$, jsou funkce definované a spojité na intervalu $\langle 0, 1\rangle$, splňující následující tři podmínky:

(1) $\lim_{t\to 0^+} \tilde{X}_k(t) = X_k$ pro $k = 0, 1, \ldots, \alpha$,

(2) $\tilde{X}_k(t) \neq \tilde{X}_\ell(t)$ pro $t \in (0, 1\rangle$, $k, \ell \in \{0, 1, \ldots, \alpha\}$, $k \neq \ell$,

(3) $\tilde{X}_k(t) \in I[x_0, x_1, \ldots, x_n]$ pro $t \in \langle 0, 1\rangle$, $k = 0, 1, \ldots, \alpha$.

Funkce $\tilde{X}_k(t)$, $k = 0, 1, \ldots, \alpha$, lze zvolit třeba takto:

$$\tilde{X}_{\beta_i+j}(t) = x_i + s(i)\frac{h_{min}}{2(\alpha_i+1)}jt, \quad j = 0, 1, \ldots, \alpha_i, \quad i = 0, 1, \ldots, n,$$

kde

$$s(i) = \begin{cases} 1 & \text{pro } i = 0, 1, \ldots, n-1 \\ -1 & \text{pro } i = n. \end{cases}, \; h_{min} = \min\{x_r - x_s \,|\, 0 \le r, s \le n, x_r > x_s\}.$$

Jestliže $\tilde{F}_k(t) = f(\tilde{X}_k(t))$, $t \in \langle 0, 1\rangle$, $k = 0, 1, \ldots, \alpha$, pak

$$\left[\tilde{X}_k(t), \tilde{F}_k(t)\right] \to [X_k, F_k] \quad \text{pro } t \to 0^+, \;\; k = 0, 1, \ldots, \alpha.$$

Body $\{[\tilde{X}_k(t), \tilde{F}_k(t)]\}_{k=0}^{\alpha}$, $t \in (0, 1\rangle$, určují interpolační polynom

$$\tilde{P}_\alpha(x, t) = \tilde{a}_0 + \tilde{a}_1(x - \tilde{X}_0) + \cdots + \tilde{a}_\alpha(x - \tilde{X}_0)(x - \tilde{X}_1)\ldots(x - \tilde{X}_{\alpha-1}),$$

kde $\tilde{a}_k = f[\tilde{X}_0, \tilde{X}_1, \ldots, \tilde{X}_k]$, $k = 0, 1, \ldots, \alpha$. Naším cílem je určit Hermitův polynom $P_\alpha(x) = \lim_{t\to 0^+} \tilde{P}_\alpha(x, t)$. Musíme proto zjistit, čemu se rovná *zobecněná poměrná diference*

$$f[X_{k-s}, \ldots, X_k] = \lim_{t\to 0^+} f[\tilde{X}_{k-s}(t), \ldots, \tilde{X}_k(t)].$$

Zřejmě $f[X_k] = F_k$, $k = 0, 1, \ldots, \alpha$. Jestliže $X_k \neq X_{k-s}$, pak

$$f[X_{k-s}, X_{k-s+1}, \ldots, X_{k-1}, X_k] = \frac{f[X_{k-s+1}, \ldots, X_k] - f[X_{k-s}, \ldots, X_{k-1}]}{X_k - X_{k-s}},$$

a když $X_{k-s} = X_k = x_i$, pak zobecněná poměrná diference

$$f[X_{k-s}, \ldots, X_k] = f[\underbrace{x_i, x_i, \ldots, x_i}_{(s+1)\text{-krát}}] = \frac{f^{(s)}(x_i)}{s\,!},$$

jak plyne z (4.26). Označíme-li $P_{ks} = f[X_{k-s}, \ldots, X_k]$, pak $a_k = P_{kk}$.

Shrnutí. Newtonův tvar Hermitova interpolačního polynomu

$$P_\alpha(x) = a_0 + a_1(x - X_0) + \cdots + a_\alpha(x - X_0)(x - X_1)\ldots(x - X_{\alpha-1}), \tag{4.32}$$

kde $a_k = P_{kk}$, $k = 0, 1, \ldots, \alpha$. Poměrné diference P_{ks} počítáme rekurencí

$$\left.\begin{aligned} &P_{k0} = F_k, \\ &P_{ks} = \begin{cases} \dfrac{P_{k,s-1} - P_{k-1,s-1}}{X_k - X_{k-s}} & \text{pro } \; X_{k-s} \neq X_k, \\ \dfrac{f^{(s)}(x_i)}{s\,!} & \text{pro } \; X_{k-s} = X_k = x_i, \end{cases} \end{aligned}\right\} \begin{aligned} s &= 1, 2, \ldots, k, \\ k &= 0, 1, \ldots, \alpha. \end{aligned}$$

Označíme-li $y_i = f(x_i)$, $y_i^{(s)} = f^{(s)}(x_i)$, $s = 0, 1, \ldots, \alpha_i$, $i = 0, 1, \ldots, n$, pak P_{kk} vypočteme následujícím algoritmem:

Algoritmus výpočtu zobecněných poměrných diferencí

$k := 0$
for $i := 0, 1, \ldots, n$ **do**
  for $j := 0, 1, \ldots, \alpha_i$ **do**
    $X_k := x_i$
    $P_{k0} := y_i$
    for $s := 1, 2, \ldots, k$ **do**
      if $X_{k-s} \neq X_k$
        $P_{ks} := (P_{k,s-1} - P_{k-1,s-1})/(X_k - X_{k-s})$
      else
        $P_{ks} := y_i^{(s)}/s!$
      end
    end s
    $k := k + 1$
  end j
end i.

Zdůrazněme, že (navzájem různé) uzly $\{x_i\}_{i=0}^n$ nemusejí být vzestupně uspořádané.

Příklad 4.6. Sestrojíme Hermitův interpolační polynom pro data z příkladu 4.5. Průběh výpočtu zaznamenáváme do tabulky. Dostaneme

X_k	P_{k0}	P_{k1}	P_{k2}	P_{k3}	P_{k4}				
-1	2					$\Longrightarrow$	a_0	$=$	2,
-1	2	-4				$\Longrightarrow$	a_1	$=$	-4,
-1	2	-4	6			$\Longrightarrow$	a_2	$=$	6,
1	2	0	2	-2		$\Longrightarrow$	a_3	$=$	-2,
1	2	4	2	0	1	$\Longrightarrow$	a_4	$=$	1,

takže $P_4(x) = 2 - 4(x+1) + 6(x+1)^2 - 2(x+1)^3 + (x+1)^3(x-1) = x^4 + 1$.

4.1.2. 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.

Nechť

$$\Delta := \{a \equiv x_0 < x_1 < \cdots < x_{i-1} < x_i < x_{i+1} < \cdots < x_{n-1} < x_n \equiv b\} \tag{4.33}$$

je dělení intervalu $\langle a, b\rangle$. Body $\{x_i\}_{i=0}^n$ se nazývají *uzly splajnu*, anglický termín pro uzel splajnu je *knot*. *Polynomickým splajnem* S na dělení Δ rozumíme funkci, která je každém intervalu $\langle x_{i-1}, x_i\rangle$, $i = 1, 2, \ldots, n$, totožná s nějakým polynomem S_i.

Lokální splajny se vyznačují tím, že polynom S_i je jednoznačně určen daty předepsanými na intervalu $\langle x_{i-1}, x_i\rangle$. Na lokálním dělení

$$\Delta_i := \{x_{i-1} = \xi_0^i < \xi_1^i < \cdots < \xi_{n_i}^i = x_i\} \tag{4.34}$$

lze S_i definovat jako Lagrangeův interpolační polynom stupně n_i určený podmínkami

$$S_i(\xi_k^i) = y_k^i, \qquad k = 0, 1, \ldots, n_i,$$

nebo jako Hermitův interpolační polynom stupně α_i určený podmínkami

$$S_i^{(j)}(\xi_k^i) = [y_k^i]^{(j)}, \qquad j = 0, 1, \ldots, \alpha_k^i, \quad k = 0, 1, \ldots, n_i,$$

kde $\alpha_i = n_i + \sum_{k=0}^{n_i} \alpha_k^i$. Přitom $[y_k^i]^{(0)} = y_k^i$ jsou předepsané hodnoty v uzlech ξ_k^i a $[y_k^i]^{(j)}$, $j \geq 1$, jsou předepsané j-té derivace v uzlech ξ_k^i.

Uvažme případ $n_i = 1$. Pak Lagrangeův interpolační polynom je lineární polynom

$$S_i(x) = y_i + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}).$$

Odpovídající splajn S je znám jako *lineární splajn*. Graficky jde o čáru spojující spojující dva sousední body $[x_{i-1}, y_{i-1}]$ a $[x_i, y_i]$ úsečkou. Nechť

$$h_i = x_i - x_{i-1}, \quad i = 1, 2, \ldots, n, \quad \text{a} \quad h = \max_{1 \leq i \leq n} h_i\,. \tag{4.35}$$

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

$$|f(x) - S(x)| \leq Ch^2, \tag{4.36}$$

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

Hermitovy interpolační polynomy na dělení $\Delta_i := \{x_{i-1} < x_i\}$ nabízejí více možností. Konkrétně podmínky

$$\begin{aligned} S_i^{(j)}(x_{i-1}) &= y_{i-1}^{(j)}, \\ S_i^{(j)}(x_i) \quad &= y_i^{(j)}, \end{aligned} \qquad j = 0, 1, \ldots, m,$$

určují Hermitův interpolační polynom stupně $p = 2m + 1$. Odpovídající splajn S je znám jako *Hermitův interpolační splajn* stupně p. Je spojitý spolu se svými derivacemi až do

řádu m včetně, tj. $S \in C^m\langle a,b\rangle$. Jestliže $y_i^{(j)} = f^{(j)}(x_i)$, $i = 0, 1, \ldots, n$, a $f \in C^{p+1}\langle a,b\rangle$, pak pro chybu interpolace platí

$$|f(x) - S(x)| \le Ch^{p+1}, \tag{4.37}$$

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

Populární *kubický Hermitův splajn* je na intervalu $\langle x_{i-1}, x_i\rangle$ určen podmínkami

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

Příslušný Hermitův polynom S_i lze zapsat ve tvaru

$$S_i(x) = y_{i-1} + sd_{i-1} + s^2\,\frac{3\delta_i - 2d_{i-1} - d_i}{h_i} + s^3\,\frac{d_{i-1} - 2\delta_i + d_i}{h_i^2}, \tag{4.38}$$

kde

$$s = x - x_{i-1} \quad \text{a} \quad \delta_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}. \tag{4.39}$$

Pro chybu kubického Hermitova splajnu podle (4.37) platí

$$|f(x) - S(x)| \le Ch^4, \tag{4.40}$$

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

Nelokální splajny. Pod nelokálním splajnem stupně p na dělení Δ budeme rozumět funkci $S : \langle a,b\rangle \to \mathbb{R}$ s následujícími dvěma vlastnostmi:

(a) $S \in C^{p-1}\langle a,b\rangle$ je spojitá spolu se svými derivacemi až do řádu $p-1$ včetně;

(b) S je na intervalu $\langle x_{i-1}, x_i\rangle$, $i = 1, 2, \ldots, n$, totožná s polynomem S_i stupně p.

Množina $\mathcal{S}_\Delta^p$ splajnů stupně p na dělení Δ je lineární prostor: jestliže $S_1, S_2 \in \mathcal{S}_\Delta^p$ a $\alpha_1, \alpha_2 \in \mathbb{R}$, pak $\alpha_1 S_1 + \alpha_2 S_2 \in \mathcal{S}_\Delta^p$.

Polynomy $\{S_i\}_{i=1}^n$ jsou určeny pomocí $n(p+1)$ koeficientů. Tyto koeficienty však nemohou nabývat libovolných hodnot, v úvahu je třeba vzít podmínky spojitosti ve vnitřních uzlech:

$$S_{i-1}^{(j)}(x_i) = S_i^{(j)}(x_i), \qquad j = 0, 1, \ldots, p-1, \quad i = 1, 2, \ldots, n-1. \tag{4.41}$$

Prostor $\mathcal{S}_\Delta^p$ má tedy dimenzi $d_\Delta^p = n(p+1) - p(n-1) = n + p$. K určení splajnu $S \in \mathcal{S}_\Delta^p$ je třeba předepsat $n + p$ podmínek. Obvykle se volí

$$S(x_i) = y_i, \quad i = 0, 1, \ldots, n, \tag{4.42}$$

kde $\{y_i\}_{i=0}^n$ jsou předepsané hodnoty v uzlech $\{x_i\}_{i=0}^n$. Zbývá předepsat dalších $p-1$ podmínek. Je-li $p = 2m-1$ liché, pak $p-1 = 2(m-1)$, což umožňuje v každém z krajních uzlů a, b předepsat $m-1$ *okrajových podmínek* (v anglicky psané literatuře se používá termín *end conditions*). Typické okrajové podmínky pro splajny lichého stupně jsou:

(a) *Hermitovy* okrajové podmínky: $S^{(j)}(a) = y_a^{(j)}$, $S^{(j)}(b) = y_b^{(j)}$, $j = 1, \ldots, m-1$,

(b) *přirozené* okrajové podmínky: $S^{(j)}(a) = y_a^{(j)}$, $S^{(j)}(b) = y_b^{(j)}$, $j = m, \ldots, 2m-2$,

(c) *periodické* okrajové podmínky: $S^{(j)}(a) = S^{(j)}(b)$, $j = 0, \ldots, 2m-2$.

V případě Hermitových a přirozených okrajových podmínek jsou $y_a^{(j)}$ resp. $y_b^{(j)}$ předepsané hodnoty derivací v bodu a resp. b. Dá se ukázat, že podmínky (4.42) doplněné o okrajové podmínky (a) nebo (b) nebo (c) definují splajn $S \in \mathcal{S}_\Delta^p$ lichého stupně $p = 2m-1$ jednoznačně, viz [20]. Poznamenejme, že přirozené okrajové podmínky se tradičně uvažují jako homogenní, tj. pro $y_a^{(j)} = y_b^{(j)} = 0$, $j = m, \ldots, 2m-2$. Následuje podrobný popis nejznámějšího kubického splajnu.

Kubický splajn je funkce $S \in \mathcal{S}_\Delta^3$. Kubický polynom S_i ve tvaru (4.38) splňuje podmínky

$$S_i(x_i) = S_{i+1}(x_i) = y_i, \quad S_i'(x_i) = S_{i+1}'(x_i) = d_i, \quad i = 1, 2, \ldots, n-1.$$

Ke splnění podmínek spojitosti (4.41) proto stačí požadovat, aby

$$S_i''(x_i) = S_{i+1}''(x_i)\,, \qquad i = 1, 2, \ldots, n-1. \tag{4.43}$$

Derivováním (4.38) 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, obdržíme

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

Dosazením do (4.43) tak dostaneme $n-1$ rovnic

$$h_{i+1}d_{i-1} + 2(h_i + h_{i+1})d_i + h_i d_{i+1} = 3(h_{i+1}\delta_i + h_i\delta_{i+1})\,, \qquad i = 1, 2, \ldots, n-1 \tag{4.44}$$

pro neznámé $d_0, d_1, \ldots, d_n$. Zbývající dvě rovnice doplníme podle zvolených okrajových podmínek.

Hermitovy okrajové podmínky. Zvolíme $S'(a) = d_a$, $S'(b) = d_b$, nebo-li

$$d_0 = d_a, \qquad d_n = d_b, \tag{4.45}$$

kde d_a resp. d_b jsou předepsané hodnoty prvních derivací na okraji a resp. b. Pak v soustavě (4.44) 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 neznámé $d_i, i = 1, 2, \ldots, n-1$. Matice soustavy je třídiagonální, diagonálně dominantní, takže soustavu lze snadno vyřešit GEM upravenou pro soustavy s třídiagonální maticí.

Extremální vlastnost. 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

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

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.

Přirozené okrajové podmínky. Zvolíme $S''(a) = M_a$, $S''(b) = M_b$. Tak dostaneme

$$4d_0 + 2d_1 = 6\delta_1 - h_1 M_a, \tag{4.46a}$$

$$2d_{n-1} + 4d_n = 6\delta_n + h_n M_b. \tag{4.46b}$$

Výslednou soustavu $n+1$ rovnic pro neznámé $d_0, d_1, \ldots, d_n$ dostaneme tak, že nejdříve vezmeme rovnici (4.46a), pak rovnice (4.44) a nakonec rovnici (4.46b).

Periodické okrajové podmínky. Protože má platit $S''(a) = S''(b)$, sestavíme rovnici

$$S''(x_0) = S''(x_n)$$

a využijeme podmínky

$$S(a) = y_0 = y_n = S(b), \qquad S'(a) = d_0 = d_n = S'(b).$$

Tak dostaneme rovnici

$$h_n d_1 + h_1 d_{n-1} + 2(h_1 + h_n) d_n = 3(h_n \delta_1 + h_1 \delta_n), \tag{4.47a}$$

kde $\delta_1 = (y_1 - y_n)/h_1$. Změní se také první z rovnic (4.44), ta bude nyní tvaru

$$2(h_1 + h_2) d_1 + h_1 d_2 + h_2 d_n = 3(h_2 \delta_1 + h_1 \delta_2). \tag{4.47b}$$

Výslednou soustavu n rovnic pro neznámé $d_1, d_2, \ldots, d_n$ dostaneme tak, že nejdříve vezmeme rovnici (4.47b), pak rovnice (4.44) pro $i = 2, 3, \ldots, n-1$, a nakonec rovnici (4.47a).

Podmínky not a knot. Jestliže první ani druhé derivace v krajních bodech a, b neznáme, osvědčil se postup, označovaný v anglicky psané literatuře jako *not a knot*. Myšlenka je jednoduchá: požadujeme, aby splajn byl 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 uzel splajnu nebo-li knot, 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 (4.43) 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}. Když pomocí (4.38) vyjádříme podmínku $S_1'''(x_1) = S_2'''(x_1)$ a upravíme ji pomocí první rovnice soustavy (4.44), dostaneme rovnici

$$h_2 d_0 + (h_1 + h_2) d_1 = [(3h_1 + 2h_2) h_2 \delta_1 + h_1^2 \delta_2] / (h_1 + h_2)\,. \tag{4.48a}$$

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

$$(h_{n-1} + h_n) d_{n-1} + h_{n-1} d_n = [h_n^2 \delta_{n-1} + (2h_{n-1} + 3h_n) h_{n-1} \delta_n] / (h_{n-1} + h_n)\,. \tag{4.48b}$$

Neznámé $d_0, d_1, \ldots, d_n$ pak dostaneme jako řešení soustavy rovnic, z nichž první je rovnice (4.48a), pak následují rovnice (4.44) a nakonec přijde rovnice (4.48b).

V MATLABu lze pro výpočet kubického splajnu použít funkci `spline`. Základní volbou jsou podmínky not a knot, použít lze ale také Hermitovy okrajové podmínky.

Aproximace křivky. Křivku $\mathbf{x} = \boldsymbol{\varphi}(t)$, $t \in \langle 0, \ell \rangle$, aproximujeme křivkou $\mathbf{x} = \mathbf{s}(t)$ tak, že i-tá složka s_i funkce $\mathbf{s}$ je kubický splajn aproximující i-tou složku φ_i funkce $\boldsymbol{\varphi}$ na vhodném dělení intervalu $\langle 0, \ell \rangle$. Pokud známe body $\{\mathbf{x}_i\}_{i=0}^n$, kterými má procházet splajn, zvolíme dělení $0 = t_0 < t_1 < \cdots < t_n \equiv \ell$ třeba tak, že $t_i = t_{i-1} + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2$, $i = 1, 2, \ldots, n$.

Chyba interpolace. Jestliže $y_i = f(x_i)$, $i = 0, 1, \ldots, n$, $d_0 = f'(x_0)$, $d_n = f'(x_n)$, $M_a = f''(a)$, $M_b = f''(b)$, a když $f \in C^4\langle a, b \rangle$, pak pro chybu interpolace platí (4.40).

Bázové splajny. Splajn $S_p \in \mathcal{S}_\Delta^p$ lze vyjádřit ve tvaru

$$S_p(x) = \sum_{j=1}^{n+p} a_j \varphi_j^p(x), \tag{4.49}$$

kde $\{\varphi_j^p\}_{j=1}^{n+p}$ je báze prostoru $\mathcal{S}_\Delta^p$. Pro numerické výpočty je velmi vhodná báze tvořená tak zvanými *bázovými splajny*, stručně *B-splajny*.

Abychom mohli definovat B-splajny, potřebujeme dělení (4.33) intervalu $\langle a, b \rangle$ rozšířit o p uzlů nalevo od a a o p uzlů napravo od b. Připustíme přitom, že tyto přidané uzly mohou být násobné. Dostaneme tak dělení

$$\bar{\Delta} := \{x_{-p} \le \cdots \le x_{-1} \le a \equiv x_0 < x_1 < \cdots < x_n \equiv b \le x_{n+1} \le \cdots \le x_{n+p}\}. \tag{4.50}$$

B-splajn řádu k příslušný uzlu x_i dělení $\bar{\Delta}$ budeme značit B_{ik}. Definice je rekurzivní, takže začneme splajnem řádu 1,

$$B_{i1} = \begin{cases} 1 & \text{pro } x_i \le x < x_{i+1}, \\ 0 & \text{jinak,} \end{cases} \tag{4.51}$$

a B-splajn řádu $k > 1$ definujeme předpisem

$$B_{ik} = \omega_{ik} B_{i,k-1} + (1 - \omega_{i+1,k}) B_{i+1,k-1}, \tag{4.52}$$

kde

$$\omega_{ik}(x) = \begin{cases} \dfrac{x - x_i}{x_{i+k-1} - x_i} & \text{když } x_i \neq x_{i+k-1}, \\ 0 & \text{jinak.} \end{cases} \tag{4.53}$$

Snadno ověříme, že B_{ik} je po částech polynom stupně $k-1$ a že $B_{ik} = 0$ pro $x \notin \langle x_i, x_{i+k})$. Pro derivaci B'_{ik} lze odvodit

$$B'_{ik} = \alpha_{ik} B_{i,k-1} - \alpha_{i+1,k} B_{i+1,k-1}, \tag{4.54}$$

kde

$$\alpha_{ik} = \begin{cases} \dfrac{k-1}{x_{i+k-1} - x_i} & \text{když } x_i \neq x_{i+k-1}, \\ 0 & \text{jinak.} \end{cases} \tag{4.55}$$

Protože lineární B-splajn B_{i2} je spojitý, $B_{ik} \in C^{k-2}\langle a, b\rangle$. Snadno ověříme, že funkce $\{B_{j,p+1}\}_{j=-p}^{n-1}$ tvoří bázi v $\mathcal{S}_\Delta^p$. Pokud se vrátíme k vyjádření (4.49), pak

$$\varphi_j^p = B_{j-p-1,p+1} \equiv B_{j-r,r}, \qquad j = 1, 2, \ldots, n+p, \tag{4.56}$$

kde $r = p + 1$. Na intervalu $\langle x_i, x_{i+1}\rangle$ mohou být nenulové jen B-splajny $\{B_{jk}\}_{j=i-k+1}^{i}$, $k = 1, 2, \ldots, r$. Proto

$$S_p(x)\Big|_{x \in \langle x_i, x_{i+1}\rangle} \equiv S_{i+1}^p(x) = \sum_{j=i-r+1}^{i} a_{j+r} B_{jr}(x) = \sum_{j=1}^{r} a_{i+j} B_{i-r+j,r}(x). \tag{4.57}$$

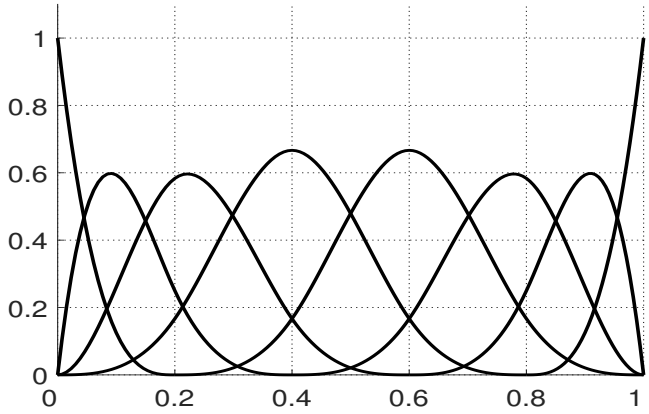
Algoritmus BSPL pro výpočet $\{B_{jr}(x)\}_{j=i-r+1}^{i}$ pro $x \in \langle x_i, x_{i+1})$.

1. zvol $r \ge 2$, $x \in \langle x_i, x_{i+1})$
2. vynuluj matici $\mathbf{B} = \{b_{ij}\}$ typu $(r+1, r)$
3. $b_{r1} := 1$
4. **for** $k := 2, \ldots, r$ **do**
5. **for** $j := r - k + 1, \ldots, r$ **do**
6. $b_{jk} := \omega_{jk}(x) b_{j,k-1} + (1 - \omega_{j+1,k}(x)) b_{j+1,k-1}$
7. **end**
8. **end**

Pak $B_{jr}(x) = b_{j+r-i,r}$, $j = i - r + 1, \ldots, i$, najdeme v posledním sloupci matice $\mathbf{B}$.

Obrázek 4.3 zobrazuje B-splajny řádu 4 na dělení

$$\bar{\Delta} := \{0 = 0 = 0 = 0 < 0{,}2 < 0{,}4 < 0{,}6 < 0{,}8 < 1 = 1 = 1 = 1\}.$$



Obr. 4.3: B-splajny $\{B_{i4}\}_{i=-3}^{4}$

Pro vyjádření okrajových podmínek potřebujeme derivace splajnu. Zřejmě

$$\left[S_{i+1}^{p}\right]^{(q)}(x)\Big|_{x\in\langle x_i,x_{i+1}\rangle} = \sum_{j=i-r+1}^{i} a_{j+r} B_{jr}^{(q)}(x) = \sum_{j=1}^{r} a_{i+j} B_{i-r+j,r}^{(q)}(x). \tag{4.58}$$

Algoritmus BSPLd pro výpočet derivací B-splajnů $\{B_{jr}^{(q)}(x)\}_{j=i-r+1}^{i}$ pro $x \in \langle x_i, x_{i+1}\rangle$.

1. zvol $r \geq 2$, $0 \leq q \leq r-1$, $x \in \langle x_i, x_{i+1}\rangle$
2. vynuluj matici $\mathbf{B} = \{b_{ij}\}$ typu $(r+1, r)$
3. $b_{r1} := 1$
4. **for** $k := 2, \ldots, r$ **do**
5. **for** $j := r-k+1, \ldots, r$ **do**
6. **if** $k \leq r-q$
7. $b_{jk} := \omega_{jk}(x) b_{j,k-1} + (1-\omega_{j+1,k}(x)) b_{j+1,k-1}$
8. **else**
9. $b_{jk} := \alpha_{jk} b_{j,k-1} - \alpha_{j+1,k} b_{j+1,k-1}$
10. **end**
11. **end**
12. **end**

Pak $B_{jr}^{(q)}(x) = b_{j+r-i,r}$, $j = i-r+1, \ldots, i$, najdeme v posledním sloupci matice $\mathbf{B}$.

Kardinální B-spajny jsou B-splajny na rovnoměrném dělení s uzly $x_i = ih$, kde h je konstantní délka kroku. Kardinální B-spajny jsou tvarově stejné, navzájem posunuté. Platí

$$B_{i-s,r}(x) = B_{ir}(x+sh) \quad \text{nebo-li} \quad B_{ir}(x) = B_{i-s,r}(x-sh).$$

Z (4.57) pomocí transformace $x = x_i + \xi h$, $\xi \in \langle 0, 1\rangle$, obdržíme

$$S_{i+1}^p(x_i + \xi h) = \sum_{j=1}^{r} a_{i+j} B_{i-r+j,r}(x_i + \xi h) = \sum_{j=1}^{r} a_{i+j} B_{j-r,r}(x_i + \xi h - ih) =$$

$$\sum_{j=1}^{r} a_{i+j} B_{j-r,r}(\xi h) = \sum_{j=1}^{r} a_{i+j} \hat{B}_{j-r,r}(\xi), \tag{4.59}$$

kde $\hat{B}_{sr}(t) = B_{sr}(th)$ je kardinální B-splajn řádu r příslušný uzlu s na rovnoměrném dělení s krokem délky jedna. Označíme-li $\hat{\varphi}_j^p(\xi) = \hat{B}_{j-r,r}(\xi)$, $j = 1, 2, \ldots, r$, pak

$$S_{i+1}^p(x) = \sum_{j=1}^{r} a_{i+j} \hat{\varphi}_j^p \left(\frac{x - x_i}{h} \right), \qquad x \in \langle x_i, x_{i+1}). \tag{4.60}$$

Algoritmus BSPLc pro výpočet kardinálních B-splajnů $\{\hat{\varphi}_j^p(\xi)\}_{j=1}^r$ pro $\xi \in \langle 0, 1\rangle$.

1. zvol $r \geq 2$, $\xi \in \langle 0, 1\rangle$
2. vynuluj matici $\mathbf{B} = \{b_{ij}\}$ typu $(r + 1, r)$
3. $b_{r1} := 1$
4. **for** $k := 2, \ldots, r$ **do**
5. **for** $j := r - k + 1, \ldots, r$ **do**
6. $b_{jk} := [((r + \xi) - j)b_{j,k-1} + (j + k - (r + \xi))b_{j+1,k-1}]/(k - 1)$
7. **end**
8. **end**

Pak $\hat{\varphi}_j^p(\xi) = b_{jr}$, $j = 1, \ldots, r$, najdeme v posledním sloupci matice $\mathbf{B}$. Kardinální B-spajny stupňů 1, 2 a 3 jsou uvedeny v tabulce 4.1.

	$p = 1$	$p = 2$	$p = 3$
$\hat{\varphi}_1^p$	$1 - \xi$	$\frac{1}{2}(1 - \xi)^2$	$\frac{1}{6}(1 - \xi)^3$
$\hat{\varphi}_2^p$	ξ	$\frac{1}{2}(-2\xi^2 + 2\xi + 1)$	$\frac{1}{6}(3\xi^3 - 6\xi^2 + 4)$
$\hat{\varphi}_2^p$		$\frac{1}{2}\xi^2$	$\frac{1}{6}(-3\xi^3 + 3\xi^2 + 3\xi + 1)$
$\hat{\varphi}_4^p$			$\frac{1}{6}\xi^3$

Tab. 4.1: Kardinální B-splajny

Koeficienty $\{a_j\}_{j=1}^{n+p}$ dostaneme řešením soustavy n+p lineárních rovnic: $n + 1$ rovnic (4.42) doplníme $p - 1$ rovnicemi pro okrajové podmínky. Matice soustavy je pásová, v každém řádku je nejvýše p nenulových koeficientů. Pro často používané dělení

$$\bar{\Delta} := \{x_{-p} = x_{-p+1} = \cdots = a \equiv x_0 < x_1 \cdots < x_n \equiv b = x_{n+1} = \cdots = x_{n+p}\}, \tag{4.61}$$

lichý stupeň splajnu $p = 2m - 1$ a řazení rovnic pro

Hermitovy okrajové podmínky	přirozené okrajové podmínky
$S(x_0) = y_0,$	$S(x_0) = y_0,$
$S^{(j)}(x_0) = y_0^{(j)}, \quad j = 1 : m-1,$	$S^{(j)}(x_0) = y_0^{(j)}, \quad j = m : 2m-2,$
$S(x_i) = y_i, \quad i = 1 : n-1,$	$S(x_i) = y_i, \quad i = 1 : n-1,$
$S^{(j)}(x_n) = y_n^{(j)}, \quad j = m-1 : -1 : 1,$	$S^{(j)}(x_n) = y_n^{(j)}, \quad j = 2m-2 : -1 : m,$
$S(x_n) = y_n,$	$S(x_n) = y_n,$

je matice soustavy p-diagonální.

Tak třeba pro rovnoměrné dělení (4.61) s krokem délky $h = (b-a)/n$, pro $p = 3$ a Hermitovy okrajové podmínky dostaneme matici soustavy

$$\begin{pmatrix} 1 & 0 & & & & & & & \\ -3/h & 3/h & 0 & & & & & & \\ & 1/4 & 7/12 & 1/6 & & & & & \\ & & 1/6 & 2/3 & 1/6 & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & 1/6 & 2/3 & 1/6 & & \\ & & & & & 1/6 & 7/12 & 1/4 & \\ & & & & & & 0 & -3/h & 3/h \\ & & & & & & & 0 & 1 \end{pmatrix}.$$

4.1.3. Trigonometrická interpolace

je vhodná pro interpolaci periodických funkcí. Řekneme, že funkce f je periodická s *periodou* T, jestliže $f(t+T) = f(t)$ pro každé t. *Frekvence* $\nu = 1/T$ a *úhlová frekvence* $\omega = 2\pi/T$. Tak třeba funkce $\sin 2\pi kt$ má periodu $T = 1/k$, frekvenci $\nu = k$ a úhlovou frekvenci $\omega = 2\pi k$. Jestliže t je čas měřený v sekundách, pak frekvence určuje počet period obsažených v jedné sekundě nebo-li počet *cyklů* za jednu sekundu. Jednotkou frekvence je *Hertz*, značíme Hz, v základních jednotkách Hz=s^{-1}. Představme si, že jednomu cyklu odpovídá úhel 2π, který opíšeme při jedné otáčce po jednotkové kružnici. Úhlová frekvence je pak úhel, který opíšeme za ν cyklů, jednotkou úhlové frekvence je počet radiánů za sekundu, značíme $\mathrm{rad\,s}^{-1}$.

Trigonometrický interpolační polynom. K interpolaci periodické funkce s periodou T použijeme *trigonometrický interpolační polynom*

$$P_n(t) = \begin{cases} \dfrac{a_0}{2} + \displaystyle\sum_{h=1}^{m} (a_h \cos \omega h t + b_h \sin \omega h t) \\ \dfrac{a_0}{2} + \displaystyle\sum_{h=1}^{m-1} (a_h \cos \omega h t + b_h \sin \omega h t) + \dfrac{a_m}{2} \cos \omega m t \end{cases} \quad \text{pro } n = \begin{cases} 2m+1, \\ \\ 2m, \end{cases} \tag{4.62}$$

s koeficienty a_h, b_h zvolenými tak, aby byly splněny interpolační podmínky

$$P_n(t_k) = f_k, \qquad k = 0, 1, \ldots, n-1, \tag{4.63}$$

kde t_k jsou uzly dělení,

$$0 \le t_0 < t_1 < \cdots < t_{n-1} < T,$$

a $f_k = f(t_k)$, $k = 0, 1, \ldots, n-1$. Pro n liché je trigonometrický interpolační polynom určen jednoznačně, viz [20]. Pro sudé n je jednoznačná existence zaručena, právě když $S := \sum_{k=0}^{n-1} t_k$ není celočíselným násobkem T, viz [47]. Dosadíme-li do (4.63) za P výraz (4.62), dostaneme soustavu n lineárních rovnic pro neznámé koeficienty a_h, b_h. Řešení takové soustavy vyžaduje $O(n^3)$ operací.

Trigonometrický interpolační polynom (4.62) lze také vyjádřit ve tvaru

$$P_n(t) = \sum_{h=0}^{n-1} f_h \ell_h(t), \tag{4.64}$$

kde ℓ_h jsou trigonometrické fundamentální polynomy,

$$\ell_h(t) = \begin{cases} \displaystyle\prod_{k=0,\,k\neq h}^{n-1} \frac{\sin\frac{1}{2}\omega(t-t_k)}{\sin\frac{1}{2}\omega(t_h-t_k)} \\[2ex] \displaystyle\frac{\sin\frac{1}{2}\omega(S+t-t_h)}{\sin\frac{1}{2}\omega S} \prod_{k=0,\,k\neq h}^{n-1} \frac{\sin\frac{1}{2}\omega(t-t_k)}{\sin\frac{1}{2}\omega(t_h-t_k)} \end{cases} \quad \text{pro } n = \begin{cases} 2m+1 \text{ liché}, \\[2ex] 2m \text{ sudé}, \end{cases} \tag{4.65}$$

viz [47]. Vzorec (4.65) je sice elegantní, pro praktické použití však příliš vhodný není: výpočet hodnoty $P_n(t)$ podle (4.64)–(4.65) potřebuje $O(n^2)$ operací pro každé t.

Fázový interpolační polynom. Pro efektivní výpočet koeficientů trigonometrického polynomu (4.62) je účelné zvolit rovnoměrné dělení

$$t_k = \frac{T}{n}k, \quad k = 0, 1, \ldots, n-1. \tag{4.66}$$

Na rovnoměrném dělení (4.66) je trigonometrický interpolační polynom jednoznačně určen. Skutečně,

$$\frac{S}{T} = \frac{1}{T}\sum_{k=0}^{n-1} t_k = \frac{1}{T}\sum_{k=0}^{n-1} \frac{T}{n}k = \frac{n(n-1)}{2n} = \frac{n-1}{2},$$

takže pro n sudé S není celočíselným násobkem T.

Je-li f v intervalu $\langle 0, T\rangle$ lipschitzovsky spojitá [3], pak $P_n(t) \to f(t)$ pro $n \to \infty$, $t \in \langle 0, T\rangle$, viz [8].

[3] Funkce $f(t)$ je v intervalu $\langle 0, T\rangle$ lipschitzovsky spojitá, jestliže existuje konstanta L taková, že $|f(x) - f(y)| \le L|x - y|$ $\forall x, y \in \langle 0, T\rangle$.

Koeficienty trigonometrického interpolačního polynomu P_n získáme prostřednictvím *fázového interpolačního polynomu*

$$p_n(t) = c_0 + c_1 e^{i\omega t} + c_2 e^{2i\omega t} + \cdots + c_{n-1} e^{(n-1)i\omega t} = \sum_{h=0}^{n-1} c_h e^{i\omega t h} \equiv \sum_{h=0}^{n-1} c_h z^h, \qquad (4.67)$$

kde $i = \sqrt{-1}$ je komplexní jednotka, $\omega = 2\pi/T$, $e^{i\omega ht} = \cos \omega ht + i \sin \omega ht$ je periodická funkce s periodou $T_h = T/h$ a frekvencí $\nu_h = h/T$, $z = e^{i\omega t}$. Komplexní koeficienty $\{c_h\}_{h=0}^{n-1}$ určíme z interpolačních podmínek

$$p_n(t_k) = f_k, \quad k = 0, 1, \ldots, n-1. \qquad (4.68)$$

Označíme-li

$$\boldsymbol{\varphi}_h = (e^{i\omega h t_0}, e^{i\omega h t_1}, \ldots, e^{i\omega h t_{n-1}})^T = (1, e^{2\pi i h/n}, \ldots, e^{2\pi i (n-1) h/n})^T, \quad h = 0, 1, \ldots, n-1,$$

pak lze interpolační podmínky (4.68) zapsat maticově ve tvaru

$$\mathbf{f} = \sum_{h=0}^{n-1} c_h \boldsymbol{\varphi}_h = \mathbf{V}_n \mathbf{c}, \qquad (4.69)$$

kde

$$\mathbf{V}_n = (\boldsymbol{\varphi}_0, \boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_{n-1}) = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1 & e^{2\pi i/n} & \ldots & e^{2\pi i (n-1)/n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & e^{2\pi i (n-1)/n} & \ldots & e^{2\pi i (n-1)^2/n} \end{pmatrix}, \qquad (4.70)$$

je symetrická Vandermontova matice, $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})^T$ a $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1})^T$. Odtud

$$\mathbf{c} = \mathbf{V}_n^{-1} \mathbf{f} = \frac{1}{n} \mathbf{F}_n \mathbf{f}, \qquad (4.71)$$

kde

$$\mathbf{F}_n = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1 & e^{-2\pi i/n} & \ldots & e^{-2\pi i (n-1)/n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & e^{-2\pi i (n-1)/n} & \ldots & e^{-2\pi i (n-1)^2/n} \end{pmatrix}. \qquad (4.72)$$

Důkaz. Ukážeme, že $\mathbf{V}_n \mathbf{F}_n = n \mathbf{I}_n$, kde $\mathbf{I}_n$ je jednotková matice řádu n.

$$[\mathbf{V}_n \mathbf{F}_n]_{rs} = \sum_{j=0}^{n-1} e^{2\pi i r j/n} e^{-2\pi i j s/n} = \sum_{j=0}^{n-1} e^{2\pi i j (r-s)/n} = \sum_{j=0}^{n-1} q_{rs}^j,$$

kde $q_{rs} = e^{2\pi i (r-s)/n}$. Pro $r = s$ je $q_{rr} = 1$, takže $[\mathbf{V}_n \mathbf{F}_n]_{rr} = n$. Pro $r \neq s$ dostaneme $[\mathbf{V}_n \mathbf{F}_n]_{rs} = [(q_{rs}^n - 1)/(q_{rs} - 1)]/n = 0$, neboť $q_{rs}^n = e^{2\pi i (r-s)} = 1$. □

Všimněte si, že

$$e^{-i\omega(n-h)x_k} = e^{-2\pi i(n-h)k/n} = e^{-2\pi ki}e^{2\pi ihk/n} = e^{2\pi ihk/n} = e^{i\omega hx_k}. \tag{4.73}$$

Odtud a z (4.71) a (4.72) tak dostaneme

$$c_0 = \frac{1}{n}\sum_{k=0}^{n-1} f_k,\ \ c_h = \frac{1}{n}\sum_{k=0}^{n-1} f_k e^{-i\omega ht_k},\ \ c_{n-h} = \frac{1}{n}\sum_{k=0}^{n-1} f_k e^{i\omega ht_k},\ \ h = 1,2,\ldots,n-1. \tag{4.74}$$

Pro $h = 1,2,\ldots,n-1$ jsou tedy koeficienty c_h a c_{n-h} komplexně sdružené, tj. $c_{n-h} = \bar{c}_h$. Koeficient c_0 je reálný a pro n sudé je reálný také koeficient $c_{n/2} = (1/n)\sum_{k=0}^{n-1}(-1)^k f_k$.

Výpočet hodnoty $p_n(t)$ lze provést pomocí Hornerova schématu aplikovaného na polynom $q_n(z) = \sum_{h=0}^{n-1} c_h z^h$ pro $z = e^{i\omega t}$.

Koeficienty trigonometrického interpolačního polynomu určíme z koeficientů fázového interpolačního polynomu. Pomocí (4.73) vyjádříme

$$\cos\omega ht_k = \frac{e^{i\omega ht_k} + e^{i\omega(n-h)t_k}}{2}, \qquad \sin\omega ht_k = \frac{e^{i\omega ht_k} - e^{i\omega(n-h)t_k}}{2i}$$

a dosadíme do rovnic $p_n(t_k) = P_n(t_k)$, $k = 0,1,\ldots,n-1$.

Pro $n = 2m+1$ po úpravě dostaneme

$$\left(c_0 - \frac{a_0}{2}\right)\boldsymbol{\varphi}_0 + \sum_{h=1}^{m}\left[\left(c_h - \frac{a_h - ib_h}{2}\right)\boldsymbol{\varphi}_h + \left(c_{n-h} - \frac{a_h + ib_h}{2}\right)\boldsymbol{\varphi}_{n-h}\right] = \mathbf{o}.$$

Protože vektory $\{\boldsymbol{\varphi}_h\}_{h=0}^{n-1}$ jsou lineárně nezávislé (neboť $\mathbf{V}_n = (\boldsymbol{\varphi}_0, \boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_{n-1})$ je regulární), koeficienty u $\boldsymbol{\varphi}_h$ musejí být rovny nule. Odtud

$$a_0 = 2c_0, \quad a_h = c_h + c_{n-h},\ \ b_h = i(c_h - c_{n-h}),\ \ h = 1,2,\ldots,m.$$

Protože $c_{n-h} = \bar{c}_h$, dostaneme

$$a_h = 2\mathrm{Re}(c_h), \quad b_h = -2\mathrm{Im}(c_h), \tag{4.75}$$

kde $\mathrm{Re}(c_h)$ resp. $\mathrm{Im}(c_h)$ je reálná resp. imaginární složka c_h.

Pro $n = 2m$ podobně odvodíme

$$a_0 = 2c_0, \quad a_h = c_h + c_{n-h},\ \ b_h = i(c_h - c_{n-h}),\ \ h = 1,2,\ldots,m-1, \quad a_m = 2c_m,$$

přičemž opět platí (4.75).

K určení koeficientů c_h, a tedy také a_h, b_h, jsme potřebovali $O(n^2)$ operací. V dalším ukážeme, že tento počet lze snížit na $O(n\log n)$ operací.

Rychlá Fourierova transformace, stručně FFT (podle anglického Fast Fourier Transform) je algoritmus, který umožňuje efektivní výpočet *diskrétní Fourierovy transformace*, stručně DFT (podle anglického Discrete Fourier Transform). DFT transformuje posloupnost čísel $\{y_k\}_{k=0}^{n-1}$ na posloupnost $\{Y_k\}_{k=0}^{n-1}$ předpisem

$$Y_k = \sum_{j=0}^{n-1} e^{-2\pi ijk/n} y_j = \sum_{j=0}^{n-1} w_n^{jk} y_j, \quad k = 0,1,\ldots,n-1.$$

kde $w_n = e^{-2\pi i/n}$. Odpovídající maticový zápis je

$$\mathbf{Y} = \mathbf{F}_n \mathbf{y},$$

kde $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})^T$, $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_{n-1})^T$ a $\mathbf{F}_n = \{w_n^{jk}\}_{j,k=0}^{n-1}$, viz (4.72). Efektivní výpočet koeficientů $\mathbf{c} = (1/n)\mathbf{F}_n\mathbf{f}$ fázového interpolačního polynomu lze proto provést pomocí rychlé Fourierovy transformace.

Algoritmus FFT je založen na strategii známé jako *rozděl a panuj* (anglicky *divide and conquer*). Podstatu FFT lze nejsnadněji vysvětlit pro případ, kdy $n = 2^q$. Pomocí rovnosti

$$w_n = e^{-2\pi i/n} = [e^{-2\pi i/(2n)}]^2 = w_{2n}^2$$

dostaneme

$$Y_k = \sum_{j=0}^{n-1} w_n^{jk} y_j = \sum_{j=0}^{n/2-1} w_n^{2jk} y_{2j} + \sum_{j=0}^{n/2-1} w_n^{(2j+1)k} y_{2j+1} = \qquad k = 0, 1, \ldots, n-1.$$
$$\sum_{j=0}^{n/2-1} w_{n/2}^{jk} y_{2j} + w_n^k \sum_{j=0}^{n/2-1} w_{n/2}^{jk} y_{2j+1}.$$

Pro $k = n/2 + s$, $s = 0, 1, \ldots, n/2 - 1$, $j = 0, 1, \ldots, n/2 - 1$, odvodíme

$$w_{n/2}^{j(n/2+s)} = \exp\left(-\frac{2\pi ij}{n/2}(n/2+s)\right) = \exp(-2\pi ij) \cdot \exp\left(-\frac{2\pi ijs}{n/2}\right) = w_{n/2}^{js},$$

$$w_n^{n/2+s} = \exp\left(-\frac{2\pi i}{n}(n/2+s))\right) = \exp(-\pi i) \cdot \exp(-2\pi is/n) = -w_n^s,$$

takže

$$\begin{aligned} Y_k &= \sum_{j=0}^{n/2-1} w_{n/2}^{jk} y_{2j} + w_n^k \sum_{j=0}^{n/2-1} w_{n/2}^{jk} y_{2j+1}, \\ Y_{n/2+k} &= \sum_{j=0}^{n/2-1} w_{n/2}^{jk} y_{2j} - w_n^k \sum_{j=0}^{n/2-1} w_{n/2}^{jk} y_{2j+1}, \end{aligned} \qquad k = 0, 1, \ldots, n/2 - 1.$$

Transformaci vektoru $\mathbf{y}$ délky n provedeme pomocí dvou transformací

$$\mathbf{u} = \mathbf{F}_{n/2}\mathbf{p}, \qquad \mathbf{v} = \mathbf{F}_{n/2}\mathbf{q}$$

vektorů $\mathbf{p} = (y_0, y_2, \ldots, y_{n-2})^T$ a $\mathbf{q} = (y_1, y_3, \ldots, y_{n-1})^T$ poloviční délky $n/2$. Pak

$$\mathbf{Y} = \begin{pmatrix} \mathbf{u} + \mathbf{d} \circ \mathbf{v} \\ \mathbf{u} - \mathbf{d} \circ \mathbf{v} \end{pmatrix},$$

kde $\mathbf{d} = (1, w_n, \ldots, w_n^{n/2-1})$ a symbol $\circ$ vyznačuje Hadamardův součin, tj.

$$\mathbf{d} \circ \mathbf{u} = (d_1 u_1, d_2 u_2, \ldots, d_{n/2} u_{n/2})^T, \qquad \mathbf{d} \circ \mathbf{v} = (d_1 v_1, d_2 v_2, \ldots, d_{n/2} v_{n/2})^T.$$

Tento postup rekurzivně opakujeme. Každý krok vyžaduje $O(n)$ operací, kroků je $q = \log_2 n$, takže celkové výpočetní náklady jsou řádu $O(n \log n)$. Následuje

Algoritmus FFT v MATLABu:

```
function y=F(x)
% x je sloupcový vektor
n=length(x);
if n==1, y=x; return; end
d=exp(-2*pi*1i/n).^((0:n/2-1)');
u=F(x(1:2:n-1));
v=F(x(2:2:n));
y=[u+d.*v;u-d.*v];
```

Algoritmus bez použití rekurzivního volání funkce je uveden třeba v [55]. Rychlý výpočet DFT založený na principu rozděl a panuj lze odvodit i pro případy, kdy n není mocninou dvojky. Algoritmus FFT pro libovolné n je implementován v MATLABu jako funkce `fft`. Koeficienty fázového interpolačního polynomu dostaneme provedením příkazu `c=fft(f)/n`.

Amplituda, fáze. Trigonometrický interpolační polynom (4.62) můžeme zapsat ekvivalentně ve tvaru

$$P_n(t) = \begin{cases} \dfrac{a_0}{2} + \displaystyle\sum_{h=1}^{m} A_h \sin(\omega h t + \phi_h) \\ \dfrac{a_0}{2} + \displaystyle\sum_{h=1}^{m-1} A_h \sin(\omega h t + \phi_h) + \dfrac{a_m}{2}\cos(\omega m t) \end{cases} \quad \text{pro } n = \begin{cases} 2m+1, \\ 2m, \end{cases} \tag{4.76}$$

kde $A_h = \sqrt{a_h^2 + b_h^2} = 2|c_h|$ je *amplituda* a $\phi_h = \operatorname{arctg2}(a_h, b_h)$ je *fáze* definovaná rovnicemi $\cos\phi_h = b_h/A_h$, $\sin\phi_h = a_h/A_h$. Skutečně,

$$a_h \cos\omega h t + b_h \sin\omega h t = A_h \left(\frac{a_h}{A_h}\cos\omega h t + \frac{b_h}{A_h}\sin\omega h t \right) =$$

$$A_h \left(\sin\phi_h \cos\omega h t + \cos\phi_h \sin\omega h t\right) = A_h \sin(\omega h t + \phi_h).$$

Funkce arctg2 je v MATLABu implementována jako funkce `atan2`. Označíme-li

$$A_0 = \frac{|a_0|}{2} = |c_0|,\ \phi_0 = \frac{\pi}{2}\operatorname{sgn}(a_0),\ \text{a pro } n \text{ sudé } A_m = \frac{|a_m|}{2} = |c_m|,\ \phi_m = \frac{\pi}{2}\operatorname{sgn}(a_m),$$

dostaneme

$$P_n(t) = \sum_{h=0}^{m} A_h \sin(\omega h t + \phi_h). \tag{4.77}$$

Funkce sgn je definována předpisem

$$\operatorname{sgn}(t) = \begin{cases} -1 & & t < 0, \\ 0 & \text{když} & t = 0, \\ 1 & & t > 0, \end{cases}$$

v MATLABu viz funkce `sign`.

V praktických aplikacích se často vyhledává tzv. *dominantní frekvence* $\nu_d = h_d/T$ příslušná maximální amplitudě $A_d = \max_{0<h<n} A_h$.

Příklad 4.7. Uvažujme zařízení, které generuje periodický signál popsaný funkcí

$$f(t) = 10 + 12\cos 100\pi t + 8\sin 160\pi t - 10\cos 300\pi t.$$

Signál f se skládá z konstanty $\bar{A}_0 = 10$ a ze tří periodických funkcí: funkce $12\cos 100\pi$ má amplitudu $\bar{A}_1 = 12$ a frekvenci $\bar{\nu}_1 = 50$, funkce $8\sin 160\pi t$ má amplitudu $\bar{A}_2 = 8$ a frekvenci $\bar{\nu}_1 = 80$ a funkce $-10\cos 300\pi t$ má amplitudu $\bar{A}_3 = 10$ a frekvenci $\bar{\nu}_1 = 150$.

Příjemce signálu funkci f nezná, může však hodnoty signálu měřit. Provede celkem n měření v časech $t_k = k\Delta t$, $k = 0, 1, \ldots, n-1$, kde $\Delta t = T/n$ a T je předpokládaná perioda signálu. Hodnotu změřenou v čase t_k označme jako f_k, $k = 0, 1, \ldots, n-1$.

Snadno zjistíme, že f má periodu $T = 0{,}1\ell$, kde ℓ je libovolné přirozené číslo. Předpokládejme nejdříve, že periodu známe a zvolíme $T = 2$. Jestliže provedeme 50 měření na každé základní periodě délky 0,1, dostaneme $n = 1000$. Výpočet amplitud dokumentuje

Program `FREQ` v MATLABu:

```
f=@(t) 10+12*cos(100*pi*t)+8*sin(160*pi*t)-10*cos(300*pi*t); % funkce
T=2; % perioda
n=1000; % počet měření
m=n/2; % polovina n
t=(0:n-1)*T/n; % dělení na časové ose
y=f(t); % hodnoty funkce
c=fft(y)/n; % koeficienty fázového polynomu
A=[1,2*ones(1,m-1),1].*abs(c(1:m+1)); % amplitudy
nu=(0:n-1)/T; % dělení na frekvenční ose
stem(nu(1:m+1),A,'k.','linewidth',1); % zobrazení amplitud
xlabel('frekvence'); ylabel('amplitudy'); % popisy os
set(gca,'xtick',[0 50 80 150 250]); % zarážky na ose x
axis([-15 265 -1 13]); grid on; % rozměry okna
```

Grafickým výstupem programu je obrázek 4.4. Vidíme, že se podařilo přesně identifikovat frekvence a amplitudy obsažené v signálu f. Výpočet fází a jejich vykreslení ponecháváme jako cvičení, vyjde $\bar{\phi}_0 = \pi/2$, $\bar{\phi}_1 = \pi/2$, $\bar{\phi}_2 = 0$ a $\bar{\phi}_3 = -\pi/2$.

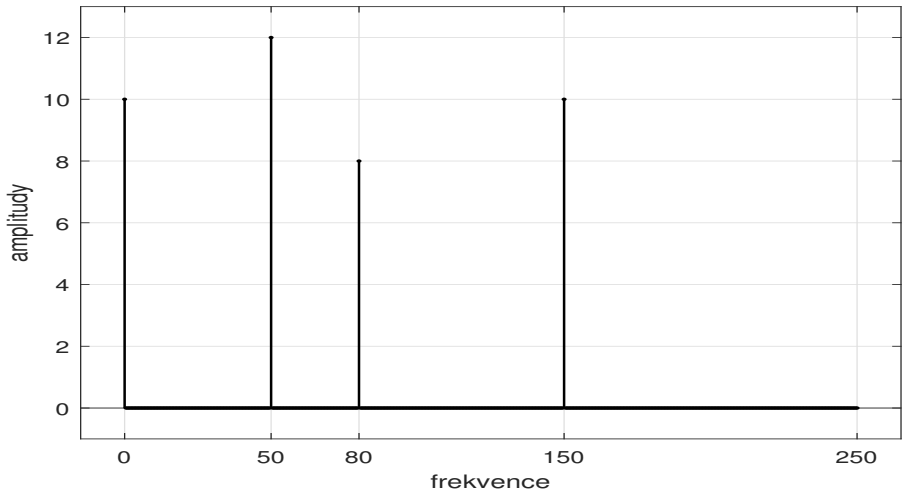
V praktických aplikacích často periodu funkce f neznáme. Frekvence a amplitudy signálu f lze přesto přibližně zjistit. Jestliže v programu `FREQ` zvolíme `T=2.045`, dostaneme obrázek 4.5. Aproximace frekvencí je skvělá, u amplitud je to horší.

Pokud je signál rušen náhodným šumem s nulovou střední hodnotou, frekvence a amplitudy signálu f lze opět celkem dobře odhadnout. Když v programu `FREQ` nahradíme příkaz `y=f(t)` příkazem

```
y=f(t)+6*(2*randn(1,n)-1);  % porušené hodnoty funkce f
```

dostaneme obrázek 4.6. Frekvence porušeného signálu zůstaly prakticky stejné.

Aproximace Fourierových koeficientů. Každou lipschitzovsky spojitou periodickou



Obr. 4.4: amplitudy: T=2.

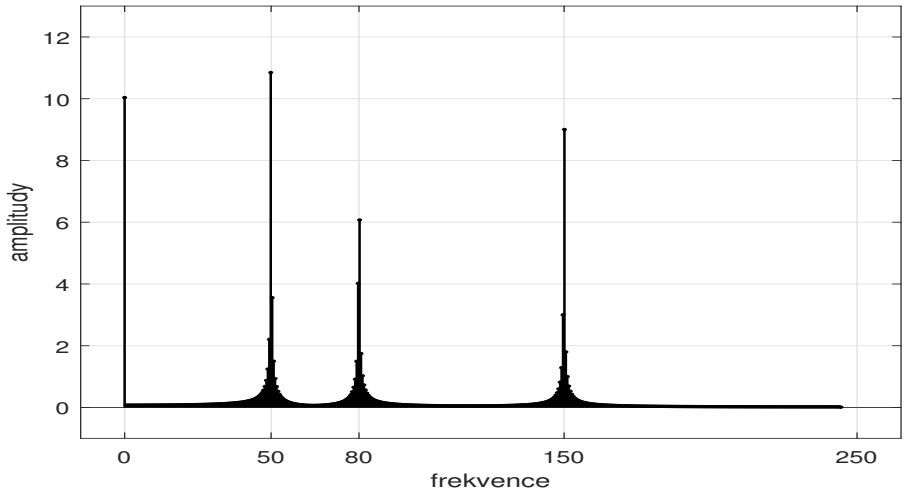
funkci f s periodou T lze vyjádřit pomocí *Fourierovy řady*

$$f(t) = \frac{\alpha_0}{2} + \sum_{h=1}^{\infty} [\alpha_h \cos \omega h t + \beta_h \sin \omega h t],$$

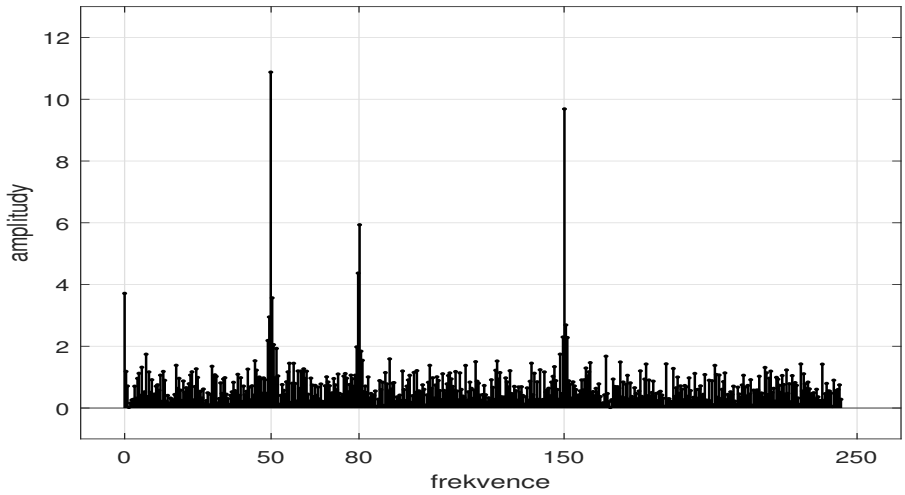
kde $\omega = 2\pi/T$ a

$$\alpha_h = \frac{2}{T} \int_0^T f(t) \cos \omega h t \, \mathrm{d}t, \quad \beta_h = \frac{2}{T} \int_0^T f(t) \sin \omega h t \, \mathrm{d}t$$

jsou *Fourierovy koeficiety.*



Obr. 4.5: amplitudy: T=2,045.



Obr. 4.6: amplitudy: T=2,045, porušená data.

Pomocí (4.74) a (4.75) dostaneme

$$a_h = \frac{2}{n}\sum_{k=0}^{n-1} f_k \cos \omega h t_k, \quad b_h = \frac{2}{n}\sum_{k=0}^{n-1} f_k \sin \omega h t_k.$$

Protože $f(t_n) = f(t_0)$, pro $f_n = f(t_n)$ platí $f_n = f_0$. Označíme-li $\Delta t = T/n$, pak $t_k = k\Delta t$, $k = 0, 1, \ldots, n$. Odtud

$$a_h = \frac{2}{T}\Delta t \left[\frac{1}{2} f_0 \cos \omega h t_0 + \sum_{k=1}^{n-1} f_k \cos \omega h t_k + \frac{1}{2} f_n \cos \omega h t_n\right] = \frac{2}{T} Q_T^n (f \cos \omega h t),$$

kde Q_T^n je složená lichoběžníková formule na n dílcích. Podobně $b_h = 2/T\, Q_T^n (f \sin \omega h t)$.

Vidíme tedy, že koeficienty $\{a_h\}_{h=0}^m$ resp. $\{b_h\}_{h=1}^m$ trigonometrického interpolačního polynomu P_{2m+1} jsou aproximace Fourierových koeficientů $\{\alpha_h\}_{h=0}^m$ resp. $\{\beta_h\}_{h=1}^m$ založené na přibližném výpočtu příslušných integrálů složenou lichoběžníkovou formulí. Algoritmus přibližného výpočtu Fourierových koeficientů α_h, β_h pro $h > m$ je popsán v [55].

4.1.4. 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 $T_k \in \mathfrak{T}$ 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

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

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 $R_{ij} = \{(x,y)\,|\,x_{i-1} \le x \le x_i, y_{j-1} \le y \le y_j\}$, $i = 1,2,\ldots,n$, $j = 1,2,\ldots,m$. 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

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

a C je konstanta nezávislá na h, k.

4.2. Metoda nejmenších čtverců

označuje postup pro přibližné řešení přeurčených nebo nepřesně zadaných soustav rovnic, založený na minimalizaci kvadrátů jejich reziduí, viz kapitola 3.

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 $R_n(t_i)$:

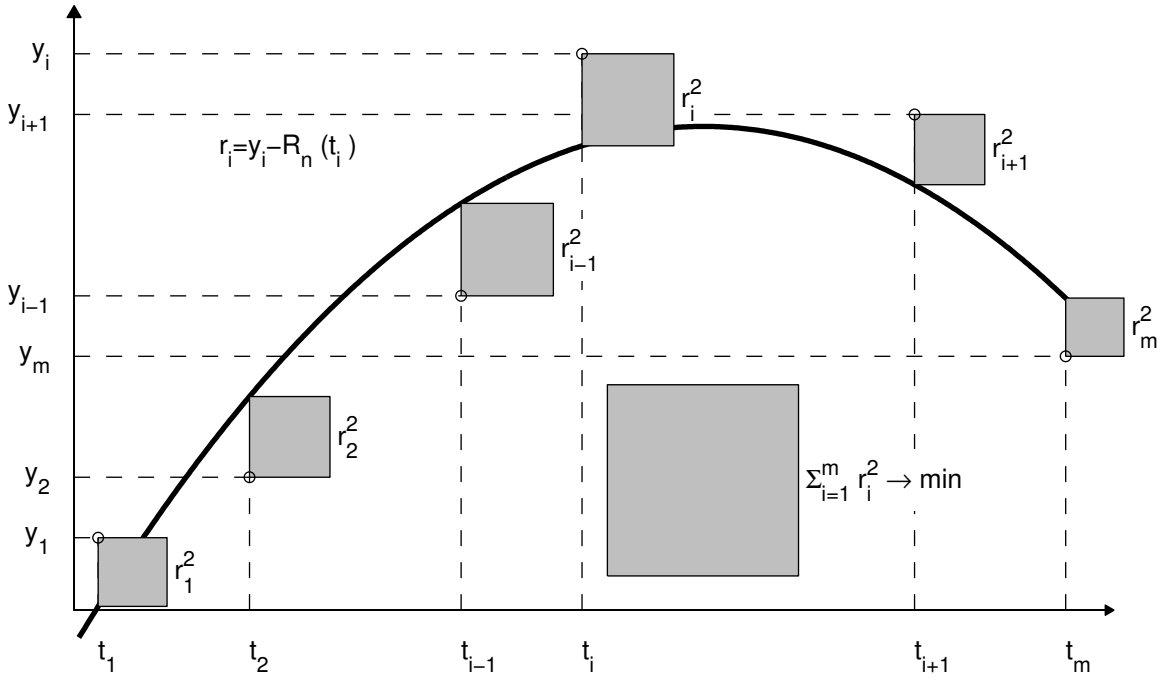
$$r_i = y_i - R_n(t_i) = y_i - \sum_{j=1}^{n} \varphi_j(t_i)x_j \equiv y_i - \sum_{j=1}^{n} a_{ij}x_j\,, \qquad i = 1, 2, \ldots, m\,,$$

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

$$\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x}\,. \tag{4.78}$$

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

$$\|\mathbf{r}\|_2^2 = \sum_{i=1}^{m} r_i^2 \to \min\,. \tag{4.79}$$



Obr. 4.7: Princip metody nejmenších čtverců

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

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

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 (4.79) 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

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y}\,. \tag{4.80}$$

Soustava lineárních rovnic (4.80) 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 (4.79):

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

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

$$\begin{pmatrix} (\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_1) & (\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2) & \dots & (\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_n) \\ (\boldsymbol{\varphi}_2,\boldsymbol{\varphi}_1) & (\boldsymbol{\varphi}_2,\boldsymbol{\varphi}_2) & \dots & (\boldsymbol{\varphi}_2,\boldsymbol{\varphi}_n) \\ \vdots & \vdots & \dots & \vdots \\ (\boldsymbol{\varphi}_n,\boldsymbol{\varphi}_1) & (\boldsymbol{\varphi}_n,\boldsymbol{\varphi}_2) & & (\boldsymbol{\varphi}_n,\boldsymbol{\varphi}_n) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} (\boldsymbol{\varphi}_1,\mathbf{y}) \\ (\boldsymbol{\varphi}_2,\mathbf{y}) \\ \vdots \\ (\boldsymbol{\varphi}_n,\mathbf{y}) \end{pmatrix}, \tag{4.81}$$

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 (4.81) 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 (4.79) 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 [6]):

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 4.8. 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 $R_2(t) = x_1 + x_2 e^{-t}$ metodou nejmenších čtverců. Zřejmě $\varphi_1(t) = 1$ a $\varphi_2(t) = e^{-t}$. Normální soustava rovnic je tvaru

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

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 4.9. 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.

Splajny v metodě nejmenších čtverců. Regresní funkci hledáme jako splajn

$$S_p(t) = \sum_{j=1}^{d} x_j \varphi_j^p(t) \tag{4.82}$$

stupně p, kde $\varphi_j^p = B_{j-r,r}$, $r = p + 1$, $d = n + p \leq m$. Koeficienty x_j, $j = 1, 2, \ldots, d$, určíme minimalizací funkce

$$F(\mathbf{x}) = \sum_{i=1}^{m} \left(y_i - S_p(t_i)\right)^2 = \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n+r-1} x_j B_{j-r,r}(t_i) \right)^2, \tag{4.83}$$

kde $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$. Označíme-li

$$\mathbf{A} = \begin{pmatrix} B_{1-r,r}(t_1) & B_{2-r,r}(t_1) & \ldots & B_{n-1,r}(t_1) \\ B_{1-r,r}(t_2) & B_{2-r,r}(t_2) & \ldots & B_{n-1,r}(t_2) \\ \vdots & \vdots & \ldots & \vdots \\ B_{1-r,r}(t_m) & B_{2-r,r}(t_m) & \ldots & B_{n-1,r}(t_m) \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix},$$

pak lze funkci F zapsat maticově ve tvaru

$$F(\mathbf{x}) = (\mathbf{y} - \mathbf{A}\mathbf{x})^T (\mathbf{y} - \mathbf{A}\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2. \tag{4.84}$$

Z podmínky $\nabla F = \mathbf{o}$ dostaneme normální soustavu rovnic (4.80).

Vyhlazovací bázový splajn. Splajn s koeficienty získanými řešením normální soustavy rovnic občas vykazuje nežádoucí zvlnění. Z toho důvodu se zavádí tzv. *vyhlazovací splajn*, jehož koeficienty se získají minimalizací funkce

$$G(\mathbf{x}) = \sum_{i=1}^{m} \left(y_i - S_p(t_i)\right)^2 + p \int_a^b \left[S_p''(t)\right]^2 \, \mathrm{d}t, \tag{4.85}$$

kde $p \geq 0$ je vhodně zvolený parametr vyhlazení. Pro vyhlazovací člen postupně dostaneme

$$\int_a^b \left[S_p''(t)\right]^2 \,\mathrm{d}t = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} \left[S_p''(t)\right]^2 \,\mathrm{d}t =$$

$$\sum_{i=0}^{n-1} \sum_{\alpha,\beta=1}^{r} x_{i+\alpha} \left[\int_{x_i}^{x_{i+1}} B''_{i-r+\alpha,r}(t) B''_{i-r+\beta,r}(t) \,\mathrm{d}t\right] x_{i+\beta} = \sum_{i=0}^{n-1} \mathbf{x}_i^T \mathbf{D}_i \mathbf{x}_i,$$

kde $\mathbf{x}_i = (x_{i+1}, x_{i+2}, \ldots, x_{i+r})^T$ a $\mathbf{D}_i = \{d^i_{\alpha\beta}\}_{\alpha,\beta=1}^r$, přičemž

$$d^i_{\alpha\beta} = \int_{x_i}^{x_{i+1}} B''_{i-r+\alpha,r}(t) B''_{i-r+\beta,r}(t) \,\mathrm{d}t, \quad \alpha, \beta = 1, 2, \ldots, r.$$

Integrály lze spočítat numericky formulí řádu alespoň $2(r-3)$, třeba Gaussovou-Legendrovou formulí s $(r-2)$-ma uzly. Pomocí lokálních matic $\mathbf{D}_i$, $i = 0, 1, \ldots, n-1$, sestavíme globální matici $\mathbf{D}$ na základě ekvivalence

$$\sum_{i=0}^{n-1} \mathbf{x}_i^T \mathbf{D}_i \mathbf{x}_i = \mathbf{x}^T \mathbf{D} \mathbf{x}.$$

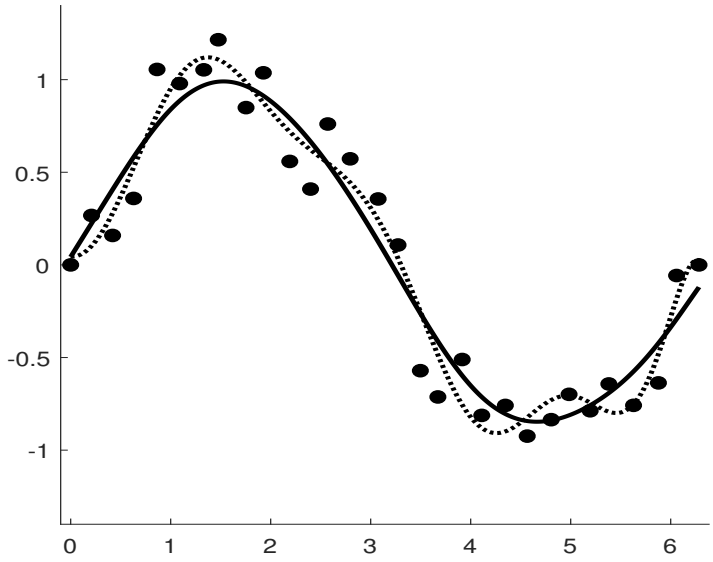
Celkem tedy

$$G(\mathbf{x}) = (\mathbf{y} - \mathbf{A}\mathbf{x})^T (\mathbf{y} - \mathbf{A}\mathbf{x}) + p \mathbf{x}^T \mathbf{D} \mathbf{x}, \tag{4.86}$$

takže vyhlazené koeficienty dostaneme jako řešení soustavy lineárních rovnic

$$\left(\mathbf{A}^T\mathbf{A} + p\mathbf{D}\right) \mathbf{x} = \mathbf{A}^T \mathbf{y}. \tag{4.87}$$

Ilustrace efektu vyhlazení. Zvolili jsme řád splajnu $r = 6$ a dělení intervalu $\langle 0, 2\pi \rangle$ na $n = 10$ dílků nestejné délky. V rozšířeném dělení $\bar{\Delta}$ jsme zvolili $x_{-5} = x_{-4} = \cdots = x_0 = 0$, $x_{10} = x_{11} = \cdots = x_{15} = 2\pi$. Dále jsme zvolili $m = 30$ bodů $[t_i, y_i]$, $i = 1, 2, \ldots, 30$, kde $0 = t_1 < t_2 < \cdots < t_m = 2\pi$ je nerovnoměrné dělení intervalu $\langle 0, 2\pi \rangle$ a kde $y_i \approx \sin t_i$ jsou hodnoty $\sin t_i$ porušené pomocí generátoru náhodných čísel s rovnoměrným rozdělením. V obrázku 4.8 tečkovaná čára vyznačuje nevyhlazený splajn a plná čára splajn vyhlazený s koeficientem vyhlazení $p = 0{,}2$.



Obr. 4.8: Vyhlazovací splajn: $r = 6$, $n = 10$, $m = 30$, $p = 0{,}2$.

5. 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.

5.1. 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

$$f'(x_s) - P_n'(x_s) = \frac{f^{(n+1)}(\xi_s)}{(n+1)!}\omega_{n+1}'(x_s)\,, \tag{5.1}$$

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 (5.1) tak dostaneme:

$$n = 1: \qquad f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(\xi)\,, \tag{5.2a}$$

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{1}{2}hf''(\xi)\,, \tag{5.2b}$$

$$n = 2: \qquad f'(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h} + \frac{1}{3}h^2 f'''(\xi)\,, \tag{5.3a}$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6}h^2 f'''(\xi)\,, \tag{5.3b}$$

$$f'(x) = \frac{3f(x) - 4f(x-h) + f(x-2h)}{2h} + \frac{1}{3}h^2 f'''(\xi)\,. \tag{5.3c}$$

Uveďme ještě nejznámější formuli pro výpočet druhé derivace. Rovnost

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}h^2 f^{(4)}(\xi)\,. \tag{5.4}$$

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

Formule ze vzorce (5.2a) je známa jako *první diference vpřed (dopředná diference)* a formule ze vzorce (5.2b) jako *první diference vzad (zpětná diference)*. Formule ze vzorce (5.3b) bývá označována jako *první centrální diference* a formule ze vzorce (5.4) 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 (5.2a) 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 (5.2) – (5.4) 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 (5.2a).

Ve skutečnosti za přibližnou hodnotu derivace $f'(x_0)$ považujeme výraz

$$\tilde{f}'(x_0) := \frac{\tilde{f}(x_0+h) - \tilde{f}(x_0)}{h} = \frac{[f(x_0+h)+\varepsilon_1] - [f(x_0)+\varepsilon_0]}{h} = f'(x_0) + \frac{1}{2}hf''(\xi_0) + \frac{\varepsilon_1 - \varepsilon_0}{h}\,,$$

kde ε_1 resp. ε_0 je zaokrouhlovací chyba, které se dopustíme při výpočtu $f(x_0 + h)$ resp. $f(x_0)$. Tedy

$$f'(x_0) = \frac{\tilde{f}(x_0 + h) - \tilde{f}(x_0)}{h} + E_d + E_r\,,$$

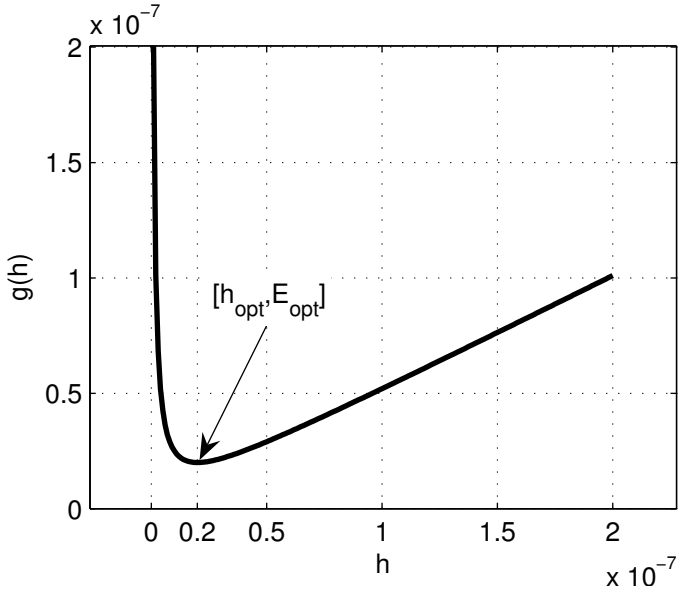
kde $E_d := -\frac{1}{2}hf''(\xi)$ je *diskretizační chyba* a $E_r := -(\varepsilon_1 - \varepsilon_0)/h$ je *chyba zaokrouhlovací*. Chování obou chyb je pro $h \to 0$ diametrálně odlišné: zatímco $|E_d| \to 0$, $|E_r| \to \infty$. Pro malá h se tedy zřejmě jedná o špatně podmíněnou úlohu: malé změny $\varepsilon_0, \varepsilon_1$ 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 $[\tilde{f}(x_0 + h) - \tilde{f}(x_0)]/h$, 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 $M_2 = \max|f''(x)|$ pro $x \in \langle x_0, x_0+h\rangle$ a $\varepsilon = \max(|\varepsilon_1|, |\varepsilon_2)|)$. 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 $f(x_0)$ i $f(x_0+h)$ 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. 5.1: 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. 5.1.

5.2. 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

$$F(h) = a_0 + a_1 h^2 + a_2 h^4 + a_3 h^6 + \dots, \quad h \in (0, h_0\rangle. \tag{5.5}$$

Konkrétní formule F je zkoumána v příkladu 5.1, viz (5.17). 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 (5.5) platí

$$F\left(\frac{h}{2}\right) = a_0 + a_1 \left(\frac{h}{2}\right)^2 + a_2 \left(\frac{h}{2}\right)^4 + a_3 \left(\frac{h}{2}\right)^6 + \dots \tag{5.6}$$

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 (5.6) odečteme rovnici (5.5) a výsledek dělíme třemi. Tak dostaneme

$$F_2(h) := \frac{4F(\frac{h}{2}) - F(h)}{3} = a_0 + a_2^{(2)} h^4 + a_3^{(2)} h^6 + \dots \tag{5.7}$$

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 (5.7) platí

$$F_2\left(\frac{h}{2}\right) = a_0 + a_2^{(2)} \left(\frac{h}{2}\right)^4 + a_3^{(2)} \left(\frac{h}{2}\right)^6 + \dots \tag{5.8}$$

Rovnici (5.8) násobíme 16, odečteme (5.7) a výsledek dělíme 15. Tak dostaneme metodu F_3, která je pro zvolené h definována předpisem

$$F_3(h) := \frac{16F_2(\frac{h}{2}) - F_2(h)}{15} = a_0 + a_3^{(3)} h^6 + \dots \tag{5.9}$$

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

$$F_{i+1}(h) = \frac{4^i F_i(\frac{h}{2}) - F_i(h)}{4^i - 1} = a_0 + a_{i+1}^{(i+1)} h^{2i+2} + \dots, \qquad i = 1, 2, \dots \tag{5.10}$$

a kde $F_1(h) = F(h)$. Pro koeficienty rozvoje přitom platí $|a_j^{(i+1)}| < |a_j|$, $j = i+1, i+2, \dots$.

Nechť $h_s = h/2^s$, $s = 0, 1, \dots$. Protože $F_i(h) = a_0 + a_i^{(i)} h^{2i} + a_{i+1}^{(i)} h^{2i+2} + \dots$, snadno odvodíme, že

$$|F_i(h_s) - a_0| \le C_i h_s^{2i}, \qquad s = 0, 1, \dots, \quad i = 0, 1, \dots, \tag{5.11}$$

kde C_i jsou konstanty nezávislé na h_s. Řekneme proto, že $F_i(h_s)$ je aproximace $F(0)$ řádu h_s^{2i} a píšeme $F_i(h_s) = F(0) + O(h_s^{2i})$.

Poznámka. (*O Landauově symbolu* $O(\varphi(h))$). Nechť $\varphi(h)$ je funkce definovaná v intervalu $(0, h_0\rangle$ a p je kladné číslo. Řekneme, že funkce $\varphi(h)$ je řádu $O(h^p)$ a píšeme $\varphi(h) = O(h^p)$, jestliže existuje číslo $C > 0$ takové, že pro všechna $0 < h \le h_0$ platí $|\varphi(h)| \le Ch^p$. □

Výpočet lze přehledně uspořádat do tabulky

$$\begin{array}{lllll}
F_1(h) & & & & \\
F_1\left(\frac{h}{2}\right) & F_2(h) & & & \\
F_1\left(\frac{h}{4}\right) & F_2\left(\frac{h}{2}\right) & F_3(h) & & \\
F_1\left(\frac{h}{8}\right) & F_2\left(\frac{h}{4}\right) & F_3\left(\frac{h}{2}\right) & F_4(h) & \\
F_1\left(\frac{h}{16}\right) & F_2\left(\frac{h}{8}\right) & F_3\left(\frac{h}{4}\right) & F_4\left(\frac{h}{2}\right) & F_5(h) \\
\vdots & \vdots & \vdots & \vdots & \vdots \quad \ddots
\end{array}
\quad \Longleftrightarrow \quad
\begin{array}{lllll}
T_{00} & & & & \\
T_{10} & T_{11} & & & \\
T_{20} & T_{21} & T_{22} & & \\
T_{30} & T_{31} & T_{32} & T_{33} & \\
T_{40} & T_{41} & T_{42} & T_{43} & T_{44} \\
\vdots & \vdots & \vdots & \vdots & \vdots \quad \ddots
\end{array}$$

Tab. 5.1 Richardsonova extrapolace

Tabulku vyplňujeme po řádcích. Zřejmě

$$T_{si} = F_{i+1}(h/2^{s-i}), \qquad F_i(h_s) = T_{i+s-1,i-1}. \tag{5.12}$$

Prvek T_{s0} v prvním sloupci tabulky vypočteme pomocí základní metody $F = F_1$,

$$T_{s0} = F(h/2^s), \qquad s = 0, 1, \dots, \tag{5.13}$$

a další prvky v tomto řádku počítáme ve shodě s (5.10) podle předpisu

$$T_{si} := \frac{4^i T_{s,i-1} - T_{s-1,i-1}}{4^i - 1} = T_{s,i-1} + \frac{T_{s,i-1} - T_{s-1,i-1}}{4^i - 1}, \quad i = 1, 2, \dots, s. \tag{5.14}$$

Výpočet ukončíme a T_{si} považujeme za dostatečně přesnou aproximaci $F(0)$, pokud

$$|T_{si} - T_{s,i-1}| < \max(\varepsilon_r |T_{si}|, \varepsilon_a), \tag{5.15}$$

kde ε_r je požadovaná relativní přesnost a ε_a požadovaná přesnost absolutní. Podle (5.11) a (5.12) přitom

$$|T_{si} - F(0)| \le C_{i+1} h_{s-i}^{2i+2}, \qquad s = 0, 1, \dots, \quad i = 0, 1, \dots \tag{5.16}$$

Příklad 5.1. Richardsonovu extrapolaci použijeme pro zpřesnění výpočtu derivace podle formule (5.3b). Jestliže má funkce f dostatečný počet spojitých derivací, pak

$$F(h) := \frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{f^{(3)}(x)}{3!}h^2 + \frac{f^{(5)}(x)}{5!}h^4 + \dots, \tag{5.17}$$

takže $F(h)$ je tvaru (5.5).

Počítejme derivaci funkce $f(x) = \cos x$ pro $x = 1$. Zvolíme např. $h = 0{,}8$ a výpočet ukončíme, když bude splněna podmínka (5.15) 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 (5.14). 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	**−0,841471**

cifry platné. □

Poznámka. Richardsonovu extrapolaci lze aplikovat na základní metodu F také v případě, když má funkce $F(h)$ obecný rozvoj

$$F(h) = a_0 + a_1 h^{p_1} + a_2 h^{p_2} + a_3 h^{p_3} + \dots \tag{5.5'}$$

kde $1 \le p_1 < p_2 < p_3 < \dots$ jsou přirozená čísla. Přesnější metodu F_{i+1} v tom případě definujeme předpisem

$$F_{i+1}(h) = \frac{2^{p_i} F_i(\frac{h}{2}) - F_i(h)}{2^{p_i} - 1} = a_0 + a_{i+1}^{(i+1)} h^{p_{i+1}} + \dots, \qquad i = 1, 2, \dots, \tag{5.10'}$$

a T_{si} počítáme podle

$$T_{si} := \frac{2^{p_i} T_{s,i-1} - T_{s-1,i-1}}{2^{p_i} - 1} = T_{s,i-1} + \frac{T_{s,i-1} - T_{s-1,i-1}}{2^{p_i} - 1}, \quad i = 1, 2, \dots, s. \tag{5.14'}$$

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 (5.5), (5.10) a (5.14). □

Příklad 5.2. Richardsonovou extrapolací zpřesníme výpočet derivace podle formule (5.2a). Z Taylorovy věty dostaneme

$$F(h) := \frac{f(x + h) - f(x)}{h} = f'(x) + \frac{f^{(2)}(x)}{2!} h + \frac{f^{(3)}(x)}{3!} h^2 + \dots, \tag{5.18}$$

což odpovídá (5.5') pro $p_i = i$. Počítat budeme stejnou úlohu jako v příkladu 5.1. 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 (5.3b) je T_{32} aproximace řádu h^6, pro formuli (5.2a) je T_{44} aproximace řádu h^5 a k dosažení požadované přesnosti bylo třeba zvolit menší h_s. □

5.3. Numerické integrování

Cílem tohoto odstavce je přibližný výpočet určitého integrálu $I(f) := \int_a^b f(x)\,\mathrm{d}x$. Existuje několik důvodů, proč tento integrál nepočítáme přesně, například

1) integrál $I(f)$ neumíme spočítat analytickými metodami;

2) analytický výpočet je příliš pracný;

3) 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 $f(x)$.

5.3.1. Základní vlastnosti kvadraturních formulí

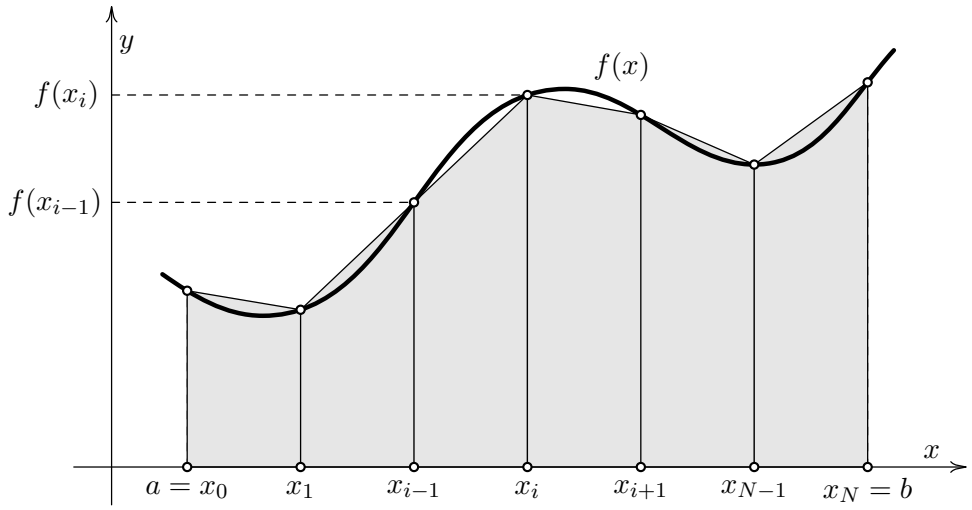
Začneme tím, že si uvedeme jednoduchý

Příklad 5.3. Pro větší názornost výkladu předpokládejme, že funkce $f(x)$ je na intervalu $\langle a, b\rangle$ kladná. Interval $\langle a, b\rangle$ rozdělíme na n stejných dílků délky $h = (b-a)/n$ pomocí dělicích bodů $x_i = a + ih$, $i = 0, 1, \ldots, n$, a obsah plochy pod křivkou $f(x)$ nahradíme obsahem plochy pod lomenou čárou spojující body $[x_{i-1}, f(x_{i-1})]$ a $[x_i, f(x_i)]$, $i = 1, 2, \ldots, n$. Funkce $\varphi(x)$ je tedy po částech lineární aproximací funkce $f(x)$ a přibližná hodnota integrálu $I(f)$ je součtem obsahů lichoběžníků s vrcholy $[x_{i-1}, 0]$, $[x_i, 0]$, $[x_i, f(x_i)]$ a $[x_{i-1}, f(x_{i-1})]$.

Výsledkem je předpis

$$Q_T^n(f) := \sum_{i=1}^{n} \tfrac{1}{2}h(f_{i-1} + f_i) = h\left[\tfrac{1}{2}f_0 + f_1 + \cdots + f_{n-1} + \tfrac{1}{2}f_n\right], \tag{5.19}$$

kde $f_i := f(x_i)$. Dostali jsme tak jednu ze základních formulí numerického integrování, tzv. *složenou lichoběžníkovou formuli*. Index $_T$ značí „trapezoid“, což je anglický překlad slova lichoběžník. V dalším budeme formule pro přibližný výpočet integrálů označovat také jako *kvadraturní formule*.



Obr. 5.2. Složená lichoběžníková formule

Diskretizační chyba. Určeme chybu, které se dopustíme, když místo přesného integrálu $I(f)$ použijeme kvadraturní formuli $Q_T^n(f)$. Na intervalu $\langle x_{i-1}, x_i\rangle$ vznikne chyba

$$r_i := \int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x - \frac{1}{2}h[f(x_{i-1}) + f(x_i)] = \int_{x_{i-1}}^{x_i} [f(x) - P_1(x)]\,\mathrm{d}x\,,$$

kde

$$P_1(x) = f(x_{i-1}) + \frac{f(x_i) - f(x_{i-1})}{h}\,(x - x_{i-1})$$

je lineární interpolační polynom funkce $f(x)$. Užitím vzorce (4.23) pro chybu interpolace $f(x) - P_1(x)$ dostaneme

$$r_i = \int_{x_{i-1}}^{x_i} \frac{1}{2}f''(\xi(x))(x - x_{i-1})(x - x_i)\,\mathrm{d}x\,.$$

Protože funkce $(x - x_{i-1})(x - x_i)$ nemění na intervalu $\langle x_{i-1}, x_i\rangle$ znaménko, z první věty o střední hodnotě integrálu[4] plyne existence bodu $\eta_i \in (x_{i-1}, x_i)$ takového, že

$$r_i = f''(\eta_i) \int_{x_{i-1}}^{x_i} \frac{1}{2}(x - x_{i-1})(x - x_i)\,\mathrm{d}x = -\frac{1}{12}f''(\eta_i)h^3\,.$$

[4]První věta o střední hodnotě integrálu, viz [44]: Je-li $f(x)$ spojitá v $\langle a, b\rangle$ a je-li $g(x)$ integrovatelná a $g(x) \geq 0$ resp. $g(x) \leq 0$, existuje aspoň jeden takový bod $c \in (a, b)$, že

$$\int_a^b f(x)g(x)\,\mathrm{d}x = f(c)\int_a^b g(x)\,\mathrm{d}x\,.$$

Pro celkovou chybu $R_T^n(f) = I(f) - Q_T^n(f)$ proto dostaneme

$$R_T^n(f) = -\frac{1}{12}\, h^3 \sum_{i=1}^{n} f''(\eta_i) = -\frac{h^2(b-a)}{12n} \sum_{i=1}^{n} f''(\eta_i) = -\frac{b-a}{12} f''(\eta) h^2\,, \tag{5.20}$$

kde $\eta \in (a,b)$. Tedy $R_T^n(f) = O(h^2)$ a *diskretizační chyba* $R_T^n(f) \to 0$ pro $h \to 0$.

Zaokrouhlovací chyba. Podívejme se ještě na chybu zaokrouhlovací. Jestliže v kvadraturní formuli (5.19) použijeme místo přesných hodnot f_i přibližné hodnoty $\tilde{f}_i = f_i + \varepsilon_i$, pak

$$\tilde{Q}_T^n(f) := h[\tfrac{1}{2}\tilde{f}_0 + \tilde{f}_1 + \cdots + \tilde{f}_{n-1} + \tfrac{1}{2}\tilde{f}_n] = Q_T^n(f) + h[\tfrac{1}{2}\varepsilon_0 + \varepsilon_1 + \cdots + \varepsilon_{n-1} + \tfrac{1}{2}\varepsilon_n]\,.$$

Zřejmě $I(f) = \tilde{Q}_T^n(f) + R_T^n(f) + E_R$, kde $E_R := -h[\frac{1}{2}\varepsilon_0 + \varepsilon_1 + \cdots + \varepsilon_{n-1} + \frac{1}{2}\varepsilon_n]$ je celková *zaokrouhlovací chyba*. Jestliže $|\varepsilon_i| \le \varepsilon := M_0\varepsilon_m$, kde $M_0 = \max_{\langle a,b\rangle} |f(x)|$ a ε_m je strojová přesnost, dostaneme odhad

$$|E_R| \le hn\varepsilon = (b-a)\varepsilon = (b-a)M_0\varepsilon_m\,.$$

Vidíme, že odhad velikosti zaokrouhlovací chyby $|E_R|$ nezávisí na h. Pokud uvážíme také zaokrouhlovací chyby aritmetických operací, odhad zaokrouhlovací chyby bude

$$|E_R| \le (b-a)M_0\varepsilon_m + (n+3)M_0\varepsilon_m.$$

Protože n nebývá příliš velké, můžeme konstatovat, že numerické integrování je podstatně méně citlivé na zaokrouhlovací chyby než numerické derivování a proto při běžném numerickém výpočtu integrálu nemusíme zaokrouhlovacím chybám věnovat žádnou zvláštní pozornost. □

Obecný tvar kvadraturní formule. Kvadraturní formuli pro přibližný výpočet integrálu $I(f) = \int_a^b f(x)\,\mathrm{d}x$ budeme zapisovat ve tvaru

$$Q(f) = w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n)\,. \tag{5.21}$$

Body $x_0, x_1, \ldots, x_n$ se nazývají *uzly* a čísla $w_0, w_1, \ldots, w_n$ *koeficienty* kvadraturní formule. Budeme předpokládat, že $a \le x_0 < x_1 < \cdots < x_n \le b$. Rozdíl $I(f) - Q(f)$ označíme $R(f)$ a nazveme *(diskretizační) chybou kvadraturní formule*, tedy

$$I(f) = Q(f) + R(f)\,.$$

Řád kvadraturní formule. *Algebraický řád* $r = r(Q)$ kvadraturní formule $Q(f)$ definujeme jako nezáporné celé číslo takové, že $R(x^j) = 0$ pro $j = 0, 1, \ldots, r$ a $R(x^{r+1}) \ne 0$. Formule je tedy řádu r, pokud integruje přesně polymomy stupně r a polynomy stupně $r+1$ už přesně neintegruje. Charakteristika kvality kvadraturní formule pomocí algebraického řádu je přirozená. Podle Weierstrassovy věty, viz [44], lze totiž každou spojitou funkci libovolně přesně aproximovat polynomem.

Nechť $P_n(x) = f(x_0)\ell_0(x) + f(x_1)\ell_1(x) + \cdots + f(x_n)\ell_n(x)$ je Lagrangeův interpolační polynom funkce $f(x)$ stupně n. Integrací $P_n(x)$ na intervalu $\langle a,b\rangle$ dostaneme tzv. *interpolační kvadraturní formuli*

$$Q(f) = I(P_n) = f(x_0)\int_a^b \ell_0(x)\,\mathrm{d}x + f(x_1)\int_a^b \ell_1(x)\,\mathrm{d}x + \cdots + f(x_n)\int_a^b \ell_n(x)\,\mathrm{d}x\,.$$

Formule je tvaru (5.21) a její koeficienty jsou

$$w_i = \int_a^b \ell_i(x)\,\mathrm{d}x\,, \quad i = 0, 1, \ldots, n\,. \tag{5.22}$$

Interpolační kvadraturní formule (5.21), (5.22) je zřejmě řádu alespoň n, lze však dokázat, že řádu vyššího než $2n+1$ být nemůže.

Zvyšování řádu? Je přirozené ptát se, zda kvadraturní formule dostatečně vysokého řádu dokáže aproximovat integrál s požadovanou přesností. Odpověď na tuto otázku nám dává následující

Věta 5.1. *Nechť $Q_i(f) := \sum_{j=0}^{n_i} w_j^i f(x_j^i)$, $i = 0, 1, \ldots,$ jsou formule s kladnými koeficienty řádu r_i, přičemž $0 \le r_0 < r_1 < \cdots < r_i < r_{i+1} < \ldots$. Pak pro každou funkci $f(x)$ spojitou na intervalu $\langle a, b\rangle$ platí*

$$\lim_{i\to\infty} Q_i(f) = I(f)\,.$$

Důkaz. K libovolnému $\varepsilon > 0$ existuje podle Weierstrassovy věty polynom $P_m(x)$ stupně m takový, že $|f(x) - P_m(x)| < \varepsilon\ \forall x \in \langle a, b\rangle$. Pak pro každé $r_i \ge m$ platí

$$|I(f) - Q_i(f)| \le |I(f - P_m)| + |I(P_m) - Q_i(P_m)| + |Q_i(P_m - f)| \le 2\varepsilon(b-a)\,,$$

neboť $|I(f - P_m)| \le \varepsilon(b-a)$, $|I(P_m) - Q_i(P_m)| = 0$ protože formule Q_i je řádu $r_i \ge m$, a dále

$$|Q_i(P_m - f)| \le \sum_{j=0}^{n_i} |w_j^i||P_m(x_j^i) - f(x_j^i)| \le \varepsilon \sum_{j=0}^{n_i} |w_j^i| = \varepsilon \sum_{j=0}^{n_i} w_j^i = \varepsilon(b-a)\,.$$

V poslední nerovnosti jsme využili toho, že $w_j^i > 0$ a že $Q_i(1) = I(1) = b - a$. □

Chceme-li tedy spočítat integrál $I(f)$ s předepsanou přesností $\varepsilon > 0$, tj. tak, aby platilo $|I(f) - Q(f)| < \varepsilon$, stačí použít interpolační kvadraturní formuli dostatečně vysokého řádu s kladnými koeficienty. Z definice (5.22) koeficientů interpolační formule je zřejmé, že pozitivnost koeficientů w_i lze zajistit jedině vhodným výběrem uzlů kvadraturní formule. V odstavci 3 poznáme Gaussovy formule, které požadované vlastnosti mají.

Složené formule. Jinou cestou umožňující docílit potřebnou přesnost numerické integrace je použití *složených kvadraturních formulí*. Jednu složenou formuli jsme již poznali, totiž složenou lichoběžníkovou formuli $Q_T^n(f)$, viz (5.19). Z odhadu chyby (5.20) plyne, že pro dostatečně jemné dělení intervalu $\langle a, b\rangle$ je chyba $|I(f) - Q_T^n(f)|$ numerické integrace složenou lichoběžníkovou formulí libovolně malá. Používají se i jiné složené formule než je formule lichoběžníková. Hlavní myšlenka je však vždy stejná: interval $\langle a, b\rangle$ se rozdělí na m podintervalů $\langle a_i, b_i\rangle$, $i = 1, 2, \ldots, m$, tak, že

$$a = a_1 < b_1 = a_2 < b_2 = a_3 < \cdots < b_{m-1} = a_m < b_m = b\,,$$

a na každém podintervalu $\langle a_i, b_i\rangle$ se použije kvadraturní formule $Q_i(f)$ nevysokého řádu. Formule na jednotlivých podintervalech jsou obvykle téhož typu. Složenou kvadraturní formulí pak rozumíme formuli $Q^m(f) = \sum_{i=1}^m Q_i(f)$. Označíme-li $\sigma = \max_i(b_i - a_i)$

délku nejdelšího podintervalu, vzniká přirozená otázka, zda $Q^m(f) \to I(f)$ pro $\sigma \to 0$? Následující věta říká, že za velmi slabých předpokladů tomu tak skutečně je.

Věta 5.2. *Nechť $a = a_1 < b_1 = a_2 < b_2 = a_3 < \cdots < b_{m-1} = a_m < b_m = b$ je dělení intervalu $\langle a, b\rangle$, $\sigma = \max_i(b_i - a_i)$, $Q^m(f) = \sum_{i=1}^m Q_i(f)$ je složená kvadraturní formule na intervalu $\langle a, b\rangle$ a $Q_i(f) := \sum_{j=0}^{n_i} w_j^i f(x_j^i)$ je kvadraturní formule na intervalu $\langle a_i, b_i\rangle$, která má kladné koeficienty a je řádu alespoň 0. Pak pro každou funkci $f(x)$, která je v intervalu $\langle a, b\rangle$ lipschitzovsky spojitá, platí*

$$\lim_{\sigma\to 0} Q^m(f) = I(f)\,.$$

Důkaz. Užitím první věty o střední hodnotě integrálu, a dále toho, že pro formuli řádu 0 platí $\sum_{j=0}^{n_i} w_j^i = Q_i(1) = \int_{a_i}^{b_i} \mathrm{d}x = b_i - a_i$, dostaneme

$$\left|\int_{a_i}^{b_i} f(x)\,\mathrm{d}x - Q_i(f)\right| = |f(\eta_i)(b_i - a_i) - Q_i(f)| =$$

$$\left|\sum_{j=0}^{n_i} w_j^i f(\eta_i) - \sum_{j=0}^{n_i} w_j^i f(x_j^i)\right| \le \sum_{j=0}^{n_i} |w_j^i|\, L\, |\eta_i - x_j^i| \le$$

$$L(b_i - a_i)\sum_{j=0}^{n_i} w_j^i = L(b_i - a_i)^2 \le L\sigma(b_i - a_i),$$

a odtud

$$|I(f) - Q^m(f)| \le L\sigma\sum_{i=1}^{m}(b_i - a_i) \le L(b-a)\sigma \to 0 \quad \text{pro } \sigma \to 0\,. \qquad \square$$

5.3.2. Newtonovy-Cotesovy kvadraturní formule

jsou interpolační formule s uzly $x_i = x_0 + ih$, $i = 0, 1, \ldots, n$. *Uzavřené* Newtonovy-Cotesovy formule dostaneme pro $n \ge 1$, $h = (b-a)/n$ a $x_0 = a$, takže $x_n = b$, tj. koncové body intervalu $\langle a, b\rangle$ jsou uzly. *Otevřené* Newtonovy-Cotesovy formule dostaneme pro $n \ge 0$, $h = (b-a)/(n+2)$ a $x_0 = a + h$, takže $x_n = b - h$, tj. koncové body a, b uzly nejsou. Lze dokázat, že Newtonovy-Cotesovy formule jsou řádu $n+1$ pro n sudé a řádu n pro n liché. Z těchto formulí mají všechny koeficienty kladné jen uzavřené Newtonovy-Cotesovy formule pro $n = 1, 2, \ldots, 7$ a $n = 9$, z otevřených Newtonových-Cotesových formulí pak jen formule pro $n = 0, 1$ a $n = 3$. Nejznámější Newtonovy-Cotesovy formule jsou formule obdélníková, lichoběžníková a Simpsonova. V dalším se nám bude hodit také formule Booleova. Všechny tyto formule si nyní uvedeme.

Obdélníková formule je otevřená Newtonova-Cotesova formule pro $n = 0$ s jediným uzlem $x_0 = \frac{1}{2}(a+b)$. Interpolační Lagrangeův polynom $P_0(x)$ stupně 0 proto je

$$P_0(x) = f_0 := f\left(\frac{a+b}{2}\right),$$

a odpovídající formule je tvaru

$$Q_M(f) := \int_a^b P_0(x)\,\mathrm{d}x = (b-a)f\left(\frac{a+b}{2}\right). \tag{5.23}$$

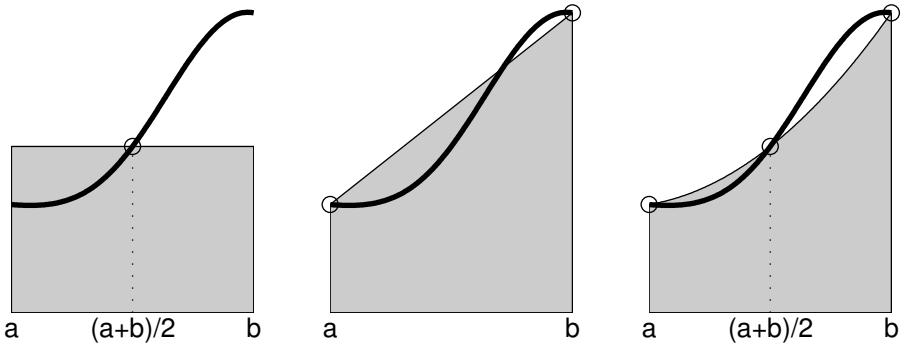
Index $_M$ značí „midpoint“, v anglicky psané literatuře se totiž obdélníková formule označuje jako „midpoint rule“. Název formule vyjadřuje skutečnost, že pro $f((a+b)/2) > 0$ je $Q_M(f)$ obsah obdélníka o stranách délky $b-a$ a $f((a+b)/2)$. Pro chybu platí

$$R_M(f) = \frac{1}{24}f''(\eta)(b-a)^3, \quad \text{kde } \eta \in (a,b). \tag{5.24}$$

Chybu obdélníkové formule odvodíme snadno pomocí Taylorova rozvoje a první věty o střední hodnotě integrálu:

$$\int_a^b f(x)\,\mathrm{d}x - (b-a)f\left(\frac{a+b}{2}\right) = \int_a^b \left[f(x) - f\left(\frac{a+b}{2}\right)\right]\mathrm{d}x =$$
$$\int_a^b \left[f'\left(\frac{a+b}{2}\right)\left(x-\frac{a+b}{2}\right) + \frac{f''(\xi(x))}{2}\left(x-\frac{a+b}{2}\right)^2\right]\mathrm{d}x =$$
$$\int_a^b \left(x-\frac{a+b}{2}\right)^2 \frac{f''(\xi(x))}{2}\,\mathrm{d}x = \frac{f''(\eta)}{2}\int_a^b \left(x-\frac{a+b}{2}\right)^2 \mathrm{d}x = \frac{1}{24}f''(\eta)(b-a)^3.$$

Obdélníková formule je řádu 1. To okamžitě plyne ze vzorce (5.24): protože druhá derivace polynomu stupně jedna je rovna nule, musí být $R_M(x^j) = 0$ pro $j = 0, 1$.



Obr. 5.3. Obdélníková, lichoběžníková a Simpsonova formule

Lichoběžníková formule je uzavřená Newtonova-Cotesova formule pro $n = 1$ s uzly $x_0 = a$, $x_1 = b$. Formuli jsme již odvodili v úvodu této kapitoly. Zopakujme tedy, že lichoběžníkovou formuli dostaneme integrací lineárního interpolačního polynomu procházejícího body $[a, f(a)]$ a $[b, f(b)]$ v mezích od a do b. Snadným výpočtem dostaneme

$$Q_T(f) := \int_a^b P_1(x)\,\mathrm{d}x = \frac{b-a}{2}[f(a) + f(b)]. \tag{5.25}$$

Název formule vyjadřuje skutečnost, že pro $f(a) > 0$, $f(b) > 0$ je $Q_L(f)$ obsah lichoběžníka, jehož rovnoběžné strany mají délky $f(a)$, $f(b)$ a jehož výška je rovna $b-a$.

Pro chybu lichoběžníkové formule platí

$$R_T(f) = -\frac{1}{12} f''(\eta)(b-a)^3, \quad \text{kde } \eta \in (a,b). \tag{5.26}$$

Vzorec pro chybu byl odvozen v úvodu této kapitoly, chyba tam byla označena jako r_i. Lichoběžníková formule je řádu 1. 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 formule 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. je-li 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 je uzavřená Newtonova-Cotesova formule pro $n = 2$ s uzly $x_0 = a$, $x_1 = (a+b)/2$ a $x_2 = b$. Integrací kvadratického Lagrangeova interpolačního polynomu $P_2(x)$ procházejícího body $[x_0, f(x_0)]$, $[x_1, f(x_1)]$ a $[x_2, f(x_2)]$ dostaneme

$$Q_S(f) = \int_a^b P_2(x)\,\mathrm{d}x = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{5.27}$$

Snadno prověříme, že $Q_S(x^j) = I(x^j)$ pro $j = 0,1,2,3$, a že $Q_S(x^4) \neq I(x^4)$, tedy Simpsonova formule je řádu 3. Odvození vzorce pro chybu $R_S(f)$ Simpsonovy formule je už obtížnější, spokojíme se proto jen se vzorcem samotným. Platí

$$R_S(f) = -\frac{1}{90} f^{(4)}(\eta)\left(\frac{b-a}{2}\right)^5, \quad \text{kde } \eta \in (a,b)\,. \tag{5.28}$$

Booleova formule je uzavřená Newtonova-Cotesova formule pro $n = 4$,

$$Q_B(f) = \frac{b-a}{90}\left[7f(a) + 32f\left(\frac{3a+b}{4}\right) + 12f\left(\frac{a+b}{2}\right) + 32f\left(\frac{a+3b}{4}\right) + 7f(b)\right]. \tag{5.29}$$

Formule je řádu 5, pro chybu platí

$$R_B(f) = -\frac{8}{945} f^{(6)}(\eta)\left(\frac{b-a}{4}\right)^7, \quad \text{kde } \eta \in (a,b)\,. \tag{5.30}$$

Složené formule uvedeme jen pro případ, kdy jednoduché formule na podintervalech jsou všechny stejné a to buďto obdélníkové nebo lichoběžníkové nebo Simpsonovy (sestavení složené Booleovy formule ponecháváme čtenáři jako cvičení). Budeme uvažovat ekvidistantní dělení $x_i = a + ih$, kde $h = (b-a)/n$.

Složenou obdélníkovou formuli dostaneme součtem jednoduchých obdélníkových formulí na jednotlivých podintervalech $\langle x_{i-1}, x_i\rangle$. Výsledkem je formule

$$Q_M^n(f) := h\left[f_{1/2} + f_{3/2} + \cdots + f_{n-1/2}\right], \quad \text{kde } f_{i-1/2} = f(x_i - \tfrac{1}{2}h). \tag{5.31}$$

Pro její chybu $R_M^n(f)$ platí

$$R_M^n(f) = \frac{1}{24}h^3 \sum_{i=1}^{n} f''(\eta_i) = \frac{h^2(b-a)}{24n} \sum_{i=1}^{n} f''(\eta_i) = \frac{b-a}{24} f''(\eta) h^2, \tag{5.32}$$

kde $\eta \in (a, b)$.

Složenou lichoběžníkovou formuli $Q_T^n(f)$ včetně její chyby $R_T^n(f)$ jsme již uvedli, viz (5.19), (5.20). V MATLABu je složená lichoběžníková metoda implementována jako program `trapz`.

Složenou Simpsonovu formuli dostaneme pro sudý počet dílků n tak, že sečteme jednoduché Simpsonovy formule na intervalech $\langle x_0, x_2\rangle$, $\langle x_2, x_4\rangle$, $\ldots$, $\langle x_{n-2}, x_n\rangle$. Dostaneme

$$\begin{aligned} Q_S^n(f) :=& \frac{2h}{6}[f_0 + 4f_1 + f_2] + \frac{2h}{6}[f_2 + 4f_3 + f_4] + \ldots \\ & \frac{2h}{6}[f_{n-4} + 4f_{n-3} + f_{n-2}] + \frac{2h}{6}[f_{n-2} + 4f_{n-1} + f_n] = \\ & \frac{h}{3}\left[f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n\right]. \end{aligned} \tag{5.33}$$

Pro chybu $R_S^n(f)$ složené Simpsonovy formule platí

$$\begin{aligned} R_S^n(f) =& -\frac{1}{90}h^5 f^{(4)}(\eta_2) - \frac{1}{90}h^5 f^{(4)}(\eta_4) - \ldots - \frac{1}{90}h^5 f^{(4)}(\eta_n) = \\ & -\frac{1}{90}h^4 \frac{b-a}{2}\frac{2}{n}\left[f^{(4)}(\eta_2) + f^{(4)}(\eta_4) + \cdots + f^{(4)}(\eta_n)\right] = \\ & -\frac{b-a}{180} f^{(4)}(\eta) h^4, \quad \text{kde } \eta \in (a, b). \end{aligned} \tag{5.34}$$

Rombergova integrace je založena na použití složené lichoběžníkové formule a opakované Richardsonovy extrapolace. Teoretickým východiskem Rombergovy metody je platnost *Eulerova–Maclaurinova vzorce*

$$\begin{aligned} & Q_T^n(f) = I(f) + \sum_{i=1}^{m} \frac{B_{2i}}{(2i)!} h^{2i}[f^{(2i-1)}(b) - f^{(2i-1)}(a)] + R_m(f), \\ & \text{kde} \quad R_m(f) = (b-a)\frac{B_{2m+2}}{(2m+2)!} h^{2m+2} f^{(2m+2)}(\eta_m), \quad \eta_m \in (a, b), \end{aligned} \tag{5.35}$$

koeficienty B_{2i} jsou tzv. *Bernoulliova čísla*, viz např. [43]. Vzorec (5.35) platí za předpokladu, že funkce $f(x)$ je v intervalu $\langle a, b\rangle$ spojitá spolu se svými derivacemi až do řádu $2m+2$ včetně. Aproximaci $Q_T^n(f)$ integrálu $I(f)$ lze zpřesnit užitím opakované Richardsonovy extrapolace. Rozvoj (5.35) je totiž stejného typu jako vzorec (5.5), na němž je opakovaná Richardsonova extrapolace založena. Srovnáním (5.5) a (5.35) vidíme, že ve vzorci (5.5) stačí položit

$$F(h) := Q_T^n(f), \quad h := \frac{b-a}{n}, \quad a_0 := I(f), \quad p_i := 2i.$$

Při výpočtu koeficientů T_{si} postupujeme podle vzorců (5.13)–(5.14), $n_s = 2^s n_0$ je počet dílků formule $Q_T^{n_s}(f)$ s krokem $h_s = h_0/2^s$. Při výpočtu $T_{s+1,0} = Q_T^{n_{s+1}}(f)$ využijeme všechny hodnoty funkce $f(x)$, které jsme již předtím použili k určení $T_{s0} = Q_T^{n_s}(f)$, viz krok 3 v následujícím algoritmu.

Algoritmus Rombergovy metody

Krok 1. Dáno a, b, n, ε_R, ε_A.

Krok 2. $s := 0$; $n_0 := n$; $h_0 := (b-a)/n$; $T_{00} := Q_T^n(f)$;

Krok 3. $T_{s+1,0} := \frac{1}{2}\left[T_{s0} + h_s \sum_{k=1}^{n_s} f(a + \left(k - \frac{1}{2}\right) h_s)\right]$;

Krok 4. $n_{s+1} := 2n_s$; $h_{s+1} := \frac{1}{2}h_s$; $s := s+1$; $i := 1$;

Krok 5. $\delta := (T_{s,i-1} - T_{s-1,i-1})/(4^i - 1)$; $T_{si} := T_{s,i-1} + \delta$;

je-li $|\delta| < \max(\varepsilon_R |T_{si}|, \varepsilon_A)$, pokračujeme krokem 6;

je-li $i = s$, pokračujeme krokem 3;

jinak položíme $i := i+1$ a opakujeme krok 5;

Krok 6. Položíme $I(f) \approx T_{si}$;

Následuje přehled základních vlastností Rombergovy metody.

a) Výpočet T_{si} odpovídá kvadraturní formuli řádu $2i+1$ s kladnými koeficienty.

b) Pro $i = 1$ dostáváme složenou Simpsonovu formuli: $T_{s1} = Q_S^{n_s}(f)$, kde $n_s = n_0 2^s$.

c) Aproximace T_{s2} je složená Booleova formule. T_{si} pro $i > 2$ však už žádnou přímou souvislost se složenými formulemi Newtonova-Cotesova typu nemá.

d) Pro chybu podle (5.35) a (5.16) platí $|T_{si} - I(f)| \leq K_i (b-a) h_{s-i}^{2i+2}$, kde $h_{s-i} = h_0/2^{s-i}$. Konstanta K_i závisí na $(2i+2)$–hé derivaci funkce $f(x)$.

e) Má-li funkce f v intervalu $\langle a, b\rangle$ spojité derivace až do řádu $2i+2$ včetně, pak $T_{si} \to I(f)$ pro $s \to \infty$. Pokud má funkce f v intervalu $\langle a, b\rangle$ derivace všech řádů, pak také $T_{ss} \to I(f)$ pro $s \to \infty$.

Zdůrazněme, že úspěšné použití Rombergovy metody předpokládá, že integrovaná funkce $f(x)$ má dostatečný počet spojitých derivací v celém intervalu $\langle a, b\rangle$.

Poznámka. Z Eulerova–Maclaurinova vzorce (5.34) plyne

$$Q_T^n(f) = I(f) + O(h^{2m+2}), \text{ pokud } f^{(2i-1)}(a) = f^{(2i-1)}(b) \text{ pro } i = 1, 2, \ldots, m.$$

Složené lichoběžníkové pravidlo je tedy velmi přesné při integraci dostatečně hladkých periodických funkcí v případě, kdy délka $b-a$ intervalu integrace $\langle a, b\rangle$ je celým násobkem periody. □

5.3.3. Gaussovy kvadraturní formule

Zabývejme se numerickým výpočtem

$$\int_a^b \omega(x) f(x)\,\mathrm{d}x,$$

kde ω je pevně daná nezáporná *váhová funkce* na intervalu $\langle a, b\rangle$. Interval $\langle a, b\rangle$ může být neomezený, třeba $\langle 0, \infty)$ nebo $(-\infty, \infty)$. O váhové funkci budeme předpokládat, že

$$\begin{aligned}
&\text{(a) } \omega \text{ je v } (a,b) \text{ spojitá a nezáporná,}\\
&\text{(b) integrály } \textstyle\int_a^b \omega(x)x^k\,\mathrm{d}x,\ k=0,1,\ldots, \text{ existují a jsou konečné,}\\
&\text{(c) je-li } p \text{ polynom, } p(x)\ge 0 \text{ v } \langle a,b\rangle,\ \textstyle\int_a^b \omega(x)p(x)\,\mathrm{d}x = 0, \text{ pak } p\equiv 0.
\end{aligned} \tag{5.36}$$

Podmínky (5.36) jsou splněny například tehdy, když funkce ω je kladná a spojitá na omezeném uzavřeném intervalu $\langle a, b\rangle$.

Pomocí první věty o střední hodnotě integrálu snadno dokážeme, že podmínka (5.36c) je ekvivalentní s podmínkou $\int_a^b \omega(x)\,\mathrm{d}x > 0$.

Nechť $P_n(x) = \sum_{i=0}^n f(x_i)\ell_i(x)$ Lagrangeův interpolační polynom funkce $f(x)$. *Interpolační kvadraturní formuli s váhou* definujeme předpisem

$$Q(f) := \int_a^b \omega(x)P_n(x)\,\mathrm{d}x = \sum_{i=0}^n w_i f(x_i),$$

kde koeficienty

$$w_i = \int_a^b \omega(x)\ell_i(x)\,\mathrm{d}x, \quad i = 0,1,\ldots,n. \tag{5.37}$$

Pro $a \le x_0 < x_1 < \cdots < x_n \le b$ je $Q(f)$ obecně jen řádu n, tj. $Q(x^i) = \int_a^b \omega(x)x^i\,\mathrm{d}x$, pro $0 \le i \le n$. Vhodnou volbou uzlů $\{x_i\}_{i=0}^n$ lze řád formule (5.37) výrazně zvýšit. K tomu použijeme

Ortogonální polynomy. Pro funkce u, v definujeme skalární součin

$$(u, v) := \int_a^b \omega(x)u(x)v(x)\,\mathrm{d}x$$

v prostoru všech funkcí f takových, pro které integrál $\int_a^b \omega(x)f^2(x)\,\mathrm{d}x$ existuje a je konečný.

Nechť $\{p_n(x)\}_{n=0}^\infty$ je soustava polynomů stupňů n. Řekneme, že tato soustava je v intervalu $\langle a, b\rangle$ ortogonální, když platí

$$(p_i, p_j) = 0 \quad \text{pro } i \ne j\,.$$

Ke konstrukci ortogonálních polynomů se využívá rekurentní vztah

$$\begin{aligned}
&p_{i+1}(x) = (x-\alpha_i)p_i(x) - \beta_i p_{i-1}(x), \quad i = 1,2,\ldots,\\
&\text{přičemž klademe} \quad p_0(x) = 1, \quad p_1(x) = x - \alpha_0\,.
\end{aligned} \tag{5.38}$$

Koeficienty $\alpha_0, \alpha_1, \beta_1, \ldots, \alpha_i, \beta_i, \ldots$ jsou zvoleny tak, aby polynomy $p_0, p_1, \ldots, p_{i+1}$ byly ortogonální. Koeficient α_0 určíme z podmínky $0 = (p_1, p_0) = (x - \alpha_0, p_0)$. Dále postupujme indukcí, tj. předpokládáme $(p_i, x^k) = 0$ pro $k < i$ a určíme α_i, β_i. Žádáme

$$0 = (p_{i+1}, p_i) = ((x - \alpha_i)p_i, p_i) - \beta_i(p_{i-1}, p_i) = (xp_i, p_i) - \alpha_i(p_i, p_i) - \beta_i(p_{i-1}, p_i),$$

a protože $(p_{i-1}, p_i) = 0$ podle indukčního předpokladu, je

$$\alpha_i = \frac{(xp_i, p_i)}{(p_i, p_i)}\,.$$

Podobně z požadavku

$$0 = (p_{i+1}, p_{i-1}) = ((x - \alpha_i)p_i, p_{i-1}) - \beta_i(p_{i-1}, p_{i-1}) = (xp_i, p_{i-1}) - \beta_i(p_{i-1}, p_{i-1})$$

dostaneme

$$\beta_i = \frac{(xp_i, p_{i-1})}{(p_{i-1}, p_{i-1})}\,.$$

Tento vztah ještě upravíme pomocí rovnosti

$$(p_i, p_i) = ((x - \alpha_{i-1})p_{i-1}, p_i) - \beta_{i-1}(p_{i-2}, p_i) = (xp_{i-1}, p_i)$$

na výsledný tvar

$$\beta_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}\,.$$

Snadno ověříme, že $(p_{i+1}, p_k) = 0$ také pro $k \le i - 2$, takže celkem $(p_{i+1}, p_k) = 0$, $k \le i$. Proto také $(p_{i+1}, x^k) = 0$, $k \le i$, neboť x^k lze vyjádřit jako lineární kombinaci $p_\ell(x)$ pro $\ell \le k$. Tím je indukční krok ukončen.

Koeficienty ortogonálních polynomů p_i tedy počítáme ze vzorců

$$\alpha_i = \frac{(xp_i, p_i)}{(p_i, p_i)}, \quad i = 0, 1, \ldots, \qquad \beta_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}, \quad i = 1, 2, \ldots \tag{5.39}$$

Ukážeme, že

Věta. *Kořeny x_i, $i = 1, 2, \ldots, n$, n-tého ortogonálního polynomu p_n jsou reálné, jednoduché a všechny leží v otevřeném intervalu (a, b).*

Důkaz. $\int_a^b \omega(x)p_n(x)\,\mathrm{d}x = (p_n, p_0) = 0$ pro $n > 0$. Proto p_n v (a, b) mění znaménko, tj. p_n má uvnitř $\langle a, b\rangle$ alespoň jeden kořen liché násobnosti. Nechť $a < z_1 < z_2 < \ldots < z_\ell < b$, kde $\{z_i\}_{i=1}^{\ell}$ jsou všechny kořeny p_n liché násobnosti. Položíme $q_\ell(x) = \prod_{i=1}^{\ell}(x - z_i)$. Pak $p_n q_\ell \ge 0$ v $\langle a, b\rangle$ a tedy $\int_a^b \omega(x)p_n(x)q_\ell(x)\,\mathrm{d}x > 0$ podle (5.36c). To však pro $\ell < n$ nemůže nastat. Proto $\ell = n$. □

Gaussova kvadraturní formule $Q_{Gn}(f) = \sum_{i=0}^{n} w_i f(x_i)$ je interpolační kvadraturní formule (5.37), jejíž uzly $\{x_i\}_{i=0}^{n}$ jsou kořeny $(n+1)$-ního ortogonálního polynomu p_{n+1}.

Tvrzení 1. *Gaussova kvadraturní formule Q_{Gn} je řádu $2n+1$.*

Důkaz. Polynom $f_{2n+1}(x)$ stupně $2n+1$ lze vyjádřit ve tvaru

$$f_{2n+1}(x) = P_n(x) + p_{n+1}(x)q_n(x),$$

kde $P_n(x) = \sum_{i=0}^{n} f_{2n+1}(x_i)\ell_i(x)$ je Lagrangeův interpolační polynom funkce $f_{2n+1}(x)$, jehož uzly $x_0, x_1 \ldots, x_n$ jsou kořeny ortogonálního polynomu $p_{n+1}(x)$, a $q_n(x)$ je nějaký polynom stupně n. Skutečně, polynom $f_{2n+1}(x) - P_n(x)$ má kořeny $x_0, x_1, \ldots, x_n$, je proto dělitelný členem $(x-x_0)(x-x_1)\ldots(x-x_n)$ a tedy také polynomem $p_{n+1}(x)$. Pak

$$\int_a^b \omega(x) f_{2n+1}(x)\,\mathrm{d}x = \int_a^b \omega(x)\left[P_n(x) + p_{n+1}(x)q_n(x)\right]\,\mathrm{d}x = Q_{Gn}(f_{2n+1}),$$

neboť $(p_{n+1}, q_n) = 0$ je důsledkem ortogonality polynomů $\{p_k(x)\}_{k=0}^{n+1}$ a

$$\int_a^b \omega(x)P_n(x)\,\mathrm{d}x = \sum_{i=0}^{n} f_{2n+1}(x_i)\int_a^b \omega(x)\ell_i(x)\,\mathrm{d}x = \sum_{i=0}^{n} w_i f_{2n+1}(x_i) = Q_{Gn}(f_{2n+1}).$$

Protože $\int_a^b \omega(x)p_{n+1}^2(x)\,\mathrm{d}x > 0$, Q_{Gn} je řádu $2n+1$. □

Tvrzení 2. *Koeficienty Gaussovy kvadraturní formule Q_{Gn} jsou kladné.*

Důkaz. Gaussova formule Q_{Gn} integruje polynom ℓ_i^2 stupně $2n$ přesně. Proto

$$\int_a^b \omega(x)\ell_i^2(x)\,\mathrm{d}x = Q_{Gn}(\ell_i^2) = \sum_{k=0}^{n} w_k \ell_i^2(x_k) = w_i\,,$$

neboť $\ell_i(x_k) = 0$ pro $i \neq k$ a $\ell_i(x_i) = 1$. Odtud $w_i = \int_a^b \omega(x)\ell_i^2(x)\,\mathrm{d}x > 0$. □

Tvrzení 3. *Interpolační kvadraturní formule Q, která integruje přesně polynomy stupňů $0, 1, \ldots, 2n+1$, je Gaussova kvadraturní formule Q_{Gn}.*

Důkaz. Jestliže interpolační kvadraturní formule (5.37) je řádu $2n+1$, pak pro ortogonální polynomy $\{p_k\}_{k=0}^{n+1}$ dostaneme

$$Q(p_{n+1}p_j) = \sum_{i=0}^{n} w_i p_{n+1}(x_i)p_j(x_i) = \int_a^b \omega(x)p_{n+1}(x)p_j(x)\,\mathrm{d}x = 0, \qquad j = 0, 1, \ldots, n,$$

nebo-li

$$\begin{pmatrix} p_0(x_0) & p_0(x_1) & \ldots & p_0(x_n) \\ p_1(x_0) & p_1(x_1) & \ldots & p_1(x_n) \\ \vdots & \vdots & \vdots & \vdots \\ p_n(x_0) & p_n(x_1) & \ldots & p_n(x_n) \end{pmatrix} \begin{pmatrix} w_0 p_{n+1}(x_0) \\ w_1 p_{n+1}(x_1) \\ \vdots \\ w_n p_{n+1}(x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{5.40}$$

Z regularity matice soustavy (5.40) plyne $w_i p_{n+1}(x_i) = 0$, $i = 0, 1, \ldots, n$, a protože $w_i \neq 0$, $p_{n+1}(x_i) = 0$. To znamená, že uzly $x_0, x_1, \ldots, x_n$ jsou kořeny ortogonálního polynomu p_{n+1}. Dokázali jsme tedy, že $Q = Q_{Gn}$. □

Důsledek. *Gaussova kvadraturní formule je interpolační kvadraturní formule maximálního řádu.*

Důkaz. Stačí zkombinovat tvrzení 1 a 3. Integruje-li interpolační kvadraturní formule přesně polynomy stupňů $0, 1, \ldots, 2n+1$, pak jde o Gaussovu kvadraturní formuli. Ta je řádu $2n+1$, tj. polynomy stupně $2n+2$ už přesně neintegruje. Řád $2n+1$ je tedy maximální a je dosažen právě pro Gaussovu kvadraturní formuli. □

Gaussova–Legendrova formule je určena pro výpočet

$$\int_{-1}^{1} f(x)\,\mathrm{d}x.$$

Polynomy ortogonální na intervalu $\langle -1, 1\rangle$ s váhou $\omega = 1$ jsou známy jako *Legendrovy polynomy*. Lze je popsat rekurentním předpisem

$$\begin{aligned} &(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \quad n = 1, 2, ... \\ &\text{přičemž klademe} \quad P_0(x) = 1, \quad P_1(x) = x\,. \end{aligned} \tag{5.41}$$

Zejména tedy

$$P_2(x) = \frac{1}{2}(3x^2 - 1), \qquad P_3(x) = \frac{1}{2}(5x^3 - 3x)\,.$$

Všimněte si, že polynomy sudého stupně obsahují jen sudé mocniny x a polynomy lichého stupně zase jen liché mocniny x. Dá se ukázat, že kořeny Legendrových polynomů jsou symetrické, tj. když $x_0 < x_1 < \cdots < x_n$, pak $x_{n-i} = -x_i$, $i = 0, 1, \ldots, n$. Pro n sudé $x_{n/2} = 0$.

Gaussova kvadraturní formule (5.37), jejíž uzly jsou kořeny Legendrova polynomu P_{n+1}, se nazývá *Gaussova–Legendrova* kvadraturní formule.

Chyba Gaussovy–Legendrovy kvadraturní formule může být vyjádřena ve tvaru

$$R_{Gn}(f) = d_n f^{(2n+2)}(\eta_{Gn}), \quad \text{kde } \eta_{Gn} \in (-1, 1) \quad \text{a} \quad d_n = \frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3}\,,$$

viz např. [20].

Koeficienty w_i Gaussových-Legendrových formulí lze vypočítat integrací Lagrangeových fundamentálních polynomů, $w_i = \int_{-1}^{1} \ell_i(x)\,\mathrm{d}x$, viz (5.22). Výhodnější je však použít formuli

$$w_i = \frac{2}{(1 - x_i^2)[P'_{n+1}(x_i)]^2}\,, \qquad i = 0, 1, \ldots, n. \tag{5.42}$$

Vybrané Gaussovy–Legendrovy formule. Pro $n = 0, 1, 2$ speciálně dostaneme formule

$$\begin{aligned} Q_{G0}(f) &= 2f(0), & R_{G0}(f) &= \frac{1}{3}f''(\eta_{G0})\,, \\ Q_{G1}(f) &= f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right), & R_{G1}(f) &= \frac{1}{135}f^{(4)}(\eta_{G1})\,, \\ Q_{G2}(f) &= \frac{5}{9}f\left(-\sqrt{0{,}6}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{0{,}6}\right), & R_{G2}(f) &= \frac{1}{15\,750}f^{(6)}(\eta_{G2})\,. \end{aligned}$$

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–Legendrova formule $Q_{G1}(f)$ je jen dvoubodová. Srovnáním tvaru zbytku $R_S(f)$ Simpsonovy formule a $R_{G1}(f)$ Gaussovy–Legendrovy formule lze usuzovat, že Gaussova–Legendrova formule je přibližně o 50% přesnější.

S rostoucím počtem uzlů formule $Q_{Gn}(f)$ koeficient d_n ve vyjádření zbytku $R_{Gn}(f)$ velmi prudce klesá, například pro 9-ti bodovou formuli $Q_{G8}(f)$ je $d_8 \approx 1,82 \cdot 10^{-21}$. Aby také chyba $R_{Gn}(f)$ byla malá, musí být integrovaná funkce $f(x)$ na intervalu $\langle -1, 1\rangle$ spojitá spolu se svými derivacemi až do řádu $2n+2$ a hodnota její $(2n+2)$–hé derivace nesmí být příliš velká: platí totiž

$$|R_{Gn}(f)| \le d_n M_{2n+2}, \quad \text{kde} \quad M_{2n+2} := \max_{\eta\in\langle -1,1\rangle} |f^{(2n+2)}(\eta)|\,.$$

Gaussovy–Legendrovy formule lze použít pro integraci na libovolném omezeném intervalu $\langle a, b\rangle$, stačí použít transformaci

$$x = \frac{a+b}{2} + \frac{b-a}{2}\xi, \qquad \xi \in \langle -1, 1\rangle,$$

pomocí které převedeme integraci z intervalu $\langle a, b\rangle$ na interval $\langle -1, 1\rangle$.

Gaussova-Čebyševova kvadraturní formule je určena pro výpočet

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\,\mathrm{d}x\,.$$

Polynomy ortogonální na intervalu $\langle -1, 1\rangle$ s váhou $w(x) = 1/\sqrt{1-x^2}$ jsou *Čebyševovy polynomy*, viz (4.28). Formule (5.37), jejíž uzly jsou kořeny Čebyševova polynomu T_{n+1}, se nazývá *Gaussova–Čebyševova* kvadraturní formule. Uzly a koeficienty jsou

$$x_i = \cos\frac{2i+1}{2n+2}\,\pi\,, \qquad w_i = \frac{\pi}{n+1}\,, \qquad i = 0, 1, \ldots, n\,. \tag{5.43}$$

Gaussova-Laguerrova kvadraturní formule umožňuje výpočet

$$\int_0^{\infty} e^{-x} f(x)\,\mathrm{d}x\,.$$

Ortogonální polynomy na intervalu $\langle 0, \infty)$ s váhou $w(x) = e^{-x}$ jsou *Laguerrovy polynomy* definované rekurentním předpisem

$$\begin{aligned} &L_0(x) = 1,\ L_1(x) = 1 - x,\\ &L_{n+1} = (2n+1-x)L_n(x) - n^2 L_{n-1}(x), \quad n = 1, 2, \ldots \end{aligned} \tag{5.44}$$

Gaussova kvadraturní formule (5.37), jejíž uzly jsou kořeny Laguerrova polynomu L_{n+1}, se nazývá *Gaussova–Laguerrova* kvadraturní formule. Uzly a koeficienty jsou

$$\begin{aligned} &x_i: \quad \text{kořeny Laguerrova polynomu } L_{n+1},\\ &w_i: \quad w_i = x_i \left[\frac{(n+1)!}{L_{n+2}(x_i)}\right]^2, \qquad i = 0, 1, \ldots, n. \end{aligned} \tag{5.45}$$

Gaussovu–Laguerrovu formuli lze použít také na intervalu $(-\infty, b\rangle$ resp. $\langle a, \infty)$: pomocí transformace $x = b - \xi$, $\xi \in (-\infty, 0\rangle$, resp. $x = a + \xi$, $\xi \in \langle 0, \infty)$, převedeme integrál přes interval $(-\infty, b\rangle$ resp. $\langle a, \infty)$ na integrál přes interval $\langle 0, \infty)$.

Gaussova-Hermitova kvadraturní formule je určena pro výpočet

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)\,\mathrm{d}x\,.$$

Ortogonální polynomy na intervalu $(-\infty, \infty)$ s váhou $w(x) = e^{-x^2}$ jsou *Hermitovy polynomy* definované rekurentním předpisem

$$H_0(x) = 1,\ H_1(x) = 2x,\ H_{n+1} = 2xH_n(x) - 2nH_{n-1}(x), \quad n = 1, 2, \ldots \tag{5.46}$$

Gaussova kvadraturní formule (5.37), jejíž uzly jsou kořeny Hermitova polynomu H_{n+1}, se nazývá *Gaussova–Hermitova* kvadraturní formule. Uzly a koeficienty jsou

$$\begin{aligned} &x_i: \quad \text{kořeny Hermitova polynomu } H_{n+1}, \\ &w_i: \quad w_i = \frac{2^{n+2}(n+1)!\sqrt{\pi}}{[H_{n+2}(x_i)]^2}, \qquad\qquad i = 0, 1, \ldots, n. \end{aligned} \tag{5.47}$$

Radauovy a Lobattovy formule V řadě aplikací je užitečné používat formule Gaussova typu, tj. maximálního možného řádu, které mají některé uzly předepsané. Zvláště důležité jsou případy, kdy mezi uzly potřebujeme zařadit jeden nebo oba koncové body intervalu $\langle -1, 1\rangle$. Je-li uzlem jen jeden z krajních bodů ± 1, dostáváme *Gaussovy–Radauovy* formule řádu $2n$, je-li uzlem jak bod -1 tak bod 1, dostáváme *Gaussovy–Lobattovy* formule řádu $2n - 1$. V následující specifikaci formulí předpokládáme, že $-1 \le x_0 < x_1 < \cdots < x_n \le 1$, takže w_0 přísluší k nejmenšímu uzlu x_0 a w_n přísluší k největšímu uzlu x_n.

Levá Gaussova-Radauova kvadraturní formule. Uzly a koeficienty jsou

$$\begin{aligned} &x_i: \quad \text{kořeny polynomu } P_{n+1} + P_n, \\ &w_i: \quad w_0 = \frac{2}{(n+1)^2}\,, \qquad w_i = \frac{1 - x_i}{[(n+1)P_n(x_i)]^2}\,, \qquad i = 1, 2, \ldots, n, \end{aligned} \tag{5.48}$$

kde P_n a P_{n+1} jsou Legendrovy polynomy.

Pravá Gaussova-Radauova kvadraturní formule. Uzly a koeficienty jsou

$$\begin{aligned} &x_i: \quad \text{kořeny polynomu } P_{n+1} - P_n, \\ &w_i: \quad w_i = \frac{1 + x_i}{[(n+1)P_n(-x_i)]^2}\,, \qquad i = 0, 1, \ldots, n-1\,, \qquad w_n = \frac{2}{(n+1)^2}\,, \end{aligned} \tag{5.49}$$

kde P_n a P_{n+1} jsou Legendrovy polynomy.

Gaussova-Lobattova kvadraturní formule má uzly a koeficienty

$$\begin{aligned} &x_i: \quad \text{kořeny polynomu } (1 - x^2)P_n', \\ &w_i: \quad w_i = \frac{2}{n(n+1)[P_n(x_i)]^2}, \qquad\qquad i = 0, 1, \ldots, n, \end{aligned} \tag{5.50}$$

kde P_n je Legendrův polynom.

Pro zajímavost uvádíme, že Lobattova formule se třemi uzly, tj. pro $n = 2$, je známá Simpsonova formule řádu 3. Lobattova formule pro $n = 3$ je řádu 5 a má tvar

$$Q(f) = \frac{1}{6}\left[f(-1) + f(1)\right] + \frac{5}{6}\left[f\left(-\frac{1}{\sqrt{5}}\right) + f\left(\frac{1}{\sqrt{5}}\right)\right]. \tag{5.51}$$

5.3.4. 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 prakticky dělá. Integrál $I(a,b) := \int_a^b f(x)\,\mathrm{d}x$ počítáme dvěma různými kvadraturními formulemi a dostaneme aproximace $Q_1(a,b)$ a $Q_2(a,b)$. Jedna z formulí bývá vždy přesnější než formule druhá. Řekněme, že přesnější je formule Q_2. Jestliže si na chvíli 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 vhodně zvolená tolerance, považujeme $Q(a,b) := Q_2(a,b)$ 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, např. 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 $Q(a,c)$ a $Q(c,b)$ integrálů $I(a,c)$ a $I(c,b)$ a nakonec položíme $Q(a,b) = Q(a,c) + Q(c,b)$. Algoritmus je rekurzivní: výpočet $Q(a,c)$ a $Q(c,b)$ na „dceřiných“ intervalech $\langle a,c\rangle$ a $\langle c,b\rangle$ probíhá analogicky jako výpočet $Q(a,b)$ na „mateřském“ intervalu $\langle a,b\rangle$.

Existuje celá řada programů pracujících na principu adaptivní integrace. Několik jich má také MATLAB. Program `quad` je založen na Simpsonově formuli Q_S^4, přesnější program `quadl` vychází z Lobattovy formule řádu 5, viz (5.51). Teoretický základ obou programů je vyložen v článku [19]. Kromě toho MATLAB nabízí ještě program `quadgk` používající Gaussovy-Kronrodovy formule řádů 7 a 15. Místo programů `quad` a `quadl` MATLAB doporučuje používat program `integral` založený na strategii popsané v článku [48].

Program `quad` používá jako $Q_1(a,b)$ složenou Simpsonovu formuli Q_S^4 řádu 3,

$$Q_1(a,b) = \frac{h}{12}[f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)]\,,$$

kde $h = b - a$, $c = \frac{1}{2}(a+b)$, $d = c - \frac{1}{4}h$, $e = c + \frac{1}{4}h$, a jako $Q_2(a,b)$ Booleovu formuli Q_B řádu 5,

$$Q_2(f) = \frac{h}{90}[7f(a) + 32f(d) + 12f(c) + 32f(e) + 7f(b)]\,.$$

Hruhý popis zaznamenává následující

algoritmus QUAD:

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 }

$c := (a+b)/2\,;$ { střed intervalu $\langle a,b\rangle$ }
if $\mathrm{abs}\,(I_2 - I_1) < \varepsilon$ **then** { je dosažena požadovaná přesnost ? }
$Q := I_2$ { ano, hodnota I_2 se akceptuje }
else { ne, rekurzivní volání funkce Q na dceři- }
$Q := Q(f,a,c,\varepsilon) + Q(f,c,b,\varepsilon)\,;$ { ných subintervalech $\langle a,c\rangle$ a $\langle c,b\rangle$ }

Kvůli jednoduchosti jsme do algoritmu QUAD 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 mateřského intervalu $\langle a,b\rangle$ do dceřiných intervalů $\langle a,c\rangle$ a $\langle c,b\rangle$.

Program `quadl` používá jako $Q_1(a,b)$ Lobattovu formuli řádu 5,

$$Q_1(a,b) = \frac{1}{6}h\{f(a)+f(b)+5[f(c-\alpha h)+f(c+\alpha h)]\}\,,$$

kde $h = \frac{1}{2}(b-a)$, $c = \frac{1}{2}(a+b)$, $\alpha = \frac{1}{\sqrt{5}}$, a jako formuli $Q_2(a,b)$ *Gaussovu-Kronrodovu* formuli řádu 9, viz [19],

$$Q_2(a,b) = \frac{h}{1470}\{77[f(a)+f(b)] + 432[f(c-\beta h)+f(c+\beta h)] + 625[f(c-\alpha h)+f(c+\alpha h)] + 672 f(c)\}\,,$$

kde $\beta = \sqrt{\frac{2}{3}}$. Jestliže $|Q_2(a,b) - Q_1(a,b)| \le \varepsilon$, klademe $Q(a,b) = Q_2(a,b)$, je-li však $|Q_2(a,b) - Q_1(a,b)| > \varepsilon$, interval $\langle a,b\rangle$ se opět rozdělí, tentokrát však na šest dceřiných intervalů $\langle a, c-\beta h\rangle$, $\langle c-\beta h, c-\alpha h\rangle$, $\langle c-\alpha h, c\rangle$, $\langle c, c+\alpha h\rangle$, $\langle c+\alpha h, c+\beta h\rangle$, $\langle c+\beta h, b\rangle$. Formule Q_1 a Q_2 používají celkem sedm hodnot funkce $f(x)$. Hodnoty v krajních bodech se přejímají z mateřského intervalu, takže na každém dceřiném intervalu je třeba spočítat pět nových funkčních hodnot. Při zjemňování dělení tedy využijeme všechny funkční hodnoty, které byly spočteny na mateřském intervalu.

5.3.5. Numerický výpočet vícerozměrných integrálů

Dvojné intergály na obdélníku. Nechť $D = \langle a,b\rangle \times \langle c,d\rangle$ je obdélník, pak

$$\iint_D f(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_a^b \int_c^d f(x,y)\,\mathrm{d}y\,\mathrm{d}x = \int_a^b g(x)\,\mathrm{d}x\,,\ \text{kde } g(x) = \int_c^d f(x,y)\,\mathrm{d}y\,.$$

Jestliže integrál $\int_a^b g(x)\,\mathrm{d}x$ aproximujeme formulí $Q_x(g) = \sum_{i=0}^n w_i^x g(x_i)$ řádu p_x a integrály $g(x_i) = \int_c^d f(x_i,y)\,\mathrm{d}y$ aproximujeme formulemi $Q_y(g(x_i)) = \sum_{j=0}^m w_j^y f(x_i,y_j)$ řádu p_y, dostaneme *součinovou kvadraturní formuli*

$$Q(f) = \sum_{i=0}^n \sum_{j=0}^m w_i^x w_j^y f(x_i,y_j)\,,$$

která integruje přesně polynomy $x^i y^j$ pro $0 \le i \le p_x$ a $0 \le j \le p_y$. Říkáme také, že formule je řádu p_x v proměnné x a řádu p_y v proměnné y. Formule Q_x a Q_y jsou nejčastěji téhož typu.

Je-li Q_x obdélníková formule na intervalu $\langle a,b\rangle$ a Q_y obdélníková formule na intervalu $\langle c,d\rangle$, dostaneme *součinovou obdélníkovou formuli*

$$Q(f) = |D| f\left(\frac{a+b}{2}, \frac{c+d}{2}\right), \tag{5.52}$$

kde $|D| = (b-a)(d-c)$ je obsah obdélníka D. Jsou-li obě formule lichoběžníkové, dostaneme *součinovou lichoběžníkovou formuli*

$$Q(f) = \frac{|D|}{4}\Big[f(a,c) + f(b,c) + f(b,d) + f(a,d)\Big], \tag{5.53}$$

a jsou-li obě formule Simpsonovy, dostaneme *součinovou Simpsonovu formuli*

$$\begin{aligned} Q(f) = \frac{|D|}{36}\Big\{ & f(a,c) + f(b,c) + f(b,d) + f(a,d) + \\ & 4\left[f\left(\frac{a+b}{2}, c\right) + f\left(b, \frac{c+d}{2}\right) + f\left(\frac{a+b}{2}, d\right) + f\left(a, \frac{c+d}{2}\right)\right] + \\ & 16 f\left(\frac{a+b}{2}, \frac{c+d}{2}\right)\Big\}. \end{aligned} \tag{5.54}$$

Formule na referenčním čtverci. Formule se často uvádějí na referenčním čtverci $\hat{D} = \langle -1,1\rangle^2$. Užitím transformace

$$x = \frac{a+b}{2} + \frac{b-a}{2}\xi, \quad y = \frac{c+d}{2} + \frac{d-c}{2}\eta, \quad -1 \le \xi, \eta \le 1,$$

lze integraci na obdélníku $\langle a,b\rangle \times \langle c,d\rangle$ převést na integraci na čtverci $\langle -1,1\rangle^2$.

Gaussovy součinové formule. Pro $Q_x = Q_y = Q_{Gn}$ dostáváme *součinové Gaussovy formule*. Je-li $n = 0$, obdržíme stejně jako v jedné dimenzi obdélníkovou formuli. Pro $n = 1$ dostaneme *Gaussovu* 2×2 *součinovou formuli*

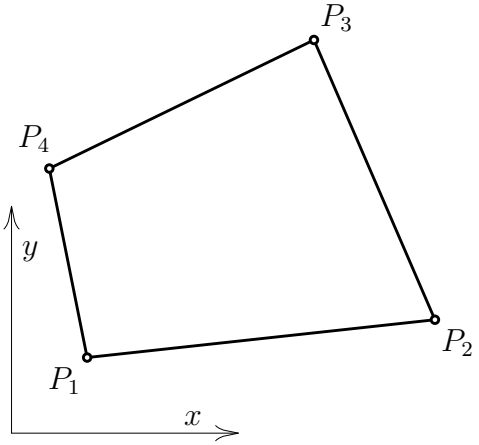
$$Q(g) = g(-\alpha,-\alpha) + g(\alpha,-\alpha) + g(\alpha,\alpha) + g(-\alpha,\alpha), \quad \alpha = \frac{1}{\sqrt{3}},$$

která na referenčním čtverci $\hat{D}$ integruje přesně polynomy $\xi^i\eta^j$ pro $0 \le i,j \le 3$. Formule je tedy stejného řádu jako formule Simpsonova, má však jen čtyři uzly zatímco formule Simpsonova jich má devět! Pro $n = 2$ dostaneme *Gaussovu* 3×3 *součinovou formuli*

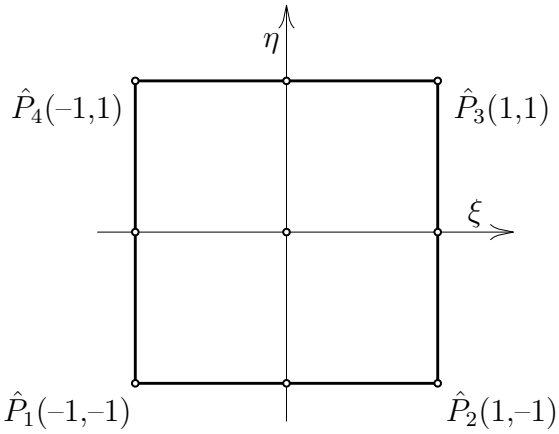
$$\begin{aligned} Q(g) = & \frac{25}{81}[g(-\alpha,-\alpha) + g(\alpha,-\alpha) + g(\alpha,\alpha) + g(-\alpha,\alpha)] + \\ & \frac{40}{81}[g(0,-\alpha) + g(\alpha,0) + g(0,\alpha) + g(-\alpha,0)] + \\ & \frac{64}{81} g(0,0), \quad \alpha = \sqrt{0{,}6}, \end{aligned}$$

která na referenčním čtverci $\hat{D}$ integruje přesně polynomy $\xi^i\eta^j$ pro $0 \le i,j \le 5$.

Dvojný integrál na konvexním čtyřúhelníku D s vrcholy $P_i(x_i, y_i)$, $i = 1,2,3,4$, viz obr. 5.4,



Obr. 5.4. Čtyřúhelník D



Obr. 5.5. Referenční čtverec $\hat{D}$

můžeme pomocí transformace

$$x = x(\xi,\eta) := \sum\nolimits_{i=1}^{4} x_i N_i(\xi,\eta), \qquad y = y(\xi,\eta) := \sum\nolimits_{i=1}^{4} y_i N_i(\xi,\eta),$$

$$\begin{aligned} \text{kde} \qquad & N_1(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1-\eta), \qquad & N_3(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1+\eta), \\ & N_2(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1-\eta), \qquad & N_4(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1+\eta), \end{aligned}$$

převést na integraci na referenčním čtverci $\hat{D} = \langle -1,1\rangle^2$, viz obr.5.5:

$$\int_D f(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_{\hat{D}} \hat{f}(\xi,\eta)|\mathrm{det}\hat{\mathbf{J}}(\xi,\eta)|\,\mathrm{d}\xi\,\mathrm{d}\eta\,, \quad \text{kde} \quad \hat{f}(\xi,\eta) = f(x(\xi,\eta), y(\xi,\eta))$$

a $\hat{\mathbf{J}}(\xi,\eta)$ je Jacobiova matice zobrazení $\hat{D} \mapsto D$,

$$\hat{\mathbf{J}}(\xi,\eta) = \begin{pmatrix} \dfrac{\partial x(\xi,\eta)}{\partial \xi} & \dfrac{\partial x(\xi,\eta)}{\partial \eta} \\ \dfrac{\partial y(\xi,\eta)}{\partial \xi} & \dfrac{\partial y(\xi,\eta)}{\partial \eta} \end{pmatrix}.$$

Integrál na referenčním čtverci $\hat{D}$ spočteme třeba součinovou Gaussovou formulí.

Umíme-li integrovat na jednom konvexním čtyřúhelníku, umíme to také na každé oblasti Ω, kterou lze z konvexních čtyřúhelníků složit. Volbou dostatečně malých čtyřúhelníků lze integrál na Ω spočítat dostatečně přesně.

Jinou možností je vykrytí oblasti Ω pomocí trojúhelníků. Proto je účelné znát dobré formule pro numerickou integraci na trojúhelnících.

Dvojný integrál na trojúhelníku. Uvažujme tedy trojúhelník T s vrcholy P_1, P_2 a P_3. Nejjednodušší formule je analogem obdélníkové formule (5.52). Jde o formuli

$$Q(f) = |T| f(P_0)\,, \tag{5.55}$$

kde $|T|$ je obsah trojúhelníka T a $P_0 = \frac{1}{3}(P_1 + P_2 + P_3)$ je jeho těžiště. Formule (5.55) integruje přesně polynomy stupně 1, tj. je přesná pro $f(x, y) = ax + by + c$, kde a, b, c jsou libovolná čísla. Stejnou přesnost má však tato formule i v případě, kdy T je libovolná ohraničená rovinná oblast. Skutečně, je známo, že pro souřadnice (x_0, y_0) těžiště T_0 oblasti T a pro její plochu $|T|$ platí

$$x_0 = \frac{1}{|T|}\iint_T x\,\mathrm{d}x\,\mathrm{d}y\,, \quad y_0 = \frac{1}{|T|}\iint_T y\,\mathrm{d}x\,\mathrm{d}y\,, \quad 1 = \frac{1}{|T|}\iint_T \mathrm{d}x\,\mathrm{d}y\,.$$

Proto

$$ax_0 + by_0 + c = \frac{1}{|T|}\iint_T [ax + by + c]\,\mathrm{d}x\,\mathrm{d}y$$

a tedy formule (5.55) je pro polynom $ax + by + c$ přesná.

Pro $f(x, y) = ax + by + c$ na trojúhelníku T platí $f(P_0) = \frac{1}{3}[f(P_1) + f(P_2) + f(P_3)]$. Máme proto další formuli

$$Q(f) = \frac{|T|}{3}[f(P_1) + f(P_2) + f(P_3)]\,, \tag{5.56}$$

která je na trojúhelníku T pro polynomy stupně 1 přesná.

Formule

$$Q(f) = \frac{|T|}{3}[f(S_1) + f(S_2) + f(S_3)], \tag{5.57}$$

kde $S_1 = \frac{1}{2}(P_1 + P_2)$, $S_2 = \frac{1}{2}(P_2 + P_3)$ a $S_3 = \frac{1}{2}(P_3 + P_1)$ jsou středy stran trojúhelníka T, integruje přesně polynomy stupně 2, tj. $x^i y^j$, $0 \le i + j \le 2$. Jako cvičení si proveďme

Důkaz. Zobrazení

$$x = x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta\,, \quad y = y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta$$

převádí referenční trojúhelník $\hat{T}$ s vrcholy $\hat{P}_1(0, 0)$, $\hat{P}_2(1, 0)$, $\hat{P}_3(0, 1)$ na trojúhelník T s vrcholy $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$ a platí

$$\iint_T f(x, y)\,\mathrm{d}x\,\mathrm{d}y = 2|T|\int_0^1 \int_0^{1-\xi} \hat{f}(\xi, \eta)\,\mathrm{d}\xi\,\mathrm{d}\eta\,,$$

kde

$$\hat{f}(\xi, \eta) = f(x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta, y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta).$$

Je-li $f(x, y)$ kvadratický polynom v proměnných x, y, je $\hat{f}(\xi, \eta)$ kvadratický polynom v proměnných ξ, η. Proto stačí ověřit, že formule (5.57) integruje přesně polynomy stupně 2 na referenčním trojúhelníku $\hat{T}$.

Je-li $f(x, y)$ lineární polynom, je $\frac{1}{3}[f(S_1) + f(S_2) + f(S_3)] = f(P_0)$, formule (5.57) tedy přechází ve formuli (5.55) a ta je pro lineární polynomy přesná. Zbývá proto prověřit

členy ξ^2, $\xi\eta$, η^2 na $\hat{T}$. Přímým výpočtem zjistíme, že

$$\int_0^1 \int_0^{1-\xi} \xi^2 \,\mathrm{d}\xi\,\mathrm{d}\eta = \frac{1}{12} = \frac{1}{2}\cdot\frac{1}{3}\left[\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 + 0^2\right],$$
$$\int_0^1 \int_0^{1-\xi} \xi\eta \,\mathrm{d}\xi\,\mathrm{d}\eta = \frac{1}{24} = \frac{1}{2}\cdot\frac{1}{3}\left[\frac{1}{2}\cdot 0 + \frac{1}{2}\cdot\frac{1}{2} + 0\cdot\frac{1}{2}\right],$$
$$\int_0^1 \int_0^{1-\xi} \eta^2 \,\mathrm{d}\xi\,\mathrm{d}\eta = \frac{1}{12} = \frac{1}{2}\cdot\frac{1}{3}\left[0^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right]. \quad \square$$

Na trojúhelnících lze konstruovat také kvadraturní formule Gaussova typu, viz [63]. Na referenčním trojúhelníku s vrcholy $\hat{P}_1(0,0)$, $\hat{P}_2(1,0)$ a $\hat{P}_3(0,1)$ lze integrál

$$I(f) = \int_0^1 \int_0^{1-\xi} f(\xi,\eta)\,\mathrm{d}\eta\,\mathrm{d}\xi$$

pomocí transformace

$$\eta = t(1-\xi)$$

převést na tvar

$$I(f) = \int_0^1 \int_0^1 (1-\xi) f(\xi, t(1-\xi))\,\mathrm{d}\xi\,\mathrm{d}t\,.$$

Další transformací

$$\xi = \tfrac{1}{2}(1-u)\,, \qquad t = \tfrac{1}{2}(1-v)$$

dostaneme

$$I(f) = \tfrac{1}{8}\int_{-1}^1 \int_{-1}^1 (1+u) f(\tfrac{1}{2}(1-u), \tfrac{1}{4}(1+u)(1-v))\,\mathrm{d}u\,\mathrm{d}v = \int_{-1}^1 \int_{-1}^1 (1+u)\tilde{f}(u,v)\,\mathrm{d}u\,\mathrm{d}v\,.$$

Tento integrál již lze počítat součinovou kvadraturní formulí:

$$Q(f) = \sum_{i,j=0}^{n} w_i^u w_j^v \tilde{f}(u_i, v_j) = \sum_{i,j=0}^{n} \tfrac{1}{8} w_i^u w_j^v f\left(\tfrac{1}{2}(1-u_i), \tfrac{1}{4}(1+u_i)(1-v_j)\right) = \sum_{i=0}^{(n+1)^2-1} w_i f(\xi_i, \eta_i),$$

kde $Q_u(\psi) := \sum_{i=0}^n w_i^u \psi(u_i)$ je Gaussova formule s váhou $1+u$ na intervalu $\langle -1, 1\rangle$ a $Q_v(\varphi) := \sum_{j=0}^n w_j^v \varphi(v_j)$ je Gaussova-Legendrova formule. Výsledná formule integruje přesně polynomy $\xi^i\eta^j$ pro $0 \le i+j \le 2n+1$. Tak například součinová 2×2 formule na referenčním trojúhelníku $\hat{T}$ je tvaru

$$Q(f) = \sum_{i=0}^{3} w_i f(\xi_i, \eta_i)\,,$$

kde

i	w_i	ξ_i	η_i
0	0,15902 06908 71989	0,15505 10257 21682	0,17855 87282 63616
1	0,09097 93091 28011	0,64494 89742 78318	0,07503 11102 22608
2	0,09097 93091 28011	0,64494 89742 78318	0,28001 99154 99074
3	0,15902 06908 71989	0,15505 10257 21682	0,66639 02460 14701

Tato formule je na referenčním trojúhelníku $\hat{T}$ přesná pro polynomy $\xi^i\eta^j$, kde $0 \le i+j \le 3$.

Populární je také sedmibodová *Radonova* formule řádu 5 (integruje přesně polynomy $\xi^i\eta^j$ pro $0 \le i+j \le 5$). Na referenčním trojúhelníku $\hat{T}$ je tvaru

$$Q(f) = af(t,t) + b[f(r,r) + f(r,s) + f(s,r)] + c[f(u,u) + f(u,v) + f(v,u)],$$

kde

$$r = (6-\sqrt{15})/21, \qquad s = (9+2\sqrt{15})/21, \qquad t = 1/3$$
$$u = (6+\sqrt{15})/21, \qquad v = (9-2\sqrt{15})/21,$$
$$a = 9/80, \qquad b = (155-\sqrt{15})/2400, \qquad c = (155+\sqrt{15})/2400.$$

Další užitečné formule pro integraci na trojúhelnících lze nalézt v [54], [61].

Trojné integrály. Formule pro výpočet trojných integrálů se obvykle uvádějí pro případ, kdy oblast integrace je šestistěn, pětistěn nebo čtyřstěn.

Integrace na referenční krychli. Nejvíce se používají Gaussovy součinové formule. Uvádějí se pro referenční krychli $\hat{K} = \langle -1,1\rangle^3$. Použijeme-li pro integraci v každém ze souřadnicových směrů stejnou Gaussovu-Legendrovu formuli $Q_{Gn}(\varphi) = \sum_{i=0}^{n} w_i\varphi(\xi_i)$, dostaneme součinovou Gaussovu formuli

$$\hat{Q}(g) = \sum_{i,j,k=0}^{n} w_i w_j w_k\, g(\xi_i,\xi_j,\xi_k)\,, \tag{5.58}$$

která na referenční krychli $\hat{K}$ integruje přesně polynomy $\xi^i\eta^j\zeta^k$ pro $0 \le i,j,k \le 2n+1$.

Integraci na konvexním šestistěnu K s vrcholy $P_i(x_i,y_i,z_i)$, $i = 1,\dots,8$, viz obr. 5.6, lze pomocí transformace

$$x = x(\xi,\eta,\zeta) := \sum\nolimits_{i=1}^{8} x_i N_i(\xi,\eta,\zeta),$$
$$y = y(\xi,\eta,\zeta) := \sum\nolimits_{i=1}^{8} y_i N_i(\xi,\eta,\zeta),$$
$$z = z(\xi,\eta,\zeta) := \sum\nolimits_{i=1}^{8} z_i N_i(\xi,\eta,\zeta),$$

kde

$$N_1(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1-\eta)(1-\zeta), \qquad N_5(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1-\eta)(1+\zeta),$$
$$N_2(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1-\eta)(1-\zeta), \qquad N_6(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1-\eta)(1+\zeta),$$
$$N_3(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1+\eta)(1-\zeta), \qquad N_7(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1+\eta)(1+\zeta),$$
$$N_4(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1+\eta)(1-\zeta), \qquad N_8(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1+\eta)(1+\zeta),$$

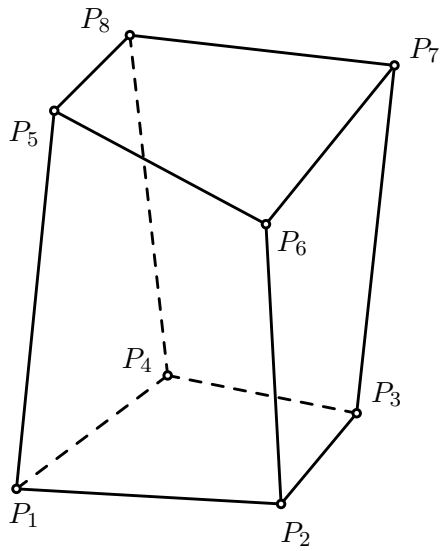
převést na integraci na referenční krychli $\hat{K} = \langle -1, 1\rangle^3$, viz obr.5.7,

$$\int_K f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = \int_{\hat{K}} \hat{f}(\xi,\eta,\zeta)\,|\det \hat{\mathbf{J}}(\xi,\eta,\zeta)|\,\mathrm{d}\xi\,\mathrm{d}\eta\,\mathrm{d}\zeta\,,$$

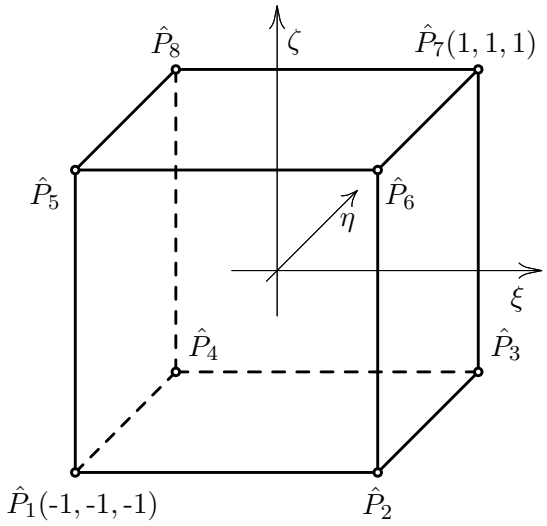
kde $\hat{f}(\xi,\eta,\zeta) = f(x(\xi,\eta,\zeta), y(\xi,\eta,\zeta), z(\xi,\eta,\zeta))$ a $\hat{\mathbf{J}}(\xi,\eta,\zeta)$ je Jacobiova matice zobrazení $\hat{K} \mapsto K$,

$$\hat{\mathbf{J}}(\xi,\eta,\zeta) = \begin{pmatrix} \dfrac{\partial x(\xi,\eta,\zeta)}{\partial\xi} & \dfrac{\partial x(\xi,\eta,\zeta)}{\partial\eta} & \dfrac{\partial x(\xi,\eta,\zeta)}{\partial\zeta} \\ \dfrac{\partial y(\xi,\eta,\zeta)}{\partial\xi} & \dfrac{\partial y(\xi,\eta,\zeta)}{\partial\eta} & \dfrac{\partial y(\xi,\eta,\zeta)}{\partial\zeta} \\ \dfrac{\partial z(\xi,\eta,\zeta)}{\partial\xi} & \dfrac{\partial z(\xi,\eta,\zeta)}{\partial\eta} & \dfrac{\partial z(\xi,\eta,\zeta)}{\partial\zeta} \end{pmatrix}.$$

Integrál na referenční krychli $\hat{K}$ pak spočteme Gaussovou součinovou formulí (5.58).



Obr. 5.6. Šestistěn K



Obr. 5.7. Referenční krychle $\hat{K}$

Integrace na jehlanu. Na čtyřbokém jehlanu S s vrchloly P_1, P_2, P_3 a P_4 se používá formule

$$Q(f) = |V| f(P_0)\,, \tag{5.59}$$

kde $|V|$ je objem jehlanu a $P_0 = \frac{1}{4}[P_1 + P_2 + P_3 + P_4]$ je jeho těžiště. Formule (5.58) integruje přesně lineární polynomy $ax + by + cz + d$. To umí také formule

$$Q(f) = \frac{|V|}{4}[f(P_1) + f(P_2) + f(P_3) + f(P_4)]\,. \tag{5.60}$$

Formuli řádu 2 si uvedeme pro referenční trojboký jehlan $\hat{S}$ s vrcholy $\hat{P}_1(0,0,0)$, $\hat{P}_2(1,0,0)$, $\hat{P}_3(0,1,0)$, $\hat{P}_4(0,0,1)$. Formule je tvaru

$$Q(g) = \frac{1}{24}[g(\alpha,\alpha,\alpha) + g(\beta,\alpha,\alpha) + g(\alpha,\beta,\alpha) + g(\alpha,\alpha,\beta)]\,,$$
$$\text{kde} \quad \alpha = 0{,}25 - \sqrt{0{,}0125}, \quad \beta = 1 - 3\alpha\,, \tag{5.61}$$

a na referenčním jehlanu $\hat{S}$ integruje přesně polynomy $\xi^i\eta^j\zeta^k$, kde $0 \le i+j+k \le 2$. Tuto a další užitečné formule lze najít v [54], [61].

6. 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.

6.1. 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 6.1. Získáme hrubý odhad kořenů rovnice $f(x) = 0$, kde

$$f(x) = 4\sin x - x^3 - 1\,.$$

Z obrázku 6.1 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

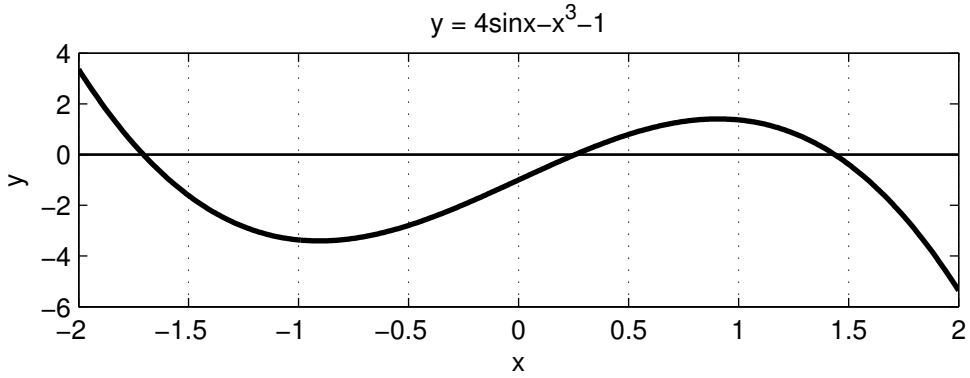
Metoda bisekce známá také jako *metoda půlení intervalů*. Předpokládejme, že funkce $f(x)$ má v koncových bodech intervalu (a_0, b_0) opačná znaménka, tj. platí $f(a_0)f(b_0) < 0$. Sestrojíme posloupnost intervalů $(a_1, b_1) \supset (a_2, b_2) \supset (a_3, b_3) \supset \ldots$, které obsahují kořen. 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

$$(a_{k+1}, b_{k+1}) = \begin{cases} (a_k, x_{k+1})\,, & \text{když} \quad f(a_k)f(x_{k+1}) < 0\,, \\ (x_{k+1}, b_k)\,, & \text{když} \quad f(a_k)f(x_{k+1}) > 0\,. \end{cases} \tag{6.1}$$

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. 6.1: Graf funkce $4\sin x - x^3 - 1$

Střed x_{k+1} intervalu (a_k, b_k) aproximuje kořen x^* s chybou

$$|x_{k+1} - x^*| \leq \tfrac{1}{2}(b_k - a_k) = 2^{-k-1}(b_0 - a_0)\,. \tag{6.2}$$

Pro $k \to \infty$ zřejmě $|I_k| \to 0$ a $x_k \to x^*$.

Příklad 6.2. Metodu bisekce aplikujeme na rovnici z příkladu 6.1. 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 (6.2) 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.

6.2. 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

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

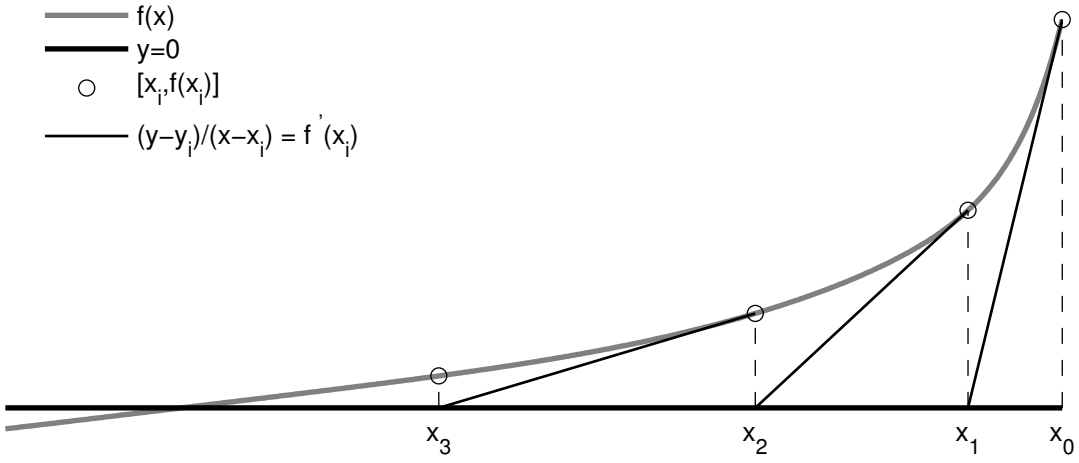
tedy dosadíme $y := 0$, vypočteme x a položíme $x_{k+1} := x$. Tak dostaneme předpis

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}\,. \tag{6.3}$$

Výpočet ukončíme a x_{k+1} považujeme za dostatečně přesnou aproximaci kořene, pokud

$$|x_{k+1} - x_k| \le \varepsilon\,, \quad \text{případně} \quad |x_{k+1} - x_k| \le \varepsilon|x_k| \quad \text{nebo} \quad |f(x_{k+1})| \le \varepsilon\,, \tag{6.4}$$

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. 6.2: Newtonova metoda

Příklad 6.3. Newtonovou metodou určíme kladný kořen rovnice z příkladu 6.1. 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

$$e_{k+1} = \frac{1}{2}\,\frac{f''(\xi)}{f'(x_k)}\,e_k^2\,, \tag{6.5}$$

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

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = C\,, \tag{6.6}$$

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 (6.5) 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 [40]. 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

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

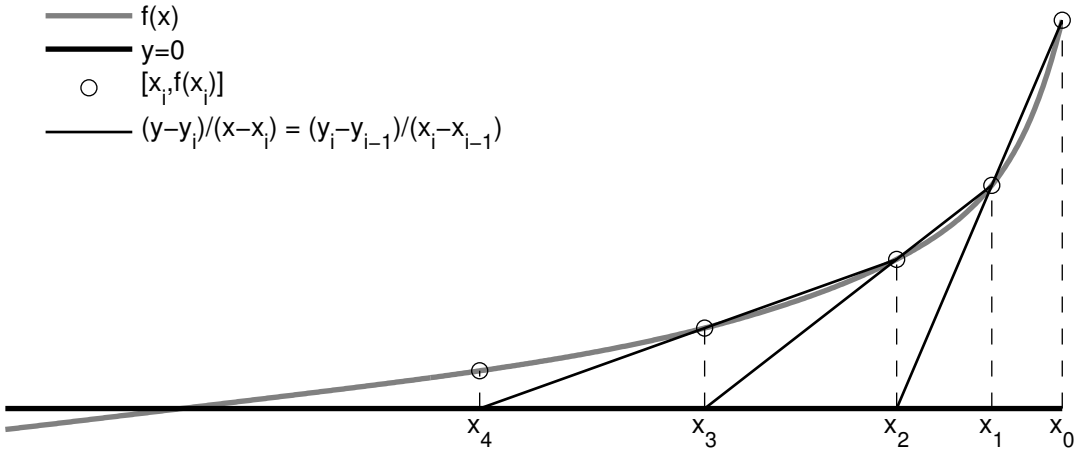
Tak dostaneme *metodu sečen*: zadáme dvě počáteční aproximace x_0, x_1 a počítáme $x_2, x_3, \ldots$ podle předpisu

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)\,. \tag{6.7}$$

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. 6.3: 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.

Příklad 6.4. Metodou sečen určíme kladný kořen rovnice z příkladu 6.1. 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

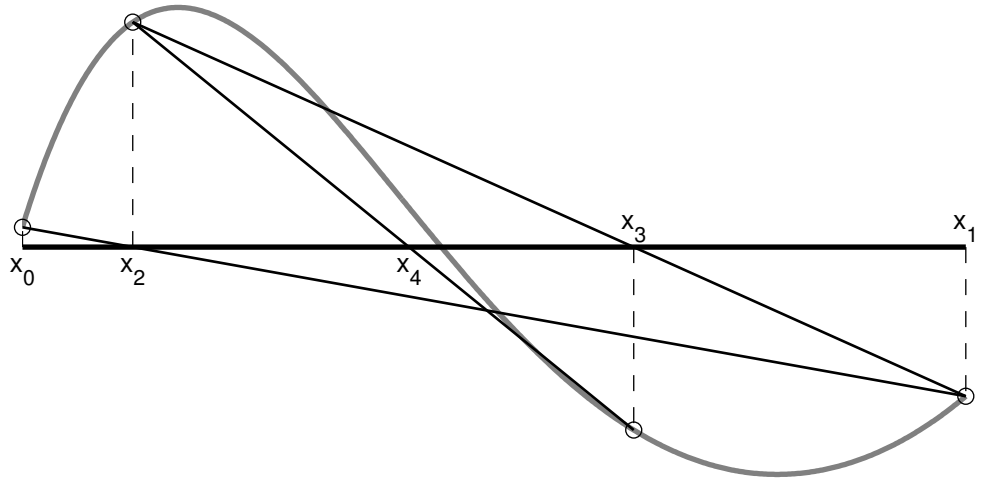
Metoda regula falsi. Počáteční aproximace x_0 a x_1 se volí tak, aby $f(x_0)f(x_1) < 0$. Nová aproximace x_{k+1} se opět získá jako průsečík sečny s osou x. Sečna však tentokrát spojuje bod $[x_k, f(x_k)]$ s bodem $[x_\ell, f(x_\ell)]$, kde ℓ je největší index, pro který $f(x_k)f(x_\ell) < 0$. Výpočet tedy probíhá podle vzorce

$$x_{k+1} = x_k - \frac{x_k - x_\ell}{f(x_k) - f(x_\ell)} f(x_k)\,, \qquad k = 1, 2, \ldots . \tag{6.8}$$

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. 6.4: Regula falsi

Příklad 6.5. Metodou regula falsi určíme kladný kořen rovnice z příkladu 6.1. 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

$$x_{k+1} = x_k - \frac{f(x_k)}{d_k}, \qquad \text{kde} \quad d_k = \frac{f(x_k + f(x_k)) - f(x_k)}{f(x_k)} \tag{6.9}$$

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 [21].

Brentova metoda. Metoda inverzní kvadratické interpolace spolu s metodou sečen a metodou bisekce jsou základem populární *Brentovy metody*, viz např. [37], dále také program `zbrent` v [40] 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 [37], [40].

Příklad 6.6. Budeme hledat kladný kořen rovnice z příkladu 6.1 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 [37].

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 6.7. 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

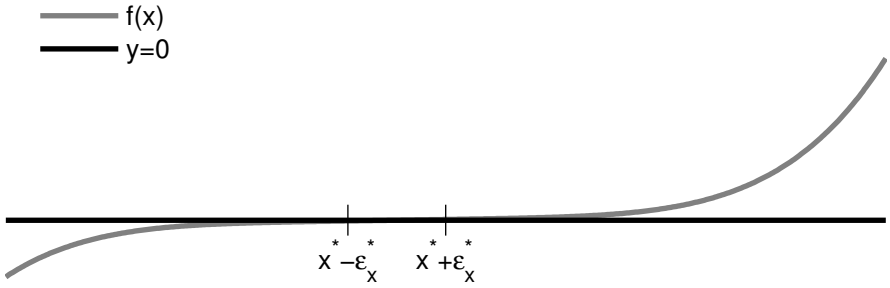
$$f(x_k) = f(x_k) - f(x^*) = f'(\xi)(x_k - x^*)\,,$$

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. 6.5. V takovém případě je výpočet kořene x^* *špatně podmíněný problém*: malá změna f vyvolá velkou změnu x^*.



Obr. 6.5: 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 [55]. □

6.3. 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

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n) = 0\,, \\ &f_2(x_1, x_2, \ldots, x_n) = 0\,, \\ &\vdots \\ &f_n(x_1, x_2, \ldots, x_n) = 0\,, \end{aligned} \qquad \text{nebo-li} \qquad \mathbf{f}(\mathbf{x}) = \mathbf{o}\,, \tag{6.10}$$

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 (6.10) 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. Newtonovu metodu odvodíme z Taylorova rozvoje

$$\mathbf{o} = \mathbf{f}(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}_k) + \mathbf{f}'(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k) + \cdots \doteq \mathbf{f}(\mathbf{x}_k) + \mathbf{f}'(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k)$$

tak, že přibližnou rovnost nahradíme rovností a místo $\mathbf{x}^*$ píšeme $\mathbf{x}_{k+1}$, tj. počítáme

$$\mathbf{f}'(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)\,, \tag{6.11}$$

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

$$\mathbf{f}'(\mathbf{x}_k)\,\mathbf{d}_k = -\mathbf{f}(\mathbf{x}_k) \qquad \text{a pak určíme} \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k\,. \tag{6.12}$$

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

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon\,, \quad \|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon\|\mathbf{x}_k\| \quad \text{nebo} \quad \|\mathbf{f}(\mathbf{x}_{k+1})\| \le \varepsilon\,, \tag{6.13}$$

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 (6.11), 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

$$\mathbf{f}'(\mathbf{x}_p)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)\,, \quad k = p, p+1, \dots, p+m-1\,, \quad p = 0, m, 2m, \dots.$$

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 $k = 0, m, 2m, \dots$. 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 $k \notin \{0, m, 2m, \dots\}$ tedy provádíme jen „laciné“ zpětné chody. V propracovaných algoritmech se přepočet Jacobiho matice nevolí staticky, tj. každých m kroků, ale dynamicky, tj. pro $p = 0 < p_1 < p_2 < \dots$, 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,

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

kde $h_j \neq 0$ jsou vhodně zvolené parametry, $\mathbf{h} = (h_1, h_2, \dots, h_n)^T$. 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 $\boldsymbol{\Delta}(\mathbf{x}, \mathbf{h})$ s prvky $\Delta_{ij}(\mathbf{x}, \mathbf{h})$ je aproximací Jacobiho matice $\mathbf{f}'(\mathbf{x})$. Když tedy v (6.11) nahradíme $\mathbf{f}'(\mathbf{x}_k)$ pomocí $\boldsymbol{\Delta}(\mathbf{x}_k, \mathbf{h}_k)$, dostaneme *diskretizovanou Newtonovu metodu*

$$\boldsymbol{\Delta}(\mathbf{x}_k, \mathbf{h}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)\,. \tag{6.14}$$

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 (6.14) totožná s předpisem (6.7)) a *zobecněnou Steffensenovu metodu* obdržíme, když položíme $h_j^{(k)} = f_j(\mathbf{x}_k)$ (pro $n = 1$ je pak rovnice (6.14) totožná s předpisem(6.9)). 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 6.8. 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 (6.12) 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 6.6 zjistíme, že soustava má celkem tři kořeny. Vybereme si třeba ten, který leží ve čtverci $\Omega := \{[x,y] \mid -2 \le x \le -1, 1 \le y \le 2\}$, a jako počáteční aproximaci zvolíme $x_0 = -1$, $y_0 = 1$. Výpočet ukončíme, když

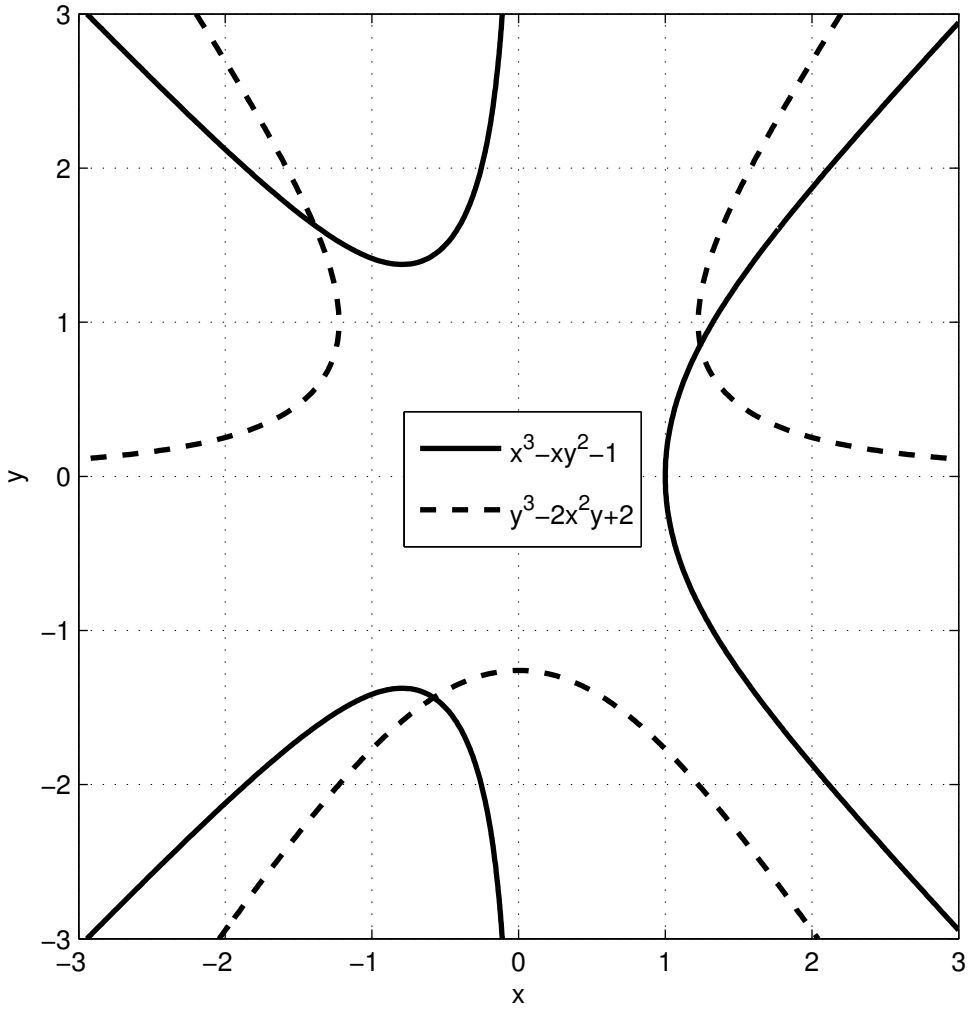
$$\|\mathbf{f}(\mathbf{x}_{k+1})\|_\infty = \max\{|f(x_{k+1}, y_{k+1})|\,; |g(x_{k+1}, y_{k+1})|\} < 10^{-5}\,.$$

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

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.



Obr. 6.6: Soustava dvou nelineárních rovnic

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 (6.12).

Podrobnější (a mnohé další) informace k tomuto tématu čtenář najde ve specializované literatuře, např. v [39].

Metoda prosté iterace. Nechť $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_n(\mathbf{x}))^T$ je vektorová funkce definovaná a spojitá v uzavřené oblasti Ω, tj. $g_i \in C(\Omega)$, $i = 1, 2, \ldots, n$, a nechť $\mathbf{g}$ splňuje následující dvě podmínky:

(α) $\mathbf{g}(\mathbf{x}) \in \Omega \qquad \forall \mathbf{x} \in \Omega,$

(β) existuje číslo q, $0 \le q < 1$, takové, že $\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \le q\|\mathbf{x} - \mathbf{y}\| \qquad \forall \mathbf{x}, \mathbf{y} \in \Omega.$

Pak rovnice $\mathbf{x} = \mathbf{g}(\mathbf{x})$ má v Ω jediné řešení $\mathbf{x}^*$ a *posloupnost postupných aproximací* $\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k)$, $k = 0, 1, \ldots$, konverguje k $\mathbf{x}^*$ pro každé $\mathbf{x}_0 \in \Omega$. Přitom

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \le q\|\mathbf{x}_k - \mathbf{x}^*\|,$$

tj. rychlost obecně jen lineární konvergence závisí na q, viz [20].

Řekneme, že Ω je *konvexní oblast*, jestliže s každými dvěma body obsahuje také úsečku, která tyto body spojuje, tj.

$$\mathbf{x}, \mathbf{y} \in \Omega \quad \Longrightarrow \quad \overline{\mathbf{xy}} := \{\mathbf{x} + t(\mathbf{y} - \mathbf{x}) \,|\, 0 \le t \le 1\} \in \Omega.$$

Jestliže Ω je uzavřená konvexní oblast a $g_i \in C^1(\Omega)$, $i = 1, 2, \ldots, n$, pak místo podmínky (β) můžeme použít silnější podmínku

(β') $\|\mathbf{g}'(\mathbf{x})\| \le q < 1 \qquad \forall \mathbf{x} \in \Omega.$

Skutečně, podle věty o střední hodnotě, viz [20],

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \le \max_{0 \le t \le 1} \|\mathbf{g}'(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))\| \cdot \|\mathbf{x} - \mathbf{y}\| \le q\|\mathbf{x} - \mathbf{y}\|.$$

Jestliže $\Omega = \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}^*\| < \varepsilon\}$, $\varepsilon > 0$, pak podmínka (α) plyne z podmínky (β):

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{x}^*\| = \|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}^*)\| \le q\|\mathbf{x} - \mathbf{x}^*\| \quad \Longrightarrow \quad (\mathbf{x} \in \Omega \Longrightarrow \mathbf{g}(\mathbf{x}) \in \Omega)\,.$$

Přibližný výpočet kořene $\mathbf{x}^*$ rovnice $\mathbf{x} = \mathbf{g}(\mathbf{x})$ podle formule $\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k)$ se nazývá *metoda prosté iterace* nebo také *metoda postupných aproximací*. Podmínky (α) a (β) nebo (β') jsou postačující pro konvergenci této metody.

Příklad 6.9. 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 (α), (β).

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 (α) je splněna.

Abychom ověřili podmínku(β), 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 (β) 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 (α), (β) 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 (2.32).

7. Výpočet vlastních čísel a vlastních vektorů

7.1. Základní vlastnosti

Nechť $\mathbf{A}$ je čtvercová matice řádu n. Úlohu

najít nenulový vektor $\mathbf{x}$ a skalár λ tak, aby platilo $\qquad \mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ (7.1)

budeme nazývat *problém vlastních čísel*. λ je *vlastní číslo* a $\mathbf{x}$ je odpovídající *pravý vlastní vektor*, stručně *vlastní vektor*. K vlastnímu číslu λ přísluší také *levý vlastní vektor* $\mathbf{y} \neq \mathbf{o}$ splňující $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$. Protože $\mathbf{A}^T\mathbf{y} = \lambda\mathbf{y}$, je $\mathbf{y}$ pravý vlastní vektor matice $\mathbf{A}^T$. Při popisu metod pro výpočet vlastních čísel a vektorů se proto omezíme na pravé vlastní vektory.

V následující kapitole 7.1.1 ukážeme, že pro reálnou matici jsou vlastní čísla a vlastní vektory obecně komplexní. Je proto přirozené definovat problém vlastních čísel pro komplexní matice, pak se vše odehrává v komplexních oboru, matice, vlastní číslo i vlastní vektor jsou komplexní. Protože se však ve většině aplikací vyskytují jen reálné matice, zaměříme se právě na ně. V dalším proto budeme předpokládat, že matice $\mathbf{A}$ je reálná. Množinu všech vlastních čísel matice $\mathbf{A}$ označíme $\lambda(\mathbf{A})$ a nazveme ji *spektrum* matice $\mathbf{A}$.

7.1.1. Existence a jednoznačnost

Rovnici $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ lze ekvivalentně zapsat ve tvaru

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{o}\,. \tag{7.2}$$

To je homogenní SLR, která má nenulové řešení tehdy a jen tehdy, když je matice soustavy singulární. Vlastní čísla matice $\mathbf{A}$ tedy musí splňovat *charakteristickou rovnici*

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0\,. \tag{7.3}$$

Zde $p(\lambda) \equiv \det(\mathbf{A} - \lambda\mathbf{I})$ je polynom stupně n v proměnné λ, tzv. *charakteristický polynom* matice $\mathbf{A}$. Vlastní čísla matice $\mathbf{A}$ jsou tedy kořeny jejího charakteristického polynomu. Ze základní věty algebry plyne, že polynom

$$p(\lambda) = c_n\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0 = c_n(\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n)\,, \qquad c_n = (-1)^n,$$

má přesně n kořenů. Matice $\mathbf{A}$ řádu n má tedy n vlastních čísel, která ale nemusejí být reálná a nemusejí být navzájem různá. Komplexní vlastní čísla se vyskytují vždy v komplexně sdružených dvojicích: když $\lambda = \alpha + i\beta$, kde $i = \sqrt{-1}$ a $\beta \neq 0$, je vlastní číslo, pak také $\bar{\lambda} = \alpha - i\beta$ je vlastní číslo.

Vlastní vektory získáme řešením SLR (7.2). Tyto soustavy mají pro každé λ nekonečně mnoho řešení $\mathbf{x} \in N(\mathbf{A} - \lambda\mathbf{I})$, kde $N(\mathbf{A} - \lambda\mathbf{I})$ je jádro matice $\mathbf{A} - \lambda\mathbf{I}$, viz (3.3). Tedy zatímco vlastní číslo $\lambda \in \lambda(\mathbf{A})$ je určeno jednoznačně jako jeden z kořenů charakteristického polynomu matice $\mathbf{A}$, odpovídající vlastní vektor $\mathbf{x} \in N(\mathbf{A} - \lambda\mathbf{I})$ jednoznačně určen není.

Vlastní čísla matic počítáme iteračně, neboť kořeny obecného polynomu stupně většího než čtyři jinak určit neumíme. To by však neměl být problém, neboť je známa řada

kvalitních numerických metod pro výpočet kořenů polynomů. Na první pohled je tedy řešení problému vlastních čísel snadné: stačí vhodnou numerickou metodou najít všechny kořeny charakteristického polynomu a pak řešením příslušných soustav lineárních rovnic dopočítat odpovídající vlastní vektory. K praktickému výpočtu vlastních čísel matic však postup založený na řešení charakteristické rovnice obecně vhodný není. Zde je několik důvodů:

- Výpočet koeficientů charakteristické rovnice je výpočetně náročný.
- Koeficienty charakteristické rovnice jsou velmi citlivé na malé změny v koeficientech matice.
- Zaokrouhlovací chyby vznikající při výpočtu koeficientů charakteristického polynomu mohou způsobit, že kořeny numericky sestaveného charakteristického polynomu se budou podstatně lišit od kořenů přesného charakteristického polynomu.
- Výpočet kořenů polynomů vysokých stupňů je obtížná úloha.

Proto transformace

$$\text{matice} \quad \longrightarrow \quad \text{charakteristický polynom} \quad \longrightarrow \quad \text{vlastní čísla}$$

nevytváří podstatně jednodušší „meziproblém“ a je obecně numericky nestabilní.

Vzhledem k existenci velmi kvalitních a stabilních algoritmů pro výpočet vlastních čísel, s některými z nichž se seznámíme v dalších kapitolách, se používá postup opačný: k danému polynomu p se sestaví přidružená matice $\mathbf{A}$, jejíž charakteristiký polynom je roven p, a vypočtou se vlastní čísla matice $\mathbf{A}$, což jsou hledané kořeny polynomu p. Pro $p(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ lze přidruženou matici zvolit např. ve tvaru

$$\mathbf{A} = \begin{pmatrix} -c_{n-1} & -c_{n-2} & \dots & -c_1 & -c_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix},$$

tj. v prvním řádku jsou zápornými znaménky opatřené koeficienty polynomu, v první poddiagonále jsou jedničky a ostatní prvky jsou nulové. V MATLABu takto počítá kořeny polynomu funkce `roots`.

7.1.2. Násobnost a diagonalizovatelnost

Algebraická násobnost vlastního čísla je jeho násobnost jako kořene charakteristické rovnice. Jestliže je algebraická násobnost vlastního čísla rovna jedné, říkáme, že vlastní číslo je *jednoduché*, a je-li algebraická násobnost větší než jedna, hovoříme o *násobném* vlastním čísle.

Geometrická násobnost vlastního čísla je počet lineárně nezávislých vlastních vektorů příslušných tomuto vlastnímu číslu. Geometrická násobnost je tedy vždy menší nebo rovna násobnosti algebraické.

Řekneme, že *vlastní číslo* je *defektní*, když jeho geometrická násobnost je menší než násobnost algebraická. Je-li algebraická i geometrická násobnost vlastního čísla stejná, říkáme, že *vlastní číslo* je *nedefektní*. Matice, která má defektní vlastní číslo, se nazývá *defektní matice*. Matice, jejíž všechna vlastní čísla jsou nedefektní, se nazývá *nedefektní matice*. Dá se ukázat, že nedefektní matice $\mathbf{A}$ řádu n má n lineárně nezávislých vlastních vektorů $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ příslušných k vlastním číslům $\lambda_1, \lambda_2, \ldots, \lambda_n$ (říkáme také, že matice má *úplný systém vlastních vektorů*). Nechť $\mathbf{D} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ a $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, pak $\mathbf{X}$ je regulární a platí

$$\mathbf{AX} = \mathbf{XD}\,, \qquad \text{takže} \qquad \mathbf{X}^{-1}\mathbf{AX} = \mathbf{D} \qquad \text{a} \qquad \mathbf{A} = \mathbf{XDX}^{-1}\,.$$

Vidíme tedy, že nedefektní matici $\mathbf{A}$ lze transformací $\mathbf{X}^{-1}\mathbf{AX}$ převést na matici diagonální. Matice s touto vlastností se nazývají *diagonalizovatelné*.

Vyjádření matice $\mathbf{A}$ ve tvaru $\mathbf{A} = \mathbf{XDX}^{-1}$, kde $\mathbf{D}$ je diagonální, nazýváme spektrálním rozkladem matice $\mathbf{A}$ (anglicky spectral decomposition, častěji se však používá přiléhavější termín eigenvalue decomposition, který bohužel nemá vhodný český ekvivalent).

7.1.3. Lokalizace vlastních čísel

Symetrická matice je diagonalizovatelná, má reálná vlastní čísla, vlastní vektory příslušné různým vlastním číslům jsou navzájem ortogonální, vlastní vektory příslušné násobným vlastním číslům lze ortogonalizovat. Symetrická matice $\mathbf{A}$ je tedy diagonalizovatelná pomocí ortonormální matice vlastních vektorů, tj. existuje ortonormální matice $\mathbf{X}$ vlastních vektorů s vlastností $\mathbf{X}^T\mathbf{AX} = \mathbf{D}$. *Pozitivně definitní* matice je symetrická matice s kladnými vlastními čísly. Důkaz uvedených tvrzení viz např. [30].

Protože spektrální poloměr $\varrho(\mathbf{A}) \leq \|\mathbf{A}\|$, viz (2.31), všechna vlastní čísla matice $\mathbf{A}$ leží v komplexní rovině v kruhu se středem v počátku a poloměrem $\|\mathbf{A}\|$.

Geršgorinova věta. Vlastní čísla matice $\mathbf{A} = \{a_{ij}\}_{i,j=1}^n$ jsou všechna obsažena ve sjednocení kruhů se středy a_{kk} a poloměry $\sum_{j=1, j\neq k}^n |a_{kj}|$.

Důkaz. Nechť λ je vlastní číslo a $\mathbf{x}$ je odpovídající vlastní vektor normalizovaný tak, že $\|\mathbf{x}\|_\infty = 1$. Nechť x_k je složka vektoru $\mathbf{x}$ taková, že $|x_k| = 1$. Protože $\mathbf{Ax} = \lambda\mathbf{x}$, je

$$(\lambda - a_{kk})x_k = \sum_{j\neq k} a_{kj}x_j\,,$$

takže

$$|\lambda - a_{kk}| \leq \sum_{j\neq k} |a_{kj}| \cdot |x_j| \leq \sum_{j\neq k} |a_{kj}|\,. \qquad \square$$

Aplikací této věty na $\mathbf{A}^T$ vidíme, že podobný výsledek platí také pro kruhy se středy a_{kk} a poloměry $\sum_{i=1, i\neq k}^n |a_{ik}|$.

7.1.4. Podmíněnost

vlastních čísel a vlastních vektorů matice vyjadřuje jejich citlivost vůči malým změnám koeficientů matice.

Jako reprezentativní ilustraci prozkoumáme citlivost vlastních čísel diagonizovatelné matice $\mathbf{A}$, takže $\mathbf{X}^{-1}\mathbf{A}\mathbf{X} = \mathbf{D}$, kde $\mathbf{X}$ je matice vlastních vektorů a $\mathbf{D}$ je diagonální matice příslušných vlastních čísel. Nechť $\Delta\mathbf{A} \equiv \mathbf{E}$ je malá změna matice $\mathbf{A}$, μ je libovolné vlastní číslo pozměněné matice $\mathbf{A}+\mathbf{E}$ a $\mathbf{F} = \mathbf{X}^{-1}\mathbf{E}\mathbf{X}$. Pak

$$\mathbf{X}^{-1}(\mathbf{A}+\mathbf{E})\mathbf{X} = \mathbf{X}^{-1}\mathbf{A}\mathbf{X} + \mathbf{X}^{-1}\mathbf{E}\mathbf{X} = \mathbf{D}+\mathbf{F}\,,$$

takže matice $\mathbf{A}+\mathbf{E}$ a $\mathbf{D}+\mathbf{F}$ jsou podobné a tedy mají stejná vlastní čísla (viz kapitola 7.1.5). Proto existuje vlastní vektor $\mathbf{v}$ matice $\mathbf{D}+\mathbf{F}$ příslušný vlastnímu číslu μ, takže $(\mathbf{D}+\mathbf{F})\mathbf{v} = \mu\mathbf{v}$, což můžeme přepsat do tvaru

$$\mathbf{v} = (\mu\mathbf{I}-\mathbf{D})^{-1}\mathbf{F}\mathbf{v}$$

za předpokladu, že μ není vlastní číslo $\mathbf{D}$ (a tedy ani $\mathbf{A}$, což je přijatelný předpoklad odpovídající situaci, kdy změna $\mathbf{A}$ vyvolá změnu všech vlastních čísel). Odtud

$$\|\mathbf{v}\|_p \le \|(\mu\mathbf{I}-\mathbf{D})^{-1}\|_p \cdot \|\mathbf{F}\|_p \cdot \|\mathbf{v}\|_p \qquad \text{a tedy} \qquad \|(\mu\mathbf{I}-\mathbf{D})^{-1}\|_p^{-1} \le \|\mathbf{F}\|_p\,.$$

Protože pro diagonální matici $\mathbf{C} = \mathrm{diag}(c_1, c_2, \ldots, c_n)$ a $p \in \langle 1, \infty)$ je $\|\mathbf{C}\|_p = \max|c_i|$, dostáváme $\|(\mu\mathbf{I}-\mathbf{D})^{-1}\|_p = 1/|\mu-\lambda_k|$, kde λ_k je vlastní číslo $\mathbf{D}$ (a tedy $\mathbf{A}$) nejbližší k μ. Celkem tedy

$$|\mu-\lambda_k| = \|(\mu\mathbf{I}-\mathbf{D})^{-1}\|_p^{-1} \le \|\mathbf{F}\|_p = \|\mathbf{X}^{-1}\mathbf{E}\mathbf{X}\|_p \le \|\mathbf{X}^{-1}\|_p \cdot \|\mathbf{E}\|_p \cdot \|\mathbf{X}\|_p = \kappa_p(\mathbf{X})\|\mathbf{E}\|_p\,.$$

Ukázali jsme tedy, že *když* $\mathbf{A}$ *je diagonizovatelná, pak*

pro každé $\mu \in \lambda(\mathbf{A}+\Delta\mathbf{A})$ *existuje* $\lambda \in \lambda(\mathbf{A})$ *takové, že* $|\mu-\lambda| \le \kappa_p(\mathbf{X})\|\Delta\mathbf{A}\|_p$.

Tento výsledek (známý jako *Bauerova-Fikova věta*, viz např. [42]) říká, že

citlivost vlastních čísel diagonizovatelné matice $\mathbf{A}$ *lze odhadnout číslem podmíněnosti* $\kappa_p(\mathbf{X}) = \|\mathbf{X}\|_p \cdot \|\mathbf{X}^{-1}\|_p$ *matice jejích vlastních vektorů.*

Odtud plyne, že vlastní čísla matice mohou být velmi citlivá na změny jejích koeficientů, pokud jsou vlastní vektory matice téměř lineárně závislé (tj. je-li matice téměř defektní).

Skvělá situace nastává pro symetrické matice: k nim totiž vždy existuje ortonormální matice $\mathbf{X}$ vlastních vektorů, takže $\kappa_2(\mathbf{X}) = 1$, což znamená, že *vlastní čísla symetrických matic jsou vždy dobře podmíněná.*

Bauerova-Fikova věta postihuje citlivost všech vlastních čísel jedním vzorcem. Citlivost jednotlivých vlastních čísel však může být značně rozdílná. Prozkoumejme tedy citlivost individuálního vlastního čísla λ na malou změnu $\Delta\mathbf{A} \equiv \mathbf{E}$ matice $\mathbf{A}$. Nechť $\mathbf{x}$ resp. $\mathbf{y}$ je pravý resp. levý vlastní vektor příslušný vlastnímu číslu λ a uvažujme pozměněný problém

$$(\mathbf{A}+\mathbf{E})(\mathbf{x}+\Delta\mathbf{x}) = (\lambda+\Delta\lambda)(\mathbf{x}+\Delta\mathbf{x})\,.$$

Výrazy na levé a pravé straně roznásobíme. Předpokládejme, že vlastní číslo λ je jednoduché. Pak lze členy $\mathbf{E}\Delta\mathbf{x}$ a $\Delta\lambda\Delta\mathbf{x}$ považovat za dostatečně malé a zanedbat je, viz [42]. Provedeme-li to a využijeme toho, že $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, dostaneme přibližnou rovnost

$$\mathbf{A}\Delta\mathbf{x} + \mathbf{E}\mathbf{x} \approx \lambda\Delta\mathbf{x} + \Delta\lambda\mathbf{x}\,.$$

Pronásobíme-li obě strany $\mathbf{y}^T$, máme

$$\mathbf{y}^T\mathbf{A}\Delta\mathbf{x} + \mathbf{y}^T\mathbf{E}\mathbf{x} \approx \lambda\mathbf{y}^T\Delta\mathbf{x} + \Delta\lambda\mathbf{y}^T\mathbf{x}\,.$$

Protože $\mathbf{y}$ je levý vlastní vektor, $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$, a odtud

$$\mathbf{y}^T\mathbf{E}\mathbf{x} \approx \Delta\lambda\mathbf{y}^T\mathbf{x}\,.$$

Pro jednoduché vlastní číslo $\mathbf{y}^T\mathbf{x} \neq 0$, viz [17], takže

$$\Delta\lambda \approx \frac{\mathbf{y}^T\mathbf{E}\mathbf{x}}{\mathbf{y}^T\mathbf{x}}\,.$$

Odtud pro $|\Delta\lambda|$ obdržíme přibližný odhad

$$|\Delta\lambda| \lessapprox \frac{\|\mathbf{y}\|_2 \cdot \|\mathbf{x}\|_2}{|\mathbf{y}^T\mathbf{x}|}\,\|\mathbf{E}\|_2 = \frac{1}{|\cos\theta_\lambda|}\,\|\mathbf{E}\|_2\,,$$

kde θ_λ je úhel, který svírají levý a pravý vlastní vektor příslušný k vlastnímu číslu λ.

Ukázali jsme tedy, že *pro jednoduché vlastní číslo λ matice $\mathbf{A}$ a odpovídající vlastní číslo $\lambda + \Delta\lambda$ mírně pozměněné matice $\mathbf{A} + \Delta\mathbf{A}$ přibližně platí*

$$|\Delta\lambda| \lesssim \kappa(\lambda)\|\Delta\mathbf{A}\|_2\,, \quad \text{kde} \quad \kappa(\lambda) = \frac{1}{|\cos\theta_\lambda|} \tag{7.4}$$

je tak zvané číslo podmíněnosti vlastního čísla λ matice $\mathbf{A}$.

Všimněte si, že pro symetrickou matici a jednoduché vlastní číslo $\mathbf{x} = \mathbf{y}$, takže $\theta_\lambda = 0$ a příslušné číslo podmíněnosti $\kappa(\lambda)$ je rovno jedné. Jsou-li však levý a pravý vlastní vektor téměř kolmé, je číslo podmíněnosti $\kappa(\lambda)$ velké, takže změna $\Delta\lambda$ vlastního čísla λ vyvolaná malou změnou $\Delta\mathbf{A}$ matice $\mathbf{A}$ je velká. V MATLABu počítá čísla podmíněnosti $\kappa(\lambda_i)$ všech vlastních čísel λ_i, $i = 1, 2, \ldots, n$, funkce `condeig`.

Analýza podmíněnosti násobných vlastních čísel komplikovanější. Platí například, že násobná nebo téměř násobná vlastní čísla jsou špatně podmíněná. Také analýza citlivosti vlastních vektorů je poměrně složitá. Ukazuje se, že když matice má dobře podmíněná a navzájem dostatečně vzdálená vlastní čísla, pak jsou vlastní vektory dobře podmíněné, [42]. Jsou-li však vlastní čísla špatně podmíněná, nebo když existují skupiny navzájem blízkých vlastních čísel, pak jsou vlastní vektory špatně podmíněné.

Podmíněnost vlastních čísel lze někdy zlepšit pomocí tzv. vyvážení matice (anglicky „balancing“), viz kapitola 7.2.5, v MATLABu funkce `balance`.

7.1.5. Transformace.

Mnohé numerické metody pro výpočet vlastních čísel a vektorů jsou založeny na redukci původní matice na jinou jednodušší matici, jejíž vlastní čísla a vlastní vektory lze vypočítat snadněji. Uvedeme si proto několik transformací, které vlastní čísla a vlastní vektory buďto nemění nebo umožňují jejich snadnou rekonstrukci.

Posun znamená odečtení konstanty od diagonálních prvků matice. Jestliže $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ a μ je konstanta, pak $(\mathbf{A} - \mu\mathbf{I})\mathbf{x} = (\lambda - \mu)\mathbf{x}$. Tedy vlastní čísla matice $\mathbf{A} - \mu\mathbf{I}$ se oproti vlastním číslům matice $\mathbf{A}$ posunou o μ, vlastní vektory se však nezmění.

Inverze. Je-li $\mathbf{A}$ regulární a $\mathbf{Ax} = \lambda\mathbf{x}$, pak $\lambda \neq 0$ a $\mathbf{A}^{-1}\mathbf{x} = (1/\lambda)\mathbf{x}$. Tedy vlastní čísla matice $\mathbf{A}^{-1}$ jsou převrácené hodnoty vlastních čísel matice $\mathbf{A}$, vlastní vektory obou matic jsou stejné.

Mocniny. Jestliže $\mathbf{Ax} = \lambda\mathbf{x}$, pak $\mathbf{A}^k\mathbf{x} = \lambda\mathbf{A}^{k-1}\mathbf{x} = \cdots = \lambda^k\mathbf{x}$ pro k přirozené. Tedy k-tá mocnina matice má za vlastní čísla k-té mocniny vlastních čísel původní matice, vlastní vektory se nemění.

Polynomy. Je-li $p(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_k t^k$ libovolný polynom stupně k, pak definujeme $p(\mathbf{A}) = c_0\mathbf{I} + c_1\mathbf{A} + c_2\mathbf{A}^2 + \cdots + c_k\mathbf{A}^k$. Jestliže $\mathbf{Ax} = \lambda\mathbf{x}$, pak $p(\mathbf{A})\mathbf{x} = p(\lambda)\mathbf{x}$. Matice $p(\mathbf{A})$, kde $p(t)$ je polynom, má vlastní čísla $p(\lambda)$, vlastní vektory matic $\mathbf{A}$ a $p(\mathbf{A})$ jsou stejné.

Podobnost. Řekneme, že matice $\mathbf{A}$ a $\mathbf{B}$ jsou *podobné*, existuje-li regulární matice $\mathbf{T}$ s vlastností

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{AT}\,.$$

Matice $\mathbf{T}$ se nazývá *matice podobnosti* nebo také *podobnostní matice*.

Jestliže λ je vlastní číslo matice $\mathbf{B}$ a $\mathbf{y}$ je odpovídající vlastní vektor, pak

$$\mathbf{By} = \lambda\mathbf{y} \implies \mathbf{T}^{-1}\mathbf{ATy} = \lambda\mathbf{y} \implies \mathbf{ATy} = \lambda\mathbf{Ty} \implies \mathbf{Ax} = \lambda\mathbf{x} \quad \text{pro} \quad \mathbf{x} = \mathbf{Ty}\,.$$

Vlastní čísla podobných matic jsou tedy stejná, vlastní vektory příslušné témuž vlastnímu číslu jsou sice různé, jeden z druhého však lze snadno získat prostřednictvím matice podobnosti.

7.1.6. Diagonální, trojúhelníkové a blokově trojúhelníkové matice.

Diagonální matice $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_n)$ má vlastní čísla $\lambda_i = d_i$ a odpovídající vlastní vektory $\mathbf{y}_i = \mathbf{e}_i$, kde $\mathbf{e}_i$ je i-tý sloupec jednotkové matice.

Diagonizovatelnou matici $\mathbf{A}$ lze pomocí podobnostní transformace $\mathbf{T}^{-1}\mathbf{AT} = \mathbf{D}$ převést na diagonální matici $\mathbf{D}$. Protože vlastní čísla mohou být komplexní, je matice $\mathbf{D}$ i matice podobnosti $\mathbf{T}$ obecně komplexní. Pro symetrickou matici $\mathbf{A}$ existuje reálná ortonormální matice podobnosti $\mathbf{Q}$ taková, že matice $\mathbf{Q}^T\mathbf{AQ} = \mathbf{D}$ je diagonální.

Trojúhelníková matice má nuly buďto pod hlavní diagonálou, pak hovoříme o *horní trojúhelníkové* matici, nebo má nuly nad hlavní diagonálou, v tom případě jde o *dolní trojúhelníkovou* matici. V dalším budeme *trojúhelníkovou maticí* rozumět horní trojúhelníkovou matici. Nechť tedy $\mathbf{T} = \{t_{ij}\}_{i,j=1}^n$ je trojúhelníková matice, tj. $t_{ij} = 0$ pro $i > j$. Z charakteristické rovnice $\det(\mathbf{T} - \lambda\mathbf{I}) = 0$ okamžitě plyne, že vlastní čísla matice $\mathbf{T}$ jsou její diagonální prvky, tj. $\lambda_i = t_{ii}$. Vlastní vektory lze dopočítat také snadno. Matice $\mathbf{T} - \lambda\mathbf{I}$ má na hlavní diagonále nuly v pozicích, v nichž $t_{ii} = \lambda$. Je-li

$$\mathbf{T} - \lambda\mathbf{I} = \begin{pmatrix} \mathbf{U}_{11} & \mathbf{u} & \mathbf{U}_{13} \\ \mathbf{o}^T & 0 & \mathbf{v}^T \\ \mathbf{O} & \mathbf{o} & \mathbf{U}_{33} \end{pmatrix}$$

a $\mathbf{U}_{11}$ je regulární, pak odpovídající vlastní vektor lze zvolit ve tvaru

$$\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ -1 \\ \mathbf{o} \end{pmatrix},$$

kde $\mathbf{y}$ je řešení rovnice $\mathbf{U}_{11}\mathbf{y} = \mathbf{u}$.

Význam trojúhelníkových matic v úloze vlastních čísel plyne ze *Schurovy věty*:

Ke každé čtvercové matici $\mathbf{A}$ *existuje unitární matice* $\mathbf{Q}$ *taková, že*

$$\mathbf{Q}^H\mathbf{A}\mathbf{Q} = \mathbf{T} = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & t_{nn} \end{pmatrix}$$

je horní trojúhelníková matice. Rozklad $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^H$ *se nazývá Schurův rozklad matice* $\mathbf{A}$*,* $\mathbf{T}$ *se nazývá Schurův tvar matice* $\mathbf{A}$ *a sloupce* $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ *matice* $\mathbf{Q}$ *se nazývají Schurovy vektory.*

Důkaz Schurovy věty lze najít např. v [30].

Pro úplnost uveďme ještě definici unitární matice. Čtvercová *matice* $\mathbf{Q}$ *je unitární*, jestliže $\mathbf{Q}^H\mathbf{Q} = \mathbf{Q}\mathbf{Q}^H = \mathbf{I}$. Přitom $\mathbf{Q}^H$ je *matice Hermitovsky sdružená*, tj. transponovaná a komplexně sdružená: když $\mathbf{Q} = \{q_{ij}\}_{i,j=1}^n$ a $\mathbf{Q}^H = \{q_{ij}^H\}_{i,j=1}^n$, pak $q_{ij}^H = \bar{q}_{ji}$ je číslo komplexně sdružené s číslem q_{ji}.

Protože $\mathbf{Q}^H = \mathbf{Q}^{-1}$, je každá čtvercová matice $\mathbf{A}$ podobná s trojúhelníkovou maticí $\mathbf{T} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q}$, přičemž matice podobnosti $\mathbf{Q}$ je unitární. Zdůrazněme, že matice $\mathbf{T}$ a $\mathbf{Q}$ jsou obecně komplexní.

Schurův tvar reálné matice má komplexní složky v případě, že matice má nějaká komplexní vlastní čísla. Naštěstí však existuje *reálná varianta Schurovy věty*:

Ke každé reálné čtvercové matici $\mathbf{A}$ *existuje ortonormální matice* $\mathbf{Q}$ *taková, že*

$$\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{T} = \begin{pmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} & \dots & \mathbf{T}_{1p} \\ \mathbf{O}_{21} & \mathbf{T}_{22} & \dots & \mathbf{T}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O}_{p1} & \mathbf{O}_{p2} & \dots & \mathbf{T}_{pp} \end{pmatrix} \tag{7.5}$$

je blokově trojúhelníková reálná matice, v níž diagonální submatice $\mathbf{T}_{ii}$ *jsou čtvercové řádu jedna nebo dva a poddiagonální submatice* $\mathbf{O}_{ij}$*,* $i > j$*, jsou nulové. Vlastní čísla diagonálních submatic řádu dva jsou komplexně sdružená vlastní čísla matice* $\mathbf{A}$*, zbývající diagonální submatice řádu jedna jsou reálná vlastní čísla matice* $\mathbf{A}$*. Rozklad* $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^T$ *se nazývá reálný Schurův rozklad matice* $\mathbf{A}$*, matice* $\mathbf{T}$ *se nazývá reálný Schurův tvar matice* $\mathbf{A}$*, sloupce* $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ *matice* $\mathbf{Q}$ *jsou reálné Schurovy vektory.*

Pro výpočet komplexního i reálného Schurova tvaru matic existuje řada kvalitních programů, viz např. [3]. V MATLABu lze použít funkci `schur`.

Blokově trojúhelníková matice. Reálný tvar Schurovy matice je speciálním případem horní blokově trojúhelníkové matice

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \dots & \mathbf{A}_{1p} \\ \mathbf{O}_{21} & \mathbf{A}_{22} & \dots & \mathbf{A}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O}_{p1} & \mathbf{O}_{p2} & \dots & \mathbf{A}_{pp} \end{pmatrix} \tag{7.6}$$

se čtvercovými diagonálními maticemi $\mathbf{A}_{ii}$ a nulovými poddiagonálními maticemi $\mathbf{O}_{ij}$. Protože determinant matice $\mathbf{A}$ je součinem determinantů diagonálních submatic (důsledek Laplaceovy věty), všechna vlastní čísla $\mathbf{A}$ dostaneme z vlastních čísel diagonálních submatic. Také vlastní vektory $\mathbf{A}$ lze určit pomocí vlastních vektorů diagonálních submatic. Problém vlastních čísel pro blokově trojúhelníkové matice lze proto rozložit na menší subproblémy, které se řeší snadněji, a mnohé algoritmy výpočtu vlastních čísel toho využívají.

7.2. Metody výpočtu

7.2.1. Mocninná metoda

Základní varianta mocninné metody, viz algoritmy 7.1 a 7.2, umožňuje výpočet vlastního vektoru příslušného k vlastnímu číslu o největší absolutní hodnotě. Algoritmus 7.3 umí určit vlastní vektor příslušný k vlastnímu číslu o nejmenší absolutní hodnotě. Pro zvolené číslo μ algoritmus 7.3 najde vlastní vektor příslušný k vlastnímu číslu, které je k μ nejblíž. Máme-li dobrou aproximaci vlastního vektoru, algoritmus 7.4 tento vlastní vektor najde.

Algoritmus 7.1 : mocninná metoda

1. $\mathbf{x}_0$ je libovolný nenulový vektor
2. **for** $k := 1, 2, \ldots$ **do**
3. $\quad \mathbf{x}_k := \mathbf{A}\mathbf{x}_{k-1}$
4. **end**

Předpokládejme, že $\mathbf{A}$ je diagonizovatelná matice s jediným dominantním vlastním číslem λ_1, tj.

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|\,, \tag{7.7}$$

a nechť $\mathbf{v}_1$ je odpovídající vlastní vektor. Pak $\mathbf{x}_k$ konverguje k násobku $\mathbf{v}_1$. Naznačme si důkaz tohoto tvrzení. Startovací vektor $\mathbf{x}_0$ vyjádříme ve tvaru $\mathbf{x}_0 = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j$, kde $\mathbf{v}_j$ jsou vlastní vektory příslušné vlastním číslům λ_j. Pak

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} = \mathbf{A}^2\mathbf{x}_{k-2} = \cdots = \mathbf{A}^k\mathbf{x}_0 = \\ \mathbf{A}^k \sum_{j=1}^{n} \alpha_j \mathbf{v}_j = \sum_{j=1}^{n} \alpha_j \mathbf{A}^k \mathbf{v}_j = \sum_{j=1}^{n} \alpha_j \lambda_j^k \mathbf{v}_j = \lambda_1^k \left(\alpha_1 \mathbf{v}_1 + \sum_{j=2}^{n} (\lambda_j/\lambda_1)^k \alpha_j \mathbf{v}_j \right).$$

Pro $j > 1$, $|\lambda_j/\lambda_1| < 1$, proto $(\lambda_j/\lambda_1)^k \to 0$ a tedy nevymizí pouze člen obsahující $\mathbf{v}_1$. Vlastní číslo λ_1 můžeme dopočítat pomocí *Rayleighova podílu*.

Rayleighův podíl pro matici $\mathbf{A}$ a vektor $\mathbf{x} \neq \mathbf{o}$ je číslo

$$R(\mathbf{A}, \mathbf{x}) = \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}}.$$

Je-li $\mathbf{x}$ vlastní vektor matice $\mathbf{A}$ a λ je odpovídající vlastní číslo, pak

$$R(\mathbf{A}, \mathbf{x}) = \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \frac{\mathbf{x}^T \lambda \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \lambda.$$

Jestliže tedy $\mathbf{x}_k \to \mathbf{v}_1$, pak $R(\mathbf{A}, \mathbf{x}_k) \to \lambda_1$.

Následuje výčet některých nedostatků mocninné metody:

- Startovací vektor $\mathbf{x}_0$ nemusí obsahovat žádnou složku ve směru dominantního vlastního vektoru $\mathbf{v}_1$ (tj. $\alpha_1 = 0$). Tento případ je velmi nepravděpodobný, pokud $\mathbf{x}_0$ volíme náhodně, a prakticky nepředstavuje žádný problém, neboť zaokrouhlovací chyby obvykle takovou složku vytvoří.

- Jestliže existuje několik vlastních čísel o stejné absolutní hodnotě, pak mocninná metoda vůbec nemusí konvergovat.

- Geometrický růst složek může vést pro $|\lambda_1| > 1$ k přetečení a pro $|\lambda_1| < 1$ k podtečení. Proto je účelné v každém kroku vektor $\mathbf{x}_k$ normovat tak, aby jeho norma byla rovna jedné.

Algoritmus 7.2 : normalizovaná mocninná metoda

1. $\mathbf{x}_0$ je libovolný nenulový vektor, $\|\mathbf{x}_0\|_2 = 1$
2. **for** $k := 1, 2, \ldots$ **do**
3. $\quad \mathbf{y}_k := \mathbf{A}\mathbf{x}_{k-1}$
4. $\quad \mathbf{x}_k := \mathbf{y}_k / \|\mathbf{y}_k\|_2$
5. $\quad \sigma_k := \mathbf{x}_k^T \mathbf{A}\mathbf{x}_k$
6. **end**

Pak $\mathbf{x}_k \to \mathbf{v}_1$, $\|\mathbf{v}_1\|_2 = 1$ a $\sigma_k = R(\mathbf{A}, \mathbf{x}_k) \to \lambda_1$. Příkaz na řádku 5 lze nahradit příkazem $\sigma_k := \mathbf{x}_k^T \mathbf{y}_k$. Další možností je vypuštění řádku 5 a dopočítání vlastního čísla až po dokončení cyklu. Výpočet vlastního čísla uvnitř cyklu má smysl pro zařazení vhodného stop-kritéria pro ukončení iterací. Kromě podmínky typu $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2 < \varepsilon$ lze použít také $|\sigma_k - \sigma_{k-1}| < \varepsilon$. Spolehlivější kritérium na ukončení iterací založené na výpočtu levých a pravých vlastních vektorů lze najít třeba v [42].

Pro chybu platí, viz [56],

$$\|\mathbf{x}_k - (\pm\mathbf{v}_1)\|_2 = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right), \qquad |\sigma_k - \lambda_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right), \tag{7.8}$$

přičemž $\pm$ znamená, že uvedený vztah platí v každém kroku jen pro jedno ze znamének plus a mínus.

7.2.2. Inverzní iterace.

Protože vlastní čísla matice $\mathbf{A}^{-1}$ jsou převrácenými hodnotami vlastních čísel matice $\mathbf{A}$, mocninná metoda aplikovaná na $\mathbf{A}^{-1}$ konverguje k vlastnímu vektoru příslušnému nejmenšímu vlastnímu číslu $\mathbf{A}$. Místo abychom v každém kroku počítali $\mathbf{y}_k = \mathbf{A}^{-1}\mathbf{x}_{k-1}$, řešíme soustavu rovnic $\mathbf{L}\mathbf{U}\mathbf{y}_k = \mathbf{x}_{k-1}$, přičemž LU rozklad $\mathbf{A} = \mathbf{L}\mathbf{U}$ provedeme jen jednou, před začátkem iterací. Tak dostaneme

Algoritmus 7.3 : inverzní iterace

1. $\mathbf{x}_0$ je libovolný nenulový vektor, $\|\mathbf{x}_0\|_2 = 1$
2. **for** $k := 1, 2, \ldots$ **do**
3. vypočti $\mathbf{y}_k$ řešením rovnice $\mathbf{A}\mathbf{y}_k = \mathbf{x}_{k-1}$
4. $\mathbf{x}_k := \mathbf{y}_k / \|\mathbf{y}_k\|_2$
5. $\sigma_k := \mathbf{x}_k^T \mathbf{A}\mathbf{x}_k$
6. **end**

Pak $\mathbf{x}_k \to \mathbf{v}_1$, $\|\mathbf{v}_1\|_2 = 1$, a $\sigma_k \to \lambda_1$, kde $|\lambda_1| < \min_{2 \le j \le n} |\lambda_j|$.

Inverzní iterace s posunem. Nechť μ je dané číslo, λ_i je vlastní číslo nejbližší k μ a λ_j je vlastní číslo, které je druhé nejbližší k μ, přičemž

$$|\mu - \lambda_i| < |\mu - \lambda_j| \le \min_{\ell \ne i,j} |\mu - \lambda_\ell|.$$

Jestliže třetí řádek algoritmu 7.3 nahradíme řádkem

3. vypočti $\mathbf{y}_k$ řešením rovnice $(\mathbf{A} - \mu\mathbf{I})\mathbf{y}_k = \mathbf{x}_{k-1}$

pak $\mathbf{x}_k \to \mathbf{v}_i$, $\|\mathbf{v}_i\|_2 = 1$ a $\sigma_k \to \lambda_i$. Pro chybu platí, viz [56],

$$\|\mathbf{x}_k - (\pm\mathbf{v}_i)\|_2 = O\left(\left|\frac{\mu - \lambda_i}{\mu - \lambda_j}\right|^k\right), \qquad |\sigma_k - \lambda_i)| = O\left(\left|\frac{\mu - \lambda_i}{\mu - \lambda_j}\right|^{2k}\right), \tag{7.9}$$

význam $\pm$ je stejný jako v (7.8). Inverzní iterace s posunem je užitečná zejména tehdy, když nějakou metodou získáme přibližné hodnoty vlastních čísel a chceme k nim dopočítat odpovídající vlastní vektory.

Další modifikací mocninné metody je následující

Algoritmus 7.4 : metoda Rayleighových podílů

1. $\mathbf{x}_0$ je libovolný nenulový vektor, $\|\mathbf{x}_0\|_2 = 1$
2. **for** $k := 1, 2, \ldots$ **do**
3. $\sigma_k = \mathbf{x}_{k-1}^T \mathbf{A}\mathbf{x}_{k-1}$
4. vypočti $\mathbf{y}_k$ řešením rovnice $(\mathbf{A} - \sigma_k\mathbf{I})\mathbf{y}_k = \mathbf{x}_{k-1}$
5. $\mathbf{x}_k := \mathbf{y}_k / \|\mathbf{y}_k\|_2$
6. **end**

Metoda konverguje pro skoro každý startovací vektor $\mathbf{x}_0$. Je-li $\mathbf{x}_0$ dostatečně dobrá aproximace vlastního vektoru $\mathbf{v}_i$, pak $(\sigma_k, \mathbf{x}_k) \to (\lambda_i, \mathbf{v}_i)$, kde λ_i je vlastní číslo odpovídající vlastnímu vektoru $\mathbf{v}_i$. Nastane-li konvergence, je kubická, tj.

$$\|\mathbf{x}_{k+1} - (\pm)\mathbf{v}_i\| = O\left(\|\mathbf{x}_k - (\pm)\mathbf{v}_i\|^3\right), \quad |\sigma_{k+1} - \lambda_i| = O\left(|\sigma_k - \lambda_i|^3\right),$$

viz [56]. Efektivnost jinak velmi rychlého algoritmu částečně snižuje to, že v každé iteraci musíme znovu faktorizovat matici $\mathbf{A} - \sigma_k\mathbf{I}$. Má-li však matice $\mathbf{A}$ speciální tvar umožňující snadnou faktorizaci, například když je třídiagonální, je metoda Rayleighova podílu vynikající.

7.2.3. Redukce.

Předpokládejme, že jsme mocninnou metodou vypočetli dominantní vlastní číslo λ_1, $|\lambda_1| > \max_{2\le i\le n} |\lambda_i|$, a odpovídající vlastní vektor $\mathbf{x}_1$. Ukažme si, jak určit další vlastní číslo λ_2 *metodou redukce* (v anglicky psané literatuře se používá termín *deflation*).

Pro jednoduchost se omezíme na případ, když matice $\mathbf{A}$ je symetrická. Pak matice $\mathbf{A}_1 = \mathbf{A} - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^T / \|\mathbf{x}_1\|_2^2$ má vlastní čísla $0, \lambda_2, \ldots, \lambda_n$, viz [43]. Jestliže je λ_2 dominantní vlastní číslo matice $\mathbf{A}_1$, tj. když $|\lambda_2| > \max_{3\le i\le n} |\lambda_i|$, mocninnou metodou aplikovanou na matici $\mathbf{A}_1$ určíme aproximaci σ_2 vlastního čísla λ_2. Pomocí inverzní iterace aplikované na původní matici $\mathbf{A}$ s posunem $\mu = \sigma_2$ pak vypočteme λ_2 a odpovídající vlastní vektor $\mathbf{x}_2$. Tento postup můžeme opakovat a vypočítat několik dalších dvojic $(\lambda_\ell, \mathbf{x}_\ell)$. Pro určení většího počtu vlastních čísel a odpovídajících vlastních vektorů je však metoda postupných redukcí příliš těžkopádná a proto dáme přednost některé z efektivnějších metod, s nimiž se seznámíme v následujících kapitolách.

7.2.4. Simultánní iterace

je určena pro výpočet několika párů vlastních čísel a vlastních vektorů současně. Nejjednodušší postup spočívá v použití mocninné metody pro několik startovacích vektorů.

Algoritmus 7.5 : simultánní iterace

1. $\mathbf{X}_0$ je libovolná matice typu (n, p) hodnosti p
2. **for** $k := 1, 2, \ldots$ **do**
3. $\quad \mathbf{X}_k := \mathbf{A}\mathbf{X}_{k-1}$
4. **end**

Nechť pro vlastní čísla $\mathbf{A}$ platí $|\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_p| > |\lambda_{p+1}| \ge \cdots \ge |\lambda_n|$ a nechť $\mathbf{v}_i$ je vlastní vektor příslušný k λ_i. Označme $\mathcal{S} = \mathrm{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p)$ a dále $\mathcal{S}_k = \mathrm{span}(\mathbf{X}_k) = \mathrm{span}(\mathbf{A}^k \mathbf{X}_0)$. Jestliže žádný ze sloupců matice $\mathbf{X}_0$ není kolmý k $\mathcal{S}$, pak $\mathcal{S}_k \to \mathcal{S}$, tj. vektorový prostor generovaný sloupci matic $\mathbf{X}_k$ konverguje k vektorovému prostoru generovanému p dominantními vlastními vektory.

Stejně jako v mocninné metodě hrozí podtečení nebo přetečení. Navíc dochází ke zhoršování podmíněnosti matic $\mathbf{X}_k$, takže sloupce $\mathbf{X}_k$ tvoří špatně podmíněnou bázi prostoru $\mathcal{S}_k$. Oba tyto nedostatky můžeme překonat, když v každém kroku budeme sloupce $\mathbf{X}_k$ ortonormalizovat pomocí QR rozkladu.

Algoritmus 7.6 : ortogonální iterace

1. $\mathbf{X}_0$ je libovolná matice typu (n, p) hodnosti p
2. **for** $k := 1, 2, \ldots$ **do**
3. $\quad \mathbf{Y}_k := \mathbf{A}\mathbf{X}_{k-1}$
4. $\quad$ proveď redukovaný QR rozklad $\mathbf{Q}_k \mathbf{R}_k = \mathbf{Y}_k$
5. $\quad \mathbf{X}_k = \mathbf{Q}_k$
6. **end**

Redukovaným QR rozkladem rozumíme vyjádření $\mathbf{Y}_k = \mathbf{Q}_k \mathbf{R}_k$, kde $\mathbf{Q}_k$ je matice typu (n, p) s vlastností $\mathbf{Q}_k^T \mathbf{Q}_k = \mathbf{I}$ a $\mathbf{R}_k$ je horní trojúhelníková matice řádu p. Redukovaný

QR rozklad určíme některou z metod uvedených v kapitole 3.4.

QR rozklad používáme k sestrojení ortonormální báze v prostoru tvořeném sloupci matice $\mathbf{Y}_k$. Za $\mathbf{X}_k$ zde bereme $\mathbf{Q}_k$ místo $\mathbf{Y}_k$. Protože prostor generovaný sloupci matic $\mathbf{Q}_k$ a $\mathbf{Y}_k$ je stejný, vytváříme stejné podprostory $\mathcal{S}_k$ jako v algoritmu 7.5, tentokrát však s ortonormální bází.

Věta (*o ortogonální iteraci*) *Matice* $\mathbf{Q}_k$ *konvergují k matici* $\mathbf{Q}$*, jejíž sloupce tvoří ortonormální bázi prostoru* $\mathcal{S}$*. Matice* $\mathbf{T}_k = \mathbf{Q}_k^T\mathbf{A}\mathbf{Q}_k$ *konvergují k horní blokově trojúhelníkové matici* $\mathbf{T} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$*. Jestliže* $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_p| > \lambda_{p+1} \geq ... \geq \lambda_n$*, pak* $\mathbf{T} = \{t_{ij}\}_{i,j=1}^p$ *je horní trojúhelníková matice a na její diagonále jsou vlastní čísla* $\lambda_i = t_{ii}$*,* $i = 1, 2, \ldots, p$*.*

Důkaz lze najít např. v [11] a [30]. Je-li $\mathbf{A}$ navíc symetrická, pak $\mathbf{T} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ a $\mathbf{Q} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p)$ je matice odpovídajících vlastních vektorů.

Ortogonalizace prováděná v každém kroku je nákladná, navíc konvergence může být velmi pomalá. V další kapitole ukážeme, jak lze tyto nepříznivé okolnosti zmírnit.

7.2.5. QR metoda

Efektivní implementace QR metody patří mezi vůbec nejpoužívanější programy pro výpočet všech vlastních čísel a vlastních vektorů plných (tj. neřídkých) matic. Proto budeme QR metodě věnovat náležitou pozornost.

Algoritmus 7.7 : QR metoda

1. $\mathbf{A}_0 := \mathbf{A}$
2. **for** $k := 1, 2, \ldots$ **do**
3. proveď QR rozklad $\mathbf{Q}_k\mathbf{R}_k = \mathbf{A}_{k-1}$
4. $\mathbf{A}_k := \mathbf{R}_k\mathbf{Q}_k$
5. **end**

Ukažme si, že jde o postup, který je ekvivalentní s ortogonální iterací pro $p = n$ a $\mathbf{X}_0 = \mathbf{A}$. Pomocí řádků 3 a 4 algoritmu QR metody odvodíme, že

$$\mathbf{A}_k = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^T\mathbf{A}_{k-1}\mathbf{Q}_k = \mathbf{Q}_k^T\mathbf{Q}_{k-1}^T\mathbf{A}_{k-2}\mathbf{Q}_{k-1}\mathbf{Q}_k = \cdots = \hat{\mathbf{Q}}_k^T\mathbf{A}\hat{\mathbf{Q}}_k\,,$$

kde $\hat{\mathbf{Q}}_k = \mathbf{Q}_1\mathbf{Q}_2\ldots\mathbf{Q}_k$ je ortonormální matice. Pak platí následující

Věta(*o QR algoritmu*) *Jestliže* $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$*, pak* $\mathbf{A}_k$ *konvergují k horní trojúhelníkové matici* $\mathbf{T} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$*, kde* $\mathbf{Q} = \lim_{k\to\infty}\hat{\mathbf{Q}}_k$*.*

Známe-li vlastní čísla, můžeme k nim dopočítat aproximace odpovídajících vlastních vektorů, např. některou z metod uvedených v [30]. Použít lze třeba inverzní iteraci s posunem, v níž jako posuny bereme vypočtené aproximace vlastních čísel.

Je-li $\mathbf{A}$ navíc symetrická, pak $\mathbf{T} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ a $\mathbf{Q} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ je matice odpovídajících vlastních vektorů.

Přeformulování ortogonální iterace na QR metodu dává elegantní algoritmus, který však stále neřeší jeho dříve zmíněné nedostatky: vysoké výpočetní náklady v jedné iteraci a pomalou konvergenci. QR metoda zapsaná ve formě algoritmu 7.7 může být dokonce

zcela nepoužitelná: pro ortonormální matici $\mathbf{A}$ (jejíž všechna vlastní čísla leží na jednotkové kružnici v komplexní rovině) dostáváme $\mathbf{A}_k = \mathbf{A}$.

Redukce nákladů na jednu iteraci. QR rozklad matice $\mathbf{A}_{k-1}$ je výpočetně náročný, pro plnou matici vyžaduje $O(n^3)$ operací. Je-li však $\mathbf{A}_{k-1}$ horní Hessenbergova matice, stačí $O(n^2)$ operací, a je-li $\mathbf{A}_{k-1}$ třídiagonální, pak jen $O(n)$ operací (k anulování nenulových poddiagonálních členů v $\mathbf{A}_{k-1}$ je výhodné použít Givensovy rovinné rotace, viz kapitola 3.4.2). Jak uvidíme, každou matici lze pomocí $O(n^3)$ operací transformovat na podobnou horní Hessenbergovu matici. Protože symetrická Hessenbergova matice je třídiagonální, symetrickou matici lze pomocí $O(n^3)$ operací transformovat dokonce na podobnou matici třídiagonální.

Snadno se ověří, že QR metoda zachovává typ matice: je-li matice $\mathbf{A}_{k-1}$ horní Hessenbergova resp. třídiagonální, je taková také matice $\mathbf{A}_k = \mathbf{R}_k \mathbf{A}_{k-1} \mathbf{R}_k^{-1}$.

Proto je zcela běžné ještě před startem QR metody matici $\mathbf{A}$ transformovat na horní Hessenbergův resp. třídiagonální tvar pomocí $O(n^3)$ operací a pak v každé iteraci provádět již jen $O(n^2)$ resp. $O(n)$ operací.

Transformace na horní Hessenbergův tvar. Algoritmus lze popsat takto:

položíme $\mathbf{A}_0 = \mathbf{A}$ a počítáme

$$\mathbf{A}_k = \mathbf{P}_k \mathbf{A}_{k-1} \mathbf{P}_k \,, \qquad k = 1, 2, \ldots, n-2,$$

kde $\mathbf{P}_k$ je ortonormální Householderova matice,

$$\mathbf{P}_k = \mathbf{I} - 2\mathbf{w}_k \mathbf{w}_k^T,$$

v níž $\mathbf{w}_k$ je sloupcový vektor jednotkové euklidovské délky,

$$\mathbf{w}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|_2} \,,$$

a složky $z_i^{(k)}, i = 1, 2, \ldots, n$, sloupcového vektoru $\mathbf{z}_k$ jsou

$$z_i^{(k)} = \begin{cases} 0 & i \le k \\ a_{k+1,k}^{(k-1)} + \beta_k \quad \text{pro} & i = k+1, \\ a_{ik}^{(k-1)} & i > k+1 \end{cases} \qquad \beta_k = \operatorname{sign}(a_{k+1,k}^{(k-1)}) \left(\sum_{i=k+1}^{n} \left[a_{ik}^{(k-1)} \right]^2 \right)^{1/2}.$$

Pak $\mathbf{A}_{n-2}$ je horní Hessenbergova matice podobná s maticí $\mathbf{A}$ a platí

$$\mathbf{A}_{n-2} = \mathbf{Q}^T \mathbf{A} \mathbf{Q} \,, \qquad \text{kde} \qquad \mathbf{Q} = \mathbf{P}_1 \mathbf{P}_2 \ldots \mathbf{P}_{n-2}$$

je ortonormální podobnostní matice.

Následuje několik komentářů k uvedenému algoritmu.

1) Householderova matice $\mathbf{P}_k$ je tvaru

$$\mathbf{P}_k = \begin{pmatrix} \mathbf{I}_k & \mathbf{O} \\ \mathbf{O}^T & \bar{\mathbf{P}}_k \end{pmatrix},$$

kde $\mathbf{I}_k$ je jednotková matice řádu k, $\mathbf{O}$ je nulová matice typu $(k, n-k)$ a $\bar{\mathbf{P}}_k$ je redukovaná Householderova matice,

$$\bar{\mathbf{P}}_k = \mathbf{I}_{n-k} - 2\bar{\mathbf{w}}_k\bar{\mathbf{w}}_k^T,$$

kde $\mathbf{I}_{n-k}$ je jednotková matice řádu $n-k$ a $\bar{\mathbf{w}}_k$ je sloupcový vektor tvořený posledními $n-k$ prvky vektoru $\mathbf{w}_k$, tj. $\bar{w}_i^{(k)} = w_{k+i}^{(k)}$, $i = 1, 2, \ldots, n-k$.

2) Jestliže

$$\mathbf{A}_{k-1} = \left(\begin{array}{c|c|c} \mathbf{B} & & \mathbf{C} \\ \hline \mathbf{O} & \mathbf{b} & \mathbf{D} \end{array}\right) \begin{array}{c} k \\ n-k \end{array} \qquad \begin{array}{ccc} k-1 & 1 & n-k \end{array} \tag{7.10}$$

kde $\mathbf{O}$ je nulová matice, pak

$$\mathbf{A}_k = \mathbf{P}_k\mathbf{A}_{k-1}\mathbf{P}_k = \left(\begin{array}{c|c|c} \mathbf{B} & & \mathbf{C}\bar{\mathbf{P}}_k \\ \hline \mathbf{O} & \bar{\mathbf{P}}_k\mathbf{b} & \bar{\mathbf{P}}_k\mathbf{D}\bar{\mathbf{P}}_k \end{array}\right) \begin{array}{c} k \\ n-k \end{array} \qquad \begin{array}{ccc} k-1 & 1 & n-k \end{array} \tag{7.11}$$

3) Vektor $\bar{\mathbf{w}}_k$ je určen tak, aby $\bar{\mathbf{P}}_k\mathbf{b} = (-\beta_k, 0, \ldots, 0)^T$. Vzorce pro složky vektoru $\bar{\mathbf{w}}_k$, vyplývající ze vzorců pro $z_i^{(k)}$, $i = k+1, k+2, \ldots, n$, odvodíme stejně jako v kapitole 3.4.1. Stačí si uvědomit, že $\mathbf{b} = (a_{k+1,k}^{(k-1)}, a_{k+2,k}^{(k-1)}, \ldots, a_{nk}^{(k-1)})^T$.

4) Výpočet $\mathbf{A}_k = \mathbf{P}_k\mathbf{A}_{k-1}\mathbf{P}_k$ provádíme podle (7.10) a (7.11). Matici $\bar{\mathbf{P}}_k$ nesestavujeme, při výpočtu $\mathbf{C}\bar{\mathbf{P}}_k$ a $\bar{\mathbf{P}}_k\mathbf{D}\bar{\mathbf{P}}_k$ pracujeme přímo s vyjádřením $\bar{\mathbf{P}}_k = \mathbf{I}_{n-k} - 2\bar{\mathbf{w}}_k\bar{\mathbf{w}}_k^T$. Celkový počet operací potřebný k výpočtu $\mathbf{A}_{n-2}$ je řádu $O(n^3)$.

5) Je-li $\mathbf{A}$ symetrická, jsou také $\mathbf{A}_{k-1}$ i $\mathbf{A}_k$ symetrické, takže v (7.10) a (7.11) je

$$\mathbf{C} = \begin{pmatrix} \mathbf{O} & \mathbf{b} \end{pmatrix}^T, \qquad \mathbf{C}\bar{\mathbf{P}}_k = \begin{pmatrix} \mathbf{O} & \bar{\mathbf{P}}_k\mathbf{b} \end{pmatrix}^T. \tag{7.12}$$

Vzhledem k symetrii $\mathbf{D}$ dále platí

$$\bar{\mathbf{P}}_k\mathbf{D}\bar{\mathbf{P}}_k = \mathbf{D} - \bar{\mathbf{w}}_k\mathbf{q}_k^T - \mathbf{q}_k\bar{\mathbf{w}}_k^T,$$

kde

$$\mathbf{q}_k = \mathbf{p}_k - (\mathbf{p}_k^T\bar{\mathbf{w}}_k)\bar{\mathbf{w}}_k \quad \text{a} \quad \mathbf{p}_k = 2\mathbf{D}\bar{\mathbf{w}}_k\,.$$

Celkový počet operací potřebný k výpočtu $\mathbf{A}_{n-2}$ je opět řádu $O(n^3)$, je však méně než poloviční oproti případu, když $\mathbf{A}$ je nesymetrická, viz [30].

6) V MATLABu lze k transformaci na Hessenbergův tvar použít funkci `hess`.

Zrychlení konvergence lze dosáhnout podobně jako ve variantách mocninné metody pomocí vhodně volených posunů.

Algoritmus 7.8 : QR metoda s posuny

1. $\mathbf{A}_0 := \mathbf{A}$
2. **for** $k := 1, 2, \ldots$ **do**
3. vyber posun μ_k
3. proveď QR rozklad $\mathbf{Q}_k\mathbf{R}_k = \mathbf{A}_{k-1} - \mu_k\mathbf{I}$
4. $\mathbf{A}_k := \mathbf{R}_k\mathbf{Q}_k + \mu_k\mathbf{I}$
5. **end**

Protože

$$\mathbf{A}_k = \mathbf{R}_k\mathbf{Q}_k + \mu_k\mathbf{I} = \mathbf{Q}_k^T(\mathbf{A}_{k-1} - \mu_k\mathbf{I})\mathbf{Q}_k + \mu_k\mathbf{I} = \mathbf{Q}_k^T\mathbf{A}_{k-1}\mathbf{Q}_k = \cdots = \hat{\mathbf{Q}}_k^T\mathbf{A}\hat{\mathbf{Q}}_k\,,$$

jsou matice $\mathbf{A}_k$ opět podobné s $\mathbf{A}$. Pro vhodně zvolené posuny lze však konvergenci značně urychlit.

Nejjednodušší volba pro posun μ_k je prvek $a_{nn}^{(k-1)}$ v pravém dolním rohu matice $\mathbf{A}_{k-1}$, známý jako *Rayleighův posun*. QR iterace s Rayleighovým posunem $a_{nn}^{(k-1)}$ konverguje kubicky pro skoro všechny symetrické třídiagonální matice.

Robustnější alternativu představuje *Wilkinsonův posun*. Nechť

$$\mathbf{A}_{k-1} = \begin{pmatrix} \mathbf{A}_{11}^{(k-1)} & \mathbf{A}_{12}^{(k-1)} \\ \mathbf{A}_{21}^{(k-1)} & \mathbf{A}_{22}^{(k-1)} \end{pmatrix}$$

je bloková matice s bloky $\mathbf{A}_{11}^{(k-1)}$ řádu $n-2$ a $\mathbf{A}_{22}^{(k-1)}$ řádu 2. Pak Wilkinsonův posun μ_k je roven tomu vlastnímu číslu μ submatice $\mathbf{A}_{22}^{(k-1)}$, které je bližší k $a_{nn}^{(k-1)}$. Je-li $\mathbf{A}$ symetrická, je také $\mathbf{A}_{22}^{(k-1)}$ symetrická a její vlastní jsou čísla reálná. Pro symetrickou třídiagonální matici QR metoda s Wilkinsonovým posunem vždy konverguje, nejméně lineárně, pro skoro všechny matice však kubicky.

Problém nastane, když vlastní čísla jsou komplexní. Pak je Wilkinsonův posun komplexní a následující výpočty musejí probíhat v komplexní aritmetice.

V případě reálné matice $\mathbf{A}$ se počítání s komplexními čísly můžeme vyhnout. Předpokládejme, že vlastní čísla matice $\mathbf{A}_{22}^{(k-1)}$ jsou komplexně sdružená čísla μ a $\bar{\mu}$. Dá se ukázat, že dvě po sobě jdoucí iterace, jedna s posunem μ a následující s posunem $\bar{\mu}$, dávají reálný výsledek. Šikovná implementace založená na této skutečnosti je známa jako *Francisův posun*.

Bohužel existují matice, pro které ani Francisův posun nevede ke konvergenci. Praktické algoritmy proto používají ještě další doplňkové posuny, pomocí nichž se pokoušejí stagnující konvergenci znovu nastartovat. Až se to zdaří, přejde se opět na Francisův posun.

Závěr. Profesionální programy založené na QR metodě používají řadu dalších opatření ke zvýšení jak spolehlivosti tak rychlosti. Proto lze propracované implementace QR metody považovat prakticky za neiterační procesy, neboť pro většinu matic k dostatečně přesnému výpočtu jednoho vlastního čísla potřebují typicky jen dvě až tři iterace, takže celkový

počet iterací je malý celočíselný násobek řádu n matice. Profesionální programy pro výpočet vlastních čísel, včetně QR metody, jsou součástí balíku programů LAPACK [1].

7.2.6. Metoda iterací v podprostorech

Při řešení problému vlastních čísel pro rozsáhlé řídké matice často vystačíme se znalostí menšího počtu vlastních čísel a vektorů. Varianty mocninné metody kombinované s redukcí se příliš nehodí, pokud počet požadovaných vlastních čísel a vektorů není v jednotkách, nýbrž spíše v desítkách či stovkách. QR metoda se zase nehodí proto, že pro velké matice, řádů v tisících, je zbytečné a neúnosně nákladné počítat všechna vlastní čísla a vektory. Z metod, kterými jsme se doposud zabývali, připadá v úvahu algoritmus 7.6 ortogonální iterace. V této kapitole si uvedeme jeho modifikaci známou jako metoda iterací v podprostorech.

Algoritmus 7.9 : metoda iterací v podprostorech

1. $\mathbf{X}_0 = (\mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_p^{(0)})$ je matice p lineárně nezávislých startovacích vektorů
2. **for** $k := 1, 2, \ldots$ **do**
3. $\quad \mathbf{Y}_k := \mathbf{A}\mathbf{X}_{k-1}$
4. $\quad$ proveď redukovaný QR rozklad $\mathbf{Q}_k\mathbf{R}_k = \mathbf{Y}_k$
5. $\quad \mathbf{B}_k := \mathbf{Q}_k^T\mathbf{A}\mathbf{Q}_k$
6. $\quad$ urči vlastní čísla $\sigma_i^{(k)}$ a vlastní vektory $\mathbf{z}_i^{(k)}$ matice $\mathbf{B}_k$, tj.
   $$\mathbf{B}_k\mathbf{Z}_k = \mathbf{Z}_k\mathbf{D}_k,$$
   kde $\mathbf{Z}_k = (\mathbf{z}_1^{(k)}, \mathbf{z}_2^{(k)}, \ldots, \mathbf{z}_p^{(k)})$ a $\mathbf{D}_k = \mathrm{diag}(\sigma_1^{(k)}, \sigma_2^{(k)}, \ldots, \sigma_p^{(k)})$
7. $\quad \mathbf{X}_k := \mathbf{Q}_k\mathbf{Z}_k$
8. **end**

K algoritmu připojíme několik poznámek.

1) Pokud by $\mathbf{Z}_k$ byla jednotková matice, dostaneme algoritmus ortogonální iterace 7.6.

2) Vlastní čísla a vlastní vektory matice $\mathbf{B}_k$ určíme například QR metodou.

3) Vlastní čísla $\sigma_i^{(k)}$ matice $\mathbf{B}_k$ se nazývají *Ritzova čísla* a vektory $\mathbf{x}_i^{(k)} = \mathbf{Q}_k\mathbf{z}_i^{(k)}$ se nazývají *Ritzovy vektory*. Přitom $\mathrm{span}(\mathbf{X}_k) \subseteq \mathrm{span}(\mathbf{Q}_k) = \mathrm{span}(\mathbf{Y}_k) = \mathrm{span}(\mathbf{A}^k\mathbf{X}_0)$. Je-li $\mathbf{Z}_k$ regulární (pro symetrickou matici $\mathbf{A}$ je $\mathbf{Z}_k$ dokonce ortonormální), pak $\mathrm{span}(\mathbf{X}_k) = \mathrm{span}(\mathbf{Q}_k)$. Protože $\mathbf{Q}_k^T(\mathbf{A}\mathbf{X}_k - \mathbf{X}_k\mathbf{D}_k) = \mathbf{O}$, ortogonální projekce vektorů $\mathbf{A}\mathbf{x}_i^{(k)} - \sigma_i^{(k)}\mathbf{x}_i^{(k)}$ do prostoru $\mathcal{S}_k := \mathrm{span}(\mathbf{Q}_k)$ je rovna nule. Říkáme také, že $\sigma_i^{(k)}$ resp. $\mathbf{x}_i^{(k)}$ jsou Ritzova čísla resp. Ritzovy vektory v podprostoru $\mathcal{S}_k$. Je-li matice $\mathbf{A}$ regulární, pak vektory $\mathbf{x}_i^{(k)}$, $i = 1, 2, \ldots, p$, jsou lineárně nezávislé, tj. tvoří bázi prostoru $\mathcal{S}_k$. To nastane třeba tehdy, když $\mathbf{A}$ je pozitivně definitní.

4) Nechť pro vlastní čísla $\mathbf{A}$ platí $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$. Předpokládejme, že žádný ze sloupců matice $\mathbf{X}_0$ není kolmý k $\mathcal{S}$ a že matice $\mathbf{Z}_k$, $k = 1, 2, \ldots$ jsou regulární. Nechť $\mathbf{v}_i$ je vlastní vektor příslušný k λ_i, $i = 1, 2, \ldots, p$,

$\mathbf{X} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p)$ a $\mathcal{S} = \text{span}(\mathbf{X})$. Pak $\mathcal{S}_k \to \mathcal{S}$, $\mathbf{X}_k \to \mathbf{X}$, $\mathbf{Q}_k \to \mathbf{Q}$, $\mathbf{D}_k \to \mathbf{D}$. Protože $\mathbf{Q}_k^T(\mathbf{A}\mathbf{X}_k - \mathbf{X}_k\mathbf{D}_k) \to \mathbf{Q}^T(\mathbf{A}\mathbf{X} - \mathbf{X}\mathbf{D}) = \mathbf{O}$ a $\text{span}(\mathbf{Q}) = \mathcal{S}$, $\text{span}(\mathbf{A}\mathbf{X} - \mathbf{X}\mathbf{D})$ je k $\mathcal{S}$ kolmý. Protože současně $\text{span}(\mathbf{A}\mathbf{X} - \mathbf{X}\mathbf{D}) \subseteq \mathcal{S}$, $\mathbf{A}\mathbf{X} - \mathbf{X}\mathbf{D} = \mathbf{O}$ nebo-li $\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{D}$, tj. Ritzova čísla konvergují k vlastním číslům $\lambda_1, \lambda_2, \ldots, \lambda_p$ a Ritzovy vektory konvergují k příslušným vlastním vektorům.

5) Konvergence k největším vlastním číslům je nejrychlejší. Chceme-li určit q největších vlastních čísel a odpovídajících vektorů, zvolíme počet p iteračních vektorů větší než q, např. ve [5] se pro pozitivně definitní matici $\mathbf{A}$ doporučuje volit $p = \min(2q, q+8)$.

5) Pokud nás zajímají nejmenší vlastní čísla, tj. $|\lambda_1| \le |\lambda_2| \le \cdots \le |\lambda_p| < |\lambda_j|$, $j > p$, nahradíme řádek 3 řádkem

3. urči $\mathbf{Q}_k$ jako řešení soustavy rovnic $\mathbf{A}\mathbf{Q}_k = \mathbf{X}_{k-1}$

Je-li $\mathbf{A}$ symetrická, lze k ověření toho, že jsme našli Ritzova čísla $\sigma_i^{(k)}$, $i = 1, 2, \ldots, q$, aproximující q nejmenších vlastních čísel λ_i, využít následující

Sturmův test (důsledek Sylvesterovy věty o setrvačnosti). *Je-li* $\mathbf{A} - \mu\mathbf{I} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, *kde* $\mathbf{L}$ *je dolní trojúhelníková matice s jedničkami na hlavní diagonále, pak počet záporných diagonálních prvků matice* $\mathbf{D}$ *je roven počtu vlastních čísel menších než* μ.

Pro $\mu_k = (1 + \varepsilon_m)\sigma_q^{(k)}$ určíme počet n_k záporných prvků matice $\mathbf{D}$. Když $n_k < \mu_k$, některá z Ritzových čísel $\sigma_i^{(k)}$, $i = 1, 2, \ldots, q$, nejsou vlastní čísla menší než μ_k. Jednou z možností je pokračovat v iteracích tak dlouho dokud nebude $n_k = \mu_k$. Další možností je opakovaný výpočet s větším počtem iteračních vektorů, tj. s větším p.

Metoda iterací v podprostorech pro výpočet nejmenších vlastních čísel a odpovídajících vlastních vektorů symetrických matic je detailně zpracována v [5].

7.2.7. Arnoldiho metoda

Zvolíme startovací vektor $\mathbf{x}_0 \neq \mathbf{o}$ a v $k-$té iteraci se počítáme Ritzova čísla a Ritzovy vektory v Krylovových podprostorech $\mathcal{K}_k \equiv \mathcal{K}_k(\mathbf{A}, \mathbf{x}_0) = \text{span}(\mathbf{x}_0, \mathbf{A}\mathbf{x}_0, \ldots, \mathbf{A}^{k-1}\mathbf{x}_0)$. Ortonormální bázi $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k\}$ v $\mathcal{K}_k$ vypočteme modifikovaným Gramovým-Schmidtovým algoritmem, viz algoritmus AGSM v kapitole 1. Současně dostaneme také horní Hessenbergovu matici $\mathbf{H}_k = \mathbf{Q}_k^T\mathbf{A}\mathbf{Q}_k$, kde $\mathbf{Q}_k = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k)$. V $k-$té iteraci se spočtou koeficienty h_{ik}, $i = 1, 2, \ldots, k$, $k-$tého sloupce matice $\mathbf{H}_k$ a koeficient $h_{k+1,k} = \|\mathbf{q}_{k+1}\|_2$. Je-li $\|\mathbf{q}_{k+1}\|_2 = 0$, iterace končí, neboť $\mathcal{K}_i = \mathcal{K}_k$ pro $i > k$. Ritzova čísla a Ritzovy vektory v podprostoru $\mathcal{K}_i$ jsou už pak vlastní čísla a vlastní vektory matice $\mathbf{A}$.

Algoritmus 7.10 : Arnoldiho metoda

1. $\mathbf{x}_0 \neq \mathbf{o}$ je daný startovací vektor, $\mathbf{q}_1 = \mathbf{x}_0/\|\mathbf{x}_0\|_2$
2. **for** $k := 1, 2, \ldots$ **do**
3. $\quad \mathbf{q}_{k+1} := \mathbf{A}\mathbf{q}_k$
4. $\quad$ **for** $i := 1, 2, \ldots, k$ **do**
5. $\quad\quad h_{ik} := \mathbf{q}_{k+1}^T\mathbf{q}_i$

6. $\quad \mathbf{q}_{k+1} := \mathbf{q}_{k+1} - h_{ik}\mathbf{q}_i$
7. **end**
8. $h_{k+1,k} := \|\mathbf{q}_{k+1}\|_2$
9. **if** $h_{k+1,k} = 0$ **then stop**
10. $\mathbf{q}_{k+1} := \mathbf{q}_{k+1}/h_{k+1,k}$
11. urči vlastní čísla $\sigma_i^{(k)}$ a vlastní vektory $\mathbf{y}_i^{(k)}$ matice $\mathbf{H}_k$, tj.

$$\mathbf{H}_k\mathbf{Y}_k = \mathbf{Y}_k\mathbf{D}_k,$$

kde $\mathbf{Y}_k = (\mathbf{y}_1^{(k)}, \mathbf{y}_2^{(k)}, \ldots, \mathbf{y}_k^{(k)})$ a $\mathbf{D}_k = \mathrm{diag}(\sigma_1^{(k)}, \sigma_2^{(k)}, \ldots, \sigma_k^{(k)})$

12. $\mathbf{X}_k := \mathbf{Q}_k\mathbf{Y}_k$
13. **end**

Mezi Ritzovými čísly $\{\sigma_i^{(k)}\}_{i=1}^k$ a Ritzovými vektory $\{\mathbf{Q}_k\mathbf{y}_i^{(k)}\}_{i=1}^k$ bývají již při malém k, tj. po několika málo iteracích, obsaženy velmi dobré aproximace extrémních vlastních čísel a příslušných vlastních vektorů matice $\mathbf{A}$. Přitom extrémním vlastním číslem rozumíme vlastní číslo, které je výrazně odděleno od zbývajících vlastních čísel. Pro zjištění konvergence lze využít vztah

$$\|(\mathbf{A} - \sigma_i^{(k)}\mathbf{I})\mathbf{Q}_k\mathbf{y}_i^{(k)}\|_2 = |h_{k+1,k}| \cdot |[\mathbf{y}_i^{(k)}]_k|\,,$$

kde $[\mathbf{y}_i^{(k)}]_k$ je k-tý prvek vektoru $\mathbf{y}_i^{(k)}$, viz. [45]. Je-li tedy číslo $|h_{k+1,k}| \cdot |[\mathbf{y}_i^{(k)}]_k|$ dostatečně malé, je $i-$té Ritzovo číslo a $i-$tý Ritzův vektor dostatečně přesnou aproximací nějakého vlastního čísla a odpovídajícího vlastního vektoru matice $\mathbf{A}$.

S rostoucím počtem iterací vzrůstá dimenze Krylovových podprostorů, což má za následek rychlý nárůst objemu výpočtů v každé iteraci. Proto v praktických implementacích běží Arnoldiho metoda jen několik iterací a pak se provede restart s vhodně zvoleným novým startovacím vektorem. Několik opakování restartovaného Arnoldiho procesu obvykle poskytne vynikající aproximace extrémních vlastních čísel a odpovídajících vlastních vektorů při přijatelném objemu výpočtů.

Lanczosova metoda je modifikace Arnoldiho metody pro případ, kdy matice $\mathbf{A}$ je symetrická. Pak $\mathbf{H}_k$ je symetrická třídiagonální matice, což vede ke zjednodušení. Algoritmus Lanczosovy metody dostaneme, když označíme $\alpha_k = h_{kk}$, $\beta_k = h_{k+1,k} = h_{k,k+1}$, položíme $\beta_0 = 0$ a řádky 4 až 10 v Arnoldiho algoritmu nahradíme takto:

$\alpha_k := \mathbf{q}_{k+1}^T\mathbf{q}_k$
$\mathbf{q}_{k+1} := \mathbf{q}_{k+1} - \alpha_k\mathbf{q}_k - \beta_{k-1}\mathbf{q}_{k-1}$
$\beta_k := \|\mathbf{q}_{k+1}\|_2$
if $\beta_k = 0$ **then stop**
$\mathbf{q}_{k+1} := \mathbf{q}_{k+1}/\beta_k$

V důsledku zaokroulovacích chyb dochází v Lancozsově metodě poměrně rychle ke ztrátě ortogonality vektorů $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$ a proto je třeba čas od času provádět jejich *reortogonalizaci*, více k tomu viz [27], [45], [5] aj.

Profesionální programy založené na Arnoldiho a Lanczově metodě umožňují efektivní výpočet několika vlastních čísel a vektorů velkých řídkých matic, viz soubor programů ARPACK [27], který využívá také funkce `eigs` v MATLABu.

7.2.8. Jacobiho metoda

je jednou z nejstarších metod pro výpočet všech vlastních čísel symetrických matic. Položíme $\mathbf{A}_0 = \mathbf{A}$ a počítáme

$$\mathbf{A}_{k+1} = \mathbf{J}_{k+1}^T \mathbf{A}_k \mathbf{J}_{k+1}\,,$$

kde

$$\mathbf{J}_{k+1} \equiv \mathbf{J}(p_k, q_k, \theta_k) = \begin{array}{c} \\ \\ \\ \\ p_k \\ \\ \\ \\ q_k \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccccccc} & & & & & p_k & & & & q_k & & \\ \end{array} \\ \begin{pmatrix} 1 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & c_k & 0 & \dots & 0 & s_k & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & -s_k & 0 & \dots & 0 & c_k & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \ddots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 \end{pmatrix} \end{array}$$

$$c_k = \cos\theta_k\,, \qquad s_k = \sin\theta_k\,,$$

je *Jacobiho matice rovninné rotace* s úhlem θ_k vybraným tak, aby se anuloval jeden pár symetricky položených prvků $a_{p_k q_k}^{(k)} = a_{q_k p_k}^{(k)}$ matice $\mathbf{A}_k$. Matici $\mathbf{J}(p_k, q_k, \theta_k)$ dostaneme z jednotkové matice tak, že v ní nahradíme prvky v pozicích (p_k, p_k), (p_k, q_k), (q_k, p_k) a (q_k, q_k) postupně čísly c_k, s_k, $-s_k$ a c_k. Matice $\mathbf{J}(p_k, q_k, \theta_k)$ je ortonormální.

Označme $p = p_k$, $q = q_k$, $a = a_{pp}^{(k)}$, $b = a_{pq}^{(k)} \neq 0$, $d = a_{qq}^{(k)}$, $c = c_k$, $s = s_k$. Koeficienty c a s určíme tak, aby matice

$$\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a & b \\ b & d \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \begin{pmatrix} c^2a - 2csb + s^2d & c^2b + cs(a-d) - s^2b \\ c^2b + cs(a-d) - s^2b & c^2d + 2csb + s^2a \end{pmatrix}$$

byla diagonální. Rovnici $c^2b + cs(a-d) - s^2b = 0$ dělíme $-c^2b$ a dostaneme

$$t^2 + 2\tau t - 1 = 0\,, \qquad \text{kde} \qquad \tau = \frac{d-a}{2b} \quad \text{a} \quad t = s/c = \operatorname{tg}\theta\,,$$

takže

$$c = 1/\sqrt{1+t^2} \quad \text{a} \quad s = ct\,.$$

Za t zvolíme kořen kvadratické rovnice s menší absolutní hodnotou, což je

$$t = \frac{\operatorname{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}\,,$$

jak se snadno přesvědčíme. Protože $|t| \leq 1$, je $|\theta| \leq \frac{1}{4}\pi$, což je numericky výhodné, jak se ukazuje v [30].

Matice $\mathbf{J}_{k+1}$ nesestavujeme, pomocí indexů p_k, q_k, koeficientů c_k, s_k a matice $\mathbf{A}_k$ přímo určíme transformovanou matici $\mathbf{A}_{k+1}$. Výpočet organizujeme v cyklech: v jednom cyklu procházíme postupně všechny poddiagonální prvky a když absolutní hodnota některého z nich je větší než zadaná tolerance, anulujeme ho pomocí Jacobiho matice rovinné rotace. V dalších krocích se mohou vynulované pozice opět zaplnit. Dá se však ukázat, že součet čtverců nediagonálních prvků se v každém cyklu zmenší nejméně $q-$krát, kde $q < 1$ je konstanta, což znamená, že posloupnost matic konverguje k diagonální matici alespoň lineárně, asymptoticky je konvergence dokonce kvadratická.

Protože $\mathbf{A}_k = \hat{\mathbf{J}}_k^T \mathbf{A} \hat{\mathbf{J}}_k$, kde $\hat{\mathbf{J}}_k = \mathbf{J}_1 \mathbf{J}_2 \dots \mathbf{J}_k$, jsou matice $\mathbf{A}_k$ podobné s maticí $\mathbf{A}$ a mají tedy stejná vlastní čísla. To znamená, že když jsou mimodiagonální prvky matice $\mathbf{A}_k$ dostatečně malé, lze její diagonální prvky považovat za dostatečně dobré aproximace vlastních čísel matice $\mathbf{A}$. $i-$tý sloupec matice $\hat{\mathbf{J}}_k$ je pak aproximací vlastního vektoru příslušného k odpovídající aproximaci $a_{ii}^{(k)}$ vlastního čísla.

Jacobiho metoda se jednoduše programuje, vlastní čísla dokáže spočítat s velkou přesností. Je však poměrně pracná, jeden cyklus vyžaduje řádově stejný počet operací jako celý výpočet QR metodou. I když k dosažení požadované přesnosti stačí obvykle provést pět až deset cyklů, je Jacobiho metoda pět až desetkrát pracnější než QR metoda. V poslední době získala Jacobiho metoda znovu jistou popularitu díky tomu, že ji lze poměrně snadno a efektivně implementovat na paralelních počítačích.

7.2.9. Metoda bisekce

Každou symetrickou matici můžeme pomocí Householderových reflexí transformovat na podobnou třídiagonální matici. Ukážeme si jednoduchou metodu, jak určit vybraná vlastní čísla takové matice. Nechť tedy

$$\begin{pmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & b_3 & & \\ & \dots & \dots & \dots & \\ & & b_{n-1} & a_{n-1} & b_n \\ & & & b_n & a_n \end{pmatrix} \tag{7.13}$$

a definujme polynomy $p_0(x), p_1(x), \dots, p_n(x)$ předpisem

$$p_0(x) = 1, \quad p_r(x) = \det(\mathbf{A}_r - x\mathbf{I}) \quad \text{pro } r = 1, 2, \dots, n,$$

kde $\mathbf{A}_r$ je submatice tvořená prvními r řádky a sloupci matice $\mathbf{A}$. Zřejmě $p_1(x) = a_1 - x$ a pro další polynomy lze snadno odvodit rekurenci

$$p_r(x) = (a_r - x)p_{r-1}(x) - b_r^2 p_{r-2}(x)\,, \qquad r = 2, 3, \dots, n\,.$$

Protože vyhodnocení $p_n(x_0)$ pro dané x_0 je výpočetně nenáročné, lze kořen charakteristického polynomu $p_n(x)$ efektivně určit metodou bisekce. Je-li $a < b$ a $p_n(a)p_n(b) < 0$, pak v intervalu $\langle a, b\rangle$ leží vlastní číslo, které pro danou přesnost ε určíme takto:

while $b - a > \varepsilon$ **do**
  $x := (a + b)/2$
  if $p_n(a)p_n(x) > 0$ **then** $a := x$ **else** $b := x$

Někdy je třeba spočítat $k-$té největší vlastní číslo matice $\mathbf{A}$ pro předepsané k. Ukažme si, jak to lze provést.

Předpokládejme, že matice $\mathbf{A}$ je *ireducibilní*, tj. že všechny koeficienty b_i jsou nenulové. To není žádné omezení, neboť když $b_i = 0$, pak je matice

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_i & \mathbf{O}^T \\ \mathbf{O} & \mathbf{C}_{n-i} \end{pmatrix}$$

blokově diagonální, takže stačí počítat vlastní čísla menších matic $\mathbf{A}_i$ a $\mathbf{C}_{n-i}$. Pro symetrickou třídiagonální ireducibilní matici $\mathbf{A}$ má posloupnost polynomů $\{p_r(x)\}_{r=0}^n$ následující vlastnost: je-li $V(\lambda)$ počet znaménkových změn v posloupnosti $p_0(\lambda), p_1(\lambda), \ldots, p_n(\lambda)$, pak $V(\lambda)$ je rovno počtu vlastních čísel menších než λ. Při výpočtu $V(\lambda)$ se přijímá tato konvence: je-li $p_{r-1}(\lambda) = 0$, považuje se znaménko $p_r(\lambda)$ za opačné ke znaménku $p_{r-1}(\lambda)$.

Předpokládejme, že $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Z Geršgorinovy věty plyne, že všechna vlastní čísla leží v intervalu $\langle a, b\rangle$, kde

$$a = \min_{1\leq i\leq n} \{a_i - |b_i| - |b_{i+1}|\}, \quad b = \max_{1\leq i\leq n} \{a_i + |b_i| + |b_{i+1}|\}, \quad b_1 = b_{n+1} = 0\,.$$

Vlastní číslo λ_k, $k = 1, 2, \ldots, n$, určíme následujícím algoritmem:

while $b - a > \varepsilon$ **do**
  $x := (a + b)/2$
  if $V(x) < k$ **then** $a := x$ **else** $b := x$

7.2.10. Metoda rozděl a panuj

Metoda rozděl a panuj (anglicky divide and conquer) označuje univerzální rekurzivní algoritmus, který řeší daný „rodičovský“ problém tak, že ho rozčlení na několik dílčích „dceřiných“ problémů a vhodnou kombinací řešení dceřiných problémů sestaví řešení problému rodičovského. Řešení každého dceřiného problému se získá stejným postupem jako řešení problému rodičovského. Tak postupně vzniká celá řada stále menších dceřiných problémů. Jakmile je dceřiný problém dostatečně malý, přestane se dělit a vyřeší se vhodnou elementární metodou.

Známým příkladem algoritmu divide-and-conquer (v dalším stručně DaC algoritmu) jsou třídicí algoritmy quick sort, merge sort nebo algoritmus rychlé Fourierovy transformace, viz kapitola 4.1.3.

Mezi metodami pro výpočet vlastních čísel a vektorů se pod metodou DaC míní metoda pro výpočet vlastních čísel a vektorů symetrické třídiagonální matice. Jde o metodu relativně mladou. Publikována byla v roce 1981, trvalo však více než 10 let, než se podařilo sestrojit spolehlivý, stabilní a rychlý algoritmus.

V současnosti je třídiagonalizace následovaná DaC metodou nejrychlejším algoritmem pro výpočet všech vlastních čísel a vektorů symetrické matice, více než dvakrát tak rychlá jako třídiagonalizace následovaná QR metodou s Wilkinsonovým posunem.

Vysvětleme si hlavní myšlenky, na nichž je DaC metoda založena. Nechť

$$\mathbf{A} = \left(\begin{array}{ccccc|ccccc} a_1 & b_1 & & & & & & & & \\ b_1 & \ddots & \ddots & & & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & & & \\ & & b_{m-1} & a_m & b_m & & & & & \\ \hline & & & b_m & a_{m+1} & b_{m+1} & & & & \\ & & & & b_{m+1} & \ddots & \ddots & & & \\ & & & & & \ddots & a_{n-1} & b_{n-1} & & \\ & & & & & & b_{n-1} & a_n & & \end{array}\right) =$$

$$= \left(\begin{array}{ccccc|ccccc} a_1 & b_1 & & & & & & & & \\ b_1 & \ddots & \ddots & & & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & & & \\ & & b_{m-1} & a_m - b_m & & & & & & \\ \hline & & & & & a_{m+1} - b_m & b_{m+1} & & & \\ & & & & & b_{m+1} & \ddots & \ddots & & \\ & & & & & & \ddots & a_{n-1} & b_{n-1} & \\ & & & & & & & b_{n-1} & a_n & \end{array}\right) +$$

$$+ \left(\begin{array}{ccc|cc} & & & & \\ & & & & \\ & & b_m & b_m & \\ \hline & & b_m & b_m & \\ & & & & \\ & & & & \end{array}\right) = \begin{pmatrix} \mathbf{A}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{A}_2 \end{pmatrix} + b_m \mathbf{v}\mathbf{v}^T, \quad \text{kde} \quad \mathbf{v} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Předpokládejme, že již máme spektrální rozklady $\mathbf{A}_1 = \mathbf{Q}_1 \Lambda_1 \mathbf{Q}_1^T$, $\mathbf{A}_2 = \mathbf{Q}_2 \Lambda_2 \mathbf{Q}_2^T$. Pak

$$\mathbf{A} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_2 \end{pmatrix} \left[\begin{pmatrix} \Lambda_1 & \mathbf{O} \\ \mathbf{O} & \Lambda_2 \end{pmatrix} + b_m \mathbf{u}\mathbf{u}^T \right] \begin{pmatrix} \mathbf{Q}_1^T & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_2^T \end{pmatrix},$$

kde

$$\mathbf{u} = \begin{pmatrix} \mathbf{Q}_1^T & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_2^T \end{pmatrix} \mathbf{v} = \begin{pmatrix} \text{poslední sloupec } \mathbf{Q}_1^T \\ \text{první sloupec } \mathbf{Q}_2^T \end{pmatrix}$$

Vlastní čísla matice $\mathbf{A}$ jsou stejná jako vlastní čísla podobné matice $\mathbf{D} + \varrho \mathbf{u}\mathbf{u}^T$, kde

$$\mathbf{D} = \begin{pmatrix} \Lambda_1 & \mathbf{O} \\ \mathbf{O} & \Lambda_2 \end{pmatrix}$$

je diagonální a $\varrho = b_m$. Bez ztráty obecnosti můžeme předpokládat, že diagonální prvky $d_1, d_2, \ldots, d_n$ matice $\mathbf{D}$ jsou uspořádány: $d_1 \leq d_2 \leq \cdots \leq d_n$. Předpokládejme dále, že vlastní číslo λ matice $\mathbf{D} + \varrho\mathbf{u}\mathbf{u}^T$ není vlastním číslem $\mathbf{D}$. Charakteristickou rovnici pak můžeme vyjádřit ve tvaru

$$\det(\mathbf{D} + \varrho\mathbf{u}\mathbf{u}^T - \lambda\mathbf{I}) = \det((\mathbf{D} - \lambda\mathbf{I})(\mathbf{I} + \varrho(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{u}\mathbf{u}^T)) = 0.$$

Protože $\mathbf{D} - \lambda\mathbf{I}$ je regulární, λ musí splňovat $\det(\mathbf{I} + \varrho(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{u}\mathbf{u}^T) = 0$. Matice $\mathbf{I} + \varrho(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{u}\mathbf{u}^T$ je součtem jednotkové matice a matice hodnosti jedna. Takový determinant lze snadno vyčíslit pomocí vztahu

$$\det(\mathbf{I} + \mathbf{x}\mathbf{y}^T) = 1 + \mathbf{x}^T\mathbf{y},$$

který je speciálním případem lemmatu známého pod anglickým názvem *matrix determinant lemma*. Proto

$$\det(\mathbf{I} + \varrho(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{u}\mathbf{u}^T) = 1 + \varrho\mathbf{u}^T(\mathbf{D} - \lambda\mathbf{I})^{-1}\mathbf{u} = 1 + \varrho\sum_{i=1}^{n}\frac{u_i^2}{d_i - \lambda} \equiv f(\lambda),$$

takže vlastní čísla $\mathbf{A}$ jsou kořeny tak zvané sekulární rovnice $f(\lambda) = 0$. Jestliže d_i jsou navzájem různá čísla a $u_i \neq 0$, funkce $f(\lambda)$ má graf, který je pro $n = 4$, $u_i = \frac{1}{2}$, $d_i = i$ a $\varrho > 0$ uveden na obrázku 7.1. Protože

$$f'(\lambda) = \varrho\sum_{i=1}^{n}\frac{u_i^2}{(d_i - \lambda)^2},$$

je funkce $f(\lambda)$ s výjimkou bodů d_i pro $\varrho > 0$ rostoucí a pro $\varrho < 0$ klesající. Kořeny $f(\lambda)$ leží po jednom v každém intervalu (d_i,d_{i+1}) a jeden další pro $\varrho > 0$ napravo od d_n a pro $\varrho < 0$ nalevo od d_1.
Kořeny sekulární rovnice lze najít velmi přesně pomocí několika kroků jisté varianty Newtonovy metody, viz [11].

Vlastní vektor příslušný vlastnímu číslu α matice $\mathbf{D} + \varrho\mathbf{u}\mathbf{u}^T$ je $(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u}$. Skutečně,

$$(\mathbf{D} + \varrho\mathbf{u}\mathbf{u}^T)[(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u}] = (\mathbf{D} - \alpha\mathbf{I} + \alpha\mathbf{I} + \varrho\mathbf{u}\mathbf{u}^T)(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u} =$$

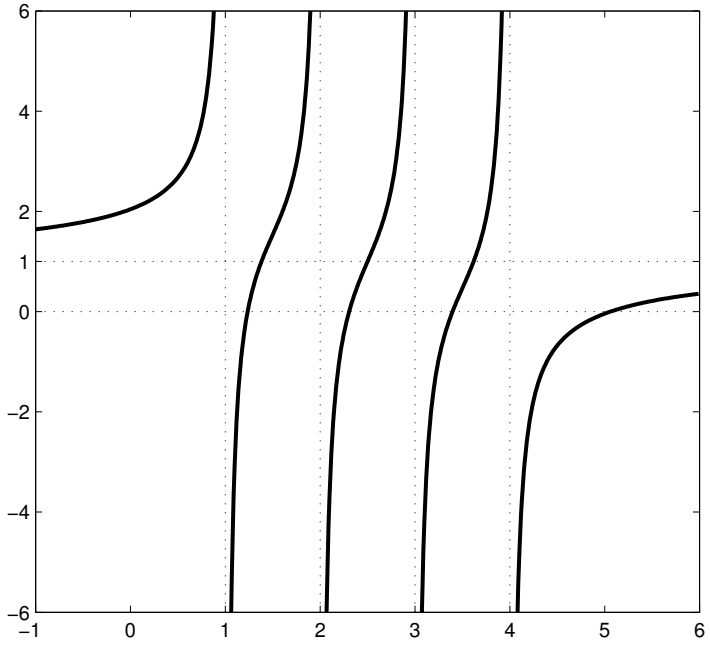
$$= \mathbf{u} + \alpha(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u} + \mathbf{u}[\varrho\mathbf{u}^T(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u}] = \mathbf{u}f(\alpha) + \alpha[(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u}] =$$

$$= \alpha[(\mathbf{D} - \alpha\mathbf{I})^{-1}\mathbf{u}].$$

Výpočet vlastních vektorů je třeba ještě modifikovat, při výpočtu $(\mathbf{D}-\alpha\mathbf{I})^{-1}\mathbf{u}$ totiž mohou vznikat velké zaokrouhlovací chyby, více o tom viz [11].

Jestliže $\tilde{\mathbf{Q}}$ je ortonormální matice vlastních vektorů matice $\mathbf{D} + \varrho\mathbf{u}\mathbf{u}^T$, pak

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_2 \end{pmatrix}\tilde{\mathbf{Q}}$$



Obr. 7.1: Graf funkce $f(\lambda) = 1 + \frac{0.5}{1-\lambda} + \frac{0.5}{2-\lambda} + \frac{0.5}{3-\lambda} + \frac{0.5}{4-\lambda}$

je ortonormální matice vlastních vektorů matice $\mathbf{A}$.

Dá se ukázat, že vyloučené případy $d_i = d_{i+1}$ a $u_i = 0$ nepředstavují komplikaci, ale naopak zjednodušení a urychlení celého výpočtu, viz [11]. I když hlavní myšlenka algoritmu je jednoduchá, efektivní implementace obsahuje celou řadu specifických postupů, bez nichž by CaD algoritmus nebyl dost dobře použitelný. Pokus o naivní implementaci se proto velmi důrazně nedoporučuje, téměř jistě by Vám přinesl zklamání. Kvalitní implementace DaC algoritmu je součástí balíku programů LAPACK [1].

7.2.11. Zobecněný problém vlastních čísel

spočívá v určení vlastního čísla λ a nenulového vlastního vektoru $\mathbf{x}$ tak, aby platilo

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x}\,.$$

Je-li alespoň jedna z matic $\mathbf{A}$, $\mathbf{B}$ regulární, lze zobecněný problém vlastních čísel převést na standardní problém vlastních čísel, buďto

$$(\mathbf{B}^{-1}\mathbf{A})\mathbf{x} = \lambda\mathbf{x} \quad \text{nebo} \quad (\mathbf{A}^{-1}\mathbf{B})\mathbf{x} = (1/\lambda)\mathbf{x}\,.$$

Tuto transformaci však obecně nelze doporučit, neboť při ní může dojít ke

- ztrátě přesnosti způsobené zaokrouhlovacími chybami při výpočtu součinu matic, zejména když matice $\mathbf{A}$ nebo $\mathbf{B}$ je špatně podmíněná;
- ztrátě symetrie, když $\mathbf{A}$ i $\mathbf{B}$ jsou symetrické.

Když jsou matice $\mathbf{A}$ a $\mathbf{B}$ symetrické a jedna z nich je pozitivně definitní, pak lze symetrii zachovat užitím Choleského rozkladu. Tak je-li například $\mathbf{B} = \mathbf{L}\mathbf{L}^T$, můžeme zobecněný problém vlastních čísel převést na standardní problém

$$(\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T})\mathbf{y} = \lambda\mathbf{y}$$

a pak dopočítat $\mathbf{x}$ ze soustavy rovnic $\mathbf{L}^T\mathbf{x} = \mathbf{y}$. Tento přístup však pořád neodstraňuje případný rušivý vliv zaokrouhlovacích chyb, navíc je nepoužitelný v případě, že obě matice jsou singulární.

Numericky lepší přístup, který funguje i případě, že matice $\mathbf{A}$ i $\mathbf{B}$ jsou singulární, je *QZ metoda*. Je založena na *zobecněném Schurově rozkladu* matic $\mathbf{A}$, $\mathbf{B}$, viz [30]:

Nechť $\mathbf{A}$ a $\mathbf{B}$ jsou reálné matice řádu n. Pak existují unitární matice $\mathbf{Q}$ a $\mathbf{Z}$ takové, že $\mathbf{Q}^H\mathbf{A}\mathbf{Z} = \mathbf{T} = \{t_{ij}\}_{i,j=1}^n$ a $\mathbf{Q}^H\mathbf{B}\mathbf{Z} = \mathbf{S} = \{s_{ij}\}_{i,j=1}^n$ jsou horní trojúhelníkové matice.

Protože

$$\det(\mathbf{A} - \lambda\mathbf{B}) = \det(\mathbf{Q}(\mathbf{T} - \lambda\mathbf{S})\mathbf{Z}^H) = \det(\mathbf{Q})\det(\mathbf{Z}^H)\prod_{i=1}^{n}(t_{ii} - \lambda s_{ii})\,,$$

vlastní čísla $\lambda_i = t_{ii}/s_{ii}$ (pro $s_{ii} = 0$ klademe $\lambda_i = \infty$). Existuje také *zobecněný reálný Schurův rozklad* matic $\mathbf{A}$, $\mathbf{B}$:

Nechť $\mathbf{A}$ a $\mathbf{B}$ jsou reálné matice řádu n. Pak existují ortonormální matice $\mathbf{Q}$ a $\mathbf{Z}$ takové, že $\mathbf{Q}^T\mathbf{A}\mathbf{Z} = \mathbf{T}$ je horní blokově trojúhelníková matice v reálném Schurově tvaru (7.5) *a $\mathbf{Q}^T\mathbf{B}\mathbf{Z} = \mathbf{S}$ je horní trojúhelníková matice s nezápornými diagonálními prvky. (Je-li $\mathbf{A}$ resp. $\mathbf{B}$ symetrická, je $\mathbf{T}$ resp. $\mathbf{S}$ diagonální.)*

QZ metoda je iterační metoda, jejímž výsledkem je QZ rozklad matic $\mathbf{A}$ a $\mathbf{B}$ a následný výpočet vlastních čísel a vektorů zobecněného problému vlastních čísel, algoritmus je uveden např. v [30]. V MATLABu lze k výpočtu zobecněného Schurova rozkladu použít funkci `qz`, zobecněný problém vlastních čísel umožňují v MATLABu funkce `eig` a `eigs`.

Zobecněný problém vlastních čísel je možné řešit také modifikacemi dalších metod, například metody iterací v podprostorech [5], Jacobiho [5], Arnoldiho a Lanczose [27].

7.2.12. Výpočet singulárních čísel a vektorů

Připomeňme, že $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ je singulární rozklad matice $\mathbf{A}$ typu (m, n), jestliže $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)$ je ortonormální matice řádu m, $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ je ortonormální matice řádu n a $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)$ je diagonální matice typu (m, n), kde $p = \min(m, n)$ a $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. σ_i jsou singulární čísla a $\mathbf{u}_i$ resp. $\mathbf{v}_i$ je $i-$tý levý resp. pravý singulární vektor. Zřejmě

$$\begin{aligned}&\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i\,,\\ &\mathbf{u}_i^T\mathbf{A} = \sigma_i\mathbf{v}_i^T,\end{aligned} \quad\Longleftrightarrow\quad \begin{aligned}&\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i\,,\\ &\mathbf{A}^T\mathbf{u}_i = \sigma_i\mathbf{v}_i\,,\end{aligned} \quad\Longleftrightarrow\quad \begin{pmatrix}\mathbf{O} & \mathbf{A}\\ \mathbf{A}^T & \mathbf{O}\end{pmatrix}\begin{pmatrix}\mathbf{u}_i\\ \mathbf{v}_i\end{pmatrix} = \sigma_i\begin{pmatrix}\mathbf{u}_i\\ \mathbf{v}_i\end{pmatrix}.$$

Singulární čísla a singulární vektory matice $\mathbf{A}$ lze tedy určit pomocí vlastních čísel a vlastních vektorů symetrické matice

$$\hat{\mathbf{A}} = \begin{pmatrix}\mathbf{O} & \mathbf{A}\\ \mathbf{A}^T & \mathbf{O}\end{pmatrix}.$$

V dalším pro konkrétnost předpokládejme, že $m \geq n$ (to není žádné omezení, neboť v případě $n > m$ lze singulární rozklad matice $\mathbf{A}$ sestavit pomocí singulárního rozkladu $\mathbf{A}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T$ matice $\mathbf{A}^T$). Nechť $\mathbf{U}_1$ je matice typu (m, n) sestavená z prvních n sloupců matice $\mathbf{U}$ a $\mathbf{U}_2$ je matice typu $(m, m-n)$ sestavená z posledních $m-n$ sloupců matice $\mathbf{U}$, tj. $\mathbf{U} = (\mathbf{U}_1, \mathbf{U}_2)$. Definujeme-li ortonormální matici $\mathbf{Q}$ předpisem

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{U}_1 & -\mathbf{U}_1 & \sqrt{2}\mathbf{U}_2 \\ \mathbf{V} & \mathbf{V} & \mathbf{O} \end{pmatrix},$$

pak snadno ověříme, že

$$\mathbf{Q}^T\hat{\mathbf{A}}\mathbf{Q} = \operatorname{diag}(\sigma_1, \ldots, \sigma_n, -\sigma_1, \ldots, -\sigma_n, 0, \ldots, 0) \equiv \mathbf{D} \quad \text{nebo-li} \quad \hat{\mathbf{A}}\mathbf{Q} = \mathbf{Q}\mathbf{D}\,.$$

Jestliže $\hat{\mathbf{Q}}$ je ortonormální matice vlastních vektorů matice $\hat{\mathbf{A}}$ a $\hat{\mathbf{D}}$ je odpovídající diagonální matice vlastních čísel, tj. $\hat{\mathbf{A}}\hat{\mathbf{Q}} = \hat{\mathbf{Q}}\hat{\mathbf{D}}$, pak ze struktury matic $\mathbf{Q}$ a $\mathbf{D}$ plyne existence permutační matici $\mathbf{P}$ takové, že $\hat{\mathbf{Q}}\mathbf{P} = \mathbf{Q}$ a $\mathbf{P}^T\hat{\mathbf{D}}\mathbf{P} = \mathbf{D}$. Z matic $\mathbf{D}$ a $\mathbf{Q}$ pak získáme $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, $\mathbf{U} = (\mathbf{U}_1, \mathbf{U}_2)$ a $\mathbf{V}$.

Jak uvidíme, singulární čísla a singulární vektory matice $\mathbf{A}$ úzce souvisejí také s vlastními čísly a vlastními vektory symetrických matic $\mathbf{A}^T\mathbf{A}$ a $\mathbf{A}\mathbf{A}^T$. Platí totiž

$$\begin{aligned} \mathbf{A}^T\mathbf{A} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T \qquad &\Longleftrightarrow \qquad (\mathbf{A}^T\mathbf{A})\mathbf{V} = \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})\,, \\ \mathbf{A}\mathbf{A}^T = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T \qquad &\Longleftrightarrow \qquad (\mathbf{A}\mathbf{A}^T)\mathbf{U} = \mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^T)\,. \end{aligned}$$

Protože $\mathbf{\Sigma}^T\mathbf{\Sigma} = \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$, $\mathbf{\Sigma}\mathbf{\Sigma}^T = \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2, 0, \ldots, 0)$, vidíme, že pravé singulární vektory $\mathbf{v}_i$ resp. levé singulární vektory $\mathbf{u}_i$ jsou vlastní vektory matice $\mathbf{A}^T\mathbf{A}$ resp. $\mathbf{A}\mathbf{A}^T$, a že odpovídající singulární čísla σ_i, $i = 1, 2, \ldots, n$, jsou nezáporné druhé odmocniny vlastních čísel matice $\mathbf{A}^T\mathbf{A}$ i $\mathbf{A}\mathbf{A}^T$. K výpočtu singulárních čísel a vektorů lze tedy použít metody pro výpočet vlastních čísel symetrických matic $\mathbf{A}^T\mathbf{A}$ a $\mathbf{A}\mathbf{A}^T$. Jednoduchý postup může vypadat třeba takto:

1. vypočti $\mathbf{C} = \mathbf{A}^T\mathbf{A}$;
2. pomocí QR metody vypočti $\mathbf{V}_1^T\mathbf{C}\mathbf{V}_1 = \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$;
3. proveď QR rozklad matice $\mathbf{A}\mathbf{V}_1$ se sloupcovým pivotováním: $\mathbf{U}^T(\mathbf{A}\mathbf{V}_1)\mathbf{P} = \mathbf{R}$, kde $\mathbf{R} = \{r_{ij}\}_{i,j=1}^n$ je horní trojúhelníková matice a $\mathbf{P}$ je permutační matice zvolená tak, aby $r_{11} \geq r_{22} \geq \cdots \geq r_{nn}$.

Jak se snadno přesvědčíme, matice $\mathbf{R}^T\mathbf{R}$ je diagonální, což znamená, že sloupce matice $\mathbf{R}$ jsou navzájem ortogonální, a protože $\mathbf{R}$ je horní trojúhelníková, musí být diagonální. Proto $\mathbf{R} = \mathbf{\Sigma}$ a $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, kde $\mathbf{V} = \mathbf{V}_1\mathbf{P}$, je hledaný singulární rozklad. Při výpočtu $\mathbf{A}^T\mathbf{A}$ však dochází ke ztrátě informace. Proto se používají lepší metody, které se přímému formování součinu $\mathbf{A}^T\mathbf{A}$ vyhýbají.

Stručně lze standardní postup výpočtu singulárního rozkladu popsat takto:

1) Pomocí finitního Golub-Kahanova bidiagonalizačního algoritmu obdržíme rozklad $\mathbf{A} = \mathbf{U}_1\mathbf{B}\mathbf{V}_1^T$, kde $\mathbf{B}$ je bidiagonální matice, jejíž nenulové prvky se mohou nacházet jen na hlavní diagonále a v první naddiagonále, a kde $\mathbf{U}_1$ i $\mathbf{V}_1$ jsou ortonormální matice, viz např. [30], [11].

2) Využijeme toho, že vlastní čísla třídiagonální matice $\mathbf{T}_r = \mathbf{B}^T\mathbf{B}$ jsou kvadráty singulárních čísel matice $\mathbf{B}$ a že vlastní vektory $\mathbf{T}_r$ jsou pravé singulární vektory $\mathbf{B}$. Nechť $\mathbf{T}_r = \mathbf{V}_2\mathbf{D}_r\mathbf{V}_2^T$ je spektrální rozklad matice $\mathbf{T}_r$. Použijeme-li $\mathbf{T}_\ell = \mathbf{B}\mathbf{B}^T$, ze spektrálního rozkladu $\mathbf{T}_\ell = \mathbf{U}_2^T\mathbf{D}_\ell\mathbf{U}_2$ dostaneme levé singulární vektory $\mathbf{B}$. Výpočet vlastních čísel a vektorů matic $\mathbf{T}_r$ resp. $\mathbf{T}_\ell$ lze efektivně provést pomocí modifikace QR metody známé jako *dqds* algoritmus, viz např. [11].

3) Nakonec dostaneme singulární rozklad $\mathbf{A} = \mathbf{U}_1\mathbf{U}_2\mathbf{\Sigma}(\mathbf{V}_1\mathbf{V}_2)^T$, matici $\mathbf{\Sigma}$ sestavíme z odmocnin diagonálních prvků matice $\mathbf{D}_r$ nebo $\mathbf{D}_\ell$.

V MATLABu singulární rozklad počítá funkce `svd`. Pro řídké matice je k dispozici funkce `svds`, která umožňuje vypočítat několik singulárních čísel a odpovídajících singulárních vektorů tak, že pomocí funkce `eigs` počítá několik vlastních čísel a odpovídajících vlastních vektorů matice $\hat{\mathbf{A}}$.

8. Obyčejné diferenciální rovnice: počáteční úlohy

V této kapitole se budeme zabývat problematikou numerického řešení počátečních úloh pro obyčejné diferenciální rovnice.

8.1. Formulace, základní pojmy

Počáteční problém pro ODR1 spočívá v určení funkce $y(t)$, která vyhovuje diferenciální rovnici

$$y'(t) = f(t, y(t)) \tag{8.1}$$

a splňuje počáteční podmínku

$$y(a) = \eta. \tag{8.2}$$

Je-li v nějakém okolí D bodu $[a, \eta]$ funkce $f(t, y)$ spojitá a splňuje-li v tomto okolí *Lipschitzovu podmínku* s konstantou L vzhledem k proměnné y, tj. platí-li

$$|f(t, u) - f(t, v)| \leq L|u - v| \quad \forall\, [t, u]\,, [t, v] \in D\,, \tag{8.3}$$

pak bodem $[a, \eta]$ prochází jediné řešení $y(t)$ rovnice (8.1). Jestliže funkce f má v D omezenou parciální derivaci vzhledem k proměnné y, tj. když $|\partial_y f(t, y)| \leq L$, Lipschitzova podmínka (8.3) platí. Jiný standardní výsledek říká, že když je funkce f spojitá na $\langle a, b\rangle \times \mathbb{R}$ a splňuje tam Lipschitzovu podmínku, pak počáteční problém (8.1), (8.2) má jediné řešení definované v celém intervalu $\langle a, b\rangle$.

Počáteční problém pro soustavu ODR1 znamená určit funkce $y_1(t), \ldots, y_d(t)$ splňující diferenciální rovnice

$$y_j'(t) = f_j(t, y_1(t), \ldots, y_d(t)), \quad j = 1, 2, \ldots, d,$$

a počáteční podmínky

$$y_j(a) = \eta_j, \quad j = 1, 2, \ldots, d.$$

Stručnější vektorový zápis je

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \;\; \mathbf{y}(a) = \boldsymbol{\eta}\,, \tag{8.4}$$

kde

$$\mathbf{y}(t) = (y_1(t), y_2(t), \ldots, y_d(t))^T, \qquad \mathbf{y}'(t) = (y_1'(t), y_2'(t), \ldots, y_d'(t))^T,$$
$$\mathbf{f}(t, \mathbf{y}(t)) = (f_1(t, \mathbf{y}(t)), f_2(t, \mathbf{y}(t)), \ldots, f_d(t, \mathbf{y}(t)))^T, \qquad \boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_d)^T.$$

Je-li v okolí D bodu $[a, \boldsymbol{\eta}]$ funkce $\mathbf{f}(t, \mathbf{y})$ spojitá a splňuje tam vzhledem k proměnné $\mathbf{y}$ Lipschitzovu podmínku s konstantou L , tj. platí-li

$$\|\mathbf{f}(t, \mathbf{u}) - \mathbf{f}(t, \mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\| \quad \forall\, [t, \mathbf{u}]\,, [t, \mathbf{v}] \in D\,, \tag{8.5}$$

pak bodem $[a, \boldsymbol{\eta}]$ prochází jediné řešení počáteční úlohy (8.4). Má-li $\mathbf{f}$ v D omezené parciální derivace $\{\partial f_i(t, \mathbf{y})/\partial y_j\}_{i,j=1}^d$, pak Lipschitzova podmínka (8.5) platí. Je-li $D = \langle a, b\rangle \times \mathbb{R}^d$, jediné řešení existuje v celém intervalu $\langle a, b\rangle$.

Rovnice vyššího řádu. Počáteční problém pro obyčejnou diferenciální rovnici řádu d,

$$y^{(d)}(t) = F(t, y(t), y'(t), \ldots, y^{(d-1)}(t)) \tag{8.6}$$

s počátečními podmínkami

$$y(a) = \eta_1, \quad y'(a) = \eta_2, \ldots, y^{(d-1)}(a) = \eta_d\,,$$

lze snadno převést na počáteční problém (8.4) pro d rovnic řádu prvního:

$$\begin{aligned}
y_1'(t) &= y_2(t), & y_1(a) &= \eta_1\,,\\
y_2'(t) &= y_3(t), & y_2(a) &= \eta_2\,,\\
&\vdots & &\vdots\\
y_{d-1}'(t) &= y_d(t), & y_{d-1}(a) &= \eta_{d-1}\,,\\
y_d'(t) &= F(t, y_1(t), y_2(t), \ldots, y_d(t)), & y_d(a) &= \eta_d\,,
\end{aligned}$$

kde $y_1(t) = y(t)$, $y_2(t) = y'(t)$, …, $y_d(t) = y^{(d-1)}(t)$.

V dalším budeme vždy předpokládat, že uvažovaná počáteční úloha má v intervalu $\langle a, b\rangle$ jediné řešení. Budeme také předpokládat, že funkce $\mathbf{f}(t, \mathbf{y})$ má tolik spojitých derivací, kolik jich v dané situaci bude zapotřebí.

Jedna rovnice prvního řádu s jednou neznámou funkcí je v aplikacích méně významná než soustavy rovnic. Metody přibližného řešení se však snadněji odvodí pro jednu rovnici a lze je aplikovat bezprostředně i na soustavy. Také analýza numerických metod je pro jednu rovnici podstatně snadnější. Proto se v následujícím výkladu převážně omezíme jen na jednu rovnici. Z velkého množství metod uvedeme ty, které jsou pro své dobré vlastnosti široce používány. Mezi ně bezesporu patří metody implementované do Matlabu a právě na ně se v tomto textu zaměříme.

Numerickým řešením počáteční úlohy rozumíme výpočet přibližných hodnot hledaného řešení $y(t)$ v bodech t_n dosti hustě vykrývajících interval $\langle a, b\rangle$. Nechť tedy

$$a = t_0 < t_1 < \cdots < t_Q = b$$

je *dělení* intervalu $\langle a, b\rangle$. Body t_n jsou *uzly*, vzdálenost $\tau_n = t_{n+1} - t_n$ dvou sousedních uzlů je *délka kroku* Jsou-li všechny kroky stejně dlouhé, tj. když $\tau_n = \tau = (b-a)/Q$, hovoříme o *rovnoměrném* (*ekvidistantním*) dělení intervalu $\langle a, b\rangle$. V tom případě je $t_n = a + n\tau$, $n = 0, 1, \ldots, Q$. Hodnotu přesného řešení v uzlu t_n budeme značit $y(t_n)$ a hodnotu přibližného řešení y_n. Jestliže se nám podaří najít přibližné řešení y_n, $n = 0, 1, \ldots, Q$, můžeme vypočítat přibližnou hodnotu řešení $y(t)$ v libovolném bodě $t \in \langle a, b\rangle$ interpolací.

Numerická metoda pro řešení počáteční úlohy (8.1), (8.2) je předpis pro postupný výpočet aproximací $y_1, y_2, \ldots, y_Q$, $y_0 = \eta$ z počáteční podmínky. Metoda se nazývá

k-kroková, závisí-li předpis pro výpočet aproximace y_{n+1} na předchozích aproximacích $y_n, y_{n-1}, \ldots, y_{n-k+1}$. Speciálně *jednokroková metoda* počítá přibližné řešení y_{n+1} v uzlu t_{n+1} jen pomocí znalosti přibližného řešení y_n v uzlu t_n, přibližná řešení y_{n-1}, $y_{n-2}, \ldots$ spočtená v předchozích uzlech t_{n-1}, $t_{n-2}, \ldots$ nepoužívá. Výpočet y_{n+1} nazýváme krokem metody od t_n do t_{n+1} (stručně *krokem*). Při popisu kroku budeme u délky kroku $\tau_n = t_{n+1} - t_n$ vypouštět index, tj. píšeme $\tau_n = \tau$.

8.2. Eulerovy metody

Explicitní Eulerova metoda. Nejjednodušší numerickou metodou pro řešení úlohy (8.1), (8.2) je *explicitní Eulerova metoda* (stručně EE metoda). EE metodu snadno odvodíme z Taylorovy formule

$$y(t_{n+1}) = y(t_n + \tau) = y(t_n) + \tau y'(t_n) + \tfrac{1}{2}\tau^2 y''(\xi_n), \quad \xi_n \in (t_n, t_{n+1}). \tag{8.7}$$

Uvážíme-li, že $y'(t_n) = f(t_n, y(t_n))$ a zanedbáme-li člen $\frac{1}{2}\tau^2 y''(\xi_n)$, dostaneme

$$y(t_{n+1}) \approx y(t_n) + \tau f(t_n, y(t_n)).$$

Výrazy $y(t_n)$ a $y(t_{n+1})$ nahradíme jejich přibližnými hodnotami y_n a y_{n+1}, znaménko přibližné rovnosti $\approx$ nahradíme znaménkem rovnosti a obdržíme předpis EE metody

$$y_{n+1} = y_n + \tau f(t_n, y_n)\,. \tag{8.8}$$

O explicitní metodě hovoříme proto, že pro určení y_{n+1} máme explicitní vzorec: dosazením známé hodnoty y_n do pravé strany rovnice (8.8) obdržíme hledanou hodnotu y_{n+1}. V anglicky psané literatuře se EE metoda označuje jako *Euler method* resp. *explicit Euler method* resp. *forward Euler method*.

Diskretizační chyby. Přesnost numerické metody měříme pomocí tzv. *lokální diskretizační chyby* (anglicky *local truncation error*)

$$\mathrm{lte}_n = y(t_{n+1}) - y(t_n) - \tau f(t_n, y(t_n))\,.$$

Lokální diskretizační chyba je tedy chyba, které se dopustíme v jednom kroku metody za tzv. *lokalizačního předpokladu*, že $y_n = y(t_n)$ je přesné řešení počáteční úlohy (8.1), (8.2). Z (8.7) pro EE metodu plyne

$$\mathrm{lte}_n = \tfrac{1}{2}\tau^2 y''(\xi_n) \;\text{ a tedy }\; |\mathrm{lte}_n| \le C\tau^2, \;\text{ kde }\; C = \tfrac{1}{2}\max_{t_n \le t \le t_{n+1}} |y''(t)|\,,$$

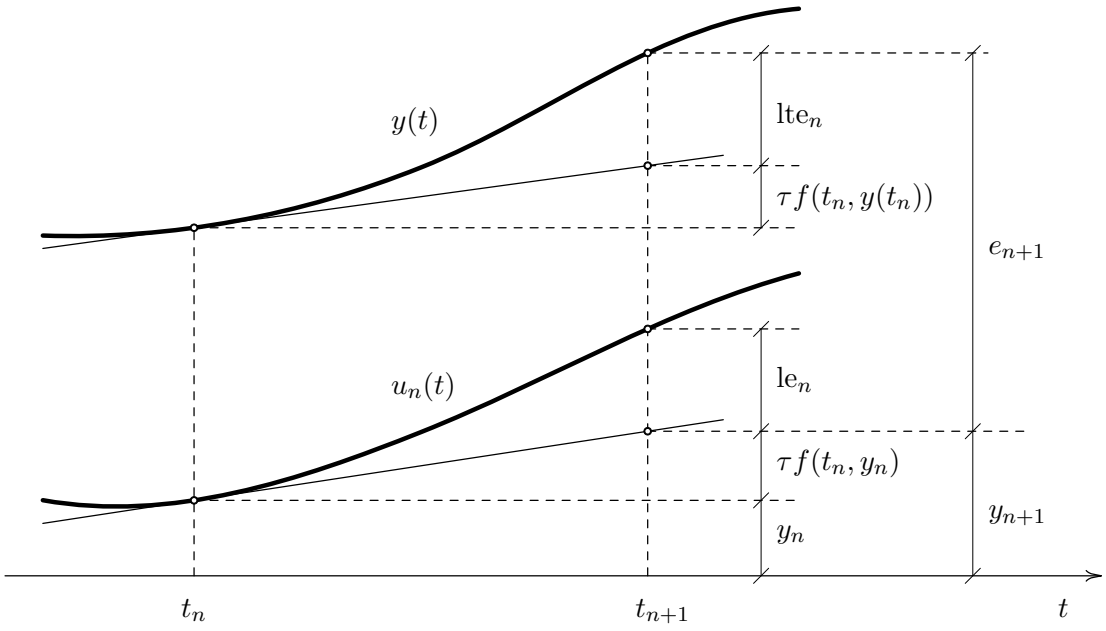
což lze stručně vyjádřit zápisem $\mathrm{lte}_n = O(\tau^2)$. Lokální diskretizační chyba při reálném výpočtu nevzniká, neboť obecně není splněn lokalizační předpoklad, tj. $y_n \neq y(t_n)$. Lokální diskretizační chyba se uplatní jen při analýze vlastností numerické metody, například konvergence $y_n \to y(t_n)$. Pro praktické účely, například pro řízení délky kroku, je nutné pracovat s tzv. *lokální chybou* (anglicky local error) definovanou předpisem

$$\mathrm{le}_n = u_n(t_{n+1}) - y_{n+1},$$

kde $u_n(t)$ je tzv. *lokální řešení* počátečního problému

$$u_n'(t) = f(t, u_n(t))\,, \qquad u_n(t_n) = y_n\,.$$

Lokální chyba le_n je tedy chyba, které se skutečně dopustíme při reálném výpočtu v kroku od t_n do t_{n+1}. Dá se ukázat, že pro výpočet s dostatečně malými délkami kroků je rozdíl mezi oběma lokálními chybami prakticky zanedbatelný.



Obr. 8.1. Diskretizační chyby

Hromaděním lokálních chyb vzniká *globální diskretizační chyba*

$$e_n = y(t_n) - y_n\,.$$

V případě rovnoměrného dělení lze dokázat, že

$$|e_n| = |y(t_n) - y_n| \le C\tau, \qquad n = 0, 1, \ldots, Q, \tag{8.9}$$

kde C je konstanta nezávislá na $\tau = (b-a)/Q$. Tuto skutečnost stručně vyjádříme tvrzením, že *globální diskretizační chyba EE metody je řádu* $O(\tau)$. Říkáme také, že *EE metoda je řádu 1.* Protože $e_n \to 0$ pro $Q \to \infty$, *numerické řešení* získané EE metodou *konverguje* k řešení přesnému. Říkáme také, že *rychlost (řád) konvergence EE metody je rovna 1.* Lokální chyby lte_n, le_n a globální chyba e_{n+1} jsou zakresleny v obrázku 8.1.

Tvrzení (8.9) lze snadno ověřit v případě, že $f(t, y) = f(t)$ nezávisí na y. Pak totiž

$$y(t_{k+1}) = y(t_k) + \tau f(t_k) + \tfrac{1}{2}\tau^2 f'(\xi_k)\,,$$
$$y_{k+1} = y_k + \tau f(t_k)\,.$$

Odečtením druhé rovnice od prvé dostaneme $e_{k+1} = e_k + \frac{1}{2}\tau^2 f'(\xi_k)$, a protože $e_0 = 0$, je

$$e_n = \tfrac{1}{2}\tau^2[f'(\xi_0) + f'(\xi_1) + \cdots + f'(\xi_{n-1})]\,.$$

Označíme-li $M = \max_{a\le\xi\le b}|f'(\xi)|$, pak

$$|e_n| \le \tfrac{1}{2}\tau^2 nM \le \tfrac{1}{2}[\tau Q]M\tau = \tfrac{1}{2}(b-a)M\tau, \quad \text{neboť } \tau Q = b-a.$$

Zaokrouhlovací chyby. Dopustíme-li se v kroku od t_n do t_{n+1} zaokrouhlovací chyby, jejíž velikost $|\Delta y_{n+1}| = |\tilde{y}_{n+1} - y_{n+1}|$ nepřesáhne ε, pak lze dokázat, že po Q krocích délky τ velikost zaokrouhlovací chyby nepřesáhne $K\varepsilon\tau^{-1}$, kde K je konstanta nezávislá na ε a τ. Pro celkovou chybu EE metody pak platí

$$\max_{0\le n\le Q}|y(t_n) - \tilde{y}_n| \le C\tau + \varepsilon K\tau^{-1}\,,$$

kde C, K jsou konstanty nezávislé na τ a ε. Protože konstanta ε je malá, vliv zaokrouhlování se projeví až pro extrémně velký počet kroků Q (tj. pro velmi malé τ). Tato situace při řešení běžných úloh nenastává a tudíž vliv zaokrouhlovacích chyb bývá nepodstatný.

Stabilita. Řekneme, že počáteční problém (8.1), (8.2) je stabilní vzhledem k počáteční podmínce, jestliže malá změna počáteční hodnoty η vyvolá malou změnu řešení. Elementárním příkladem takového problému je *testovací úloha*

$$y' = \lambda y\,, \quad y(0) = 1\,, \tag{8.10}$$

kde $\lambda = \alpha + i\beta$ je komplexní číslo se zápornou reálnou složkou, tj. $\mathrm{Re}(\lambda) = \alpha < 0$. Skutečně, jestliže

$$\begin{aligned} u'(t) &= \lambda u(t)\,, \qquad u(0) = 1 \qquad \Longrightarrow \quad u(t) = e^{\lambda t} \\ v'(t) &= \lambda v(t)\,, \qquad v(0) = 1+\delta \quad \Longrightarrow \quad v(t) = (1+\delta)e^{\lambda t}, \end{aligned}$$

pak

$$|u(t) - v(t)| \le |\delta| \cdot |e^{(\alpha+i\beta)t}| = |\delta|e^{\alpha t} \cdot |\cos\beta t + i\sin\beta t| = |\delta|e^{\alpha t} \le |\delta|\,.$$

Pro řešení $y(t) = e^{\lambda t}$ testovací úlohy (8.10) rovněž platí

$$|y(t)| = |e^{\lambda t}| = e^{\alpha t}|\cos\beta t + i\sin\beta t| \to 0 \quad \text{pro} \quad t\to\infty\,.$$

Je proto přirozené požadovat, aby na rovnoměrném dělení $t_n = n\tau$, $n = 0, 1, \ldots,$ numerické řešení y_n testovací úlohy (8.10) splňovalo analogickou relaci, tzv. *podmínku stability*

$$y_n \to 0 \quad \text{pro} \quad n\to\infty. \tag{8.11}$$

Aplikujeme-li EE metodu na testovací rovnici (8.10), dostaneme

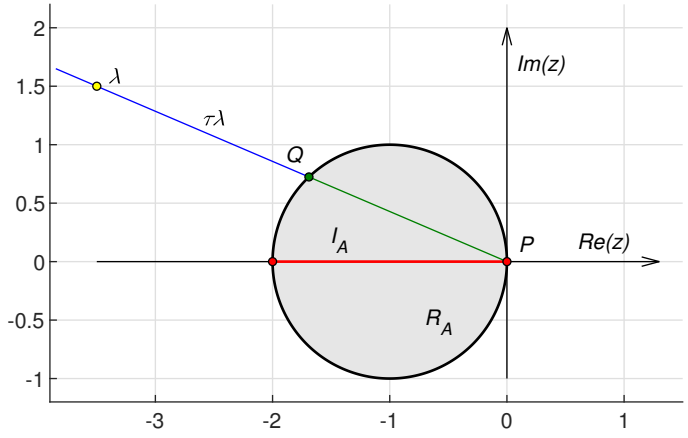
$$y_{n+1} = y_n + \tau\lambda y_n = (1+\tau\lambda)y_n = (1+\tau\lambda)^2 y_{n-1} = \cdots = (1+\tau\lambda)^{n+1}y_0\,.$$

Podmínka stability (8.11) bude splněna, právě když $|1+\tau\lambda| < 1$, neboli když $\tau\lambda$ leží v tzv. *oblasti absolutní stability* R_A:

$$\tau\lambda \in R_A = \{z \in \mathbb{C} \,|\, |z+1| < 1\}.$$

Oblast absolutní stability EE metody je tedy vnitřek jednotkového kruhu $|z+1| < 1$ komplexní roviny $\mathbb{C}$ se středem v bodě $[-1, 0]$. Průnik oblasti absolutní stability se zápornou částí reálné osy je *interval absolutní stability* I_A. Pro EE metodu $I_A = (-2, 0)$. Pro reálné $\lambda < 0$ podmínka stability (8.11) vyžaduje volit krok $\tau < 2/|\lambda|$. Z ilustračního obrázku 8.2 vyčteme, že pro zvolené λ musíme τ zvolit tak, aby $\tau\lambda$ byl vnitřní bod úsečky $\overline{PQ}$.

Tvar a velikost oblasti absolutní stability metody je spolu s řádem metody základní charakteristikou kvality numerické metody. EE metoda z tohoto pohledu příliš kvalitní není: je pouze řádu 1 a oblast její absolutní stability je malá. EE metoda se používá jen výjimečně.



Obr. 8.2. Oblast absolutní stability EE metody

Lineární stabilita. Testovací úloha (8.10) je dobrým modelem pro posouzení tzv. *lineární stability* obecné počáteční úlohy $y'(t) = f(t, y(t))$, $y(t_\alpha) = y_\alpha$. Když v Taylorově rozvoji okolo bodu (t_α, y_α),

$$f(t, y(t)) = f(t_\alpha, y_\alpha) + \frac{\partial f(t_\alpha, y_\alpha)}{\partial t}(t - t_\alpha) + \frac{\partial f(t_\alpha, y_\alpha)}{\partial y}(y(t) - y_\alpha) + O((t - t_\alpha)^2),$$

zanedbáme chybový člen, dostaneme aproximující lineární problém

$$y'(t) = \lambda_\alpha y(t) + g_\alpha(t), \quad y(t_\alpha) = y_\alpha, \tag{8.10'}$$

kde $\lambda_\alpha = \partial_y f(t_\alpha, y_\alpha)$, $g_\alpha(t) = \partial_t f(t_\alpha, y_\alpha)(t - t_\alpha) - \partial_y f(t_\alpha, y_\alpha) y_\alpha$. Jestliže

$$u' = \lambda_\alpha u + g_\alpha(t), \;\; u(t_\alpha) = y_\alpha, \qquad v' = \lambda_\alpha v + g_\alpha(t), \;\; v(t_\alpha) = y_\alpha + \delta_a,$$

pak $|u(t) - v(t)| = |\delta_\alpha| e^{\lambda_\alpha (t - t_\alpha)} \to 0$ pro $t \to \infty$, právě když $\mathrm{Re}(\lambda_\alpha) < 0$. V testovací úloze (8.10) si tedy pod λ můžeme představit hodnotu $\partial_y f(t_\alpha, y_\alpha)$ v nějakém bodě (t_α, y_α).

Implicitní Eulerova metoda. Vyjdeme opět z Taylorova rozvoje

$$y(t_n) = y(t_{n+1} - \tau) = y(t_{n+1}) - \tau y'(t_{n+1}) + \tfrac{1}{2}\tau^2 y''(\xi_n), \quad \xi_n \in (t_n, y_{n+1})\,. \tag{8.11}$$

Vypuštěním členu $\frac{1}{2}y''(\xi_n)$ a užitím rovnosti $y'(t_{n+1}) = f(t_{n+1}, y(t_{n+1}))$ obdržíme *implicitní Eulerovu metodu* (stručně IE metodu) jako předpis

$$y_{n+1} = y_n + \tau f(t_{n+1}, y_{n+1})\,. \tag{8.12}$$

V anglicky psané literatuře se IE metoda označuje jako *implicit Euler method* resp. *backward Euler method*. O implicitní metodě mluvíme proto, že neznámá y_{n+1} je rovnicí (8.12) určena implicitně. Pro dostatečně malé τ má rovnice (8.12) jediné řešení y_{n+1}, dokažte! Určit y_{n+1} znamená řešit obecně nelineární rovnici. To je ve srovnání s EE metodou problém navíc. Aby mělo použití IE metody nějaký smysl, musí mít IE metoda oproti EE metodě také nějakou přednost. Pokusme se ji najít.

Nejdříve prozkoumáme přesnost IE metody. Lokální diskretizační chyba IE metody je definována rovnicí

$$y(t_{n+1}) = y(t_n) + \tau f(t_{n+1}, y(t_{n+1})) + \mathrm{lte}_n,$$

kde $y(t)$ je řešení úlohy (8.1), (8.2). Z (8.11) plyne

$$\mathrm{lte}_n = -\tfrac{1}{2}\tau^2 y''(\xi_n) = O(\tau^2)\,.$$

IE metoda je tedy řádu 1 stejně jako EE metoda. Při podrobnějším zkoumání lze zjistit, že

$$\mathrm{lte}_n = \begin{cases} \frac{1}{2}y''(t_n)\tau^2 + O(\tau^3) & \text{pro EE metodu,} \\ -\frac{1}{2}y''(t_n)\tau^2 + O(\tau^3) & \text{pro IE metodu.} \end{cases}$$

Hlavní členy lokálních diskretizačních chyb (v případě Eulerových metod to jsou členy obsahující τ^2) jsou co do absolutní hodnoty stejné, liší se jen znaménkem. Můžeme proto oprávněně soudit, že obě metody jsou stejně přesné.

Podívejme se také na stabilitu IE metody. Pro testovací úlohu (8.10) je

$$y_{n+1} = y_n + \tau\lambda y_{n+1}\,, \quad \text{odtud} \quad y_{n+1} = \frac{1}{1-\tau\lambda}y_n = \cdots = \left(\frac{1}{1-\tau\lambda}\right)^{n+1} y_0$$

a tedy podmínka stability (8.11) platí, právě když $|1 - \tau\lambda| > 1$, neboli když $\tau\lambda$ leží v oblasti absolutní stability R_A:

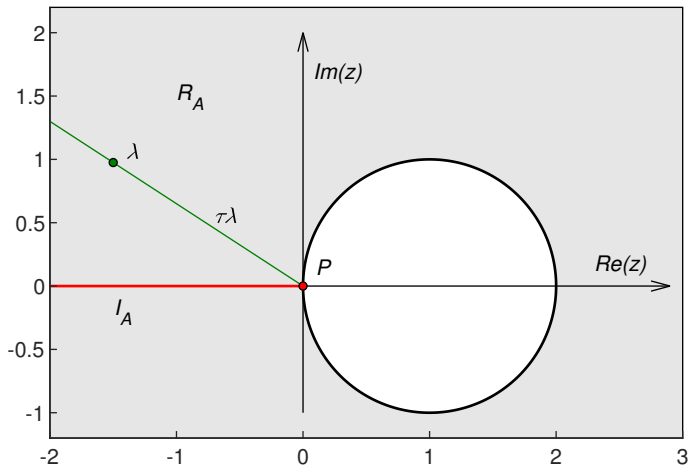
$$\tau\lambda \in R_A = \{z \in \mathbb{C} \,|\, |z - 1| > 1\}.$$

Oblast absolutní stability IE metody je tedy obrovská, je to celý vnějšek $|z - 1| > 1$ jednotkového kruhu komplexní roviny $\mathbb{C}$ se středem v bodě $[1, 0]$. Interval I_A absolutní stability IE metody je $I_A = (-\infty, 0)$. Podmínka stability (8.11) délku kroku IE metody zřejmě nijak neomezuje, viz obrázek 8.3.

Je to právě mimořádná stabilita, která je onou hledanou předností IE metody ve srovnání s EE metodou. Tento klad je však třeba vykoupit nutností řešit obecně nelineární rovnici. y_{n+1} získáme jako přibližné řešení rovnice $g(z) = 0$, kde

$$g(z) = z - y_n - \tau f(t_{n+1}, z)\,.$$

Protože dobrou počáteční aproximaci lze získat extrapolací z hodnot $y_n, y_{n-1}, \ldots$, dá se očekávat rychlá konvergence Newtonovy metody (pro řešení nelineárních rovnic). Praktická zkušenost potvrzuje, že tomu tak skutečně je.



Obr. 8.3. Oblast absolutní stability IE metody

Lichoběžníkovou metodu dostaneme jako aritmetický průměr EE metody a IE metody:

$$y_{n+1} = y_n + \tfrac{1}{2}\tau[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]\,. \tag{8.13}$$

Pro lokální diskretizační chybu lte_n, definovanou rovnicí

$$y(t_{n+1}) = y(t_n) + \tfrac{1}{2}\tau[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))] + \mathrm{lte}_n,$$

užitím Taylorovy věty odvodíme

$$\mathrm{lte}_n = -\tfrac{1}{12}\tau^3 y'''(t_n) + O(\tau^4)\,.$$

Lichoběžníková metoda (stručně TR metoda podle anglického *trapezoidal rule*) je tedy implicitní metoda řádu 2. Stabilitu TR metody zjistíme řešením testovací úlohy (8.10):

$$y_{n+1} = y_n + \tfrac{1}{2}\tau\lambda[y_n + y_{n+1}], \quad \text{odtud} \quad y_{n+1} = \frac{2+\tau\lambda}{2-\tau\lambda} y_n = \cdots = \left[\frac{2+\tau\lambda}{2-\tau\lambda}\right]^{n+1} y_0\,.$$

Není těžké ověřit, že podmínka stability (8.11) platí, právě když

$$\tau\lambda \in R_A = \{z \in \mathbb{C} \,|\, \mathrm{Re}(z) < 0\}.$$

Oblast absolutní stability TR metody tedy obsahuje celou zápornou polorovinu komplexní roviny $\mathbb{C}$, interval absolutní stability $I_A = (-\infty, 0)$. TR metoda (s podporou IE metody) je v Matlabu implementována jako program `ode23t`.

8.3. Explicitní Rungovy-Kuttovy metody

Obecný tvar *s-stupňové explicitní Rungovy-Kuttovy metody* je

$$y_{n+1} = y_n + \tau(b_1 k_1 + b_2 k_2 + \cdots + b_s k_s)\,, \tag{8.14}$$

kde koeficienty k_i, $i = 1, 2, \ldots, s$, jsou určeny předpisem

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \tau c_2, y_n + \tau a_{21} k_1), \\ k_3 &= f(t_n + \tau c_3, y_n + \tau(a_{31} k_1 + a_{32} k_2)), \\ &\vdots \\ k_s &= f(t_n + \tau c_s, y_n + \tau(a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{s,s-1} k_{s-1})), \end{aligned} \tag{8.15}$$

a kde b_i, c_i, a_{ij} jsou konstanty definující konkrétní metodu. Rungova-Kuttova metoda (8.14), (8.15) je explicitní: nejdříve spočteme k_1, pak k_2 pomocí k_1, pak k_3 pomocí k_1, k_2 a tak dále, až nakonec spočteme k_s pomocí $k_1, k_2, \ldots, k_{s-1}$. Vypočtené koeficienty k_i, $i = 1, 2, \ldots, s$, dosadíme do (8.14) a dostaneme y_{n+1}.

V dalším budeme hovořit jen o Rungových-Kuttových metodách (stručně RK metodách), tj. slůvko „explicitní“ vynecháme. Je však třeba připomenout, že existují také implicitní Rungovy-Kuttovy metody, těmi se však zabývat nebudeme.

RK metody jsou zřejmě jednokrokové: k výpočtu y_{n+1} potřebujeme znát jen y_n, předchozí hodnoty $y_{n-1}, y_{n-2}, \ldots$ v kroku od t_n do t_{n+1} nepoužijeme.

Koeficient k_i je směrnicí lokálního řešení procházejícího bodem $[t_i^*, y_i^*]$, kde

$$[t_1^*, y_1^*] = [t_n, y_n],\ \ t_i^* = t_n + \tau c_i,\ \ y_i^* = y_n + \tau(a_{i1} k_1 + a_{i2} k_2 + \cdots + a_{i,i-1} k_{i-1}),\ \ i = 2, 3, \ldots, s.$$

Do bodu $[t_{n+1}, y_{n+1}]$ se tedy dostaneme z bodu $[t_n, y_n]$ tak, že se posuneme po přímce, jejíž směrnice $k^* = b_1 k_1 + b_2 k_2 + \cdots + b_s k_s$ je váženým průměrem směrnic $k_1, k_2, \ldots, k_s$ (podle (8.16) pro všechny prakticky používané metody řádu alespoň jedna platí $\sum_{i=1}^{s} b_i = 1$).

Abychom dostali konkrétní metodu, musíme určit stupeň s a dále konstanty c_i, b_i, a_{ij}. Konstanty RK metod je zvykem zapisovat do tabulky známé jako *Butcherova tabulka*:

c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$				
c_s	a_{s1}	a_{s2}	$\ldots$	$a_{s,s-1}$	
	b_1	b_2	$\ldots$	b_{s-1}	b_s

Jedním z kritérií při volbě konstant RK metody je dosažení dostatečné přesnosti. Tu

měříme pomocí lokální diskretizační chyby

$$\mathrm{lte}_n = y(t_{n+1}) - \left[y(t_n) + \tau \sum_{i=1}^{s} b_i k_i(t_n, y(t_n)) \right],$$

kde $k_1(t_n, y(t_n)) = f(t_n, y(t_n))$,

$$k_i(t_n, y(t_n)) = f(t_n + \tau c_i, y(t_n) + \tau \sum_{j=1}^{i-1} a_{ij} k_j(t_n, y(t_n))), \quad i = 2, 3, \ldots, s.$$

Lokální diskretizační chyba je chyba, které se dopustíme v jednom kroku za *lokalizačního předpokladu* $y_n = y(t_n)$. RK metoda je řádu p, pokud lokální diskretizační chyba je řádu $O(\tau^{p+1})$. Pro $p = 1, 2, 3$ lze odvodit následující tzv. *podmínky řádu*:

$$\begin{aligned}
&\text{řád 1:} \quad \sum_{i=1}^{s} b_i = 1\,, \\
&\text{řád 2:} \quad \sum_{i=1}^{s} b_i = 1\,, \quad \sum_{i=2}^{s} b_i c_i = \tfrac{1}{2}\,, \\
&\text{řád 3:} \quad \sum_{i=1}^{s} b_i = 1\,, \quad \sum_{i=2}^{s} b_i c_i = \tfrac{1}{2}\,, \quad \sum_{i=2}^{s} b_i c_i^2 = \tfrac{1}{3}\,, \quad \sum_{i=2}^{s} \sum_{j=2}^{i-1} b_i a_{ij} c_j = \tfrac{1}{6}\,.
\end{aligned} \tag{8.16}$$

Odvození podmínek řádu pro $p = 1, 2, 3, 4, 5$ lze najít třeba v [50]. Protože všechny prakticky používané metody splňují podmínku

$$c_i = a_{i1} + a_{i2} + \cdots + a_{i,i-1}\,, \quad i = 2, 3, \ldots, s\,, \tag{8.17}$$

budeme i my předpokládat, že podmínka (8.17) platí.

RK metoda řádu $p \geq 1$ má globální diskretizační chybu řádu $O(\tau^p)$. Předpokladem pro platnost tohoto tvrzení je dostatečná hladkost pravé strany f, konkrétně je třeba, aby funkce $f(t, y)$ měla spojité derivace až do řádu p včetně. Pokud f má spojité derivace jen do řádu $s \leq p$, pak lze pro globální chybu dokázat pouze řád $O(\tau^s)$, viz [50].

Označme $p(s)$ maximální dosažitelný řád s-stupňové RK metody. Platí

$$\begin{aligned}
&p(s) = s \quad \text{pro } s = 1, 2, 3, 4, && p(8) = 6, \\
&p(5) = 4, && p(9) = 7 \\
&p(6) = 5, && p(s) \leq s - 2 \quad \text{pro} \quad s = 10, 11, \ldots \\
&p(7) = 6, &&
\end{aligned}$$

Vidíme, že s-stupňové RK metody řádu s existují jen pro $1 \leq s \leq 4$. Například metoda řádu 5 je nejméně 6-ti stupňová. Uveďme si několik nejznámějších metod.

Metoda řádu 1. Pro $s = p = 1$ existuje jediná explicitní metoda a tou je nám již známá EE metoda $y_{n+1} = y_n + \tau f(t_n, y_n)$.

Metody řádu 2. Pro $s = p = 2$ má explicitní RK metoda Butcherovu tabulku

c_2	a_{21}	
	b_1	b_2

Podmínky (8.16) pro metodu řádu 2 stanoví

$$b_1 + b_2 = 1, \quad b_2 c_2 = \tfrac{1}{2}\,,$$

a protože ve shodě s (8.17) předpokládáme $a_{21} = c_2$, dostáváme tabulku

a	a	
	$1-b$	b

kde $ab = \frac{1}{2}$. Parametry a, b jsou tedy svázány jednou podmínkou, takže zvolíme-li $a \neq 0$, je $b = 1/(2a)$.

Pro $a = \frac{1}{2}$ je $b = 1$ a dostáváme metodu

$$y_{n+1} = y_n + \tau k_2, \quad \text{kde} \quad k_2 = f(t_n + \tfrac{1}{2}\tau, y_n + \tfrac{1}{2}\tau k_1),\ k_1 = f(t_n, y_n), \tag{8.18}$$

známou pod názvem *modifikovaná Eulerova metoda*. Budeme ji značit EM1 jako *první modifikace Eulerovy metody*. V anglicky psané literatuře je metoda (8.18) známa jako *midpoint Euler formula*.

Pro $a = 1$ je $b = \frac{1}{2}$ a dostáváme metodu

$$y_{n+1} = y_n + \tfrac{1}{2}\tau(k_1 + k_2), \quad \text{kde} \quad k_1 = f(t_n, y_n),\ k_2 = f(t_n + \tau, y_n + \tau k_1). \tag{8.19}$$

Budeme ji značit EM2 jako *druhou modifikaci Eulerovy metody*. Metoda (8.19) se také často uvádí pod názvem *Heunova metoda*.

Pro $a = \frac{2}{3}$ je $b = \frac{3}{4}$ a dostáváme metodu

$$y_{n+1} = y_n + \tfrac{1}{4}\tau(k_1 + 3k_2), \quad \text{kde} \quad k_1 = f(t_n, y_n),\ k_2 = f(t_n + \tfrac{2}{3}\tau, y_n + \tfrac{2}{3}\tau k_1),$$

známou jako *Ralstonova metoda řádu* 2 (stručně R2 metoda).

Metody řádu 3. Pro $s = p = 3$ dostáváme Butcherovu tabulku

c_2	c_2		
c_3	$c_3 - a_{32}$	a_{32}	
	b_1	b_2	b_3

a 4 podmínky pro metodu řádu 3:

$$b_1 + b_2 + b_3 = 1\,, \quad b_2 c_2 + b_3 c_3 = \tfrac{1}{2}\,,$$
$$b_2 c_2^2 + b_3 c_3^2 = \tfrac{1}{3}\,, \quad b_3 a_{32} c_2 = \tfrac{1}{6}\,.$$

Když zvolíme dva parametry $0 < c_2 < c_3$, jsou tím všechny koeficienty metody jednoznačně určeny. Volba $c_2 = \frac{1}{2}$, $c_3 = \frac{3}{4}$ vede na *Ralstonovu metodu řádu* 3 (stručně R3 metodu):

$\frac{1}{2}$	$\frac{1}{2}$		
$\frac{3}{4}$	0	$\frac{3}{4}$	
	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$

$$y_{n+1} = y_n + \tfrac{1}{9}\tau\,(2k_1 + 3k_2 + 4k_3)\,,$$
$$k_1 = f(t_n, y_n)\,, \quad k_2 = f(t_n + \tfrac{1}{2}\tau, y_n + \tfrac{1}{2}\tau k_1),$$
$$k_3 = f(t_n + \tfrac{3}{4}\tau, y_n + \tfrac{3}{4}\tau k_2).$$

Ralstonova metoda řádu 3 je základem *Runge-Kutta-Bogacki-Shampine* metody, viz [50], která je implementována do Matlabu jako funkce `ode23` a jejíž popis uvedeme v této kapitole později.

Metody řádu 4. Pro $s = p = 4$ je nejznámější *klasická Rungova-Kuttova metoda*

$$\begin{array}{c|cccc} \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

$$\begin{aligned} &y_{n+1} = y_n + \tfrac{1}{6}\tau\,(k_1 + 2k_2 + 2k_3 + k_4)\,, \\ &k_1 = f(t_n, y_n)\,, \qquad\qquad k_2 = f(t_n + \tfrac{1}{2}\tau, y_n + \tfrac{1}{2}\tau k_1), \\ &k_3 = f(t_n + \tfrac{1}{2}\tau, y_n + \tfrac{1}{2}\tau k_2), \quad k_4 = f(t_n + \tau, y_n + \tau k_3). \end{aligned}$$

Klasická Rungova-Kuttova metoda (stručně cRK4) byla velmi populární v době, kdy se ještě nepoužívaly samočinné počítače a kdy proto velmi významným kritériem byla jednoduchost metody. Toto hledisko však v současné době ztratilo na významu a proto se používají jiné metody. Kvalitní dvojice metod řádu 4 a 5 jsou součástí metod *Runge-Kutta-Fehlberg* nebo *Runge-Kutta-Dormand-Prince*, viz např. [26]. Posledně jmenovaná dvojice metod je použita v matlabovské funkci `ode45`, popis uvedeme v této kapitole později.

Řízení délky kroku. V profesionálních programech uživatel zadá toleranci ε a program délku kroku vybírá tak, aby velikost odhadu est_n lokální chyby le_n nabývala pořád přibližně stejné hodnoty ε. Krok od y_n do y_{n+1} je úspěšný, když

$$|\mathrm{est}_n| \le \varepsilon\,. \tag{8.20}$$

Je-li podmínka (8.20) splněna, krok je úspěšný a pokračujeme výpočtem y_{n+2}. Pokud podmínka (8.20) splněna není, krok je neúspěšný a výpočet y_{n+1} opakujeme. Novou délku kroku τ^* určíme v případě úspěchu i neúspěchu stejným postupem, který si teď vysvětlíme.

Předpokládejme, že y_{n+1} počítáme metodou řádu p, takže

$$\mathrm{le}_n \doteq C\tau^{p+1} \doteq \mathrm{est}_n \quad \Longrightarrow \quad C \doteq \mathrm{est}_n/\tau^{p+1}\,.$$

Novou délku kroku τ^* zvolíme tak, aby velikost $|\mathrm{le}_n^*|$ lokální chyby $\mathrm{le}_n^* \doteq C(\tau^*)^{p+1}$ byla přibližně rovna zadané toleranci ε, tj.

$$|\mathrm{le}_n^*| \doteq |C|(\tau^*)^{p+1} \doteq |\mathrm{est}_n/\tau^{p+1}|(\tau^*)^{p+1} \doteq \varepsilon \quad \Longrightarrow \quad \tau^* \doteq \tau\,(\varepsilon/|\mathrm{est}_n|)^{1/(p+1)}\,.$$

Nová délka kroku τ^* se ještě redukuje pomocí parametru $\theta < 1$, takže

$$\tau^* = \theta\tau\,(\varepsilon/|\mathrm{est}_n|)^{1/(p+1)}\,. \tag{8.21}$$

V matlabovských programech `ode23` a `ode45` se bere $\theta = 0{,}8$. Současně se ještě uplatňují následující zásady.

1) Tolerance ε se uvažuje ve tvaru

   $$\varepsilon = \max\{\varepsilon_r \max\{|y_n|, |y_{n+1}|\}, \varepsilon_a\},$$

   kde ε_r je relativní tolerance a ε_a je tolerance absolutní. Matlabem přednastavené hodnoty jsou $\varepsilon_r = 10^{-3}$ a $\varepsilon_a = 10^{-6}$.

2) Označme τ_{min} resp. τ_{max} minimální resp. maximální povolenou délku kroku. Jestliže $\tau^* < \tau_{min}$, výpočet končí konstatováním, že danou diferenciální rovnici program neumí s požadovanou přesností vyřešit. Přitom $\tau_{min} = 16\varepsilon_m(t_n)$, kde $\varepsilon_m(t_n)$ je tzv. relativní přesnost aritmetiky pohyblivé řádové čárky, viz funkce `eps` v Matlabu. Jestliže $\tau^* > \tau_{max}$, položí se $\tau = \tau_{max}$, kde je $\tau_{max} = 0{,}1(b-a)$.

3) V případě neúspěšného kroku navrženou délku τ^* redukujeme:

$$\tau^* = \max(\tau^*, q_{min}\tau)\,,$$

kde $q_{min} = 0{,}5$ resp. 0,1 v `ode23` resp. `ode45` při prvním neúspěchu v rámci jednoho kroku a $\tau^* = 0{,}5\tau$ při opakovaném neúspěchu v témže kroku.

4) V případě úspěšného kroku délku kroku τ^* redukujeme předpisem

$$\tau^* = \min(\tau^*, q_{max}\tau),$$

kde $q_{max} = 5$.

5) V kroku bezprostředně následujícím po neúspěšném kroku se délka kroku nesmí zvětšit.

6) Počáteční délka kroku

$$\tau = \theta\varepsilon_r^{1/(p+1)} \max(|\eta|, \varepsilon_a/\varepsilon_r)/|f(a,\eta)|\,, \tag{8.22}$$

přičemž pro $\tau < \tau_{min}$ změníme τ na τ_{min} a pro $\tau > \tau_{max}$ změníme τ na τ_{max}.

7) Při programování je třeba postupovat opatrně. Příkazy programu je nutné uspořádat tak, aby nemohlo dojít k dělení nulou, viz (8.21) a (8.22).

Podrobnější informace týkající se řízení délky kroku lze najít v [50].

Odhad lokální chyby. Základní myšlenka je jednoduchá. Použijí se dvě metody, z nichž jedna je řádu p a druhá řádu $p+1$. Z výchozí hodnoty y_n spočteme y_{n+1}^{**} přesnější metodou a y_{n+1}^{*} méně přesnou metodou. Použitelný odhad lokální chyby je

$$\mathrm{est}_n = y_{n+1}^{**} - y_{n+1}^{*}.$$

Obě metody používají tutéž množinu koeficientů $\{k_j\}_{j=1}^s$. V tom případě je totiž získání odhadu „laciné“. Dvojice Rungových-Kuttových metod se popisují pomocí rozšířené Butcherovy tabulky,

c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$				
c_s	a_{s1}	a_{s2}	$\dots$	$a_{s,s-1}$	
	b_1^*	b_2^*	$\dots$	b_{s-1}^*	b_s^*
	b_1^{**}	b_2^{**}	$\dots$	b_{s-1}^{**}	b_s^{**}
	E_1	E_2	$\dots$	E_{s-1}	E_s

přičemž

$y_{n+1}^* = y_n + \tau\sum_{j=1}^s b_j^* k_j$ je metoda řádu p,

$y_{n+1}^{**} = y_n + \tau\sum_{j=1}^s b_j^{**} k_j$ je metoda řádu $p+1$,

$\mathrm{est}_n = \tau\sum_{j=1}^s E_j k_j$ je odhad lokální chyby,

takže $E_j = b_j^{**} - b_j^*$, $j = 1, 2, \dots, s$.

Pokud ve výpočtu pokračujeme přesnější metodou, tj. když $y_{n+1} = y^{**}_{n+1} = y^{*}_{n+1} + \mathrm{est}_n$, říkáme, že jsme použili dvojici metod *s lokální extrapolací*. Tento postup se v současných programech upřednostňuje. Druhou možností je pokračovat méně přesnou metodou, tj. položit $y_{n+1} = y^{*}_{n+1}$. V tom případě se přesnější metoda použije jen pro získání odhadu chyby a říkáme, že jsme dvojici metod použili *bez lokální extrapolace*. Obě níže uvedené metody BS32 a DP54 se používají jako metody s lokální extrapolací. Příkladem metody, která se obvykle používá bez lokální extrapolace, je Runge-Kutta-Fehlbergova metoda řádu 4, označovaná stručně jako RKF45, viz např. [26]. Na vysvětlenou k použitým zkratkám uveďme, že první číslo značí řád metody a druhé číslo řád pomocné metody použité pro odhad lokální chyby.

Bogacki–Shampine metoda, stručně BS32 metoda, je implementována v Matlabu jako funkce `ode23`. Rozšířená Butcherova tabulka BS32 metody je

$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$
	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	0
	$-\frac{5}{72}$	$\frac{1}{12}$	$\frac{1}{9}$	$-\frac{1}{8}$

Přesnější z obou metod páru je Ralstonova R3 metoda

$$y^{**}_{n+1} = y_n + \tau\left[\tfrac{2}{9}k_1 + \tfrac{1}{3}k_2 + \tfrac{4}{9}k_3\right] = y_{n+1}$$

řádu 3. Pomocná metoda

$$y^{*}_{n+1} = y_n + \tau\left[\tfrac{7}{24}k_1 + \tfrac{1}{4}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{8}k_4\right]$$

řádu 2 používá kromě koeficientů k_1, k_2 a k_3 navíc ještě koeficient $k_4 = f(t_{n+1}, y_{n+1})$. V každém kroku se tedy počítají jen 3 nové hodnoty funkce f, neboť koeficient k_1 ve stávajícím kroku je roven koeficientu k_4 z kroku předchozího, takže nově se počítají jen koeficienty k_2, k_3 a k_4. Zařazení koeficientu k_4 do metody řádu 2 nás tedy téměř nic nestojí, umožní však zlepšit vlastnosti této metody a tím i celého páru. Metoda, ve které $k_s = f(t_{n+1}, y_{n+1})$, bývá označována jako FSAL podle anglického *First Same As Last*. Zdůrazněme, že BS32 metoda se používá jako metoda s lokální extrapolací, tj. $y_{n+1} = y^{**}_{n+1}$.

Hodnoty přibližného řešení pro $t \in \langle t_n, t_{n+1}\rangle$ spočteme dostatečně přesně pomocí kubického Hermitova polynomu $H_3(t)$ určeného podmínkami

$$\begin{aligned} H_3(t_n) &= y_n, & H_3'(t_n) &= k_1\,, \\ H_3(t_{n+1}) &= y_{n+1}, & H_3'(t_{n+1}) &= k_4\,. \end{aligned}$$

Metoda BS32 je tedy skvělá: je řádu 3, v každém kroku se pravá strana f počítá jen 3-krát, a to stačí jak na řízení délky kroku tak na výpočet řešení mezi uzly t_n a t_{n+1}.

Dormand-Prince metoda, stručně DP54 metoda, je definována rozšířenou Butcherovou tabulkou 8.1. Metoda DP54 je typu FSAL, neboť $k_7 = f(t_{n+1}, y_{n+1})$. Proto se v každém úspěšném kroku metody počítají jen koeficienty $k_2, \ldots, k_7$, koeficient k_1 byl už vypočten jako koeficient k_7 v předchozím kroku. Zdůrazněme, že DP54 metoda se používá jako metoda s lokální extrapolací, tj. $y_{n+1} = y^{**}_{n+1}$.

$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19\,372}{6\,561}$	$-\frac{25\,360}{2\,187}$	$\frac{64\,448}{6\,561}$	$-\frac{212}{729}$			
1	$\frac{9\,017}{3\,168}$	$-\frac{355}{33}$	$\frac{46\,732}{5\,247}$	$\frac{49}{176}$	$-\frac{5\,103}{18\,656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1\,113}$	$\frac{125}{192}$	$-\frac{2\,187}{6\,784}$	$\frac{11}{84}$	
	$\frac{5\,179}{57\,600}$	0	$\frac{7\,571}{16\,695}$	$\frac{393}{640}$	$-\frac{92\,097}{339\,200}$	$\frac{187}{2\,100}$	$\frac{1}{40}$
	$\frac{35}{384}$	0	$\frac{500}{1\,113}$	$\frac{125}{192}$	$-\frac{2\,187}{6\,784}$	$\frac{11}{84}$	0
	$\frac{71}{57\,600}$	0	$-\frac{71}{16\,695}$	$\frac{71}{1\,920}$	$-\frac{17\,253}{339\,200}$	$\frac{22}{525}$	$-\frac{1}{40}$

Tab 8.1. Dormand-Prince (5,4) metoda

Hodnoty přibližného řešení pro $t \in \langle t_n, t_{n+1}\rangle$ spočteme dostatečně přesně pomocí interpolačního polynomu $H_4(t) = y_n + \tau \mathbf{k}\mathbf{B}\mathbf{q}$, kde $\mathbf{k} = (k_1, k_2, k_3, k_4, k_5, k_6, k_7)$,

$$\mathbf{B} = \begin{bmatrix} 1 & -\frac{183}{64} & \frac{37}{12} & -\frac{145}{128} \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1\,500}{371} & -\frac{1\,000}{159} & \frac{1\,000}{371} \\ 0 & -\frac{125}{32} & \frac{125}{12} & -\frac{375}{64} \\ 0 & \frac{9\,477}{3\,392} & -\frac{729}{106} & \frac{25\,515}{6\,784} \\ 0 & -\frac{11}{7} & \frac{11}{3} & -\frac{55}{28} \\ 0 & \frac{3}{2} & -4 & \frac{5}{2} \end{bmatrix},$$

$\mathbf{q} = (q, q^2, q^3, q^4)^T$, přičemž $q = (t - t_n)/\tau$.

Metoda DP54 je rovněž vynikající: je řádu 5, v každém kroku se pravá strana počítá jen 6-krát, a to stačí jak pro řízení délky kroku tak pro výpočet řešení v intervalu $\langle t_n, t_{n+1}\rangle$.

Stabilita. Řešíme-li testovací rovnici (8.10) RK metodou na rovnoměrném dělení s krokem τ, dostaneme

$$y_{n+1} = P_s(\tau\lambda) y_n \,,$$

kde P_s je polynom stupně s určený pomocí konstant b_i, a_{ij} definujících RK metodu. Podmínka stability (8.11) tedy platí, právě když $|P_s(\tau\lambda)| < 1$, neboli když $\tau\lambda$ leží v oblasti absolutní stability R_A:

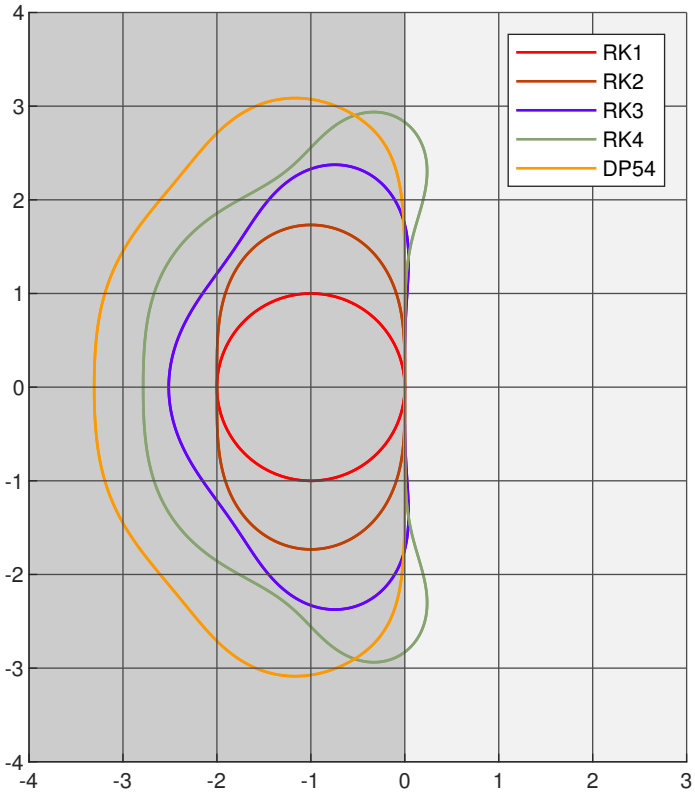
$$\tau\lambda \in R_A = \{z \in \mathbb{C} \,|\, |P_s(z)| < 1\}.$$

Explicitní RK metody mají omezenou oblast absolutní stability, neboť $|P_s(z)| \to \infty$ pro $|z| \to \infty$. Dá se ukázat, že pro $p = s \le 4$ je $P_s(z) = \sum_{i=0}^{s} z^i/i!$. Proto každá explicitní s-stupňová RK metoda řádu $p = s \le 4$ (stručně RKs metoda) má stejnou oblast absolutní

stability. Oblasti absolutní stability RKs metod pro $s = 1, 2, 3, 4$ a metody DP54 jsou uvedeny v obrázku 8.4. Intervaly absolutní stability těchto metod jsou $I_A = (\alpha, 0)$, kde

metoda	RK1	RK2	RK3	RK4	DP54
α	-2	-2	$-2{,}51$	$-2{,}79$	$-3{,}31$

Zdůrazněme, že z pohledu stability je BS32 metoda ekvivalentní s RK3 metodou, obě metody tedy maií steinou oblast a steinv interval absolutní stabilitv.



Obr. 8.4. Oblast absolutní stability RK metod

8.4. Lineární mnohokrokové metody

V této kapitole se budeme zabývat metodami, které počítají přibližné řešení y_{n+1} v uzlu t_{n+1} pomocí dříve spočtených aproximací $y_n, y_{n-1}, y_{n-2}, \ldots$ a odpovídajících hodnot $f(t_n, y_n)$, $f(t_{n-1}, y_{n-1}), f(t_{n-2}, y_{n-2}), \ldots$ pravé strany diferenciální rovnice. Tyto hodnoty jsou znovu použity tak, abychom získali y_{n+1} s vysokou přesností pomocí jen několika málo nových vyhodnocení funkce $f(t, y)$. Nejznámějšími metodami tohoto typu jsou *Adamsovy metody* a *metody zpětného derivování*. Obě skupiny metod patří do obecné třídy metod známých jako *lineární mnohokrokové metody*, stručně LMM.

8.4.1. Obecná lineární mnohokroková metoda

LMM je předpis

$$\alpha_0 y_{n+1} + \alpha_1 y_n + \cdots + \alpha_k y_{n+1-k} = \tau[\beta_0 f(t_{n+1}, y_{n+1}) + \beta_1 f_n + \cdots + \beta_k f_{n+1-k}]\,, \quad (8.23)$$

ze kterého počítáme y_{n+1}. Přitom α_j a β_j jsou číselné koeficienty, které formuli jednoznačně určují, a f_j je zkrácený zápis pro $f(t_j, y_j)$. V dalším budeme předpokládat, že platí *normalizační podmínka* $\alpha_0 = 1$. Jestliže alespoň jeden z koeficientů α_k nebo β_k je různý od nuly, *metoda* je *k-kroková*.

Pro $\beta_0 \neq 0$ je nová hodnota y_{n+1} určena implicitně, hovoříme proto o *implicitní metodě*, pro $\beta_0 = 0$ máme *metodu explicitní*. Abychom v implicitní metodě určili y_{n+1}, musíme vyřešit obecně nelineární rovnici.

LMM lze použít, jen když jsou zadány *startovací hodnoty* $y_0, y_1, \ldots, y_{k-1}$. $y_0 = \eta$ určíme z počáteční podmínky, zbývající startovací hodnoty je však třeba získat jinou vhodnou metodou, y_r metodou nejvýše r-krokovou.

Lokální diskretizační chyba je chyba, která vznikne, když do formule (8.23) dosadíme místo přibližného řešení y_{n+1-j} přesné řešení $y(t_{n+1-j})$, tedy

$$\mathrm{lte}_n = \sum_{j=0}^{k} \alpha_j y(t_{n+1-j}) - \tau \sum_{j=0}^{k} \beta_j f(t_{n+1-j}, y(t_{n+1-j})).$$

Jestliže

$$\mathrm{lte}_n = C_{p+1}\tau^{p+1} y^{(p+1)}(t_n) + O(\tau^{p+2}),$$

řekneme, že metoda je řádu p. Člen $C_{p+1}\tau^{p+1}y^{(p+1)}(t_n)$ se nazývá *hlavní člen lokální diskretizační chyby*, konstanta C_{p+1} je tzv. *chybová konstanta*. LMM je tím přesnější, čím je vyššího řádu. Z několika metod téhož řádu je pak nejpřesnější ta metoda, pro kterou je velikost chybové konstanty $|C_{p+1}|$ nejmenší.

D-stabilita. Řekneme, že LMM je *stabilní ve smyslu Dahlquista* (stručně *D-stabilní*), jestliže všechny kořeny *prvního charakteristického polynomu*

$$\varrho(\xi) = \xi^k + \alpha_1 \xi^{k-1} + \cdots + \alpha_{k-1}\xi + \alpha_k$$

leží uvnitř jednotkového kruhu $|z| \leq 1$ komplexní roviny $\mathbb{C}$ a pokud některý kořen leží hranici $|z| = 1$, pak je jednoduchý.

Význam D-stability lze vysvětlit na rovnici $y' = 0$. Její řešení pomocí LMM vede na předpis $\sum_{j=0}^{k} \alpha_j y_{n+1-j} = 0$. Zvolíme-li startovací hodnoty $y_j = \varepsilon r^j$, $j = 0, 1, \ldots, k-1$, kde $\varrho(r) = 0$ a ε je libovolné číslo, pak $y_n = \varepsilon r^n$ pro každé n. Skutečně,

$$\sum_{j=0}^{k} \alpha_j \varepsilon r^{n+1-j} = \varepsilon r^{n+1-k} \sum_{j=0}^{k} \alpha_j r^{k-j} = \varepsilon r^{n+1-k}\varrho(r) = 0.$$

Pro $|r| > 1$ a $\varepsilon \neq 0$ je $\lim_{n\to\infty} |y_n| = \infty$, což je nepřijatelné: pro $\varepsilon = 0$ je $y_n = 0$ přesné řešení rovnice $y' = 0$, avšak pro $\varepsilon \neq 0$, $|\varepsilon| \ll 1$, tj. již pro nepatrnou poruchu startovacích

hodnot y_j, $j = 0, 1, \ldots, k-1$, dostaneme řešení zcela nevyhovující. Dá se ukázat, že vyloučit musíme také případ, kdy $|r| = 1$ je kořen $\varrho(\xi)$ násobnosti větší než 1.

Konvergence. Uvažujme D-stabilní LMM řádu $p \geq 1$. Jestliže startovací hodnoty zadáme s chybou řádu $O(\tau^p)$, pak globální diskretizační chyba je rovněž řádu $O(\tau^p)$.

Precizní formulaci a důkaz této věty lze najít např. v [59], [26]. Předpokladem její platnosti je dostatečná hladkost pravé strany f, konkrétně je třeba, aby funkce $f(t, y)$ měla spojité derivace až do řádu p včetně. Pokud f má spojité derivace jen do řádu $s \leq p$, pak lze pro globální chybu dokázat pouze řád $O(\tau^s)$.

Absolutní stabilita. Řešíme-li testovací úlohu (8.10) LMM na rovnoměrném dělení s krokem τ, dostaneme

$$\sum_{j=0}^{k} (\alpha_j - \tau\lambda\beta_j) y_{n+1-j} = 0\,. \tag{8.24}$$

Řešení hledejme ve tvaru $y_n = r^n$. Po dosazení do diferenční rovnice (8.24) obdržíme

$$\sum_{j=0}^{k} (\alpha_j - \tau\lambda\beta_j) r^{n+1-j} = r^{n+1-k} \sum_{j=0}^{k} (\alpha_j - \tau\lambda\beta_j) r^{k-j} = r^{n+1-k} \pi(r, \tau\lambda) = 0\,,$$

$$\text{kde} \quad \pi(\xi, z) = \sum_{j=0}^{k} (\alpha_j - z\beta_j) \xi^{k-j}$$

je tzv. *polynom stability* LMM. Jestliže $\pi(r, \tau\lambda) = 0$, pak $y_n = r^n$ je řešením diferenční rovnice (8.24) a podmínka stability $y_n \to 0$ pro $n \to \infty$ platí, právě když $|r| < 1$.

Oblast R_A absolutní stability LMM metody definujeme jako množinu takových bodů z komplexní roviny $\mathbb{C}$, pro které každý kořen ξ polynomu $\pi(\xi, z)$ leží uvnitř jednotkového kruhu komplexní roviny, tj. $|\xi| < 1$. Podmínka absolutní stability (8.11) tedy platí, když

$$\tau\lambda \in R_A = \{z \in \mathbb{C} \,|\, \pi(\xi, z) = 0 \;\Rightarrow\; |\xi| < 1\}.$$

8.4.2. Adamsovy metody

Integrací diferenciální rovnice (8.1) od t_n do t_{n+1} dostaneme

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))\,\mathrm{d}t.$$

Funkci $f(t, y(t))$ aproximujeme pomocí interpolačního polynomu $P_{k-1}(t)$ stupně $k-1$, tj.

$$y(t_{n+1}) \approx y(t_n) + \int_{t_n}^{t_{n+1}} P_{k-1}(t)\,\mathrm{d}t, \quad \text{kde} \quad P_{k-1}(t_{n+1-j}) = f(t_{n+1-j}, y(t_{n+1-j})).$$

Když přibližnou rovnost nahradíme rovností a přesné řešení nahradíme řešením přibližným, dostaneme předpis

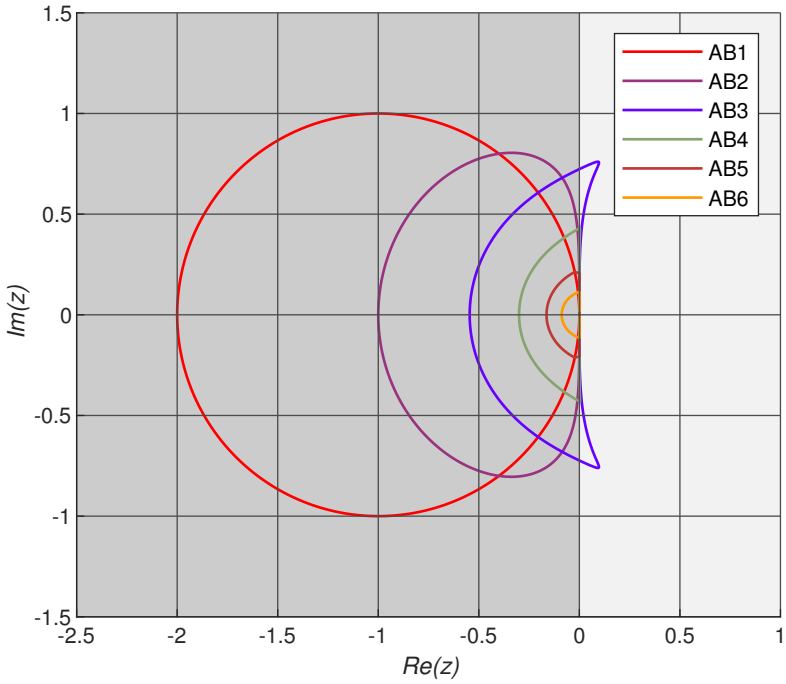
$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} P_{k-1}(t)\,\mathrm{d}t, \quad \text{kde} \quad P_{k-1}(t_{n+1-j}) = f(t_{n+1-j}, y_{n+1-j}). \tag{8.25}$$

Adams-Bashforthovy metody dostaneme, když v (8.25) zvolíme $j = 1, 2, \ldots, k$. Konstrukci polynomu $P_{k-1}(t)$ lze přehledně vyjádřit tabulkou

t	t_n	t_{n-1}	$\ldots$	t_{n+1-k}
$P_{k-1}(t)$	f_n	f_{n-1}	$\ldots$	f_{n+1-k}

Adams-Bashforthovu metodu lze zapsat ve tvaru

$$y_{n+1} = y_n + \tau \sum_{j=1}^{k} \beta^*_{k,j}\, f_{n+1-j}\,.$$



Obr. 8.5. Oblast absolutní stability ABk metod

Stručně ji budeme označovat jako ABk metodu. ABk metoda je explicitní, k-kroková, řádu k, D-stabilní. Pro konstantní délku kroku, tj. když

$$t_{n+1-j} = t_{n+1} - j\tau, \quad j = 1, 2, \ldots, k,$$

jsou koeficienty ABk metod pro $k = 1, 2, \ldots, 6$, spolu s chybovými konstantami C^*_{k+1} a dolními mezemi α^*_k intervalů absolutní stability $(\alpha^*_k, 0)$, uvedeny v následující tabulce:

k	$\beta^*_{k,1}$	$\beta^*_{k,2}$	$\beta^*_{k,3}$	$\beta^*_{k,4}$	$\beta^*_{k,5}$	$\beta^*_{k,6}$	C^*_{k+1}	α^*_k
1	1						$\frac{1}{2}$	-2
2	$\frac{3}{2}$	$-\frac{1}{2}$					$\frac{5}{12}$	-1
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$				$\frac{3}{8}$	$-\frac{8}{11}$
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$			$\frac{251}{720}$	$-\frac{3}{10}$
5	$\frac{1\,901}{720}$	$-\frac{2\,774}{720}$	$\frac{2\,616}{720}$	$-\frac{1\,274}{720}$	$\frac{251}{720}$		$\frac{95}{288}$	$-\frac{90}{551}$
6	$\frac{4\,277}{1\,440}$	$-\frac{7\,923}{1\,440}$	$\frac{9\,982}{1\,440}$	$-\frac{7\,298}{1\,440}$	$\frac{2\,877}{1\,440}$	$-\frac{475}{1\,440}$	$\frac{19\,087}{60\,480}$	$-\frac{5}{57}$

Všimněte si, že AB1 metoda je totožná s EE metodou, tj. AB1≡EE.

Adams-Moultonovy metody dostaneme, když v (8.25) zvolíme $j = 0, 1, \ldots, k-1$. Konstrukci polynomu $P_{k-1}(t)$ lze přehledně vyjádřit tabulkou

t	t_{n+1}	t_n	$\ldots$	t_{n+2-k}
$P_{k-1}(t)$	f_{n+1}	f_n	$\ldots$	f_{n+2-k}

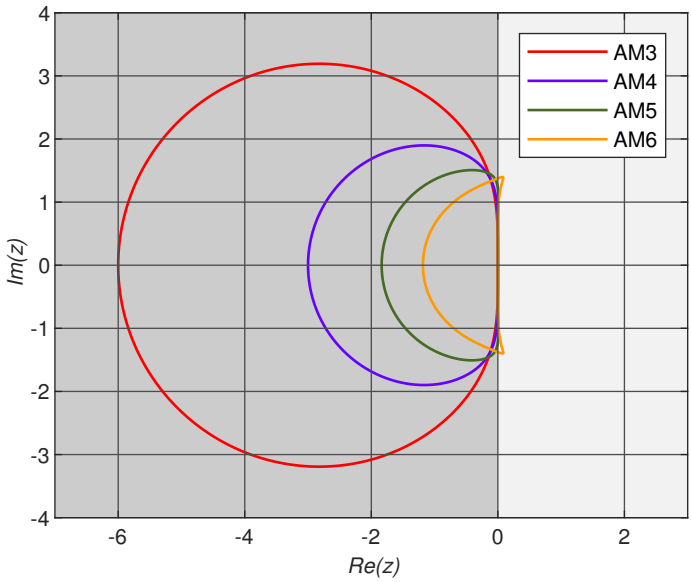
Adams-Moultonovu metodu lze zapsat ve tvaru

$$y_{n+1} = y_n + \tau\beta_{k,0}\, f(t_{n+1}, y_{n+1}) + \tau \sum_{j=1}^{k-1} \beta_{k,j}\, f_{n+1-j}\,.$$

Stručně ji budeme označovat jako AMk metodu. AMk metoda je implicitní, pro $k = 1$ je jednokroková a pro $k > 1$ je $(k-1)$-kroková, je řádu k a D-stabilní. Koeficienty AMk metod pro konstantní délku kroku a pro $k = 1, 2, \ldots, 6$, spolu s chybovými konstantami C_{k+1} a dolními mezemi α_k intervalů absolutní stability $(\alpha_k, 0)$, jsou uvedeny v následující tabulce:

k	$\beta_{k,0}$	$\beta_{k,1}$	$\beta_{k,2}$	$\beta_{k,3}$	$\beta_{k,4}$	$\beta_{k,5}$	C_{k+1}	α_k
1	1						$-\frac{1}{2}$	$-\infty$
2	$\frac{1}{2}$	$\frac{1}{2}$					$-\frac{1}{12}$	$-\infty$
3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$				$-\frac{1}{24}$	-6
4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$			$-\frac{19}{720}$	-3
5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$		$-\frac{3}{160}$	$-\frac{90}{49}$
6	$\frac{475}{1\,440}$	$\frac{1\,427}{1\,440}$	$-\frac{798}{1\,440}$	$\frac{482}{1\,440}$	$-\frac{173}{1\,440}$	$\frac{27}{1\,440}$	$-\frac{863}{60\,480}$	$-\frac{45}{38}$

Všimněte si, že AM1 metoda je totožná s IE metodou, tj. AM1≡IE, a že AM2 metoda je totožná s TR metodou, tj. AM2≡TR.



Obr. 8.6. Oblast absolutní stability AMk metod

Metody prediktor-korektor. AM metody jsou přesnější a stabilnější než AB metody. Nevýhodou AM metod je jejich implicitnost. Abychom určili y_{n+1}, musíme řešit rovnici

$$y_{n+1} = \varphi(y_{n+1})\,, \quad \text{kde} \quad \varphi(z) = y_n + \tau\beta_{k,0}\, f(t_{n+1}, z) + \tau \sum_{j=1}^{k-1} \beta_{k,j}\, f_{n+1-j}\,.$$

Použít můžeme metodu prosté iterace: zvolíme počáteční aproximaci $y_{n+1}^{(0)}$ a postupně počítáme $y_{n+1}^{(s)} = \varphi(y_{n+1}^{(s-1)})$, $s = 1, 2, \ldots$. Pro dostatečně malé τ metoda konverguje. Jestliže počáteční aproximaci $y_{n+1}^{(0)}$ určíme pomocí AB metody, provedeme jen několik málo iterací

$$y_{n+1}^{(s)} = \varphi(y_{n+1}^{(s-1)})\,, \quad s = 1, 2, \ldots, S\,,$$

a nakonec položíme

$$y_{n+1} = y_{n+1}^{(S)}\,, \quad f_{n+1} = f(t_{n+1}, y_{n+1})\,,$$

dostaneme *metodu prediktor-korektor*, kterou schématicky označujeme P(EC)^{S}E. Přitom P značí předpověď počáteční aproximace AB metodou, C korekci AM metodou a E vyhodnocení pravé strany $f(t_{n+1}, y_{n+1}^{(s)})$. Zvolíme-li jako prediktor metodu ABk a jako korektor metodu AMk, dostaneme metodu, kterou značíme ABk-AMk-P(EC)^{S}E. Její chybová konstanta je rovna chybové konstantě C_{p+1} korektoru, oblast absolutní stability se blíží oblasti absolutní stability korektoru AMk až pro $S \to \infty$. Nejčastěji se používá schéma PECE, kdy se korekce provede jen jednou a pravá strana se počítá dvakrát. V dalším se omezíme právě na schéma PECE.

Abychom mohli řídit délku kroku, potřebujeme znát odhad est_n lokální chyby. Jestliže $y_{n+1}^{*} \equiv y_{n+1}^{(0)}$ spočteme prediktorem ABk a $y_{n+1}^{**} \equiv y_{n+1}^{(1)}$ korektorem AMk, pak tzv. *Milneův*

odhad lokální chyby dává

$$\mathrm{est}_n = \frac{C_{k+1}}{C^*_{k+1} - C_{k+1}}(y^{**}_{n+1} - y^*_{n+1}), \tag{8.26}$$

odvození viz [26]. Nakonec položíme

$$y_{n+1} = y^{**}_{n+1} + \mathrm{est}_n. \tag{8.27}$$

Korekce y^{**}_{n+1} pomocí odhadu lokální chyby est_n se nazývá *lokální extrapolace*. Celý krok označujeme zkratkou ABk-AMk-PECLE, přičemž písmeno L vyznačuje použití lokální extrapolace. Metoda ABk-AMk-PECLE je řádu $k+1$, oblast absolutní stability je větší než u prediktoru ABk ale menší než u korektoru AMk, viz [26], [2].

Alternativní odhad lokální chyby lze získat také tak, že y_{n+1} spočteme metodou AM(k+1) a položíme

$$\mathrm{est}_n = y_{n+1} - y^{**}_{n+1}. \tag{8.28}$$

Výslednou metodu lze označit jako ABk-AM(k+1)-PECE. Odhady (8.27) a (8.28) nejsou sice totožné, pro malé τ jsou však prakticky nerozlišitelné.

Konkrétně pro metodu AB2-AM2-PECLE organizujeme výpočet následovně:

P:	AB2:	$y^*_{n+1} = y_n + \frac{1}{2}\tau(3f_n - f_{n-1})$
E:		$f^*_{n+1} = f(t_{n+1}, y^*_{n+1})$
C:	AM2:	$y^{**}_{n+1} = y_n + \frac{1}{2}\tau(f^*_{n+1} + f_n)$
L:		$\mathrm{est}_n = -\frac{1}{6}(y^{**}_{n+1} - y^*_{n+1}), \quad y_{n+1} = y^{**}_{n+1} + \mathrm{est}_n$
E:		$f_{n+1} = f(t_{n+1}, y_{n+1}).$

V případě AB2-AM3-PECE metody nahradíme řádek L řádkem

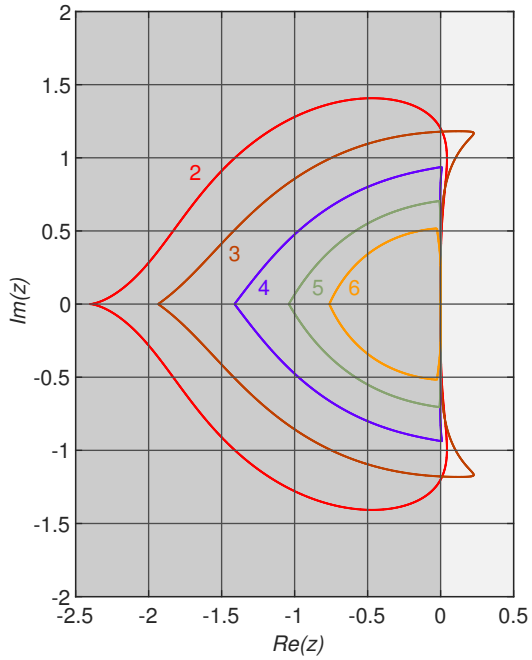
$$\text{C:} \quad \text{AM3:} \quad y_{n+1} = y_n + \tfrac{1}{12}\tau(5f^*_{n+1} + 8f_n - f_{n-1}), \quad \mathrm{est}_n = y_{n+1} - y^{**}_{n+1}.$$

Startovací hodnotu y_1 lze získat třeba pomocí EM2 metody.

Na obrázku 8.7 je vyznačena oblast absolutní stability metod ABk-AM(k+1)-PECE pro $k = 2, 3, \ldots, 6$: je větší než oblast absolutní stability prediktoru ABk, viz obrázek 8.5, avšak menší než oblast absolutní stability korektoru AM(k+1), viz obrázek 8.6.

Řízení délky kroku a řádu metody. Kvalitní programy založené na metodách prediktor-korektor mění jak délku kroku tak řád metody. Tak například matlabovský program `ode113` používá metody ABk-AM(k+1)-PECE pro $k = 1, 2, \ldots, 12$.

Změna řádu se provádí současně se změnou délky kroku. Algoritmy tohoto typu se označují jako VSVO (*variable step variable order*). Základní myšlenka je jednoduchá. Předpokládejme, že jsme vypočetli y_{n+1} metodou ABk-AM(k+1)-PECE s krokem délky τ. Určíme odhad est^k_n lokální chyby podle (8.27) nebo (8.28). Jestliže $|\mathrm{est}^k_n| > \varepsilon$, jde o neúspěch a výpočet y_{n+1} je třeba opakovat, v opačném případě pokračujeme výpočtem přibližného řešení y_{n+2}. V každém případě je však třeba stanovit novou délku kroku a nový řád. Řád



Obr. 8.7. Oblast absolutní stability pro metody ABk-AM(k+1)-PECE, $k = 2, 3, \ldots, 6$

se může změnit nejvýše o jednotku, tj. v metodě ABk-AM(k+1)-PECE místo k může být nově také $k-1$ nebo $k+1$. Odhady odpovídajících lokálních chyb est_n^{k-1} a est_n^{k+1} lze získat snadno, viz [50]. Nové délky kroků τ_{k-1}^*, τ_k^* a τ_{k+1}^* stanovíme podobně jako pro jednokrokovou metodu,

$$\tau_\ell^* = \theta\tau(\varepsilon/|\mathrm{est}_n^\ell|)^{1/(\ell+1)} \quad \text{pro } \ell = k-1,\ k,\ k+1\,,$$

a největší z čísel τ_{k-1}^*, τ_k^*, τ_{k+1}^* určí jak novou délku kroku tak nový řád. Konkrétně, je-li největší τ_k^*, k se nemění a jako novou délku kroku vezmeme $\tau^* = \tau_k^*$, je-li největší τ_{k+1}^*, zvětšíme k o jedničku a pokračujeme s krokem délky $\tau^* = \tau_{k+1}^*$ a je-li největší τ_{k-1}^*, k o jedničku snížíme a pokračujeme s krokem délky $\tau^* = \tau_{k-1}^*$.

Pro krok délky $\tau^* \neq \tau$ je třeba vypočítat hodnoty f_{n+1-j}^* pro $t_{n+1-j}^* = t_{n+1} - j\tau^*$. To lze snadno provést interpolací pomocí $f_{n+1}, f_n, \ldots$. Podrobný popis strategie VSVO lze najít např. v [26], [50].

Start metody není žádný problém: začneme metodou AB1-AM2-PECE, počáteční délku kroku určíme např. podle (8.22) a algoritmus VSVO se už sám rychle vyladí na správné hodnoty jak délky kroku tak řádu metody.

8.4.3. Metody zpětného derivování

Při řešení tzv. tuhých problémů, viz kapitola 1.5, je vhodné používat metody s neomezenou oblastí absolutní stability. Metody zpětného derivování (stručně BDF podle *backward differentiation formulas*) tuto vlastnost mají. Pro k-krokovou metodu zpětného derivování použijeme zkrácený zápis BDFk metoda. Dostaneme ji tak, že v rovnici

$$y'(t_{n+1}) = f(t_{n+1}, y(t_{n+1}))$$

nahradíme derivaci $y'(t_{n+1})$ pomocí derivace $P_k'(t_{n+1})$ interpolačního polynomu $P_k(t)$ stupně k procházejícího body $[t_{n+1}, y(t_{n+1})]$, $[t_n, y(t_n)]$, $\ldots$, $[t_{n+1-k}, y(t_{n+1-k})]$. Když pak nahradíme $y(t_{n+1-j})$ přibližnými hodnotami y_{n+1-j}, $j = 0, 1, \ldots, k$, dostaneme metodu

$$P_k'(t_{n+1}) = f(t_{n+1}, y_{n+1}).$$

Přehledné vyjádření aproximujícího polynomu $P_k(t)$ uvádí následující tabulka

t	t_{n+1}	t_n	$\ldots$	t_{n+1-k}
$P_k(t)$	y_{n+1}	y_n	$\ldots$	y_{n+1-k}

BDFk metodu zapíšeme ve tvaru

$$\alpha_{k,0} y_{n+1} + \sum_{j=1}^{k} \alpha_{k,j} y_{n+1-j} = \tau \beta_{k,0} f(t_{n+1}, y_{n+1}) \,.$$

Metoda BDFk je implicitní, k-kroková, řádu k a D-stabilní pro $k \leq 6$. Pro konstantní délku kroku jsou koeficienty $\alpha_{k,j}$, $\beta_{k,0}$ a chybové konstanty C_{k+1} BDFk metod uvedeny v následující tabulce:

k	$\alpha_{k,0}$	$\alpha_{k,1}$	$\alpha_{k,2}$	$\alpha_{k,3}$	$\alpha_{k,4}$	$\alpha_{k,5}$	$\alpha_{k,6}$	$\beta_{k,0}$	C_{k+1}	α_k
1	1	-1						1	$-\frac{1}{2}$	90°
2	1	$-\frac{4}{3}$	$\frac{1}{3}$					$\frac{2}{3}$	$-\frac{2}{9}$	90°
3	1	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$				$\frac{6}{11}$	$-\frac{3}{22}$	88°
4	1	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$			$\frac{12}{25}$	$-\frac{12}{125}$	73°
5	1	$-\frac{300}{137}$	$\frac{300}{137}$	$-\frac{200}{137}$	$\frac{75}{137}$	$-\frac{12}{137}$		$\frac{60}{137}$	$-\frac{10}{137}$	52°
6	1	$-\frac{360}{147}$	$\frac{450}{147}$	$-\frac{400}{147}$	$\frac{225}{147}$	$-\frac{72}{147}$	$\frac{10}{147}$	$\frac{60}{147}$	$-\frac{20}{343}$	18°

Všimněte si, že BDF1 metoda je totožná s IE metodou, tj. BDF1≡IE.

BDFk metody jsou implicitní, ve srovnání s implicitními AMk metodami ale mají značně větší chybové konstanty. Na druhé straně však metody zpětného derivování mají jednu ohromnou přednost, která plně ospravedlňuje jejich používání, a tou je neomezená oblast R_A absolutní stability. Pro metody BDF1 a BDF2 oblast R_A absolutní stability obsahuje celou zápornou polorovinu komplexní roviny $\mathbb{C}$, tj. $\mathrm{R}_A \supseteq \{z \in \mathbb{C} \,|\, \mathrm{Re}\, z < 0\}$. Takové metody se nazývají *A-stabilní*.

Abychom mohli popsat oblast absolutní stability zbývajících BDFk metod, zavedeme si jeden nový pojem. Řekneme, že numerická metoda je *A(α)-stabilní*, $\alpha \in (0, \pi/2\rangle$, jestliže její oblast absolutní stability R_A obsahuje nekonečný klín

$$W_\alpha = \{re^{i\varphi} \in \mathbb{C} \,|\, r > 0, |\varphi - \pi| < \alpha\}.$$

BDFk metody jsou (pro $k \leq 6$) A(α)-stabilní, příslušné úhly α_k (pro větší názornost ve stupních) jsou uvedeny jako poslední sloupec výše uvedené tabulky.

BDFk metody (pro $k \leq 6$) jsou dokonce L(α)-stabilní. L(α) stabilní metodu přitom definujeme jako A(α)-stabilní metodu takovou, že když $z \in R_A$, $\pi(\xi, z) = 0$, $Re(z) \to -\infty$, pak $\xi \to 0$. Pro $\alpha = \frac{\pi}{2}$ dostáváme L-stabilní metodu. BDF1 a BDF2 jsou tedy L-stabilní.

Úhel 18° metody BDF6 je příliš malý a proto se tato metoda obvykle nepoužívá. V matlabovském programu `ode15s` jsou implementovány metody zpětného derivování řádů 1 až 5. Program `ode15s` je typu VSVO, tj. volí optimální délku kroku i řád metody. Základní metodou programu `ode15s` je metoda NDF (podle *numerical differentiation formula*). NDFk metody jsou modifikace BDFk metod, mají o něco menší chybové konstanty (o 26% pro $k = 1, 2, 3$ a o 15% pro $k = 4$) a poněkud menší úhly $A(\alpha)$ stability (o 7% pro $k = 3$ a o 10% pro k=4) než odpovídající BDFk metody. Podrobnosti týkající se NDFk metod lze najít v [52].

Nelineární rovnice pro výpočet y_{n+1} se řeší pomocí několika málo kroků Newtonovy metody.

8.5. Tuhé problémy

Tuhý počáteční problém (v anglicky psané literatuře *stiff problem*) lze nepřímo popsat pomocí jeho charakteristických projevů. Jinak to nejde, neboť přesná definice tuhého systému neexistuje. Praktická a snadno ověřitelná je

Charakteristika 1. *Počáteční problém je tuhý, když počet kroků, který k jeho vyřešení potřebuje metoda s omezenou oblastí absolutní stability, je podstatně větší než počet kroků, který k jeho vyřešení potřebuje metoda s neomezenou oblastní absolutní stability.*

Použitelnost této charakteristiky ukážeme v následujícím

Příklad 8.1

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1000 & -1001 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad x \in (0, \ell), \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{8.29}$$

Řešení je $y_1 = -e^{-t}$, $y_2 = e^{-t}$. Úlohu (8.29) jsme řešili matlabovským programem `ode45` (DP54 metoda řádu 5 s omezenou oblastí absolutní stability) a matlabovským programem `ode15s` (BDF metody VSVO řádů 1-5 s neomezenými oblastmi absolutní stability). Oba programy jsme použili se stejným požadavkem na přesnost: $\varepsilon_r = 10^{-3}$ a $\varepsilon_a = 10^{-6}$. Efektivnost obou metod lze přibližně porovnat podle počtu úspěšně provedených kroků `pk` a počtu `pf` vyhodnocení pravé strany. V následující tabulce jsou uvedeny hodnoty `pk/pf` pro několik délek ℓ intervalu integrace.

ℓ	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
`ode45`	10/61	22/151	269/1 747	2 953/18 919	30 071/192 475
`ode15s`	10/24	10/24	12/28	42/88	71/146

Pro malé $\ell = 10^{-2}$ se na intervalu $\langle 0; 10^{-2} \rangle$ řešení poměrně rychle mění. Délku kroku zde určuje požadavek na přesnost a protože obě metody jsou téhož řádu, potřebují přibližně

stejný počet kroků. Jak však ℓ vzrůstá, délku kroku stále více začíná ovlivňovat podmínka stability. □

Abychom porozuměli dalším charakteristikám tuhosti, je vhodné pochopit, co je to

Stabilní problém. *Počáteční problém* $\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \mathbf{y}(a) = \boldsymbol{\eta}$, *je stabilní, když malá změna dat* $\mathbf{f}, a, \boldsymbol{\eta}$ *způsobí malou změnu řešení* $\mathbf{y}$. Zabývejme se speciálně *stabilitou vzhledem k počáteční podmínce*, konkrétně jak změna počáteční hodnoty $\boldsymbol{\eta}$ ovlivní řešení $\mathbf{y}$.

Pro ilustraci prozkoumejme stabilitu lineárního počátečního problému

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{g}(t), \qquad \mathbf{y}(a) = \boldsymbol{\eta}, \tag{8.30}$$

kde $\mathbf{A}$ je číselná matice řádu d, která má navzájem různá vlastní čísla $\{\lambda_j\}_{j=1}^d$. Nechť $\mathbf{u}$ je řešení problému (8.30) a $\mathbf{v}$ je řešení téže diferenciální rovnice, avšak s porušenou počáteční podmínkou, tj.

$$\begin{aligned} \mathbf{u}' &= \mathbf{A}\mathbf{u} + \mathbf{g}, \qquad \mathbf{u}(a) = \boldsymbol{\eta}, \\ \mathbf{v}' &= \mathbf{A}\mathbf{v} + \mathbf{g}, \qquad \mathbf{v}(a) = \boldsymbol{\eta} + \boldsymbol{\delta}. \end{aligned}$$

Pro $\mathbf{w} = \mathbf{v} - \mathbf{u}$ dostaneme problém

$$\mathbf{w}' = \mathbf{A}\mathbf{w}, \qquad \mathbf{w}(a) = \boldsymbol{\delta},$$

který popisuje šíření počáteční poruchy $\boldsymbol{\delta}$. Pak

$$\mathbf{w}(t) = \sum_{j=1}^{d} \kappa_j e^{\lambda_j (t-a)} \mathbf{v}_j,$$

kde $\mathbf{v}_j$ jsou vlastní vektory příslušné vlastním číslům λ_j a κ_j jsou konstanty, které určíme z počáteční podmínky:

$$\mathbf{V}\mathbf{c} = \boldsymbol{\delta}, \quad \text{kde} \quad \mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d), \quad \mathbf{c} = (c_1, c_2, \ldots, c_d)^T.$$

Odtud $\mathbf{w}(t) = \mathbf{V}\mathbf{S}(t)\mathbf{V}^{-1}\boldsymbol{\delta}$, kde $\mathbf{S} = \operatorname{diag}\{e^{\lambda_1(t-a)}, e^{\lambda_2(t-a)}, \ldots, e^{\lambda_n(t-a)}\}$. Můžeme tedy vyslovit tyto závěry:

1) Jestliže $Re(\lambda_j) < 0$ *pro všechna* j, pak $\|\mathbf{w}(t)\| \to 0$ exponenciálně pro $t \to \infty$, tj. *porucha se velmi rychle zmenšuje.*

2) Jestliže $Re(\lambda_j) \le 0$ *pro všechna* j, pak $\|\mathbf{w}(t)\| \le \|\mathbf{V}\| \cdot \|\mathbf{V}^{-1}\| \cdot \|\boldsymbol{\delta}\|$, tj. *porucha bude omezená*, přitom $\|\mathbf{w}(t)\| \to 0$ pro $\boldsymbol{\delta} \to \mathbf{o}$.

3) Jestliže *nějaké vlastní číslo* λ_j *má kladnou reálnou složku* a počáteční porucha $\boldsymbol{\delta}$ je taková, že $c_j \neq 0$, pak $\|\mathbf{w}(t)\| \to \infty$ exponenciálně pro $t \to \infty$, tj. *porucha se velmi rychle zvětšuje.*

Problém (8.30) je tedy stabilní (vzhledem k počáteční podmínce), jestliže všechna vlastní čísla matice $\mathbf{A}$ jsou navzájem různá a mají nekladnou reálnou složku.

Toto tvrzení lze rozšířit a připustit i násobná vlastní čísla se zápornou reálnou složkou:

Problém (8.30) *je stabilní (vzhledem k počáteční podmínce), jestliže:*
(a) *všechna vlastní čísla matice* $\mathbf{A}$ *mají nekladnou reálnou složku,*
(b) *ryze imaginární vlastní čísla jsou navzájem různá.*

Řešení úlohy (8.30) lze zapsat ve tvaru

$$\mathbf{y}(t) = \mathbf{y}_h(t) + \mathbf{p}(t), \qquad \text{kde} \quad \mathbf{y}_h(t) = \sum_{j=1}^{d} c_j e^{\lambda_j t} \mathbf{v}_j$$

je obecné řešení přidružené homogenní rovnice, $\mathbf{p}$ je partikulární řešení a c_j jsou konstanty, které určíme z počátečních podmínek. Předpokládejme, že $\mathrm{Re}(\lambda_j) < 0$, $j = 1, 2, \ldots, d$. Pak $\mathbf{y}(t) \to \mathbf{p}(t)$ pro $t \to \infty$. Z toho důvodu mluvíme o funkci $\mathbf{y}_h$ jako o *přechodovém řešení* a o funkci $\mathbf{p}$ jako o *ustáleném řešení.*

Označme jako λ_{max} a λ_{min} taková vlastní čísla, pro která

$$|\mathrm{Re}(\lambda_{max})| \geq |\mathrm{Re}(\lambda_j)| \geq |\mathrm{Re}(\lambda_{min})|\,, \quad j = 1, 2, \ldots, d\,.$$

Chceme-li určit ustálené řešení, musíme úlohu řešit alespoň do toho bodu, v němž je už nejpomaleji klesající exponenciála v přechodovém řešení zanedbatelná.

Řešíme-li homogenní úlohu $\mathbf{y}' = \mathbf{A}\mathbf{y}$ numericky, pak lze snadno ukázat, že podmínka stability $\mathbf{y}_n \to \mathbf{o}$ pro $t_n = n\tau \to \infty$ platí, právě když

$$\{\lambda_1\tau, \lambda_2\tau, \ldots, \lambda_d\tau\} \subseteq R_A, \tag{8.31}$$

kde R_A je oblast absolutní stability uvažované numerické metody.

Podle (8.31) tedy pro délku kroku dostaneme omezení $\tau|\mathrm{Re}(\lambda_{min})| < |I_A|$, kde $|I_A|$ je délka intervalu absolutní stability. Počet Q kroků potřebných k uspokojivému vyřešení úlohy (8.30) lze tedy přibližně odhadnou výrazem

$$Q = \frac{b-a}{\tau} \doteq C\frac{|\mathrm{Re}(\lambda_{max})|}{|\mathrm{Re}(\lambda_{min})|}, \quad \text{kde} \quad C = \frac{|\ln\varepsilon|}{|I_A|}.$$

Ve shodě s charakteristikou 1 je proto přirozené považovat podíl

$$S(\mathbf{A}) = \frac{|\mathrm{Re}(\lambda_{max}(\mathbf{A}))|}{|\mathrm{Re}(\lambda_{min}(\mathbf{A}))|}\,,$$

nazývaný *poloměr tuhosti* matice $\mathbf{A}$, za měřítko tuhosti soustavy (8.30). Můžeme proto vyslovit další

Charakteristiku 2. *Problém* (8.30) *je tuhý, jestliže všechna vlastní čísla matice* $\mathbf{A}$ *mají zápornou reálnou část a poloměr tuhosti* $S(\mathbf{A})$ *matice* $\mathbf{A}$ *je velký.*

Podobná je

Charakteristika 3. *Problém* (8.30) *je tuhý, jestliže všechna vlastní čísla matice* $\mathbf{A}$ *mají zápornou reálnou část a jestliže součin spektrálního poloměru* $\varrho(\mathbf{A})$ *matice* $\mathbf{A}$ *a délky intervalu integrace* $b-a$ *je velký, tj. když* $\varrho(\mathbf{A})(b-a) \gg 1$.

Vraťme se nyní k úloze (8.30).

Příklad 8.1 – pokračování. Pravá strana diferenciální rovnice je

$$\mathbf{f} = \mathbf{A}\mathbf{y}, \qquad \text{kde} \quad \mathbf{A} = \begin{pmatrix} 0 & 1 \\ -1000 & -1001 \end{pmatrix}$$

Vlastní čísla matice $\mathbf{A}$ jsou $\lambda_1 = -1$, $\lambda_2 = -1000$, takže $S(\mathbf{A}) = \varrho(\mathbf{A}) = 1000$. Pro větší ℓ je $\varrho(\mathbf{A})(b-a) = 1000\,\ell \gg 1$, tj. jde o tuhý problém. Metoda DP54 má interval absolutní stability $(-3{,}31; 0)$, takže podmínka stability (8.31) vyžaduje, aby $-3{,}31 < -1000\tau$, tj. $\tau < 0{,}00331$. Na intervalu délky ℓ je tak třeba $n_\ell > \ell/0{,}00331$ kroků délky menší než 0,00331, což je v souladu s tabulkou uvedenou v první části příkladu 1.1. Skutečně, na intervalu$\langle 10; 100\rangle$ délky 90 je třeba alespoň $90/0{,}00331 \doteq 27\,190$ kroků délky 0,00331, ve skutečnosti program `ode45` provedl přibližně stejný počet $30\,071 - 2\,953 = 27\,118$ kroků proměnné délky. Protože přesná řešení $y_1 = -e^{-t}$ a $y_2 = e^{-t}$ jsou na intervalu $\langle 10; 100\rangle$ téměř konstantní, rovná nule, je zřejmé, že délka kroku je omezena z důvodu stability a ne z důvodu přesnosti.

Nenechte se zmást tím, že nestabilita se projevuje prostřednictvím lokální chyby, tj. pomocí nástroje pro měření přesnosti metody: jestliže délka kroku nesplňuje podmínku stability, dojde v několika málo krocích k takovému nárustu lokální chyby, že na to zareaguje mechanismus automatického řízení délky kroku a krok patřičně zkrátí. □.

Právě uvedený příklad vystihuje charakteristikou vlastnost tuhých soustav ODR. Tuto vlastnost sformulujeme jako

Charakteristika 4. *Tuhost soustavy ODR se projevuje tím, že při jejím numerickém řešení délku kroku omezuje spíše stabilita metody než její přesnost.*

Lineární stabilita. Až dosud jsme naše úvahy o stabilitě numerické metody zakládali na analýze jejího chování při řešení lineární soustavy (8.30) s konstantní maticí $\mathbf{A}$. Je jistě rozumné požadovat, aby numerická metoda fungovala dobře na speciální třídě rovnic (8.30), je však třeba mít na paměti, že tak dostáváme jen hrubou informaci o možném chování numerické metody při řešení problémů složitějších. Ukažme si nyní, proč se můžeme domnívat, že taková informace má vůbec nějakou vypovídací hodnotu.

Pro obecný problém $\mathbf{u}' = \mathbf{f}(t, \mathbf{u})$, $\mathbf{u}(t_\alpha) = \mathbf{y}_\alpha$, aproximujeme funkci $\mathbf{f}$ v blízkosti bodu $(t_\alpha, \mathbf{y}_\alpha)$ funkcí lineární v proměnné $\mathbf{u}$,

$$\mathbf{f}(t, \mathbf{u}) \approx \mathbf{f}(t_\alpha, \mathbf{y}_\alpha) + \frac{\partial \mathbf{f}}{\partial t}(t_\alpha, \mathbf{y}_\alpha)(t - t_\alpha) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_\alpha, \mathbf{y}_\alpha)(\mathbf{u} - \mathbf{y}_\alpha),$$

a doufáme, že chování numerického řešení původního problému lze v blízkosti bodu $(t_\alpha, \mathbf{y}_\alpha)$ objasnit z chování numerického řešení aproximujícího lineárního problému

$$\mathbf{u}' = \mathbf{A}_\alpha \mathbf{u} + \mathbf{g}_\alpha(t), \qquad \mathbf{u}(t_\alpha) = \mathbf{y}_\alpha, \tag{8.32}$$

kde $\mathbf{A}_\alpha = \mathbf{f}_\mathbf{y}(t_\alpha, \mathbf{y}_\alpha)$, $\mathbf{g}_\alpha(t) = \mathbf{f}(t_\alpha, \mathbf{y}_\alpha) + \mathbf{f}_t(t_\alpha, \mathbf{y}_\alpha)(t - t_\alpha) - \mathbf{A}_\alpha \mathbf{y}_\alpha$.

Aproximující problém je téhož typu jako problém (8.30), takže stabilitu numerické metody můžeme posoudit známým způsobem pomocí délky kroku a vlastních čísel konstantní matice $\mathbf{A}_\alpha$. Jestliže to uděláme, tak vlastně tiše předpokládáme, že stabilita metody aplikované na aproximující problém je stejná jako stabilita metody aplikované na

problém obecný. To obecně není pravda, přesto však tento přístup může být pro praxi přínosem, nesmíme ale zapomenout, že případná podmínka stability omezující délku kroku má lokální charakter, tj. platí jen v okolí zvoleného bodu $(t_\alpha, \mathbf{y}_\alpha)$.

Právě uvedený postup odpovídá klasickému postupu, který se používá v teorii diferenciálních rovnic, kdy se stabilita řešení nelineárního problému zkoumá prostřednictvím stability aproximujícího lineárního problému. V našem případě je třeba analyzovat navíc ještě stabilitu numerické metody aplikované na aproximující problém. Z teorie obyčejných diferenciálních rovnic je známo, že analýza lineární stability je užitečná, je však třeba mít na zřeteli její omezenou působnost. Totéž platí tím spíše i pro analýzu lineární stability numerických metod. Dokonalejší nástroje pro posuzování stability numerických metod při řešení nelineárních problémů vycházejí z teorie nelineární stability ODR, tím se však my zde zabývat nebudeme, základní informace k tomuto tématu lze najít např. v [26].

Příklad 8.2 Uvažujme počáteční problém

$$y' = y^2 - y^3\,, \qquad t \in (0, 2/\delta)\,, \qquad y(0) = \delta\,. \tag{8.33}$$

Diferenciální rovnice má dvě konstantní řešení $y = 0$ a $y = 1$: zvolíme-li počáteční podmínku $y(0) \le 0$, pak $y(t) \to 0$ pro $t \to \infty$, zatímco pro $y(0) > 0$ dostaneme $y(t) \to 1$ pro $t \to \infty$. Zvolíme-li $\delta > 0$ velmi malé, pak pravá strana $y^2 - y^3$ diferenciální rovnice nabývá malých kladných hodnot, tj. funkce $y(t)$ velmi pomalu roste a na poměrně dlouhém intervalu zůstávají její hodnoty blízké k 0. Konkrétně pro $\delta = 10^{-4}$ je $y(t) < 10^{-2}$ ještě pro $t = 9\,900$, pak $y(t)$ začíná prudce růst a pro $t > 10\,020$ je už $y(t)$ prakticky rovno jedné.

Tuhost úlohy (8.33) posoudíme pomocí lineární stability, viz předchozí odstavec. Aproximující lineární problém v bodu (t_α, y_α) je podle (8.32) tvaru

$$y' = A_\alpha y, \quad \text{kde} \quad A_\alpha = 2y_\alpha - 3y_\alpha^2.$$

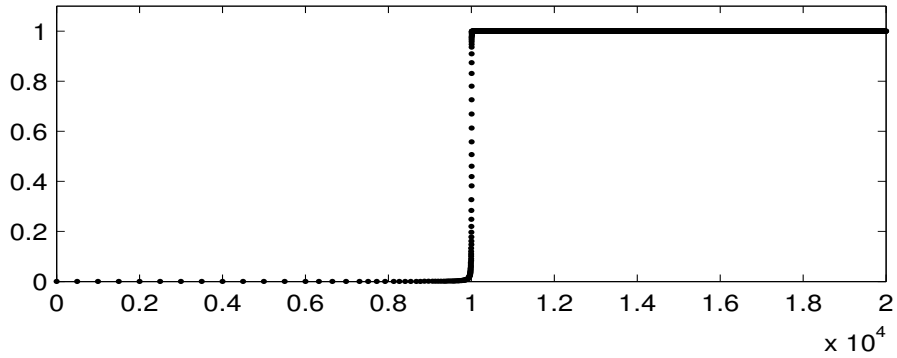
Spektrální poloměr $\varrho(A_\alpha) = |2y_\alpha - 3y_\alpha^2|$. Pro malé $y_\alpha \approx 0$ je $|2y_\alpha - 3y_\alpha^2| \approx 0$, pro $y_\alpha \approx 1$ je však $|2y_\alpha - 3y_\alpha^2| \approx 1$. Pro $t_\alpha \in \langle 0, 9\,900\rangle$ je výraz $|2y_\alpha - 3y_\alpha^2|t_\alpha$ charakterizující tuhost poměrně malý, nejde zde proto o tuhý problém, takže délka kroku se řídí především přesností metody. V intervalu $(9\,900; 10\,020)$ se řešení prudce mění, to mechanismus automatického řízení délky kroku zachytí a krok zkrátí. Důvodem zkrácení kroku je zde spíše prudká změna řešení než narůstající tuhost. Poté, co je řešení rovno přibližně jedné, je délka kroku řízena stabilitou. Skutečně, pro $t_\alpha \in \langle 10\,020, 20\,000\rangle$ je $|2y_\alpha - 3y_\alpha^2|(20\,000 - t_\alpha) \doteq 20\,000 - t_\alpha$, pro $t_\alpha = 10\,020$ dostaneme velké číslo 9 980 signalizující tuhost.

Úlohu (8.33) jsme řešili explicitní metodou DP54 (matlabovský program `ode45`) a implicitní TR metodou (matlabovský program `ode23t`). Oba programy jsme použili se stejnou přesností ($\varepsilon_r = 10^{-4}$, $\varepsilon_a = 10^{-7}$).

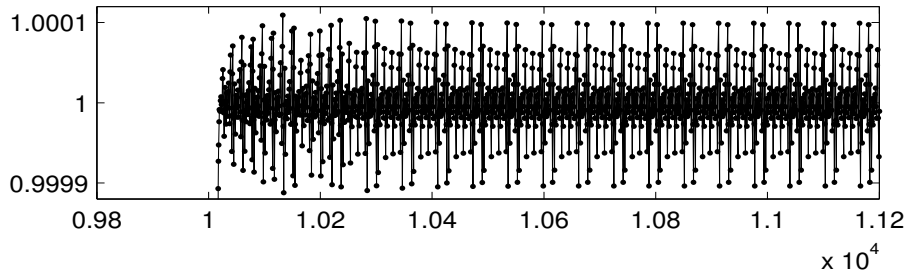
DP54 je explicitní Rungova-Kuttova metoda řádu 5 s intervalem absolutní stability $(-3{,}31; 0)$. Délka kroku proto musí splňovat podmínku stability $\tau|2y - 3y^2| < 3{,}31$, což pro $t > 10\,020$ znamená volit $\tau \doteq 3{,}31$. TR metoda je A-stabilní metoda řádu 2, takže tato metoda délku kroku z důvodu stability nijak neomezuje.

Vidíme, že v intervalu $(10\,020, 20\,000)$, kde přesné řešení je prakticky rovno 1, je TR-metoda velmi efektivní, na zdolání intervalu $(10\,020, 20\,000)$ potřebovala jen 8 kroků. Zato

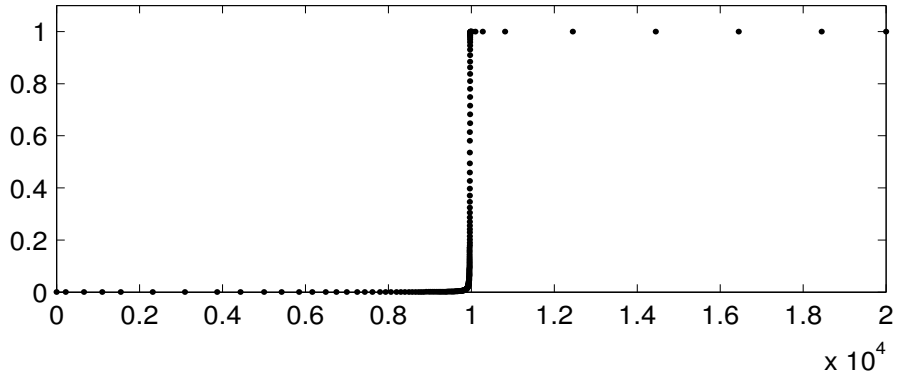
metoda DP54 na tomto intervalu provedla 3 005 kroků délky $\tau \doteq 3{,}31$, takže její použití rozhodně vhodné není. □



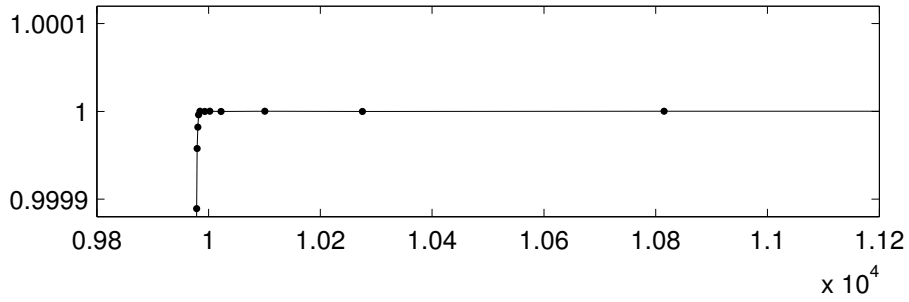
Obr. 8.8 Příklad 8.2 řešený DP54 metodou, celý výpočet



Obr. 8.9 Příklad 8.2 řešený DP54 metodou, detail



Obr. 8.10 Příklad 8.2 řešený TR metodou, celý výpočet



Obr. 8.11. Příklad 8.2 řešený TR metodou, detail

Výsledky příkladu 8.2 umožňují vyslovit praktické kritérium, podle něhož lze vcelku spolehlivě zjistit, zda uvažovaný problém je tuhý.

Charakteristika 5. *Jestliže numerická metoda s omezenou oblastí absolutní stability, aplikovaná na počáteční problém, je nucena v jistém intervalu integrace používat krok, jehož délka je nepřiměřeně malá vzhledem k hladkosti přesného řešení v tomto intervalu, pak to znamená, že problém je v tomto intervalu tuhý.*

Velmi zjednodušený postup řešení počáteční úlohy, o níž nevíme, zda je či není tuhá, je tedy tento: zkusíme úlohu vyřešit přesnou explicitní metodou (v MATLABu třeba programem `ode113` založeným na Adamsových formulích řádů 1 až 13) a pokud řešení bude trvat příliš dlouho v důsledku volby extrémně krátkých kroků, výpočet přerušíme a zkusíme program pro řešení tuhých úloh (v MATLABu třeba program `ode15s` založený na metodách zpětného derivování řádů 1 až 5). Pokud pravá strana diferenciální rovnice není příliš hladká, je vhodnější použít metody nižších řádů, třeba explicitní metodu BS32 (v MATLABu program `ode23`) a v případě neúspěchu A-stabilní implicitní metodu, třeba TR (v MATLABu program `ode23t`) nebo TR-BDF2 (v MATLABu program `ode23tb`).

Pokud dopředu víme, že problém je resp. není tuhý, neexperimentujeme zbytečně a přímo použijeme program určený pro řešení příslušného typu úloh.

Chceme-li efektivně využít všechny možnosti, které nám dostupné programy nabízejí, je třeba se seznámit s jejich vlastnostmi, vybrat správný program a vhodně nastavit jeho řídicí parametry (v kolekci programů v MATLABu jsou to například parametry ovlivňující řízení délky kroku, výstup výsledků, u implicitních metod je třeba určit způsob výpočtu Jacobiho matice pravé strany $\mathbf{f}$, a jsou ještě další parametry, o nichž se lze poučit z programové dokumentace).

Metody pro řešení tuhých problémů. Pro řešení tuhých problémů je třeba používat metody s neomezenou oblastí absolutní stability. V Matlabu jsou to metody NDF a BDF implementované v programu `ode15s`, TR metoda implementovaná v programu `ode23t` a dvě L-stabilní metody řádu 2: TR-BDF2 metoda (jde o kombinaci metod TR a BDF2) implementovaná v programu `ode23tb` a Rosenbrockova metoda implementovaná v programu `ode23s`, podrobnosti viz [32]. Programy `ode15s`, `ode23t` a `ode23tb` vyžadují řešení nelineárních soustav rovnic tvaru

$$\mathbf{y}_{n+1} = \tau\gamma\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}) + \boldsymbol{\psi}, \tag{8.34}$$

kde parametr γ je charakteristická konstanta metody a $\boldsymbol{\psi}$ je vektor nezávislý na $\mathbf{y}_{n+1}$.

Například pro TR metodu (8.13) je

$$\gamma = \tfrac{1}{2} \text{ a } \boldsymbol{\psi} = \mathbf{y}_n + \tfrac{1}{2}\tau\mathbf{f}(t_n, \mathbf{y}_n).$$

Rovnici (8.34) řešíme zjednodušenou Newtonovou metodou. Počáteční aproximaci $\mathbf{y}_{n+1}^{(0)}$ získáme dostatečně přesnou extrapolací z hodnot $\mathbf{y}_n, \mathbf{y}_{n-1}, \ldots$. Další aproximace $\mathbf{y}_{n+1}^{(s+1)}$ získáme řešením soustav lineárních rovnic

$$(\mathbf{I} - \tau\gamma\mathbf{J})\,(\mathbf{y}_{n+1}^{(s+1)} - \mathbf{y}_{n+1}^{(s)}) = \tau\gamma\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}^{(s)}) + \boldsymbol{\psi} - \mathbf{y}_{n+1}^{(s)}\,, \tag{8.35}$$

kde $\mathbf{I}$ je jednotková matice a $\mathbf{J}$ je Jacobiho matice $\mathbf{f}_\mathbf{y}(t_{n+1-k}, \mathbf{y}_{n+1-k})$ pro nějaké $k > 0$ (jedná se tedy o zjednodušenou Newtonovu metodu, pokud by $\mathbf{J} = \mathbf{f}_\mathbf{y}(t_{n+1}, \mathbf{y}_{n+1}^{(s)})$, šlo by o klasickou Newtonovu metodu). Výpočet organizujeme takto: označíme

$$\mathbf{G} = \mathbf{I} - \tau\gamma\mathbf{J}, \qquad \mathbf{d}_s = \mathbf{y}_{n+1}^{(s+1)} - \mathbf{y}_{n+1}^{(s)}, \qquad \mathbf{g}_s = \tau\gamma\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}^{(s)}) + \boldsymbol{\psi} - \mathbf{y}_{n+1}^{(s)}$$

a vypočteme nejdříve $\mathbf{d}_s$ jako řešení soustavy lineárních rovnic $\mathbf{G}\mathbf{d}_s = \mathbf{g}_s$ a pak dopočítáme $\mathbf{y}_{n+1}^{(s+1)} = \mathbf{y}_{n+1}^{(s)} + \mathbf{d}_s$. Je-li přírustek $\mathbf{d}_s$ dostatečně malý, klademe $\mathbf{y}_{n+1} = \mathbf{y}_{n+1}^{(s+1)}$.

Matice soustavy $\mathbf{G}$ obsahuje tři členy, které se mohou měnit: τ při změně délky kroku, γ při změně metody (třeba ve VSVO implementaci metod zpětného derivování) a $\mathbf{J}$ při přepočítání Jacobiho matice. Pokud se žádný z těchto členů nezmění, zůstává matice $\mathbf{G}$ stejná. Toho je třeba využít: pouze při změně $\mathbf{G}$ provedeme výpočetně náročný LU rozklad matice soustavy $\mathbf{G} = \mathbf{L}\mathbf{U}$, kde $\mathbf{L}$ je dolní trojúhelníková matice a $\mathbf{U}$ horní trojúhelníková matice. V následujících iteracích, kdy se matice soustavy $\mathbf{G}$ nemění, provádíme výpočetně nenáročná řešení dvou soustav lineárních rovnic s trojúhelníkovou maticí soustavy, tj. $\mathbf{L}\boldsymbol{\delta}_s = \mathbf{g}_s$ a pak $\mathbf{U}\mathbf{d}_s = \boldsymbol{\delta}_s$.

Program, který má pracovat efektivně, musí mít promyšlenou strategii, podle níž rozhodne, kdy změní $\mathbf{J}$, což je výpočetně nejnáročnější, a kdy jen τ nebo γ, což znamená nový LU rozklad matice $\mathbf{G}$. Používá se „konzervativní strategie“: přepočítání Jacobiho matice se provede až tehdy, když Newtonova metoda nekonverguje dostatečně rychle, délku kroku případně metodu změníme, až když očekávaný zisk takové akce převýší náklady spojené s LU rozkladem.

Jacobiho matici lze zadat přesně nebo ji lze spočítat přibližně numericky (Matlab užívá funkci `odenumjac` z privátní knihovny toolboxu `matlab\funfun`). Pokud je Jacobiho matice řídká a uživatel zadá pozice jejích nenulových prvků, výpočet Jacobiho matice lze značně urychlit. Do dalších podrobností už zacházet nebudeme, zájemce odkazujeme na skvělou monografii [50] a na matlabovskou dokumentaci [32]. Významným zdrojem poučení je studium kódů jednotlivých matlabovských programů, příliš snadné čtení to však není.

Program `ode23s`, založený na implementaci Rosenbrockovy metody, se od zbývajících programů `ode15s`, `ode23t` a `ode23tb` liší v tom, že se vyhýbá řešení nelineárních rovnic. V každém kroku Rosenbrockovy metody je třeba sestavit Jacobiho matici $\mathbf{f}_\mathbf{y}(t_n, \mathbf{y}_n)$ a vektor derivací $\mathbf{f}_t(t_n, \mathbf{y}_n) = \{\partial f_i(t_n, \mathbf{y}_n)/\partial t\}_{i=1}^d$, provést LU rozklad matice $\mathbf{I} - \gamma\tau\mathbf{f}_\mathbf{y}(t_n, \mathbf{y}_n)$ a užitím tohoto rozkladu vyřešit tři soustavy lineárních rovnic, podrobnosti viz [52].

Statistika o činnosti matlabovských programů pro řešení ODR. Kvalitní programy uživateli vždy poskytují informaci o tom, jak úspěšně si při řešení konkrétního

problému počínaly. V Matlabu se dodávají tato čísla:

- `pk` počet úspěšných kroků
- `pn` počet neúspěšných kroků
- `pf` počet vyhodnocených pravých stran $\mathbf{f}$
- `pj` počet sestavených Jacobiho matic $\mathbf{J}$
- `pr` počet LU rozkladů $\mathbf{J} = \mathbf{LU}$
- `ps` počet řešených soustav lineárních rovnic $\mathbf{LU}\mathbf{d}_s = \mathbf{g}_s$

Pro ilustraci uvádíme

Příklad 8.3 Robertsonův problém

$$\begin{aligned} y_1' &= -0{,}04\, y_1 + 10^4\, y_2 y_3\,, & y_1(0) &= 1\,, \\ y_2' &= 0{,}04\, y_1 - 10^4\, y_2 y_3 - 3 \cdot 10^7\, y_2^2\,, & y_2(0) &= 0\,, \\ y_3' &= 3 \cdot 10^7\, y_2^2\,, & y_3(0) &= 0 \end{aligned}$$

popisuje koncentrace tří příměsí v chemické reakci, tj. $0 \le y_1, y_2, y_3 \le 1$, nezávisle proměnná t je čas, blíže viz [50]. Jacobiho matice této soustavy je

$$\mathbf{J} = \begin{pmatrix} -0{,}04 & 10^4\, y_3 & 10^4\, y_2 \\ 0{,}04 & -10^4\, y_3 - 6 \cdot 10^7 y_2 & -10^4 y_2 \\ 0 & 6 \cdot 10^7 y_2 & 0 \end{pmatrix}.$$

V čase $t = 0$, tj. pro $y_1 = 1$, $y_2 = y_3 = 0$, má Jacobiho matice vlastní čísla $\{-0{,}04; 0; 0\}$. Z fyzikálních úvah plyne, že $y_1, y_2 \to 0$ a $y_3 \to 1$ pro $t \to \infty$. Vlastní čísla Jacobiho matice pro $y_1 = y_2 = 0$, $y_3 = 1$, jsou $\{-10\,000{,}04; 0; 0\}$. Při řešení na intervalu $(0, 10^{10})$ je Robertsonův problém tuhý. O tom se lze ostatně snadno přesvědčit experimentálně: explicitní metody selhávají, metody pro řešení tuhých problémů zabírají. Numerickým výpočtem lze zjistit, že již pro $t > 0{,}01$ je spektrální poloměr $\varrho(\mathbf{J}) > 2 \cdot 10^3$, takže Robertsonův problém lze považovat za tuhý již na nepoměrně kratším intervalu délky řádově v jednotkách.

Úlohu jsme řešili dvěma matlabovskými programy určenými pro tuhé problémy: programem `ode23t` (TR metoda) a programem `ode15s` (metody BDFk, k=1,2,...,5). Délku kroku jsme řídili pomocí tolerancí $\varepsilon_r = 10^{-3}$, $\varepsilon_a = 10^{-6}$, činnost programu `ode15s` jsme omezili tak, aby pracoval jen s BDF metodami řádů 1, 2 a 3. Do následující tabulky jsme zapsali „statistiku“ výpočtu, tj. čísla `pk`, `pn`, `pf`, `pj`, `pr`, `ps`:

	pk	pn	pf	pj	pr	ps
ode23t	238	74	794	37	188	644
ode15s	245	15	504	11	67	458

Z tabulky plyne, že oba testované programy Robertsonův problém úspěšně vyřešily, program `ode15s` se podle statistiky jeví jako efektivnější.

9. Obyčejné diferenciální rovnice: okrajové úlohy

Okrajový problém pro soustavu ODR1 spočívá v určení funkce $\mathbf{y}(x)$, která v intervalu (a, b) splňuje diferenciální rovnici

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)) \tag{9.1}$$

a v koncových bodech intervalu (a, b) vyhovuje okrajové podmínce

$$\mathbf{r}(\mathbf{y}(a), \mathbf{y}(b)) = \mathbf{o}. \tag{9.2}$$

Stejně jako v kapitole 8.1 předpokládáme, že počet rovnic jakož i počet okrajových podmínek je roven d, tedy

$$\mathbf{y}(x) = (y_1(x), y_2(x), \ldots, y_d(x))^T, \qquad \mathbf{y}'(x) = (y_1'(x), y_2'(x), \ldots, y_d'(x))^T,$$
$$\mathbf{f}(x, \mathbf{y}(x)) = (f_1(x, \mathbf{y}(x)), f_2(x, \mathbf{y}(x)), \ldots, f_d(x, \mathbf{y}(x)))^T,$$
$$\mathbf{r}(\mathbf{y}(a), \mathbf{y}(b)) = (r_1(\mathbf{y}(a), \mathbf{y}(b)), r_2(\mathbf{y}(a), \mathbf{y}(b)), \ldots, r_d(\mathbf{y}(a), \mathbf{y}(b)))^T.$$

Ve speciálním případě, když

$$\mathbf{r}(\mathbf{y}(a), \mathbf{y}(b)) = \mathbf{y}(a) - \boldsymbol{\eta} = \mathbf{o} \quad \text{nebo-li} \quad \mathbf{y}(a) = \boldsymbol{\eta}, \tag{9.3}$$

přechází problém (9.1)–(9.2) v počáteční problém (8.4). Pokud je však v podmínkách (9.2) obsažena alespoň jedna složka vektoru $\mathbf{y}(a)$ a současně také alespoň jedna složka vektoru $\mathbf{y}(b)$, jde o problém okrajový. Právě tímto případem se budeme v dalším zabývat.

Podstatný rozdíl mezi počáteční a okrajovou úlohou spočívá v tom, že zatímco řešení úlohy s počátečními podmínkami existuje a je jediné pro dosti širokou třídu diferenciálních rovnic, u okrajové úlohy s velmi jednoduchou diferenciální rovnicí je možné, že řešení neexistuje nebo naopak, že řešení je nekonečně mnoho. Tuto skutečnost si ukažme na rovnici

$$y' = y, \quad \text{jejíž obecné řešení je} \quad y = Ce^x.$$

Odtud plyne, že pro okrajovou podmínku

(a) $y(0) = y(1)$ existuje jediné řešení $y = 0$,

(b) $e\,y(0) = y(1)$ existuje nekonečně mnoho řešení $y = Ce^x$, C libovolné,

(c) $e\,y(0) = y(1) + 1$ řešení neexistuje.

V kapitole 9.1 ukážeme, jak řešit okrajový problém pro soustavu ODR1 metodou střelby. V následujících kapitolách pak popíšeme tři nejznámější metody řešení okrajového problému pro ODR2: v kapitole 9.2 diferenční metodu, v kapitole 9.3 metodu konečných objemů a v kapitole 9.4 metodu konečných prvků.

9.1. Metoda střelby

Metoda střelby je založena na numerickém řešení počátečního problému

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)), \qquad \mathbf{y}(a) = \boldsymbol{\eta}, \tag{9.4}$$

za omezující podmínky

$$\mathbf{r}(\boldsymbol{\eta}, \mathbf{y}(b)) = \mathbf{o}. \tag{9.5}$$

Neznámá počáteční hodnota $\boldsymbol{\eta}$ je určena implicitně rovnicí (9.5).

Lichoběžníková metoda. Nechť $a = x_0 < x_1 < \cdots < x_N = b$ je dělení intervalu $\langle a, b\rangle$, $h_i = x_{i+1} - x_i$ je délka kroku a $h = \max_i h_i$. Přibližné řešení $\mathbf{y}_i$, $i = 0, 1, \ldots, N$, dostaneme jako řešení soustavy $d(N+1)$ rovnic

$$\begin{aligned} &\mathbf{y}_{i+1} = \mathbf{y}_i + \tfrac{1}{2}h_i[\mathbf{f}(x_i, \mathbf{y}_i) + \mathbf{f}(x_{i+1}, \mathbf{y}_{i+1})], \quad i = 0, 1, \ldots, N-1\,, \\ &\mathbf{r}(\mathbf{y}_0, \mathbf{y}_N) = 0\,. \end{aligned} \tag{9.6}$$

Soustava (9.6) je obecně nelineární. Její řešení se obvykle počítá pomocí nějaké varianty Newtonovy metody. Aby nastala konvergence, je třeba dodat dosti dobrou počáteční aproximaci $\mathbf{y}_i^{(0)} \approx \mathbf{y}(x_i)$, $i = 0, 1, \ldots, N$. Lichoběžníková metoda je řádu 2, tj. je-li funkce $\mathbf{f}$ dostatečně hladká, pro chybu platí

$$\mathbf{y}(x_i) - \mathbf{y}_i = O(h^2),$$

čímž se míní, že řádu $O(h^2)$ je každá složka vektoru $\mathbf{y}(x_i) - \mathbf{y}_i$.

Ve speciálním případě, když $\mathbf{f}(x, \mathbf{y})$ je lineární v proměnné $\mathbf{y}$ a $\mathbf{r}(\mathbf{u}, \mathbf{v})$ je lineární v obou proměnných $\mathbf{u}$ i $\mathbf{v}$, bude soustava rovnic (9.6) lineární. Nechť tedy

$$\begin{aligned} &\mathbf{y}' = \mathbf{A}(x)\mathbf{y} + \mathbf{q}(x)\,, \\ &\mathbf{B}_a\mathbf{y}(a) + \mathbf{B}_b\mathbf{y}(b) = \mathbf{c}\,. \end{aligned} \tag{9.7}$$

Soustava (9.6) je pak tvaru

$$\begin{aligned} &\left[-\mathbf{I} - \tfrac{1}{2}h_i\mathbf{A}(x_i)\right]\mathbf{y}_i + \left[\mathbf{I} - \tfrac{1}{2}h_i\mathbf{A}(x_{i+1})\right]\mathbf{y}_{i+1} = \tfrac{1}{2}h_i[\mathbf{q}(x_i) + \mathbf{q}(x_{i+1})]\,, \\ &\mathbf{B}_a\mathbf{y}_0 + \mathbf{B}_b\mathbf{y}_N = \mathbf{c}\,, \end{aligned}$$

kde $\mathbf{I}$ je jednotková matice. Maticový zápis této soustavy je

$$\begin{pmatrix} \mathbf{R}_0 & \mathbf{S}_0 & & & \\ & \mathbf{R}_1 & \mathbf{S}_1 & & \\ & & \ddots & \ddots & \\ & & & \mathbf{R}_{N-1} & \mathbf{S}_{N-1} \\ \mathbf{B}_a & & & & \mathbf{B}_b \end{pmatrix} \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{N-1} \\ \mathbf{y}_N \end{pmatrix} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{N-1} \\ \mathbf{c} \end{pmatrix}, \tag{9.8}$$

kde

$$\mathbf{R}_i = -\mathbf{I} - \tfrac{1}{2}h_i\mathbf{A}(x_i)\,, \quad \mathbf{S}_i = \mathbf{I} - \tfrac{1}{2}h_i\mathbf{A}(x_{i+1})\,, \quad \mathbf{v}_i = \tfrac{1}{2}h_i[\mathbf{q}(x_i) + \mathbf{q}(x_{i+1})]\,.$$

Simpsonova metoda. Matlab nabízí pro řešení okrajového problému (9.1)–(9.2) program `bvp4c`, který je založen na použití Simpsonovy formule

$$\begin{aligned} &\mathbf{y}_{i+1} = \mathbf{y}_i + \tfrac{1}{6}h_i[\mathbf{f}(x_i,\mathbf{y}_i) + 4\mathbf{f}(x_{i+1/2},\mathbf{y}_{i+1/2}) + \mathbf{f}(x_{i+1},\mathbf{y}_{i+1})], \\ \text{kde}\quad &x_{i+1/2} = x_i + \tfrac{1}{2}h_i, \quad \mathbf{y}_{i+1/2} = \tfrac{1}{2}(\mathbf{y}_i + \mathbf{y}_{i+1}) + \tfrac{1}{8}h_i[\mathbf{f}(x_i,\mathbf{y}_i) - \mathbf{f}(x_{i+1},\mathbf{y}_{i+1})]\,, \end{aligned} \tag{9.9}$$

Simpsonova formule (9.9) je řádu 4, tj. pro chybu platí

$$\mathbf{y}(x_i) - \mathbf{y}_i = O(h^4)\,.$$

Další informace o Simpsonově formuli (9.9) lze načerpat v [51], [26] a také v [32].

V případě lineární úlohy (9.7) řešíme soustavu lineárních rovnic (9.8), matice $\mathbf{R}_i$, $\mathbf{S}_i$ a vektory $\mathbf{v}_i$ získáme z formule (9.9), v níž klademe $\mathbf{f}(x,\mathbf{y}) = \mathbf{A}(x)\mathbf{y} + \mathbf{q}(x)$. Odvození vzorců pro $\mathbf{R}_i$, $\mathbf{S}_i$ a $\mathbf{v}_i$ ponecháváme čtenáři jako cvičení.

9.2. Diferenční metoda

Ve zbytku kapitoly 9 se budeme převážně zabývat lineární ODR2

$$-\,[p(x)u']' + q(x)u = f(x), \qquad x \in (0,\ell)\,. \tag{9.10}$$

Předpokládejme, že funkce p, p', q i f jsou spojité, dále že $p(x) \geq p_0 > 0$, $q(x) \geq 0$, a uvažujme nejdříve jednoduché okrajové podmínky

$$u(0) = g_0\,, \tag{9.11a}$$

$$u(\ell) = g_\ell\,. \tag{9.11b}$$

Za uvedených předpokladů má úloha (9.10) – (9.11) jediné řešení.

Úloha (9.10) – (9.11) může popisovat například problém *tahu–tlaku prutu*, tedy prutu namáhaného pouze tahem popřípadě tlakem. V tom případě je u posunutí střednicové čáry prutu, $p = EA$, kde E je Youngův modul pružnosti a A je plocha průřezu prutu, q je měrný odpor podloží, na němž prut spočívá, f je intenzita zatížení a g_0 a g_ℓ jsou předepsaná posunutí koncových bodů prutu.

Jinou aplikací, popsanou stejnou rovnicí a stejnými okrajovými podmínkami, je například *stacionární úloha vedení tepla v tyči*. Pak u je teplota, p je koeficient tepelné vodivosti, $f - qu$ je intenzita tepelných zdrojů, g_0 a g_ℓ jsou teploty koncových bodů tyče.

Úlohu (9.10) – (9.11) lze přeformulovat do tvaru (9.1)–(9.2), stačí položit

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} u \\ pu' \end{pmatrix}, \qquad \mathbf{f} = \begin{pmatrix} y_2/p \\ qy_1 - f \end{pmatrix}, \qquad \mathbf{r} = \begin{pmatrix} y_1(0) - g_0 \\ y_1(\ell) - g_\ell \end{pmatrix}, \tag{9.12}$$

a následně řešit metodou střelby, viz kapitola 9.1. My si ale vysvětlíme jinou techniku, známou jako *diferenční metoda* (stručně FDM podle anglického *finite difference method*).

FDM je klasická diskretizační metoda, zevrubně popsaná a analyzovaná v celé řadě publikací, viz např. [29], [59].

Nechť $0 = x_0 < x_1 < \cdots < x_N = \ell$ je dělení intervalu $\langle 0, \ell \rangle$ a $h_i = x_i - x_{i-1}$ je délka kroku. Body x_i nazýváme *uzly* a množinu $\{x_i\}_{i=0}^N$ uzlů nazýváme *sítí*. Pro jednoduchost se omezíme na ekvidistantní dělení, takže $h_i = h = \ell/N$ a $x_i = ih$, $i = 0, 1, \ldots, N$.

Splnění rovnice (9.10) budeme vyžadovat pouze ve vnitřních uzlech sítě, tj.

$$-(pu')'\big|_{x=x_i} + q_i u(x_i) = f_i, \quad i = 1, 2, \ldots, N-1\,, \tag{9.13}$$

kde $q_i = q(x_i)$ a $f_i = f(x_i)$. Člen $-(pu')'\big|_{x=x_i}$ nahradíme *diferenčním podílem*. Pomocí označení

$$x_{i-1/2} = x_i - \tfrac{1}{2}h\,,\ x_{i+1/2} = x_i + \tfrac{1}{2}h\,,\ p_{i-1/2} = p(x_{i-1/2})\,,\ p_{i+1/2} = p(x_{i+1/2})$$

lze přibližně položit

$$\begin{aligned} -(pu')'\big|_{x=x_i} &\doteq -\frac{pu'\big|_{x=x_{i+1/2}} - pu'\big|_{x=x_{i-1/2}}}{h} = -\frac{p_{i+1/2}u'(x_{i+1/2}) - p_{i-1/2}u'(x_{i-1/2})}{h} \\ &\doteq -\frac{p_{i+1/2}\dfrac{u(x_{i+1}) - u(x_i)}{h} - p_{i-1/2}\dfrac{u(x_i) - u(x_{i-1})}{h}}{h}\,. \end{aligned}$$

Užitím Taylorova rozvoje snadno ověříme, že tato aproximace je řádu $O(h^2)$, tj. že platí

$$-(pu')'\big|_{x=x_i} = \frac{-p_{i-1/2}u(x_{i-1}) + (p_{i-1/2} + p_{i+1/2})u(x_i) - p_{i+1/2}u(x_{i+1})}{h^2} + O(h^2)\,. \tag{9.14}$$

Po zanedbání chyby $O(h^2)$ tak z (9.14) a (9.13) dostáváme pro přibližné řešení $u_i \approx u(x_i)$ soustavu rovnic

$$\frac{-p_{i-1/2}u_{i-1} + (p_{i-1/2} + p_{i+1/2} + h^2 q_i)u_i - p_{i+1/2}u_{i+1}}{h^2} = f_i\,, \tag{9.15}$$

pro $i = 1, 2, \ldots, N-1$. Z okrajových podmínek máme

$$u_0 = g_0\,, \tag{9.16a}$$

$$u_N = g_\ell\,. \tag{9.16b}$$

To nám umožní dosadit do první rovnice $u_0 = g_0$ a člen $-p_{1/2}g_0/h^2$ převést na pravou stranu. Podobně v poslední rovnici položíme $u_N = g_\ell$ a člen $-p_{N-1/2}g_\ell/h^2$ převedeme na pravou stranu. Matice takto upravené soustavy rovnic (9.15) je třídiagonální, pozitivně definitní, diagonálně dominantní (pro $q_i > 0$ ryze). Tyto vlastnosti zaručují, že matice soustavy je regulární a soustava rovnic má jediné řešení.

Jsou-li funkce p, q a f dostatečně hladké, pak pro chybu platí

$$u(x_i) - u_i = O(h^2)\,, \tag{9.17}$$

přesná formulace a příslušný důkaz viz [59].

Okrajové podmínky s derivací. Okrajové podmínky (9.11) se nazývají *Dirichletovy*. V aplikacích se objevují ještě další typy okrajových podmínek. Podmínky

$$p(0)u'(0) = \alpha_0 u(0) - \beta_0, \tag{9.18a}$$

$$-p(\ell)u'(\ell) = \alpha_\ell u(\ell) - \beta_\ell \tag{9.18b}$$

se nazývají *Newtonovy* nebo také *Robinovy*. Pokud $\alpha_0 = 0$ resp. $\alpha_\ell = 0$, hovoříme o *Neumannově* okrajové podmínce. Aby byla zaručena jednoznačná existence řešení, předpokládejme $\alpha_0 \geq 0$, $\alpha_\ell \geq 0$, a pokud uvažujeme okrajové podmínky (9.18a) a (9.18b), pak předpokládejme navíc buďto $\alpha_0 > 0$ nebo $\alpha_\ell > 0$ nebo $q(x) \geq q_0 > 0$ alespoň na části intervalu $(0, \ell)$. Pokud $\alpha_0 = \alpha_\ell = 0$ a $q(x) = 0$ v $(0, \ell)$, pak úloha (9.10) – (9.18) buďto nemá řešení nebo má nekonečně mnoho řešení. Skutečně, integrací (9.10) a užitím (9.18) dostaneme nutnou podmínku existence řešení, tzv. *podmínku rovnováhy*

$$\int_0^\ell [f + (pu')'] \, \mathrm{d}x = \int_0^\ell f \, \mathrm{d}x + pu'\big|_{x=0}^{x=\ell} = \int_0^\ell f \, \mathrm{d}x + \beta_0 + \beta_\ell = 0 \,.$$

Pokud podmínka rovnováhy splněna není, řešení neexistuje. Je-li však podmínka rovnováhy splněna a u je řešení, pak je řešením také funkce $u + C$, kde C je libovolná konstanta.

Interpretujeme-li Newtonovy okrajové podmínky v úloze tahu–tlaku prutu, je pu' normálová síla, α_0, α_ℓ jsou tuhosti pružin v koncových bodech prutu a β_0, β_ℓ jsou zadané síly působící na koncích prutu. V úloze stacionárního vedení tepla je pu' tepelný tok, α_0, α_ℓ jsou koeficienty přestupu tepla a β_0, β_ℓ jsou zadané tepelné toky na okrajích tyče. Tepelné toky se často uvažují ve tvaru $\beta_0 = \alpha_0 u_0^e$, $\beta_\ell = \alpha_\ell u_\ell^e$, kde u_0^e resp. u_ℓ^e je vnější teplota okolo levého resp. pravého konce tyče.

V případě zadané okrajové podmínky (9.18a) eventuálně (9.18b) musíme sestavit rovnici umožňující výpočet neznámé u_0 eventuálně u_N. Ukažme si to třeba pro případ předepsané okrajové podmínky (9.18a).

Pomůžeme si tak, že budeme požadovat, aby rovnice (9.10) platila také v levém krajním uzlu $x_0 = 0$, tj. aby rovnice (9.13) byla splněna rovněž pro $i = 0$.

Člen $-(pu')'\big|_{x=x_0}$ vyjádříme nejdříve pomocí jednostranné diference a pak pomocí centrální diference a vztahu (9.18a), tj.

$$-(pu')'\big|_{x=x_0} = -\frac{pu'\big|_{x=x_{1/2}} - pu'\big|_{x=x_0}}{\frac{1}{2}h} + O(h) =$$

$$\frac{-p_{1/2}\dfrac{u(x_1) - u(x_0)}{h} + \alpha_0 u(x_0) - \beta_0}{\frac{1}{2}h} + O(h) \,.$$

Zanedbáme-li chyby, dostaneme rovnici pro neznámou u_0

$$\frac{(p_{1/2} + h\alpha_0 + \frac{1}{2}h^2 q_0)u_0 - p_{1/2}u_1}{h^2} = \frac{1}{h}\beta_0 + \frac{1}{2}f_0 \,. \tag{9.19a}$$

V případě okrajové podmínky (9.18b) postupujeme obdobně, tj. požadujeme splnění rovnice (9.13) také pro $i = N$, a po úpravách obdržíme rovnici pro neznámou u_N

$$\frac{-p_{N-1/2}u_{N-1} + (p_{N-1/2} + h\alpha_\ell + \frac{1}{2}h^2 q_N)u_N}{h^2} = \frac{1}{h}\beta_\ell + \frac{1}{2}f_N \,. \tag{9.19b}$$

Je-li předepsána okrajová podmínka (9.18a), zapíšeme jako první rovnici (9.19a), pak následují rovnice (9.15) pro $i = 1, 2, \ldots, N-1$, a je-li předepsána okrajová podmínka (9.18b), připojíme jako poslední rovnici (9.19b). Jsou-li předepsány Dirichletovy okrajové podmínky, dosadíme g_0 za u_0 resp. g_ℓ za u_N a příslušné členy převedeme na pravou stranu. Matice výsledné soustavy rovnic je opět třídiagonální, pozitivně definitní a diagonálně dominantní. I když se při odvození rovnic (9.19) dopouštíme chyby řádu $O(h)$, pro chybu přibližného řešení platí zase vztah (9.17).

Rovnice s konvekčním členem. V aplikacích často vzniká potřeba řešit poněkud obecnější rovnici

$$-\left[p(x)u' - r(x)u\right]' + q(x)u = f(x), \qquad x \in (0, \ell)\,. \tag{9.20}$$

Tato rovnice popisuje například *transport chemické příměsi v tekutině*. V tom případě je u koncentrace příměsi v tekutině, $r = \varrho v$, kde ϱ je hustota tekutiny a v její rychlost, p je koeficient difúze a $f - qu$ je intenzita objemového zdroje příměsi. Rovnici (9.20) lze použít také k popisu *teplotního pole v tekutině*. Pak u je teplota, $r = c\varrho v$, kde c je tepelná kapacita, ϱ hustota a v rychlost tekutiny, p je tepelná vodivost a $f - qu$ je intenzita vnitřních tepelných zdrojů. S přihlédnutím k typické fyzikální interpretaci říkáme, že $-(pu')'$ je *difúzní* člen, $(ru)'$ je *konvekční člen* a $f - qu$ je *zdrojový člen*.

Pokud jde o okrajové podmínky, budeme postupovat takto: „na vtoku“, tj. bodě $x = 0$ pro $r(0) > 0$ resp. v bodě $x = \ell$ pro $r(\ell) < 0$, můžeme předpokládat, že veličinu $u(x)$ známe, a proto její hodnotu předepíšeme prostřednictvím Dirichletovy okrajové podmínky. „Na výtoku“, tj. bodě $x = 0$ pro $r(0) \le 0$ resp. v bodě $x = \ell$ pro $r(\ell) \ge 0$, můžeme zadat jak Dirichletovu tak Newtonovu okrajovou podmínku.

V [23] je dokázáno, že rovnice (9.20) doplněná o okrajové podmínky má jediné řešení třeba tehdy, když kromě dřívějších předpokladů navíc platí

$$r'(x) \ge 0, \quad \alpha_0 - \tfrac{1}{2}r(0) \ge 0, \quad \alpha_\ell + \tfrac{1}{2}r(\ell) \ge 0.$$

Jsou-li na obou okrajích $x = 0$ i $x = \ell$ předepsány Dirichletovy okrajové podmínky, stačí předpokládat

$$r'(x) \ge 0, \quad x \in \langle 0, \ell \rangle. \tag{9.21}$$

Zadáme-li Newtonovu okrajovou podmínku jen na výtoku, pak podmínka $\alpha_0 - \frac{1}{2}r(0) \ge 0$ resp. $\alpha_\ell + \frac{1}{2}r(\ell) \ge 0$ zřejmě platí, takže opět stačí předpokládat jen (9.21).

Je-li $r(0) \le 0$ a $r(\ell) \le 0$, pak levý okraj $x = 0$ je výtok a pravý okraj $x = \ell$ je vtok. Je-li $r(0) \le 0$ a $r(\ell) > 0$, pak oba okraje představují výtok. Je-li $r(0) > 0$, pak podle (9.21) také $r(\ell) > 0$, takže $x = 0$ je vtok a $x = \ell$ je výtok. Oba konce nemohou být vtokem: $r(0) > 0$ a $r(\ell) < 0$ není podle (9.21) možné.

V dynamice tekutin se ukazuje, že v nestlačitelné tekutině rychlost $\mathbf{v} = (v_1, v_2, v_3)^T$ splňuje *rovnici kontinuity* $\operatorname{div} \mathbf{v} = 0$. V jedné dimenzi tedy $v' = 0$, takže v je konstanta. Volba konstantní funkce r tedy představuje fyzikálně opodstatněnou možnost. Podmínka (9.21) je pro konstantní funkci r triviálně splněna.

Věnujme se tedy diskretizaci. Konvekční člen vyjádříme ve vnitřních uzlech pomocí centrální diference

$$(ru)'\big|_{x=x_i} = \frac{ru|_{x_{i+1/2}} - ru|_{x_{i-1/2}}}{h} + O(h^2) \tag{9.22}$$

a v koncových uzlech pomocí jednostranné diference

$$\begin{aligned} (ru)'\big|_{x=x_0} &= \frac{ru|_{x_{1/2}} - ru|_{x_0}}{\frac{1}{2}h} + O(h), \\ (ru)'\big|_{x=x_N} &= \frac{ru|_{x_N} - ru|_{x_{N-1/2}}}{\frac{1}{2}h} + O(h)\,. \end{aligned} \tag{9.23}$$

V (9.22) a (9.23) vyjádříme konvekční tok ru ve středech $x_{i+1/2}$ úseček $\langle x_i, x_{i+1}\rangle$ interpolací jako aritmetický průměr hodnot u v koncových bodech,

$$ru|_{x_{i+1/2}} = \tfrac{1}{2} r_{i+1/2}[u(x_i) + u(x_{i+1})] + O(h^2), \qquad i = 0, 1, \ldots, N-1. \tag{9.24}$$

Po zanedbání chybových členů v (9.22)–(9.24), obdržíme soustavu rovnic, jejíž tvar vyjádříme prostřednictvím dříve odvozených rovnic (9.15) a (9.19) takto: na levou stranu rovnice

$$\left.\begin{matrix}(9.19a)\\(9.15)\\(9.19b)\end{matrix}\right\} \quad \text{přidáme člen} \quad \begin{cases} \left[\frac{1}{2} r_{1/2}(u_0 + u_1) - r_0 u_0\right]/h\,, \\ \frac{1}{2}[r_{i+1/2}(u_i + u_{i+1}) - r_{i-1/2}(u_{i-1} + u_i)]/h\,, \\ \left[r_N u_N - \frac{1}{2} r_{N-1/2}(u_{N-1} + u_N)\right]/h. \end{cases} \tag{9.25}$$

Matice takto vzniklé soustavy rovnic je třídiagonální a nesymetrická. Dá se ukázat, že když

$$\tfrac{1}{2}h|r_0| < p_{1/2}, \quad \tfrac{1}{2}h|r_{i+1/2}| < p_{i+1/2}, \ \ i = 0, 1, \ldots, N-1, \quad \tfrac{1}{2}h|r_N| < p_{N-1/2}, \tag{9.26}$$

pak je matice soustavy regulární. Pro dostatečně jemné dělení intervalu $\langle 0, \ell\rangle$ bude podmínka (9.26) jistě splněna. Pro chybu opět platí (9.17).

Dominantní konvekce. Podmínka (9.26) může být značně omezující v případě, kdy konvekční koeficient výrazně převažuje nad koeficientem difúzním, tedy pro $|r| \gg p$. Pak je účelné vyjádřit konvekční tok ru ve středech $x_{i+1/2}$ úseček $\langle x_i, x_{i+1}\rangle$ takto:

$$ru|_{x_{i+1/2}} = \begin{cases} r_{i+1/2} u(x_i) + O(h) & \text{pro } r_{i+1/2} \geq 0, \\ r_{i+1/2} u(x_{i+1}) + O(h) & \text{pro } r_{i+1/2} < 0. \end{cases} \tag{9.27}$$

Aproximace (9.27) je z fyzikálního hlediska přirozená: informaci o řešení u v uzlu $x_{i+1/2}$ čerpáme ze znalosti řešení proti „proudu“, proti „větru“: pro $r_{i+1/2} > 0$ „fouká zleva“, proto použijeme hodnotu $u(x_i)$ v uzlu x_i ležícím nalevo od bodu $x_{i+1/2}$, pro $r_{i+1/2} < 0$ „fouká zprava“ a proto použijeme hodnotu $u(x_{i+1})$ v uzlu x_{i+1} ležícím napravo od bodu

$x_{i+1/2}$. Jednostrannou aproximaci konvekčního toku podle (9.27) nazýváme *upwind aproximací*. Pomocí označení

$$a^+ = \max(a,0), \qquad a^- = \min(a,0), \quad \text{kde } a \text{ je libovolné číslo,}$$

lze (9.27) zapsat v kompaktním tvaru

$$ru|_{x_{i+1/2}} = r^+_{i+1/2}u(x_i) + r^-_{i+1/2}u(x_{i+1}) + O(h), \quad i = 0,1,\ldots,N-1. \tag{9.28}$$

Po zanedbání chybových členů v (9.22), (9.23) a (9.28) obdržíme výslednou soustavu rovnic, jejíž tvar vyjádříme prostřednictvím dříve odvozených rovnic (9.15) a (9.19) takto: na levou stranu rovnice

$$\left.\begin{array}{l}(9.19a)\\(9.15)\\(9.19b)\end{array}\right\} \text{ přidáme člen } \left\{\begin{array}{l}[(r^+_{1/2}-r_0)u_0 + r^-_{1/2}u_1]/h,\\ [r^+_{i+1/2}u_i + r^-_{i+1/2}u_{i+1}]/h - [r^+_{i-1/2}u_{i-1} + r^-_{i-1/2}u_i]/h,\\ [-r^+_{N-1/2}u_{N-1} + (r_N - r^-_{N-1/2})u_N]/h\,.\end{array}\right. \tag{9.29}$$

Matice takto vzniklé soustavy je regulární nezávisle na jemnosti dělení intervalu $\langle 0,\ell\rangle$. Pro chybu však platí jen

$$u(x_i) - u_i = O(h)\,. \tag{9.30}$$

Pro dosažení přesnosti řádu $O(h^2)$ je třeba používat přesnější upwind aproximaci konvekčního toku, třeba

$$ru|_{x_{i+1/2}} = r^+_{i+1/2}\left[\tfrac{3}{2}u(x_i) - \tfrac{1}{2}u(x_{i-1})\right] + r^-_{i+1/2}\left[\tfrac{3}{2}u(x_{i+1}) - \tfrac{1}{2}u(x_{i+2})\right] + O(h^2), \tag{9.28'}$$

$i = 0,1,\ldots,N-1$, kde $u(x_{-1}) := 2u(x_0) - u(x_1)$, $u(x_{N+1}) := 2u(x_N) - u(x_{N-1})$.

Příklad 9.1. Zabývejme se řešením modelové úlohy

$$-\varepsilon u'' + u' = 0 \quad \text{pro} \quad x \in (0,1), \quad u(0) = 0,\ u(1) = 1,$$

kde $\varepsilon > 0$ je konstanta. Přesné řešení

$$u(x) = \frac{1 - e^{x/\varepsilon}}{1 - e^{1/\varepsilon}}$$

je rostoucí funkce.

Diskretizací konvekčního členu pomocí centrální diference (9.25_2), tj. pomocí druhého vztahu v (9.25), dostaneme rovnici

$$-\varepsilon\,\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \frac{u_{i+1} - u_{i-1}}{2h} = 0\,,$$

kterou lze při označení $\kappa := h/\varepsilon$ zapsat ekvivalentně ve tvaru

$$-(1 + \tfrac{1}{2}\kappa)u_{i-1} + 2u_i - (1 - \tfrac{1}{2}\kappa)u_{i+1} = 0\,.$$

Pro $\kappa = 2$ dostaneme $u_i = u_{i-1}$, takže řešení je $u_i = 0$ pro $i < N$ a $u_N = 1$. Pro $\kappa \neq 2$ snadným výpočtem ověříme, že diferenční rovnici vyhovuje řešení

$$u_i = C_1 + C_2 \left[\frac{2+\kappa}{2-\kappa}\right]^i,$$

kde C_1 a C_2 jsou konstanty, které určíme z okrajových podmínek. Pro $\kappa > 2$ přibližné řešení u_i osciluje okolo C_1, což je v rozporu s chováním přesného řešení u, rostoucí posloupnost $\{u_i\}_{i=0}^N$ dostaneme jen pro $|\kappa| < 2$, což je podmínka (9.26) pro $p = \varepsilon$, $r = 1$.

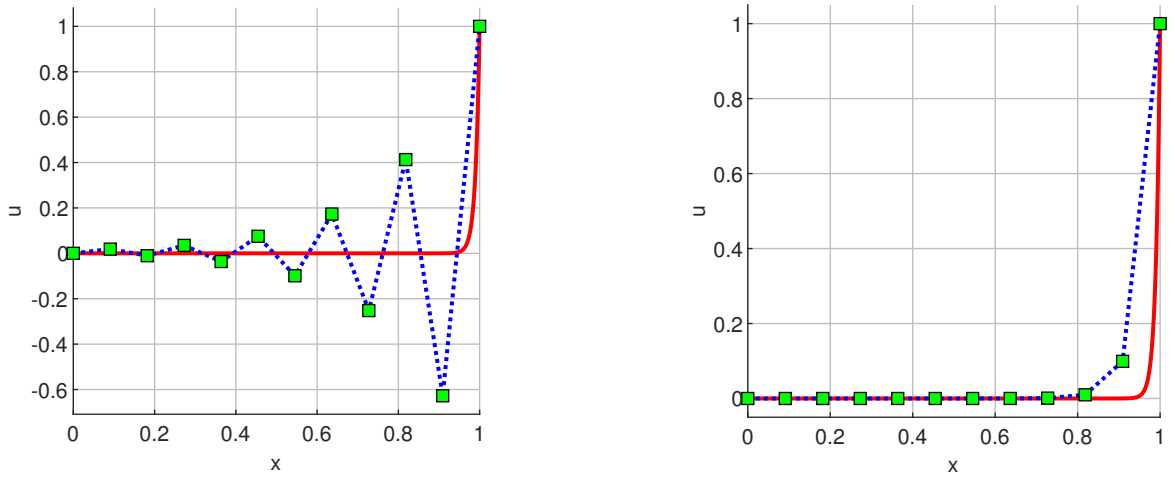
Naši modelovou úlohu vyřešíme také upwind technikou. Konvekční člen nahradíme levostrannou diferencí podle (9.29_2) a dostaneme rovnici

$$-\varepsilon \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \frac{u_i - u_{i-1}}{h} = 0.$$

Diferenční rovnici vyhovuje řešení

$$u_i = C_1 + C_2(1+\kappa)^i,$$

které je rostoucí nezávisle na velikosti κ. □



Obr. 9.1. $\varepsilon = \frac{1}{100}$, $N = 11$, vlevo centrální diference, vpravo upwind aproximace

9.3. Metoda konečných objemů

(stručně FVM podle anglického *finite volume method*) se používá především pro řešení problémů proudění ve více prostorových proměnných, např. viz [57], [16]. Princip této metody však lze objasnit i pro jednodimenzionální úlohu popsanou diferenciální rovnicí (9.20) a okrajovými podmínkami (9.11) a (9.0).

Ke každému uzlu x_i přiřadíme *konečný objem* B_i (stručně *buňku*) takto: pro vnitřní uzly $B_i = \langle x_{i-1/2}, x_{i+1/2}\rangle$, $i = 1, 2, \ldots, N-1$, a pro koncové uzly $B_0 = \langle 0, x_{1/2}\rangle$,

$B_N = \langle x_{N-1/2}, \ell\rangle$. Zřejmě $\langle 0, \ell\rangle = \bigcup_{i=0}^{N} B_i$. Integrací rovnice (9.20) přes buňku B_i dostaneme *bilanční rovnici*

$$\int_{B_i} -[pu' - ru]' \,\mathrm{d}x + \int_{B_i} qu \,\mathrm{d}x = \int_{B_i} f \,\mathrm{d}x\,. \tag{9.31}$$

Předpokládejme nejdříve, že B_i přísluší vnitřnímu uzlu. Pak dostaneme

$$-[pu' - ru]_{x=x_{i-1/2}}^{x=x_{i+1/2}} + \int_{x_{i-1/2}}^{x_{i+1/2}} qu \,\mathrm{d}x = \int_{x_{i-1/2}}^{x_{i+1/2}} f \,\mathrm{d}x\,. \tag{9.32}$$

Difúzní tok aproximujeme pomocí centrální diference,

$$p_{i-1/2}u'(x_{i-1/2}) = p_{i-1/2}\frac{u(x_i) - u(x_{i-1})}{h} + O(h^2), \quad i = 1, 2, \ldots, N-1, \tag{9.33}$$

a konvekční tok aproximujeme pomocí centrální aproximace (9.24) resp. upwind aproximace (9.28). Integrály v rovnici (9.32) spočteme obdélníkovou formulí,

$$\int_{x_{i-1/2}}^{x_{i+1/2}} qu \,\mathrm{d}x = hq_i u(x_i) + O(h^3)\,, \qquad \int_{x_{i-1/2}}^{x_{i+1/2}} f \,\mathrm{d}x = hf_i + O(h^3).$$

Zanedbáme-li chyby, dostaneme aproximaci bilanční rovnice (9.32) ve tvaru (9.25_2) resp. (9.29_2).

Je-li v koncovém bodě $x = 0$ resp. $x = \ell$ předepsána Newtonova okrajová podmínka (9.18a) resp. (9.18b), je třeba uvážit bilanční rovnici (9.31) také pro buňku B_0 resp. B_N. Tak například pro buňku B_0 máme

$$-(pu' - ru)\big|_{x=x_0}^{x=x_{1/2}} + \int_{x_0}^{x_{1/2}} qu \,\mathrm{d}x = \int_{x_0}^{x_{1/2}} f \,\mathrm{d}x\,.$$

Difúzní tok v bodě $x_{1/2}$ aproximujeme centrální diferencí (9.33), konvekční tok v bodě $x_{1/2}$ aproximujeme pomocí (9.24) resp. (9.28), člen $p(0)u'(0)$ vyjádříme pomocí okrajové podmínky (9.18a) a integrály spočteme levostrannou obdélníkovou formulí,

$$\int_{x_0}^{x_{1/2}} qu \,\mathrm{d}x = \tfrac{1}{2}hq_0 u(x_0) + O(h^2)\,, \qquad \int_{x_0}^{x_{1/2}} f \,\mathrm{d}x = \tfrac{1}{2}hf_0 + O(h^2)\,.$$

Zanedbáme-li chyby, dostaneme rovnici (9.25_1) resp. (9.29_1). Rovnici (9.25_3) resp. (9.29_3) odvodíme obdobně z bilanční rovnice (9.31) zapsané pro buňku B_N.

Metodou konečných objemů jsme tedy dostali stejné rovnice jako rovnice odvozené metodou diferenční. Přednosti metody konečných objemů ve srovnání s metodou diferenční vyniknou až při řešení parciálních diferenciálních rovnic ve více prostorových proměnných.

9.4. Metoda konečných prvků

Nejuniverzálnější metodou diskretizace okrajových úloh je *metoda konečných prvků* (stručně FEM podle anglického *finite element method*, česky MKP). V úlohách mechaniky

pevné fáze je to jednoznačně nejpoužívanější metoda. I když přednosti MKP lze plně ocenit teprve u úloh ve dvou a třech prostorových proměnných, podstatu metody lze objasnit i na jednodimenzionální úloze. Z řady publikací věnovaných MKP zmiňme např. [18], [61], [5].

Východiskem pro diskretizaci metodou konečných prvků je slabá formulace okrajové úlohy. Proto si teď ukážeme, jak z *klasické formulace* (9.20), (9.11), (9.18) formulaci slabou dostaneme. Nejdříve zavedeme následující

Označení. Symbolem $C\langle 0,\ell\rangle$ budeme značit prostor všech funkcí, které jsou v intervalu $\langle 0,\ell\rangle$ spojité, a symbolem $C^k\langle 0,\ell\rangle$ pak prostor všech funkcí, které jsou v intervalu $\langle 0,\ell\rangle$ spojité spolu se svými derivacemi až do řádu k včetně (C podle anglického *continuous*).

Říkáme, že bod a je pro funkci $f(x)$ *bodem nespojitosti prvního druhu*, existuje-li v a konečná limita zprava i zleva [označme tyto limity $f(a+0)$ resp. $f(a-0)$] a je-li $f(a+0)\neq f(a-0)$.

Funkce $f(x)$ definovaná v intervalu $\langle 0,\ell\rangle$ se nazývá *po částech spojitá v intervalu* $\langle 0,\ell\rangle$, je-li v $\langle 0,\ell\rangle$ spojitá s výjimkou konečného počtu bodů, v nichž má nespojitost prvního druhu.

Prostor funkcí po částech spojitých v intervalu $\langle 0,\ell\rangle$ označíme $PC\langle 0,\ell\rangle$ (PC podle anglického *piecewise continuous*). Symbolem $PC^k\langle 0,\ell\rangle$ značíme prostor funkcí, které jsou v intervalu $\langle 0,\ell\rangle$ spojité spolu se svými derivacemi až do řádu $k-1$ včetně, a jejichž k-tá derivace je v intervalu $\langle 0,\ell\rangle$ po částech spojitá.

V dalším budeme používat zejména prostor $C^1\langle 0,\ell\rangle$ funkcí, které jsou v intervalu $\langle 0,\ell\rangle$ spojité spolu se svou první derivací, a prostor $PC^1\langle 0,\ell\rangle$ funkcí, které jsou v intervalu $\langle 0,\ell\rangle$ spojité a mají v něm po částech spojitou první derivaci.

Slabá formulace. Začneme tím, že zavedeme pojem *testovací funkce*: funkci $v\in C^1\langle 0,\ell\rangle$ nazveme testovací, jestliže $v=0$ v tom krajním bodě intervalu $\langle 0,\ell\rangle$, v němž je předepsána Dirichletova okrajová podmínka. Pro konkrétnost se omezíme na okrajové podmínky (9.11a) a (9.18b), takže testovací funkce v splňuje $v(0)=0$. Násobme rovnici (9.10) testovací funkcí v a integrujme přes $\langle 0,\ell\rangle$. Integrací per-partes členu $\int_0^\ell[-(pu'-ru)']v\,\mathrm{d}x$ a následným užitím okrajové podmínky (9.18b) a vztahu $v(0)=0$ obdržíme

$$\int_0^\ell fv\,\mathrm{d}x=\int_0^\ell\left[-(pu'-ru)'+qu\right]v\,\mathrm{d}x=-(pu'-ru)v\big|_{x=0}^{x=\ell}+$$

$$+\int_0^\ell\left[(pu'-ru)v'+quv\right]\,\mathrm{d}x=\left[\alpha_\ell u(\ell)-\beta_\ell+r(\ell)u(\ell)\right]v(\ell)+\int_0^\ell\left[(pu'-ru)v'+quv\right]\,\mathrm{d}x.$$

Odvodili jsme tedy, že řešení u úlohy (9.10), (9.11a) a (9.18b) musí splňovat kromě Dirichletovy okrajové podmínky $u(0)=g_0$ také rovnost

$$\int_0^\ell\left[(pu'-ru)v'+quv\right]\,\mathrm{d}x+[\alpha_\ell+r(\ell)]u(\ell)v(\ell)=\int_0^\ell fv\,\mathrm{d}x+\beta_\ell v(\ell)\tag{9.34}$$

pro každou funkci $v\in C^1\langle 0,\ell\rangle$, $v(0)=0$. Okrajová podmínka (9.18b) Newtonova typu, která se stala součástí integrální rovnice (9.34) a je tak automaticky splněna, se nazývá *přirozenou okrajovou podmínkou*. Dirichletovu okrajovou podmínku (9.11a), která součástí

rovnice (9.34) není a jejíž explicitní splnění proto musíme vyžadovat, nazýváme *podstatnou* nebo také *hlavní okrajovou podmínkou*. Rovnice (9.34) je dobře definována i v případě, kdy funkce u a v jsou z prostoru $X \equiv PC^1\langle 0,\ell\rangle$. Testovací funkce pak volíme z *prostoru* $V = \{v \in X |\, v(0) = 0\}$ *testovacích funkcí* a řešení u z *množiny* $W = \{v \in X |\, v(0) = g_0\}$ *přípustných řešení*. Dále označíme

$$\begin{aligned} a(u,v) &= \int_0^\ell [(pu' - ru)v' + quv]\,\mathrm{d}x + [\alpha_\ell + r(\ell)]u(\ell)v(\ell), \\ L(v) &= \int_0^\ell fv\,\mathrm{d}x + \beta_\ell v(\ell). \end{aligned} \tag{9.35}$$

Pak úlohu

$$\text{najít } u \in W \text{ splňující } a(u,v) = L(v) \quad \forall v \in V \tag{9.36}$$

nazýváme *slabou formulací* problému (9.10), (9.11a), (9.18b). Řešení úlohy (9.36) nazveme *slabým řešením*. Slabá formulace je obecnější než formulace klasická, neboť klade nižší nároky na hladkost dat:

$$\text{klasická formulace:} \quad p, r \in C^1\langle 0,\ell\rangle, q, f \in C\langle 0,\ell\rangle\,, \tag{9.37a}$$

$$\text{slabá formulace:} \quad p, r, q, f \in PC\langle 0,\ell\rangle\,. \tag{9.37b}$$

Jestliže $p, r', q, f \in PC\langle 0,\ell\rangle$, $p \geq p_0 > 0$, $q + \frac{1}{2}r' \geq 0$, $\alpha_\ell + \frac{1}{2}r(\ell) \geq 0$, pak úloha (9.36) má jediné slabé řešení, viz [23].

Ukázali jsme si, že klasické řešení je vždy také řešení slabé, viz odvození rovnice (9.34). Opak obecně neplatí, tj. slabé řešení nemusí být řešení klasické. Jsou-li však funkce p, r, q a f dostatečně hladké, konkrétně platí-li podmínky (9.37a), pak lze dokázat, že slabé řešení $u \in C^2\langle 0,\ell\rangle$ je řešení klasické.

Slabá formulace má v úloze tahu–tlaku prutu (kdy $r = 0$) význam *principu virtuálních posunutí* a samotné testovací funkce $v \in V$ mají význam virtuálních posunutí δu přípustných řešení $u \in W$. Slabá formulace je tedy zcela přirozená, neboť konkrétně pro úlohu tahu–tlaku prutu popisuje základní fyzikální zákon mechaniky kontinua.

Slabá formulace pro všechny kombinace okrajových podmínek. Uveďme si tvar V, W, $a(u,v)$ a $L(v)$ pro všechny možné kombinace okrajových podmínek.

(DD) Okrajové podmínky (9.11a), (9.11b)

$$V = \{v \in X \,|\, v(0) = v(\ell) = 0\}, \quad W = \{v \in X \,|\, v(0) = g_0,\, v(\ell) = g_\ell\}, \tag{9.38-DD}$$

$$a(u,v) = \int_0^\ell [(pu' - ru)v' + quv]\,\mathrm{d}x, \quad L(v) = \int_0^\ell fv\,\mathrm{d}x\,.$$

(DN) Okrajové podmínky (9.11a), (9.18b)

$$V = \{v \in X \,|\, v(0) = 0\}, \quad W = \{v \in X \,|\, v(0) = g_0\}, \tag{9.38-DN}$$

$$a(u,v) = \int_0^\ell [(pu' - ru)v' + quv]\,\mathrm{d}x + [\alpha_\ell + r(\ell)]u(\ell)v(\ell), \quad L(v) = \int_0^\ell fv\,\mathrm{d}x + \beta_\ell v(\ell)\,.$$

(ND) Okrajové podmínky (9.18a), (9.11b)

$$V = \{v \in X \,|\, v(\ell) = 0\}, \quad W = \{v \in X \,|\, v(\ell) = g_\ell\}, \tag{9.38-ND}$$

$$a(u,v) = \int_0^\ell [(pu' - ru)v' + quv]\,\mathrm{d}x + [\alpha_0 - r(0)]u(0)v(0), \quad L(v) = \int_0^\ell fv\,\mathrm{d}x + \beta_0 v(0)\,.$$

(NN) Okrajové podmínky (9.18a), (9.18b)

$$V = X, \quad W = X, \tag{9.38-NN}$$

$$a(u,v) = \int_0^\ell [(pu' - ru)v' + quv]\,\mathrm{d}x + [\alpha_0 - r(0)]u(0)v(0) + [\alpha_\ell + r(\ell)]u(\ell)v(\ell),$$

$$L(v) = \int_0^\ell fv\,\mathrm{d}x + \beta_0 v(0) + \beta_\ell v(\ell)\,.$$

Ve všech čtyřech případech je zaručena jednoznačná existence slabého řešení úlohy (9.36), pokud $p, r', q, f \in PC\langle 0,\ell\rangle$, $p \geq p_0 > 0$, $q + \frac{1}{2}r' \geq 0$, $\alpha_0 - \frac{1}{2}r(0) \geq 0$, $\alpha_\ell + \frac{1}{2}r(\ell) \geq 0$. V případě úlohy (9.38-NN) je třeba navíc předpokládat $\alpha_0 - \frac{1}{2}r(0) > 0$ nebo $\alpha_\ell + \frac{1}{2}r(\ell) > 0$ nebo $q + \frac{1}{2}r' \geq q_0 > 0$ alespoň na části $\langle 0,\ell\rangle$, viz [23].

Diskretizace užitím lineárního prvku. Na intervalu $\langle 0,\ell\rangle$ zvolíme dělení $0 = x_0 < x_1 < \cdots < x_N = \ell$ a na každé úsečce $\langle x_{i-1}, x_i\rangle$ délky $h_i = x_i - x_{i-1}$ hledáme přibližné řešení $U(x)$ ve tvaru lineárního polynomu procházejícího body $[x_{i-1}, u_{i-1}]$ a $[x_i, u_i]$, takže

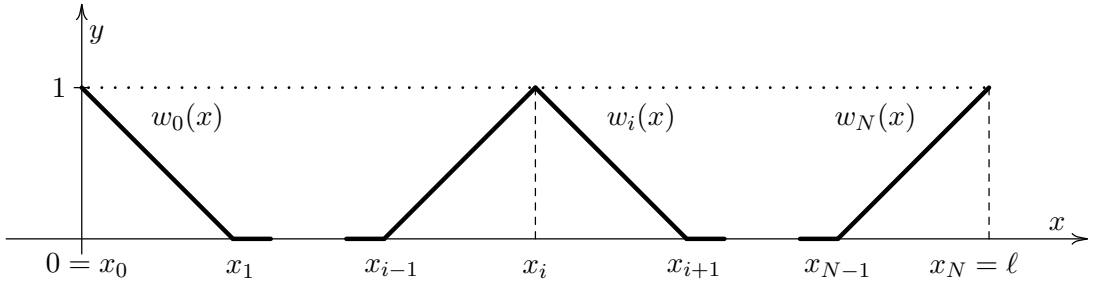
$$U(x) = U_{i-1}w_{i-1}(x) + U_i w_i(x), \quad \text{kde } w_{i-1}(x) = \frac{x_i - x}{h_i}, \quad w_i(x) = \frac{x - x_{i-1}}{h_i}.$$

Funkce $U(x)$ je tedy na celém intervalu $\langle 0,\ell\rangle$ *spojitou po částech lineární funkcí* určenou předpisem

$$U(x) = \sum_{i=0}^{N} U_i w_i(x), \tag{9.39}$$

kde $w_i(x)$ se jsou tzv. *bázové funkce*, lineární na každé úsečce $\langle x_{k-1}, x_k\rangle$ a takové, že

$$w_i(x_j) = \begin{cases} 1 \text{ pro } i = j, \\ 0 \text{ pro } i \neq j. \end{cases}$$



Obr. 9.2. Lineární Lagrangeovy bázové funkce

Úsečku $\langle x_{i-1}, x_i\rangle$, na které je definována lineární funkce určená svými hodnotami U_{i-1} resp. U_i v uzlech x_{i-1} resp. x_i, nazýváme *Lagrangeovým lineárním prvkem* nebo také *Lagrangeovým lineárním elementem*. Délku největšího dílku dělení $\{x_i\}_{i=0}^N$ označíme jako h, tj. $h = \max_{1\le i\le N} h_i$. Nechť X_h je prostor všech spojitých po částech lineárních funkcí tvaru $\sum_{i=0}^N \Theta_i w_i(x)$, kde $\{\Theta_i\}_{i=0}^N$ jsou libovolná reálná čísla. Zřejmě $X_h \subset X$. Nechť $V_h = V \cap X_h$ a $W_h = W \cap X_h$. Pak přibližné řešení U, tzv. *MKP řešení*, obdržíme z *diskrétní slabé formulace*

$$\text{najít } U \in W_h \text{ splňující } a_h(U,v) = L_h(v) \quad \forall v \in V_h. \tag{9.40}$$

Přitom index h v $a_h(U,v)$ resp. $L_h(v)$ značí, že integrál $\int_0^\ell [(pU'-r)v' + qUv]\,\mathrm{d}x$ v $a(U,v)$ resp. $\int_0^\ell fv\,\mathrm{d}x$ v $L(v)$ počítáme numericky kvadraturní formulí řádu alespoň jedna.

V dalším budeme pro konkrétnost uvažovat slabou formulaci (9.38-NN). Označíme-li $Q^i(\varphi)$ přibližně spočtenou hodnotu $\int_{x_{i-1}}^{x_i} \varphi\,\mathrm{d}x$, je

$$\begin{aligned} a_h(U,v) &= \sum_{i=1}^N Q^i([pU'-rU]v' + qUv) + [\alpha_0 - r(0)]U(x_0)v(x_0) + [\alpha_\ell + r(\ell)]U(x_\ell)v(x_\ell), \\ L_h(v) &= \sum_{i=1}^N Q^i(fv) + \beta_0 v(x_0) + \beta_\ell v(x_N)\,. \end{aligned} \tag{9.41}$$

Nechť $v(x) = \sum_{i=0}^N \Theta_i w_i(x) \in V_h$ je libovolná testovací funkce (tj. $\Theta_i = v(x_i)$ je libovolné číslo) a $U(x) = \sum_{j=0}^N \Delta_j w_j(x)$ je MKP řešení (tj. $\Delta_j = U(x_j)$). Pak z (9.40) pro úlohu (9.38-NN) plyne

$$\begin{aligned} 0 = a_h(U,v) - L_h(v) &= a_h\left(\sum_{j=0}^N \Delta_j w_j, \sum_{i=0}^N \Theta_i w_i\right) - L_h\left(\sum_{i=0}^N \Theta_i w_i\right) = \\ &= \sum_{i=0}^N \Theta_i \left[\sum_{j=0}^N a_h(w_j,w_i)\Delta_j - L_h(w_i)\right] = \boldsymbol{\theta}^T\left[\mathbf{K}\boldsymbol{\Delta} - \mathbf{F}\right], \end{aligned} \tag{9.42}$$

kde $\boldsymbol{\theta} = (\Theta_0, \Theta_1, \ldots, \Theta_N)^T$, $\mathbf{K} = \{k_{ij}\}_{i,j=0}^N$ pro $k_{ij} = a_h(w_j, w_i)$, $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \ldots, \Delta_N)^T$ a $\mathbf{F} = (F_0, F_1, \ldots, F_N)^T$ pro $F_i = L_h(w_i)$. Protože $\boldsymbol{\theta}$ je libovolný vektor, musí platit

$$\mathbf{K}\boldsymbol{\Delta} = \mathbf{F}. \tag{9.43}$$

Matice $\mathbf{K}$ bývá označována jako *matice tuhosti* a vektor $\mathbf{F}$ jako *vektor zatížení*. Toto pojmenování pochází z prvních aplikací MKP v pružnosti a stalo se univerzálním označením pro matici soustavy a pro vektor pravé strany v soustavě rovnic vzniklé diskretizací jakékoliv úlohy MKP.

Soustavu rovnic (9.43) sestavíme pomocí tzv. *elementárních matic tuhosti* $\mathbf{K}^i$ a *elementárních vektorů zatížení* $\mathbf{F}^i$ příslušných elementům $\langle x_{i-1}, x_i\rangle$, $i = 1, 2, \ldots, N$. Pomocí obdélníkové formule vyjádříme

$$Q^i(p\,U'v') = h_i p_{i-1/2} \frac{\Delta_i - \Delta_{i-1}}{h_i} \frac{\Theta_i - \Theta_{i-1}}{h_i} = [\boldsymbol{\theta}^i]^T \mathbf{K}^{i1} \boldsymbol{\Delta}^i,$$

kde

$$\boldsymbol{\theta}^i = \begin{pmatrix} \Theta_{i-1} \\ \Theta_i \end{pmatrix}, \qquad \mathbf{K}^{i1} = \frac{p_{i-1/2}}{h_i} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \quad \text{a} \quad \boldsymbol{\Delta}^i = \begin{pmatrix} \Delta_{i-1} \\ \Delta_i \end{pmatrix}.$$

Znovu pomocí obdélníkové formule dostaneme

$$Q^i(-r\,Uv') = -\,h_i r_{i-1/2} \frac{\Delta_i + \Delta_{i-1}}{2} \frac{\Theta_i - \Theta_{i-1}}{h_i} = [\boldsymbol{\theta}^i]^T \mathbf{K}^{i2} \boldsymbol{\Delta}^i,$$

kde

$$\mathbf{K}^{i2} = \tfrac{1}{2} r_{i-1/2} \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}.$$

Dále pomocí lichoběžníkové formule obdržíme

$$Q^i(qUv) = \tfrac{1}{2} h_i (q_{i-1}\Theta_{i-1}\Delta_{i-1} + q_i \Theta_i \Delta_i) = [\boldsymbol{\theta}^i]^T \mathbf{K}^{i3} \boldsymbol{\Delta}^i,$$

kde

$$\mathbf{K}^{i3} = \tfrac{1}{2} h_i \begin{pmatrix} q_{i-1} & 0 \\ 0 & q_i \end{pmatrix},$$

a položíme $\mathbf{K}^i = \mathbf{K}^{i1} + \mathbf{K}^{i2} + \mathbf{K}^{i3}$. Nakonec, opět pomocí lichoběžníkové formule, dostaneme

$$Q^i(fv) = \tfrac{1}{2} h_i (f_{i-1}\Theta_{i-1} + f_i \Theta_i) = [\boldsymbol{\theta}^i]^T \mathbf{F}^i, \quad \text{kde} \quad \mathbf{F}^i = \tfrac{1}{2} h_i \begin{pmatrix} f_{i-1} \\ f_i \end{pmatrix}.$$

Z rovnice

$$\begin{aligned} 0 = {} & a_h(U,v) - L_h(v) = \\ = {} & \left[\sum_{i=1}^{N} Q^i([pU' - rU]v' + qUv) + [\alpha_0 - r(x_0)]U(x_0)v(x_0) + [\alpha_\ell + r(x_N)]U(x_N)v(x_N) \right] - \\ & \left[\sum_{i=1}^{N} Q^i(fv) + \beta_0 v(x_0) + \beta_\ell v(x_N) \right] = \\ = {} & \sum_{i=1}^{N} [\boldsymbol{\theta}^i]^T [\mathbf{K}^i \boldsymbol{\Delta}^i - \mathbf{F}^i] + \Theta_0[(\alpha_0 - r_0)\Delta_0 - \beta_0] + \Theta_N[(\alpha_\ell + r_N)\Delta_N - \beta_\ell] \end{aligned}$$

a z rovnice (9.42) tak dostaneme rovnost

$$\begin{aligned} \boldsymbol{\theta}^T [\mathbf{K}\boldsymbol{\Delta} - \mathbf{F}] = {} & \sum_{i=1}^{N} [\boldsymbol{\theta}^i]^T [\mathbf{K}^i \boldsymbol{\Delta}^i - \mathbf{F}^i] + \\ & \Theta_0[(\alpha_0 - r_0)\Delta_0 - \beta_0] + \Theta_N[(\alpha_\ell + r_N)\Delta_N - \beta_\ell], \end{aligned} \tag{9.44}$$

z níž plyne postup, jak pomocí elementárních matic $\mathbf{K}^i = \{k^i_{rs}\}_{r,s=1}^2$, elementárních vektorů $\mathbf{F}^i = (F^i_1, F^i_2)^T$ a čísel α_0, r_0, β_0, α_ℓ, r_N, β_ℓ sestavit globální matici $\mathbf{K}$ a globální vektor $\mathbf{F}$:

$k_{11}^1+\alpha_0-r_0$	k_{12}^1		$\dots$			$\beta_0+F_1^1$
k_{21}^1	$k_{22}^1+k_{11}^2$	k_{12}^2	$\dots$			$F_2^1+F_1^2$
	k_{21}^2	$k_{22}^2+k_{11}^3$	$\dots$			$F_2^2+F_1^3$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
			$\dots$	$k_{22}^{N-1}+k_{11}^N$	k_{12}^N	$F_2^{N-1}+F_1^N$
			$\dots$	k_{21}^N	$k_{22}^N+\alpha_\ell+r_N$	$F_2^N+\beta_\ell$

Tab 9.1: Matice soustavy $\mathbf{K}$ a vektor pravé strany $\mathbf{F}$.

stačí srovnat členy se stejnými indexy u parametrů Θ a Δ (pro určení prvků matice $\mathbf{K}$) nebo jen u parametru Θ (pro určení prvků vektoru $\mathbf{F}$) na levé a na pravé straně rovnice (9.44). Struktura matice soustavy $\mathbf{K}$ a vektoru pravé strany $\mathbf{F}$ je patrná z tabulky 2.1.

Pro ekvidistantní dělení je výsledná soustava rovnic (9.43) stejná jako soustava rovnic (9.25), kterou jsme odvodili diferenční metodou.

Výpočet probíhá podle následujícího algoritmu:

1) Matici $\mathbf{K}$ a vektor $\mathbf{F}$ vynulujeme.

2) Postupně procházíme jednotlivé prvky $\langle x_{i-1}, x_i\rangle$, $i = 1, 2, \dots, N$, na každém z nich vypočteme elementární matici $\mathbf{K}^i = \{k_{rs}^i\}_{r,s=1}^2$ a elementární vektor $\mathbf{F}^i = \{F_r^i\}_{r=1}^2$ a koeficienty k_{rs}^i resp. F_r^i přičteme k odpovídajícím prvkům matice $\mathbf{K}$ resp. vektoru $\mathbf{F}$ v souladu s tabulkou 9.1.

3) Matici $\mathbf{K}$ a vektor $\mathbf{F}$ modifikujeme podle uvažovaných okrajových podmínek:

   a) Je-li v levém krajním bodě předepsána Dirichletova okrajová podmínka (9.11a), odstraníme první řádek matice $\mathbf{K}$ a první řádek vektoru $\mathbf{F}$, pak od pravé strany odečteme první sloupec matice $\mathbf{K}$ násobený předepsanou hodnotou g_0 a nakonec vynecháme také první sloupec matice $\mathbf{K}$.

   b) Je-li v levém krajním bodě předepsána Newtonova okrajová podmínka (9.18a), přičteme k prvku v levém horním rohu matice $\mathbf{K}$ číslo $\alpha_0 - r_0$ a k prvnímu prvku vektoru $\mathbf{F}$ přičteme číslo β_0.

   c) Je-li v pravém krajním bodě předepsána Dirichletova okrajová podmínka (9.11b), odstraníme poslední řádek matice $\mathbf{K}$ a poslední řádek vektoru $\mathbf{F}$, pak od pravé strany odečteme poslední sloupec matice $\mathbf{K}$ násobený předepsanou hodnotou g_ℓ a nakonec vynecháme také poslední sloupec matice $\mathbf{K}$.

   d) Je-li v pravém krajním bodě předepsána Newtonova okrajová podmínka (9.18b), přičteme k prvku v pravém dolním rohu matice $\mathbf{K}$ číslo $\alpha_\ell + r_N$ a k poslednímu prvku vektoru $\mathbf{F}$ přičteme číslo β_ℓ.

4) Vyřešíme soustavu lineárních rovnic $\mathbf{Ku} = \mathbf{F}$. Podle zvolených okrajových podmínek tak získáme

$\mathbf{u} = (u_1, u_2, \ldots, u_{N-1})^T$ v případě okrajových podmínek (9.11a), (9.11b),
$\mathbf{u} = (u_1, u_2, \ldots, u_N)^T$ v případě okrajových podmínek (9.11a), (9.18b),
$\mathbf{u} = (u_0, u_1, \ldots, u_{N-1})^T$ v případě okrajových podmínek (9.18a), (9.11b),
$\mathbf{u} = (u_0, u_1, \ldots, u_N)^T$ v případě okrajových podmínek (9.18a), (9.18b).

Pro chybu $u - U$ a její derivaci platí za předpokladu $u \in C^2\langle 0, \ell\rangle$ odhad

$$u - U = O(h^2)\,, \qquad u' - U' = O(h)\,. \tag{9.45}$$

Dominantní konvekce. Upwind aproximaci konvekčního členu dostaneme z modifikované rovnosti (9.40), a sice

$$\text{najít } U \in W_h \text{ splňující } a_h(U, v) + c_h(U, v) = L_h(v) \quad \forall v \in V_h, \tag{9.40'}$$

kde člen

$$c_h(U, v) = \sum_{i=1}^{N} Q^i\left(\delta_i U' v'\right), \quad \delta_i = \tfrac{1}{2} h_i |\sigma_i r|, \tag{9.46}$$

reprezentuje tzv. *umělou difúzi*. Vhodnou volbou koeficientů $\{\sigma_i\}_{i=1}^N$ ovlivňujeme velikost dodané umělé difúze. Rovnost (9.40') dostaneme z rovnosti (9.40) tak, že v ní nahradíme „difúzní člen“ $\sum_{i=1}^N Q^i(pU'v')$ členem $\sum_{i=1}^N Q^i([p+\delta_i]U'v')$, tj. na prvku $\langle x_{i-1}, x_i\rangle$ přidáme k difúzi p umělou difúzi δ_i. Pomocí obdélníkové formule dostaneme

$$Q^i\left(\tfrac{1}{2} h_i |\sigma_i r| U'v'\right) = [\boldsymbol{\theta}^i]^T \mathbf{K}^{i4} \boldsymbol{\Delta}^i, \tag{9.47}$$

kde

$$\mathbf{K}^{i4} = \tfrac{1}{2} |\sigma_i r_{i-1/2}| \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \tag{9.48}$$

Položíme $\mathbf{K}^i = \mathbf{K}^{i1} + \mathbf{K}^{i2} + \mathbf{K}^{i3} + \mathbf{K}^{i4}$ a soustavu rovnic sestavíme podle tabulky 2.1. Pro ekvidistantní dělení a $\sigma_i = 1$, $i = 1, 2, \ldots, N$, dostaneme rovnice (9.29). Ověřte! Pro chybu v tom případě platí pouze

$$u - U = O(h). \tag{9.49}$$

Když p, r jsou konstanty, $q = f = 0$, okrajové podmínky jsou Dirichletovy, viz (9.11), a dělení je ekvidistantní, pak pro

$$\sigma_i \equiv \sigma = \coth\kappa - \frac{1}{\kappa}, \quad \text{kde} \quad \kappa = \frac{rh}{2p}, \tag{9.50}$$

dostaneme přesné řešení, viz [13]. Číslo κ je známo jako *lokální Pecletovo číslo*. Snadno ověříme, že funkce $\sigma(\kappa)$ je rostoucí, $\lim\limits_{\kappa\to-\infty} \sigma(\kappa) = -1$, $\lim\limits_{\kappa\to 0} \sigma(\kappa) = 0$, $\lim\limits_{\kappa\to\infty} \sigma(\kappa) = 1$. To je žádoucí chování: pro dominantní konvekci $|\sigma| \to 1$, tj. dostaneme čistý upwind, a pro dominantní difúzi $|\sigma| \to 0$, tj. dostaneme standardní schéma bez umělé difúze.

Proto lze doporučit univerzální volbu

$$\sigma_i = \coth\kappa_i - \frac{1}{\kappa_i}, \quad \text{kde} \quad \kappa_i = \frac{r_{i-1/2}h_i}{2p_{i-1/2}}, \quad i = 1, 2, \ldots, N, \tag{9.51}$$

která je vhodná pro každou konvekčně-difúzní úlohu, nezávisle na tom, zda je konvekce dominantní či nikoliv.

Příklad 9.3. Uvažujme konvekčně difúzní úlohu

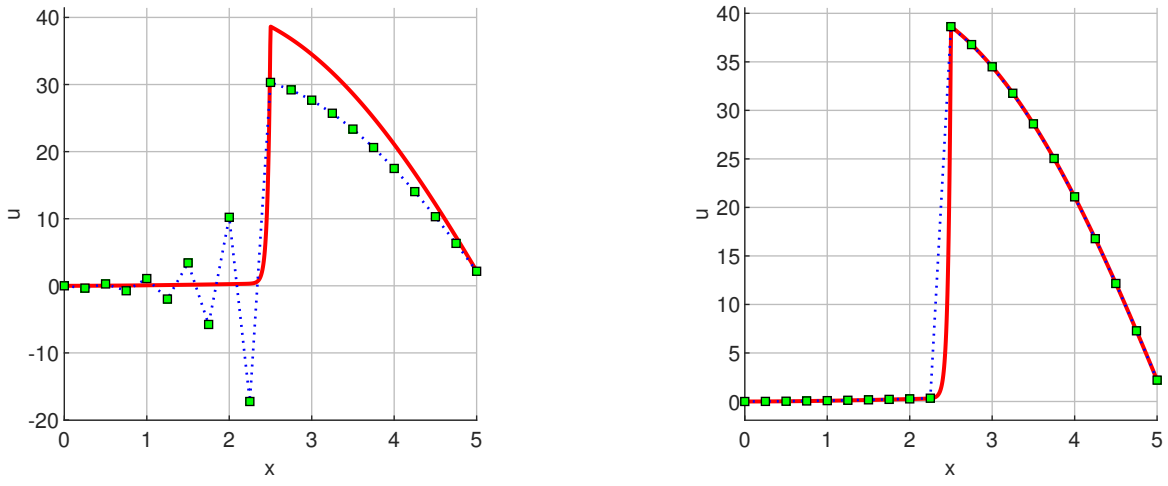
$$-u'' + (r(x)u)' = x(5-x), \quad x \in (0,5),$$
$$u(0) = 0, \quad u'(5) + 100(u(5) - 2) = 0,$$

přičemž

$$r(x) = \begin{cases} 30 \\ 0{,}1 \end{cases} \quad \text{pro} \quad \begin{cases} x < 2{,}5, \\ x \geq 2{,}5. \end{cases}$$

Řešení získané MKP pro $N = 20$ je zakresleno na obrázku 9.3. Červená křivka zobrazuje řešení spočtené programem `bvp4c` s vysokou přesností, tedy prakticky přesné řešení. Zelené čtverečky zobrazují numerické řešení, vlevo bez umělé difúze, tj. když $\mathbf{K}^{i4} = \mathbf{O}$, vpravo s umělou difúzí, viz (9.48), (9.49) a (9.51). Pro $\mathbf{K}^{i4} = \mathbf{O}$ a $N > 75$ dostaneme numerické řešení bez oscilací, jak lze experimentálně ověřit. Kritická hodnota $N = 75$ je v souladu s podmínkami (9.26) pro $p = 1$, $r = 30$ a $h = 5/N$:

$$\frac{1}{2}rh < p, \quad \text{tedy} \quad \frac{1}{2}30\frac{5}{N} < 1, \quad \text{takže} \quad N > 75. \qquad \square$$



Obr. 9.3. $N = 20$, vlevo bez umělé difúze, vpravo s umělou difúzí.

10. Parciální diferenciální rovnice

Parciální diferenciální rovnice (stručně PDR) vyjadřuje vztah mezi funkcí několika proměnných a jejími parciálními derivacemi. Parciální rovnice a jejich soustavy jsou matematickým modelem mnoha technických úloh. Četné modely byly vytvořeny již v minulých staletích, jejich praktické řešení však umožnily teprve výkonné počítače. Prostředky klasické matematické analýzy se zkoumá existence, jednoznačnost, hladkost a další vlastnosti řešení v závislosti na koeficientech rovnice, okrajových a počátečních podmínkách a na oblasti, ve které má být rovnice splněna. Tuto oblast budeme standardně značit symbolem Ω. Pro některé jednodušší úlohy lze těmito prostředky najít i přesné řešení, často ve tvaru nekonečné řady. Převážnou většinu těchto úloh však dovedeme řešit pouze přibližně, numericky.

Označení *eliptické, parabolické* nebo *hyperbolické* získaly PDR na základě formální podobnosti s rovnicemi kuželoseček. *Stacionární* úlohy jsou na čase nezávislé, úlohy *nestacionární* na čase závisejí. K jednoznačnému určení řešení nestačí samotná diferenciální rovnice, je nutno zadat ještě okrajové podmínky a u nestacionárních úloh také počáteční podmínky. Ve třech následujících kapitolách uvedeme nejjednodušší rovnice druhého řádu. Omezíme se přitom na PDR ve dvou proměnných, tj. ve stacionárním případě jde o úlohu ve dvou prostorových proměnných x, y a v nestacionárním případě se uvažují úlohy s jedinou prostorovou proměnnou x, druhou proměnnou je čas t.

Diskretizaci v prostorových proměnných provedeme pomocí diferenční metody, metody konečných objemů a metody konečných prvků. Metoda konečných prvků jednoznačně dominuje při řešení problémů mechaniky pevné fáze, zatímco pro proudění tekutin se více používají programy pracující na bázi metody konečných objemů.

Několik pojmů. *Uzávěr* množiny $M \in \mathbb{R}^d$ je sjednocením bodů množiny M a bodů ležících na její hranici ∂M. Uzávěr M značíme $\overline{M}$, tj. $\overline{M} = M \cup \partial M$.

Oblastí rozumíme otevřenou souvislou množinu v $\mathbb{R}^d$.

Nechť Ω je oblast. Prostor funkcí, které jsou v $\bar{\Omega}$ spojité spolu se svými derivacemi až do řádu k, značíme $C^k(\overline{\Omega})$. Prostor $C^0(\overline{\Omega})$ funkcí spojitých v $\overline{\Omega}$ značíme stručně $C(\overline{\Omega})$.

Řekneme, že funkce u je v Ω po částech spojitá, jestliže $\overline{\Omega}$ je sjednocením uzávěrů konečného počtu navzájem disjunktních podoblastí, tj. $\overline{\Omega} = \bigcup \overline{\Omega}_i$, $\Omega_i \cap \Omega_j = \emptyset$ pro $i \neq j$, a jestliže u je na každé z podoblastí Ω_i spojitá a spojitě prodloužitelná až do hranice, tj. existuje spojitá funkce $\overline{u}_i \in C(\overline{\Omega}_i)$ s vlastností $\overline{u}_i = u_i$ v Ω_i. Prostor po částech spojitých funkcí značíme $PC(\Omega)$. Symbolem $PC^k(\Omega)$ značíme prostor funkcí, které jsou v oblasti $\overline{\Omega}$ spojité spolu se všemi svými derivacemi až do řádu $k-1$ včetně a jejichž k-té derivace jsou po částech spojité. Tak třeba $PC^1(\Omega)$ je prostor funkcí, které jsou v $\overline{\Omega}$ spojité a jejichž první derivace jsou v Ω po částech spojité.

Definici funkce spojité a funkce po částech spojité na hranici lze prostřednictvím parametrického vyjádření hranice převést na definici funkce spojité a funkce po částech spojité na úsečce, viz kapitola 9.4. Pokud jde o značení, tak třeba $PC(\Gamma_\ell)$ je prostor po částech spojitých funkcí na části $\Gamma_\ell \subset \partial\Omega$ hranice oblasti Ω.

10.1. Úloha eliptického typu

10.1.1. Formulace úlohy

Buď Ω omezená *oblast* v $\mathbb{R}^2$. O *hranici* $\Gamma = \partial\Omega$ oblasti Ω předpokládejme, že je sjednocením uzávěrů dvou navzájem disjunktních částí Γ_1 a Γ_2, tj. $\Gamma = \overline{\Gamma}_1 \cup \overline{\Gamma}_2$, $\Gamma_1 \cap \Gamma_2 = \emptyset$. Dále $\mathbf{n} = (n_1, n_2)^T$ nechť je *jednotkový vektor vnější normály* hranice a

$$\frac{\partial u}{\partial n} = \mathbf{n} \cdot \nabla u = n_1 \frac{\partial u}{\partial x} + n_2 \frac{\partial u}{\partial y}$$

je *derivace ve směru vnější normály.*

Nechť $p(x,y) \geq p_0 > 0$, $q(x,y) \geq 0$, $f(x,y)$, $g(x,y)$, $\alpha(x,y) \geq 0$ a $\beta(x,y)$ jsou dané funkce. Naším úkolem je určit funkci $u(x,y)$, která uvnitř Ω vyhovuje diferenciální rovnici

$$-\frac{\partial}{\partial x}\Big(p(x,y)\frac{\partial u}{\partial x}\Big) - \frac{\partial}{\partial y}\Big(p(x,y)\frac{\partial u}{\partial y}\Big) + q(x,y)u = f(x,y) \qquad \text{v } \Omega\,, \tag{10.1}$$

na hranici Γ_1 splňuje Dirichletovu okrajovou podmínku

$$u = g(x,y) \qquad \text{na } \Gamma_1\,, \tag{10.2}$$

a na hranici Γ_2 splňuje Newtonovu okrajovou podmínku

$$-p(x,y)\frac{\partial u}{\partial n} = \alpha(x,y)u - \beta(x,y) \qquad \text{na } \Gamma_2\,. \tag{10.3}$$

Je-li v (10.3) $\alpha = 0$, dostaneme *Neumannovu* okrajovou podmínku.

V případě, že p je konstanta a $q = 0$, dělíme rovnici (10.1) číslem p a vznikne *Poissonova rovnice*

$$-\Delta u = f(x,y) \quad \text{v } \Omega, \tag{10.4}$$

kde *Laplaceův operátor* Δ aplikovaný na funkci u je

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\,.$$

Rovnice $\Delta u = 0$ se nazývá *Laplaceva rovnice.*

Fyzikální význam. Úlohu (10.1) – (10.3) můžeme interpretovat jako *stacionární vedení tepla* v nekonečném hranolu o průřezu Ω. Pak u je teplota, p tepelná vodivost, $-p\nabla u$ je tepelný tok, q je měrný tepelný odpor, f je intenzita vnitřních tepelných zdrojů, okrajová podmínka (10.2) předepisuje teplotu na povrchu a v okrajové podmínce (10.3) je $-p\,\partial u/\partial n$ tepelný tok ve směru vnější normály, α je koeficient přestupu tepla a β je předepsaný tepelný tok.

Poissonova rovnice s homogenní Dirichletovou okrajovou podmínkou $u = 0$ vyjadřuje např. průhyb membrány upevněné na okraji a zatížené tlakem úměrným funkci f.

Laplaceova rovnice s Neumannovou okrajovou podmínkou popisuje např. potenciální proudění: u je potenciál vektoru rychlosti $\mathbf{v} = (v_1, v_2)^T$, kde $v_1 = \partial u/\partial x$, $v_2 = \partial u/\partial y$.

Rovnice $0 = \Delta u = \mathrm{div}\mathbf{v} = 0$ je známa jako podmínka nestlačitelnosti. Neumannova okrajová podmínka $\partial u/\partial n = \mathbf{v}\cdot\mathbf{n} = \beta$ předepisuje normálovou složku rychlosti $v_n = \mathbf{v}\cdot\mathbf{n}$. Pomocí Gauss-Ostrogradského věty, viz např. [44],

$$\int_\Omega \mathrm{div}\,\mathbf{v}\,\mathrm{d}x\,\mathrm{d}y = \int_{\partial\Omega} \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \int_{\partial\Omega} \frac{\partial u}{\partial n}\,\mathrm{d}S = \int_{\partial\Omega} \beta\,\mathrm{d}S = 0,$$

dostáváme nutnou podmínku existence řešení. Potenciál u není určen jednoznačně: je-li u řešení, je také $u+C$ řešení, kde C je libovolná konstanta. Rychlost $\mathbf{v}$ však už jednoznačně určena je.

Existence a jednoznačnost řešení. Podmínky zaručující existenci a jednoznačnost klasického řešení $u \in C^2(\overline{\Omega})$ problému (10.1)–(10.3) jsou komplikované a proto je zde uvádět nebudeme. V praktických aplikacích vystačíme s existencí tzv. *slabého řešení*, které existuje za podmínek v aplikacích běžně splněných.

V dalším budeme předpokládat, že Ω je mnohoúhelník, $p \geq p_0 > 0$, $q \geq 0$, f jsou po částech spojité v Ω, g je spojitá na Γ_1, $\alpha \geq 0$, β jsou po částech spojité na Γ_2, a pokud $\Gamma = \Gamma_2$, pak buďto $q \geq q_0 > 0$ na části Ω nebo $\alpha \geq \alpha_0 > 0$ na části Γ_2. Za těchto předpokladů existuje jediné slabé řešení $u \in PC^1(\Omega)$ problému (10.1)–(10.3), viz např. [23]. Je-li $\Gamma = \Gamma_2$, $q = 0$, $\alpha = 0$ a pokud $\int_\Omega f\,\mathrm{d}x\,\mathrm{d}y + \int_\Gamma \beta\,\mathrm{d}s = 0$, pak má úloha (10.1)–(10.3) nekonečně mnoho řešení: je-li u řešení, pak také $u+C$ je řešení, kde C je libovolná konstanta.

Zesílením uvedených předpokladů lze docílit toho, že slabé řešení je také řešení klasické. Tyto zesílené předpoklady však obvykle odporují požadavkům praktických aplikací.

10.1.2. Diferenční metoda

Diskretizace okrajové úlohy ve dvou dimenzích je analogická diskretizaci jednodimenzionální úlohy, viz kapitola 9.2.

Princip metody. Abychom výklad nezatěžovali detaily nepodstatnými z hlediska numerické metody, začneme řešením Dirichletovy úlohy pro Poissonovu rovnici na čtverci, tj. řešíme rovnici (10.4) s okrajovou podmínkou (10.2) pro $\Gamma_1 = \Gamma$, když $\Omega = (0,\ell)\times(0,\ell)$ je čtverec se stranou délky ℓ.

Na Ω vytvoříme pravidelnou čtvercovou síť. Diferenční metoda se proto také často nazývá *metoda sítí*. Zvolme tedy $N > 1$ celé a definujme *krok* $h = \ell/N$. Označme $x_i = ih$, $i = 0,1,\ldots,N$, $y_j = jh$, $j = 0,1,\ldots,N$. Body $[x_i, y_j]$, $i,j = 0,1,\ldots,N$, nazveme *uzly sítě*. Rovnice (10.4) má být splněna ve všech bodech $[x,y]$ uvnitř Ω, musí tedy být také splněna ve všech vnitřních uzlech, tj.

$$-\frac{\partial^2 u(x_i,y_j)}{\partial x^2} - \frac{\partial^2 u(x_i,y_j)}{\partial y^2} = f(x_i,y_j), \quad i,j = 1,2,\ldots,N-1\,. \tag{10.5}$$

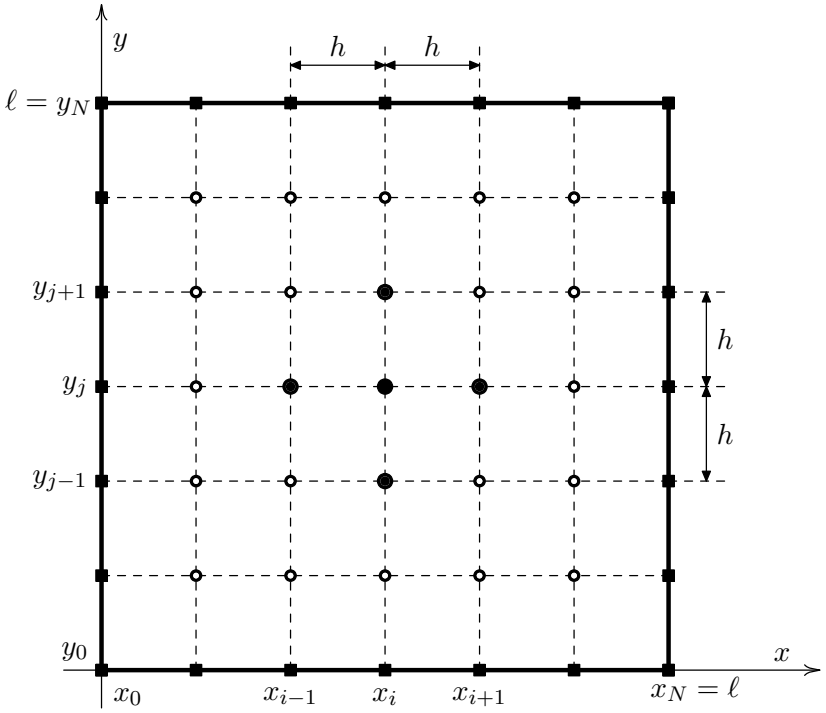
Parciální derivace vyjádříme pomocí centrálních diferencí

$$-\frac{\partial^2 u(x_i,y_j)}{\partial x^2} = \frac{-u(x_{i-1},y_j) + 2u(x_i,y_j) - u(x_{i+1},y_j)}{h^2} + O(h^2)\,, \tag{10.6a}$$

$$-\frac{\partial^2 u(x_i,y_j)}{\partial y^2} = \frac{-u(x_i,y_{j-1}) + 2u(x_i,y_j) - u(x_i,y_{j+1})}{h^2} + O(h^2)\,, \tag{10.6b}$$

dosadíme do rovnice (10.5) a chybové členy $O(h^2)$ zanedbáme. Po vynásobení h^2 dostaneme soustavu *síťových rovnic*

$$-u_{i-1,j} - u_{i,j-1} + 4u_{ij} - u_{i,j+1} - u_{i+1,j} = h^2 f_{ij}\,, \quad i,j = 1,2,\ldots,N-1\,, \tag{10.7}$$



Obr. 10.1. Síť

kde u_{ij} je aproximace $u(x_i, y_j)$ a $f_{ij} = f(x_i, y_j)$. Z okrajové podmínky (10.2) dostaneme

$$u_{ij} = g_{ij} \quad \text{pro } i = 0 \text{ nebo } i = N \text{ nebo } j = 0 \text{ nebo } j = N\,, \tag{10.8}$$

přičemž $g_{ij} = g(x_i, y_j)$. Když z (10.8) dosadíme do (10.7) a na levé straně ponecháme pouze členy s neznámými u_{ij}, dostaneme soustavu $(N-1)^2$ lineárních algebraických rovnic, kterou můžeme zapsat maticově ve tvaru

$$\mathbf{Ku} = \mathbf{F}\,. \tag{10.9}$$

Pro $N = 4$ má soustava rovnic (10.9) následující tvar:

$$\left(\begin{array}{rrr|rrr|rrr}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
\hline
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{array}\right)
\left(\begin{array}{c} u_{11} \\ u_{12} \\ u_{13} \\ \hline u_{21} \\ u_{22} \\ u_{23} \\ \hline u_{31} \\ u_{32} \\ u_{33} \end{array}\right)
=
\left(\begin{array}{l} h^2 f_{11} + g_{01} + g_{10} \\ h^2 f_{12} + g_{02} \\ h^2 f_{13} + g_{03} + g_{14} \\ \hline h^2 f_{21} + \qquad g_{20} \\ h^2 f_{22} \\ h^2 f_{23} + \qquad g_{24} \\ \hline h^2 f_{31} + g_{41} + g_{30} \\ h^2 f_{32} + g_{42} \\ h^2 f_{33} + g_{43} + g_{34} \end{array}\right)$$

Pravá strana rovnice odpovídající uzlu, který není sousedem hranice, obsahuje pouze člen $h^2 f_{ij}$, pro uzly nejbližší vrcholům čtverce přibudou dva členy s funkcí g a pro ostatní sousedy hranice jeden člen s funkcí g.

Soustavu (10.9) lze napsat v blokovém tvaru

$$\begin{pmatrix} \mathbf{B} & -\mathbf{I} & \mathbf{O} & \dots & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ -\mathbf{I} & \mathbf{B} & -\mathbf{I} & \dots & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & -\mathbf{I} & \mathbf{B} & \dots & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \dots & -\mathbf{I} & \mathbf{B} & -\mathbf{I} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \dots & \mathbf{O} & -\mathbf{I} & \mathbf{B} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \vdots \\ \mathbf{u}_{N-2} \\ \mathbf{u}_{N-1} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \mathbf{F}_3 \\ \vdots \\ \vdots \\ \mathbf{F}_{N-2} \\ \mathbf{F}_{N-1} \end{pmatrix} \tag{10.10}$$

kde $\mathbf{I}$ je jednotková matice řádu $N-1$, $\mathbf{O}$ je nulová matice řádu $N-1$ a $\mathbf{B}$ je třídiagonální matice řádu $N-1$ tvaru

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

Vektory $\mathbf{u}_i = (u_{i1}, u_{i2}, \dots, u_{i,N-1})^T$, $i = 1, 2, \dots, N-1$, jsou části celkového vektoru $\mathbf{u}$ neznámých, vektory $\mathbf{F}_i = (F_{i1}, F_{i2}, \dots, F_{i,N-1})^T$, $i = 1, 2, \dots, N-1$, jsou části celkového vektoru $\mathbf{F}$ pravých stran.

Pro homogenní okrajovou podmínku $g(x, y) = 0$ je $F_{ij} = h^2 f_{ij}$, pro nehomogenní okrajovou podmínku je v některých rovnicích příspěvek z hranice, přesně

$$\begin{aligned} F_{ij} &= h^2 f_{ij}\,, & & 2 \le i \le N-2, \quad 2 \le j \le N-2\,, \\ F_{1j} &= h^2 f_{1j} + g_{0j}\,, & F_{N-1,j} &= h^2 f_{N-1,j} + g_{Nj}\,, \quad 2 \le j \le N-2\,, \\ F_{i1} &= h^2 f_{i1} + g_{i0}\,, & F_{i,N-1} &= h^2 f_{i,N-1} + g_{iN}\,, \quad 2 \le i \le N-2\,, \\ F_{11} &= h^2 f_{11} + g_{01} + g_{10}\,, & F_{1,N-1} &= h^2 f_{1,N-1} + g_{0,N-1} + g_{1N}\,, \\ F_{N-1,1} &= h^2 f_{N-1,1} + g_{N1} + g_{N-1,0}\,, & F_{N-1,N-1} &= h^2 f_{N-1,N-1} + g_{N,N-1} + g_{N-1,N}\,. \end{aligned}$$

Matice $\mathbf{K}$ soustavy (10.9) má řadu vynikajících vlastností. Některé z nich jsou patrné okamžitě: $\mathbf{K}$ je symetrická, diagonálně dominantní, pásová a pětidiagonální. Matice $\mathbf{K}$ je pro velké N řídká, neboť počet jejích nenulových prvků je malý: z celkového počtu $(N-1)^4$ je jich nenulových jen $5(N-1)^2 - 4(N-1)$. Dá se dokázat, že $\mathbf{K}$ je pozitivně definitní a tedy regulární.

Jsou-li funkce f a g dostatečně hladké, pak pro chybu metody platí

$$u(x_i, y_j) - u_{ij} = O(h^2)\,. \tag{10.11}$$

Obdélníková oblast. Je-li $\Omega = (0,a)\times(0,b)$ obdélník a na něm chceme zvolit pravidelnou síť, musíme ve směru osy y vybrat obecně jiný krok než ve směru osy x. Nechť tedy $N \geq 1$ je počet dílků ve směru osy x a $M \geq 1$ je počet dílků ve směru osy y. Položíme $h = a/N$, $k = b/M$ a definujeme $x_i = ih$, $i = 0,1,\ldots,N$, a $y_j = jk$, $j = 0,1,\ldots,M$. Síť je tvořena uzly $[x_i, y_j]$ pro $i = 0,1,\ldots,N$, $j = 0,1,\ldots,M$. Při diskretizaci Poissonovy rovnice (10.4) postupuje analogicky jako dříve, jen místo (10.6b) použijeme

$$-\frac{\partial^2 u(x_i,y_j)}{\partial y^2} = \frac{-u(x_i,y_{j-1}) + 2u(x_i,y_j) - u(x_i,y_{j+1})}{k^2} + O(k^2)\,, \tag{10.6b'}$$

a tak místo rovnic (10.7) dostaneme pro $i = 1,2,\ldots,N-1$ a $j = 1,2,\ldots,M-1$ rovnice

$$\frac{-u_{i-1,j} + 2u_{ij} - u_{i+1,j}}{h^2} + \frac{-u_{i,j-1} + 2u_{ij} - u_{i,j+1}}{k^2} = f_{ij}\,. \tag{10.7'}$$

Je-li řešení u dostatečně hladké, pro chybu platí

$$u(x_i,y_j) - u_{ij} = O(h^2 + k^2)\,. \tag{10.11'}$$

Obecnější rovnice. Při diskretizaci rovnice (10.1) aproximujeme členy $-[pu_x]_x$ a $-[pu_y]_y$ pomocí vzorce (9.14). Tak ve vnitřních uzlech dostaneme místo (10.7') rovnici

$$\begin{aligned}&\frac{-p_{i-1/2,j}u_{i-1,j} + (p_{i-1/2,j} + p_{i+1/2,j})u_{ij} - p_{i+1/2,j}u_{i+1,j}}{h^2} + \\ &\frac{-p_{i,j-1/2}u_{i,j-1} + (p_{i,j-1/2} + p_{i,j+1/2})u_{ij} - p_{i,j+1/2}u_{i,j+1}}{k^2} + q_{ij}u_{ij} = f_{ij}\,.\end{aligned} \tag{10.7"}$$

Přitom index $_{i\pm1/2}$ znamená, že za argument x dosadíme $x_i \pm \frac{1}{2}h$, a podobně index $_{j\pm1/2}$ znamená, že za y dosadíme $y_j \pm \frac{1}{2}k$.

Je-li řešení u dostatečně hladké, pro chybu opět platí (10.11').

Newtonova okrajová podmínka. Při diskretizaci postupujeme obdobně jako v jednodimenzionálním případě, viz odvození vztahů (9.19). Ukážeme si to na příkladu podmínky

$$-p(a,y)\frac{\partial u(x,y)}{\partial x}\Big|_{x=a} = \alpha^e(y)u(a,y) - \beta^e(y)\,, \qquad 0 < y < b\,, \tag{10.12e}$$

na východní (east) straně obdélníka Ω. Splnění rovnice (10.5) požadujeme i v uzlech $[x_N, y_j]$ na straně $x = a$. Při diskretizaci $-[pu_x]_x(x_N,y_j)$, $1 \leq j \leq M-1$, postupujeme zcela analogicky jako v jedné dimenzi, tj.

$$\begin{aligned}-[pu_x]_x(x_N,y_j) = &-\frac{p_{Nj}u_x(x_N,y_j) - p_{N-1/2,j}u_x(x_{N-1/2},y_j)}{\frac{1}{2}h} + O(h) = \\ &\frac{\alpha_j^e u(x_N,y_j) - \beta_j^e + p_{N-1/2,j}\dfrac{u(x_N,y_j) - u(x_{N-1},y_j)}{h}}{\frac{1}{2}h} + O(h)\,.\end{aligned} \tag{10.13e}$$

Pomocí (10.13e) a užitím standardní aproximace členu $-[pu_y]_y(x_N, y_j)$ dostaneme ve vnitřních uzlech $[x_N, y_j]$ východní strany $x = a$, tj. pro $j = 1, 2, \ldots, M-1$, rovnici

$$\begin{aligned}
&\frac{-p_{N-1/2,j}u_{N-1,j} + (p_{N-1/2,j} + h\alpha_j^e)u_{Nj}}{h^2} + \\
&\frac{-p_{N,j-1/2}u_{N,j-1} + (p_{N,j-1/2} + p_{N,j+1/2})u_{Nj} - p_{N,j+1/2}u_{N,j+1}}{2k^2} + \\
&\frac{1}{2}q_{Nj}u_{Nj} = \frac{1}{2}f_{Nj} + \frac{1}{h}\beta_j^e\,.
\end{aligned} \tag{10.14e}$$

Předpokládejme, že Newtonova okrajová podmínka je předepsána také na severní (north) straně straně Ω, tj. že platí

$$-p(x,b)\frac{\partial u(x,y)}{\partial y}\Big|_{y=b} = \alpha^n(x)u(x,b) - \beta^n(x)\,, \qquad 0 < x < a\,. \tag{10.12n}$$

Podobně jako při odvození (10.13e) dostaneme

$$-[pu_y]_y(x_i, y_M) = \frac{\alpha_i^n u(x_i, y_M) - \beta_i^n + p_{i,M-1/2}\dfrac{u(x_i,y_M) - u(x_i,y_{M-1})}{k}}{\frac{1}{2}k} + O(k)\,. \tag{10.13n}$$

Odtud a užitím standardní aproximace členu $-[pu_x]_x(x_i, y_M)$ dostaneme pro vnitřní uzly horní strany $y = b$, tj. pro $i = 1, 2, \ldots, N-1$, rovnici

$$\begin{aligned}
&\frac{-p_{i-1/2,M}u_{i-1,M} + (p_{i-1/2,M} + p_{i+1/2,M})u_{iM} - p_{i+1/2,M}u_{i+1,M}}{2h^2} + \\
&\frac{-p_{i,M-1/2}u_{i,M-1} + (p_{i,M-1/2} + k\alpha_i^n)u_{iM}}{k^2} + \frac{1}{2}q_{iM}u_{iM} = \frac{1}{2}f_{iM} + \frac{1}{k}\beta_i^n\,.
\end{aligned} \tag{10.14n}$$

Pokud je Newtonova okrajová podmínka předepsána současně na východní i severní straně, dostaneme v severovýchodním rohu $[x_N, y_M]$ (pomocí (10.13e) pro $j = M$ a (10.13n) pro $i = N$) rovnici

$$\begin{aligned}
&\frac{-p_{N-1/2,M}u_{N-1,M} + (p_{N-1/2,M} + h\alpha_M^e)u_{NM}}{2h^2} + \\
&\frac{-p_{N,M-1/2}u_{N,M-1} + (p_{N,M-1/2} + k\alpha_N^n)u_{NM}}{2k^2} + \\
&\frac{1}{4}q_{NM}u_{NM} = \frac{1}{4}f_{NM} + \frac{1}{2h}\beta_M^e + \frac{1}{2k}\beta_N^n\,.
\end{aligned} \tag{10.14ne}$$

Při diskretizaci Newtonovy okrajové podmínky na ostatních stranách a v rozích postupujeme podobně. Je-li řešení u dostatečně hladké, pro chybu opět platí (10.11').

Rovnice s konvekčním členem je tvaru

$$-\frac{\partial}{\partial x}\Big(p(x,y)\frac{\partial u}{\partial x} - r_1(x,y)u\Big) - \frac{\partial}{\partial y}\Big(p(x,y)\frac{\partial u}{\partial y} - r_2(x,y)u\Big) + q(x,y)u = f(x,y). \tag{10.15}$$

Nechť $\mathbf{r} = (r_1, r_2)^T$. Na části $\Gamma_1 = \{\mathbf{x} \in \partial\Omega \mid \mathbf{r} \cdot \mathbf{n} \leq 0\}$ hranice předepíšeme Dirichletovu okrajovou podmínku a na zbývající části $\Gamma_2 = \{\mathbf{x} \in \partial\Omega \mid \mathbf{r} \cdot \mathbf{n} > 0\}$ hranice předepíšeme Newtonovu okrajovou podmínku. V dynamice tekutin Γ_1 může být vtok, to když $\mathbf{r} \cdot \mathbf{n} < 0$, nebo neprostupná stěna, to když $\mathbf{r} \cdot \mathbf{n} = 0$. Část Γ_2 reprezentuje výtok. Abychom zajistili jednoznačnou existenci řešení, předpokládejme, že

$$\operatorname{div} \mathbf{r}(x, y) \equiv \frac{\partial r_1(x, y)}{\partial x} + \frac{\partial r_2(x, y)}{\partial y} \geq 0, \qquad (x, y) \in \Omega. \tag{10.16}$$

V dynamice tekutin $\operatorname{div} \mathbf{r}(x, y) = 0$. Konvekční členy $(r_1 u)_x$ a $(r_2 u)_y$ aproximujeme podobně jako v jednorozměrné úloze, viz. kapitola 9.2. Ukažme si to pro konvekční člen $(r_1 u)_x$. Omezíme se přitom jen na vnitřní uzel $[x_i, y_j]$. Pomocí centrální diference dostaneme

$$(r_1 u)_x(x_i, y_j) \approx \left([r_1 u](x_{i+1/2}, y_j) - [r_1 u](x_{i-1/2}, y_j) \right) / h.$$

Dalším krokem je aproximace konvekčních toků $[r_1 u](x_{i+1/2}, y_j)$ a $[r_1 u](x_{i-1/2}, y_j)$. Protože aproximace obou těchto toků je založena na stejných pravidlech, věnujme se podrobně jen aproximaci $[r_1 u](x_{i+1/2}, y_j)$. Ta závisí na dvourozměrné analogii podmínky (9.26): pokud platí

$$\tfrac{1}{2} h |[r_1]_{0j}| < p_{1/2,j}, \quad \tfrac{1}{2} h |[r_1]_{i+1/2,j}| < p_{i+1/2,j},\ i = 0, 1, \ldots, N-1, \quad \tfrac{1}{2} h |[r_1]_{Nj}| < p_{N-1/2,j}, \tag{10.17}$$

určíme $u(x_{i+1/2}, y_j)$ interpolací z hodnot $u(x_i, y_j)$ a $u(x_{i+1}, y_j)$, tj.

$$[r_1 u](x_{i+1/2}, y_j) \approx \tfrac{1}{2} [r_1]_{i+1/2,j} \left(u(x_i, y_j) + u(x_{i+1}, y_j) \right), \tag{10.18}$$

v opačném případě použijeme upwind aproximaci

$$[r_1 u](x_{i+1/2}, y_j) \approx [r_1]^+_{i+1/2,j} u(x_i, y_j) + [r_1]^-_{i+1/2,j} u(x_{i+1}, y_j). \tag{10.19}$$

Konvekční člen $(r_2 u)_y(x_i, y_j)$ aproximujeme obdobně jako člen $(r_1 u)_x(x_i, y_j)$, při rozhodování o typu aproximace konvekčního toku $[r_2 u](x_i, y_{j+1/2})$ se řídíme podmínkou

$$\tfrac{1}{2} k |[r_2]_{i0}| < p_{i,1/2}, \quad \tfrac{1}{2} k |[r_2]_{i,j+1/2}| < p_{i,j+1/2},\ j = 0, 1, \ldots, M-1, \quad \tfrac{1}{2} k |[r_2]_{iM}| < p_{i,M-1/2}. \tag{10.20}$$

Aproximujeme-li konvekční členy interpolací, pak za předpokladu dostatečné hladkosti řešení u pro chybu platí (10.11'). Jestliže však konvekční členy aproximujeme užitím upwind aproximace, řád chyby se o jednotku sníží, pro chybu platí jen

$$u(x_i, y_j) - u_{ij} = O(h + k).$$

Pro dosažení přesnosti řádu $O(h^2 + k^2)$ je třeba používat přesnější upwind aproximaci konvekčního toku, viz (9.28').

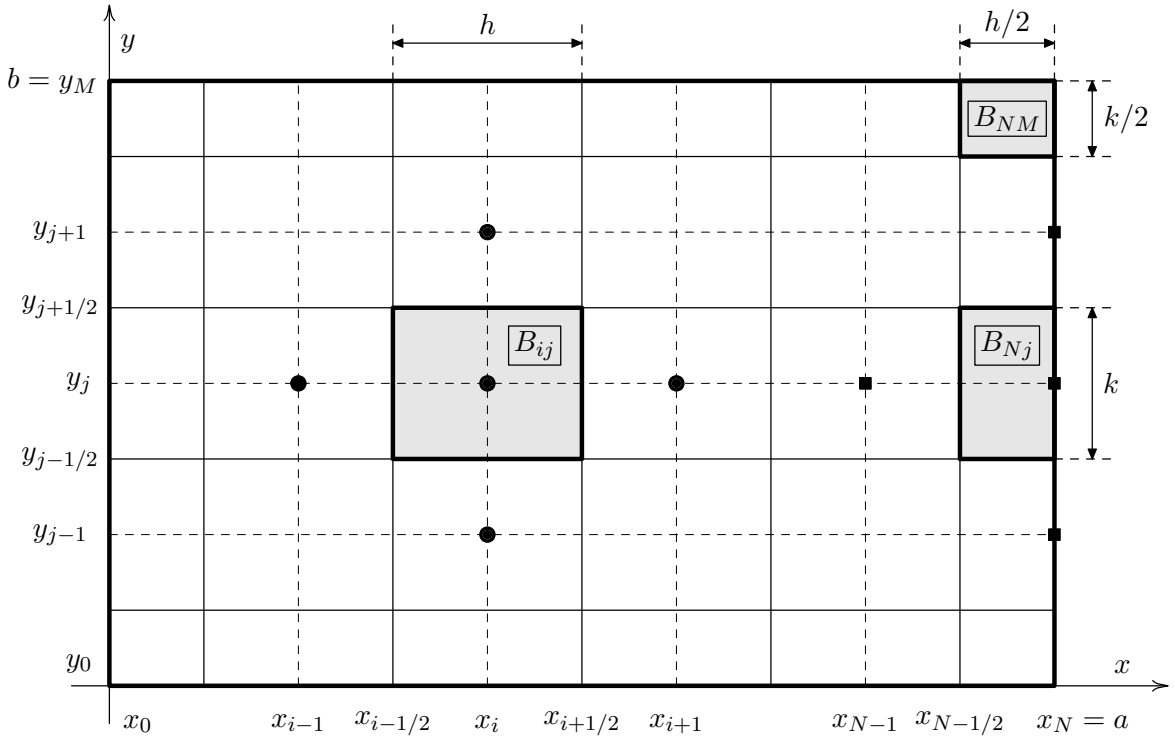
Dirichletova okrajová podmínka. Standardní postup je tento: je-li v uzlu $[x_i, y_j]$ předepsána hodnota řešení $u(x_i, y_j) = g_{ij}$, rovnici pro tento uzel vůbec nesestavujeme a v rovnicích obsahujících u_{ij} položíme $u_{ij} = g_{ij}$.

Podmínky $u_{ij} = g_{ij}$ lze vynutit i jinak. Nejdříve sestavíme soustavu rovnic jako kdyby na části Γ_1 hranice byla předepsána Newtonova okrajová podmínka (10.3) s $\alpha = \beta = 0$. Následně modifikujeme rovnice příslušné uzlům ležícím na Γ_1. Předpokládejme, že uzlu $[x_i, y_j]$ s předepsanou hodnotou $u(x_i, y_j) = g_{ij}$ přísluší r-tá rovnice soustavy $\mathbf{Ku} = \mathbf{F}$. Pak provedeme $k_{rr} := \kappa k_{rr}, F_r := \kappa k_{rr} g_{ij}$, kde κ je velké číslo, např. $\kappa = 10^{20}$. To způsobí, že mimodiagonální koeficienty v r-té rovnici budou oproti velkému diagonálnímu koeficientu prakticky zanedbatelné, takže r-tá rovnice nabude přibližně tvaru $\kappa k_{rr} u_r \doteq \kappa k_{rr} g_{ij}$ nebo-li $u_r \doteq g_{ij}$, což jsme potřebovali zajistit.

10.1.3. Metoda konečných objemů

Metodu vysvětlíme pro konvekčně-difúzní úlohu (10.15), (10.2), (10.3).

Pravidelná síť. Uvažujme nejdříve případ, kdy $\Omega = (0, a) \times (0, b)$ je obdélník. Na Ω zvolme uzly $[x_i, y_j] = [ih, jk]$, $i = 0, 1, \ldots, N$, $j = 0, 1, \ldots, M$, kde $h = a/N$, $k = b/M$.



Obr. 10.2. Vnitřní buňka B_{ij}, hraniční buňka B_{Nj} a rohová buňka B_{NM}

Obdélník

$$B_{ij} = \left[\langle x_{i-1/2}, x_{i+1/2}\rangle \times \langle y_{j-1/2}, y_{j+1/2}\rangle\right] \cap \overline{\Omega}\,, \qquad i = 0, 1, \ldots, N\,, \quad j = 0, 1, \ldots, M,$$

nazveme *konečným objemem* (stručně *buňkou*), viz Obr. 10.2. Integrací diferenciální rovnice (10.15) přes buňku B_{ij} dostaneme bilanční rovnici

$$\int_{B_{ij}} \left[-(pu_x - r_1 u)_x - (pu_y - r_2 u)_y + qu\right] \mathrm{d}x\,\mathrm{d}y = \int_{B_{ij}} f\,\mathrm{d}x\,\mathrm{d}y\,. \tag{10.21}$$

Uvažujme nejdříve případ, kdy B_{ij} je vnitřní buňka, tj. když $0 < i < N$ a $0 < j < M$. Pak $B_{ij} = \langle x_{i-1/2}, x_{i+1/2}\rangle \times \langle y_{j-1/2}, y_{j+1/2}\rangle$, viz Obr.10.2. Integrací per-partes obdržíme

$$\begin{aligned}
&-\int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_{i+1/2}, y)\,\mathrm{d}y + \int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_{i-1/2}, y)\,\mathrm{d}y - \\
&-\int_{x_{i-1/2}}^{x_{i+1/2}} [pu_y - r_2 u]\,(x, y_{j+1/2})\,\mathrm{d}x + \int_{x_{i-1/2}}^{x_{i+1/2}} [pu_y - r_2 u]\,(x, y_{j-1/2})\,\mathrm{d}x + \qquad (10.22)\\
&+\int_{B_{ij}} qu\,\mathrm{d}x\,\mathrm{d}y = \int_{B_{ij}} f\,\mathrm{d}x\,\mathrm{d}y\,.
\end{aligned}$$

Ukážeme si, jak provést aproximaci prvního členu v (10.22), zbývající jednoduché integrály se aproximují podobně. Pomocí obdélníkové formule dostaneme

$$-\int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_{i+1/2}, y)\,\mathrm{d}y \approx k[-p_{i+1/2,j} u_x(x_{i+1/2}, y_j) + [r_1 u](x_{i+1/2}, y_j)]\,,$$

derivaci $u_x(x_{i+1/2}, y_j)$ aproximujeme pomocí centrální diference

$$u_x(x_{i+1/2}, y_j) \approx [u(x_{i+1}, y_j) - u(x_i, y_j)]/h$$

a $[r_1 u](x_{i+1/2}, y_j)$ aproximujeme stejně jako v diferenční metodě, viz (10.17)–(10.19).

Dvojné integrály v (10.22) aproximujeme pomocí součinové obdélníkové formule:

$$\int_{B_{ij}} qu\,\mathrm{d}x\,\mathrm{d}y \approx hkq_{ij}u(x_i, y_j)\,, \qquad \int_{B_{ij}} f\,\mathrm{d}x\,\mathrm{d}y \approx hkf_{ij}\,.$$

Po dosazení do (10.22) dospějeme ke stejné rovnici, jakou bychom dostali pomocí diferenční metody.

Věnujme se nyní případu, kdy uzel $[x_i, y_j]$ leží na hranici $\partial\Omega$, tj. když $i = 0$ nebo $i = N$ nebo $j = 0$ nebo $j = M$. Jestliže uzel $[x_i, y_j]$ leží na části $\overline{\Gamma}_1$ hranice a je v něm tedy předepsána hodnota $u(x_i, y_j) = g(x_i, y_j)$, pak bilanční rovnici (10.21) pro takový uzel vůbec nesestavujeme. Uvažujme tedy případ, kdy uzel $[x_i, y_j]$ je vnitřním bodem hranice Γ_2. Pro konkrétnost budeme uvažovat vnitřní uzel východní strany obdélníka Ω, tj. uzel $[x_N, y_j]$, $0 < j < M$, pro který $B_{Nj} = \langle x_{N-1/2}, x_N\rangle \times \langle y_{j-1/2}, y_{j+1/2}\rangle$, viz Obr. 10.2. Bilanční rovnici (10.21) upravíme integrací per-partes,

$$\begin{aligned}
&-\int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_N, y)\,\mathrm{d}y + \int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_{N-1/2}, y)\,\mathrm{d}y - \\
&-\int_{x_{N-1/2}}^{x_N} [pu_y - r_2 u]\,(x, y_{j+1/2})\,\mathrm{d}x + \int_{x_{N-1/2}}^{x_N} [pu_y - r_2 u]\,(x, y_{j-1/2})\,\mathrm{d}x + \qquad (10.23)\\
&+\int_{B_{Nj}} qu\,\mathrm{d}x\,\mathrm{d}y = \int_{B_{Nj}} f\,\mathrm{d}x\,\mathrm{d}y\,.
\end{aligned}$$

První integrál v (10.23) upravíme užitím Newtonovy okrajové podmínky (10.12e),

$$-\int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_N, y)\,\mathrm{d}y = \int_{y_{j-1/2}}^{y_{j+1/2}} [\alpha^e(y) u(x_N, y) - \beta^e(y) + [r_1 u](x_N, y)]\,\mathrm{d}y\,,$$

a pak použijeme obdélníkovou formuli, takže

$$-\int_{y_{j-1/2}}^{y_{j+1/2}} [pu_x - r_1 u]\,(x_N, y)\,\mathrm{d}y \approx k[(\alpha_j^e + [r_1]_{Nj})u(x_N, y_j) - \beta_j^e]\,.$$

Druhý integrál v (10.23) aproximujeme stejně jako obdobný integrál ve vnitřní buňce. Třetí integrál v (10.23) aproximujeme užitím jednostranné obdélníkové formule

$$-\int_{x_{N-1/2}}^{x_N} [pu_y - r_2 u]\,(x, y_{j+1/2})\,\mathrm{d}x \approx -\tfrac{1}{2}h[pu_y - r_2 u](x_N, y_{j+1/2})\,,$$

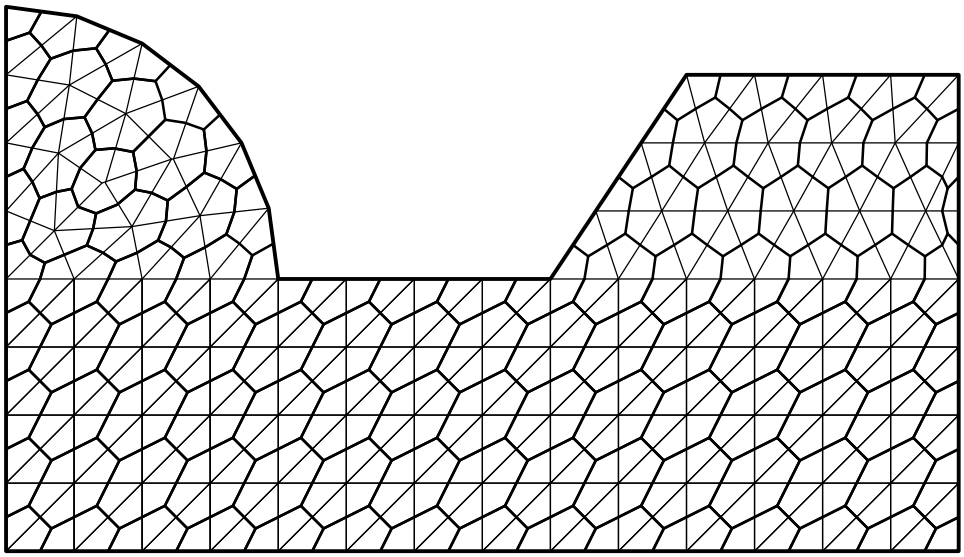
zbývající aproximace se provedou stejně jako ve vnitřní buňce. Čtvrtý integrál v (10.23) aproximujeme obdobně jako třetí integrál. Poslední dva integrály v (10.23) aproximujeme užitím jednostranné součinové obdélníkové formule

$$\int_{B_{Nj}} qu\,\mathrm{d}x\,\mathrm{d}y \approx \tfrac{1}{2}hkq_{Nj}u(x_N, y_j), \qquad \int_{B_{Nj}} f\,\mathrm{d}x\,\mathrm{d}y \approx \tfrac{1}{2}hkf_{Nj}\,.$$

Po dosazení do (10.22) opět dojdeme ke stejné rovnici, jakou bychom dostali pomocí diferenční metody.

Diskretizaci bilanční rovnice pro rohové konečné objemy lze provést pomocí dříve uvedených obratů a proto zde již tuto diskretizaci neuvádíme.

Obecnější síť buněk. Mnohoúhelník $\overline{\Omega}$ vyjádříme jako sjednocení konečného počtu uzavřených trojúhelníků, z nichž každé dva jsou buďto disjunktní nebo mají společný vrchol nebo stranu. Vrcholy trojúhelníků nazveme uzly. Soubor všech trojúhelníků vytváří tzv. *triangulaci* oblasti Ω, v metodě konečných objemů označovanou jako *primární síť*, viz Obr. 10.3. Ke každému uzlu P_i přiřadíme buňku B_i. Sestavíme ji z přilehlých částí C_{ijk} všech trojúhelníků s vrcholem P_i. Objasněme si to podrobněji.



Obr. 10.3. Duální síť

Nechť T_{ijk} je trojúhelník s vrcholy P_i, P_j a P_k. Označme P_{ij} střed strany $\overline{P_iP_j}$, P_{ik} střed strany $\overline{P_iP_k}$ a P_{ijk} těžiště trojúhelníka T_{ijk}. Pak do buňky B_i zahrneme čtyřúhelník C_{ijk} s vrcholy P_i, P_{ij}, P_{ijk} a P_{ik}, viz Obr. 10.4. Tento postup opakujeme pro všechny trojúhelníky T_{ijk}, které obsahují uzel P_i, a sjednocením přilehlých částí C_{ijk} dostaneme buňku B_i. Vnitřní buňka B_i příslušná vnitřnímu uzlu $P_i \notin \partial\Omega$ je zakreslena na Obr. 10.5, hraniční buňka pro $P_i \in \partial\Omega$ pak na Obr. 10.6. Množina všech buněk se nazývá *duální síť*, viz Obr. 10.3.

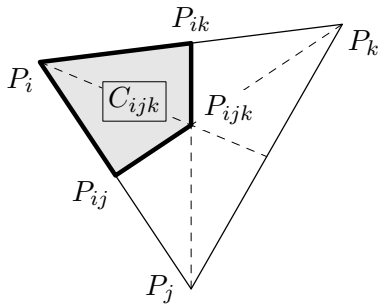
Diskretizace vychází opět z bilanční rovnice (10.21). Pomocí Gauss-Ostrogradského věty

$$\int_{B_i} \operatorname{div} \mathbf{w}\,\mathrm{d}x\,\mathrm{d}y = \int_{\partial B_i} (\mathbf{w}\cdot\mathbf{n})\,\mathrm{d}s, \quad \text{kde} \quad \mathbf{w} = \begin{pmatrix} pu_x - r_1 u \\ pu_y - r_2 u \end{pmatrix},$$

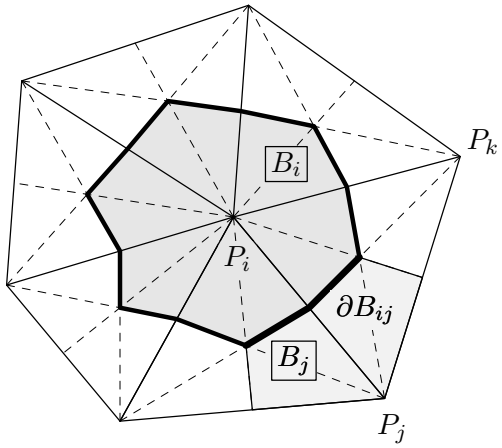
dostaneme analog rovnice (10.22),

$$\int_{\partial B_i} \left[-p\frac{\partial u}{\partial n_i} + (\mathbf{r}\cdot\mathbf{n}_i)u \right] \mathrm{d}s + \int_{B_i} qu\,\mathrm{d}x\,\mathrm{d}y = \int_{B_i} f\,\mathrm{d}x\,\mathrm{d}y\,, \tag{10.22'}$$

kde $\mathbf{r}\cdot\mathbf{n}_i = r_1 n_1 + r_2 n_2$ a $\mathbf{n}_i = (n_1^i, n_2^i)^T$ je jednotkový vektor vnější normály hranice ∂B_i buňky B_i.



Obr. 10.4. $C_{ijk} = B_i \cap T_{ijk}$



Obr. 10.5. Vnitřní buňka B_i

Klíčová je aproximace křivkového integrálu. Věnujme se nejdříve případu, kdy B_i je vnitřní buňka. Hranici ∂B_i vyjádříme jako sjednocení společných částí $\partial B_{ij} = \partial B_i \cap \partial B_j$ buňky B_i a sousedních buněk B_j, tj. $\partial B_i = \bigcup_j \partial B_{ij}$. Protože $\int_{\partial B_i} = \sum_j \int_{\partial B_{ij}}$, stačí popsat aproximaci jen na ∂B_{ij}. To lze provést třeba takto:

$$\int_{\partial B_{ij}} \left[-p\frac{\partial u}{\partial n_i} + (\mathbf{r}\cdot\mathbf{n}_i)u \right] \mathrm{d}s \approx$$

$$|\partial B_{ij}| \left[p(P_{ij}) \frac{u(P_i) - u(P_j)}{|\overline{P_iP_j}|} + (\mathbf{r}\cdot\mathbf{n})_{ij} u(P_{ij}) \right],$$

kde $|\partial B_{ij}|$ je délka ∂B_{ij}, $|\overline{P_iP_j}|$ je délka úsečky $\overline{P_iP_j}$,

$$(\mathbf{r}\cdot\mathbf{n})_{ij} = \mathbf{r}(P_{ij})\cdot\mathbf{n}_{ij} = r_1(P_{ij})n_1^{ij} + r_2(P_{ij})n_2^{ij} \quad \text{a} \quad \mathbf{n}_{ij} = (n_1^{ij}, n_2^{ij})^T = (P_j - P_i)/|\overline{P_iP_j}|$$

je jednotkový vektor ve směru vektoru $\overrightarrow{P_iP_j}$. Zbývá aproximovat hodnotu $u(P_{ij})$ ve středu úsečky P_iP_j. Jestliže je dominantní difúze, třeba když

$$\tfrac{1}{2}|\overline{P_iP_j}||(\mathbf{r}\cdot\mathbf{n})_{ij}| < p(P_{ij})\,,$$

zvolíme aritmetický průměr

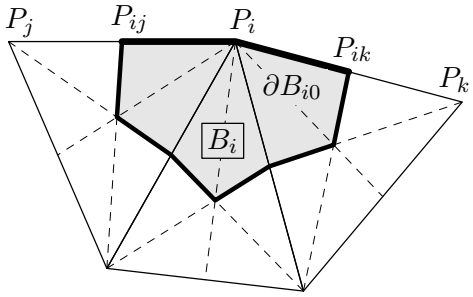
$$u(P_{ij}) \approx \tfrac{1}{2}(u(P_i) + u(P_j))\,,$$

v opačném případě užijeme upwind aproximaci,

$$u(P_{ij}) \approx \begin{cases} u(P_i), \\ u(P_j), \end{cases} \quad \text{pokud} \quad (\mathbf{r}\cdot\mathbf{n})_{ij} \quad \begin{cases} \geq 0, \\ < 0. \end{cases}$$

Dvojné integrály aproximujeme jako součin obsahu $|B_{ij}|$ buňky B_{ij} a hodnoty integrandu v bodu P_i, tj.

$$\int_{B_i} qu\,\mathrm{d}x\,\mathrm{d}y \approx |B_i|q(P_i)u(P_i)\,, \qquad \int_{B_i} f\,\mathrm{d}x\,\mathrm{d}y \approx |B_i|f(P_i)\,.$$



Obr. 10.6. Hraniční buňka B_i

Pro uzel P_i ležící na hranici Γ_1 buňku B_i nepotřebujeme a klademe $u(P_i) = g(P_i)$. Pro vnitřní uzel P_i hranice Γ_2 sestrojíme hraniční buňku, viz Obr. 10.6. Hranice $\partial B_i = \bigcup_j \partial B_{ij} \cup \partial B_{i0}$ je sjednocením společných částí $\partial B_{ij} = \partial B_i \cap \partial B_j$ buňky B_i a sousedních buněk B_j a dále části $\partial B_{i0} = \partial B_i \cap \Gamma_2$ hranice buňky B_i ležící na hranici Γ_2, $\partial B_{i0} = \overline{P_iP_{ij}} \cup \overline{P_iP_{ik}}$ na Obr. 10.6.

Při aproximaci $\int_{\partial B_{i0}}$ využijeme předepsanou Newtonovu okrajovou podmínku (10.3),

$$\int_{\partial B_{i0}} \left[-p\frac{\partial u}{\partial n_i} + (\mathbf{r}\cdot\mathbf{n}_i)u\right]\mathrm{d}s = \int_{\partial B_{i0}} [\alpha u - \beta + (\mathbf{r}\cdot\mathbf{n}_i)u]\,\mathrm{d}s \approx$$
$$|\partial B_{i0}|\,[\alpha(P_i)u(P_i) - \beta(P_i)] + \mathbf{r}(P_i)\cdot\big[\,|\overline{P_iP_{ij}}|\mathbf{n}_{ij}^i + |\overline{P_iP_{ik}}|\mathbf{n}_{ik}^i\,\big]u(P_i),$$

kde $\mathbf{n}_{ij}^i$ resp. $\mathbf{n}_{ik}^i$ je vektor vnější normály buňky B_i na straně $\overline{P_iP_j}$ resp. $\overline{P_iP_k}$. Zbývající aproximace se provedou stejně jako u vnitřní buňky.

Více podrobností o metodě konečných objemů, včetně přesnější varianty upwind aproximace konvekčního členu, lze najít např. v [23].

10.1.4. Metoda konečných prvků

Metodu vysvětlíme pro konvekčně-difúzní úlohu (10.15), (10.2), (10.3).

Slabé řešení. Stejně jako v kapitole 9.4 úlohu převedeme na tvar vhodný pro nasazení MKP, tj. odvodíme slabou formulaci naší úlohy. K tomu účelu násobíme rovnici (10.1) testovací funkcí $v \in C^1(\overline{\Omega})$ s vlastností $v = 0$ na Γ_1 a integrujeme přes oblast Ω, tj. provedeme

$$-\int_\Omega [(pu_x - r_1 u)_x + (pu_y - r_2 u)_y] v \,\mathrm{d}x\,\mathrm{d}y + \int_\Omega quv \,\mathrm{d}x\,\mathrm{d}y = \int_\Omega fv \,\mathrm{d}x\,\mathrm{d}y\,. \tag{10.24}$$

První člen na levé straně upravíme pomocí Gauss-Ostrogradského věty

$$\int_\Omega \operatorname{div} \mathbf{w} \,\mathrm{d}x\,\mathrm{d}y = \int_{\partial\Omega} (\mathbf{w}\cdot\mathbf{n})\,\mathrm{d}s, \quad \text{kde} \quad \mathbf{w} = \begin{pmatrix} (pu_x - r_1 u)v \\ (pu_y - r_2 u)v \end{pmatrix},$$

na tvar

$$-\int_\Omega [(pu_x - r_1 u)_x + (pu_y - r_2 u)_y] v \,\mathrm{d}x\,\mathrm{d}y = -\int_{\partial\Omega} [(pu_x - r_1 u)n_1 + (pu_y - r_2 u)n_2] v \,\mathrm{d}s +$$
$$\int_\Omega [(pu_x - r_1 u)v_x + (pu_y - r_2 u)v_y] \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}x\,\mathrm{d}y\,.$$

Křivkový integrál dále upravíme: využijeme toho, že $v = 0$ na Γ_1 a že na Γ_2 platí okrajová podmínka (10.3). Tak dostaneme

$$-\int_{\partial\Omega} [(pu_x - r_1 u)n_1 + (pu_y - r_2 u)n_2] v \,\mathrm{d}s = \int_{\Gamma_2} \left[-p\frac{\partial u}{\partial n} + (r_1 n_1 + r_2 n_2)u \right] v \,\mathrm{d}s =$$
$$\int_{\Gamma_2} [\alpha u - \beta + (\mathbf{r}\cdot\mathbf{n})u] v \,\mathrm{d}s\,.$$

Dosadíme-li z posledních dvou rovností do (10.24) vidíme, že řešení u úlohy (10.15), (10.2), (10.3) splňuje rovnici

$$\int_\Omega [p(u_x v_x + u_y v_y) - u(r_1 v_x + r_2 v_y) + quv] \,\mathrm{d}x\,\mathrm{d}y + \int_{\Gamma_2} (\alpha + \mathbf{r}\cdot\mathbf{n})uv] \,\mathrm{d}s =$$
$$\int_\Omega fv \,\mathrm{d}x\,\mathrm{d}y + \int_{\Gamma_2} \beta v \,\mathrm{d}s \tag{10.25}$$

pro každou funkci $v \in C^1(\overline{\Omega})$ s vlastností $v = 0$ na Γ_1. Rovnice (10.25) má zřejmě smysl i v případě, když funkce u a v jsou jen z prostoru $X \equiv PC^1(\overline{\Omega})$. Testovací funkce tedy volíme z *prostoru* $V = \{v \in X \,|\, v = 0 \text{ na } \overline{\Gamma}_1\}$ *testovacích funkcí* a řešení u z *množiny* $W = \{v \in X \,|\, v = g \text{ na } \overline{\Gamma}_1\}$ *přípustných řešení*. Označíme-li

$$\begin{aligned} a(u,v) &= \int_\Omega [\, p\,(\nabla u\cdot\nabla v) - u\,(\mathbf{r}\cdot\nabla v) + quv] \,\mathrm{d}x\,\mathrm{d}y + \int_{\Gamma_2} (\alpha + \mathbf{r}\cdot\mathbf{n})uv \,\mathrm{d}s\,, \\ L(v) &= \int_\Omega fv \,\mathrm{d}x\,\mathrm{d}y + \int_{\Gamma_2} \beta v \,\mathrm{d}s, \end{aligned} \tag{10.26}$$

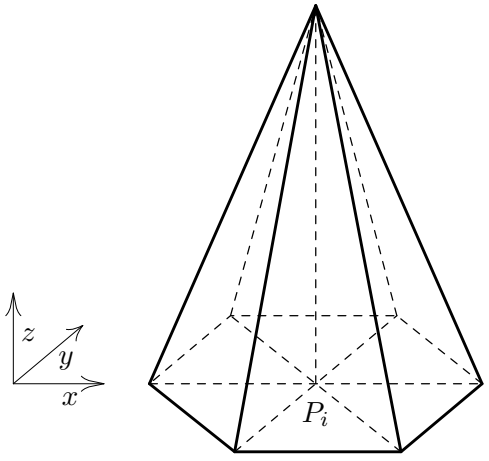
pak úlohu

$$\text{najít } u \in W \text{ splňující } a(u,v) = L(v) \quad \forall v \in V \tag{10.27}$$

nazveme *slabou formulací* úlohy (10.15), (10.2), (10.3) a funkci u nazveme *slabým řešením*.

Diskretizace. Omezíme se na případ, že Ω je mnohoúhelník. $\overline{\Omega}$ vyjádříme jako sjednocení konečného počtu uzavřených trojúhelníků T_e, z nichž každé dva jsou buďto disjunktní nebo mají společný vrchol nebo společnou stranu, viz Obr. 10.3. Množinu $\mathcal{T}$ všech trojúhelníků nazveme triangulací oblasti Ω. Trojúhelníky budeme značit $T_1, T_2, \ldots, T_{N_T}$. Vrcholy trojúhelníků budeme nazývat uzly, značíme je $P_1, P_2, \ldots, P_M$. Předpokládejme, že společné body částí $\overline{\Gamma}_1$ a $\overline{\Gamma}_2$ hranice Γ jsou uzly triangulace $\mathcal{T}$. Množinu stran trojúhelníků $T \in \mathcal{T}$, jejichž sjednocení je $\overline{\Gamma}_2$, označíme jako $\mathcal{S}$. Strany budeme značit $S_1, S_2, \ldots, S_{N_S}$. Nejdelší stranu trojúhelníků triangulace $\mathcal{T}$ označíme jako h.

Funkci, která je v $\overline{\Omega}$ spojitá a která je na každém trojúhelníku $T \in \mathcal{T}$ lineární, nazveme *spojitou po částech lineární funkcí*. Každá taková funkce $v(x,y)$ je jednoznačně určena svými hodnotami $v(x_i, y_i)$ ve vrcholech $P_i \equiv [x_i, y_i]$ triangulace $\mathcal{T}$. Prostor všech spojitých po částech lineárních funkcí označíme X_h. Speciálním případem funkcí z X_h jsou bázové funkce $w_i(x,y)$, které jsou v P_i rovny jedné a v ostatních uzlech jsou rovny nule, tj.



Obr. 10.7. Bázová funkce $w_i(x,y)$

$$w_i(P_j) = \begin{cases} 1 & \text{pro } \begin{cases} i = j, \\ i \neq j. \end{cases} \\ 0 \end{cases}$$

Každá funkce $v \in X_h$ může být pomocí svých hodnot v uzlech a pomocí bázových funkcí vyjádřena ve tvaru

$$v(x,y) = \sum_{i=1}^{M} v(x_i, y_i) w_i(x,y)\,.$$

Dále definujme prostor testovacích funkcí

$$V_h = \{v \in X_h \mid v(x_i, y_i) = 0 \ \forall P_i \in \overline{\Gamma}_1\}$$

a množinu přípustných řešení

$$W_h = \{v \in X_h \mid v(x_i, y_i) = g(x_i, y_i) \ \forall P_i \in \overline{\Gamma}_1\}.$$

Trojúhelník, na němž je definována lineární funkce jednoznačně určená svými hodnotami ve vrcholech, se nazývá *Lagrangeův lineární trojúhelníkový prvek*.

Nyní už máme vše potřebné k dispozici: přibližné řešení U, tzv. *MKP řešení*, obdržíme z *diskrétní slabé formulace*

$$\text{najít } U \in W_h \text{ splňující } a_h(U,v) = L_h(v) \quad \forall v \in V_h\,, \tag{10.28}$$

kde

$$a_h(U,v) = \sum_{T_e \in \mathcal{T}} \left[Q^{T_e}(p\,(\nabla U \cdot \nabla v)) + Q^{T_e}(-U\,(\mathbf{r} \cdot \nabla v)) + Q^{T_e}(qUv) \right] + \sum_{S_e \in \mathcal{S}} Q^{S_e}((\alpha + \mathbf{r} \cdot \mathbf{n})\,Uv),$$

$$L_h(v) = \sum_{T_e \in \mathcal{T}} Q^{T_e}(fv) + \sum_{S_e \in \mathcal{S}} Q^{S_e}(\beta v). \tag{10.29}$$

Přitom symbolem $Q^{T_e}(\varphi)$ jsme označili kvadraturní formuli pro výpočet $\int_{T_e} \varphi \,\mathrm{d}x\,\mathrm{d}y$ na trojúhelníku T_e a symbolem $Q^{S_e}(\varphi)$ formuli pro výpočet $\int_{S_e} \varphi \,\mathrm{d}s$ na straně S_e. Jako vhodné kvadraturní formule lze doporučit:

1) členy $p(\nabla U \cdot \nabla v)$ a $U(\mathbf{r}\cdot\nabla v)$ integrujeme na trojúhelníku T formulí

$$Q^T(\varphi) = |T|\varphi(P_0)\,, \tag{10.30}$$

kde $|T|$ je plocha trojúhelníka T, $P_0 = \frac{1}{3}(P_1+P_2+P_3)$ je těžiště T a P_1, P_2, P_3 jsou vrcholy T;

2) členy qUv a fv integrujeme na trojúhelníku T formulí

$$Q^T(\varphi) = \tfrac{1}{3}|T|\left[\varphi(P_1)+\varphi(P_2)+\varphi(P_3)\right]\,; \tag{10.31}$$

3) členy $(\alpha + \mathbf{r}\cdot\mathbf{n})Uv$ a βv integrujeme na straně S lichoběžníkovou formulí

$$Q^S(\varphi) = \tfrac{1}{2}|S|[\varphi(P_1)+\varphi(P_2)]\,, \tag{10.32}$$

kde $|S|$ je délka strany S a P_1, P_2 jsou koncové body S.

Formule (10.30), (10.31) a (10.32) jsou řádu 1, tj. jsou přesné, když φ je polynom stupně 1.

Předpokládejme, že uzly jsou očíslovány tak, že $P_1, P_2, \ldots, P_N$ leží buďto uvnitř oblasti Ω nebo uvnitř hranice Γ_2, a že $P_{N+1}, P_{N+2}, \ldots, P_M$ leží na hranici $\overline{\Gamma}_1$. To není žádné omezení, neboť uzly lze vždy přečíslovat tak, aby byl tento předpoklad splněn. Pak

$$U(x,y) = \sum_{j=1}^{N} \Delta_j w_j(x,y) + \sum_{j=N+1}^{M} g_j w_j(x,y)\,, \qquad v(x,y) = \sum_{i=1}^{N} \Theta_i w_i(x,y)\,, \tag{10.33}$$

kde $\Delta_j = U(P_j)$, $g_j = g(P_j)$ a $\Theta_i = v(P_i)$. Dosazením do (10.28) obdržíme

$$\begin{aligned} 0 =& a_h(U,v) - L_h(v) = a_h\left(\sum_{j=1}^{N}\Delta_j w_j + \sum_{j=N+1}^{M} g_j w_j, \sum_{i=1}^{N}\Theta_i w_i\right) - L_h\left(\sum_{i=1}^{N}\Theta_i w_i\right) = \\ =& \sum_{i=1}^{N}\Theta_i \sum_{j=1}^{N} a_h(w_j,w_i)\Delta_j - \sum_{i=1}^{N}\Theta_i\left[L_h(w_i) - \sum_{j=N+1}^{M} a_h(w_j,w_i)g_j\right] = \\ =& \boldsymbol{\theta}^T\left[\mathbf{K}\boldsymbol{\Delta} - \mathbf{F}\right], \end{aligned} \tag{10.34}$$

kde $\boldsymbol{\theta} = (\Theta_1, \Theta_2, \ldots, \Theta_N)^T$, $\mathbf{K} = \{k_{ij}\}_{i,j=1}^{N}$ pro $k_{ij} = a_h(w_j, w_i)$, $\boldsymbol{\Delta} = (\Delta_1, \Delta_2, \ldots, \Delta_N)^T$ a $\mathbf{F} = (F_1, F_2, \ldots, F_N)^T$ pro $F_i = L_h(w_i) - \sum_{j=N+1}^{M} a_h(w_j, w_i) g_j$. Protože $\boldsymbol{\theta}$ je libovolný vektor, musí platit

$$\mathbf{K}\boldsymbol{\Delta} = \mathbf{F}. \tag{10.35}$$

Matice $\mathbf{K}$ je řídká a při vhodném očíslování uzlů je pásová. Pro $\mathbf{r} = \mathbf{o}$ je $\mathbf{K}$ pozitivně definitní a tedy regulární. Pro $\mathbf{r} \neq \mathbf{o}$ je $\mathbf{K}$ nesymetrická, postačující podmínkou regularity

matice $\mathbf{K}$ je dostatečně malé h neboli dostatečně jemná triangulace. Vyřešením soustavy lineárních rovnic (10.35) získáme $\Delta_j = U(P_j)$, $j = 1, 2, \ldots, N$.

Za předpokladu dostatečné hladkosti slabého řešení u pro chybu platí

$$u - U = O(h^2)\,, \qquad u_x - U_x = O(h)\,, \quad u_y - U_y = O(h). \tag{10.36}$$

Algoritmus. Matici $\mathbf{K}$ a vektor $\mathbf{F}$ sestavíme pomocí elementárních matic $\mathbf{K}^{T_e}$ a elementárních vektorů $\mathbf{F}^{T_e}$ pro $T_e \in \mathcal{T}$ a elementárních matic $\mathbf{K}^{S_e}$ a elementárních vektorů $\mathbf{F}^{S_e}$ pro $S_e \in \mathcal{S}$. Matici $\mathbf{K}$ budeme nazývat také globální maticí tuhosti a vektor $\mathbf{F}$ globálním vektorem zatížení.

Elementární matice a elementární vektor na trojúhelníku. Nechť `ELEM` je tabulka typu $N_T \times 3$, která v řádku e obsahuje čísla vrcholů trojúhelníka T_e. Uvažme jeden konkrétní trojúhelník T_e triangulace $\mathcal{T}$ s vrcholy $P_1^e = [x_1^e, y_1^e]$, $P_2^e = [x_2^e, y_2^e]$ a $P_3^e = [x_3^e, y_3^e]$. Pro uzel P_r^e, $r = 1, 2, 3$, je r lokálním číslem uzlu na trojúhelníku T_e. Globálním číslem uzlu P_r^e je číslo $i = \texttt{ELEM}(e, r)$. P_r^e a P_i jsou tedy jen různá označení téhož uzlu.

Řešení U a testovací funkce v je na trojúhelníku T_e tvaru

$$\begin{aligned} U(x,y)\Big|_{T_e} &\equiv U^e(x,y) = \Delta_1^e w_1^e(x,y) + \Delta_2^e w_2^e(x,y) + \Delta_3^e w_3^e(x,y)\,, \\ v(x,y)\Big|_{T_e} &\equiv v^e(x,y) = \Theta_1^e w_1^e(x,y) + \Theta_2^e w_2^e(x,y) + \Theta_3^e w_3^e(x,y)\,, \end{aligned} \tag{10.37}$$

kde $\Delta_r^e = U(P_r^e)$, $\Theta_r^e = v(P_r^e)$ a kde

$$w_r^e(x,y) = a_r^e x + b_r^e y + c_r^e \tag{10.38}$$

je bázová funkce příslušná k uzlu P_r^e, $r = 1, 2, 3$. Z (10.37) a (10.38) dostaneme

$$\nabla U^e = \begin{pmatrix} \partial U^e/\partial x \\ \partial U^e/\partial y \end{pmatrix} = \mathbf{B}^e \boldsymbol{\Delta}^{T_e}, \quad \nabla v^e = \begin{pmatrix} \partial v^e/\partial x \\ \partial v^e/\partial y \end{pmatrix} = \mathbf{B}^e \boldsymbol{\theta}^{T_e},$$

$$\text{kde} \quad \mathbf{B}^e = \begin{pmatrix} a_1^e & a_2^e & a_3^e \\ b_1^e & b_2^e & b_3^e \end{pmatrix} \quad \text{a} \quad \boldsymbol{\Delta}^{T_e} = \begin{pmatrix} \Delta_1^e \\ \Delta_2^e \\ \Delta_3^e \end{pmatrix}, \quad \boldsymbol{\theta}^{T_e} = \begin{pmatrix} \Theta_1^e \\ \Theta_2^e \\ \Theta_3^e \end{pmatrix}$$

jsou vektory parametrů na trojúhelníku T_e. Koeficienty a_r^e a b_r^e lze snadno spočítat z rovnic

$$w_r^e(P_s^e) = \begin{cases} 1 \\ 0 \end{cases} \quad \text{pro} \quad \begin{cases} r = s\,, \\ r \neq s\,, \end{cases} \qquad r, s = 1, 2, 3,$$

nebo-li maticově

$$\begin{pmatrix} x_1^e & y_1^e & 1 \\ x_2^e & y_2^e & 1 \\ x_3^e & y_3^e & 1 \end{pmatrix} \begin{pmatrix} a_1^e & a_2^e & a_3^e \\ b_1^e & b_2^e & b_3^e \\ c_1^e & c_2^e & c_3^e \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Označíme-li

$$\mathbf{D}^e = \begin{pmatrix} x_1^e & y_1^e & 1 \\ x_2^e & y_2^e & 1 \\ x_3^e & y_3^e & 1 \end{pmatrix}, \quad \text{pak} \quad \mathbf{B}^e = [\mathbf{D}^e]^{-1}_{[1:2,1:3]},$$

tj. matice $\mathbf{B}^e$ je rovna prvním dvěma řádkům matice inverzní k matici $\mathbf{D}^e$.

Nechť $P_0^e = \frac{1}{3}(P_1^e + P_2^e + P_3^e)$ je těžiště trojúhelníka T_e. Užitím kvadraturní formule (10.30) pak obdržíme

$$Q^{T_e}(p\,(\nabla U \cdot \nabla v)) = |T_e|\,p(P_0^e)\left(\mathbf{B}^e\boldsymbol{\theta}^{T_e}\right)^T\mathbf{B}^e\boldsymbol{\Delta}^{T_e} = [\boldsymbol{\theta}^{T_e}]^T\mathbf{K}^{T_e,1}\boldsymbol{\Delta}^{T_e}, \tag{10.39}$$

kde elementární matice

$$\mathbf{K}^{T_e,1} = |T_e|\,p_0^e\,[\mathbf{B}^e]^T\mathbf{B}^e \tag{10.40}$$

a

$$|T_e| = \tfrac{1}{2}|\det(\mathbf{D}^e)|$$

je plocha trojúhelníka T_e.

Označíme-li $\mathbf{w}^e = (w_1^e, w_2^e, w_3^e)^T$, pak $U^e = [\mathbf{w}^e]^T\boldsymbol{\Delta}^{T_e}$ podle (10.37). Protože

$$-U^e(\mathbf{r}\cdot\nabla v^e) = -[\nabla v^e]^T\mathbf{r}\,U^e = -[\mathbf{B}^e\boldsymbol{\theta}^{T_e}]^T\mathbf{r}[\mathbf{w}^e]^T\boldsymbol{\Delta}^{T_e} = -[\boldsymbol{\theta}^{T_e}]^T[\mathbf{B}^e]^T\mathbf{r}[\mathbf{w}^e]^T\boldsymbol{\Delta}^{T_e},$$

pomocí kvadraturní formule (10.30) dostaneme

$$Q^{T_e}(-U\,(\mathbf{r}\cdot\nabla v)) = [\boldsymbol{\theta}^{T_e}]^T\left[-|T_e|[\mathbf{B}^e]^T\mathbf{r}(P_0^e)(\tfrac{1}{3},\tfrac{1}{3},\tfrac{1}{3})\right]\boldsymbol{\Delta}^{T_e} = [\boldsymbol{\theta}^{T_e}]^T\mathbf{K}^{T_e,2}\boldsymbol{\Delta}^{T_e}, \tag{10.41}$$

kde elementární matice

$$\mathbf{K}^{T_e,2} = -\tfrac{1}{3}|T_e|[\mathbf{B}^e]^T\mathbf{r}_0^e(1,1,1) \equiv (\mathbf{s}^e,\mathbf{s}^e,\mathbf{s}^e) \quad \text{pro} \quad \mathbf{s}^e = -\tfrac{1}{3}|T_e|[\mathbf{B}^e]^T\mathbf{r}_0^e. \tag{10.42}$$

Užitím kvadraturní formule (10.31) dostaneme

$$\begin{aligned} Q^{T_e}(qUv) =& \tfrac{1}{3}|T_e|[q(P_1^e)U(P_1^e)v(P_1^e) + q(P_2^e)U(P_2^e)v(P_2^e) + q(P_3^e)U(P_3^e)v(P_1^e)] = \\ & \tfrac{1}{3}|T_e|[q(P_1^e)\Theta_1^e\Delta_1^e + q(P_2^e)\Theta_2^e\Delta_2^e + q(P_3^e)\Theta_3^e\Delta_3^e] = [\boldsymbol{\theta}^{T_e}]^T\mathbf{K}^{T_e,3}\boldsymbol{\Delta}^{T_e}, \end{aligned} \tag{10.43}$$

kde elementární matice

$$\mathbf{K}^{T_e,3} = \tfrac{1}{3}|T_e|\begin{pmatrix} q_1^e & 0 & 0 \\ 0 & q_2^e & 0 \\ 0 & 0 & q_3^e \end{pmatrix}. \tag{10.44}$$

Celkem tak

$$\sum_{T_e\in\mathcal{T}}\left\{Q^{T_e}(p\,(\nabla U\cdot\nabla v)) + Q^{T_e}(-U\,(\mathbf{r}\cdot\nabla v)) + Q^{T_e}(qUv)\right\} = [\boldsymbol{\theta}^{T_e}]^T\mathbf{K}^{T_e}\boldsymbol{\Delta}^{T_e}, \tag{10.45}$$

kde

$$\mathbf{K}^{T_e} = \mathbf{K}^{T_e,1} + \mathbf{K}^{T_e,2} + \mathbf{K}^{T_e,3} \tag{10.46}$$

je elementární matice na trojúhelníku T_e. Užitím kvadraturní formule (10.31) obdržíme

$$\begin{aligned} Q^{T_e}(fv) =& \tfrac{1}{3}|T_e|[f(P_1^e)v(P_1^e) + f(P_2^e)v(P_2^e) + f(P_3^e)v(P_3^e)] = \\ & \tfrac{1}{3}|T_e|[f(P_1^e)\Theta_1^e + f(P_2^e)\Theta_2^e + f(P_3^e)\Theta_3^e] = [\boldsymbol{\theta}^{T_e}]^T\mathbf{F}^{T_e}, \end{aligned} \tag{10.47}$$

kde

$$\mathbf{F}^{T_e} = \tfrac{1}{3}|T_e| \begin{pmatrix} f_1^e \\ f_2^e \\ f_3^e \end{pmatrix} \tag{10.48}$$

je elementární vektor na trojúhelníku T_e.

Elementární matice a elementární vektor na straně. Nechť `SIDE` je tabulka typu $N_S \times 2$, která v řádku e obsahuje čísla krajních bodů strany S_e. Uvažme konkrétní stranu S_e hranice Γ_2 s koncovými body $P_1^e = [x_1^e, y_1^e]$ a $P_2^e = [x_2^e, y_2^e]$. Pro uzel P_r^e, $r = 1, 2$, je r lokálním číslem uzlu na straně S_e. Globálním číslem uzlu P_r^e je číslo $i = \texttt{SIDE}(e, r)$. P_r^e a P_i jsou tedy jen různá označení téhož uzlu.

Řešení U a testovací funkce v je na straně S_e tvaru

$$U(x,y)\Big|_{S_e} \equiv U^e(x,y) = \Delta_1^e w_1^e(x,y) + \Delta_2^e w_2^e(x,y)\,,$$

$$v(x,y)\Big|_{S_e} \equiv v^e(x,y) = \Theta_1^e w_1^e(x,y) + \Theta_2^e w_2^e(x,y)\,,$$

kde $\Delta_r^e = U(P_r^e)$, $\Theta_r^e = v(P_r^e)$ a kde $w_r^e(x,y)$ je bázová funkce příslušná k uzlu P_r^e, $r = 1, 2$. Nechť $\mathbf{n}_e = (n_1^e, n_2^e)^T$ je jednotkový vektor vnější normály na straně S_e. Užitím formule (10.32) obdržíme

$$\begin{aligned} & Q^{S_e}((\alpha + \mathbf{r}\cdot\mathbf{n})\,Uv) = \tfrac{1}{2}|S_e|\{[(\alpha + \mathbf{r}\cdot\mathbf{n}_e)Uv](P_1^e) + [(\alpha + \mathbf{r}\cdot\mathbf{n}_e)Uv](P_2^e)\} = \\ & \tfrac{1}{2}|S_e|[\Theta_1^e(\alpha(P_1^e) + \mathbf{r}(P_1^e)\cdot\mathbf{n}_e]\Delta_1^e + \Theta_2^e(\alpha(P_2^e) + \mathbf{r}(P_2^e)\cdot\mathbf{n}_e)\Delta_2^e] = [\boldsymbol{\theta}^{S_e}]^T\mathbf{K}^{S_e}\boldsymbol{\Delta}^{S_e}, \end{aligned} \tag{10.49}$$

kde

$$\mathbf{K}^{S_e} = \tfrac{1}{2}|S_e| \begin{pmatrix} \alpha_1^e + \mathbf{r}_1^e\cdot\mathbf{n}_e & 0 \\ 0 & \alpha_2^e + \mathbf{r}_2^e\cdot\mathbf{n}_e \end{pmatrix} \tag{10.50}$$

je elementární matice na straně S_e a

$$\boldsymbol{\Delta}^{S_e} = \begin{pmatrix} \Delta_1^e \\ \Delta_2^e \end{pmatrix}, \qquad \boldsymbol{\theta}^{S_e} = \begin{pmatrix} \Theta_1^e \\ \Theta_2^e \end{pmatrix}$$

jsou vektory parametrů na straně S_e. Podobně odvodíme

$$\begin{aligned} Q^{S_e}(\beta v) =& \tfrac{1}{2}|S_e|[\beta(P_1^e)v(P_1^e) + \beta(P_2^e)v(P_2^e)] = \\ & \tfrac{1}{2}|S_e|[\beta(P_1^e)\Theta_1^e + \beta(P_2^e)\Theta_2^e] = [\boldsymbol{\theta}^{S_e}]^T\mathbf{F}^{S_e}, \end{aligned} \tag{10.51}$$

kde

$$\mathbf{F}^{S_e} = \tfrac{1}{2}|S_e| \begin{pmatrix} \beta_1^e \\ \beta_2^e \end{pmatrix} \tag{10.52}$$

je elementární vektor na straně S_e.

Sestavení soustavy rovnic. Zkombinujeme-li (10.34), (10.29), (10.45), (10.47), (10.49) a (10.51) vidíme, že pro MKP řešení U a libovolnou testovací funkci $v \in V_h$ platí

$$\begin{aligned} 0 &= a_h(U,v) - L_h(v) = \boldsymbol{\theta}^T \left[\mathbf{K}\boldsymbol{\Delta} - \mathbf{F}\right] = \\ &= \sum_{T_e \in \mathcal{T}} \left[\boldsymbol{\theta}^{T_e}\right]^T \left[\mathbf{K}^{T_e}\boldsymbol{\Delta}^{T_e} - \mathbf{F}^{T_e}\right] + \sum_{S_e \in \mathcal{S}} \left[\boldsymbol{\theta}^{S_e}\right]^T \left[\mathbf{K}^{S_e}\boldsymbol{\Delta}^{S_e} - \mathbf{F}^{S_e}\right]. \end{aligned}$$

Z této rovnosti lze odvodit pravidla pro sestavení globální matice $\mathbf{K}$ a globálního vektoru $\mathbf{F}$ z lokálních matic $\mathbf{K}^{T_e}$, $\mathbf{K}^{S_e}$ a lokálních vektorů $\mathbf{F}^{T_e}$, $\mathbf{F}^{S_e}$. Postupujeme podle následujícího algoritmu:

1) Matici $\mathbf{K}$ řádu N a sloupcový vektor $\mathbf{F}$ délky N naplníme nulami.

2) Pro každý trojúhelník $T_e \in \mathcal{T}$ sestavíme elementární matici $\mathbf{K}^{T_e} = \{k_{rs}^{T_e}\}_{r,s=1}^3$, viz (10.46), a elementární vektor $\mathbf{F}^{T_e} = \{F_r^{T_e}\}_{r=1}^3$, viz (10.48).

   Pro $r,s = 1,2,3$ položíme $i = \texttt{ELEM}(e,r)$, $j = \texttt{ELEM}(e,s)$ a

$$\text{pokud} \begin{cases} i \le N \text{ a } j \le N, \\ i \le N \text{ a } j > N, \\ i \le N, \end{cases} \quad \text{provedeme} \begin{cases} k_{ij} \leftarrow k_{ij} + k_{rs}^{T_e}, \\ F_i \leftarrow F_i - k_{rs}^{T_e} g(P_j), \\ F_i \leftarrow F_i + F_r^{T_e}. \end{cases}$$

3) Pro každou stranu $S_e \in \mathcal{S}$ sestavíme elementární matici $\mathbf{K}^{S_e} = \{k_{rs}^{S_e}\}_{r,s=1}^2$, viz (10.50), a elementární vektor $\mathbf{F}^{S_e} = \{F_r^{S_e}\}_{r=1}^2$, viz (10.52).

   Pro $r = 1,2$ položíme $i = \texttt{SIDE}(e,r)$ a

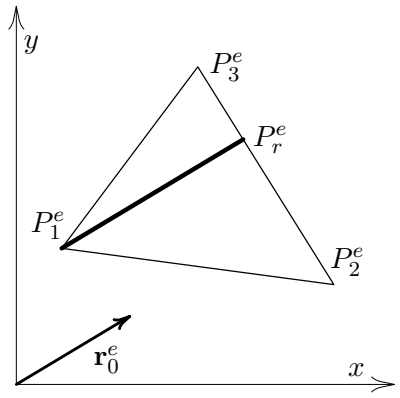
$$\text{pokud} \quad i \le N, \quad \text{provedeme} \quad k_{ii} \leftarrow k_{ii} + k_{rr}^{S_e}, \quad F_i \leftarrow F_i + F_r^{S_e}.$$

Dominantní konvekce. Jednou z možností, jak se vypořádat s dominantní konvekcí, je postup známý jako *metoda umělé směrové difúze*, zkratka *SD* (podle anglického *Streamline Diffusion*), viz [13]. Vyjdeme z modifikované diskrétní slabé formulace

$$\text{najít } U \in W_h \text{ splňující } a_h(U,v) + c_h(U,v) = L_h(v), \quad \forall v \in V_h, \tag{10.28'}$$

kde

$$c_h(U,v) = \sum_{i=1}^{N} Q^{T_e} \left(\delta_e \left(\mathbf{r}\cdot\nabla U\right)\left(\mathbf{r}\cdot\nabla v\right)\right).$$



Obr. 10.8. $h_r^e = |\overline{P_1^e P_r^e}|$

Nechť $|\mathbf{r}| = \sqrt{r_1^2 + r_2^2}$ je délka vektoru $\mathbf{r}$ a nechť h_r^e je průměr trojúhelníka T_e ve směru vektoru $\mathbf{r}_0^e$, viz Obr. 3.8. Pak

$$\delta_e = \frac{h_r^e}{2|\mathbf{r}|}|\sigma_e|.$$

Parametr σ_e lze zvolit následovně:

$$\sigma_e = \coth\kappa_e - \frac{1}{\kappa_e}, \quad \text{kde} \quad \kappa_e = \frac{r_0^e h_r^e}{2p_0^e}$$

je lokální Pecletovo číslo. Pomocí obdélníkové formule dostaneme

$$Q^{T_e}\left(\delta_e\left(\mathbf{r}\cdot\nabla U\right)\left(\mathbf{r}\cdot\nabla v\right)\right) = [\boldsymbol{\theta}^{T_e}]^T\mathbf{K}^{e4}\boldsymbol{\Delta}^{T_e}, \quad \text{kde} \quad \mathbf{K}^{e4} = \delta_e|T_e|[\mathbf{B}^e]^{T_e}\mathbf{r}_0^e[\mathbf{r}_0^e]^T\mathbf{B}^e.$$

Položíme $\mathbf{K}^e = \mathbf{K}^{e1} + \mathbf{K}^{e2} + \mathbf{K}^{e3} + \mathbf{K}^{e4}$ a sestavíme soustavu rovnic podle výše uvedeného algoritmu.

Rovnost (10.28') dostaneme z rovnosti (10.28) tak, že v ní nahradíme „difúzní člen“ $Q^{T_e}(p\nabla U\cdot\nabla v)$ členem $Q^{T_e}\big([(p\mathbf{I} + \delta_e\mathbf{r}\mathbf{r}^T)\nabla U]\cdot\nabla v\big)$, kde $\mathbf{I}$ je jednotková matice řádu 2. Matice $\delta_e\mathbf{r}\mathbf{r}^T$ reprezentuje umělou směrovou difúzi.

Pro řešení konvekčně-difúzních úloh na bázi metody konečných prvků se používají důmyslnější postupy, zejména metoda SUPG (podle anglického *Streamline Upwind Petrov-Galerkin*), viz [13], nebo metoda DG-FEM (podle anglického *Discontinuous Galerkin Finite Element Method*), viz [15].

10.2. Úloha parabolického typu

Formulace úlohy. Hledáme funkci $u(x,t)$ definovanou pro $x \in \langle 0,\ell\rangle$, $t \in \langle 0,T\rangle$, která vyhovuje diferenciální rovnici

$$c(x)\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(p(x)\frac{\partial u}{\partial x}\right) + q(x)u = f(x,t)\,, \qquad x \in (0,\ell),\ t \in (0,T)\,, \tag{10.53}$$

Dirichletovým okrajovým podmínkám

$$u(0,t) = g_0(t)\,, \qquad u(\ell,t) = g_\ell(t), \qquad t \in (0,T), \tag{10.54}$$

nebo Newtonovým okrajovým podmínkám

$$\begin{aligned} p(0)\frac{\partial u(0,t)}{\partial x} &= \alpha_0 u(0,t) - \beta_0(t), \\ -p(\ell)\frac{\partial u(\ell,t)}{\partial x} &= \alpha_\ell u(\ell,t) - \beta_\ell(t), \end{aligned} \qquad t \in (0,T), \tag{10.55}$$

a počáteční podmínce

$$u(x,0) = \varphi(x), \qquad x \in (0,\ell)\,. \tag{10.56}$$

Možný je také případ, kdy je na jednom okraji předepsána Dirichletova podmínka a na druhém podmínka Newtonova. Úloha (10.53) – (10.56) popisuje *nestacionární vedení tepla v tyči* délky ℓ, T je doba trvání děje. Proměnná x je prostorová, t má význam času, $u(x,t)$ je teplota v bodě x a v čase t, funkce p, q, f, g_0, g_ℓ, β_0, β_ℓ a konstanty α_0, α_ℓ mají stejný význam jako v kapitole 9, c je objemová tepelná kapacita (tj. tepelná kapacita vztažená na jednotku objemu), φ je teplota tyče v čase $t = 0$. Tepelné toky se často uvažují ve tvaru $\beta_0(t) = \alpha_0 u_0^e(t)$, $\beta_\ell(t) = \alpha_\ell u_\ell^e(t)$, kde u_0^e resp. u_ℓ^e je teplota okolí levého resp. pravého konce tyče.

Předpokládejme že funkce c, p, q, f, g_0, g_ℓ, β_0, β_ℓ a φ jsou dostatečně hladké a že jsou splněny podmínky nezápornosti $c \geq c_0 > 0$, $p \geq p_0 > 0$, $q \geq 0$, $\alpha_0 \geq 0$, $\alpha_\ell \geq 0$. Aby existovalo klasické řešení úlohy (10.53) – (10.56), je třeba pro Dirichletovy okrajové podmínky připojit ještě tzv. *podmínky souhlasu*

$$\varphi(0) = g_0(0)\,, \quad \varphi(\ell) = g_\ell(0)\,.$$

Tyto vztahy, vyjadřující soulad počáteční podmínky a Dirichletovy okrajové podmínky, nebývají v aplikacích obvykle splněny. Také funkce c, p, q, f, g_0, g_ℓ, β_0, β_ℓ bývají často jen po částech spojité. V tom případě má úloha (10.53) – (10.56) pouze tzv. slabé řešení (jehož přesnou definici zde uvádět nebudeme). Pro praktické účely je slabé řešení zcela vyhovující a numerické metody, které si uvedeme, budou poskytovat přibližné hodnoty takového slabého řešení.

V úloze vedení tepla funkce c, p, q, α_0, α_ℓ popisují vlastnosti materiálu. Jejich závislost na čase t není obvyklá, ale z hlediska algoritmu (pro určení přibližného řešení), který je možno aplikovat i na jiné úlohy, je budeme považovat za funkce x i t. Algoritmus to nikterak nekomplikuje. Závislost funkcí f, g_0, g_ℓ, β_0 a β_ℓ na čase pravděpodobná je. Naopak zanedbáváme možnou závislost vlastností materiálu c, p, q, α_0, α_ℓ na teplotě. Kdybychom tak neučinili, museli bychom řešit nelineární úlohu, což je mnohem obtížnější.

Úloha (10.53) – (10.56) je okrajovou úlohou druhého řádu vzhledem k proměnné x a počáteční úlohou prvního řádu vzhledem k proměnné t. Při její diskretizaci proto zkombinujeme postupy z prvních dvou kapitol.

Prostorová diskretizace se provádí metodami kapitoly 9. Pro jednoduchost budeme uvažovat úlohu s Dirichletovými okrajovými podmínkami, diferenční metodu a rovnoměrné dělení intervalu $\langle 0,\ell\rangle$ s krokem $h = \ell/N$, tj. $x_i = ih$, $i = 0,1,\ldots,N$. Budeme požadovat, aby rovnice (10.53) byla splněna ve vnitřních uzlech x_i, $i = 1,2,\ldots,N-1$, tj.

$$c(x_i)\frac{\partial u(x_i,t)}{\partial t} - \left[\frac{\partial}{\partial x}\left(p\frac{\partial u}{\partial x}\right)\right](x_i,t) + q(x_i)u(x_i,t) = f(x_i,t)\,.$$

Derivaci podle x vyjádříme pomocí diference analogicky jako v (9.14), tj.

$$-\left[\frac{\partial}{\partial x}\left(p\,\frac{\partial u}{\partial x}\right)\right](x_i,t) = \frac{1}{h^2}\Big[-p_{i-1/2}u(x_{i-1},t) + \\ + \big[p_{i-1/2}+p_{i+1/2}\big]u(x_i,t) - p_{i+1/2}u(x_{i+1},t)\Big] + O(h^2)\,,$$

Zanedbáme-li chybu, dostaneme soustavu rovnic

$$c_i\dot{u}_i(t) + \frac{1}{h^2}\Big[-p_{i-1/2}u_{i-1}(t) + (p_{i-1/2} + p_{i+1/2} + h^2q_i)u_i(t) - \tag{10.57}$$

$$- p_{i+1/2}u_{i+1}(t)\Big] = f_i(t)\,, \qquad i = 1, 2, \ldots, N-1, \quad t \in (0, T)\,,$$

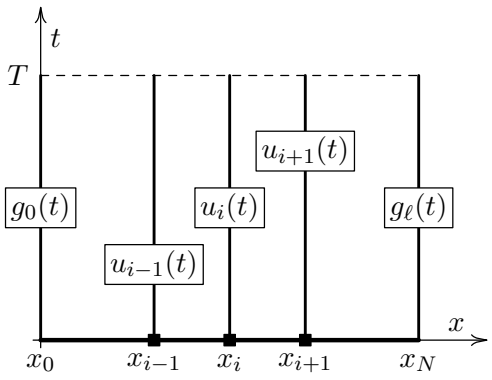
v níž $u_i(t)$ je aproximace $u(x_i, t)$, $\dot{u}_i(t) = \dfrac{\mathrm{d}u_i(t)}{\mathrm{d}t}$ je aproximace $\partial_t u(x_i, t)$ a $f_i(t) = f(x_i, t)$. Z okrajových podmínek (10.54) máme

$$u_0(t) = g_0(t)\,, \qquad u_N(t) = g_\ell(t)\,, \qquad t \in (0, T), \tag{10.58}$$

což dosadíme do (10.57) pro $i = 1$ a $i = N-1$. Z počáteční podmínky obdržíme

$$u_i(0) = \varphi(x_i), \qquad i = 1, 2, \ldots, N-1\,. \tag{10.59}$$

(10.57) je soustava obyčejných diferenciálních rovnic prvního řádu pro $N-1$ hledaných funkcí $u_i(t)$, $i = 1, 2, \ldots, N-1$, s počátečními podmínkami (10.59). Původní úlohu, tj. určení jedné funkce $u(x, t)$, která na obdélníku $\langle 0, \ell\rangle \times \langle 0, T\rangle$vyhovuje (10.53), (10.54) a (10.56), jsme nahradili počáteční úlohou pro soustavu $N-1$ obyčejných diferenciálních rovnic s neznámými funkcemi $u_i(t)$ definovanými na úsečkách $x = x_i$, $i = 1, 2, \ldots, N-1$, $t \in \langle 0, T\rangle$. Tento postup se nazývá *metoda přímek* (anglicky *method of lines*, stručně MOL). Postup, při němž se diskretizace provádí jen vzhledem k některým (nezávisle) proměnným, se nazývá *semidiskretizace*.



Obr. 10.9. Metoda přímek

Počáteční problém (10.57)–(10.59) jsme tedy získali semidiskretizací úlohy (10.53), (10.54) a (10.56) vzhledem k prostorové proměnné x. Úlohu (10.57) – (10.59) můžeme zapsat maticově ve tvaru

$$\mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}(t), \;\; t \in (0, T), \;\; \mathbf{u}(0) = \boldsymbol{\varphi}\,, \tag{10.60}$$

kde $\mathbf{C}$ je diagonální matice s kladnými prvky na diagonále, tzv. *matice tepelné kapacity*,

$\mathbf{K}$ je třídiagonální pozitivně definitní matice známá jako *matice tepelné vodivosti*, $\mathbf{F}(t)$ je tzv. *vektor tepelných zdrojů*, $\mathbf{u}(t) = (u_1(t), u_2(t), \ldots, u_{N-1}(t))^T$ je vektor neznámých funkcí a $\boldsymbol{\varphi} = (\varphi(x_1), \varphi(x_2), \ldots, \varphi(x_{N-1}))^T$ je vektor počátečních teplot.

Newtonova okrajová podmínka. Protože Newtonovy okrajové podmínky se v úloze vedení tepla používají nejčastěji, uvedeme si rovnice reprezentující diskterizaci Newtonovy okrajové podmínky vzhledem k proměnné x v levém resp. pravém koncovém bodu intervalu $\langle 0, \ell\rangle$. Bude-li tedy Newtonova okrajová podmínka předepsána v bodě $x = 0$, zařadíme před rovnice (10.57) jako první rovnici

$$\frac{1}{2}c_0\dot{u}_0(t) + \frac{1}{h^2}\left[(p_{1/2} + h\alpha_0 + \tfrac{1}{2}h^2q_0)u_0(t) - p_{1/2}u_1(t)\right] = \frac{1}{h}\beta_0(t) + \frac{1}{2}f_0(t), \tag{10.57a}$$

a bude-li Newtonova okrajová podmínka předepsána v bodě $x = \ell$, přidáme za poslední z rovnic (10.57) rovnici

$$\frac{1}{2}c_N\dot{u}_N(t)+\frac{1}{h^2}\left[-p_{N-1/2}u_{N-1}(t)+(p_N+h\alpha_\ell+\tfrac{1}{2}h^2q_N)u_N(t)\right]=\frac{1}{h}\beta_\ell(t)+\frac{1}{2}f_N(t). \quad (10.57b)$$

Výsledná matice $\mathbf{K}$ je opět třídiagonální a symetrická. Jestliže $\alpha_0 = \alpha_\ell = 0$ a $q_i = 0$, $i = 0, 1, \ldots, N$, pak je matice $\mathbf{K}$ pozitivně semidefinitní, ve všech ostatních případech je $\mathbf{K}$ pozitivně definitní. Připomeňme: matice $\mathbf{A}$ je pozitivně semidefinitní, jestliže je symetrická a $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ pro každý vektor $\mathbf{x}$.

Časová diskretizace. Prozkoumejme nejdříve charakter počáteční úlohy (10.60) za zjednodušujících předpokladů. Pro $c = p = 1$, $q = f = 0$, $u(0,t) = u(\ell,t) = 0$, lze úlohu (10.60) zapsat ve tvaru

$$\dot{\mathbf{u}} = \mathbf{A}\mathbf{u}\,, \quad t \in (0,T)\,, \qquad \mathbf{u}(0) = \boldsymbol{\varphi}\,, \tag{10.61}$$

kde

$$\mathbf{A} = -\frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \ldots & 0 & 0 \\ -1 & 2 & -1 & \ldots & 0 & 0 \\ 0 & -1 & 2 & \ldots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}. \tag{10.62}$$

Vlastní čísla matice $\mathbf{A}$

$$\lambda_i = -\frac{4N^2}{\ell^2}\sin^2\frac{\pi i}{2N}\,, \qquad i = 1, 2, \ldots, N-1,$$

jsou reálná záporná a $\max_i |\lambda_i| \to \infty$ pro $N \to \infty$. Problém (10.61) je tedy pro velké N tuhý. Totéž platí také pro úlohu (10.60), tj. jde o tuhý problém. Vzhledem k (8.31) je proto vhodné řešit počáteční problém (10.60) metodou, jejíž oblast absolutní stability obsahuje celou zápornou reálnou poloosu komplexní roviny. Metody s touto vlastností se nazývají *A_0-stabilní*. Patří mezi ně všechny metody doporučené v kapitole 8.5 pro řešení tuhých problémů, zejména tedy implicitní Eulerova metoda, lichoběžníková metoda nebo metody zpětného derivování. V Matlabu lze použít některý z programů `ode23s`, `ode23t`, `ode23tb` a `ode15s`. Při řešení úloh vedení tepla se často používá metoda časové diskretizace známá jako

Theta metoda, kterou v následujícím textu stručně popíšeme. Nechť tedy $0 = t_0 < t_1 < \cdots < t_Q = T$ je dělení intervalu $\langle 0, T\rangle$, $\tau_n = t_{n+1} - t_n$ je délka kroku a $\mathbf{u}^n$ je přibližné řešení v čase t_n, tj. $u_i^n \approx u_i(t_n) \approx u(x_i, t_n)$. Nechť θ je pevně zvolené číslo z intervalu $\langle 0, 1\rangle$. Označme $t_{n+\theta} = t_n + \theta\tau_n$. Rovnici (10.60) zapíšeme pro $t = t_{n+\theta}$ a dostaneme

$$\mathbf{C}\dot{\mathbf{u}}^{n+\theta} + \mathbf{K}\mathbf{u}^{n+\theta} = \mathbf{F}^{n+\theta}\,, \tag{10.63}$$

kde $\mathbf{F}^{n+\theta} = \mathbf{F}(t_{n+\theta})$, $\mathbf{u}^{n+\theta} = \mathbf{u}(t_{n+\theta})$, $\dot{\mathbf{u}}^{n+\theta} = \dot{\mathbf{u}}(t_{n+\theta})$. Předpokládejme, že $\mathbf{u}(t)$ je na intervalu $\langle t_n, t_{n+1}\rangle$ lineární funkce, tj.

$$\mathbf{u}(t) = \mathbf{u}^n + \frac{t - t_n}{\tau_n}(\mathbf{u}^{n+1} - \mathbf{u}^n)\,.$$

Pak $\dot{\mathbf{u}}^{n+\theta} = \dfrac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\tau_n}$ a $\mathbf{u}^{n+\theta} = (1-\theta)\mathbf{u}^n + \theta\mathbf{u}^{n+1}$. Po dosazením do (10.63) a malé úpravě dostaneme *θ-metodu*

$$(\mathbf{C}+\tau_n\theta\mathbf{K})\mathbf{u}^{n+1} = (\mathbf{C}-\tau_n(1-\theta)\mathbf{K})\mathbf{u}^n + \tau_n\mathbf{F}^{n+\theta}\,. \tag{10.64}$$

Snadno ověříme, že pro $\frac{1}{2} \le \theta \le 1$ je θ-metoda A-stabilní a tedy také A_0-stabilní, a že pro $0 \le \theta < \frac{1}{2}$ je oblast absolutní stability θ-metody omezená a interval absolutní stability je interval $(-2/(1-2\theta), 0)$. Pokud jde o přesnost, θ-metoda je řádu 1 pro $\theta \ne \frac{1}{2}$ a řádu 2 pro $\theta = \frac{1}{2}$. Všimněte si, že z θ-metody dostaneme pro $\theta = 0$ EE metodu, pro $\theta = 1$ IE metodu a pro $\theta = \frac{1}{2}$ TR metodu. Metoda (10.64) pro $\theta = \frac{1}{2}$ je známa jako *Crankova-Nicolsonova metoda*. Často se zapisuje v analogickém tvaru

$$(\mathbf{C}+\tfrac{1}{2}\tau_n\mathbf{K})\mathbf{u}^{n+1} = (\mathbf{C}-\tfrac{1}{2}\tau_n\mathbf{K})\mathbf{u}^n + \tfrac{1}{2}\tau_n(\mathbf{F}^n+\mathbf{F}^{n+1})\,. \tag{10.65}$$

Přesnost i stabilita metody (10.64) pro $\theta = \frac{1}{2}$ a metody (10.65) je stejná.

Matice $\mathbf{C}+\tau_n\theta\mathbf{K}$ soustavy (10.64) nezávisí na čase a je pozitivně definitní. Soustavu (10.64) je proto účelné řešit pomocí Choleského rozkladu, který stačí provést jen jednou.

Nelineární úloha vedení tepla vznikne tehdy, když některé z funkcí $c, p, q, f, \alpha, \beta$ závisejí na teplotě u. Odpovídající počáteční úlohu

$$\mathbf{C}(\mathbf{u})\dot{\mathbf{u}} = \mathbf{F}(t,\mathbf{u}) - \mathbf{K}(\mathbf{u})\mathbf{u}, \quad t \in (0,T)\,, \qquad \mathbf{u}(0) = \boldsymbol{\varphi}, \tag{10.60'}$$

lze v Matlabu vyřešit programem `ode23t` nebo `ode15s`. Řešení jednodimenzionálního parabolického problému, a to i nelineárního, umožňuje matlabovský program `pdepe`.

Rovnice s konvekčním členem

$$c(x)\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(p(x)\frac{\partial u}{\partial x} - r(x)u\right) + q(x)u = f(x,t), \quad x \in (0,\ell),\ t \in (0,T), \tag{10.53'}$$

popisuje šíření tepla v tekutině, $r = cv$, kde $v(x)$ je rychlost proudění. Diskretizaci konvekčního členu provedeme stejně jako v kapitole 9.2, viz (9.24), (9.28), (9.28'). Pro časovou diskretizaci úlohy s dominantní konvekcí lze doporučit metody IE, TR nebo BDF2.

Rovinná úloha. Hledáme funkci $u(\mathbf{x},t)$, $\mathbf{x} = (x,y)$, která splňuje diferenciální rovnici

$$cu_t - (pu_x - r_1u)_x - (pu_y - r_2u)_y + qu = f, \quad \mathbf{x} \in \Omega,\ t \in (0,T),$$

okrajové podmínky

$$\left.\begin{aligned} u &= g, && \mathbf{x} \in \Gamma_1,\\ -p\frac{\partial u}{\partial n} &= \alpha u - \beta, && \mathbf{x} \in \Gamma_2, \end{aligned}\right\} \quad t \in (0,T)$$

a počáteční podmínku

$$u\big|_{t=0} = \varphi, \quad \mathbf{x} \in \Omega.$$

Prostorovou diskretizaci provedeme metodou konečných prvků, viz kapitola 10.1.4. Slabá diskrétní formulace je tvaru

$$\sum_{T_e\in\mathcal{T}} Q^{T_e}(c\dot{U}v) + a_h(U,v) = L_h(v).$$

Užitím kvadraturní formule (10.31) dostaneme

$$Q^{T_e}(c\dot{U}v) = [\boldsymbol{\theta}^{T_e}]^T \mathbf{C}^{T_e} \boldsymbol{\Delta}^{T_e}, \quad \text{kde} \quad \mathbf{C}^{T_e} = \tfrac{1}{3}|T_e| \begin{pmatrix} c_1^e & 0 & 0 \\ 0 & c_2^e & 0 \\ 0 & 0 & c_3^e \end{pmatrix}. \tag{10.66}$$

Globální matici $\mathbf{C} = \{c_{ij}\}_{i,j=1}^N$ sestavíme z elementárních matic $\mathbf{C}^{T_e}$. Protože matice $\mathbf{C}^{T_e}$ jsou diagonální, je také $\mathbf{C}$ diagonální a

$$\sum_{T_e\in\mathcal{T}} Q^{T_e}(c\dot{U}v) = \sum_{T_e\in\mathcal{T}} [\boldsymbol{\theta}^{T_e}]^T \mathbf{C}^{T_e} \boldsymbol{\Delta}^{T_e}) = \boldsymbol{\theta}^T \mathbf{C} \dot{\boldsymbol{\Delta}}.$$

Sestavování matice $\mathbf{C}$ začneme tím, že $\mathbf{C}$ vynulujeme. Postupně pro každý trojúhelník $T_e \in \mathcal{T}$ sestavíme elementární matici $\mathbf{C}^{T_e} = \{c_{rs}^{T_e}\}_{r,s=1}^3$, viz (10.66), pro $r = 1, 2, 3$ určíme $i = \texttt{ELEM}(e, r)$ a když $i \le N$, provedeme $c_{ii} \leftarrow c_{ii} + c_{rr}^{T_e}$. Není těžké ukázat, že

$$c_{ii} = c(x_i, y_i)|B_i|, \quad i = 1, 2, \ldots, N,$$

kde $|B_i|$ je plocha konečného objemu B_i, viz obrázky 10.5 a 10.6. Časovou diskretizaci počáteční úlohy

$$\mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}(t), \quad t \in (0,T), \quad \mathbf{u}(0) = \boldsymbol{\varphi}\,, \tag{10.67}$$

řešíme θ-metodou. Přitom $\mathbf{u}(t) = (u_1(t), u_2(t), \ldots, u_N(t))^T$, kde $u_i(t) \approx u(x_i, y_i, t)$, $\mathbf{K}$ a $\mathbf{F}(t)$ viz kapitola 10.1.4 (složky $F_i(t)$ vektoru $\mathbf{F}(t)$ lze vyjádřit jako $F_i(t) = f(x_i, y_i, t)|B_i|$, $i = 1, 2, \ldots, N$), $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_N)^T$.

10.3. Úloha hyperbolického typu

Formulace úlohy. Hledáme funkci $u(x,t)$ definovanou pro $x \in \langle 0, \ell\rangle$, $t \in \langle 0, T\rangle$, která vyhovuje diferenciální rovnici

$$\rho(x)\frac{\partial^2 u}{\partial t^2} + c(x)\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(p(x)\frac{\partial u}{\partial x}\right) + q(x)u = f(x,t), \quad x \in (0,\ell),\ t \in (0,T), \tag{10.68}$$

Dirichletovým okrajovým podmínkám

$$u(0,t) = g_0(t), \qquad u(\ell,t) = g_\ell(t), \qquad t \in (0,T), \tag{10.69}$$

nebo Newtonovým okrajovým podmínkám

$$\begin{aligned} p(0)\frac{\partial u(0,t)}{\partial x} &= \alpha_0 u(0,t) - \beta_0(t), \\ -p(\ell)\frac{\partial u(\ell,t)}{\partial x} &= \alpha_\ell u(\ell,t) - \beta_\ell(t), \end{aligned} \qquad t \in (0,T), \tag{10.70}$$

a počátečním podmínkám

$$u(x,0)=\varphi(x),\quad \frac{\partial u(x,0)}{\partial t}=\psi(x),\qquad x\in(0,\ell). \tag{10.71}$$

Možný je také případ, kdy na jednom okraji je předepsána Dirichletova podmínka a na druhém Newtonova. Tato úloha vyjadřuje *příčné kmitání tenké struny* (nebo *podélné kmitání prutu*) délky ℓ. Proměnná x je prostorová, t má význam času, $u(x,t)$ je deformace (tj. příčná výchylka pro strunu nebo podélné posunutí v případě prutu) v bodě x a v čase t, ρ je hustota, c udává tlumení (pro $c>0$ jsou kmity tlumené, pro $c=0$ netlumené), p charakterizuje tuhost, q odpor okolního prostředí a f vnější síly. Dirichletovy okrajové podmínky předepisují deformaci, Newtonovy okrajové podmínky vnitřní síly: α_0 resp. α_ℓ reprezentuje tuhost pružné podpory a β_0 resp. β_ℓ bodovou vnější sílu. Počáteční podmínky určují počáteční deformaci φ a její počáteční rychlost ψ.

Předpokládejme, že funkce ρ, c, p, q, f, g_0, g_ℓ, β_0, β_ℓ, φ a ψ jsou dostatečně hladké, že jsou splněny podmínky nezápornosti $\rho\ge\rho_0>0$, $c\ge 0$, $p\ge p_0>0$, $q\ge 0$, $\alpha_0\ge 0$, $\alpha_\ell\ge 0$ a že platí podmínky souhlasu

$$\varphi(0)=g_0(0)\,,\quad \psi(0)=g_0'(0)\,,\qquad\qquad \varphi(\ell)=g_\ell(0)\,,\quad \psi(\ell)=g_\ell'(0)\,,$$

vyjadřující soulad počátečních podmínek a Dirichletových okrajových podmínek. Pak má úloha (10.68) – (10.71) jediné klasické řešení.

Podmínky souhlasu v aplikacích někdy nebývají splněny. Také funkce ρ, c, p, q, f, g_0, g_ℓ, β_0, β_ℓ bývají často jen po částech spojité. V tom případě má úloha (10.53) – (10.56) pouze tzv. slabé řešení (jehož přesnou definici zde ovšem uvádět nebudeme). Pro praktické účely je slabé řešení zcela vyhovující a numerické metody, které si uvedeme, budou poskytovat přibližné hodnoty takového slabého řešení.

Úloha (10.68) – (10.71) je okrajovou úlohou druhého řádu vzhledem k proměnné x a počáteční úlohou druhého řádu vzhledem k proměnné t. Při její diskretizaci proto opět použijeme postupy z prvních dvou kapitol.

Prostorová diskretizace se provede stejně jako u parabolického problému metodou přímek, viz kapitola 10.2. Opět předpokládáme rovnoměrné dělení a Dirichletovu okrajovou podmínku. Pro $u_i(t)$ aproximující $u(x_i,t)$ při označení $\dot u_i(t)=\dfrac{\mathrm{d}u_i(t)}{\mathrm{d}t}$, $\ddot u_i(t)=\dfrac{\mathrm{d}^2u_i(t)}{\mathrm{d}t^2}$ dostaneme soustavu rovnic

$$\rho_i\ddot u_i(t)+c_i\dot u_i(t)+\frac{1}{h^2}\Big(-p_{i-1/2}u_{i-1}(t)+[p_{i-1/2}+p_{i+1/2}+h^2q_i]u_i(t)- \tag{10.72}$$

$$-p_{i+1/2}u_{i+1}(t)\Big)=f_i(t)\,,\qquad i=1,2,\dots,N-1,\quad t\in(0,T)\,.$$

Z okrajových podmínek (10.69) máme

$$u_0(t)=g_0(t),\qquad u_N(t)=g_\ell(t),\qquad t\in(0,T)\,, \tag{10.73}$$

což dosadíme do (10.72) pro $i=1$ a $i=N-1$. Z počátečních podmínek (10.71) dostaneme

$$u_i(0)=\varphi(x_i),\qquad \dot u_i(0)=\psi(x_i),\qquad i=1,2,\dots,N-1\,. \tag{10.74}$$

(10.72) je soustava obyčejných diferenciálních rovnic druhého řádu pro $N-1$ hledaných funkcí $u_i(t)$, $i = 1, 2, \ldots, N-1$, s počátečními podmínkami (10.74).

Počáteční úlohu (10.72)–(10.74) můžeme zapsat maticově ve tvaru

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}(t)\,, \quad t \in (0,T), \qquad \mathbf{u}(0) = \boldsymbol{\varphi}, \quad \dot{\mathbf{u}}(0) = \boldsymbol{\psi}\,, \tag{10.75}$$

kde $\mathbf{M}$ je diagonální matice s kladnými prvky na diagonále, tzv. *matice hmotnosti*, $\mathbf{C}$ je diagonální matice s nezápornými prvky na diagonále, tzv. *matice útlumu*, $\mathbf{K}$ je třídiagonální pozitivně definitní matice, tzv. *matice tuhosti*, $\mathbf{F}(t)$ je tzv. *vektor (vnějšího) zatížení*, $\boldsymbol{\varphi} = (\varphi(x_1), \varphi(x_2), \ldots, \varphi(x_{N-1}))^T$, $\boldsymbol{\psi} = (\psi(x_1), \psi(x_2), \ldots, \psi(x_{N-1}))^T$ jsou vektory počátečních hodnot a $\mathbf{u}(t) = (u_1(t), u_2(t), \ldots, u_{N-1}(t))^T$ je vektor neznámých funkcí.

Časová diskretizace. Abychom mohli posoudit tuhost problému (10.75), zapíšeme ho jako počáteční problém prvního řádu,

$$\mathbf{R}\dot{\mathbf{w}} + \mathbf{S}\mathbf{w} = \mathbf{G}(t)\,, \quad t \in (0,T)\,, \qquad \mathbf{w}(0) = \boldsymbol{\kappa}\,, \tag{10.76}$$

kde

$$\mathbf{R} = \begin{pmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{M} \end{pmatrix}, \ \mathbf{w} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}, \ \mathbf{S} = \begin{pmatrix} \mathbf{O} & -\mathbf{I} \\ \mathbf{K} & \mathbf{C} \end{pmatrix}, \ \mathbf{G}(t) = \begin{pmatrix} \mathbf{o} \\ \mathbf{F}(t) \end{pmatrix}, \ \boldsymbol{\kappa} = \begin{pmatrix} \boldsymbol{\varphi} \\ \boldsymbol{\psi} \end{pmatrix}, \tag{10.77}$$

přičemž $\mathbf{v} = \dot{\mathbf{u}}$, $\mathbf{I}$ je jednotková matice, $\mathbf{O}$ je nulová matice a $\mathbf{o}$ je nulový vektor.

Prozkoumejme charakter počáteční úlohy (10.76) za zjednodušujících předpokladů. Pro
$\rho = p = 1$, $c = konst$, $q = f = 0$, $u(0,t) = u(\ell,t) = 0$, je $\mathbf{M} = \mathbf{I}$, $\mathbf{K} = -\mathbf{A}$ a $\mathbf{C} = c\mathbf{I}$, kde $\mathbf{A}$ je definována předpisem (10.62). Rovnice (10.76) se tak zjednoduší, dostaneme

$$\dot{\mathbf{w}} = \mathbf{P}\mathbf{w}, \quad \mathbf{w}(0) = \boldsymbol{\kappa}, \qquad \text{kde} \quad \mathbf{P} = \begin{pmatrix} \mathbf{O} & \mathbf{I} \\ \mathbf{A} & -c\mathbf{I} \end{pmatrix}. \tag{10.78}$$

Dá se ukázat, že vlastní čísla matice $\mathbf{P}$

$$\lambda_i = -\tfrac{1}{2}c \pm \sqrt{\tfrac{1}{4}c^2 - \omega_i^2}, \quad \text{kde} \quad \omega_i = \frac{2N}{\ell}\sin\frac{\pi i}{2N}\,, \quad i = 1, 2, \ldots, N-1.$$

Zřejmě $\max_i |\lambda_i| \to \infty$ pro $N \to \infty$. Pro $c = 0$ jsou vlastní čísla ryze imaginární a pro $c > 0$ mají zápornou reálnou složku. Problém (10.78) je pro velké N tuhý. Protože reálné složky vlastních čísel jsou zdola ohraničené, $\mathrm{Re}(\lambda_i) \geq -c$, a velikost imaginárních složek je neomezená, říkáme, že problém (10.78) je *oscilatoricky tuhý*. Problém (10.76) se chová obdobně. Vzhledem k (8.31) je proto vhodné řešit úlohu (10.78) pro $c > 0$ metodou, jejíž oblast absolutní stability obsahuje pás $R_c = \{z \in \mathbb{C} \,|\, -c < \mathrm{Re}(z) < 0\}$.

Pro $c = 0$ lze odvodit rovnost

$$[\mathbf{u}(t)]^T\mathbf{u}(t) + [\mathbf{v}(t)]^T\mathbf{K}^{-1}\mathbf{v}(t) = \boldsymbol{\varphi}^T\boldsymbol{\varphi} + \boldsymbol{\psi}^T\mathbf{K}^{-1}\boldsymbol{\psi}, \qquad t > 0,$$

vyjadřující stabilitu počátečního problému (10.78) vzhledem k počáteční podmínce. Pro přibližné řešení $\mathbf{w}_n \approx \mathbf{w}(n\tau)$ spočtené lichoběžníkovou metodou lze odvodit analogickou rovnost

$$\mathbf{u}_n^T\mathbf{u}_n + \mathbf{v}_n^T\mathbf{K}^{-1}\mathbf{v}_n = \boldsymbol{\varphi}^T\boldsymbol{\varphi} + \boldsymbol{\psi}^T\mathbf{K}^{-1}\boldsymbol{\psi}, \qquad n = 1, 2, \ldots$$

To je skvělé, lichoběžníková metoda aplikovaná na řešení úlohy (10.78) pro $c = 0$ vykazuje stejnou stabilitu jako přesné řešení.

Protože lichoběžníková metoda je A-stabilní, je vhodnou metodou pro každé $c \geq 0$. Z matlabovských programů lze doporučit programy `ode23t` a `ode23tb`, které jsou na lichoběžníkové metodě založeny.

Lichoběžníková metoda aplikovaná na úlohu (10.76) vede na předpis

$$(\mathbf{R} + \tfrac{1}{2}\tau_n \mathbf{S})\mathbf{w}^{n+1} = (\mathbf{R} - \tfrac{1}{2}\tau_n \mathbf{S})\mathbf{w}^n + \tfrac{1}{2}\tau_n(\mathbf{G}^n + \mathbf{G}^{n+1}), \tag{10.79}$$

viz (10.65). Pro efektivní výpočet složek $\mathbf{u}^{n+1}$ a $\mathbf{v}^{n+1}$ vektoru $\mathbf{w}^{n+1}$ je vhodné soustavu rovnic (10.79) zapsat po složkách, tj.

$$\mathbf{u}^{n+1} - \tfrac{1}{2}\tau_n \mathbf{v}^{n+1} = \mathbf{u}^n + \tfrac{1}{2}\tau_n \mathbf{v}^n,$$
$$(\mathbf{M} + \tfrac{1}{2}\tau_n \mathbf{C})\mathbf{v}^{n+1} + \tfrac{1}{2}\tau_n \mathbf{K}\mathbf{u}^{n+1} = (\mathbf{M} - \tfrac{1}{2}\tau_n \mathbf{C})\mathbf{v}^n - \tfrac{1}{2}\tau_n \mathbf{K}\mathbf{u}^n + \tfrac{1}{2}\tau_n(\mathbf{F}^{n+1} + \mathbf{F}^n).$$

Z prvé rovnice vyjádříme $\mathbf{v}^{n+1}$, viz (10.81), dosadíme do druhé rovnice a obdržíme rovnici pro výpočet $\mathbf{u}^{n+1}$:

$$\hat{\mathbf{K}}\mathbf{u}^{n+1} = \hat{\mathbf{F}}, \qquad \text{kde} \tag{10.80}$$
$$\hat{\mathbf{K}} = \frac{4}{\tau_n^2}\mathbf{M} + \frac{2}{\tau_n}\mathbf{C} + \mathbf{K}\,, \qquad \hat{\mathbf{F}} = \left(\frac{4}{\tau_n^2}\mathbf{M} + \frac{2}{\tau_n}\mathbf{C} - \mathbf{K}\right)\mathbf{u}^n + \frac{4}{\tau_n}\mathbf{M}\mathbf{v}^n + \mathbf{F}^n + \mathbf{F}^{n+1}.$$

Až vypočítáme $\mathbf{u}^{n+1}$, dopočítáme

$$\mathbf{v}^{n+1} = \frac{2}{\tau_n}(\mathbf{u}^{n+1} - \mathbf{u}^n) - \mathbf{v}^n. \tag{10.81}$$

Startujeme přitom z počátečních hodnot

$$\mathbf{u}^0 = \boldsymbol{\varphi}, \quad \mathbf{v}^0 = \boldsymbol{\psi}. \tag{10.82}$$

Matice $\hat{\mathbf{K}}$ soustavy (10.80) nezávisí na čase a je pozitivně definitní. Soustavu (10.80) je proto účelné řešit pomocí Choleského rozkladu, který stačí provést jen jednou.

Metoda (10.80)–(10.82) je speciálním případem *Newmarkovy metody* hojně používané v inženýrské praxi, viz [5].

Rovinná úloha. Hledáme funkci $u(\mathbf{x}, t)$, $\mathbf{x} = (x, y)$, která splňuje diferenciální rovnici

$$\varrho u_{tt} + c u_t - (p u_x)_x - (p u_y)_y + q u = f, \quad \mathbf{x} \in \Omega,\ t \in (0, T),$$

okrajové podmínky

$$\left.\begin{aligned} u &= g, && \mathbf{x} \in \Gamma_1, \\ -p\frac{\partial u}{\partial n} &= \alpha u - \beta, && \mathbf{x} \in \Gamma_2, \end{aligned}\right\} \quad t \in (0, T)$$

a počáteční podmínky

$$u\big|_{t=0} = \varphi, \qquad u_t\big|_{t=0} = \psi, \quad \mathbf{x} \in \Omega.$$

Prostorovou diskretizaci provedeme metodou konečných prvků, viz kapitola 10.1.4. Slabá diskrétní formulace je tvaru

$$\sum_{T_e\in\mathcal{T}} Q^{T_e}(\varrho\ddot{U}v) + \sum_{T_e\in\mathcal{T}} Q^{T_e}(c\dot{U}v) + a_h(U,v) = L_h(v).$$

První dva členy zpracujeme pomocí kvadraturní formule (10.31). Standardním postupem metody konečných prvků pak dostaneme počáteční úlohu

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}(t), \quad t\in(0,T), \qquad \mathbf{u}(0) = \boldsymbol{\varphi},\ \dot{\mathbf{u}}(0) = \boldsymbol{\psi}. \tag{10.83}$$

Zde

$\mathbf{u}(t) = (u_1(t), u_2(t), \dots, u_N(t))^T$, kde $u_i(t) \approx u(x_i, y_i, t)$,

$\mathbf{M} = \mathrm{diag}(\varrho(x_1,y_1)|B_1|, \varrho(x_2,y_2)|B_2|, \dots, \varrho(x_N,y_N)|B_N|)$,

$\mathbf{C} = \mathrm{diag}(c(x_1,y_1)|B_1|, c(x_2,y_2)|B_2|, \dots, c(x_N,y_N)|B_N|)$,

$\mathbf{K}$ viz kapitola 10.1.4, kde $\mathbf{K}^{T_e} = \mathbf{K}^{T_e,1} + \mathbf{K}^{T_e,3}$ místo (10.46),

$\mathbf{F}(t) = \mathrm{diag}(f(x_1,y_1,t)|B_1|, f(x_2,y_2,t)|B_2|, \dots, f(x_N,y_N,t)|B_N|)$,

$\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \dots, \varphi_N)^T$, $\quad \boldsymbol{\psi} = (\psi_1, \psi_2, \dots, \psi_N)^T$.

Časovou diskretizaci provedeme lichoběžníkovou metodou podle vzorců (10.80)–(10.82).

10.4. Hyperbolická rovnice prvního řádu

Formulace úlohy. Hledáme funkci $u(x,t)$ definovanou pro $x\in\langle 0,\ell\rangle$, $t\in\langle 0,T\rangle$, která vyhovuje diferenciální rovnici

$$\frac{\partial u}{\partial t} + a(x,t)\frac{\partial u}{\partial x} = f(x,t)\,, \quad x\in(0,\ell)\,,\ t\in(0,T)\,, \tag{10.84}$$

okrajovým podmínkám

$$\begin{aligned} u(0,t) &= g_0(t)\,, \\ u(\ell,t) &= g_\ell(t)\,, \end{aligned} \quad \text{pokud} \quad \begin{aligned} a(0,t) &> 0\,, \\ a(\ell,t) &< 0\,, \end{aligned} \qquad t\in(0,T)\,, \tag{10.85}$$

a počáteční podmínce

$$u(x,0) = \varphi(x)\,. \tag{10.86}$$

Nechť $Q = \langle 0,\ell\rangle \times \langle 0,T\rangle$ je obdélník, v němž hledáme řešení, a ∂Q^+ ta je část hranice ∂Q obdélníka Q, na níž je předepsána počáteční nebo okrajová podmínka, tj.

$$\partial Q^+ = \{[x,t]\in\partial Q \,|\, t=0 \text{ nebo } x=0 \text{ (pokud } a(0,t)>0) \text{ nebo } x=\ell \text{ (pokud } a(\ell,t)<0)\}\,.$$

Pro každý bod $P_0 = [x_0,t_0]\in Q\setminus\partial Q^+$ určeme řešení $x(t)$ počátečního problému

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = a(x(t),t)\,, \qquad t<t_0\,, \quad x(t_0) = x_0. \tag{10.87}$$

Křivka $x(t)$ se nazývá *charakteristika* příslušná bodu $[x_0, t_0]$. Předpokládejme, že charakteristika protíná ∂Q^+ v jediném bodě $P_0^* = [x_0^*, t_0^*]$. Tomuto bodu říkáme *pata charakteristiky*. Podle pravidla o derivování složené funkce

$$\frac{\mathrm{d}u(x(t),t)}{\mathrm{d}t} = \frac{\partial u(x(t),t)}{\partial x}\frac{\mathrm{d}x(t)}{\mathrm{d}t} + \frac{\partial u(x(t),t)}{\partial t} = \frac{\partial u(x(t),t)}{\partial t} + a(x(t),t)\frac{\partial u(x(t),t)}{\partial x}\,.$$

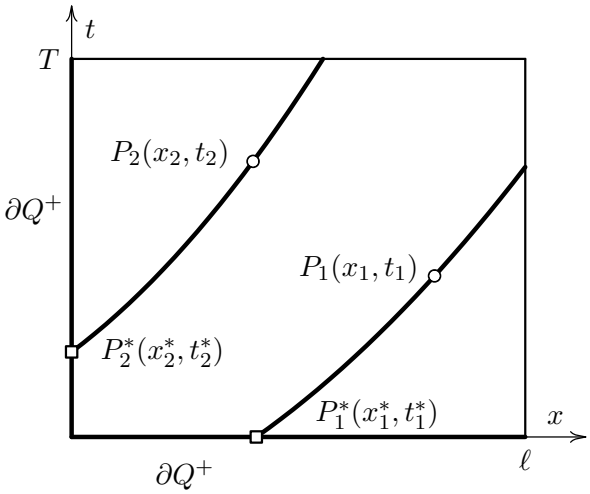
Úlohu (10.84)–(10.86) proto můžeme na charakteristice $x(t)$ zapsat ve tvaru

$$\frac{\mathrm{d}u(x(t),t)}{\mathrm{d}t} = f(x(t),t),\ \ t \in (t_0^*, t_0)\,, \quad u(x(t_0^*), t_0^*) = \begin{cases} \varphi(x_0^*) & \text{pro } \ t_0^* = 0\,, \\ g_0(t_0^*) & \text{pro } \ x_0^* = 0\,, \\ g_\ell(t_0^*) & \text{pro } \ x_0^* = \ell\,. \end{cases} \tag{10.88}$$

Předpokládejme, že funkce a, f, g_0, g_ℓ a φ jsou spojité, že funkce $a(0,t)$ i $a(\ell,t)$ nemění znaménko a že okrajová podmínka je kompatibilní s počáteční podmínkou, tj. platí

$$\varphi(0) = g_0(0)\ (\text{pokud } a(0,0) > 0)\,, \quad \varphi(\ell) = g_\ell(0)\ (\text{pokud } a(\ell,0) < 0)\,.$$

Pak úloha (10.87)–(10.88), a tedy rovněž úloha (10.84)–(10.86), má jediné klasické řešení.



Obr. 10.10. Charakteristiky

Rovnice (10.84) bývá označována jako *advekční* rovnice nebo také *transportní* rovnice. Úloha (10.84)–(10.86) popisuje třeba šíření příměsi prouděním, tj. bez vlivu difúze: $u(x,t)$ je koncentrace příměsi v tekutině v bodu x a v čase t, $a = \varrho v$ je součin hustoty ϱ tekutiny a její rychlosti v, f je zdrojový člen. Charakteristika $x(t)$ je trajektorie, po které se částečka příměsi pohybuje. Pokud bychom připustili také difúzní šíření příměsi, dostali bychom *konvekčně-difúzní rovnici*

$$c(x)\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(r(x,t)u) - \frac{\partial}{\partial x}\left(p(x)\frac{\partial u}{\partial x}\right) = f(x,t), \quad x \in (0,\ell)\,,\ t \in (0,T),$$

v níž p je koeficient difúze, viz (10.53'). Na rovnici advekce lze proto nahlížet jako na limitní případ konvekčně-difúzní rovnice, v níž zanedbáme účinky difúze. Jiná forma advekční rovnice tedy je

$$c(x)\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(r(x,t)u) = f(x,t)\,, \quad x \in (0,\ell)\,,\ t \in (0,T). \tag{10.84'}$$

Jestliže rovnice (10.84') popisuje přenos tepla advekcí, pak u je teplota, c je objemová tepelná kapacita (tj. tepelná kapacita vztažená na jednotku objemu), $r = cv$, kde v je rychlost proudění a f je tepelný zdroj.

Metoda přímek. Uvažujme rovnoměrné dělení intervalu $\langle 0, \ell\rangle$, tj. $x_i = ih$, $i = 0, 1, \ldots, N$, kde $h = \ell/N$. Předpokládejme, že funkce $a(x,t)$ v krajních bodech intervalu $\langle 0, \ell\rangle$ nemění znaménko. Splnění rovnice (10.84) budeme požadovat ve vnitřních uzlech $x_1, x_2, \ldots, x_{N-1}$ a pro $a(0,t) < 0$ také v uzlu x_0 a pro $a(\ell,t) > 0$ také v uzlu x_N. V rovnici

$$\frac{\partial u(x_i,t)}{\partial t} + a(x_i,t)\frac{\partial u(x_i,t)}{\partial x} = f(x_i,t)$$

aproximujeme člen $u_x(x_i,t)$ ve vnitřních uzlech centrální diferencí a v krajních uzlech jednostrannou diferencí. Pokud třeba $a(0,t) > 0$, $a(\ell,t) > 0$ pro všechna $t \in \langle 0, T\rangle$, dostaneme

$$\begin{aligned}
&\dot u_1(t) + a_1(t)[u_2(t) - g_0(t)]/(2h) = f_1(t)\,,\\
&\dot u_i(t) + a_i(t)[u_{i+1}(t) - u_{i-1}(t)]/(2h) = f_i(t)\,, \quad i = 2, 3, \ldots, N-1\,,\\
&\dot u_N(t) + a_N(t)[3u_N(t) - 4u_{N-1}(t) + u_{N-2}(t)]/(2h) = f_N(t)\,,
\end{aligned} \tag{10.89}$$

kde $a_i(t) = a(x_i,t)$, $f_i(t) = f(x_i,t)$ a $u_i(t)$ je aproximace $u(x_i,t)$. Z (10.86) dostaneme počáteční podmínky

$$u_i(0) = \varphi_i\,, \quad i = 1, 2, \ldots, N\,, \tag{10.90}$$

kde $\varphi_i = \varphi(x_i)$. Počáteční problém (10.89)–(10.90) řešíme vhodnou numerickou metodou. K jejímu výběru nám poslouží zjednodušený model. Předpokládejme, že $a = 1$, $f = 0$ a uvažme periodické okrajové podmínky $u(0,t) = u(\ell,t)$. Pomocí centrálních diferencí odvodíme

$$\begin{aligned}
&\dot u_1(t) + [u_2(t) - u_N(t)]/(2h) = 0\,,\\
&\dot u_i(t) + [u_{i+1}(t) - u_{i-1}(t)]/(2h) = 0\,, \quad i = 2, 3, \ldots, N-1\,,\\
&\dot u_N(t) + [u_1(t) - u_{N-1}(t)]/(2h) = 0\,.
\end{aligned} \tag{10.89'}$$

Počáteční problém (10.89')–(10.90) zapíšeme maticově, tj.

$$\dot{\mathbf{u}} = \mathbf{A}\mathbf{u}\,, \quad t \in (0,T)\,, \qquad \mathbf{u}(0) = \boldsymbol{\varphi}\,, \tag{10.91}$$

kde $\mathbf{u}(t) = (u_1(t), u_2(t), \ldots, u_N(t))^T$, $\boldsymbol{\varphi}(t) = (\varphi_1(t), \varphi_2(t), \ldots, \varphi_N(t))^T$ a

$$\mathbf{A} = -\frac{1}{2h}\begin{pmatrix}
0 & 1 & 0 & \ldots & 0 & -1\\
-1 & 0 & 1 & \ldots & 0 & 0\\
0 & -1 & 0 & \ldots & 0 & 0\\
\vdots & & \ddots & \ddots & \ddots & \vdots\\
0 & 0 & 0 & -1 & 0 & 1\\
1 & 0 & 0 & & -1 & 0
\end{pmatrix}. \tag{10.92}$$

Dá se ukázat, že vlastní čísla matice $\mathbf{A}$

$$\lambda_i(\mathbf{A}) = I\frac{N}{\ell}\sin\frac{2\pi i}{N}, \quad I = \sqrt{-1}, \qquad i = 1, 2, \ldots, N,$$

leží na imaginární ose, $\max_i |\lambda_i| \to \infty$ pro $N \to \infty$, tj. pro velké N je počáteční problém (10.91) oscilatoricky tuhý.

Antisymetrická matice $\mathbf{A}$ je diagonizovatelná pomocí unitární matice[5] vlastních vektorů, tj. platí $\mathbf{V}^H\mathbf{A}\mathbf{V} = \mathbf{D}$, kde $\mathbf{D} = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$, viz [55]. Řešení počáteční úlohy (10.91) lze zapsat ve tvaru $\mathbf{u}(t) = \mathbf{V}\mathbf{S}(t)\mathbf{V}^H\boldsymbol{\varphi}$, kde $\mathbf{S}(t) = \mathrm{diag}\{e^{\lambda_1 t}, e^{\lambda_2 t}, \ldots, e^{\lambda_N t}\}$. Ověřte! Odtud již snadno obdržíme $\|\mathbf{u}(t)\|_2 = \|\boldsymbol{\varphi}\|_2$ pro každé $t > 0$. Ověřte! K numerickému řešení se proto skvěle hodí lichoběžníková metoda, pro kterou $\|\mathbf{u}_n\|_2 = \|\boldsymbol{\varphi}\|_2$ pro každé $n = 1, 2, \ldots$ Ověřte!

Problém (10.89)–(10.90) se chová podobně: vlastní čísla odpovídající matice $\mathbf{A}$ leží v záporné komplexní polorovině v blízkosti imaginární osy a $|\lambda_{max} - IN/\ell| \to 0$ pro $N \to \infty$. Pro řešení počátečního problému (10.89)–(10.90) lze kromě lichoběžníkové metody doporučit také metody, jejichž oblast absolutní stability obsahuje obdélník

$$R_{\alpha\beta} = \{z \in \mathbb{C} \,|\, -\alpha \le \mathrm{Re}(z) \le 0, -\beta \le \mathrm{Im}(z) \le \beta\}.$$

Pro metodu BS32 je $(\alpha;\beta) \doteq (1{,}64\,;\,1{,}73)$ a pro metodu DP54 je $(\alpha;\beta) \doteq (3{,}19\,;\,0{,}99)$, viz např. [26]. Délka kroku těchto dvou metod je sice z důvodu stability omezena, toto omezení však není nijak dramatické, délka kroku bude řádu $O(h)$. Z matlabovských programů lze tedy doporučit programy `ode23t`, `ode23` a `ode45`.

Metoda charakteristik je další vhodnou technikou pro řešení úlohy (10.84)–(10.86). Její podstatu vysvětlíme nejdříve pro zjednodušenou úlohu

$$\begin{aligned} &\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0\,, \quad x \in (0,\ell)\,,\ t \in (0,T)\,, \\ &u(0,t) = g_0(t), \quad t \in (0,T)\,, \\ &u(x,0) = \varphi(x), \quad x \in (0,\ell)\,, \end{aligned} \tag{10.93}$$

kde $a > 0$ je konstanta. Diferenciální rovnici přepíšeme pomocí charakteristik na tvar

$$\frac{\mathrm{d}u(x(t),t)}{\mathrm{d}t} = 0. \tag{10.94}$$

Zvolme rovnoměrné dělení intervalu $\langle 0,\ell\rangle$ s krokem $h = \ell/N$, tj. $x_i = ih$, $i = 0, 1, \ldots, N$, a rovnoměrné dělení intervalu $\langle 0,T\rangle$ s krokem $\tau = T/Q$, tj. $t_n = n\tau$, $n = 0, 1, \ldots, Q$. Charakteristika vycházející z bodu $[x_i, t_{n+1}]$ je přímka $x_i(t) = x_i + a(t - t_{n+1})$. V čase $t = t_n$ je

$$x_i^n := x_i(t_n) = x_i - a\tau\,.$$

Předpokládejme, že $a\tau \le h$. Pak pro $i = 1, 2, \ldots, N$ bod $x_i^n \in \langle x_{i-1}, x_i\rangle$, zejména tedy $x_i^n \in \langle 0,\ell\rangle$, takže $u(x_i^n, t_n)$ má smysl. Integrací rovnice (10.94) od t_n do t_{n+1} obdržíme

$$\int_{t_n}^{t_{n+1}} \frac{\mathrm{d}u(x_i(t),t)}{\mathrm{d}t}\,\mathrm{d}t = u(x_i, t_{n+1}) - u(x_i^n, t_n) = 0\,,$$

[5] Čtvercová *matice* $\mathbf{V}$ *je unitární*, jestliže $\mathbf{V}^H\mathbf{V} = \mathbf{V}\mathbf{V}^H = \mathbf{I}$. Přitom $\mathbf{V}^H$ je *matice Hermitovsky sdružená*, tj. transponovaná a komplexně sdružená: když $\mathbf{V} = \{v_{ij}\}_{i,j=1}^N$ a $\mathbf{V}^H = \{v_{ij}^H\}_{i,j=1}^N$, pak $v_{ij}^H = \bar{v}_{ji}$ je číslo komplexně sdružené s číslem v_{ji}.

a odtud

$$u(x_i, t_{n+1}) = u(x_i^n, t_n), \quad i = 1, 2, \ldots, N. \tag{10.95}$$

Numerickou metodu dostaneme tak, že $u(x_i^n, t_n)$ určíme přibližně interpolací, tj. počítáme

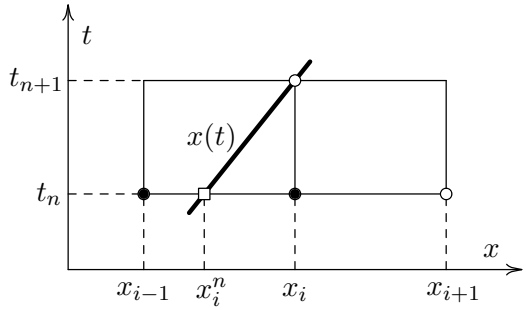
$$u_i^{n+1} = P_k(x_i^n)\,, \tag{10.96}$$

kde $P_k(x)$ je interpolační polynom stupně $k \geq 1$ splňující interpolační podmínky

$$P_k(x_{i-1}) = u_{i-1}^n\,, \quad P_k(x_i) = u_i^n, \tag{10.97}$$

které v případě $k > 1$ doplníme o dalších $k-1$ vhodně zvolených interpolačních podmínek. Pro lineární polynom P_1 z podmínek (10.97) odvodíme

$$P_1(x) = u_i^n + \frac{u_i^n - u_{i-1}^n}{h}(x - x_i) \quad \text{a odtud} \quad P_1(x_i^n) = u_i^n - \frac{a\tau}{h}(u_i^n - u_{i-1}^n)\,.$$



Obr. 10.11. Upwind metoda

Dostali jsme tak *upwind* metodu

$$u_i^{n+1} = u_i^n - a\tau\frac{u_i^n - u_{i-1}^n}{h}. \tag{10.98}$$

Název metody nám připomíná, že veličina u v uzlu x_i závisí jen hodnotách této veličiny v uzlech ležících „proti proudu, proti větru“, pro $a > 0$ tedy nalevo od x_i. V bodu $x_0 = 0$ užijeme okrajovou podmínku, takže $u_0^{n+1} = g_0(t_{n+1})$.

Dá se ukázat, že když

$$\nu := \frac{a\tau}{h} \leq 1\,, \tag{10.99}$$

pak za předpokladu dostatečné hladkosti přesného řešení pro chybu platí

$$u(x_i, t_n) - u_i^n = O(\tau)\,, \tag{10.100}$$

viz [29]. Říkáme, že metoda (10.98) je upwind metoda řádu jedna. Pro $\nu = 1$ dokonce $u_i^n = u(x_i, t_n)$ je přesné! Číslo $\nu = a\tau/h$ se nazývá *Courantovo číslo* a podmínka (10.99) se nazývá *Courantova-Friedrichsova-Lewyova podmínka*, stručně *CFL podmínka*.

Metodu řádu dva dostaneme tak, že v (10.96) použijeme interpolační polynom druhého stupně. Když k podmínkám (10.97) pro $k = 2$ přidáme ještě podmínku

$$P_2(x_{i+1}) = u_{i+1}^n\,, \tag{10.101}$$

vypočteme $P_2(x_i^n)$ a dosadíme do (10.96), dostaneme *Laxovu-Wendroffovu* metodu

$$u_i^{n+1} = u_i^n - a\tau\frac{u_{i+1}^n - u_{i-1}^n}{2h} + \frac{1}{2}(a\tau)^2\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}\,. \tag{10.102}$$

V bodu x_0 užijeme okrajovou podmínku, takže $u_0^{n+1} = g_0(t_{n+1})$. Pro aproximaci v uzlu x_N můžeme použít interpolační polynom $P_2(x)$, který kromě podmínek (10.97) vyžaduje navíc splnění podmínky $P_2(x_{N-2}) = u_{N-2}^n$.

Jestliže pro obecný uzel x_i přidáme k podmínkám (10.97) podmínku

$$P_2(x_{i-2}) = u_{i-2}^n\,, \tag{10.103}$$

vypočteme $P_2(x_i^n)$ a dosadíme do (10.96), obdržíme *Beamovu-Warmingovu* metodu

$$u_i^{n+1} = u_i^n - a\tau\frac{3u_i^n - 4u_{i-1}^n + u_{i-2}^n}{2h} + \frac{1}{2}(a\tau)^2\frac{u_i^n - 2u_{i-1}^n + u_{i-2}^n}{h^2}\,. \tag{10.104}$$

Jestliže $u_0^{n+1} = g_0(t_{n+1})$ a počítáme-li u_i^{n+1}, $i = 1, 2, \ldots, N-1$, podle (10.102) a u_N^{n+1} podle (10.104), je-li splněna CFL podmínka (10.99) a je-li přesné řešení u dostatečně hladké, pro chybu platí

$$u(x_i, t_n) - u_i^n = O(\tau^2)\,. \tag{10.105}$$

Jestliže $u_0^{n+1} = g_0(t_{n+1})$ a počítáme-li u_1^{n+1} podle (10.102) a u_i^{n+1}, $i = 2, 3, \ldots, N$, podle (10.104), je-li splněna CFL podmínka (10.99) a je-li přesné řešení u dostatečně hladké, pro chybu opět platí (10.105).

Věnujme se dále případu, kdy konstanta $a < 0$. Pak okrajová podmínka bude předepsána vpravo, tj. podmínku (10.93_2) nahradí podmínka

$$u(\ell, t) = g_\ell(t)\,, \qquad t \in (0, T)\,. \tag{$10.93_2'$}$$

Charakteristika vycházející z bodu $[x_i, t_{n+1}]$ bude směřovat doprava, takže za předpokladu platnosti CFL podmínky

$$\nu := \frac{|a|\tau}{h} \le 1 \tag{10.106}$$

bude $x_i(t_n) \in \langle x_i, x_{i+1}\rangle$ pro $i = 0, 1, \ldots, N-1$. Podobně jako dříve odvodíme upwind metodu

$$u_i^{n+1} = u_i^n - a\tau\frac{u_{i+1}^n - u_i^n}{h}. \tag{10.98'}$$

Laxova-Wendroffova metoda na znaménku a nezávisí, takže opět platí (10.102). Pro Beamovu-Warmingovu metodu dostaneme

$$u_i^{n+1} = u_i^n - a\tau\frac{-3u_i^n + 4u_{i+1}^n - u_{i+2}^n}{2h} + \frac{1}{2}(a\tau)^2\frac{u_i^n - 2u_{i+1}^n + u_{i+2}^n}{h^2}. \tag{10.104'}$$

V obecném případě $a = a(x, t)$ určíme x_i^n numericky: $x_i^n \approx x_i(t_n)$, kde

$$\frac{\mathrm{d}x_i(t)}{\mathrm{d}t} = a(x_i(t), t)\,, \qquad x_i(t_{n+1}) = x_i\,. \tag{10.106}$$

Pro upwind metodu použijeme EE metodu, takže

$$x_i^n = x_i - a_i^n\tau\,, \quad \text{kde} \quad a_i^n = a(x_i, t_{n+1})\,.$$

Pro Laxovu-Wendroffovu metodu a Beamovu-Warmingovu použijeme EM2 metodu, tj.

$$x_i^n = x_i - a_i^n\tau, \quad \text{kde} \quad a_i^n = \tfrac{1}{2}(k_1+k_2), \quad k_1 = a(x_i, t_{n+1})\,,\ k_2 = a(x_i - \tau k_1, t_n)\,.$$

Vzorce (10.98), (10.98'), (10.102), (10.104) a (10.104') se změní jen v tom, že v nich místo a píšeme a_i^n. Délka kroku musí splňovat CFL podmínku

$$\frac{\max_i |a_i^n|\tau}{h} \le 1. \tag{10.106'}$$

Jestliže v rovnici (10.93_1) uvažujeme nenulovou pravou stranu $f(x,t)$, tj. řešíme-li místo rovnice (10.94) rovnici

$$\frac{\mathrm{d}u(x(t),t)}{\mathrm{d}t} = f(x(t),t), \tag{10.94'}$$

přičteme k pravým stranám formulí aproximaci $Q_i^n(f)$ integrálu $\int_{t_n}^{t_{n+1}} f(x_i(t),t)\,\mathrm{d}t$. Pro upwind metodu stačí použít jednostrannou obdélníkovou formuli

$$Q_i^n(f) = \tau f(x_i, t_{n+1}).$$

Pro Laxovu-Wendroffovu metodu a Beamovu-Warmingovu metodu užijeme lichoběžníkovou formuli

$$Q_i^n(f) = \tfrac{1}{2}\tau[f(x_i,t_{n+1}) + f(x_i^n, t_n)].$$

Shrnutí. Upwind metodu řádu jedna pro rovnici (10.84) zapíšeme ve tvaru

$$u_i^{n+1} = u_i^n - [a_i^{n+1}\tau]^+ \frac{u_i^n - u_{i-1}^n}{h} + [a_i^{n+1}\tau]^- \frac{u_{i+1}^n - u_i^n}{h} + \tau f_i^{n+1}, \tag{10.108}$$

kde $f_i^{n+1} = f(x_i, t_{n+1})$. Připomeňme, že $c^+ = \max(c,0)$ a $c^- = \min(c,0)$.

Beamovu-Warmingovu metodu, kterou lze považovat za upwind metodu řádu dva, zapíšeme ve tvaru

$$\begin{aligned} u_i^{n+1} = u_i^n &- [a_i^{n+1}\tau]^+ \frac{3u_i^n - 4u_{i-1}^n + u_{i-2}^n}{2h} + \frac{1}{2}\left([a_i^{n+1}\tau]^+\right)^2 \frac{u_i^n - 2u_{i-1}^n + u_{i-2}^n}{h^2} \\ &- [a_i^{n+1}\tau]^- \frac{-3u_i^n + 4u_{i+1}^n - u_{i+2}^n}{2h} + \frac{1}{2}\left([a_i^{n+1}\tau]^-\right)^2 \frac{u_i^n - 2u_{i+1}^n + u_{i+2}^n}{h^2} \\ &+ \frac{1}{2}\tau[f_i^{n+1} + f(x_i^n, t_n)]. \end{aligned} \tag{10.109}$$

Laxova-Wendrofova metoda řádu dva, založená na centrálních diferencích, je tvaru

$$u_i^{n+1} = u_i^n - a_i^n\tau\frac{u_{i+1}^n - u_{i-1}^n}{2h} + \frac{1}{2}(a_i^n\tau)^2\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} + \frac{1}{2}\tau[f_i^{n+1} + f(x_i^n, t_n)]. \ \square \tag{10.110}$$

Metoda konečných objemů. Pro diskretizaci rovnice (10.84') se výborně hodí metoda konečných objemů na časoprostorových objemech $B_i = \langle x_{i-1/2}, x_{i+1/2}\rangle \times \langle t_n, t_{n+1}\rangle$, více o tom viz [28].

Literatura

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J.Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen: *LAPACK User's Guide,* Third Edition, SIAM, Philadelphia, 1999, [on line], dostupné z `http://www.netlib.org/lapack/lug`.

[2] R. Ashino, M. Nagase, R. Vaillancourt: *A survey of the MATLAB ODE suite,* Technical Report CRM-2651, Centre de recherches mathematiques, University of Ottawa, 2000.

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. V. Vorst: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, Philadelphia, 2000, [on line], dostupné z `http://www.cs.utk.edu/~dongarra/etemplates`

[4] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, Ch. Romine, H. V. Vorst: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods,* SIAM, Philadelphia, 1994, [on line], dostupné z `http://www.netlib.org/templates`.

[5] K. J. Bathe: *Finite Elements Procedures,* Prentice-Hall, Upper Saddle River, NJ, 1996.

[6] I.S. Berezin, N.P. Židkov: *Čislennyje metody I,II,* Nauka, Moskva, 1962.

[7] J. P. Berrut, L. N. Trefethen: *Barycentric Lagrange Interpolation,* SIAM Rev., Vol. 46, No. 3, pp. 501–517, 2004.

[8] J.P. Berrut, A. Welscher: *Fourier and barycentric formulae for equidistant Hermite trigonometric interpolation,* Appl. Comput. Harmon. Anal. 23 (2007), 307–320.

[9] M. Brandner, J. Egermaier, H. Kopincová: *Numerické metody pro řešení evolučních parciálních diferenciálních rovnic,* učební text ZČU a VŠB, Plzeň, 2012.

[10] G. Dahlquist, G. Å Bjőrk: *Numerical Methods,* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[11] J. W. Demmel: *Applied Numerical Linear Algebra,* SIAM, Philadelphia, 1997.

[12] M. O. Deville, P. F. Fischer, E. H. Mund: *Higher Order Methods for Incompressible Flow,* Cambridge University Press, Cambridge, 2002.

[13] J. Donea, A. Huerta: *Finite Element Methods for Flow Problems,* John Wiley & Sons Ltd, Chichester, 2003.

[14] D. R. Durran: *Numerical Methods for Fluid Dynamics: with Application to Geophysics (2nd edition),* Springer, New York, 2010.

[15] M. Feistauer, J. Felcman, I. Straškraba: *Mathematical and Computational Methods for Compressible Flow,* Oxford Science Publications, New York, 2003.

[16] J. H. Ferziger, M. Perić: *Computational Methods for Fluid Dynamics (3rd edition),* Springer, Berlin, 2002.

[17] M. Fiedler: *Speciální matice a jejich použití v numerické matematice,* SNTL, Praha, 1981.

[18] J. Fish, T. Belytschko: *A First Course in Finite Elements,* John Willey & Sons Ltd, Chichester, 2007.

[19] W. Gander, W. Gautschi: *Adaptive Quadrature - Revisited,* BIT, Vol. 40, No. 1 (2000), str. 84–101.

[20] G. Hämmerlin, K. H. Hoffmann: *Numerical Mathematics,* Springer-Verlag, Berlin, 1991.

[21] M. T. Heath: *Scientific Computing. An Introductory Survey,* McGraw-Hill, New York, 2002.

[22] I. Horová, J. Zelinka: *Numerické metody,* učební text Masarykovy univerzity, Brno, 2004.

[23] P. Knabner, L. Angermann: *Numerical Methods for Elliptic and Parabolic Partial Differential Equations,* Springer, New York, 2003.

[24] J. Kobza: *Splajny,* učební text Palackého univerzity, Olomouc, 1993.

[25] A. N. Kolmogorov, S. .V. .Fomin: *Základy teorie funkcí a funkcionální analýzy,* SNTL, Praha, 1975.

[26] J. D. Lambert: *Numerical Methods in Ordinary Differential Systems. The Initial Value Problem,* J. Willey & Sons, Chichester, 1993.

[27] R. B. Lehoucq, D. C. Sorensen, C. Young: *Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods,* SIAM, Philadelphia, 1998, [on line], dostupné z `http://www.caam.rice.edu/software/ARPACK`.

[28] R. J. LeVecque: *Finite Volume Methods for Hyperbolic Problems,* Cambridge University Press, Cambridge, 2002.

[29] R. J. LeVecque: *Finite Difference Methods for Ordinary and Partial Differential Equations,* Siam, Philadelphia, 2007.

[30] C. F. Van Loan, G. H. Golub: *Matrix Computations,* 3th ed., Johns Hopkins University Press, Baltimore, 1996.

[31] J. H. Mathews, K. D. Fink: *Numerical Methods Using MATLAB,* Pearson Prentice Hall, New Jersey, 2004.

[32] The MathWorks, Inc.: *MATLAB® Mathematics*, R2019b, [on-line], dostupné z `http://www.mathworks.com/help/pdf_doc/matlab/math.pdf.`

[33] G. Meurant: *Computer Solution of Large Linear Systems,* Elsevier, Amsterodam, 1999.

[34] S. Míka: *Numerické metody algebry,* SNTL, Praha, 1985.

[35] S. Míka, P. Přikryl: *Numerické metody řešení obyčejných diferenciálních rovnic, okrajové úlohy,* učební text ZČU Plzeň, 1994.

[36] S. Míka, P. Přikryl, M. Brandner: *Speciální numerické metody: Numerické metody řešení okrajových úloh pro diferenciální rovnice,* Vydavatelský servis, Plzeň, 2006.

[37] C. B. Moler: *Numerical Computing with MATLAB,* Siam, Philadelphia, 2004, [on line], dostupné z `http://www.mathworks.com/moler`.

[38] J. Nečas: *Aplikovaná matematika I,II, oborová encyklopedie,* SNTL, 1977.

[39] J. Nocedal, S. J. Wright: *Numerical Optimization, Springer Series in Operations Research,* Springer, Berlin, 1999.

[40] 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.

[41] P. Přikryl: *Numerické metody matematické analýzy,* SNTL, Praha, 1985.

[42] A. Quarteroni, R. Sacco, F. Saleri: *Numerical Mathematics,* Springer, Berlin, 2000.

[43] A. R. Ralston: *Základy numerické matematiky,* Academia, Praha, 1973.

[44] K. Rektorys: *Přehled užité matematiky I,II,* Prometheus, Praha, 1995.

[45] Y. Saad: *Numerical Methods for Large Eigenvalue Problems,* John Wiley & Sons, New York, 1992.

[46] Y. Saad: *Iterative Methods for Sparse Linear Systems,* 2nd ed., SIAM, Philadelphia, 2003.

[47] E. Scheiber: *On the Interpolation Trigonometric Polynomial with an Arbitrary Even Number of Nodes,* 2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, 2011, pp. 71-74.

[48] L. F. Shampine: *Vectorized adaptive quadrature in MATLAB,* Journal of Computational and Applied Mathematics, 211, 2008, str. 131–140.

[49] L. F. Shampine: *Some practical Runge-Kutta formulas,* Math. Comp., Vol. 46, Num. 173 (1986), str. 135-150.

[50] L. F. Shampine: *Numerical Solution of ordinary differential equations,* Chapman & Hall, New York, 1994.

[51] L. F. Shampine, I. Gladwell, S. Thompson: *Solving ODEs with MATLAB,* Cambridge University Press, Cambridge, 2003.

[52] L. F. Shampine, L. W. Reichelt: *The MATLAB ODE suite,* SIAM J. Sci. Comput., Vol. 18 (1997), No. 1, str. 1-22.

[53] F. Šik: *Lineární algebra,* Masarykova univerzita, Brno, 1998.

[54] K. Šolín, K. Segeth, I. Doležel: *Higher-Order Finite Element Method,* Chapman & Hall/CRC, Boca Raton, 2004.

[55] J. Stoer, R. Bulirsch: *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1993.

[56] L. N. Thefethen, D. Bau, III: *Numerical Linear Algebra,* Siam, Philadelphia, 1997.

[57] H. K. Versteeg, W. Malalasekera: *An Introduction to Computational Fluid Dynamics: The finite volume method (2nd edition),* Prentice Hall, Harlow, 2007.

[58] E. Vitásek: *Numerické metody,* SNTL, Praha, 1987.

[59] E. Vitásek: *Základy teorie numerických metod pro řešení diferenciálních rovnic,* Academia, Praha, 1994.

[60] W. Y. Yang, W. Cao, T. S. Chung, J. Morris: *Applied Numerical Methods Using Matlab,* John Willey & Sons, New Jersey, 2005.

[61] O. C. Zienkiewicz, R. L. Taylor: *The Finite Element Method, Volume I: The Basis,* Butterworth-Heinemann, Oxford, 2000.

[62] A. Ženíšek: *Lebesguevův integrál a základy funkcionální analýzy.,* skripta FSI VUT, PC-DIR, Brno 1999.

[63] A. Ženíšek: *Numerická integrace na trojúhelníkových prvcích,* Vodohosp. Čas. 25, č. 6, strany 586-593, 1977.