Poznámky k předmětu Numerická lineární algebra I.

Michal Merta*

*Katedra aplikované matematiky, VŠB-Technická univerzita Ostrava, e-mail: michal.merta@vsb.cz

1 Iterační metody pro řešení soustav lineárních rovnic

Přímé metody řešení soustav lineárních rovnic (LU, LDLT, Choleského faktorizace atd.) vyžadují $\mathcal{O}(n^3)$ operací a velikost soustavy, kterou jimi dokážeme vyřešit, je značně omezená. V případě husté matice $\mathsf{A} \in \mathbb{R}^{n\times n}$ se přibližná velikost řešitelné soustavy historicky vyvíjela takto:

- 1950: $n = 20$ (Wilkinson)
- 1965: $n = 200$ (Forsythe a Moler)
- 1980: $n = 2000$ (LINPACK)
- 1995: $n = 20000$ (LAPACK)
- 2010: $n = 200000$ (HDSS)

Pracujeme-li s řídkými maticemi, jsme schopni řešit i systémy s mnohem větší dimenzí (miliony, desítky milionů neznámých) – zejména, pokud je řešič schopen pracovat paralelně. V případě použití přímého řešiče na řídkou matici však může dojít k jejímu zaplnění. Např. využitím prvního řádku následující matice k vynulování prvního sloupce dojde k zaplnění všech ostatních prvků v matici:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 \\ \times & 0 & 0 & \times & 0 \\ \times & 0 & 0 & 0 & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}$$

Tento problém lze částečně řešit vhodnou pivotizací.

Mezi další nevýhody přímých řešičů patří:

- Známe-li již přibližné řešení soustavy, nedokážeme tuto znalost využít ke snížení celkového počtu operací a zkrácení doby výpočtu.
- Naopak, pokud nám postačuje znalost pouze přibližného řešení, nemůžeme výpočet pomocí přímého řešiče ukončit předčasně.

Alternativou k přímým řešičům jsou iterační řešiče, které generují posloupnost přibližných řešení $\{\boldsymbol{x}^k\}$ a pracují téměř výhradně s násobením matice-vektor, které má náročnost $\mathcal{O}(n^2)$. Důležitou vlastností každé iterační metody je rychlost konvergence posloupnosti $\{\boldsymbol{x}^k\}$ k řešení. Může se totiž stát, že pro některé matice A iterační metoda konverguje velmi pomalu nebo vůbec.

1.1 Lineární iterační metody

Prvním typem iteračních metod, kterým se budeme zabývat, jsou tzv. lineární iterační metody. Ty hledají posloupnost řešení soustavy $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ s regulární maticí ve tvaru

$$\boldsymbol{x}^{k+1} := \mathsf{M}\boldsymbol{x}^k + \mathsf{N}\boldsymbol{b}, \tag{1.1}$$

kde M a N jsou nějaké matice odpovídajících rozměrů[1].

Definice Lineární iterační metodu nazveme konzistentní, řeší-li rovnici $\mathsf{M}\boldsymbol{x} + \mathsf{N}\boldsymbol{b} = \boldsymbol{x}$ právě jeden vektor $\boldsymbol{x} = \mathsf{A}^{-1}\boldsymbol{b}$. Je možné ukázat, že metoda je konzistentní právě tehdy, je-li splněno $\mathsf{M} = \mathsf{I} - \mathsf{NA}$.

Definice Iterační metodu nazveme konvergentní platí-li $\forall \boldsymbol{b}, \forall \boldsymbol{x}^0 : \boldsymbol{x}^k \to \boldsymbol{x} = \mathsf{A}^{-1}\boldsymbol{b}$ pro $k \to \infty$. Je možné ukázat, že metoda je konvergentní, právě tehdy, je-li splněno $\|\mathsf{M}\| < 1$, kde $\|\mathsf{M}\| = \max_{\boldsymbol{v}\in\mathbb{R}^n, \boldsymbol{v}\neq\boldsymbol{o}} \frac{\|\mathsf{M}\boldsymbol{v}\|_v}{\|\boldsymbol{v}\|_v}$ je maticová norma indukovaná vektorovou normou.

Při odvozování následujících iteračních metod budeme využívat rozkladu matice A na součet dolní trojúhelníkové, diagonální a horní trojúhelníhové matice, tedy $\mathsf{A} = \mathsf{L} + \mathsf{D} + \mathsf{U}$ (pozor, nepleťte si matice $\mathsf{L}, \mathsf{D}, \mathsf{U}$ se stejně nazvanými maticemi, které se vyskytovaly u přímých řešičů). Např. matici

$$\mathsf{A} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

rozložíme na

$$\mathsf{L} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathsf{D} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \mathsf{U} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

1.1.1 Jacobiho metoda

Vyjděme z rovnice $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$. Dosazením $\mathsf{A} = \mathsf{L} + \mathsf{D} + \mathsf{U}$ dostaneme

$$(\mathsf{L} + \mathsf{D} + \mathsf{U})\boldsymbol{x} = \boldsymbol{b}.$$

Roznásobme a přezávorkujme výraz na levé straně

$$\mathsf{D}\boldsymbol{x} + (\mathsf{L} + \mathsf{U})\boldsymbol{x} = \boldsymbol{b}$$

Jacobiho metodu odvodíme tak, že přidáme indexy $k+1$ a k k příslušným vektorům $\boldsymbol{x}$

$$\mathsf{D}\boldsymbol{x}^{k+1} + (\mathsf{L} + \mathsf{U})\boldsymbol{x}^k = \boldsymbol{b}.$$

Osamostatněním $\boldsymbol{x}^{k+1}$ dostaneme předpis pro $k+1$ aproximaci vektoru $\boldsymbol{x}$

$$\boldsymbol{x}^{k+1} := \mathsf{D}^{-1}(\boldsymbol{b} - (\mathsf{L} + \mathsf{U})\boldsymbol{x}^k) = \underbrace{-\mathsf{D}^{-1}(\mathsf{L} + \mathsf{U})}_{=\mathsf{M}}\boldsymbol{x}^k + \underbrace{\mathsf{D}^{-1}}_{=\mathsf{N}}\boldsymbol{b}.$$

[1] Index k označuje číslo aktuální iterace.

Jednotlivé složky vektoru $\boldsymbol{x}^{k+1}$ můžeme vyjádřit jako

$$(\boldsymbol{x}^{k+1})_i := \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{i,j} (\boldsymbol{x}^k)_j \right) = \tag{1.2}$$

$$= \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} (\boldsymbol{x}^k)_j - \sum_{j=i+1}^{n} a_{i,j} (\boldsymbol{x}^k)_j \right) \tag{1.3}$$

Pro snadnější a přehlednější zápis budeme i-tý prvek vektoru $\boldsymbol{x}^{k+1}$ také značit jako x_i^{k+1} (horní index tedy označuje číslo iterace, dolní index značí pořadí prvku ve vektoru).

Konzistence metody vyplývá z jejího odvození, můžeme však ještě ověřit, že $\mathsf{M} = \mathsf{I} - \mathsf{NA}$. V případě Jacobiho metody je $\mathsf{M} = -\mathsf{D}^{-1}(\mathsf{L} + \mathsf{U})$ a $\mathsf{N} = \mathsf{D}^{-1}$ (viz výše). Platí tedy

$$\begin{aligned} \mathsf{I} - \mathsf{NA} = \mathsf{I} - \mathsf{D}^{-1}\mathsf{A} = \mathsf{I} - \mathsf{D}^{-1}(\mathsf{L} + \mathsf{D} + \mathsf{U}) = \\ = \mathsf{I} - \mathsf{D}^{-1}(\mathsf{L} + \mathsf{U}) - \mathsf{D}^{-1}\mathsf{D} = -\mathsf{D}^{-1}(\mathsf{L} + \mathsf{U}) = \mathsf{M}. \end{aligned}$$

Metoda je tedy konzistentní.

Lze dokázat, že metoda je konvergentní právě tehdy, když $\|\mathsf{M}\| = \|\mathsf{D}^{-1}(\mathsf{L} + \mathsf{U})\| < 1$. To splňují např. striktně diagonálně dominantní matice (tedy matice, pro které platí $\forall i : |a_{i,i}| > \sum_{j=1, j\neq i}^{n} |a_{i,j}|$). Konvergenci metody pro diagonálně dominantní matice se dá poměrně snadno dokázat. Vyjděme ze vztahu pro i-tý prvek aproximovaného vektoru v iteraci $k+1$:

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

Jelikož je metoda konzistentní, musí tuto rovnost splňovat i prvky vektoru přesného řešení $\boldsymbol{x}$:

$$x_i = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{i,j} x_j \right).$$

Odečteme-li od první rovnosti druhou, dostaneme

$$\underbrace{x_i^{k+1} - x_i}_{=e_i^{k+1}} = -\frac{1}{a_{i,i}} \sum_{j=1, j\neq i}^{n} a_{i,j} \underbrace{(x_j^k - x_j)}_{=e_i^k},$$

kde $\boldsymbol{e}^k$ je vektor chyby v k-tém kroku. Chybu v kroku $k+1$ můžeme tedy odhadnout pomocí vlastností striktně diagonálně dominantní matice:

$$|e_i^{k+1}| \leq \frac{1}{|a_{i,i}|} \sum_{j=1, j\neq i}^{n} |a_{i,j}||e_j^k| \leq \frac{1}{|a_{i,i}|} \sum_{j=1, j\neq i}^{n} |a_{i,j}| \max_{j=1,\ldots,n, j\neq i} |e_j^k| =$$

$$= \max_{j=1,\ldots,n, j\neq i} |e_j^k| \underbrace{\frac{1}{|a_{i,i}|} \sum_{j=1, j\neq i}^{n} |a_{i,j}|}_{<1} < \max_{j=1,\ldots,n, j\neq i} |e_j^k|$$

Každý prvek vektoru chyby v kroku $k+1$ je tedy v absolutní hodnotě menší než maximální prvek vektoru chyby v předchozím kroku. Vektor chyby tedy konverguje k nulovému vektoru.

1.1.2 Gaussova-Seidelova metoda

Všimněme si, že při výpočtu x_i^{k+1} využíváme v sumě $\sum_{j=1}^{i-1} a_{i,j} x_j^k$ ve výrazu (1.3) pouze prvky $x_1^k, \ldots, x_{i-1}^k$. Tyto prvky tedy můžeme nahradit již vypočtenými prvky aktuálními iterace $x_1^{k+1}, \ldots, x_{i-1}^{k+1}$. Dostaneme tak předpis Gaussovy-Seidelovy metody:

$$x_i^{k+1} := \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) \tag{1.4}$$

Podobně jako v předchozím případě můžeme metodu odvodit, nahradíme-li v soustavě $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ matici A součtem $\mathsf{L} + \mathsf{D} + \mathsf{U}$. Tentokrát ovšem vektor s indexem $k+1$ ponecháme u součtu $\mathsf{L} + \mathsf{D}$

$$(\mathsf{L} + \mathsf{D})\boldsymbol{x}^{k+1} = \boldsymbol{b} - \mathsf{U}\boldsymbol{x}^k, \tag{1.5}$$

tedy

$$\boldsymbol{x}^{k+1} = \underbrace{-(\mathsf{L} + \mathsf{D})^{-1}\mathsf{U}}_{=\mathsf{M}} \boldsymbol{x}^k + \underbrace{(\mathsf{L} + \mathsf{D})^{-1}}_{=\mathsf{N}} \boldsymbol{b}.$$

Vztah mezi maticovým zápisem a zápisem po prvcích (1.4) je nejlépe vidět na rovnosti (1.5). Jedná se o soustavu rovnic s dolní trojúhelníkovou maticí $\mathsf{L} + \mathsf{D}$, vektorem pravé strany $\boldsymbol{b} - \mathsf{U}\boldsymbol{x}^k$ a neznámým vektorem $\boldsymbol{x}^{k+1}$. Všimněte si, že výraz (1.4) pak přesně odpovídá algoritmu pro dopřednou substituci pro řešení takovéto soustavy.

Podobně jako v případě Jacobiho metody můžeme ověřit, zda je metoda konzistentní porovnáním $\mathsf{I} - \mathsf{NA}$ a M.

$$\mathsf{I} - \mathsf{NA} = \mathsf{I} - (\mathsf{L} + \mathsf{D})^{-1}\mathsf{A} = \mathsf{I} - (\mathsf{L} + \mathsf{D})^{-1}(\mathsf{L} + \mathsf{D} + \mathsf{U}) =$$
$$= \mathsf{I} - (\mathsf{L} + \mathsf{D})^{-1}(\mathsf{L} + \mathsf{D}) - (\mathsf{L} + \mathsf{D})^{-1}\mathsf{U} = -(\mathsf{L} + \mathsf{D})^{-1}\mathsf{U} = \mathsf{M}.$$

Metoda je tedy konzistentní.

Metoda je konvergentní, právě když $\|(\mathsf{L} + \mathsf{D})^{-1}\mathsf{U}\| < 1$, což opět platí pro diagonálně dominantní matice.

1.1.3 Richardsonova metoda

Iterace Richardsonovy metody je dána předpisem

$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \omega \boldsymbol{r}^k,$$

kde $\omega \in \mathbb{R}_+$ a $\boldsymbol{r}^k = \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^k$ je reziduum, které určuje, jak dobře je splněna původní rovnice. Vztah mezi reziduem a chybou $\boldsymbol{e}^k = \boldsymbol{x}^k - \boldsymbol{x}$ lze odvodit přenásobením definice chyby maticí A

$$\mathsf{A}\boldsymbol{e}^k = \mathsf{A}\boldsymbol{x}^k - \mathsf{A}\boldsymbol{x} = \mathsf{A}\boldsymbol{x}^k - \boldsymbol{b} = -\boldsymbol{r}^k. \tag{1.6}$$

Studujme konvergenci metody pro symetrickou pozitivně definitní matici A. V takovém případě vlastní čísla λ_i a vlastní vektory $\boldsymbol{v}_i$ (tedy skaláry a vektory, pro které platí $\mathsf{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, \|\boldsymbol{v}\| = 1$) splňují

$$0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

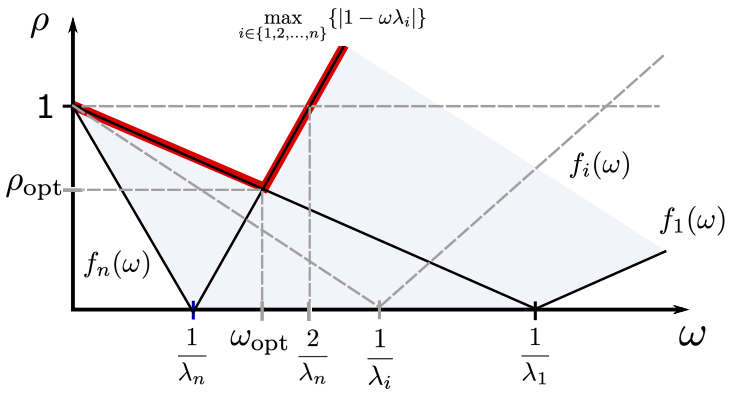
a z vlastních vektorů lze utvořit ortonormální bázi $\mathbb{R}^n$.

Díky předchozímu poznatku můžeme reziduum vyjádřit jako lineární kombinaci prvků báze tvořené vlastními vektory, tedy $\boldsymbol{r}^{k+1} = \sum_{i=1}^n \alpha_i^{k+1} \boldsymbol{v}_i$. Studujme nyní, jak se chová reziduum (a tedy i chyba) v jednotlivých iteracích. Na základě toho se později pokusíme odvodit optimální hodnotu ω pro co nejrychlejší konvergenci.

$$\begin{aligned}
\sum_{i=1}^n \underline{\alpha_i^{k+1}} \boldsymbol{v}_i = \boldsymbol{r}^{k+1} &= \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{k+1} = \underbrace{\boldsymbol{b} - \mathsf{A}(\boldsymbol{x}^k}_{=\boldsymbol{r}^k} + \omega \boldsymbol{r}^k) = \\
&= \boldsymbol{r}^k - \mathsf{A}\omega\boldsymbol{r}^k = (\mathsf{I} - \omega\mathsf{A}) \underbrace{\boldsymbol{r}^k}_{=\sum_{i=1}^n \alpha_i^k \boldsymbol{v}_i} = (\mathsf{I} - \omega\mathsf{A}) \sum_{i=1}^n \alpha_i^k \boldsymbol{v}_i = \\
&= \sum_{i=1}^n \alpha_i^k \boldsymbol{v}_i - \sum_{i=1}^n \alpha_i^k \omega \underbrace{\mathsf{A}\boldsymbol{v}_i}_{=\lambda_i \boldsymbol{v}_i} = \sum_{i=1}^n \alpha_i^k \boldsymbol{v}_i - \sum_{i=1}^n \alpha_i^k \omega \lambda_i \boldsymbol{v}_i = \\
&= \sum_{i=1}^n \underline{(1 - \omega\lambda_i)\alpha_i^k} \boldsymbol{v}_i.
\end{aligned}$$

Všimněme si, že jsme vyjádřili koeficienty rozvoje rezidua $\boldsymbol{r}^{k+1}$ v bázi $\{\boldsymbol{v}_i\}_{i=1}^n$ pomocí násobků koeficientů v předchozím kroku (viz podtržené části předchozího výrazu). Koeficienty se tedy budou zmenšovat (a jednotlivé složky vektoru rezidua budou konvergovat k nule) právě tehdy, když $|1 - \omega\lambda_i| < 1$ pro všechna $i = 1, 2, \ldots, n$. Rychlost konvergence bude záviset na největší hodnotě $|1 - \omega\lambda_i|$. Shrněme tento poznatek do následující věty.

Věta Richardsonova metoda konverguje, právě když $\forall i \in \{1, 2, \ldots, n\} : |1 - \omega\lambda_i| < 1$. Konvergenční faktor $\rho = \max_{i \in \{1,2,\ldots,n\}} \{|1 - \omega\lambda_i|\}$ určuje rychlost konvergence: $\|\boldsymbol{r}^{k+1}\| \leq \rho \|\boldsymbol{r}^k\|$.



Obrázek 1.1: Konvergenční faktor Richardsonovy metody v závislosti na ω.

Čím menší bude konvergenční faktor $\rho = \max_{i\in\{1,2,\dots,n\}}\{|1-\omega\lambda_i|\}$, tím rychleji bude metoda konvergovat. Vzhledem k tomu, že vlastní čísla matice A jsou daná, můžeme konvergenční faktor ovlivnit pouze vhodnou volbou parametru ω. Odvození ideální hodnoty ω ilustrujme na Obrázku 1.1. Jsou na něm znázorněny funkce $f_1(\omega) = |1-\omega\lambda_1|$ a $f_n(\omega) = |1-\omega\lambda_n|$. Protože směrnice funkcí $f_i(\omega) = |1-\omega\lambda_i|$ jsou určeny vlastními čísly matice A a ta jsou seřazena od nejmenšího po největší, budou grafy všech funkcí $f_i, i = 2,\dots,n-1$, ležet "mezi" grafy f_1 a f_n (v šedě vyznačené oblasti). Funkci

$$\max_{i\in\{1,2,\dots,n\}}\{|1-\omega\lambda_i|\}$$

tedy můžeme vykreslit jako červeně zvýrazněnou lomenou čáru tvořenou částí funkce f_1 a částí funkce f_n. Z grafu této funkce tedy rovnou můžeme odvodit:

1. Interval, ve kterém musí ω ležet. Aby metoda konvergovala, musí platit $\rho = \max_{i\in\{1,2,\dots,n\}}\{|1-\omega\lambda_i| < 1$. Červená funkce tedy musí ležet pod zakreslenou konstantní funkcí $\rho = 1$. Levý krajní bod intervalu je 0, pravý určíme jako průsečík příslušné části funkce f_n s konstantní funkcí 1:

   $$-(1-\omega\lambda_n) = 1 \quad \Rightarrow \quad \omega = \frac{2}{\lambda_n}.$$

   Metoda tedy konverguje pro $\omega \in (0, 2/\lambda_n)$.

2. Optimální ω je bod, ve kterém červeně vyznačená funkce dosahuje minima. Tento bod dostaneme jako průsečík funkcí f_1 a f_n:

   $$1-\omega_{\text{opt}}\lambda_1 = -(1-\omega_{\text{opt}}\lambda_n) \quad \Rightarrow \quad \omega_{\text{opt}} = \frac{2}{\lambda_1+\lambda_n}.$$

Zjistili jsme tedy, že nejlepší konvergence dosáhneme, zvolíme-li $\omega_{\text{opt}} = \frac{2}{\lambda_1+\lambda_n}$. Konvergenční faktor bude v tomto případě

$$\rho_{\text{opt}} = 1 - \omega_{\text{opt}}\lambda_1 = 1 - \frac{2\lambda_1}{\lambda_1+\lambda_n} = \frac{\lambda_1+\lambda_n-2\lambda_1}{\lambda_1+\lambda_n} = \\ = \frac{\lambda_n-\lambda_1}{\lambda_n+\lambda_1}\frac{\frac{1}{\lambda_1}}{\frac{1}{\lambda 1}} = \frac{\frac{\lambda_n}{\lambda_1}-1}{\frac{\lambda_n}{\lambda_1}+1} = \frac{\kappa(\mathsf{A})-1}{\kappa(\mathsf{A})+1},$$

kde $\kappa(\mathsf{A}) = \lambda_n/\lambda_1$ je číslo podmíněnosti matice A.

Můžeme také odvodit, kolik iterací je třeba, abychom dosáhli požadované relativní změny normy rezidua. Hledáme tedy k, pro které platí

$$\frac{\|\boldsymbol{r}^k\|}{\|\boldsymbol{r}^0\|} \le \varepsilon, \qquad \text{tedy} \qquad \|\boldsymbol{r}^k\| \le \varepsilon\|\boldsymbol{r}^0\|$$

Využijme toho, že $\|\boldsymbol{r}^k\| \le \rho_{\text{opt}}^k\|\boldsymbol{r}^0\|$ a přepišme nerovnici na

$$\rho_{\text{opt}}^k\|\boldsymbol{r}^0\| \le \varepsilon\|\boldsymbol{r}^0\|.$$

Vykrácením normy a zlogaritmováním obou stran nerovnice dostanem řešení $k \ge \frac{\log\varepsilon}{\log\rho_{\text{opt}}}$ (nezapomeňte, že protože $\rho_{\text{opt}} \in (0,1)$, je třeba otočit znaménko nerovnosti).

1.1.4 Ukončovací podmínky

Při použití iteračního řešiče většinou nemáme předem zadaný počet iterací, které mají proběhnout. Chceme výpočet ukončit ve chvíli, kdy se s odhadem řešení dostaneme dostatečně blízko přesnému řešení. Vzhledem k tomu, že přesné řešení (tedy ani přesnou chybu v dané iteraci) neznáme, musíme si pomoci jinak.

Jednou z možností je ukončit cyklus ve chvíli, kdy se s novým odhadem řešení příliš nepohneme od předchozího odhadu (tzn. $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| < \varepsilon$). Tato podmínka ale nijak nebere v potaz velikost prvků v matici soustavy a vektoru pravé strany (jiná situace nastane, pokud jsou prvky matice a vektoru v řádech tisíců, jiná pokud jsou v řádech tisícin). Proto je vhodné tuto podmínku zvolit relativně např. vzhledem k normě vektoru pravé strany (tzn. $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| < \|\boldsymbol{b}\|\varepsilon$, tedy $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|/\|\boldsymbol{b}\| < \varepsilon$).

Nejčastěji se ovšem k výpočtu ukončovací podmínky používá normy vektoru rezidua $\boldsymbol{r}^{k+1} = \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{k+1}$. To nám poskytuje přirozený odhad toho, jak dobře je splněna původní rovnice. Ukončovací podmínku lze tedy volit ve tvaru $\|\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{k+1}\| < \varepsilon$. Podobně jako v předchozím případě je i zde vhodnější použít relativní změnu rezidua oproti vektoru pravé strany ($\|\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{k+1}\|/\|\boldsymbol{b}\| < \varepsilon$) nebo počátečnímu reziduu ($\|\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{k+1}\|/\|\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^0\| < \varepsilon$).

1.2 Gradientní iterační metody

Věta Řešení soustavy $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ se symetrickou pozitivně definitní maticí A je ekvivalentní s minimalizací kvadratické formy

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}.$$

Důkaz Dokažme nejdříve implikaci $\mathsf{A}\boldsymbol{x} = \boldsymbol{b} \Rightarrow \boldsymbol{x} = \arg\min_{\boldsymbol{v}\in\mathbb{R}^n} f(\boldsymbol{v})$. Podívejme se, jak se změní funkční hodnota f, posuneme-li se z bodu $\boldsymbol{x}$ o nějaký nenulový vektor $\boldsymbol{c}$:

$$\begin{aligned} f(\boldsymbol{p}) = f(\boldsymbol{x}+\boldsymbol{c}) &= \frac{1}{2}(\boldsymbol{x}+\boldsymbol{c})^T\mathsf{A}(\boldsymbol{x}+\boldsymbol{c}) - \boldsymbol{b}^T(\boldsymbol{x}+\boldsymbol{c}) = \\ &= \frac{1}{2}\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} + \boldsymbol{c}^T\underbrace{\mathsf{A}\boldsymbol{x}}_{=\boldsymbol{b}} + \frac{1}{2}\boldsymbol{c}^T\mathsf{A}\boldsymbol{c} - \boldsymbol{b}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{c} = \\ &= \underbrace{\frac{1}{2}\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}}_{=f(\boldsymbol{x})} + \underbrace{\boldsymbol{c}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{c}}_{=0} + \frac{1}{2}\boldsymbol{c}^T\mathsf{A}\boldsymbol{c} = f(\boldsymbol{x}) + \underbrace{\frac{1}{2}\boldsymbol{c}^T\mathsf{A}\boldsymbol{c}}_{>0}. \end{aligned}$$

Díky pozitivní definitnosti A je výraz $\boldsymbol{c}^T\mathsf{A}\boldsymbol{c}$ kladný. Posuneme-li se tedy z bodu $\boldsymbol{x}$ v libovolném směru, hodnota funkce f se zvětší. V bodě $\boldsymbol{x}$ tedy nastává minimum.

K důkazu opačné implikace $\boldsymbol{x} = \arg\min_{\boldsymbol{v}\in\mathbb{R}^n} f(\boldsymbol{v}) \Rightarrow \mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ je třeba si uvědomit nutnou podmínku minima funkce $f : \mathbb{R}^n \to \mathbb{R}$, tedy nulovost gradientu:

$$\boldsymbol{x} = \arg\min_{\boldsymbol{v}\in\mathbb{R}^n} f(\boldsymbol{v}) \Rightarrow \nabla f(\boldsymbol{x}) = \boldsymbol{o}. \tag{1.7}$$

Lze ukázat, že pro gradient funkce f platí

$$\nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1}(\boldsymbol{x}), \ldots, \frac{\partial f}{\partial x_n}(\boldsymbol{x})\right]^T = \frac{1}{2}\mathsf{A}^T\boldsymbol{x} + \frac{1}{2}\mathsf{A}\boldsymbol{x} - b = \mathsf{A}\boldsymbol{x} - \boldsymbol{b}.$$

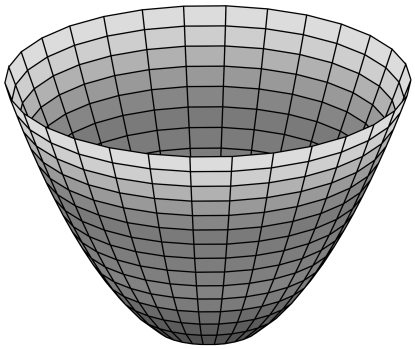
Z podmínky (1.7) tedy vyplývá $\mathsf{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{o}$. □

V případě symetrické pozitivně definitní matice $\mathsf{A} \in \mathbb{R}^{n\times n}$ máme tedy dvě možnosti, jak geometricky nahlížet na řešení soustavy lineárních rovnic. První přístup je chápat každou rovnici jako předpis nadroviny v n-rozměrném prostoru. Řešení soustavy pak odpovídá hledání průsečíku těchto rovin. Druhý přístup, který využijeme při odvozování následujících algoritmů, odpovídá minimalizaci příslušné pozitivně definitní kvadratické formy. Grafem pozitivně definitní kvadratické formy $f : \mathbb{R}^n \to \mathbb{R}$ je n-dimenzionální paraboloid, který má minimum (viz Obrázek 1.2 pro $n = 2$).

1.2.1 Metoda největšího spádu

Metoda největšího spádu je iterační metoda s předpisem

$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \alpha^k\boldsymbol{v}^k, \tag{1.8}$$



Obrázek 1.2: Graf kvadratické formy s pozitivně definitní maticí A (zdroj Wikipedia)

kde $\boldsymbol{v}^k$ volíme jako směr největšího poklesu funkce f. Všimněme si, že pro gradient platí $\nabla f(\boldsymbol{x}^k) = \mathsf{A}\boldsymbol{x}^k - \boldsymbol{b}$ a pro reziduum k-tém kroku $\boldsymbol{r}^k = \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^k$. Tedy $\boldsymbol{r}^k = -\nabla f(\boldsymbol{x}^k)$. Protože gradient odpovídá směru největšího růstu funkce v daném bodě, reziduum je směr největšího spádu. Logicky, protože chceme dosáhnout minima dané funkce, vydáváme se v každém kroku ve směru rezidua, tedy $\boldsymbol{v}^k = \boldsymbol{r}^k$.

Otázkou je, jak daleko se v každém kroku v tomto směru vydat, tedy jak zvolit koeficient α^k. Metoda největšího spádu volí tento koeficient tak, aby v každém kroku dosáhla minima funkce f ve směru rezidua. Definujme si tedy pomocnou funkci $F : \mathbb{R} \to \mathbb{R}$:

$$\begin{aligned} F(\alpha) = f(\boldsymbol{x}^k + \alpha\boldsymbol{r}^k) &= \frac{1}{2}(\boldsymbol{x}^k + \alpha\boldsymbol{r}^k)^T\mathsf{A}(\boldsymbol{x}^k + \alpha\boldsymbol{r}^k) - \boldsymbol{b}^T(\boldsymbol{x}^k + \alpha\boldsymbol{r}^k) = \\ &= \frac{1}{2}(\boldsymbol{x}^k)^T\mathsf{A}\boldsymbol{x}^k + \alpha(\boldsymbol{x}^k)^T\mathsf{A}\boldsymbol{r}^k + \frac{1}{2}\alpha^2(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k - \boldsymbol{b}^T\boldsymbol{x}^k - \alpha\boldsymbol{b}^T\boldsymbol{r}^k \end{aligned}$$

Hledáme α, ve kterém tato funkce dosahuje minima, její derivace se tedy musí rovnat nule:

$$F'(\alpha) = \alpha(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k + (\boldsymbol{r}^k)^T \underbrace{\mathsf{A}\boldsymbol{x}^k}_{=(\boldsymbol{b}-\boldsymbol{r}^k)} - \boldsymbol{b}^T\boldsymbol{r}^k = \alpha(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k - (\boldsymbol{r}^k)^T\boldsymbol{r} = 0$$

Odtud

$$\alpha^k = \frac{(\boldsymbol{r}^k)^T\boldsymbol{r}^k}{(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k}. \tag{1.9}$$

Stejný předpis můžeme odvodit, použijeme-li místo pomocné funkce funkci f a položíme její derivaci ve směru $\boldsymbol{r}^k$ rovnu nule (vzpomeňme si, že platí $\frac{\mathrm{d}f(\boldsymbol{x})}{\mathrm{d}\boldsymbol{h}} =$

$(\nabla f(\boldsymbol{x}))^T \boldsymbol{h})$:

$$\frac{\mathrm{d}f(x^{k+1})}{\mathrm{d}\boldsymbol{r}^k} = 0 \tag{1.10}$$

$$(\nabla f(\boldsymbol{x}^{k+1}))^T \boldsymbol{r}^k = 0 \tag{1.11}$$

$$(-\boldsymbol{r}^{k+1})^T \boldsymbol{r}^k = 0 \tag{1.12}$$

Dosazením $\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \alpha^k \mathsf{A}\boldsymbol{r}^k$ do předchozí rovnice a jednoduchou úpravou dostaneme stejný předpis pro α^k jako v předchozím případě (1.16). Předchozí odvození nám také prozradilo důležitou vlastnost metody největšího spádu – každý směr $\boldsymbol{v}^k = \boldsymbol{r}^k$ je kolmý na předchozí směr. Jak brzy uvidíme, není to vždy žádaná vlastnost.

Algoritmus tedy počítá jednotlivé aproximace pomocí následujících předpisů:

$$\begin{aligned}
\boldsymbol{r}^k &:= \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^k = \boldsymbol{b} - \mathsf{A}(\boldsymbol{x}^{k-1} + \alpha^{k-1}\boldsymbol{r}^{k-1}) = \underbrace{\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{k-1}}_{\boldsymbol{r}^{k-1}} - \alpha^{k-1}\mathsf{A}\boldsymbol{r}^{k-1} = \\
&= \boldsymbol{r}^{k-1} - \alpha^{k-1}\mathsf{A}\boldsymbol{r}^{k-1} \\
\alpha^k &:= \frac{(\boldsymbol{r}^k)^T\boldsymbol{r}^k}{(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k} \\
\boldsymbol{x}^{k+1} &:= \boldsymbol{x}^k + \alpha^k \boldsymbol{r}^k
\end{aligned}$$

Díky úpravě předpisu pro výpočet $\boldsymbol{r}^k$ jsme ušetřili jedno násobení matice-vektor ($\mathsf{A}\boldsymbol{x}^k$) – výsledek $\mathsf{A}\boldsymbol{r}^{k-1}$ si totiž můžeme zapamatovat z předchozí iterace.

Ukažme si nyní, že metoda konverguje. Konvergenci budeme dokazovat v tzv. energetické normě $\|\cdot\|_A$, tedy normě indukované skalárním součinem $(\mathsf{A}\boldsymbol{x}, \boldsymbol{x}) = \boldsymbol{x}^T\mathsf{A}\boldsymbol{x} : \|\boldsymbol{x}\|_A = \sqrt{\boldsymbol{x}^T\mathsf{A}\boldsymbol{x}}$.

$$\begin{aligned}
\|\boldsymbol{e}^{k+1}\|_A^2 &= (\boldsymbol{e}^{k+1})^T\mathsf{A}\boldsymbol{e}^{k+1} = (\boldsymbol{e}^k + \alpha^k\boldsymbol{r}^k)^T\mathsf{A}(\boldsymbol{e}^k + \alpha^k\boldsymbol{r}^k) = \\
&= (\boldsymbol{e}^k)^T\mathsf{A}\boldsymbol{e}^k + 2\alpha^k(\boldsymbol{r}^k)^T \underbrace{\mathsf{A}\boldsymbol{e}^k}_{=-\boldsymbol{r}^k} + (\alpha^k)^2(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k = \\
&= \|\boldsymbol{e}^k\|_A^2 - 2\underbrace{\frac{(\boldsymbol{r}^k)^T\boldsymbol{r}^k}{(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k}}_{=\alpha^k}(\boldsymbol{r}^k)^T\boldsymbol{r}^k + (\frac{(\boldsymbol{r}^k)^T\boldsymbol{r}^k}{(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k})^2(\boldsymbol{r}^k)\mathsf{A}\boldsymbol{r}^k = \\
&= \|\boldsymbol{e}^k\|_A^2 - \frac{((\boldsymbol{r}^k)^T\boldsymbol{r}^k)^2}{(\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k} = \\
&= \|\boldsymbol{e}^k\|_A^2\Big(1 - \frac{((\boldsymbol{r}^k)^T\boldsymbol{r}^k)^2}{((\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k)\ \underbrace{((\boldsymbol{e}^k)^T\mathsf{A}\boldsymbol{e}^k)}_{=(\mathsf{A}^{-1}\boldsymbol{r}^k)^T\mathsf{A}(\mathsf{A}^{-1}\boldsymbol{r}^k)=(\boldsymbol{r}^k)^T\mathsf{A}^{-1}\boldsymbol{r}^k}}\Big).
\end{aligned}$$

Rovnost v poslední závorce vyplývá ze vztahu (1.6) mezi reziduem a chybou. Zjistili jsme tedy, že chyba v kroku $k+1$ je nějakým násobkem chyby v před-

chozím kroku. Podívejme se na tento násobek podrobněji:

$$1 - \frac{((\boldsymbol{r}^k)^T\boldsymbol{r}^k)^2}{((\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k)((\boldsymbol{r}^k)^T\mathsf{A}^{-1}\boldsymbol{r}^k)} \leq 1 - \frac{((\boldsymbol{r}^k)^T\boldsymbol{r}^k)^2}{\|\mathsf{A}\|\|r^k\|^2\|\mathsf{A}^{-1}\|\|r^k\|^2} = \\ = 1 - \frac{1}{\|\mathsf{A}\|\|\mathsf{A}^{-1}\|} = q,$$

kde první nerovnost vyplývá z Cauchyho-Schwarzovy nerovnosti a definice maticové normy indukované vektorovou normou. Výraz $\|\mathsf{A}\|\|\mathsf{A}^{-1}\|$ nazývýme číslem podmíněnosti (viz také Richardsonova metoda) a v případě symetrické pozitivně definitní matice jej vypočítáme jako podíl největšího a nejmenšího vlastního čísla $\kappa(\mathsf{A}) = \frac{\lambda_{\max}}{\lambda_{\min}}$.

Protože $\kappa(\mathsf{A}) = \lambda_{\max}/\lambda_{\min} \geq 1$, platí $q = 1 - 1/\kappa(\mathsf{A}) < 1$. Chyba se tedy v každém kroku zmenšuje a metoda konverguje k řešení. Navíc

$$\|\boldsymbol{e}^{k+1}\|_A^2 \leq \left(1 - \frac{1}{\kappa(\mathsf{A})}\right)\|\boldsymbol{e}^k\|_A^2$$

a je možné dokázat vztah mezi počáteční chybou a chybou v k-té iteraci:

$$\|\boldsymbol{e}^{k+1}\|_A \leq \left(\frac{\kappa(\mathsf{A}) - 1}{\kappa(\mathsf{A}) + 1}\right)^k \|\boldsymbol{e}^0\|_A. \tag{1.13}$$

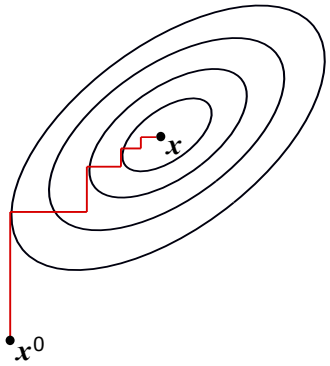
Metoda tedy konverguje k řešení pomalu. Zejména je-li $\kappa(\mathsf{A})$ velké, je zlomek ve výrazu (1.13) blízký 1 a dostaneme velmi pomalou konvergenci.

Celý algoritmus můžeme napsat na několika řádcích:

function STEEPESTD_DESCENT(A, $\boldsymbol{b}$, $\boldsymbol{x}^0$)
  $\boldsymbol{r}^0 = b - \mathsf{A}\boldsymbol{x}^0$
  $k = 0$
  while $\|\boldsymbol{r}^k\|/\|\boldsymbol{r}^0\| > \varepsilon$ **do**
    $\alpha_k = ((\boldsymbol{r}^k)^T\boldsymbol{r}^k)/((\boldsymbol{r}^k)^T\mathsf{A}\boldsymbol{r}^k)$
    $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k\boldsymbol{r}^k$
    $\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \alpha_k\mathsf{A}\boldsymbol{r}^k$
    $k = k + 1$
  end while
end function

Poznámka Z (1.12) vyplývá, že směr $\boldsymbol{r}^{k+1}$, ve kterém se v každém kroku vydáme, je kolmý na předchozí směr. To může vést k situacím jako na Obrázku 1.3, kdy procházíme „cik-cak“ údolí tvořené minimalizovanou kvadratickou formou.

Vydáváme se tedy několikrát v tom samém směru a postupně zkracujeme krok. Ideální by bylo najít metodu, která by minimalizovala celkovou chybu v daném směru vždy pouze jednou (v každé iteraci vynulovala chybu v daném směru). Taková metoda by byla schopná vyřešit soustavu o n neznámých během n iterací. Zkusme si tedy pro $\mathsf{A} \in \mathbb{R}^{n\times n}$ zvolit n navzájem ortogonálních směrů $\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1}$ (např. rovnoběžných s osami souřadnic) a použít předpis: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k\boldsymbol{d}^k$.



Obrázek 1.3: Pomalá konvergence metody největšího spádu (vrstevnice minimalizované kvadratické formy a jednotlivé směry $\boldsymbol{r}^k$).

Jak ale získat α_k? Ilustrujme to ve dvou dimenzích na Obrázku 1.4. Na začátku (Obrázek 1.4 vlevo) si zvolíme nějaký počáteční odhad řešení $\boldsymbol{x}^0$; počáteční chybu $\boldsymbol{e}^0$ pak lze rozložit na dvě kolmé složky rovnoběžné s osami souřadnic ($\boldsymbol{e}^0 = \boldsymbol{e}^0_x + \boldsymbol{e}^0_y$). V první iteraci si zvolme vektor $\boldsymbol{d}^0$ rovnoběžný s osou x (Obrázek 1.4 vpravo). Z obrázku je patrné, že abychom vynulovali chybu ve směru osy x, musíme bod $\boldsymbol{x}^1$ zvolit tak, aby vektor nové chyby $\boldsymbol{e}^1$ byl kolmý k $\boldsymbol{d}^0$.

Obecně tedy chceme, aby nová chyba byla kolmá k aktuálnímu směru:

$$(\boldsymbol{d}^k)^T \boldsymbol{e}^{k+1} = 0.$$

Z definice chyby dostaneme

$$0 = (\boldsymbol{d}^k)^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}) = (\boldsymbol{d}^k)^T(\boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k - \boldsymbol{x}) = (\boldsymbol{d}^k)^T(\boldsymbol{e}^k + \alpha_k \boldsymbol{d}^k)$$

a odtud

$$\alpha_k = -\frac{(\boldsymbol{d}^k)^T \boldsymbol{e}^k}{(\boldsymbol{d}^k)^T \boldsymbol{d}^k}.$$

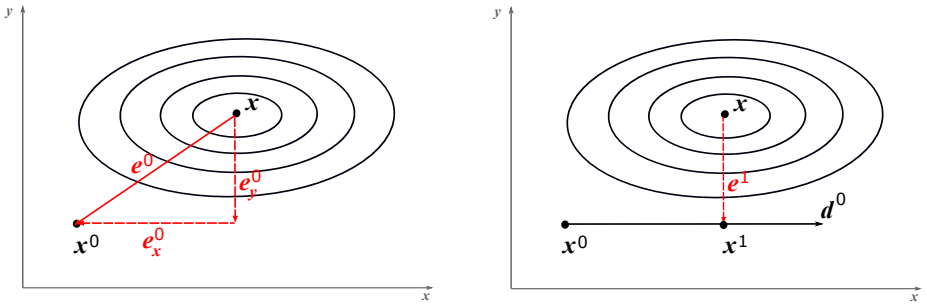
Je jasné, že jsme si příliš nepomohli. Abychom vypočítali α_k, potřebovali bychom znát chybu $\boldsymbol{e}^k$. Tu ale bez znalosti přesného řešení nespočítáme. Jak si ale ukážeme v příští lekci, pokud požadavek na ortogonalitu směrů $\boldsymbol{d}^k$ nahradíme za tzv. A-ortogonalitu, budeme skutečně schopni odvodit metodu, která (alespoň teoreticky) dokonverguje k řešení během n iterací.

1.2.2 Metoda sdružených gradientů

Definice Buď A symetrická pozitivně definitní matice. Nenulové vektory $\{\boldsymbol{p}_i\}_{i=0}^{n-1}$ nazveme sdružené (A-ortogonální), platí-li

$$\forall i, j \in \{0, 2, \ldots, n-1\} : i \neq j \Rightarrow \boldsymbol{p}_i^T \mathsf{A} \boldsymbol{p}_j = 0.$$

Dva navzájem A-ortogonální vektory někdy označujeme $\boldsymbol{a} \perp_{\mathsf{A}} \boldsymbol{b}$.



Obrázek 1.4: Ortogonální složky počáteční chyby (vlevo). Chceme-li vynulovat složku chyby ve směru $\boldsymbol{d}^0$, musí být $\boldsymbol{e}^1$ kolmý k $\boldsymbol{d}^0$ (vpravo).

Lemma Prvky množiny sdružených vektorů jsou lineárně nezávislé.

Důkaz Chceme ukázat, že platí

$$\alpha_0\boldsymbol{p}_0 + \alpha_1\boldsymbol{p}_1 + \ldots + \alpha_{n-1}\boldsymbol{p}_{n-1} = 0 \Rightarrow \forall i : \alpha_i = 0.$$

Přenásobme rovnici výrazem $(\mathsf{A}\boldsymbol{p}_k)^T, k \in \{0, 1, \ldots, n-1\}$ zleva:

$$\alpha_0\boldsymbol{p}_k^T\mathsf{A}\boldsymbol{p}_0 + \alpha_1\boldsymbol{p}_k^T\mathsf{A}\boldsymbol{p}_1 + \ldots + \alpha_{n-1}\boldsymbol{p}_k^T\mathsf{A}\boldsymbol{p}_{n-1} = 0.$$

Protože jsou vektory sdružené, všechny členy v této sumě až na jeden jsou nulové:

$$\alpha_k\boldsymbol{p}_k^T\mathsf{A}\boldsymbol{p}_k = 0.$$

Současně víme, že matice A je symetrická pozitivně definitní a součin $\boldsymbol{p}_k^T\mathsf{A}\boldsymbol{p}_k$ musí být kladný. Takže $\alpha_k = 0$. □

Vraťme se k poznámce na konci předchozí kapitoly a ortogonální směry nahraďme za A-ortogonální. Předpokládejme tedy, že známe n A-ortogonálních směrů $\{\boldsymbol{d}^0, \boldsymbol{d}^1 \ldots, \boldsymbol{d}^{n-1}\}$ a podobně jako na konci předchozí kapitoly se pokusme odvodit iterační metodu ve tvaru $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k\boldsymbol{d}^k$. Požadavek na ortogonalitu nové chyby k aktuálnímu směru nahraďme za požadavek na jejich vzájemnou A-ortogonalitu, tzn. $\boldsymbol{e}^{k+1} \perp_{\mathsf{A}} \boldsymbol{d}^k$. Ukažme si, že toto nastane v minimu naší kvadratické formy $f(\boldsymbol{x}) = 1/2\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ ve směru $\boldsymbol{d}^k$. Hledejme tedy nový bod $\boldsymbol{x}^{k+1}$ tak, aby se v něm derivace f ve směru $\boldsymbol{d}^k$ rovnala nule (nutná podmínka minima):

$$\frac{\mathrm{d}f(\boldsymbol{x}^{k+1})}{\mathrm{d}\boldsymbol{d}^k} = \mathrm{grad}f(\boldsymbol{x}^{k+1})^T\boldsymbol{d}^k = -(\underbrace{\boldsymbol{r}^{k+1}}_{=-\mathsf{A}\boldsymbol{e}^{k+1}})^T\boldsymbol{d}^k = (\boldsymbol{e}^{k+1})^T\mathsf{A}\boldsymbol{d}^k = 0.$$

Chyba $\boldsymbol{e}^{k+1}$ je tedy v bodu minima skutečně A-ortogonální ke směru $\boldsymbol{d}^k$. Nyní snadno můžeme získat předpis pro koeficient α_k (použijeme již několikrát zmíněný vztah $\boldsymbol{e}^{k+1} = \boldsymbol{e}^k + \alpha_k\boldsymbol{d}^k$):

$$(\boldsymbol{e}^{k+1})^T\mathsf{A}\boldsymbol{d}^k = (\boldsymbol{e}^k + \alpha_k\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^k = 0 \Rightarrow \alpha_k = -\frac{(\boldsymbol{e}^k)^T\mathsf{A}\boldsymbol{d}^k}{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}} = \frac{(\boldsymbol{r}^k)^T\boldsymbol{d}^k}{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}}$$

Věta Algoritmus spočte přesné řešení $\boldsymbol{x}$ v nejvýše n krocích.

Důkaz Vyjádříme si počáteční chybu $\boldsymbol{e}^0$ v bázi tvořené A-ortogonálními (tedy nezávislými) vektory $\{\boldsymbol{d}^i\}_{i=0}^{n-1}$:

$$\boldsymbol{e}^0 = \sum_{j=0}^{n-1} \delta_j \boldsymbol{d}^j.$$

Souřadnice δ_j najdeme tak, že rovnici přenásobíme zleva výrazem $(\boldsymbol{d}^k)^T\mathsf{A}, k \in \{0, 1, \ldots, n-1\}$ a využijeme toho, že většina členů výsledné sumy bude díky A-ortogonalitě nulová.

$$(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{e}^1 = \sum_{j=0}^{n-1} \delta_j (\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^j = \delta_k (\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^k.$$

Odtud

$$\delta_k = \frac{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{e}^1}{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^k} = \frac{(\boldsymbol{d}^k)^T\mathsf{A}(\boldsymbol{e}^1 + \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}^i)}{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^k} = \frac{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{e}^k}{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^k} = -\frac{(\boldsymbol{d}^k)^T\boldsymbol{r}^k}{(\boldsymbol{d}^k)^T\mathsf{A}\boldsymbol{d}^k}.$$

Zde si uvědomme, že přičtením sumy $\sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}^i$ jsme výraz nijak nezměnili (všechna $\boldsymbol{d}^i$ v sumě jsou A-ortogonální k $\boldsymbol{d}^k$ před závorkou, součin je tedy nulový). Dále rovnost $\boldsymbol{e}^0 + \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}^i = \boldsymbol{e}^k$ snadno vyplyne z již známého vztahu $\boldsymbol{e}^{k+1} = \boldsymbol{e}^k + \alpha_k \boldsymbol{d}^k$.

Všimněme si, že $\delta_k = -\alpha_k$. Vyjádřeme si tedy chybu v i-té iteraci jako

$$\boldsymbol{e}^i = \boldsymbol{e}^0 + \sum_{j=0}^{i-1} \alpha_j \boldsymbol{d}^j = \sum_{j=0}^{n-1} \delta_j \boldsymbol{d}^j + \sum_{j=0}^{i-1} \alpha_j \boldsymbol{d}^j = \sum_{j=0}^{n-1} \delta_j \boldsymbol{d}^j - \sum_{j=0}^{i-1} \delta_j \boldsymbol{d}^j = \sum_{j=i}^{n-1} \delta_j \boldsymbol{d}^j.$$

V každé iteraci se tedy odstraní jedna složka počáteční chyby $\boldsymbol{e}^0$. Po n iteracích je každá složka chyby v bázi $\{\boldsymbol{d}_i\}_{i=0}^{n-1}$ vynulovaná $\Rightarrow \boldsymbol{e}^n = \boldsymbol{o}$. □

Zbývá ukázat, jak najít jednotlivé A-ortogonální směry $\{\boldsymbol{d}^i\}_{i=0}^{n-1}$. Předpokládejme, že známe $\{\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^k\}$ a hledejme $\boldsymbol{d}^{k+1}$. Vyjdeme z vektoru rezidua $\boldsymbol{r}^{k+1}$ a pomocí Gramova-Schmidtova procesu (připomeňte si) jej A-ortogonalizujeme vůči předchozím směrům. Hledejme nový vektor ve tvaru

$$\boldsymbol{d}^{k+1} = \boldsymbol{r}^{k+1} - \sum_{j=0}^{k} \beta_{k,j} \boldsymbol{d}^j. \tag{1.14}$$

Určeme koeficienty $\beta_{k,j}$ tak, aby výsledný vektor byl A-ortogonální k předchozím vektorům $\{\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^k\}$. Přenásobme proto rovnost zleva výrazem $(\boldsymbol{d}^i)^T\mathsf{A}, i \in \{0, 1, \ldots, k\}$

$$(\boldsymbol{d}^i)^T\mathsf{A}\boldsymbol{d}^{k+1} = (\boldsymbol{d}^i)^T\mathsf{A}\boldsymbol{r}^{k+1} - \sum_{j=0}^{k} \beta_{k,j} (\boldsymbol{d}^i)^T\mathsf{A}\boldsymbol{d}^j$$

a využijme toho, že díky A-ortogonalitě je výraz na levé straně rovnosti stejně jako většina členů sumy napravo roven nule. Tedy

$$0 = (\boldsymbol{d}^i)^T \mathsf{A}\boldsymbol{r}^{k+1} - \beta_{k,i}(\boldsymbol{d}^i)^T \mathsf{A}\boldsymbol{d}^i \Rightarrow \beta_{k,i} = \frac{(\boldsymbol{d}^i)^T \mathsf{A}\boldsymbol{r}^{k+1}}{(\boldsymbol{d}^i)^T \mathsf{A}\boldsymbol{d}^i}.$$

Dosazením vypočtených koeficientů do (1.14) dostaneme

$$\boldsymbol{d}^{k+1} = \boldsymbol{r}^{k+1} - \sum_{j=0}^{k} \frac{(\boldsymbol{d}^j)^T \mathsf{A}\boldsymbol{r}^{k+1}}{(\boldsymbol{d}^j)^T \mathsf{A}\boldsymbol{d}^j}\boldsymbol{d}^j. \tag{1.15}$$

Všimněme si, že abychom pomocí tohoto vzorce vypočítali nový směr, museli bychom si pamatovat všechny předchozí směry, což by z paměťového hlediska bylo značně nevýhodné. Ukážeme si ale nyní, že všechny koeficienty $\beta_{k,j}$ kromě $\beta_{k,k}$ jsou nulové. Dokažme tedy následující tři lemmata.

Lemma Reziduum $\boldsymbol{r}^{k+1}$ je ortogonální k $\boldsymbol{d}^0, \boldsymbol{d}^1 \ldots, \boldsymbol{d}^k$.

Důkaz Přenásobme $\boldsymbol{r}^{k+1}$ vektory $\boldsymbol{d}^i, i \in \{0, 1, \ldots, k\}$.

$$(\boldsymbol{d}^i)^T \underbrace{\boldsymbol{r}^{k+1}}_{=-\mathsf{A}\boldsymbol{e}^{k+1}} = -(\boldsymbol{d}^i)^T \mathsf{A}\boldsymbol{e}^{k+1} = -(\boldsymbol{d}^i)^T \mathsf{A} \sum_{j=k+1}^{n-1} \delta_j \boldsymbol{d}^j = 0,$$

kde poslední rovnost vyplývá z A-ortogonality všech vektorů $\boldsymbol{d}^j$ v sumě k vektoru $\boldsymbol{d}^i$. □

Lemma Reziduum $\boldsymbol{r}^{k+1}$ je ortogonální k $\boldsymbol{r}^0, \boldsymbol{r}^1, \ldots, \boldsymbol{r}^k$.

Důkaz Využijme předpis (1.14) a vyjádřeme $\boldsymbol{r}^i$:

$$\boldsymbol{r}^i = \boldsymbol{d}^i + \sum_{j=0}^{i-1} \beta_{i,j}\boldsymbol{d}^j.$$

Pak díky předchozímu lemmatu dostaneme pro všechna $i \leq k$:

$$(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^i = (\boldsymbol{r}^{k+1})^T (\boldsymbol{d}^i + \sum_{j=0}^{i-1} \beta_{i,j}\boldsymbol{d}^j) = (\boldsymbol{r}^{k+1})^T \boldsymbol{d}^i + \sum_{j=0}^{i-1} \beta_{i,j} (\boldsymbol{r}^{k+1})^T \boldsymbol{d}^j = 0.$$

□

Lemma Nechť je dána množina reziduí $\{\boldsymbol{r}^i\}_{i=0}^{n-1}$ a sdružených směrů $\{\boldsymbol{d}^i\}_{i=0}^{n-1}$. Potom platí:

$$j < k : (\boldsymbol{r}^{k+1})^T \mathsf{A}\boldsymbol{d}^j = 0$$
$$j = k : (\boldsymbol{r}^{k+1})^T \mathsf{A}\boldsymbol{d}^j \neq 0$$

Důkaz Pro reziduum platí

$$\boldsymbol{r}^{j+1} = \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{j+1} = \boldsymbol{b} - \mathsf{A}(\boldsymbol{x}^j + \alpha_j \boldsymbol{d}^j) = \boldsymbol{r}^j - \alpha_j \mathsf{A}\boldsymbol{d}^j$$

Odtud

$$\mathsf{A}\boldsymbol{d}^j = -\frac{1}{\alpha_j}(\boldsymbol{r}^{j+1} - \boldsymbol{r}^j)$$

a

$$(\boldsymbol{r}^{k+1})^T \mathsf{A}\boldsymbol{d}^j = (\boldsymbol{r}^{k+1})^T (\frac{1}{\alpha_j}(\boldsymbol{r}^j - \boldsymbol{r}^{j+1})) = \frac{1}{\alpha_j}(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^j - \frac{1}{\alpha_j}(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^{j+1}$$

Pro $j < k$ je tento výraz díky ortogonalitě reziduí nulový. Pro $j = k$ dostaneme

$$(\boldsymbol{r}^{k+1})^T \mathsf{A}\boldsymbol{d}^k = \frac{1}{\alpha_k}(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^k - \frac{1}{\alpha_k}(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^{k+1} = -\frac{1}{\alpha_k}(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^{k+1}. \quad (1.16)$$

□

Vraťme se nyní zpět k předpisu pro výpočet koeficientů $\beta_{k,i}$ v Gramově-Schmidtově procesu. Díky předchozímu lemmatu jsme zjistili, že pro $j = 0, \ldots, k-1$ jsou čitatelé v sumě v (1.15) nulové. Předpis se tedy zjednoduší na

$$\boldsymbol{d}^{k+1} = \boldsymbol{r}^{k+1} - \frac{(\boldsymbol{d}^k)^T \mathsf{A}\boldsymbol{r}^{k+1}}{(\boldsymbol{d}^k)^T \mathsf{A}\boldsymbol{d}^k}\boldsymbol{d}^k.$$

K výpočtu nového směru nám stačí znát nové reziduum a předchozí směr. Protože všechny ostatní koeficienty v dané sumě jsou nulové, budeme pro jednoduchost psát místo $\beta_{k,k}$ jen β_k.

Využijeme-li rovnosti $\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \alpha_k \mathsf{A}\boldsymbol{d}^k$, tedy $\mathsf{A}\boldsymbol{d}^k = \frac{1}{\alpha_k}(\boldsymbol{r}^k - \boldsymbol{r}^{k+1})$, můžeme přepsat čitatele na

$$(\boldsymbol{r}^{k+1})^T \mathsf{A}\boldsymbol{d}^k = \frac{1}{\alpha_k}(\boldsymbol{r}^{k+1})^T (\boldsymbol{r}^k - \boldsymbol{r}^{k+1}) = -\frac{1}{\alpha_k}(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^{k+1}.$$

Podobně lze upravit jmenovatele

$$(\boldsymbol{d}^k)^T \mathsf{A}\boldsymbol{d}^k = (\boldsymbol{r}^k - \beta_{k-1}\boldsymbol{d}^{k-1})^T \mathsf{A}\boldsymbol{d}^k = \frac{1}{\alpha_k}(\boldsymbol{r}^k)^T (\boldsymbol{r}^k - \boldsymbol{r}^{k+1}) = \frac{1}{\alpha_k}(\boldsymbol{r}^k)^T \boldsymbol{r}^k.$$

Po dosazení do předpisu pro β_k dostaneme

$$\beta_k = -\frac{(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^{k+1}}{(\boldsymbol{r}^k)^T \boldsymbol{r}^k}$$

V podobném duchu můžeme upravit i čitatele v předpisu α_k

$$(\boldsymbol{r}^k)^T \boldsymbol{d}^k = (\boldsymbol{r}^k)^T \boldsymbol{d}^k = \boldsymbol{r}^k - \beta_{k-1}\boldsymbol{d}^{k-1} = (\boldsymbol{r}^k)^T \boldsymbol{r}^k - \beta_{k,k-1} \underbrace{(\boldsymbol{r}^k)^T \boldsymbol{d}^{k-1}}_{=0},$$

čímž získáme

$$\alpha_k = \frac{(\boldsymbol{d}^k)^T \boldsymbol{r}^k}{(\boldsymbol{d}^k)^T \mathsf{A}\boldsymbol{d}^k} = \frac{(\boldsymbol{r}^k)^T \boldsymbol{r}^k}{(\boldsymbol{d}^k)^T \mathsf{A}\boldsymbol{d}^k}.$$

Pokud použijme tyto předpisy pro α_k, β_k, ušetříme násobení matice-vektor a skalární součin.

Zapišme nyní celý algoritmus:

function CONJUATE_GRADIENT(A, $\boldsymbol{b}$, $\boldsymbol{x}^0$)
  $\boldsymbol{d}^0 = \boldsymbol{r}^0 = b - \mathsf{A}\boldsymbol{x}^0$
  $k = 0$
  while $\|\boldsymbol{r}^k\|/\|\boldsymbol{r}^0\| > \varepsilon$ **do**
    $\alpha_k = \frac{(\boldsymbol{r}^k)^T \boldsymbol{r}^k}{(\boldsymbol{d}^k)^T \mathsf{A}\boldsymbol{d}^k}$
    $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k$
    $\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - \alpha_k \mathsf{A}\boldsymbol{r}^k$
    $\beta_k = \frac{(\boldsymbol{r}^{k+1})^T \boldsymbol{r}^{k+1}}{(\boldsymbol{r}^k)^T \boldsymbol{r}^k}$
    $\boldsymbol{d}^{k+1} = \boldsymbol{r}^{k+1} + \beta_k \boldsymbol{d}^k$
    $k = k + 1$
  end while
end function

Doplňme ještě informace o konvergenci. Chybu můžeme odhadnout pomocí vztahů

$$\|\boldsymbol{e}^{k+1}\|_{\mathsf{A}} \leq \frac{\kappa(\mathsf{A}) - 1}{\kappa(\mathsf{A}) + 1} \|\boldsymbol{e}^k\|_{\mathsf{A}}$$

a

$$\|\boldsymbol{e}^k\|_{\mathsf{A}} \leq 2 \left(\frac{\sqrt{\kappa(\mathsf{A})} - 1}{\sqrt{\kappa(\mathsf{A})} + 1} \right)^k \|\boldsymbol{e}^0\|_{\mathsf{A}}.$$

Můžeme také odhadnout maximální počet iterací nutných k dosažení relativní přesnosti ε (tedy $\|\boldsymbol{e}^k\| \leq \varepsilon \|\boldsymbol{e}^0\|$):

$$k \leq \lceil \frac{1}{2} \sqrt{\kappa(\mathsf{A})} \ln \frac{2}{\varepsilon} \rceil.$$

Porovnejme to s metodou největšího spádu, pro kterou platí

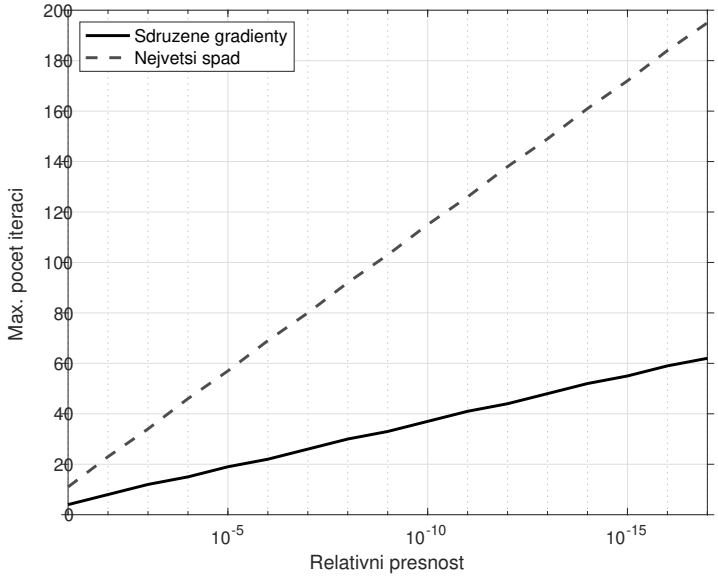
$$k \leq \lceil \frac{1}{2} \kappa(\mathsf{A}) \ln \frac{1}{\varepsilon} \rceil.$$

Rozdíl mezi metodami je dobře patrný na Obrázku 1.5.

Na závěr poznamenejme, že metoda sdružených gradientů patří mezi tzv. Krylovovské metody. Tyto metody generují posloupnost Krylovových podprostorů ve tvaru $\mathcal{K}^k(\mathsf{A}, \boldsymbol{b}) = \text{span}\left\{\boldsymbol{b}, \mathsf{A}\boldsymbol{b}, \mathsf{A}^2\boldsymbol{b}, \ldots, \mathsf{A}^k\boldsymbol{b}\right\}$ a v každé iteraci minimalizují chybu v energetické normě na daném podprostoru, tedy

$$\boldsymbol{x}^{k+1} = \underset{\overline{\boldsymbol{x}} \in \boldsymbol{x}^0 + \mathcal{K}^k(\mathsf{A}, \boldsymbol{r}^0)}{\arg\min} \|\boldsymbol{x} - \overline{\boldsymbol{x}}\|_{\mathsf{A}},$$

kde $\boldsymbol{x}$ je přesné řešení.



Obrázek 1.5: Porovnání maximálního počtu iterací nutných k dosažení dané relativní přesnosti ($\kappa(\mathsf{A}) = 10$).

1.3 Předpodmínění

V předchozích kapitolách jsme viděli, že rychlost konvergence iteračních metod k řešení závisí na čísle podmíněnosti $\kappa(A)$ matice soustavy. Předpodmínění je nenáročná ekvivalentní úprava soustavy, jejímž cílem je snížení čísla podmíněnosti, tedy snížení počtu potřebných iterací.

Předpokládejme, že známe matici M takovou, že platí $\kappa(\mathsf{M}^{-1}\mathsf{A}) \ll \kappa(\mathsf{A})$. Pak místo původní soustavy $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ můžeme řešit ekvivalentní systém

$$\mathsf{M}^{-1}\mathsf{A}\boldsymbol{x} = \mathsf{M}^{-1}\boldsymbol{b}. \tag{1.17}$$

Vzhledem k menšímu číslu podmíněnosti matice $\mathsf{M}^{-1}\mathsf{A}$ by měl oproti původní soustavě klesnout počet iterací nutných k jejímu vyřešení. Matici M nazýváme *levým předpodmiňovačem*.

Je-li matice soustavy A symetrická pozitivně definitní, je výhodné původní soustavu řešit metodou největšího spádu nebo metodou sdružených gradientů. Po předpodmiňovači M pak budeme chtít, aby byl také symetrický pozitivně definitní. Problémem však je, že součin dvou symetrických pozitivně definitních matic obecně nemusí být symetrická pozitivně definitní matice. Systém (1.17) tedy můžeme vyřešit pomocí iteračního řešiče, který nevyžaduje tyto vlastnosti matice, ale na rozdíl od původní soustavy $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ nemůžeme použít metodu největšího spádu ani metodu sdružených gradientů.

Pokusme se tedy odvodit takový tvar předpodmínění, abychom zachovali symetrii a pozitivní definitnost matice systému. Využijeme toho, že každou symetrickou pozitivně definitní matici můžeme rozložit (např. pomocí Choleského rozkladu) na součin $\mathsf{M} = \mathsf{L}\mathsf{L}^T$, kde L je dolní trojúhelníková matice. Dále využijme toho, že matice $\mathsf{M}^{-1}\mathsf{A}$ a $\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}$ mají stejná vlastní čísla (a tedy stejné číslo podmíněnosti). Je-li totiž $\boldsymbol{v}$ vlastní vektor matice $\mathsf{M}^{-1}\mathsf{A}$ s vlastním číslem λ, pak $\mathsf{L}^T\boldsymbol{v}$ je vlastní vektor $\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}$ se stejným vlastním číslem λ:

$$\begin{aligned}(\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T})(\mathsf{L}^T\boldsymbol{v}) &= \mathsf{L}^{-1}\mathsf{A}\underbrace{\mathsf{L}^{-T}\mathsf{L}^T}_{=\mathsf{I}}\boldsymbol{v} = \mathsf{L}^{-1}\mathsf{A}\boldsymbol{v} = (\underbrace{\mathsf{L}^T\mathsf{L}^{-T}}_{=\mathsf{I}})\mathsf{L}^{-1}\mathsf{A}\boldsymbol{v} \\ &= \mathsf{L}^T\underbrace{\mathsf{L}^{-T}\mathsf{L}^{-1}}_{=\mathsf{M}^{-1}}\mathsf{A}\boldsymbol{v} = \mathsf{L}^T\underbrace{\mathsf{M}^{-1}\mathsf{A}\boldsymbol{v}}_{=\lambda\boldsymbol{v}} = \lambda\mathsf{L}^T\boldsymbol{v}\end{aligned}$$

Přenásobme nyní původní soustavu $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ zleva maticí L^{-1}:

$$\mathsf{L}^{-1}\mathsf{A}\boldsymbol{x} = \mathsf{L}^{-1}\boldsymbol{b}$$

Abychom zachovali symetrii a pozitivní definitnost, musíme matici A přenásobit L^{-T} zprava. "Vložme" tedy mezi A a $\boldsymbol{x}$ jednotkovou matici ve tvaru $\mathsf{I} = \mathsf{L}^{-T}\mathsf{L}^T$, tím soustavu nijak nezměníme:

$$\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}\mathsf{L}^T\boldsymbol{x} = \mathsf{L}^{-1}\boldsymbol{b}.$$

Místo původní soustavy $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ a místo soustavy (1.17) řešme ekvivalentní systém

$$(\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T})(\mathsf{L}^T\boldsymbol{x}) = \mathsf{L}^{-1}\boldsymbol{b}.$$

Zaveďme značení $\hat{\mathsf{A}} = \mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}, \hat{\boldsymbol{x}} = \mathsf{L}^T\boldsymbol{x}, \hat{\boldsymbol{b}} = \mathsf{L}^{-1}\boldsymbol{b}$. Soustavu pak můžeme zapsat jako

$$\hat{\mathsf{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}. \tag{1.18}$$

V tomto případě je matice soustavy $\hat{\mathsf{A}} = \mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}$ symetrická a pozitivně definitní, k řešení (1.18) tedy můžeme použít metodu největšího spádu či metodu sdružených gradientů. Po nalezení $\hat{\boldsymbol{x}}$ získáme řešení původní soustavy jako $\boldsymbol{x} = \mathsf{L}^{-T}\hat{\boldsymbol{x}}$.[2]

Poznamenejme, že nevýhodou tohoto přístupu je nutnost znalosti rozkladu $\mathsf{M} = \mathsf{L}\mathsf{L}^T$. Vhodnou manipulací s předpisy jednotlivých iteračních metod se však této potřeby zbavíme. Demonstrujme to pro jednoduchost na Richardsonově metodě, která má při aplikaci na (1.18) předpis

$$\hat{\boldsymbol{x}}^{k+1} = \hat{\boldsymbol{x}}^k + \omega(\hat{\boldsymbol{b}} - \hat{\mathsf{A}}\hat{\boldsymbol{x}}^k),$$

[2]Inverzní matice L^{-1} a L^{-T} samozřejmě nemusíme explicitně sestavovat. Např. násobení $\boldsymbol{w} = \mathsf{L}^{-1}\boldsymbol{u}$ je ekvivalentní s řešením soustavy $\mathsf{L}\boldsymbol{w} = \boldsymbol{u}$. Matice L je dolní trojúhelníková, řešení tedy snadno získáme pomocí dopředné substituce. Obdobně postupujeme i v případě násobení maticí L^{-T}. Není také třeba a není výhodné, abychom explicitně sestavovali matici $\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}$ jakou maticový součin. Součin $\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}\boldsymbol{u}$ vypočítáme jako posloupnost tří po sobě jdoucích součinů matice-vektor.

tedy

$$\mathsf{L}^T\boldsymbol{x}^{k+1} = \mathsf{L}^T\boldsymbol{x}^k + \omega(\mathsf{L}^{-1}\boldsymbol{b} - \mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}\mathsf{L}^T\boldsymbol{x}^k).$$

Přenásobme tento předpis zleva maticí L^{-T}:

$$\mathsf{L}^{-T}\mathsf{L}^T\boldsymbol{x}^{k+1} = \mathsf{L}^{-T}\mathsf{L}^T\boldsymbol{x}^k + \omega(\mathsf{L}^{-T}\mathsf{L}^{-1}\boldsymbol{b} - \mathsf{L}^{-T}\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-T}\mathsf{L}^T\boldsymbol{x}^k).$$

Odtud snadno získáme předpis předpodmíněné Richardsonovy metody, který již nevyžaduje znalost rozkladu M:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \omega\mathsf{M}^{-1}(\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^k).$$

Podobným způsobem bychom mohli upravit předpisy metody největšího spádu či metody sdružených gradientů. Všimněme si, že při použití tohoto předpisu potřebujeme aplikovat M^{-1} (tedy řešit soustavu s maticí M). Předpodmiňovač tedy musíme volit tak, abychom toto řešení byli schopni najít rychle a efektivně.

Jak tedy zvolit M? Požadavky na předpodmiňovač se dají shrnout takto: inverze M by měla aproximovat inverzi A ($\mathsf{M}^{-1} \approx \mathsf{A}^{-1}$), řešení soustavy s M by mělo být snadné a efektivní a pokud je matice A symetrická a pozitivně definitní, měl by i předpodmiňovač být symetrický pozitivně definitní. Dva extrémní případy jsou:

- $\mathsf{M} = \mathsf{A}$ – v takovém případě inverze M aproximuje inverzi A přesně, ale aplikace samotného předpodmínění je stejně náročná jako řešení původní úlohy;
- $\mathsf{M} = \mathsf{I}$ – v tomto případě je sice aplikace předpodmiňovače triviální, nedosáhneme ale žádného zlepšení čísla podmíněnosti (získáme původní nepředpodmíněný algoritmus).

Mezi těmito nepoužitelnými extrémy existuje celé spektrum sofistikovaných metod předpodmínění, pro jednoduchost však zmiňme dva dobře známé přístupy:

- diagonální předpodmiňovač – volíme $\mathsf{M} = \operatorname{diag}\mathsf{A}$. V tomto případě je výpočet inverze M velmi snadný. Tento předpodmiňovač je efektivní např. pro soustavy s diagonálně dominantní maticí.
- neúplný LU/Choleského rozklad – je vhodný pro soustavy s řídkou maticí. Problémem klasického LU/Choleského rozkladu je, že přestože je matice A řídká, mohou být její faktory L, U plné nebo více zaplněné. Princip neúplného LU/Choleského rozkladu je jednoduchý - použijeme stejný algoritmus jako u klasického LU/Choleského rozkladu, ale nenulový prvek do matice L nebo U uložíme pouze tehdy, existuje-li nenulový prvek na stejné pozici v rozkládané matici A. Dostaneme neúplné faktory $\tilde{\mathsf{L}}, \tilde{\mathsf{U}}$ a příslušné předpodmíňovače definujeme jako $\mathsf{M} = \tilde{\mathsf{L}}\tilde{\mathsf{U}}$ nebo $\mathsf{M} = \tilde{\mathsf{L}}\tilde{\mathsf{L}}^T$. Řešit soustavu s M je pak snadné, protože známe její rozklad na součin trojúhelníkových matic a můžeme použít dopřednou a zpětnou substituci.

References

[1] Trefethen, L. N, Bau, D. Numerical Linear Algebra. SIAM. 1997.

[2] Schewchuk, J. R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. 1994. Dostupné z `https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`

[3] Lukáš, D. Zápisky z přednášek. Dostupné z `https://homel.vsb.cz/~luk76`