Numerické metody

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

Ústav matematiky
Fakulta strojního inženýrství
Vysoké učení technické v Brně

6. února 2006

Obsah

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

Budeme se tedy zabývat řešením soustavy lineárních rovnic

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

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

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.

Obsah

Gaussova eliminační metoda

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

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

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

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

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

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

Přímý chod GEM popisuje následující

algoritmus GEMz (základní, bez výběru hlavního prvku):

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

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

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

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

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

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

LU rozklad. Po ukončení přímého chodu je horní trojúhelníková matice **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 **c** v (2.4) je transformovanou pravou stranou $\mathbf{b}^{(n-1)}$, tj.

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

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

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

definované předpisem

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

Dá se ukázat, že platí

$$\mathbf{A} = \mathbf{LU}\,. \tag{2.8}$$

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

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

$$\mathbf{Ly} = \mathbf{b}\,, \qquad \mathbf{Ux} = \mathbf{y}\,. \tag{2.9}$$

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

Soustavu $\mathbf{Ly} = \mathbf{b}$ vyřešíme snadno, z první rovnice vypočítáme y_1, ze druhé rovnice y_2 atd. až nakonec z poslední rovnice vypočítáme y_n. Soustavu $\mathbf{Ux} = \mathbf{y}$ 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 $\mathbf{Ax} = \mathbf{b}$ 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 symetrická matice $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{n}$ je pozitivně definitní, jestliže

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) téměř 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. [Fiedler], [Meurant]).

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).

Výběr hlavního prvku

Začneme příkladem.

Příklad 2.1. Máme vyřešit soustavu rovnic

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

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

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

V prvním kroku eliminujeme poddiagonální prvky v prvním sloupci a dostaneme

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

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

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

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

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

Zpětný chod začne výpočtem

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

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

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

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

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

Nakonec vypočteme x_1 z první rovnice

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

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

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

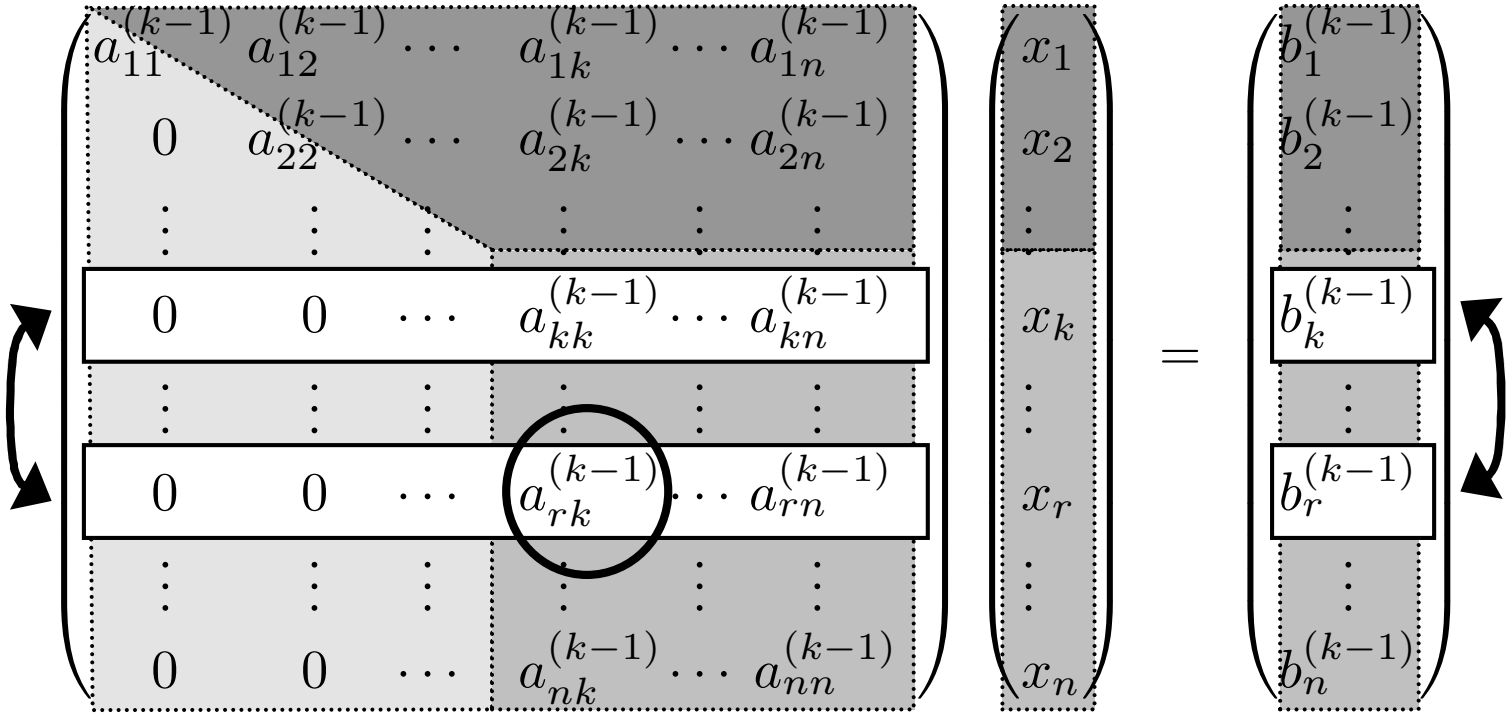
Kde vznikl problém? Nedošlo k žádnému hromadění chyb způsobenému prováděním tisíců operací. Matice soustavy není blízká matici singulární. Potíž je jinde, působí ji malý pivot ve druhém kroku eliminace. Tím vznikne multiplikátor 2400 a v důsledku toho má poslední rovnice koeficienty zhruba 1000 krát větší než

koeficienty původní rovnice. Zaokrouhlovací chyby, které jsou malé vzhledem k těmto velkým koeficientům, jsou nepřijatelné pro koeficienty původní matice a také pro samotné řešení.
Snadno se prověří, že když druhou a třetí rovnici prohodíme, nevzniknou žádné velké multiplikátory a výsledek je zcela přesný. Ukazuje se, že to platí obecně: jestliže jsou absolutní hodnoty multiplikátorů menší nebo nejvýše rovny 1, pak je numericky spočtené řešení vyhovující. □

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

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

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

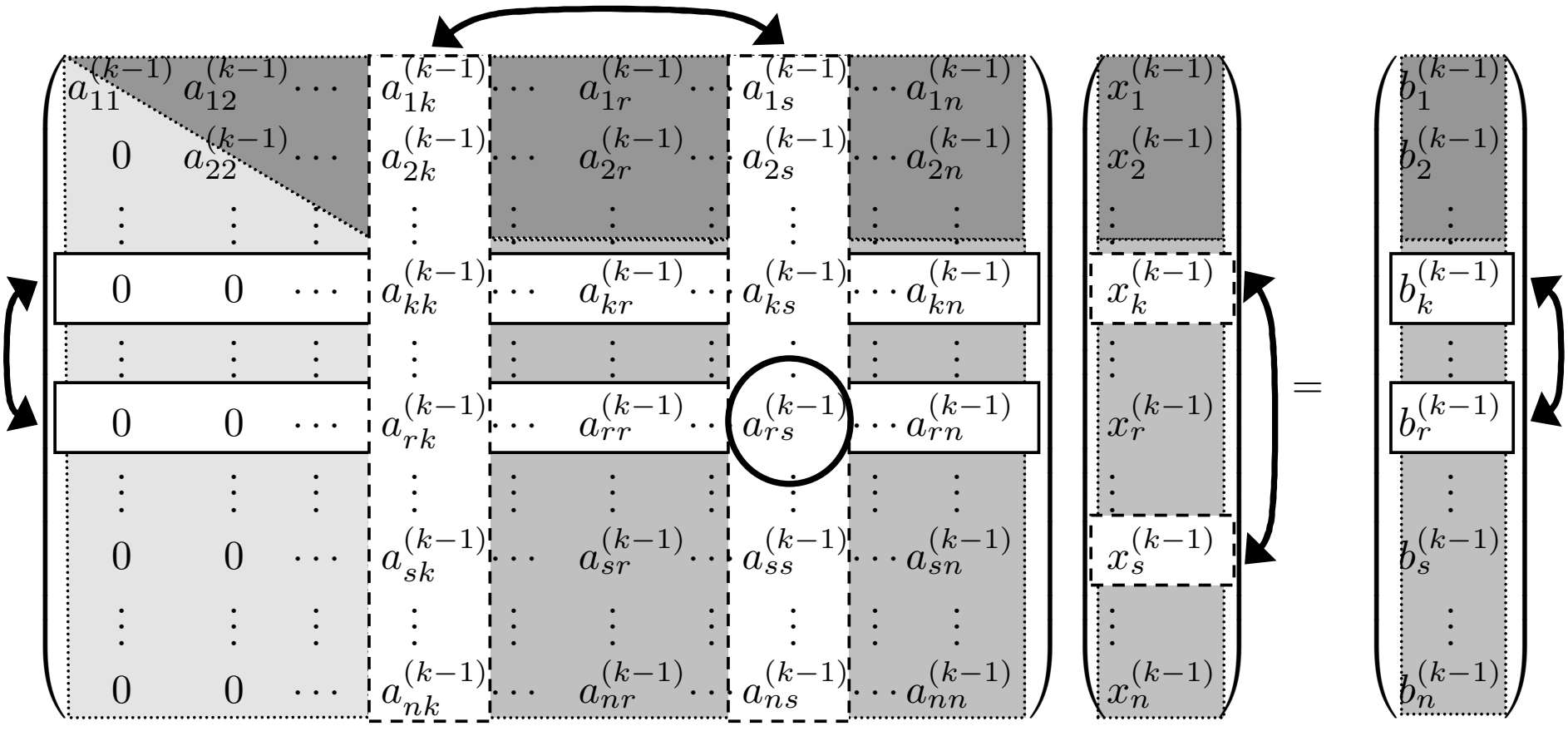


Obr. 2.1: GEM s částečným výběrem hlavního prvku (v kroužku)

Úplný výběr hlavního prvku je postup, který může absolutní hodnoty multiplikátorů zmenšit ještě výrazněji. Dociluje se toho tím, že v k-tém kroku eliminace se jako pivot vybírá prvek s největší absolutní hodnotou v dosud neeliminované části matice $\mathbf{A}^{(k-1)}$, tj. v řádcích $i \geq k$ a sloupcích $j \geq k$. Nechť tedy r je řádkový a s sloupcový index vybraný tak, že

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

Pak prohodíme k-tou a r-tou rovnici a k-tou a s-tou neznámou. Ze soustavy rovnic $\mathbf{A}^{(k-1)}\mathbf{x}^{(k-1)} = \mathbf{b}^{(k-1)}$ 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 $\mathbf{x}$.



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

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 zaokroulovací chyby zůstaly dostatečně malé a neznehodnotily výsledné řešení. Dalším důvodem, proč částečnému výběru hlavních prvků dáváme přednost, je to, že jeho realizace vyžaduje výrazně menší počet operací než výběr úplný.

LU rozklad s částečným výběrem hlavního prvku je standardní rutina dostupná v každé knihovně programů pro numerické řešení úloh lineární algebry. Vstupním parametrem je matice **A**. Výstupní parametry jsou tři: dolní trojúhelníková matice **L** (s jedničkami na hlavní diagonále), horní trojúhelníková matice **U** a tzv. permutační matice **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 **A** s částečným výběrem hlavních prvků.

Algoritmus LUp (s částečným výběrem hlavního prvku)

1) Položíme $\mathbf{A}^{(0)} = \mathbf{A}$, $\mathbf{P}^{(0)} = \mathbf{I}$.
2) Postupně pro $k = 1, 2, \ldots, n-1$ provádíme
   - 2a) Určíme index r pivotního řádku, viz (2.14).
   - 2b) Prohodíme řádky k a r v matici $\mathbf{A}^{(k-1)}$ a takto získanou matici označíme jako $\mathbf{A}^{(k)}$. Prohodíme rovněž řádky k a r v matici $\mathbf{P}^{(k-1)}$ a takto získanou matici označíme jako $\mathbf{P}^{(k)}$.
   - 2c) Upravíme matici $\mathbf{A}^{(k)}$ tak, že eliminujeme poddiagonální prvky v k-tém sloupci, tj. postupně pro $i = k+1, k+2, \ldots, n$ počítáme

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

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

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

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

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

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

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

Uchovávat historii prohazování řádků v matici $\mathbf{P}$ je zřejmé plýtvání paměťovým místem, vždyť z celkového počtu n^2 prvků matice $\mathbf{P}$ je jich pouze n nenulových. Proto místo s maticí $\mathbf{P}$ stačí pracovat jen s vektorem řádkových permutací $\mathbf{p}$. Na začátku položíme $\mathbf{p}^{(0)} = (1, 2, \ldots, n)^T$ a ve zbytku algoritmu pak prohazujeme prvky vektoru $\mathbf{p}^{(k-1)}$. Matice $\mathbf{P}$ byla zapotřebí jen pro sestavení vektoru $\mathbf{z}$. To ale dokážeme s pomocí vektoru $\mathbf{p}$ také: do i-té složky vektoru $\mathbf{z}$ vložíme p_i-tou složku vektoru $\mathbf{b}$, postupně pro $i = 1, 2, \ldots, n$.

Příklad 2.2. Provedeme LU rozklad matice

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

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

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

První řádek se už měnit nebude. Ve druhém kroku najdeme pivota ve druhém sloupci. Je to číslo 3,5 ve čtvrtém řádku. Proto prohodíme druhý a čtvrtý řádek

jak v matici $\mathbf{A}^{(1)}$ tak ve vektoru $\mathbf{p}^{(1)}$. Pak eliminujeme prvky v pozicích (3,2) a (4,2) a na jejich místa zapíšeme použité multiplikátory.
Výsledkem je

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

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

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

Dostali jsme tedy

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

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

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

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

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

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

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

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

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

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

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

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

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

Řešení soustavy rovnic s více pravými stranami nepředstavuje žádný problém, místo s jednou pravou stranou pracujeme současně s m pravými stranami. Vzorce (2.9) a (2.17) zůstávají v platnosti, jediný rozdíl je v tom, že teď $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m)$ je matice pravých stran, takže také $\mathbf{y}$, $\mathbf{x}$ a případně $\mathbf{z}$ jsou matice téhož typu jako $\mathbf{b}$. Zejména tedy i-tý sloupec matice $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ je řešení příslušné i-té pravé straně. Případ úplného výběru hlavních prvků zde rozebírat nebudeme.
Pokud jde o počet operací, tak pro každou pravou stranu je třeba započítat přibližně n^2 operací násobicích a stejný počet operací sčítacích, takže celkem jde o mn^2 operací. Je-li počet m pravých stran malý ve srovnání s počtem n rovnic, jsou náklady na provedení LU rozkladu, tj. přibližně $\frac{1}{3}n^3$ operací, výrazně převažující.

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

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 potřebuje LU rozklad bez výběru hlavních prvků jen přibližně $ns(s + 1)$ operací a zpětný chod (pro jednu pravou stranu) jen přibližně $n(2s + 1)$ operací (násobicích a stejně tak sčítacích).

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.

Vliv zaokrouhlovacích chyb

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

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

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

Příbližné 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 první rovnice odečítáme m_{21}-násobek druhé 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.

Podmíněnost

Abychom mohli určit podmíněnost úlohy „najít řešení **x** soustavy linearních rovnic $\mathbf{Ax} = \mathbf{b}$“, potřebujeme posoudit, jak moc se změní řešení **x**, když trochu změníme data, tj. matici soustavy **A** a vektor pravé strany **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.

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

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

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

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

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

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

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

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

Uvažujme systém rovnic

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

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

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

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

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

takže pro $m \neq 0$,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Jestliže vektorová a maticová norma splňují vztah (2.24), říkáme, že jsou souhlasné.
Existují i jiné než přirozené maticové normy. Jednou z nich je Frobeniova norma

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

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

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

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

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

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

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

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

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

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

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

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

Podle (2.21) můžeme odhadnout číslo podmíněnosti

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

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

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

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

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

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

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 soustav lineárních rovnic. Jestliže program zjistí, že číslo podmíněnosti je příliš velké, vydá varování nebo dokonce výpočet přeruší.

Shrnutí. Soustava lineárních rovnic je dobře (špatně) podmíněná, právě když je matice soustavy dobře (špatně) podmíněná. Podmíněnost matice soustavy $\mathbf{A}$ měříme pomocí čísla podmíněnosti $\kappa(\mathbf{A}) \geq 1$. Je-li číslo $\kappa(\mathbf{A})$ malé, řekněme menší než 100, je matice $\mathbf{A}$ dobře podmíněná. V opačném případě, tj. když $\kappa(\mathbf{A}) \geq 100$, je matice $\mathbf{A}$ špatně podmíněná. Špatně podmíněnou soustavu rovnic lze obvykle jen velmi obtížně řešit. Pomoci může výpočet s vícemístnou mantisou (je vhodné použít dvojnásobnou nebo ještě větší přesnost). Existují však výjimky: je-li například $\mathbf{A}$ diagonální matice, ve které $a_{ii} = 10^i$, pak je $\kappa(\mathbf{A}) = 10^{n-1}$, což je pro velké n velké číslo, a přesto řešení $x_i = 10^{-i} b_i$ získáme bez problémů pro libovolně velký počet rovnic.
Předpokládejme, že matice soustavy je dobře podmíněná. Pak je GEM s částečným (nebo úplným) výběrem hlavního prvku dobře podmíněný algoritmus: protože velikost multiplikátorů nepřesahuje jedničku, vznikající zaokrouhlovací chyby se dalším výpočtem „nezesilují“. Když naopak výběr hlavních prvků neprovádíme, můžeme dostat multiplikátory, jejichž absolutní hodnota je větší než jedna, což má za následek zvětšování dříve vzniklých zaokrouhlovacích chyb. GEM bez výběru hlavních prvků je tedy obecně špatně podmíněný algoritmus. Výjimku z tohoto pravidla přestavuje řešení soustav se speciální maticí soustavy, např. když je matice soustavy ostře diagonálně dominantní nebo pozitivně definitní, pak je i GEM bez výběru hlavních prvků dobře podmíněný algoritmus.

Obsah

Mnoho praktických problémů vyžaduje řešení rozsáhlých soustav lineárních rovnic $\mathbf{Ax} = \mathbf{b}$, v nichž matice $\mathbf{A}$ je naštěstí řídká, tj. má relativně málo nenulových prvků. Standardní eliminační metody studované v předchozí kapitole 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 $\mathbf{x}_0 \longrightarrow \mathbf{x}_1 \longrightarrow \mathbf{x}_2 \ldots$, která konverguje k hledanému řešení $\mathbf{x}$. Společným rysem všech iteračních metod je fakt, že každý jednotlivý iterační krok $\mathbf{x}_k \longrightarrow \mathbf{x}_{k+1}$ vyžaduje objem výpočtů srovnatelný s násobením matice $\mathbf{A}$ vektorem, což je pro řídké matice objem nevelký (pokud je v každém řádku matice $\mathbf{A}$ řádu n nejvýše m nenulových prvků, jde o nm operací násobení a sčítání). Přijatelný objem výpočtů lze proto dosáhnout i pro poměrně velký počet iterací.

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

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

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

přičemž počáteční aproximace $\mathbf{x}_0$ je daná.
Řekneme, že iterační metoda konverguje, a píšeme $\mathbf{x}_k \longrightarrow \mathbf{x}$, když číselná posloupnost $\|\mathbf{x}_k - \mathbf{x}\| \to 0$. Označme $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}$ chybu v k-té iteraci. Protože $\mathbf{M}\mathbf{x} = \mathbf{N}\mathbf{x} + \mathbf{b}$, dostaneme

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Je dokázáno, že

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

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

1. Konvergence Gaussovy-Seidelovy metody je pro mnohé matice $\mathbf{A}$ rychlejší než konvergence Jacobiovy metody. Tak je tomu třeba v případě, když $\mathbf{A}$ je ryze diagonálně dominantní.
2. Existují matice, pro které Gaussova-Seidelova metoda konverguje a Jacobiova metoda nekonverguje a naopak, pro které konverguje Jacobiova metoda a Gaussova-Seidelova metoda nekonverguje.
3. Jacobiova metoda umožňuje paralelní výpočet (všechny složky $x_i^{(k+1)}$ mohou být počítány současně, každá na jiném procesoru), zatímco Gaussova-Seidelova metoda je ze své podstaty sekvenční ($x_i^{(k+1)}$ lze vypočítat až po té, co byly spočteny všechny složky $x_j^{(k+1)}$ pro $j < i$). Pro speciální typy matic $\mathbf{A}$ jsou však vypracovány postupy umožňující paralelizovat i Gaussovu-Seidelovu metodu.

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

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

kde $\omega > 0$ je tzv. relaxační parametr. Volíme ho tak, abychom vylepšili konvergenci základní metody. Pro $\omega = 1$ dostáváme původní metodu. Zvolíme-li $\omega < 1$, hovoříme o dolní relaxaci, v případě $\omega > 1$ jde o horní relaxaci. Efektivní volba relaxačního parametru ω závisí na zvolené základní metodě a na matici soustavy $\mathbf{A}$.
Praktické zkušenosti potvrzují, že dolní relaxace může zajistit konvergenci v případě, když základní metoda nekonverguje. Vhodnou volbou relaxačního parametru lze rychlost konvergence původní metody podstatně zrychlit. Pro zvolenou metodu a speciální tvar matice $\mathbf{A}$ jsou známy vzorce pro optimální hodnotu ω_{opt} relaxačního parametru. Tyto vzorce však mají význam spíše teoretický, neboť výpočet podle nich je příliš náročný. Proto se pracuje s proměnným relaxačním faktorem, v k-té iteraci s ω_k, a jeho hodnota se v každé iteraci zpřesňuje tak, aby se postupně blížila k optimálnímu ω_{opt}. Konkrétní metody lze najít ve specializované literatuře.

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

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

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

SOR metoda zapsaná po složkách je tvaru

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

Když na pravé straně místo $x_j^{(k+1)}$ píšeme $x_j^{(k)}$, dostaneme relaxaci Jacobiovy metody.

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

$$\mathtt{K} = gallery('Poisson', n)$$

vygenerujeme matici, jejíž strukturu nyní popíšeme. [1]

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

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

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

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

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

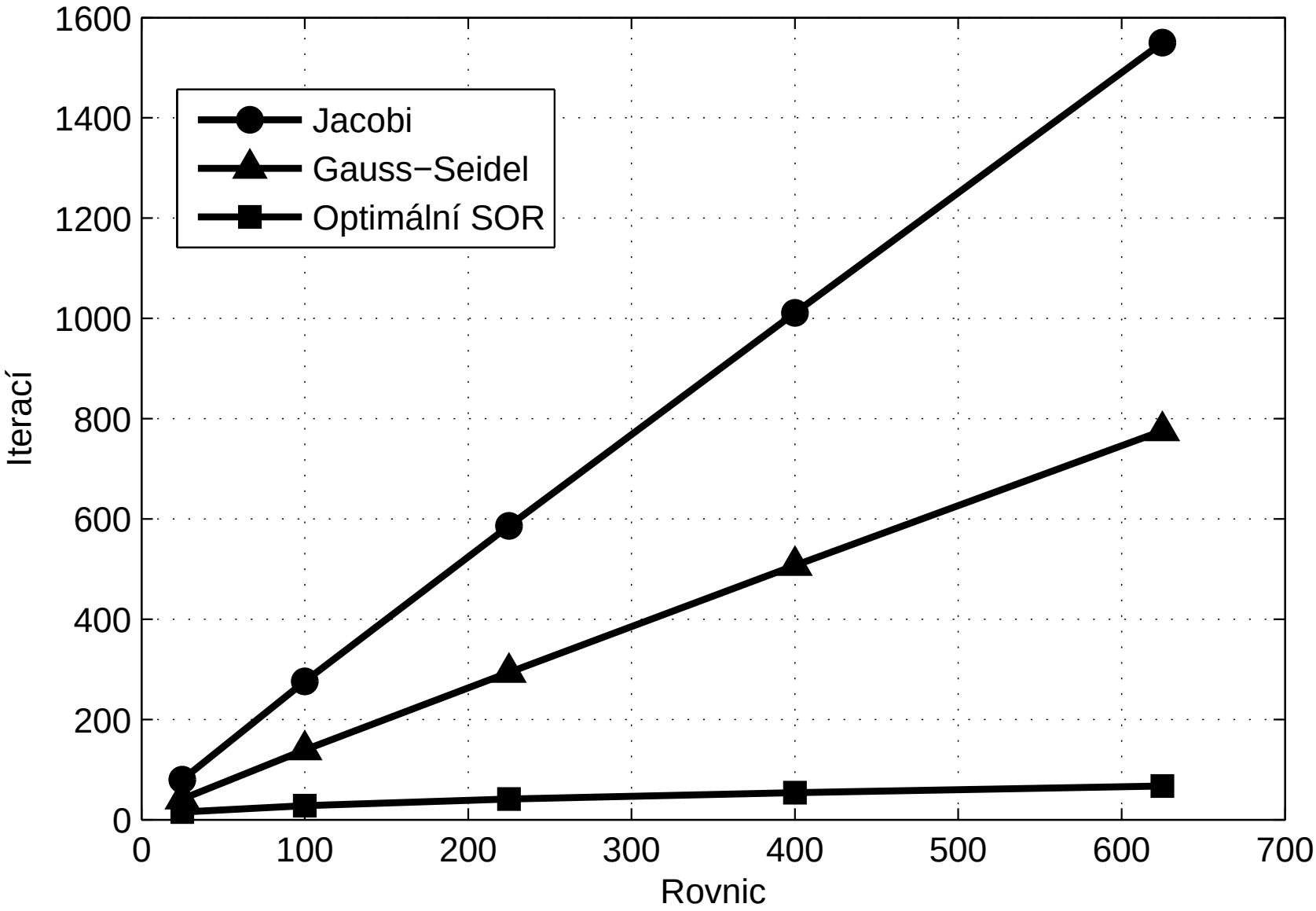
Obecně je $\mathbf{K}$ složena z čtvercových submatic řádu n (tzv. bloků) $\mathbf{B}$, $-\mathbf{I}$, $\mathbf{O}$, které jsou ve čtvercové matici řádu n^2 rozmístěny ve třech diagonálách

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

Matice **I** je jednotková matice, **O** je čtvercová matice s nulovými prvky a **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 **K** se často testuje účinnost numerických metod pro řešení soustav lineárních rovnic s řídkou maticí. Na obrázku 2.3 je vidět závislost počtu iterací (potřebných pro dosažení zvolené přesnosti) na počtu rovnic n^2. Při metodě SOR bylo zvoleno optimální ω.



Obr. 2.3: Srovnání klasických iteračních metod

Poznámka. Iterační metody, se kterými jsme se seznámili, bývají označovány jako klasické iterační metody. V současnosti se používají už jen zřídka. Do učebního textu jsme je zařadili hlavně proto, že jsou poměrně jednoduché a přitom na nich lze dobře ukázat, jak iterační metody fungují.
Na druhé straně metody, které se skutečně používají, jsou poměrně složité k pochopení, takže je v tomto základním kurzu nelze dost dobře vysvětlit. Spokojíme se tedy s konstatováním, že existuje značné množství výkonných iteračních metod, viz např. [Meurant]. Tak třeba pro soustavy s pozitivně definitní maticí patří mezi nejpopulárnější metoda sdružených gradientů, jejíž implementaci najdeme např. v MATLABu. Základní verze této metody je stručně uvedena v kapitole o optimalizaci.
Pro soustavy s nesymetrickou maticí je situace komplikovanější, jednoznačný „favorit“ mezi metodami pro jejich řešení neexistuje. Jednou z mnoha používaných metod je třeba metoda bikonjugovaných gradientů, viz [Meurant], implementace opět viz např. MATLAB.

Obsah

- I.S. Berezin, N.P. Židkov: *Čislennyje metody I,II*, Nauka, Moskva, 1962.
- G. Dahlquist, G. Å Bjőrk: *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- M. Fiedler: *Speciální matice a jejich použití v numerické matematice*, SNTL, Praha, 1981.
- D. Hanselman, B. Littlefield: *Mastering MATLAB 7*, Pearson Prentice Hall, Upper Saddle River, NJ, 2005.
- G. Hämmerlin, K. H. Hoffmann: *Numerical Mathematics*, Springer-Verlag, Berlin, 1991.
- M. T. Heath: *Scientific Computing. An Introductory Survey*, McGraw-Hill, New York, 2002.
- I. Horová, J. Zelinka: *Numerické metody*, učební text Masarykovy univerzity, Brno, 2004.
- J. Kobza: *Splajny*, učební text Palackého univerzity, Olomouc, 1993.
- J. Klapka, J. Dvořák, P. Popela: *Metody operačního výzkumu*, učební text, FSI VUT Brno, 2001.

- J. H. Mathews, K. D. Fink: *Numerical Methods Using MATLAB*, Pearson Prentice Hall, New Jersey, 2004.
- MATLAB: *Mathematics*, Version 7, The MathWorks, Inc., Natick, 2004.
- G. Meurant: *Computer Solution of Large Linear Systems*, Elsevier, Amsterodam, 1999.
- S. Míka: *Numerické metody algebry*, SNTL, Praha, 1985..
- C. B. Moler: *Numerical Computing with MATLAB*, Siam, Philadelphia, 2004.
  `http://www.mathworks.com/moler`.
- J. Nocedal, S. J. Wright: *Numerical Optimization, Springer Series in Operations Research*, Springer, Berlin, 1999.
- A. Quarteroni, R. Sacco, F. Saleri: *Numerical Mathematics*, Springer, Berlin, 2000.
- 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.

- P. Přikryl: *Numerické metody matematické analýzy,* SNTL, Praha, 1985.
- A. R. Ralston: *Základy numerické matematiky,* Academia, Praha, 1973.
- K. Rektorys: *Přehled užité matematiky I,II,* Prometheus, Praha, 1995.
- J. Stoer, R. Bulirsch: *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1993.
- E. Vitásek: *Numerické metody,* SNTL, Praha, 1987.
- W. Y. Yang, W. Cao, T. S. Chung, J. Morris: *Applied Numerical Methods Using Matlab,* John Willey & Sons, New Jersey, 2005.