Práce s maticemi numericky

Mějme čtvercovou matici $A = (a_{i,j})_{i,j=1}^{n}$ reálných čísel. Naším cílem je najít metody pro řešení úloh spojených s takovou maticí. Primárně jde o řešení soustavy rovnic $A\vec{x} = \vec{b}$, dotkneme se i určení determinantu $|A|$ či nalezení inverzní matice A^{-1} a na závěr se podíváme na určování vlastních čísel a vlastních vektorů. Zajímavou shodou okolností lze první tři úlohy (determinant, inverze, soustavy) řešit shodným postupem, tedy převedením dané matice (případně vhodně rozšířené) na horní trojúhelníkový tvar.

1. Regulární matice a eliminace

Hlavní motivací pro tuto kapitolu je řešení soustav lineárních rovnic. Omezíme se na regulární matice, na singulární případ se podíváme ve specializované sekci níže. Eliminace umí převést matici na horní trojúhelníkovou, tedy matici, která má pod diagonálou jen nuly. Protože se takovýto tvar hodí nejen při řešení soustav, podíváme se na obecnější situaci. Budeme pracovat s regulární maticí A, kterou můžeme (ale nemusíme) dále rozšířit o c sloupců, čímž vznikne matice C.

1a. Gaussova eliminační metoda (Gauss elimination)

Algoritmus je všeobecně známý. Pro jednoduchost se podíváme na čtvercovou matici A. Nejprve v prvním sloupci zajistíme, aby byly na všech řádcích od druhého dále nuly, a to odečtením vhodného násobku prvního řádku. Přesně řečeno, prvky nové matice budou pro $i = 2, \dots, n$ a všechna j rovny $a^{(2)}_{i,j} = a_{i,j} - \frac{a_{i,1}}{a_{1,1}} a_{1,j}$. Tím je dán první krok algoritmu:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \mapsto \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 0 & a^{(2)}_{2,2} & \cdots & a^{(2)}_{2,n} \\ \vdots & \vdots & & \vdots \\ 0 & a^{(2)}_{n,2} & \cdots & a^{(2)}_{n,n} \end{pmatrix}.$$

Číslu $a_{1,1}$ se v této souvislosti říká **pivot**. V druhém kroku algoritmu se pivotem stává číslo $a^{(2)}_{2,2}$ a vyrábíme nuly ve sloupci pod ním. Vzorec je obdobný, pro $i = 3, \dots, n$ a $j \geq 2$ máme $a^{(3)}_{i,j} = a^{(2)}_{i,j} - \frac{a^{(2)}_{i,2}}{a^{(2)}_{2,2}} a^{(2)}_{2,j}$. Děláme tedy

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{13} & \cdots & a_{1,n} \\ 0 & a^{(2)}_{2,2} & a^{(2)}_{2,3} & \cdots & a^{(2)}_{2,n} \\ 0 & a^{(2)}_{3,2} & a^{(2)}_{3,3} & \cdots & a^{(2)}_{3,n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & a^{(2)}_{n,2} & a^{(2)}_{n,3} & \cdots & a^{(2)}_{n,n} \end{pmatrix} \mapsto \begin{pmatrix} a_{1,1} & a_{1,2} & a_{13} & \cdots & a_{1,n} \\ 0 & a^{(2)}_{2,2} & a^{(2)}_{2,3} & \cdots & a^{(2)}_{2,n} \\ 0 & 0 & a^{(3)}_{3,3} & \cdots & a^{(3)}_{3,n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a^{(3)}_{n,3} & \cdots & a^{(3)}_{n,n} \end{pmatrix}.$$

Takto pokračujeme.

Pokud vychází nulový pivot, je možné zkusit vyrobit nenulový pivot výměnou řádků, v k-tém kroce hledáme nenulové číslo mezi prvky $a^{(k)}_{k,k}, a^{(k)}_{k+1,k}, \dots, a^{(k)}_{n,k}$. Pokud se mezi nimi nenulový prvek nenajde, pak algoritmus selhal. Konkrétní interpretace tohoto selhání závisí na tom, jakou úlohu zrovna řešíme, zde by to znamenalo, že matice A není regulární, tudíž se s ní máme vydat do příslušné sekce dále.

Tento výběr vhodného pivota se v praxi dělá i v případech, kdy je první kandidát nenulový. Jak totiž dále uvidíme, je numericky výhodné mít pivot co největší, přesněji řečeno mít jako pivot takové číslo, které nejvíce dominuje svému řádku, tedy maximalizuje $\frac{|a_{i,k}|}{\max_{j>k} |a_{i,j}|}$ mezi všemi řádky $i = k, \dots, n$.

Tedy než se dáme do nulování sloupce, musíme zvolit nejvíce dominujícího pivota. Pokud je k tomu třeba prohodit řádky, pak řekneme, že jsme pivotovali.

Algoritmus (GEM: Gaussova eliminace pro převod regulární matice na horní trojúhelníkový tvar)**.**
0. Zadána matice $C = (c_{i,j})_{i,j=1}^{n,m}$ reálných čísel, kde $m \geq n$.
Pro účely algoritmu bereme $c_{i,j}^{(0)} = c_{i,j}$. Nechť $k = 1$.
1. Mezi řádky $i = k$ až n s nenulovým $c_{i,k}^{(k-1)}$ vybereme takový řádek k', který má nejmenší možnou hodnotu $\frac{\max_{j>k} |c_{i,j}^{(k-1)}|}{|c_{i,k}^{(k-1)}|}$.
1a. Jestliže žádný řádek s nenulovým $c_{i,k}^{(k-1)}$ není, pak jde o singulární matici, algoritmus ukončíme a použijeme metody z příslušné podsekce.
1b. Předpokládejme, že se našel řádek k' s nejlepším kandidátem na pivota. Jestliže je $k' \neq k$, zaměníme řádky k a k'.
Pokračujeme krokem **2**.
2. k-tý řádek označíme $c_{k,j}^{(k)}$. Pro $i = k+1, \ldots, n$ provedeme následující:
Určíme $l_{i,k} = \frac{c_{i,k}^{(k-1)}}{c_{k,k}^{(k)}}$, nastavíme $c_{i,k}^{(k)} = 0$ a pro $j > k$ počítáme $c_{i,j}^{(k)} = c_{i,j}^{(k-1)} - l_{i,k} c_{k,j}^{(k)}$.
Jestliže $k < n$, zvýšíme k o jedničku a jdeme zpět na krok **1**.
Jinak algoritmus skončil, výstupem je matice

$$\begin{pmatrix} c_{1,1}^{(1)} & c_{1,2}^{(1)} & \cdots & c_{1,n-1}^{(1)} & c_{1,n}^{(1)} & \cdots & c_{1,m}^{(1)} \\ 0 & c_{2,2}^{(2)} & \cdots & c_{2,n-1}^{(2)} & c_{2,n}^{(2)} & \cdots & c_{2,m}^{(2)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & c_{n-1,n-1}^{(n-1)} & c_{n-1,n}^{(n-1)} & \cdots & c_{n-1,m}^{(n-1)} \\ 0 & 0 & \cdots & 0 & c_{n,n}^{(n)} & \cdots & c_{n,m}^{(n)} \end{pmatrix}.$$

△

Stojí za zmínku, že jsme vlastně mohli skončit již při $k = n-1$, protože matice už pak je v horním trojúhelníkovém tvaru. A opravdu, ten poslední cyklus ($k = n$) nedělá žádné výpočty, jen otestuje, zda je poslední člen diagonály nenulový (chceme, aby algoritmus také rozhodl, zda je matice regulární), a správně pojmenuje poslední řádek. Dalo se to zařídit i jinak, skutečný program by určitě neběhal prázdnými výpočty, ale takto je to pěkně kompaktní.

Kolik operací s reálnými čísly (floating point operations, flops) potřebujeme na provedení GEM? Uvažujme matici $n \times n$ rozšířenou o c sloupců (kde lze mít i $c = 0$).
Ve chvíli, kdy zbývá zpracovat matici $k \times k$, budeme upravovat $k-1$ řádků. Pro každý nejprve jedním dělením připravíme koeficient $l_{i,k}$, pak potřebujeme násobit a odečíst, tedy celkem $2(k+c-1)+1$ operací na řádek. Úprava matice $k \times k$ nás tedy stojí $(k-1)(2k+2c-1)$ flops a děláme to pro matice od velikosti $k = n$ po velikost $k = 2$. Dohromady to dává

$$\begin{aligned} \sum_{k=2}^{n} (k-1)(2k+2c-1) &= \sum_{k=1}^{n} [2k^2 + (2c-3)k + (1-2c)] \\ &= 2\sum_{k=1}^{n} k^2 + (2c-3)\sum_{k=1}^{n} k + (1-2c)\sum_{k=1}^{n} 1 \\ &= 2 \cdot \tfrac{1}{6} n(n+1)(2n+1) + (2c-3)\tfrac{1}{2} n(n+1) + (1-2c)n \\ &= \tfrac{2}{3} n^3 + \tfrac{1}{2}(2c-1)n^2 - \tfrac{1}{6}(6c+1)n. \end{aligned}$$

Pokud řešíme soustavu rovnic (či jejich konečný pevně daný počet), dostáváme následující.

Fakt.
Jestliže aplikujeme GEM na matice velikosti $n \times (n+b)$, kde b je konstanta nezávisející na n, pak je výpočetní náročnost rovna

$$\tfrac{2}{3} n^3 + O(n^2).$$

Jsou aplikace, kde velikost rozšíření (tedy c) roste s velikostí matice (třeba výpočet inverzní matice), pak bude třeba se k obecnému vzorci vrátit a znovu jej vyhodnotit.

1b. Gaussova-Jordanova metoda

Je to jednoduchá modifikace GEM, při které se směřuje k diagonální matici tak, že vyrábíme nuly i nad pivotem. Dále se po výsledné matici chce, aby měla místo pivotů jedničky.

Algoritmus (GJM: Gaussova-Jordanova eliminace pro převod matice na jednotkovou)**.**
0. Zadána matice $C = (c_{i,j})_{i,j=1}^{n,m}$ reálných čísel, kde $m \geq n$.
Pro účely algoritmu bereme $c_{i,j}^{(0)} = c_{i,j}$. Nechť $k = 1$.
1. Mezi řádky $i = k$ až n s nenulovým $c_{i,k}^{(k-1)}$ vybereme takový řádek k', který má nejmenší možnou hodnotu $\frac{\max_{j>k} |c_{i,j}^{(k-1)}|}{|c_{i,k}^{(k-1)}|}$.
1a. Jestliže žádný řádek s nenulovým $c_{i,k}^{(k-1)}$ není, pak jde o singulární matici, algoritmus ukončíme a použijeme metody z příslušné podsekce.
1b. Předpokládejme, že se našel řádek k' s nejlepším kandidátem na pivota. Jestliže je $k' \neq k$, zaměníme řádky k a k'.
Pokračujeme krokem **2**.
2. Nastavíme $c_{k,k}^{(k)} = 1$, pro $j > k$ definujeme $c_{k,j}^{(k)} = \frac{c_{k,j}^{(k-1)}}{c_{k,k}^{(k-1)}}$.
Pro všechna $i \neq k$ nastavíme $c_{i,k}^{(k)} = 0$ a pro $j > k$ počítáme $c_{i,j}^{(k)} = c_{i,j}^{(k-1)} - c_{i,k}^{(k-1)} c_{k,j}^{(k)}$.
Jestliže $k < n$, zvýšíme k o jedničku a jdeme zpět na krok **1**.
Jinak algoritmus skončil, výstupem je matice

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & c_{1,n+1}^{(n)} & \cdots & c_{1,m}^{(n)} \\ 0 & 1 & \cdots & 0 & 0 & c_{2,n+1}^{(n)} & \cdots & c_{2,m}^{(n)} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & c_{n-1,n+1}^{(n)} & \cdots & c_{n-1,m}^{(n)} \\ 0 & 0 & \cdots & 0 & 1 & c_{n,n+1}^{(n)} & \cdots & c_{n,m}^{(n)} \end{pmatrix}.$$

△

Jaká je výpočetní náročnost? Uvažujme matici $n \times n$ rozšířenou o c sloupců (kde lze mít i $c = 0$).
Rozebereme etapu, kdy vpravo dole u A zbývá upravit podmatice velikosti $k \times k$ na jednotkovou. Nejprve vydělíme pivotový řádek, to je $k + c - 1$ operací. Teď je třeba vynulovat $n - 1$ řádků, přičemž každý nás stojí (podobně jako u Gaussovy eliminace) $2k+2c-2$ flops. Celkem $(n-1)(2k+2c-2)+k+c-1$ operací na k-tou etapu. Těchto etap je n, náročnost je proto

$$\begin{aligned} \sum_{k=1}^{n} [(n-1)(2k+2c-2) + k + c - 1] &= \sum_{k=1}^{n} [(2n-1)k + (1 + 2cn - 2n - c)] \\ &= (2n-1) \sum_{k=1}^{n} k + (1 + 2cn - 2n - c) \sum_{k=1}^{n} 1 \\ &= (2n-1) \tfrac{1}{2} n(n+1) + (1 + 2cn - 2n - c)n \\ &= n^3 + \tfrac{1}{2}(4c-3)n^2 + \tfrac{1}{2}(1-2c)n. \end{aligned}$$

Fakt.
Jestliže aplikujeme GJM na matice velikosti $n \times (n + c)$, kde c je konstanta nezávisející na n, pak je výpočetní náročnost rovna

$$n^3 + O(n^2).$$

Vidíme, že Gauss-Jordanova eliminace je sice asymptoticky stejná jako Gaussova eliminace, například $\Theta(n^3)$ pro c konstantní, ale při bližším pohledu je horší. Obecně očekáváme (pro velká n) celkem $\frac{2}{3}n^3 + cn^2$ u GEM a $n^3 + 2cn^2$ u GJM, díky čemuž se GJM na řešení soustav prakticky nepoužívá.

I zde je často (pokud má výchozí matice jen celá čísla) možné použít verzi algoritmu bez dělení, vznikne diagonální matice, po doběhu je třeba každý řádek vydělit tak, aby na pivotech vznikly jedničky. Vychází to opět výpočetně náročnější ($\frac{3}{2}n^3$), ale může to stát za to, pokud reálně hrozí problémy s chybami či počítáme rukou.

1c. Gaussova eliminace a soustavy lineárních rovnic

Z matice A se vyrobí rozšířená matice C připojením pravé strany soustavy jako posledního sloupce, tedy symbolicky $C = (A|\vec{b})$. Nabízí se použít Gauss-Jordanovu eliminaci, řešení rovnice pak rovnou najdeme jako pravý sloupec výsledné matice. Víme již, že nás to bude stát $n^3 + O(n^2)$ flops.

Když použijeme „levnější“ Gaussovu eliminaci, budeme situaci, kdy vlastně řešíme novou soustavu $U\vec{x} = \vec{d}$, kde U je horní trojúhelníková a pravá strana je obvykle po eliminaci jiná, takže jsme nemohli použít písmenko $\vec{b}$. Tato nová soustava reprezentuje rovnice

$$\begin{aligned} u_{1,1}x_1 + u_{1,2}x_2 + \cdots + u_{1,n}x_n &= d_1 \\ u_{2,2}x_2 + \cdots + u_{2,n}x_n &= d_2 \\ &\vdots \\ u_{n,n}x_n &= d_n \end{aligned}$$

Odtud již neznámé snadno dopočítáme tzv. **zpětným chodem** či **zpětným dosazením** (back substitution), tedy

$$\begin{aligned} x_n &= \frac{d_n}{u_{n,n}} \\ x_{n-1} &= \frac{d_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}} \\ x_{n-2} &= \frac{d_{n-2} - u_{n-2,n}x_n - u_{n-2,n-1}x_{n-1}}{u_{n-2,n-2}} \\ &\vdots \\ x_1 &= \frac{d_1 - u_{1,n}x_n - u_{1,n-1}x_{n-1} - \cdots - u_{1,2}x_2}{u_{1,1}} \end{aligned}$$

Kolik je na to třeba operací? V k-tém řádku je v čitateli celkem $k-1$ odčítání a násobení, pak je jedno dělení, celkem $2k-1$ operací. Zpětný chod proto vyžaduje

$$\sum_{k=1}^{n}(2k-1) = 2\sum_{k=1}^{n}k - \sum_{k=1}^{n}1 = 2\tfrac{1}{2}n(n+1) - n = n^2$$

flops. Vzhledem k asymptotické náročnosti $\Theta(n^3)$ Gaussovy eliminace to je zanedbatelné, řešení soustavy rovnic touto metodou tedy vyžaduje $\frac{2}{3}n^3 + O(n^2)$ operací, což je výhodnější než přístup pomocí Gauss-Jordanovy eliminace s jeho n^3 operacemi, ale není to rozdíl v řádu.

Obdobná situace platí pro případ $L\vec{x} = \vec{d}$, kde L je dolní trojúhelníková. Soustavu rovnic

$$\begin{aligned} l_{1,1}x_1 &= d_1 \\ l_{2,1}x_1 + l_{2,2}x_2 &= d_2 \\ &\vdots \\ l_{n,1}x_1 + l_{n,2}x_2 + \cdots + l_{n,n}x_n &= d_n \end{aligned}$$

hravě vyřešíme **dopředným dosazením** (forward substitution), tedy

$$\begin{aligned}
x_1 &= \frac{d_1}{l_{1,1}} \\
x_2 &= \frac{d_2 - l_{2,1}x_1}{l_{2,2}} \\
x_3 &= \frac{d_3 - l_{3,1}x_1 - l_{3,2}x_2}{l_{3,3}} \\
&\vdots \\
x_n &= \frac{d_n - l_{n,1}x_1 - l_{n,2}x_2 - \cdots - l_{n,n-1}x_{n-1}}{l_{n,n}}
\end{aligned}$$

Z programátorského hlediska má tato situace další plus, jsou to přímé vzorce, které lze naprogramovat hravě dvěma vnořenými cykly.

Fakt.
Soustavu $A\vec{x} = \vec{b}$, jejíž regulární matice A je horní (resp. dolní) trojúhelníková, lze řešit zpětným (resp. dopředným) dosazením s výpočetní náročností n^2.

Zkouška a zpřesňování výsledku.

Jak zjistíme, nakolik je nalezené řešení správné? Uděláme zkoušku: Nalezené řešení $\vec{x}$ dosadíme do levých stran rovnic a dostaneme pravé strany $A\vec{x}$. Porovnáním se zadanou pravou stranou získáme **reziduum** $\vec{r} = \vec{b} - A\vec{x}$. Pokud počítáme ve floating point, tak se dá čekat, že občas dojde k chybce, tudíž nám samé nuly nevyjdou. Pak samozřejmě doufáme, že je to vektor s velmi malými čísly.

Nás zajímá chyba řešení, tedy $\vec{E}_x = \vec{x}_0 - \vec{x}$, kde $\vec{x}_0$ je přesné řešení soustavy. Je zde přímá vazba,

$$\vec{r} = A\vec{x}_0 - A\vec{x} = A(\vec{x}_0 - \vec{x}) = A\vec{E}_x \implies \vec{E}_x = A^{-1}\vec{r}.$$

Potřebujeme vědět, zda v případě, že je $\vec{r}$ vektor s malými čísly, bude totéž platit i pro vektor $\vec{E}_x$. Protože inverzní matice je nepříliš pohodlný objekt, raději se vrátíme o krok zpět a položíme si zásadní otázku.

- Jestliže je vektor $\vec{r}$ malý, musí být i řešení řešení soustavy $A\vec{E}_x = \vec{r}$ malé?

To je velmi obecná otázka, kterou odložíme do speciální sekce o podmíněnosti matic.

Zde se alespoň podíváme, jak pomocí znalosti rezidua vylepšit nalezené řešení. Pokud $\vec{E}_x$ přičteme k původnímu řešení $\vec{x}$, získáme řešení $\vec{x}_0$. Kdybychom to $\vec{E}_x$ spočítali přesně, tak už máme přesné řešení, ale my samozřejmě při výpočtu $\vec{E}_x$ zase uděláme nějaké chyby. V praxi to nicméně často dopadá tak, že jsou menší než chyby udělané při výpočty $\vec{x}$, takže aspoň získáme vylepšené řešení. To se dá opakovat, lze tedy následovat tento rekurentní postup:

1. Najdeme nějaké řešení $\vec{x}$ soustavy $A\vec{x} = \vec{b}$.
2. Najdeme reziduum $\vec{r} = \vec{b} - A\vec{x}$, z něj spočítáme $\vec{E}_x$ řešením rovnice $A\vec{E}_x = \vec{r}$. Jestliže je vektor chyb $\vec{E}_x$ tak malý, jak potřebujeme, tak proces ukončíme. Jinak namísto $\vec{x}$ vezmeme $\vec{x} + \vec{E}_x$ a krok 2 opakujeme.

Vzniká tak vlastně iterační metoda, která je v této chvíli docela drahá (každý cyklus by nás stál řádově n^3 operací). Brzy uvidíme, že pokud je matice soustavy stále stejná (což zde je), lze se dostat na n^2 operací v každém cyklu. Nicméně co se týče iteračních metod, dává se přednost jiným, ty si ukážeme později.

1d. Eliminace a determinant

Je známo, že determinant trojúhelníkové matice je součin prvků na diagonále. Gaussova eliminace přitom determinant nemění s výjimkou změny znaménka při permutacích řádků. Po doběhnutí GEM tedy uděláme následující závěr: Determinant matice A je roven nule, pokud algoritmus selhal, nebo součinu prvků na diagonále výsledné matice krát $(-1)^z$, kde z je počet prohození dvou řádků.

Z praktického pohledu je nejjednodušší zavést konstantu $K = 1$. Při každém prohození řádků se udělá $K := -K$, po (zdárném) doběhu algoritmu máme $|A| = K \prod_{k=1}^{n} a_{k,k}^{(k)}$.

Pronásobení diagonály stojí $n-1$ flops, případné násobení číslem K jednu operaci, to vše je vůči Gaussově eliminaci zanedbatelné. Výpočet determinantu tímto způsobem má tedy výpočetní náročnost $\frac{2}{3}n^3 + O(n^2)$.

1e. Eliminace a inverzní matice

Zde je třeba rozšířit A o jednotkovou matici, vznikne tak matice C o rozměru $n \times 2n$, na kterou je třeba aplikovat Gauss-Jordanovu eliminaci. Pokud selže, byla matice A singulární, jinak dostáváme A^{-1} v pravých n sloupcích. Symbolicky:

$$(A|E_n) \mapsto (E_n|A^{-1}).$$

Pokud dosadíme $c = n$ do vzorce pro GJM, dostáváme po úpravě, že výpočetní náročnost získání inverzní matice tímto způsobem je $3n^3 + O(n^2)$.

Stojí za zmínku, že pokud je už daná matice A diagonální, dolní trojúhelníková či horní trojúhelníková, pak matice k ní inverzní je nutně stejného typu a lze ji najít zpětným dosazením.

Uvažujme například horní trojúhelníkovou matici A s obecnými prvky $a_{i,j}$ pro $j \geq i$, zatímco $a_{i,j} = 0$ pro $j < i$. Pak je i inverzní matice U horní trojúhelníková a její prvky splňují tyto rovnice: Pro $k = 1, \ldots, n$ a pro $m = k, \ldots, n$ platí $\sum_{j=k}^{m} a_{k,j}u_{j,m} = \delta_{k,m}$. Rozepíšeme-li si je podle sloupců, dostaneme:

$$\begin{aligned}
&a_{1,1}u_{1,1} = 1, \\
&a_{2,2}u_{2,2} = 1, \quad a_{1,1}u_{1,2} + a_{1,2}u_{2,2} = 0 \\
&a_{3,3}u_{3,3} = 1, \quad a_{2,2}u_{2,3} + a_{2,3}u_{3,3} = 0, \quad a_{1,1}u_{1,3} + a_{1,2}u_{2,3} + a_{1,3}u_{4,3} = 0 \\
&\vdots
\end{aligned}$$

Teď stačí řešit po řádcích zpětným dosazování, dostáváme

$$\begin{aligned}
u_{1,1} &= \frac{1}{a_{1,1}}, \\
u_{2,2} &= \frac{1}{a_{2,2}}, \quad u_{1,2} = -\frac{1}{a_{1,1}}a_{1,2}u_{2,2} \\
u_{3,3} &= \frac{1}{a_{3,3}}, \quad u_{2,3} = -\frac{1}{a_{2,2}}a_{2,3}u_{3,3}, \quad u_{1,3} = -\frac{1}{a_{1,1}}(a_{1,2}u_{2,3} + a_{1,3}u_{3,3}) \\
&\vdots \\
u_{k,k} &= \frac{1}{a_{k,k}}, \quad u_{j,k} = -\frac{1}{a_{j,j}}(a_{j,j+1}u_{j+1,k} + a_{j,j+2}u_{j+2,k} + \cdots + a_{j,k-1}u_{k-1,k} + a_{j,k}u_{k,k}) \\
&\qquad\qquad \text{pro } j = k-1 \text{ až } j = 1.
\end{aligned}$$

Počet operací potřebných pro vypočtení neznámých $u_{k,k}, u_{k-1,k}, \ldots, u_{1,k}$ (tedy k-tý sloupec matice U) je

$$1 + 2 + 4 + \cdots + (2k-2) = 1 + 2\sum_{j=1}^{k-1} j = 1 + (k-1)k,$$

Celkový počet operací tedy vychází

$$\sum_{k=1}^{n}[k^2-k+1] = \tfrac{1}{6}n(n+1)(2n+1) - \tfrac{1}{2}n(n+1) + n = \tfrac{1}{3}n^3 + \tfrac{2}{3}n.$$

Náročnost je $\frac{1}{3}n^3 + O(n^3)$, což je dokonce méně než jednoduchá GEM. Další výhodou je, že nejde o algoritmus s opakovanými kroky, ale dostáváme rovnou přímé vzorce.

1f. Eliminace a singulární matice

Co se stane, když Gaussovu eliminaci aplikujeme na singulární matici? V některém kroku nebudeme mít kandidáta na pivot. To ovšem znamená, že v dotyčném sloupci jsou pod diagonálou již nuly a vlastně ušetříme práci, v onom kroku nic nemusíme dělat a rovnou přejdeme na další k. Dostáváme tak opět horní trojúhelníkovou matici, která ovšem bude mít občas nuly na diagonále, třeba takto:

$$\begin{pmatrix} 2 & -2 & 4 \\ 0 & 0 & 2 \\ 0 & 0 & -6 \end{pmatrix}.$$

Pokud se ovšem snažíme řešit soustavu (což je tady naše hlavní motivace), tak dáváme přednost dalším úpravám vedoucím na tzv. **řádkově schodovitý tvar** (row echelon form), kdy každý řádek začíná nulami, a pokud se objeví nenulový prvek, tak ten první (pivot) už musí mít ve sloupci pod sebou samé nuly. Takže matice výše to rozhodně není, ještě bychom ji potřebovali upravit na tvar

$$\begin{pmatrix} 2 & -2 & 4 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}.$$

V situaci, kdy používáme Gauss-Jordanovu eliminaci, dokonce chceme ještě více, pivoty mají být jedničky a nuly chceme i nad nimi, tedy u naší matice bychom měli dojít sem:

$$\begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Tomu se říká **redukovaný řádkově schodovitý tvar** (reduced row echelon form).

Algoritmy pro GEM a GJM lze snadno upravit, aby produkovaly žádané tvary matice. Hlavní změna je, že už pivoty nemusí ležet na hlavní diagonále, budeme tedy pro uchovávání jejich pozice potřebovat dva parametry, řádkový a sloupcový. Pokud při hledání nenulového pivotu v určitém sloupci selžeme, tak se posouváme o sloupec doprava, ale použijeme stejný pivotovací řádek.

Algoritmus (Gaussova eliminace pro nalezení řádkově schodovitého tvaru)**.**
0. Zadána matice $C = (c_{i,j})_{i,j=1}^{n,m}$ reálných čísel, kde $m \geq n$.
Pro účely algoritmu bereme $c_{i,j}^{(0)} = c_{i,j}$. Nechť $k = 1$, $s = 1$.
1. Mezi řádky $i = k$ až n s nenulovým $c_{i,s}^{(k-1)}$ vybereme takový řádek k', který má nejmenší možnou hodnotu $\frac{\max_{j>s}|c_{i,j}^{(k-1)}|}{|c_{i,s}^{(k-1)}|}$.
1a. Jestliže žádný řádek s nenulovým $c_{i,s}^{(k-1)}$ není a $s < n$, tak zvýšíme s o jedničku a jdeme zpět na krok **1**.
1b. Jestliže žádný řádek s nenulovým $c_{i,s}^{(k-1)}$ není a $s = n$, algoritmus skončil.
1c. Jestliže se řádek s nenulovým $c_{i,s}^{(k-1)}$ našel, algoritmus pokračuje. Jestliže je $k' \neq k$, zaměníme řádky k a k'.
Pokračujeme krokem **2**.
2. k-tý řádek označíme $c_{k,j}^{(k)}$. Pro $i = k+1, \ldots, n$ provedeme následující:
Určíme $l_{i,s} = \frac{c_{i,s}^{(k-1)}}{c_{k,s}^{(k)}}$, nastavíme $c_{i,s}^{(k)} = 0$ a pro $j > s$ počítáme $c_{i,j}^{(k)} = c_{i,j}^{(k-1)} - l_{i,k}c_{k,j}^{(k)}$.
Jestliže $s < n$, zvýšíme k o jedničku, zvýšíme s o jedničku a jdeme zpět na krok **1**.
Jinak algoritmus skončil.
△

Výstupem je matice, která má schodovitou strukturu u nenulové části. Pokud je výsledný prvek na pozici (n, n) nenulový, je tato matice regulární a lze ji využít jako výše. Pokud je tento prvek nula, je dotyčná matice singulární, ale v této upravené formě stále užitečná pro řešení soustavy rovnic, určení hodnoty matice A a další aplikace.

Jen stručně zopakujme, že pokud má rozšířená matice stejnou hodnost jako matice soustavy, pak řešení existuje a dostáváme jej klasickým zpětným chodem, jediný rozdíl je v tom, že proměnné, v jejichž sloupci chybí pivot, bereme jako volitelný parametr.

To ovlivní i naše obecné úvahy o řešení soustav s trojúelníkovou maticí, u singulárních matic již nemáme přímé vzorce, ale je třeba do algoritmu vnést určitou rozhodovací logiku. Nic to ale nemění na výpočetní náročnosti n^2 pro trojúhelníkové matice.

Algoritmus na GJM lze upravit obdobným způsobem. Je ovšem nutno podotknout, že to nemá moc smyslu. GJM se používá prakticky výhradně na hledání inverzní matice, která ovšem pro singulární matice neexistuje.

1g. Stabilita eliminační metody, praktické aspekty

Zajímají nás dvě věci. Numerická stabilita, tedy jak si elminace poradí s chybou na vstupu, a dále její náchylnost ke vzniku nových chyb v průběhu výpočtu.

Situace rozhodně nevypadá dobře. Pokud se někde ve výpočtu objeví chyba, pak má tendenci se zvětšovat při odečítání, což u eliminace děláme pořád. Relativní chyba se také až zdvojnásobuje při dělení a násobení, což normálně moc nebolí, jenže tady násobíme a dělíme opakovaně, dokonce zhruba n^3 krát. Je tu tedy potenciál, že pokud se ve výpočtu objeví chyba, tak se může silně zvětšovat.

Co se týče vzniku nových chyb při výpočtu, kritické je sčítání či odčítání podobných čísel, což nemůžeme vyloučit.

Když to shrneme, vypadá to, že eliminační metoda je numericky nestabilní a riskantní.

Podíváme se na stabilitu podrobněji. Výchozí situace je, že máme matici A_0 a vektor pravých stran $\vec{b}_0$, které by při přesném průběhu výpočtu vedly na přesné řešení $\vec{x}_0$. Pokud ovšem místo toho použijeme pravou stranu $\vec{b}$ s chybou $\vec{E}_b = \vec{b}_0 - \vec{b}$, dostaneme místo toho řešení $\vec{x}$. Chyba na výstupu je pak $\vec{E}_x = \vec{x}_0 - \vec{x}$. Metoda bude stabilní, pokud relativně malému vektoru $\vec{E}_b$ bude odpovídat relativně malý vektor $\vec{E}_x$, slovem relativne naznačujeme, že nás posléze bude hlavně zajímat relativní chyba $\vec{\varepsilon}_x$ a $\vec{\varepsilon}_b$. Nejprve se ale musí zjistit chyby absolutní, vztah mezi nimi se najde takto:

$$A\vec{E}_x = A(\vec{x}_0 - \vec{x}) = A\vec{x}_0 - A\vec{x} = \vec{b}_0 - \vec{b} = \vec{E}_b.$$

Máme tedy vazbu $A\vec{E}_x = \vec{E}_b$, a chceme zjistit, zda je při malém vektoru pravých stran také malé řešení. Přesně stejný problém se v jiném značení objevil v sekci o zkoušce, a tak jsme konstatovali, že se k němu vrátíme, až se naučíme měřit velikosti vektorů a matic. Máme tedy další motivaci se na to podívat, hned s tím začneme.

Podíváme se tam také na druhou a komplikovanější otázku. Jmenovitě nás zajímá, jaká bude chyba řešení $\vec{E}_x$, pokud použijeme správnou pravou stranu $\vec{b}_0$, ale nepřesnou matici A. Budeme chtít porovnat $\vec{E}_x$ s E_A, následně pak relativní chyby.

Teoretických odpovědí jsme se zatím nedočkali, pro praxi je důležité vědět, že existují věci, které si lze pohlídat, aby se vznik chyb minimalizoval.

Jako nejdůležitější faktor se ukazuje, že čím jsou pivoty dominantnější, tím méně eliminace zlobí. V algoritmu jsme to dělali a vyplácí se to.

Zajímavá modifikace je možná v situaci, kdy nevadí násobení řádků (inverzní matice, řešení soustav). Pak je totiž možné zcela vyřadit ze hry dělení a použít vzorec $a^{(k)}_{i,j} = a^{(k)}_{k,k} \cdot a^{(k-1)}_{i,j} - a^{(k-1)}_{i,k} \cdot a^{(k)}_{k,j}$. Zejména u matic s celočíselnými prvky je toto numericky výrazně lepší varianta, které na druhou stranu hrozí přetečení. Je ovšem výpočetně náročnější, je třeba tří operací namísto dvou, čímž náročnost GEM stoupá na $n^3 + O(n^2)$, obdobně pro GJM. Může to ovšem stát za to, zvlášť když přihlédneme k tomu, že operace s celými čísly bývají výrazně rychlejší než floating point operace. Rozhodně je to zajímavá metoda pro ruční výpočty.

2. Stabilita a podmíněnost úlohy

Zde se budeme zabývat čistě citlivostí úlohy řešení soustavy lineárních rovnic na chyby ve vstupních datech, zejména u pravé strany. Ukazuje se, že to velmi záleží na dotyčné matici soustavy. Mluví se o podmíněnosti, přičemž různé obory numerické matematiky se ne zcela shodnou na přesném výkladu, takže se do definice pouštět nebudeme a zatím se spokojíme s představou, že dobře podmíněná matice reaguje na změny pravé strany velmi rezervovaně.

Při matematickém zkoumání se naše úvahy o stabilitě a věrohodnosti zkoušky zkondenzovaly do otázky, zda v situaci $A\vec{E}_x = \vec{r}$ je malá pravá strana zárukou malého řešení. Abychom tuto otázku uměli zodpovědět, musíme se nejpve naučit měřit vektory a matice.

2a. Norma vektoru a matice

Vektory obvykle měříme „Euklidovskou normou“ založenou na Pythagorově větě, ale jsou i jiné způsoby. Ukážeme si několik populárních.

$$\|\vec{x}\| = \sqrt{\sum_{k=1}^{n} |x_k|^2} \qquad \text{(Euklidovská norma)},$$
$$\|\vec{x}\|_\infty = \max_{k=1,\dots,n} |x_k| \qquad \text{(maximová norma)},$$
$$\|\vec{x}\|_1 = \sum_{k=1}^{n} |x_k| \qquad \text{(součtová norma)}.$$

Všechny tři zmíněné lze vnímat jako speciální případ „l_p-normy“ $\|\vec{x}\|_p = \left(\sum_{k=1}^{n} |x_k|^p\right)^{1/p}$, takže klasická Euklidovská norma se také značí $\|\vec{x}\|_2$. Všechny tři výše zmíněné mají ještě jedno společné, pokud jsou malé, pak nutí jednotlivé složky vektoru, aby byly malé také, a naopak. Jsou tedy vhodné pro naše účely. Existuje mezi nimi hierarchie a také se nemohou příliš lišit.

Fakt.
Nechť $n \in \mathbb{N}$. Pak pro všechny vektory $\vec{x} \in \mathbb{R}^n$ platí

$$\|\vec{x}\|_\infty \le \|\vec{x}\|_2 \le \|\vec{x}\|_1 \le n\,\|\vec{x}\|_\infty,$$
$$\|\vec{x}\|_1 \le \sqrt{n}\,\|\vec{x}\|_2, \qquad \|\vec{x}\|_\infty \le \sqrt{n}\,\|\vec{x}\|_2.$$

Z toho vyplývá, že v mnoha situacích je vlastně jedno, kterou normu použijeme.

Obecně u libovolného vektorového prostoru (třeba prostoru posloupností, prostoru funkcí atd.) se můžeme pokusit měřit velikost prvků.

Definice.
Nechť V je vektorový prostor. Zobrazení $|||\cdot|||\colon V \mapsto \mathbb{R}$ se nazývá **norma**, jestliže splňuje tyto vlastnosti:
- $|||\vec{x}||| \ge 0$ pro všechna $\vec{x} \in \mathbb{R}^n$;
- $|||\vec{x}||| = 0 \iff \vec{x} = \vec{0}$;
- $|||c\vec{x}||| = |c| \cdot |||\vec{x}|||$ pro všechna $\vec{x} \in \mathbb{R}^n$ a $c \in \mathbb{R}$;
- $|||\vec{x} + \vec{y}||| \le |||\vec{x}||| + |||\vec{y}|||$ pro všechna $\vec{x}, \vec{y} \in \mathbb{R}^n$ (trojúhelníková nerovnost).

Všechny normy zmíněné výše toto splňují.

Vraťme se k rovnosti $A\vec{E}_x = \vec{r}$. Zvolíme-li určitou normu, pak zkoumáme situaci, kdy $|||\vec{r}|||$ je malé číslo, tedy $|||A\vec{E}_x|||$ je malé. Lze z toho odvodit něco o normě vektoru $\vec{E}_x$? Kdybychom takto pracovali s čísly, tak snadno z $|ae| = |r|$ spočítáme $|e| = \frac{1}{|a|}|r|$. Rádi bychom tedy našli nějaký způsob, jak poměřovat

působení matice A, tedy pokud možno normu na prostoru matic. Vlastnosti v definici výše ale nestačí, nechceme tedy jen „normu“. Matice lze totiž násobit a rádi bychom, aby si norma s touto operací rozuměla.

Definice.
Nechť V je vektorový prostor matic. Zobrazení $|||\cdot|||\colon V \mapsto \mathbb{R}$ se nazývá **maticová norma**, jestliže je to norma a navíc splňuje
• $|||AB||| \leq |||A||| \cdot |||B|||$ pro všechny $A, B \in M_{n\times n}$.

Vztahu $|||AB||| \leq |||A||| \cdot |||B|||$ se také říká „Schwarzova nerovnost“. Pokud jej aplikujeme na případ $A = B = E_n$, tak hned vidíme, že každá maticová norma musí splňovat $|||E_n||| \geq 1$.

Opět máme na výběr z mnoha kandidátů, které úzce souvisejí s velikostí jednotlivých položek matice, oblíbené jsou zejména tyto:

$$\|A\|_\infty = \|A\|_R = \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{i,j}| \qquad \text{(řádková norma, row-sum norm)},$$

$$\|A\|_1 = \|A\|_C = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{i,j}| \qquad \text{(sloupcová norma, column-sum norm)},$$

$$\|A\| = \|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{i,j}|^2} \qquad \text{(Frobeniova norma, Frobenius norm)}.$$

Existují i další, některé z nich jsou teoreticky lepší, ale zase se hůře počítají, tyto tři jsou populární zejména díky snadnému výpočtu.

Tyto maticové normy shodně poznají matici s velmi malými prvky, i když každá trochu jinak. Na rozdíl od vektorů neexistuje mezi populárními maticovými normami jasná hierarchie, něco jako obecná nerovnost typu $\|\cdot\|_1 \leq \|\cdot\|_F$. Jsou ale srovnatelné.

Fakt.
Nechť $n \in \mathbb{N}$. Pro libovolnou $n \times n$ matici A platí

$$\begin{aligned} \|A\|_1 &\leq n\,\|A\|_\infty, & \|A\|_\infty &\leq n\,\|A\|_1, \\ \|A\|_F &\leq \sqrt{n}\,\|A\|_1, & \|A\|_F &\leq \sqrt{n}\,\|A\|_\infty, \\ \|A\|_1 &\leq \sqrt{n}\,\|A\|_F, & \|A\|_\infty &\leq \sqrt{n}\,\|A\|_F. \end{aligned}$$

Máme tedy nějakou normu vektoru $|||\cdot|||$ a normu matice $|||\cdot|||_M$ a jsme v situaci, kdy víme, že $|||A\vec{x}|||$ je malé číslo. Dokážeme toto nějak spojit s čísly $|||A|||_M$ a $|||\vec{x}|||$? Obecně bohužel ne. Musíme si vybrat normu vektoru a normu matice tak, aby spolu nějakým způsobem souvisely.

Definice.
Uvažujme normu $|||\vec{x}|||$ pro vektory z $\mathbb{R}^n$ a normu $|||A|||_M$ pro matice z $M_{n\times n}$. Řekneme, že tyto normy jsou **souhlasné**, jestliže $|||A\vec{x}||| \leq |||A|||_M \cdot |||\vec{x}|||$ pro všechny $A \in M_{n\times n}$ a $\vec{x} \in \mathbb{R}^n$.

V praxi se obvykle udělá ještě krok navíc a dosáhne se ideálního spárování.

Věta.
(i) Nechť $|||\cdot|||$ je vektorová norma. Číslo definované pro $A \in M_{n\times n}$ vzorcem

$$|||A|||_M = \sup\Big\{\frac{|||A\vec{x}|||}{|||\vec{x}|||};\ \vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\}\Big\} = \sup\{|||A\vec{x}|||;\ \vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\} \wedge |||\vec{x}||| \leq 1\}$$

pak dává maticovou normu souhlasnou s $|||\cdot|||$.
(ii) Nechť $|||\cdot|||_M$ je maticová norma. Číslo definované pro $\vec{x} \in \mathbb{R}^n$ vzorcem

$$|||\vec{x}||| = \left|\left|\left| \begin{pmatrix} x_1 & 0 & \dots & 0 \\ x_2 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ x_n & 0 & \dots & 0 \end{pmatrix} \right|\right|\right|_M$$

se nazývá norma indukovaná maticovou normou $|||\cdot|||_M$.
Tyto dvě normy jsou souhlasné.

Pokud z normy vektoru $|||\cdot|||$ vytvoříme podle vzorce z věty normu matice, říkáme jí maticová norma **indukovaná** normou $|||\cdot|||$ (induced norm). Je to ten nejbližší možný vztah a přesně takto se snažíme s normami pracovat.

Jak již znamení naznačuje, maticová norma $\|A\|_1$ je indukovaná normou $\|\vec{x}\|_1$, zatímco maticová norma $\|A\|_\infty$ je indukovaná normou $\|\vec{x}\|_\infty$. Frobeniova norma není indukovaná ničím, tedy ani Euklidovskou $\|\vec{x}\|_2$. Jaká je tedy indukovaná maticová norma $\|A\|_2$? Říká se jí **spektrální norma** a je dána vzorcem $\|A\|_2 = \rho(A^*A)$, jde tedy o největší (v absolutní hodnotě) vlastní číslo matice A^*A. Protože nalezení vlastních čísel je drsná úloha, která tu ostatně bude mít vlastní sekci, bývá tato norma častá spíše v teoretických úvahách, v praxi je snažší používat jednu ze tří norem uvedených výše.

Indukované normy jsou pro numerickou matematiku velice užitečné, jako malou ochutnávku si ukážeme souvislost s invertováním.

Fakt.
Uvažujme $n \times n$ matici B.
(i) Jestliže je $|||\cdot|||$ nějaká indukovaná maticová norma a $|||B||| < 1$, pak je matice $I - B$ invertibilní a platí

$$|||(E_n - B)^{-1}||| \leq \frac{1}{1 - |||B|||}.$$

Tato inverze se dá získat součtem řady

$$(E_n - B)^{-1} = E_n + B + B^2 + B^3 + B^4 + \cdots$$

(ii) Jestliže je matice $E_n - B$ singulární, pak $|||B||| \geq 1$ pro všechny maticové normy.

Teď už si ukážeme, k čemu jsou maticové normy numerickému matematikovi.

2b. Podmíněnost matice

Zvolme tedy nějaké souhlasné normy $|||\cdot|||$ pro matice a vektory, těch se nyní budeme držet. Vracíme se k problému, jak vytěžit informace z rovnosti $A\vec{E} = \vec{E}_b$, zde jsme tedy zvolili interpretaci, kdy zkoumáme chybu řešení v závislosti na chybě pravé strany na vstupu.

Definice.
Pro $n \times n$ matici A zavádíme její **číslo podmíněnosti** (condition number) jako

$$\mathrm{cond}(A) = |||A||| \cdot |||A^{-1}|||.$$

Ze Schwarzovy nerovnosti odvodíme, že vždy $\mathrm{cond}(A) \geq 1$. Z vlastností normy vyplývá, že toto číslo je nezávislé na násobení matice konstantou. To naznačuje rozumné chování nového pojmu. Jeho užitečnost je vyjádřena v následujícím tvrzení.

Věta.
Předpokládejme, že vektory $\vec{x}_0, \vec{x}$ a $\vec{b}_0, \vec{b}$ jsou svázány vztahem $A\vec{x}_0 = \vec{b}_0$ a $A\vec{x} = \vec{b}$.
Označme $\vec{E}_x = \vec{x}_0 - \vec{x}$ a $\vec{E}_b = \vec{b}_0 - \vec{b}$. Pak platí

$$\frac{|||\vec{E}_x|||}{|||\vec{x}|||} \leq \mathrm{cond}(A) \frac{|||\vec{E}_b|||}{|||\vec{b}|||}.$$

Důkaz: Z rovnosti $\vec{E}_x = A^{-1}\vec{E}_b$ máme $|||\vec{E}_x||| \leq |||A^{-1}||| \cdot |||\vec{E}_b|||$. Máme ovšem také $\vec{b} = A\vec{x}$, tedy $|||\vec{b}||| \leq |||A||| \cdot |||\vec{x}|||$. Proto

$$\frac{|||\vec{E}_x|||}{|||\vec{x}|||} \leq \frac{|||A^{-1}||| \cdot |||\vec{E}_b|||}{|||\vec{x}|||} \leq \frac{|||A^{-1}||| \cdot |||\vec{E}_b|||}{\frac{1}{|||A|||}|||\vec{b}|||} = |||A^{-1}||| \cdot |||A||| \frac{|||\vec{E}_b|||}{|||\vec{b}|||}.$$

Výsledek je nasnadě.

□

Výrazy ve zlomku jsou relativní chyby, výsledek lze tedy přepsat následovně:

$$\varepsilon_x \leq \mathrm{cond}(A)\varepsilon_b.$$

To je samozřejmě jen horní odhad, tedy nejpesimičtější možný výsledek, u typického příkladu bude nárust chyby menší, ale spoléhat se na to nedá. Vidíme, že čím větší číslo podmíněnosti, tím větší je problém s hodnověrností výsledku.

Ony dvě nerovnosti použité v důkazu výše úplně stačí k odvození obecnějšího výsledku, který bere v úvahu i změnu matice.

Věta.
Předpokládejme, že matice A_0, A a vektory $\vec{x}_0, \vec{x}$ a $\vec{b}_0, \vec{b}$ jsou svázány vztahem $A_0\vec{x}_0 = \vec{b}_0$ a $A\vec{x} = \vec{b}$.
Označme $E_A = A_0 - A$, $\vec{E}_x = \vec{x}_0 - \vec{x}$ a $\vec{E}_b = \vec{b}_0 - \vec{b}$. Pak platí

$$|||\vec{E}_x||| \leq |||A_0^{-1}|||(|||\vec{E}_b||| + |||E_A||| \cdot |||\vec{x}|||),$$
$$\varepsilon_x \leq \mathrm{cond}(A)\Big(\varepsilon_b + \varepsilon_A \cdot \frac{|||\vec{x}|||}{|||\vec{x}_0|||}\Big).$$

Důkaz: Odečtením výchozích rovnic dostáváme

$$\vec{E}_b = A_0\vec{x}_0 - A\vec{x} = A_0\vec{x}_0 - A_0\vec{x} + A_0\vec{x} - A\vec{x} = A_0\vec{E}_x + E_A\vec{x}.$$

Odtud

$$\vec{E}_x = A_0^{-1}(\vec{E}_b - E_A\vec{x}).$$

Vlastnosti maticové normy (submultiplikativita, trojúhelníková nerovnost) dají první vztah.
Ten pak vydělíme číslem $|||\vec{x}_0|||$ a na pravé straně násobíme a dělíme číslem $|||A_0|||$, dostáváme

$$|||\vec{\varepsilon}_x||| = |||A_0^{-1}||| \cdot |||A_0|||\Big(\frac{|||\vec{E}_b|||}{|||A_0||| \cdot |||\vec{x}_0|||} - |||\varepsilon_A||| \cdot \frac{|||\vec{x}|||}{|||\vec{x}_0|||}\Big).$$

Aplikací $|||A\vec{x}_0||| \leq |||A_0|||\,|||\vec{x}_0|||$ na jmenovatel získáme žádaný vztah.

□

Vidíme, že pokud jsou si $\vec{x}$ a $\vec{x}_0$ blízké, tak je jejich podíl skoro jedna a dá se zanedbat, při situaci beze změny pravé strany pak máme v zásadě

$$\varepsilon_x \leq \mathrm{cond}(A)\varepsilon_A.$$

Interpretace.
1. Interpretace $A\vec{E}_x = \vec{r}$, tedy zkoumání rezidua při dělání zkoušky, nám říká, že pokud u matic s malým číslem podmíněnosti vyjde velmi malé residuum, pak se již dá řešení docela věřit. Praktický pohled vzpadá takto:

Fakt.
Jestliže jsou matice A a vektor $\vec{b}$ věrohodné na k desetinných míst, pak řešení $\vec{x}$ rovnice $A\vec{x} = \vec{b}$ je věrohodné na $k - \log_{10}(\mathrm{cond}(A))$ míst.

2. Interpretace $A\vec{E}_x = \vec{E}_b$ a podobné výsledky ukazující souvislost mezi chybou řešení a chybou pravé strany či matice mají mnohem zásadnější dopad než jen ohledně chyby na vstupu. Ve skutečnosti nám toto ukazuje i na dopad zaokrouhlovacích chyb, které se stávají vstupem pro další kroky algoritmu, přičemž o jejich šíření vypovídá číslo podmíněnosti aktuální matice.

Pokud tedy chceme omezit šíření chyb vzniklých zaokrouhlováním v průběhu výpočtu, musíme se snažit o co nejmenší číslo podmíněnosti u všech matic během výpočtu. To nás zase přivádí k pivotování. Není těžké nahlédnout, že výměna řádků nemění číslo podmíněnosti dotyčné matice, takže z pohledu matice samotné to nemá dopad na její chování. Může to ale výrazně ovlivnit vlastnosti matice, která vznikne eliminačními kroky. Zlepšení se očekávat nedá, snažíme se tedy alespoň o to, abychom číslo podmíněnosti příliš nezhoršili. Obecné pravidlo říká, že čím je pivot ve svém řádku dominantnější, tím méně se zhorší podmíněnost.

3. LU rozklad (LU decomposition/factorization)

V praxi dochází někdy k situaci, že se soustava $A\vec{x} = \vec{b}$ řeší opakovaně se stejnou maticí, jen se mění pravá strana. Opakovat GEM s $\frac{2}{3}n^3$ je nepříjemné, šlo by to lépe?

Z prvního běhu $(A|\vec{b}) \longrightarrow (U|\vec{d})$ se při změně pravé strany zachová matice U, takže vlastně stačí na nový vektor $\vec{b}$ aplikovat přesně stejné řádkové operace jako předtím a získáme správný vektor $\vec{d}$ k matici U. Není problém si ty operace v průběhu GEM ukládat algoritmicky, jsou to vlastně konstanty $l_{i,k}$, ale matematicky zajímavější je udělat to strukturálně.

Definice.
Uvažujme $n \times n$ matici A reálných čísel. Řekneme, že $n \times n$ matice L, U jsou jejím **LU rozkladem** (LU decomposition, LU factorization), jestliže je U horní trojúhelníková matice, L je dolní trojúhelníková matice s jedničkami na diagonále a $A = LU$.

Dá se ukázat, že rozklad buď neexistuje, nebo existuje a je jediný. Existence LU rozkladu selhává i u příjemných matic (regulární, symetrická). Rozpoznat, kdy existence existuje, není úplně jendoduché, jak ukazuje následující věta.

Věta.
Nechť A je reálná $n \times n$ matice. Jestliže má hodnost n a jejích prvních n hlavních minorů je nenulových, pak má LU rozklad.

Počítání determinantů je docela drahé, je tedy jednodušší to prostě zkusit a uvidí se, zda to vyjde.

Fakt.
Jestliže GEM bez pivotování aplikovaná na A vede na trojúhelníkovou matici U, pak LU rozklad matice A existuje a U je ta správná matice. Matici L získáme z koeficientů $l_{i,k}$.

Náznak důkazu: Víme, že každou řádkovou operaci Gaussovy eliminace lze realizovat jako násobení matice zleva jinou maticí, tzv. elementární. Například pokud chceme odečíst c násobek k-tého řádku od řádku i v matici A, pak to lze realizovat jako $L \cdot A$, kde L má tvar jednotkové matice, u níž je navíc na pozici (i,k) číslo $-c$. Rovnou si rozmyslíme, že inverzní matice by měla tuto operaci napravit, takže L^{-1} je jednotková matice, u které jsme na pozici (i,k) přidali číslo c, v našem algoritmu označené $l_{i,k}$. Obě matice jsou dolní trojúhelníkové, pokud máme $i > k$, což je v GEM splněno.

Matice obdržená Gaussovou eliminací se tedy dá získat také jako $U = L_N \cdots L_2 \cdot L_1 A$. Pak ovšem $A = L_1^{-1} \cdot L_2^{-1} \cdots L_N^{-1} U$. Označme $L = L_1^{-1} \cdot L_2^{-1} \cdots L_N^{-1}$. Protože v algoritmu Gaussovy eliminace máme vždy $i > k$, jsou matice L_j^{-1} všechny dolní trojúhelníkové, tudíž i matice L. Našli jsme tedy příslušný rozklad.

Je dobré se zamyslet, co se stane, když někde v algoritmu najdeme nulový pivot. Pak se podíváme do sloupce pod něj. Pokud jsou tam nuly, pak vlastně nemusíme nic dělat, matice už tam vyhovuje podmínce na dolní trojúhelníkovost, a můžeme postoupit k dalšímu k. Pokud někde pod nulovým pivotem najdeme nenulový člen, tak díky nemožnosti pivotování algoritmus ukázal, že LU rozklad není možný.

Tato procedura by sice vytvořila horní trojúhelníkovou matici U dle definice LU rozkladu, ale brzy uvidíme, že preferujeme U v řádkově schodovitém tvaru, tedy v případě nul pod nulovým pivotem se přesuneme o slopec doprava, ale se stejným pivotovacím řádkem. Jak to ovlivní rozklad? Nijak, matici L zajímá čistě informace $l_{i,k}$, tedy který řádek se odčítal kolikrát od kterého, a nestaral se, který sloupec nás k tomu inspiroval. To je ostatně vidět na tom, že se ve značení $l_{i,k}$ nevyskytuje sloupec s.

Algoritmus (LU rozklad matice)**.**
0. Zadána $n \times n$ matice $A = (a_{i,j})_{i,j=1}^{n}$ reálných čísel.
Pro účely algoritmu bereme $u_{i,j}^{(0)} = a_{i,j}$. Nechť $k = 1$, $s = 1$.
1. Jestliže $u_{k,s}^{(k-1)} \neq 0$, tak k-tý řádek označíme $u_{k,j}^{(k)}$ a pokračujeme krokem **2**.
Jinak se podíváme na prvky $u_{i,s}^{(k-1)} \neq 0$ pro $i > k$. Pokud je mezi nimi nenulové číslo, tak algoritmus skončil, LU rozklad je nemožný.
Pokud jsou všechna čísla $u_{k,s}^{(k-1)}$, $u_{k+1,s}^{(k-1)}$, … , $u_{n,s}^{(k-1)}$ nulová, pak v případě $s = n$ algoritmus skončil a jdeme na výstup, jinak zvýšíme s o jedničku a jdeme zpět na krok **1**.
2. Pro $i = k+1, \ldots, n$ provedeme následující: Určíme $l_{i,k} = \frac{u_{i,s}^{(k-1)}}{u_{k,s}^{(k)}}$, nastavíme $u_{i,s}^{(k)} = 0$ a pro $j > s$ počítáme $u_{i,j}^{(k)} = u_{i,j}^{(k-1)} - l_{i,k} u_{k,j}^{(k)}$.
Jestliže $s < n$, zvýšíme k o jedničku, zvýšíme s o jedničku a jdeme zpět na krok **1**.
Jinak algoritmus skončil. Výstupem jsou matice

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ l_{2,1} & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots \\ l_{n-1,1} & l_{n-1,2} & \cdots & 1 & 0 \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n-1} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ 0 & u_{2,2}^{(2)} & \cdots & u_{2,n-1}^{(2)} & u_{2,n}^{(2)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & u_{n-1,n-1}^{(n-1)} & u_{n-1,n}^{(n-1)} \\ 0 & 0 & \cdots & 0 & u_{n,n}^{(n)} \end{pmatrix}.$$

△

Protože jde vlastně o GEM, nepočítá se nic navíc. Výpočetní náročnost LU rozkladu je $\frac{2}{3}n^3 + O(n^2)$.
Pohled na výstup ukazuje, že matice U má nuly právě tam, kde si u L potřebujeme pamatovat čísla. V implementacích je proto možné během Gaussovy eliminace ukládat $l_{i,j}$ pro $i > j$ namísto nul do upravované matice.

Pokud při LU rozkladu povolíme pivotování, pak lze algoritmus provést pro každou matici. Součin LU pak ale nedává A, ale A s permutovanými řádky, přičemž permutace odpovídají přesně těm, které jsme dělali v průběhu algoritmu. Existuje tedy permutační matice P taková, že $LU = PA$.

Fakt.
Pro každou čtvercovou matici A existují permutační matice P, dolní trojúhelníková matice L a horní trojúhelníková matice U takové, že $PA = LU$.

Tomuto se říká **LUP rozklad**.

Jak to proběhne v praxi? Bohužel si nemůžeme představit, že jakoby chytře zpermutujeme A a pak aplikujeme rozklad, čímž by vzniklo přímo $PA = LU$. Je třeba zasahovat i do vznikající L. Není to ale složité.

Poznamenejme, že obecně jsou permutační matice právě jednotkové matice s permutovanými řádky a mají zajímavé vlastnosti. Determinant je vždy jednotkový, z praktického pohledu je také velice užitečné, že $P^{-1} = P^T$.

Algoritmus (LUP rozklad matice)**.**
0. Zadána $n \times n$ matice $A = (a_{i,j})_{i,j=1}^n$ reálných čísel.
Označíme $U = A$. Nechť $k = 1$, $s = 1$, nechť $L = P = E_n$ (jednotková matice).
1. Pokud platí $u_{k,s} = u_{k+1,s} = \cdots = u_{n,s} = 0$, tak zvýšíme s o jedničku. Jestliže $s > n$, algoritmus skončil. Jinak se vrátíme na krok **1**.
Předpokládejme, že mezi $u_{k,s}, u_{k+1,s}, \ldots, u_{n,s}$ existuje nenulový prvek. Mezi řádky $i = k$ až n s nenulovými $u_{i,s}$ vybereme takový řádek k', který má nejmenší možnou hodnotu $\frac{\max_{j>s}|u_{i,j}|}{|u_{i,s}|}$.
Jestliže $k' > k$, tak zaměníme řádky k a k' v maticích U a P, v matici L zaměníme úseky prvních $k-1$ prvků řádků k a k'.
Pokračujeme krokem **2**.
2. Pro $i = k+1, \ldots, n$ provedeme následující: Určíme $l_{i,k} = \frac{u_{i,s}}{u_{k,s}}$, nastavíme $u_{i,s} = 0$ a pro $j > s$ definujeme nové $u_{i,j}$ jako $u_{i,j} - l_{i,k}u_{k,j}$.
Pokračujeme krokem **3**.
3. Jestliže $s < n$, zvýšíme k o jedničku, zvýšíme s o jedničku a jdeme zpět na 1.
Jinak algoritmus skončil. Výstupem jsou matice P, L, U.
△

Posuny sloupců věci komplikují, proto se autoři knih tradičně omezují na regulární matice. Má to jisté odůvodnění, v praktických aplikacích jsou singulární matice velkou výjimkou, nicméně běžné softwarové balíčky si se singulárními maticemi umějí proadit a je dobré vědět, co vlastně dělají.

3a. LU(P) rozklad a soustavy rovnic

Vracíme se k soustavě $\vec{x} = A^{-1}\vec{b}$, u které čekáme nové pravé strany. Lze samozřejmě spočítat jednou A^{-1} s cenou $n^3 + O(n^2)$ a pak dopočítávat $\vec{x} = A^{-1}\vec{b}$, to nás stojí jen $2n^2 + O(n)$ operací, je to tedy lepší než dělat opakovaně Gaussovku.

Existuje ale ještě lepší varianta. Pokud se nám podaří udělat LUP rozklad $PA = LU$, nabízí se následující myšlenka: Pro pravou stranu $\vec{b}$ máme $A\vec{x} = \vec{b}$ neboli $PA\vec{x} = P\vec{b}$, neboli $LU\vec{x} = P\vec{b}$.

Pokud si označíme $U\vec{x} = \vec{y}$, dostáváme soustavu rovnic $L\vec{y} = P\vec{b}$, kterou hravě vyřešíme dopředným dosazením díky tomu, že je L dolní trojúhelníková. Tím známe $\vec{y}$ a lze spočítat z rovnice $U\vec{x} = \vec{y}$ zpětným dosazením žádané řešení $\vec{x}$. Jak jsme odvodili v sekci 1c, dosazování má náročnost n^2, takže i s násobením permutační maticí to dává $3n^2$. Je to na první pohled mírně víc než při násobení inverzní maticí, ale matice P má v každém řádku jen jednu jedničku, takže reálně to je hned.

Celkem je proto situace výrazně lepší než při použití inverzní matice, protože LUP rozklad stojí $\frac{2}{3}n^3$ versus $3n^3$ pro inverzní matici. Závěr: LU(P) rozklad je ten pravý přístup k řešení soustav v případech, kdy se nám průběžně mění pravé strany. Symbolicky:

$$(L|P\vec{b}) \longrightarrow \vec{d}, \qquad (U|\vec{d}) \longrightarrow \vec{x}.$$

4. Iterační metody řešení rovnic

Pokud je dáno zobrazení T z $\mathbb{R}^n$ do $\mathbb{R}^n$, pak pevným bodem rozumíme vektor $\vec{x}_f$ splňující $T(\vec{x}_f) = \vec{x}_f$. Odtud pak iterační metoda $\vec{x}_{k+1} = T(\vec{x}_k)$, u které doufáme v konvergenci.

Uvažujme posloupnost vektorů $\vec{x}_k \in \mathbb{R}^n$ a $\vec{x} \in \mathbb{R}^n$, souřadnice vektoru $\vec{y}$ teď budeme značit $(\vec{y})_i$. Řekneme, že $\lim(\vec{x}_k) = \vec{x}$, také značeno $\vec{x}_k \to \vec{x}$, jestliže $\lim((\vec{x}_k)_i) = (\vec{x})_i$ pro všechna $i = 1, \dots, n$.

Je snadné ukázat, že tato podmínka je ekvivalentní podmínce $\|\vec{x}_k - \vec{x}\| \to 0$, přičemž na použité normě díky jejich ekvivalenci nezáleží. Podobně konvergenci posloupnosti matic $\{A_k\}$ k matici A definujeme podmínkou $(A_k)_{i,j} \to (A)_{i,j}$ a lze to převést na $|||A_k - A||| \to 0$.

Nás budou zajímat rovnice speciálního tvaru, $\vec{x} = B\vec{x} + \vec{c}$. Odtud pak dostáváme přirozené iterační schéma $\vec{x}_{k+1} = B\vec{x}_k + \vec{c}$. Zde je $\varphi(\vec{x}) = B\vec{x} + \vec{c}$ a násobení matic je při práci se souhlasnými normami spojitá operace, takže podobně jako u funkcí můžeme odvodit, že pokud přinutíme posloupnost $\{\vec{x}_k\}$ ke konvergenci k nějakému vektoru $\vec{x}_f$, tak už půjde o hledaný pevný bod.

Nejpřirozenějším nástrojem k vynucení konvergence je kontrakce. Zajímá nás tedy, zda při značení $\varphi(\vec{x}) = B\vec{x} + \vec{c}$ dokážeme vyrobit $q < 1$ tak, aby $\|\varphi(\vec{x}) - \varphi(\vec{y})\| \le q\|\vec{x} - \vec{y}\|$. Pokud dosadíme za φ, tak dostáváme díky linearitě výraz $\|B(\vec{x}-\vec{y})\| \le q\|\vec{x}-\vec{y}\|$, což po přeznačení vede na nerovnost $\|B\vec{z}\| \le q\|\vec{z}\|$. Můžeme něco takového zajistit? Naše zkoumání norem matic okamžitě nabízí výsledek.

Věta.
Jestliže matice B splňuje $|||B|||_M < 1$ pro některou souhlasnou maticovou formu, pak příslušná iterační metoda konverguje k $\vec{x}_f$ pro libovolnou volbu $\vec{x}_0$. Konvergence je lineární a platí

$$|||\vec{x}_f - \vec{x}_{k+1}||| \le \frac{|||B|||_M}{1 - |||B|||_M} |||\vec{x}_{k+1} - \vec{x}_k|||.$$

Vzorec dává zajímavou možnost odhadovat chybu odhadu $\vec{x}_k$ pomocí posledních dvou iterací, podobně jako tomu bylo při hledání pevného bodu pro funkce.

Důkaz: Odhadujeme:

$$|||\vec{x}_f - \vec{x}_{k+1}||| = |||(B\vec{x}_f + \vec{c}) - (B\vec{x}_k + \vec{c})||| = |||B(\vec{x}_f - \vec{x}_k)||| \le |||B||| \cdot |||\vec{x}_f - \vec{x}_k|||$$

Metoda je tedy opravdu lineární. Odtud indukcí

$$|||\vec{x}_f - \vec{x}_k||| \le |||B|||^k \cdot |||\vec{x}_f - \vec{x}_0|||,$$

což dokazuje konvergenci $\{\vec{x}_k\}$. Když od $|||B\vec{x}_f - B\vec{x}_k|||$ pokračujeme jinak, dostaneme

$$|||\vec{x}_f - \vec{x}_{k+1}||| = |||B(\vec{x}_f - \vec{x}_{k+1}) + B(\vec{x}_{k+1} - \vec{x}_k)||| \le |||B||| \cdot |||\vec{x}_f - \vec{x}_{k+1}||| + |||B||| \cdot |||\vec{x}_{k+1} - \vec{x}_k|||.$$

Převedením členu $|||B||| \cdot |||\vec{x}_f - \vec{x}_{k+1}|||$ na levou stranu dostaneme kýžený odhad.

□

Podíváme se na situaci obecně. Označme $\vec{e}_k = \vec{x}_f - \vec{x}_k$, rádi bychom viděli situaci, kdy $\vec{e}_k \to \vec{0}$. Co o této chybě víme? Výpočet v důkazu výše dává v našem značení následující:

$$\vec{e}_{k+1} = \vec{x}_f - \vec{x}_{k+1} = B(\vec{x}_f - \vec{x}_k) = B\vec{e}_k.$$

Indukcí pak dostáváme $\vec{e}_k = B^k\vec{e}_0$. Vidíme, že konvergence naší iterační metody závisí čistě na chování mocnin matice B (tedy nezáleží na počáteční volbě $\vec{x}_0$), potřebujeme $B^k \to 0_{n\times n}$. Na to zná teorie matic odpověď, dokonce přesnou, pozná se to podle spektrálního poloměru. Dostáváme tak následující:

Věta.
Iterační metoda $\vec{x}_{k+1} = B\vec{x}_k + \vec{c}$ konverguje právě tehdy, když $\rho(B) < 1$.

Z praktického pohledu není spektrální poloměr příliš příjemná odpověď, protože hledat vlastní čísla není sranda. Naštěstí máme zajímavý výsledek.

Fakt.
Uvažujme $n \times n$ matici B.
Pro všechny indukované maticové normy platí $\rho(B) \leq |||B|||$.
Naopak pro každé $\varepsilon > 0$ existuje indukovaná maticová norma $||| \cdot |||$ taková, že $|||B||| - \varepsilon < \rho(B)$.

Nás zajímá zejména první tvrzení, které se navíc snadno dokáže. Uvažujme vlastní číslo λ matice B a příslušný vlastní vektor (tedy nenulový) $\vec{x}$. Pak můžeme odhadovat

$$|\lambda| \cdot |||\vec{x}||| = |||\lambda\vec{x}||| = |||B\vec{x}||| \leq |||B||| \cdot |||\vec{x}|||,$$

po zkrácení $|\lambda| \leq |||B|||$.

Vidíme, že spektrální poloměr obsahuje tu nejpřesnější informaci o chování matice, normami se k této informaci lze přiblížit, ale my samozřejmě nevíme, jak blízko naše norma je, pokud nespočítáme spektrální poloměr. Podobně jako u zkoumání stability máme jeden výsledek z pohledu věrohodnosti.

• Jestliže $\rho(B) < 1$, tak pro $\vec{x}_k$ blízké $\vec{x}_f$ v každé iteraci příslušné metody vylepšíme přesnost výsledku o přibližně $-\log_{10}(\rho(B))$ desetinných míst.

Chceme-li řešit soustavu $A\vec{x} = \vec{b}$, tak naše teoretická příprava ukazuje cestu. Budeme tedy hledat takový převod na tvar $\vec{x} = B\vec{x} + \vec{c}$, aby měla výsledná matice B co nejmenší normu.

4a. Jacobiho iterace

Předpokládáme, že máme nějaké hodnoty $\vec{x}_k$ a chceme vytvořit další generaci. Soustava $A\vec{x} = \vec{b}$ nabízí n rovnic, my si z každé z nich vyjádříme jednu proměnnou, nejpřirozenější je z i-té rovnice vyjádřit x_i. Dostáváme předpis, který poslouží jako návod na iteraci:

$$(\vec{x}_{k+1})_i = -\frac{1}{a_{i,i}}\Big(\sum_{j=1}^{i-1} a_{i,j}(\vec{x}_k)_j + \sum_{j=i+1}^{n} a_{i,j}(\vec{x}_k)_j\Big) + \frac{b_i}{a_{i,i}}.$$

Zde předpokládáme, že $a_{i,i} \neq 0$, což připadně zařídíme pivotováním.

Jak by toto vypadalo maticově? $\vec{x}_{k+1} = -D^{-1}(L+U)\vec{x}_k + D^{-1}\vec{b}$, kde D je matice diagonálních prvků z A, v U máme ostře horní trojúhelníkovou část (bez diagonály) matice A a L je její ostře dolní trojúhelníková část, takže $A = D + L + U$.

Dostali jsme schéma správného tvaru s volbou $B_{\mathrm{JIM}} = -D^{-1}(L+U)$ a $\vec{c} = D^{-1}\vec{b}$. Bráno jako rovnice pro pevný bod je tato úloha opravdu ekvivalentní té původní,

$$\vec{x} = -D^{-1}(L+U)\vec{x} + D^{-1}\vec{b} \iff D\vec{x} = -(L+U)\vec{x} + \vec{b} \iff (D+L+U)\vec{x} = \vec{b}.$$

Kdy výpočet ukončíme? Tradičně se zadává mez ε a iterace skončí, pokud $|||\vec{x}_{k+1} - \vec{x}_k||| < \varepsilon$, obvykle se používá maximová norma. Někdy se bere podmínka $\frac{\|\vec{x}_{k+1} - \vec{x}_k\|_\infty}{\|\vec{x}_{k+1}\|_\infty} < \varepsilon$.

Zároveň je časté zadávat maximální povolený počet iterací. Algoritmus pak má dva možné výstupy, buď dá výsledný vektor, nebo zahlásí, že v rámci povoleného počtu iterací se nedospělo k uspokojivému výsledku.

4b. Gauss-Seidelova iterace

Východiskem je Jacobiho iterace. Protože ve chvíli, kdy počítáme $(\vec{x}_{k+1})_i$, již máme nové souřadnice $1, \ldots, i-1$ spočítány, nabízí se nápad použít je k výpočtu, čímž by se procedura mohla urychlit. Vychází vzorec

$$(\vec{x}_{k+1})_i = -\frac{1}{a_{i,i}}\Big(\sum_{j=1}^{i-1} a_{i,j}(\vec{x}_{k+1})_j + \sum_{j=i+1}^{n} a_{i,j}(\vec{x}_k)_j\Big) + \frac{b_i}{a_{i,i}}.$$

Tento iterační vzorec dává postup zvaný **Gauss-Seidelova iterační metoda** (GSM).
Maticově zapsáno to znamená

$$\begin{aligned}\vec{x}_{k+1} &= -D^{-1}(L\vec{x}_{k+1} + U\vec{x}_k) + D^{-1}\vec{b} \implies D\vec{x}_{k+1} = -L\vec{x}_{k+1} - U\vec{x}_k + \vec{b} \\ &\implies D\vec{x}_{k+1} + L\vec{x}_{k+1} = -U\vec{x}_k + \vec{b} \implies (D+L)\vec{x}_{k+1} = -U\vec{x}_k + \vec{b} \\ &\implies \vec{x}_{k+1} = -(D+L)^{-1}U\vec{x}_k + (D+L)^{-1}\vec{b}.\end{aligned}$$

Přepsali jsme to do obecného iteračního tvaru, tedy $B_{\mathrm{GSM}} = -(D+L)^{-1}U$ a $\vec{c} = (D+L)^{-1}\vec{b}$. Řeší to vůbec správnou rovnici?

$$\vec{x} = -(D+L)^{-1}U\vec{x} + (D+L)^{-1}\vec{b} \iff (D+L)\vec{x} = -U\vec{x} + \vec{b} \iff A\vec{x} = \vec{b}.$$

Funguje to.

I zde se tradičně ukončuje podmínkou $|||\vec{x}_{k+1} - \vec{x}_k||| < \varepsilon$ či $\frac{\|\vec{x}_{k+1} - \vec{x}_k\|_\infty}{\|\vec{x}_{k+1}\|_\infty} < \varepsilon$. A také u GSM je vhodné omezit maximální možný počet iterací.

4c. Konvergence JIM a GSM

Nejprve se podíváme na náročnost. Ze vzorců se snadno odvodí, že vytvoření nové generace, tedy vektoru $\vec{x}_{k+1}$, zabere u obou metod stejně, $2n^2 + O(n)$ operací. Toto číslo vypadá velice dobře v porovnání s konkurencí (Gauss, LU-rozklad), ale my tento výpočet musíme opakovat. Významnou roli při rozhodování tedy hraje vztah velikosti matice a rychlosti konvergence iterace.

Citovali jsme větu, podle které konvergence úzce souvisí se spektráním poloměrem matice B. Co víme u našich dvou metod? Pokud $\rho(B_{\mathrm{JIM}}) = 0$ či $\rho(B_{\mathrm{GSM}}) = 0$, pak bychom měli dostat přesné řešení v konečném počtu kroků. Obecný vztah mezi $\rho(B_{\mathrm{GSM}})$ a $\rho(B_{\mathrm{JIM}})$ neexistuje, klidně se může stát, že jedna nekonverguje a druhá ano.
Zároveň se ale pro důležité typy matic se dá obvykle dokázat, že $\rho(B_{\mathrm{GSM}}) < \rho(B_{\mathrm{JIM}})$. Například pokud je matice A třídiagonální (jen diagonála a prvky nalevo a napravo od ní jsou nenulové), pak platí $\rho(B_{\mathrm{GSM}}) = \rho(B_{\mathrm{JIM}})^2$, tedy pokud jde o čísla menší než 1, je GSM výrazně rychlejší. Pro $\vec{x}_k$ blízké $\vec{x}_f$ pak GSM zpřesňuje výsledek dvakrát rychleji než JIM.

Již jsme poznamenali, že hledání vlastních čísel není zrovna snadné (viz níže). Proto se hledají jiné způsoby, jak zajistit, že metoda konverguje.

Definice.
Uvažujeme $n \times n$ matici A.
Řekneme, že A je **ostře diagonálně dominantní**, jestliže pro všechna $i = 1, \ldots, n$ platí

$$|a_{i,i}| > \sum_{j \neq i} |a_{i,j}|.$$

Řekneme, že matice A je **positivně definitní**, jestliže pro všechny nenulové vektory $\vec{x} \in \mathbb{R}^n$ platí $\vec{x}^T A\vec{x} > 0$.

Poznamenejme, že ostře diagonální matice se nám moc líbily již při řešení soustavy rovnic eliminací. Positivní definitnost se pozná například z toho, že všechna vlastní čísla jsou kladná.
Obě vlastnosti se objevují v aplikacích, například matice pro interpolaci pomocí splinů je ostře diagonálně dominantní, matice vzniklá u metody nejmenších čtverců je zase positivně definitní a symetrická.

Věta.
Jestliže je A ostře diagonálně dominantní, pak JIM i GSM konvergují pro libovolnou počáteční volbu vektoru.
Jestliže je A symetrická a positivně definitní, pak GSM konverguje pro libovolnou počáteční volbu vektoru.

Vraťme se k problému výpočetní náročnosti. Pokud je třeba N iterací, pak náročnost celého postupu je $2Nn^2$. Kdy je toto výhodné? Pokud je matice A opravdu velká, tak to může být lepší než tradičních $\frac{1}{3}n^3$ eliminační metody.

4d. Superrelaxační metoda

Podobně jako u iterační metody pro nalézání kořene, i zde se můžeme snažit urychlit konvergenci úpravou matice tak, aby měla menší spektrální poloměr. Zavedeme si parametr $\lambda > 0$ ukazující, jak moc věříme metodě GSM. Nechť $\vec{x}_{k+1}^G$ je vektor, který získáme provedením GSM, my se ale rozhodneme vzít místo toho $\vec{x}_{k+1} = (1-\lambda)\vec{x}_k + \lambda\vec{x}_{k+1}^G$. Evidentně $\lambda = 1$ dává GSM. Vzorec pro iteraci po složkách je

$$(\vec{x}_{k+1})_i = (1-\lambda)(\vec{x}_k)_i - \frac{\lambda}{a_{i,i}}\Big(\sum_{j=1}^{i-1} a_{i,j}(\vec{x}_{k+1})_j + \sum_{j=j+1}^{n} a_{i,j}(\vec{x}_k)_j\Big) + \frac{\lambda b_i}{a_{i,i}}.$$

Maticovou podobu odvodíme přímo z původní rovnice, abychom si zároveň ověřili, že ji nové iterační schéma bude řešit. Vyjdeme z tvaru rovnice vhodného pro odvození GSM:

$$\begin{aligned} A\vec{x} = \vec{b} &\iff \vec{x} = -D^{-1}(L\vec{x} + U\vec{x}) + D^{-1}\vec{b} \iff \lambda\vec{x} = -\lambda D^{-1}(L\vec{x} + U\vec{x}) + \lambda D^{-1}\vec{b} \\ &\iff \vec{x} = (1-\lambda)\vec{x} - \lambda D^{-1}(L\vec{x} + U\vec{x}) + \lambda D^{-1}\vec{b}. \end{aligned}$$

Tím je dáno ono schéma, které upravíme do standardního tvaru.

$$\begin{aligned} \vec{x}_{k+1} &= (1-\lambda)\vec{x}_k - \lambda D^{-1}L\vec{x}_{k+1} - \lambda D^{-1}U\vec{x}_k + \lambda D^{-1}\vec{b} \\ &\implies (E_n + \lambda D^{-1}L)\vec{x}_{k+1} = (1-\lambda)E_n\vec{x}_k - \lambda D^{-1}U\vec{x}_k + \lambda D^{-1}\vec{b} \\ &\implies (D + \lambda L)\vec{x}_{k+1} = (1-\lambda)D\vec{x}_k - \lambda U\vec{x}_k + \lambda\vec{b} \\ &\implies \vec{x}_{k+1} = (D+\lambda L)^{-1}[(1-\lambda)D - \lambda U]\vec{x}_k + \lambda(D+\lambda L)^{-1}\vec{b}. \end{aligned}$$

Dostáváme schéma zvané **superrelaxační metoda** (SOR, Successive OverRelaxation Method). Matice jsou $B_\lambda = (D+\lambda L)^{-1}[(1-\lambda)D - \lambda U]$ a $\vec{c}_\lambda = \lambda(D+\lambda L)^{-1}\vec{b}$.

Věta. (Ostrovský)
Nechť A je symetrická $n \times n$ matice s kladnými prvky na diagonále. Pak $\rho(B_\lambda) < 1$ právě tehdy, když je A positivně definitní a $0 < \lambda < 2$.

Na zvolení ideální volby λ není vzorec (vyjma speciálních případů), odhaduje se empiricky (experimentálně), tudíž se SOR pro jednu soustavu nevyplácí. Pokud ale řešíme soustavu opakovaně, pak máme možnost při každém dalším řešení s λ hýbat a sledovat důsledky, metoda se pak vyplácí.

5. Výpočet vlastních čísel a vektorů

Nejprve si připomeneme základní pojmy.

Definice.
Uvažujme $n \times n$ matici A. Číslo λ je jejím **vlastním číslem**, jestliže existuje nenulový vektor $\vec{x}$ takový, že $A\vec{x} = \lambda\vec{x}$. Tomuto vektoru $\vec{x}$ pak říkáme **vlastní vektor** příslušný k číslu λ.
Jsou-li $\{\lambda_j\}$ všechna (i komplexní) vlastní čísla matice A, pak definujeme její **spektrální poloměr** $\varrho(A) = \max_j(|\lambda_j|)$.

Z teorie rovnic víme, že aby měla rovnice $(A - \lambda E_n)\vec{x} = \vec{0}$ netriviální řešení, musí být matice nalevo singulární a tedy $\det(A - \lambda E_n) = 0$. Determinant dá polynom stupně n, takže každá čtvercová matice má n vlastních čísel (včetně násobnosti). Snadno se také dokáže, že pro jedno vlastní číslo je množina příslušných vlastních vektorů (plus vektor $\vec{0}$) vektorovým prostorem.

Pár faktů:
• Je-li reálná matice symetrická, pak má reálná vlastní čísla.
• Je-li reálná matice diagonalizovatelná, pak existuje báze prostoru $\mathbb{R}^n$ složená z vlastních vektorů.
Podrobněji, diagonalizovatelnou matici lze převést na tvar $A = VDV^{-1}$, kde D je diagonální matice a V je ortogonální matice, tedy její sloupce tvoří ortogonální bázi. Z našeho pohledu je zajímavé, že na diagonále vidíme vlastní čísla λ_j matice A a j-tý sloupec ve V dává odpovídající vlastní vektor, protože $AV = VD$ a matice VD má jako j-tý sloupec vektor λ_j krát j-tý sloupec V.

Pojem symetrie nedává u komplexních matic tu správnou informaci. Namísto transponované matice A^T se proto obvykle pracuje s maticí konjugovanou A^*, což je $(\overline{A})^T$. Pokud je matice reálná, tak samozřejmě $A^* = A^T$. Konjugovat lze i vektory, $\vec{x}^* = \overline{\vec{x}^T}$. Zde je dobře vidět důvod, u komplexních vektorů $\vec{x}^T\vec{x}$ nedává nic zajímavého, ale $\vec{x}^*\vec{x} = \|\vec{x}\|^2$, přesně jak jsme zvyklí z reálného případu.
Matici splňující $A^T = A$ se říká symetrická, podobně matici splňující $A^* = A$ se říká Hermitovská.

Efektivní numerické metody pro hledání vlastních čísel vlastně hledají nikoliv vlastní čísla, ale vlastní vektory. Když tedy máme vlastní vektor $\vec{x}$, jak se pak dostaneme k jeho vlastnímu číslu?

Definice.
Pro $n \times n$ matici A a vektor $\vec{x} \in \mathbb{R}^n$ definujeme **Rayleighův podíl** (Rayleigh quotient) vzorcem

$$\frac{\vec{x}^*A\vec{x}}{\vec{x}^*\vec{x}}.$$

Proč nás to zajímá?

Fakt.
Je-li $\vec{x}$ vlastní vektor matice A, pak je $\frac{\vec{x}^*A\vec{x}}{\vec{x}^*\vec{x}}$ roven příslušnému vlastnímu číslu.

Důkaz je snadný,

$$\frac{\vec{x}^*A\vec{x}}{\vec{x}^*\vec{x}} = \frac{\vec{x}^*\lambda\vec{x}}{\vec{x}^*\vec{x}} = \lambda\frac{\|\vec{x}\|^2}{\|\vec{x}\|^2} = \lambda.$$

Co když náš vektor $\vec{x}$ přesný není? Tak chceme nejlepšího kandidáta. Zformulujme to přesně: Máme vektor $\vec{x}$ a hledáme číslo λ takové, aby byl výraz $\|\lambda\vec{x} - A\vec{x}\|$ co nejmenší. Dá se dokázat, že minimuma dotyčného výrazu se dosáhne právě při volbě $\lambda = \frac{\vec{x}^*A\vec{x}}{\vec{x}^*\vec{x}}$, tedy Raleighův podíl je nejlepším možným odhadem vlastního čísla.
Výrazy v Rayleighově podílu mají své interpretace, $\vec{x}^*\vec{x} = \|\vec{x}\|^2$ a $\vec{x}^*A\vec{x}$ je klasická kvadratická forma daná maticí A, viz třeba positivní definitnost. Raleighův podíl má i zajímavé teoretické souvislosti. Pokud si u Hermitovské matice A srovnáme vlastní čísla od největšího λ_1 po nejmenší λ_n, pak $\vec{x} \mapsto \frac{\vec{x}^*A\vec{x}}{\vec{x}^*\vec{x}}$ má hodnoty v rozmezí λ_n až λ_1, přičemž maxima i minima se nabývá (příslušnými vlastními vektory).

5a. Metoda mocnin (power method)

Nechť $\{\vec{v}_j\}$ je báze sestavená z vlastních vektorů příslušejícím vlastním číslům λ_j. Uvažujme vektor $\vec{x}_0 = \sum c_j\vec{v}_j$. Pak můžeme pomocí indukce ukázat, že

$$A^k\vec{x}_0 = \sum c_j\lambda_j^k\vec{v}_j = \lambda_1^k\sum c_j\frac{\lambda_j^k}{\lambda_1^k}\vec{v}_j \implies \frac{A^k\vec{x}_0}{\lambda_1^k} = \sum\Big(\frac{\lambda_j}{\lambda_1}\Big)^k c_j\vec{v}_j.$$

Jestliže platí že $|\lambda_1| > |\lambda_j|$ pro $j \geq 2$, pak $\frac{A^k\vec{x}_0}{\lambda_1^k} \to c_1\vec{v}_1$, což je také vlastní vektor odpovídající λ_1. To vypadá jako zajímavý nápad na iteraci $\vec{x}_{k+1} = \frac{A\vec{x}_k}{\lambda_1}$, my ovšem λ_1 neznáme.
V praxi se to řeší tak, že vektory $A^k\vec{x}_0$ normujeme, nejčastěji maximovou normou.

Algoritmus (metoda mocnin pro výpočet vlastního vektoru)**.**
Zadána $n \times n$ matice A a tolerance $\varepsilon > 0$.
0. Zvolíme libovolný nenulový vektor $\vec{x}_0$.
Chceme-li komplexní vlastní čísla, musíme tento vektor zvolit tak, aby každá jeho složka měla nenulovou imaginární část.
1. Definujeme

$$\vec{x}_{k+1} = \frac{A\vec{x}_k}{\|A\vec{x}_k\|_\infty},$$

Pokud $\|\vec{x}_{k+1} - \vec{x}_k\|_\infty \geq \varepsilon$, zvýšíme k o jedničku a jdeme zpět na krok **1**.
△

Uvedli jsme nejobvyklejší podmínku ukončení, často se také používá chyba relativní, tedy čeká se na $\frac{\|\vec{x}_{k+1}-\vec{x}_k\|_\infty}{\|\vec{x}_k\|_\infty} < \varepsilon$. I zde se doporučuje zadat maximální povolený počet iterací.

Věta.
Nechť A je $n \times n$ matice s vlastními čísly λ_j, přičemž $|\lambda_1| > |\lambda_j|$ pro $j \geq 2$. Nechť $M = \max_{j\geq 2} |\lambda_j|$.
Jestliže posloupnost konverguje $\{\vec{x}_k\}$ generovaná metodou mocnin konverguje k nějakému vektoru $\vec{v}$, tak je $\vec{v}$ vlastní vektor odpovídající λ_1 a $\|\vec{v} - \vec{x}_{k+1}\| \leq \frac{M}{|\lambda_1|}\|\vec{v} - \vec{x}_k\|$.

Vidíme, že $\|\vec{e}_{k+1}\| \leq \frac{M}{|\lambda_1|}\|\vec{e}_k\|$ neboli rychlost konvergence metody je lineární, rychlost tudíž závisí na tom, jak významně je λ_1 oddělena od ostatních vlastních čísel. Příslušné vlastní číslo λ_1 pak získáme Rayleighovým podílem.

Jak je to ale s konvergencí? Není težké se přesvědčit, že v případě splnění podmínky $|\lambda_1| > |\lambda_j|$ pro $j \geq 2$ dostáváme konvergenci, pokud je $\lambda_1 > 0$, ale jinak ne.

Věta.
Nechť A je $n \times n$ matice s vlastními čísly λ_j, přičemž $|\lambda_1| > |\lambda_j|$ pro $j \geq 2$. Nechť $M = \max_{j\geq 2} |\lambda_j|$.
Jestliže $\{\vec{x}_k\}$ je posloupnost generovaná metodou mocnin, tak existují $\phi_j \in \mathbb{R}$, $\vec{r}_k \in \mathbb{R}^k$ a vlastní vektor $\vec{v}$ příslušný λ_1 takové, že $\vec{x}_k = e^{i\phi_k}\vec{v} + \vec{r}_k$ a $\vec{r}_k \to 0$.

Vlastně je to dobrá zpráva, protože pokud si s vektorů $\vec{x}_k$ budeme pomocí Rayleighova podílu dopočítávat průběžně kandidáty $l_k = \frac{\vec{x}_k^* A\vec{x}_k}{\vec{x}_k^*\vec{x}_k}$, tak už $l_k \to \lambda_1$.

Vše jde ovšem ještě vylepšit. Naše úvahy ukazují, že při volbě $\vec{x}_{k+1} = \frac{1}{\lambda_1}A\vec{x}_k$ bychom měli dostat posloupnost konvergující k $\vec{v}$, a odhady λ_1 dokážeme průběžně dopočítávat. Tím získáme lepší algoritmus. Pořád se vyplácí vektory normovat, což se dá upravit na rozumný tvar.

Algoritmus (metoda mocnin pro výpočet největšího vlastního čísla a příslušného vlastního vektoru)**.**
Zadána $n \times n$ matice A a tolerance $\varepsilon > 0$.
0. Zvolíme libovolný nenulový vektor $\vec{x}_0$.
Chceme-li komplexní vlastní čísla, musíme tento vektor zvolit tak, aby každá jeho složka měla nenulovou imaginární část.
Spočítáme $l_0 = \frac{\vec{x}_0^* A\vec{x}_0}{\vec{x}_0^*\vec{x}_0}$.
1. Definujeme

$$\vec{x}_{k+1} = \frac{|l_k| \cdot A\vec{x}_k}{l_k \cdot \|A\vec{x}_k\|_\infty}, \qquad l_{k+1} = \frac{\vec{x}_{k+1}^* A\vec{x}_{k+1}}{\vec{x}_{k+1}^*\vec{x}_{k+1}}.$$

Pokud $\|\vec{x}_{k+1} - \vec{x}_k\|_\infty \geq \varepsilon$, zvýšíme k o jedničku a jdeme zpět na krok **1**.
△

Alternativa: Použije se Euklidovská norma, tedy

$$\vec{x}_{k+1} = \frac{A\vec{x}_k}{\|A\vec{x}_k\|}, \qquad l_{k+1} = \vec{x}_{k+1}^* A\vec{x}_{k+1}.$$

Věta.
Nechť A je $n \times n$ matice s vlastními čísly λ_j, přičemž $|\lambda_1| > |\lambda_j|$ pro $j \geq 2$. Nechť $M = \max_{j\geq 2} |\lambda_j|$.
Pro $\vec{x}_0$ nechť $\{l_k\}$, $\{\vec{x}_k\}$ jsou posloupnosti generované modifikovanou metodou mocnin.
Pak $\{l_k\}$ konverguje k λ_1 a $\{\vec{x}_k\}$ konverguje k nějakému vlastnímu vektoru $\vec{v}$ příslušnému λ_1, přičemž $|\lambda_1 - l_k| = O\big(\big[\frac{M}{|\lambda_1|}\big]^{2k}\big)$ a $\|\vec{v}_1 - \vec{x}_k\| = O\big(\big[\frac{M}{|\lambda_1|}\big]^{k}\big)$.

Vztahy z věty se dají přibližně přepsat na $|E(\vec{v})_k| \sim C \cdot \big(\frac{M}{|\lambda_1|}\big)^k$ a $|E(\lambda)_k| \sim C \cdot \big(\frac{M}{|\lambda_1|}\big)^{2k}$. To pak znamená, že $|E(\vec{v})_{k+1}| \sim \frac{M}{|\lambda_1|}|E(\vec{v})_k|$ a $|E(\lambda)_{k+1}| \sim \big(\frac{M}{|\lambda_1|}\big)^2 |E(\lambda)_k|$. Vidíme lineární řád konvergence, přičemž posloupnost pro λ má o něco lepší konstantu.

Výhody: Je to jednoduchý algoritmus.

Nevýhody: Když už konverguje, tak je to typicky dost pomalé. Najde jen největší vlastní hodnotu. Zcela selhává, pokud je největší vlastní hodnota vícenásobná či dosažena více vlastními čísly (třeba když má 2×2 matice vlastní čísla ± 1).

Inverzní mocninná iterace (Inverse power method).

Klíčem je posun spektra neboli vlastních hodnot.

Fakt.
Nechť A je $n \times n$ matice, $\mu \in \mathbb{R}$. Pak platí:
(i) λ je vlastní číslo matice A s vlastním vektorem $\vec{v}$ právě tehdy, je-li $\lambda - \mu$ vlastní číslo matice $A - \mu E_n$ s vlastním vektorem $\vec{v}$.
(ii) Nechť je A regulární. λ je vlastní číslo matice A s vlastním vektorem $\vec{v}$ právě tehdy, je-li $\frac{1}{\lambda}$ vlastní číslo matice A^{-1} s vlastním vektorem $\vec{v}$.

Důkaz: Stačí upravovat:
(i) $A\vec{v} = \lambda\vec{v} \iff A\vec{v} - \mu\vec{v} = \lambda\vec{v} - \mu\vec{v} \iff A\vec{v} - \mu E_n\vec{v} = \lambda\vec{v} - \mu\vec{v} \iff (A - \mu E_n)\vec{v} = (\lambda - \mu)\vec{v}$.
(ii) $A\vec{v} = \lambda\vec{v} \iff A^{-1}A\vec{v} = A^{-1}\lambda\vec{v} \iff \vec{v} = \lambda A^{-1}\vec{v} \iff \frac{1}{\lambda}\vec{v} = A^{-1}\vec{v}$.
□

Vztahy z Faktu říkají, že λ je vlastní číslo matice A právě tehdy, když je $\frac{1}{\lambda-\mu}$ vlastním číslem matice $(A - \mu E_n)^{-1}$, se shodným vlastním vektorem.

Metoda mocnin aplikovaná na $(A - \mu E_n)^{-1}$ dá číslo α, které je největším mezi čísly $\frac{1}{\lambda-\mu}$, tedy $\frac{1}{\alpha}$ je nejmenším mezi čísly $\lambda - \mu$, tedy $\frac{1}{\alpha} + \mu$ je nejbližší vlastní číslo k μ.

Algoritmus (inverzní metoda mocnin pro výpočet vlastního čísla a příslušného vlastního vektoru)**.**
Zadána $n \times n$ matice A a tolerance $\varepsilon > 0$.
0. Zvolíme libovolné reálné číslo μ, které není vlastní číslo A.
Zvolíme libovolný nenulový vektor $\vec{x}_0$.
Chceme-li komplexní vlastní čísla, musíme tento vektor zvolit tak, aby každá jeho složka měla nenulovou imaginární část.
1. Najdeme $B = (A - \mu E_n)^{-1}$.
Metodou mocinné iterace najdeme největší vlastní číslo λ_B matice B a příslušný vlastní vektor $\vec{v}$.
2. Číslo $\mu + \frac{1}{\lambda_B}$ je vlastním číslem matice A s vlastním vektorem $\vec{v}$.
Je to nejbližší vlastní číslo A k číslu μ.
△

Výhoda: Je možné se takto dostat i k jiným vlastním číslům.

Nevýhoda: Je to pracné (nutnost počítat inverzi), ale vlastně ji stačí spočítat jen jednou. Mnohem větší problém je, že nevíme, jak zvolit μ, abychom odchytali další vlastní čísla. Volba $\mu = 0$ najde nejmenší vlastní číslo, dál už to tak jasné není. Prostě zkoušíme.

Rayleighova podílová iterace (Rayleigh quotient iteration method).

Algoritmus konverguje tím rychleji, čím je μ blíže k hledanému λ a dále od ostatních vlastních čísel. My ovšem v průběhu algoritmu teprve zjišťujeme, kde tušit nějaké vlastní číslo. Proč tohoto tušení tedy nevyužít a neměnit s dynamicky, například použít nejlepší možný odhad l_k? Vychází pak tato iterace:

$$l_k = \frac{(\vec{x}_k)^* A\vec{x}_k}{\vec{x}_k^*\vec{x}_k}, \qquad \vec{x}_{k+1} = \frac{(A - l_k E_n)^{-1}\vec{x}_k}{\|(A - l_k E_n)^{-1}\vec{x}_k\|_\infty}.$$

Opět je možné výpočet dále zjednodušit tím, že normujeme pomocí normy $\|\cdot\|_2$ namísto maximové.

Algoritmus (Rayleighova iterace pro výpočet vlastního čísla a příslušného vlastního vektoru)**.**
Zadána $n \times n$ matice A a tolerance $\varepsilon > 0$.
0. Zvolíme libovolný nenulový vektor $\vec{x}_0$ tak, aby $\|\vec{x}_0\| = 1$.
Chceme-li komplexní vlastní čísla, musíme tento vektor zvolit tak, aby každá jeho složka měla nenulovou imaginární část.
1. Definujeme

$$l_k = (\vec{x}_k)^* A\vec{x}_k, \qquad \vec{y} \text{ řeší } (A - l_k E_n)\vec{y} = \vec{x}_k, \qquad \vec{x}_{k+1} = \frac{\vec{y}}{\|\vec{y}\|}.$$

Pokud $\|\vec{x}_{k+1} - \vec{x}_k\|_\infty \geq \varepsilon$, zvýšíme k o jedničku a jdeme zpět na krok **1**.
△

Takovýto algoritmus již má rychlejší konvergenci.

Věta.
Nechť A je $n \times n$ reálná symetrická matice s vlastními čísly λ_j.
Pak pro skoro všechny (s výjimkou množiny míry nula) jednotkové vektory $\vec{x}_0$ bude Rayleighova podílová iterace konvergovat k nějakému vlastnímu číslu λ_j a vlastnímu vektoru $\vec{v}_j$. V takovém případě pro $\vec{x}_k$ blízko $\vec{v}_j$ platí $\|\vec{v}_j - \vec{x}_{k+1}\| = O(\|\vec{v}_j - \vec{x}_k\|^3)$ a $|\lambda_j - l_{k+1}| = O(|\lambda_1 - l_k|^2)$.

Nevýhoda je ovšem zásadní, v každém kroku musíme počítat novou inverzní matici. To se dá přepsat na řešení rovnice $(A - \lambda_k E_n)\vec{y} = \vec{x}_k$, ale i tak koukáme na $O(n^3)$. Pro supervelké matice by se asi dala použít iterační metoda řešení soustav (iterace v iteraci!), ale stejně to je pořád komplikace ve srovnání s inverzní mocninnou iterací.

To se ovšem změní, pokud má matice A nějakou strukturu řídkosti. Pokud bychom například věděli, že má jen tři nenulové diagonální řady (je tzv. třídiagonální, což platí pro spoustu matic pocházejících z numerických řešení diferenciálních rovnic), můžeme navrhnout speciální procedury pro výpočet inverzní matice s náročností klesající až k $O(n)$. Pak už Rayleighova iterace začne vypadat zajímavěji.

Algoritmus se snaží dostat k vlastnímu vektoru $\vec{v}_j$ nejbližšímu počáteční volbě $\vec{x}_0$. Takto se dají postupně vychytávat různé vlastní podprostory a tedy i různá vlastní čísla, zkoušíme volit $\vec{x}_0$ kolmé k již nalezeným vlastním vektorům.

Deflation.

Trochu jiný způsob jak nalézat různá vlastní čísla funguje takto: Uvažujme matici A s reálnými vlastními čísly λ_j seřazenými tak, že $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Nejprve iterační metodou najdeme λ_1 a $\vec{v}_1$.

Předpokládejme dále, že vlastní vektory tvoří ortogonální bázi, po přenormování bázi ortonormální, tedy $\|\vec{v}_i\| = 1$. Můžeme si například představit, že A je reálná symetrická matice. Pro libovolný vektor $\vec{x}$ pak máme rozklad $\vec{x} = \sum c_i\vec{v}_i$ a také $A\vec{x} = \sum \lambda_i c_i \vec{v}_i$. Vidíme, že pokud teď zvolíme $\vec{x}_0$ kolmý na $\vec{v}_1$, tak $c_1 = 0$ a při iteraci budou rovněž všechny $\vec{x}_k$ kolmé na $\vec{v}_1$, algoritmus tedy najde λ_2, největší zbývající vlastní číslo.

V praxi bývá jednodušší volit $\vec{x}$ dle libosti a pak odstranit složku ve směru $\vec{v}_1$, tedy použít projekci pomocí skalárního součinu a brát $\vec{x}_0 = \vec{x} - \frac{(\vec{v}_1 \bullet \vec{x})}{\|\vec{v}_1\|^2}\vec{v}_1$. Díky normalizaci je to $\vec{x}_0 = \vec{x} - (\vec{v}_1 \bullet \vec{x})\vec{v}_1$.

Bohužel, kvůli zaokrouhlovacím chybám se složka $\vec{v}_1$ může zase do vektorů vetřít, je tedy nutné ji takto odřezávat v každém kroku. Z vektoru $\vec{x}_k$ tedy spočítáme $\vec{z} = A\vec{x}_k$, posléze $\vec{y} = \vec{z} - (\vec{v}_1 \bullet \vec{z})\vec{v}_1$ a nakonec $\vec{x}_{k+1} = \frac{\vec{y}}{\|\vec{y}\|}$.

Abychom to zjednodušili, podíváme se na to blíže. Když vyjádříme $\vec{x}_k = \sum c_i \vec{v}_i$, dostaneme

$$\vec{z} = \sum \lambda_i c_i \vec{v}_i \implies \vec{v}_1 \bullet \vec{z} = \lambda_1 c_1 \implies$$
$$\vec{y} = A\vec{x}_k - \lambda_1 c_1 \vec{v}_1 = A\vec{x}_k - \lambda_1 (\vec{v}_1 \bullet \vec{x}_k)\vec{v}_1.$$

Akci A a odstranění složky ve směru $\vec{v}_1$ lze tedy udělat v jednom kroku.

Toto se snadno zobecní na případ, že už známe více párů vlastních čísel a vektorů.

Algoritmus (deflační metoda pro výpočet vlastního čísla a vektoru u symetrické matice)**.**
Předpokládáme, že již známe vlastní čísla λ_1 až λ_N matice A s vlastními vektory $\vec{v}_1$ až $\vec{v}_N$ tvořícími ortonormální množinu.
0. Zvolíme $\vec{x}_0$.
1. Nechť $\vec{y} = \vec{x}_k - \sum_{j=1}^{N} \lambda_j (\vec{v}_j \bullet \vec{x}_k)\vec{v}_j$. Definujeme $\vec{x}_{k+1} = \frac{A\vec{y}}{\|A\vec{y}\|}$, $l_{k+1} = \vec{x}_{k+1}^* A \vec{x}_{k+1}$.
Pokud $\|\vec{x}_{k+1} - \vec{x}_k\|_\infty \geq \varepsilon$, zvýšíme k o jedničku a jdeme zpět na krok **1**.
△

Pak $\lambda_k \to \lambda_{N+1}$, $\vec{x}_k \to \vec{v}_{N+1}$. Je to relativně jednoduchá metoda dávající přesně to, co po ní chceme. Bohužel ne všechny matice jsou symetrické a ne všechny symetrické matice mají vlastní čísla různě daleko od počátku, jak je zde potřeba. Dobrá otázka tedy je, jak si poradíme v obecnějším případě. V numerické matematice se pak používají mocnější metody založené na pokročilejších partiích lineární algebry.

5b. Jacobiho iterační metoda

Geometricky, rozklad $A = VDV^{-1}$ znamená, že prostor pomocí rotace dané V natočíme tak, aby zobrazení dané maticí A mělo vůči nové bázi diagonální matici. Cílem je tedy utlumit (totálně) prvky mimo diagonálu. Hlavní myšlenkou Jacobiho iterace je rozložit toto utlumení na více dílčích kroků.

Základním nástrojem je „Givensova rotační matice“, která pro zvolené i, j realizuje rotaci prostoru v rovině dané vektory $\vec{e}_i, \vec{e}_j$ (tedy kolem osy, která je na tuto rovinu kolmá) o úhel θ proti směru hodinových ručiček. Tato matice $G(i, j, \theta)$ má (pro $i > j$) následující strukturu: Označíme-li $s = \sin(\theta)$ a $c = \cos(\theta)$, pak

$$g_{i,j} = s$$
$$g_{j,i} = -s$$
$$g_{i,i} = g_{j,j} = c,$$
$$g_{k,k} = 1 \text{ pro } k \neq i, j,$$
$$g_{k,l} = 0 \text{ jinak.}$$

V případě $i < j$ se prohodí znaménko u sinů. V aplikacích nás často ani nezajímá, kolik je úhel, takže se prostě pracuje s čísly s a c takovými, že $s^2 + c^2 = 1$, a prohazování znamének se nemusí řešit. Je také snadné nahlédnout, že matice G^{-1} vznikne z G právě změnou znamének u s.

Výhoda této matice/rotace je, že pokud takto zrotujeme prostor, tak to při změně z matice A na matici GA ovlivní jen dva řádky. Je snadné spočítat přesně, jakou matici G je třeba zvolit, aby se v GA vynuloval přesně prvek na pozici (i, j). My budeme chtít, aby se vynulovaly ve výrazu $G^{-1}AG$ prvky na souřadnicích (i, j) i (j, i). Proč jsme dali inverzi k prvnímu G a ne druhému? Pokud dosáhneme nakonec $V^{-1}AV = D$, dá to $A = VDV^{-1}$, přesně jak potřebujeme.

Požadavek tedy je, aby (při $i < j$)

$$-s[ca_{i,i} + sa_{j,i}] + c[ca_{i,j} + sa_{j,j}] = 0 \implies -sca_{i,i} - s^2 a_{j,i} + c^2 a_{i,j} + sca_{j,j} = 0.$$

Pomocí symetrie matice A se toto zjednoduší na $sc(a_{i,i} - a_{j,j}) + (s^2 - c^2)a_{i,j} = 0$. Trik: Označíme $\beta = \frac{a_{i,i}-a_{j,j}}{2a_{i,j}}$, pak máme rovnici $s^2 - c^2 + 2\beta sc = 0$ neboli $(2s^2 - c^2)^2 = \beta^2 s^2(1 - s^2)$. Vyřešíme pro s^2, dopočítáme c^2, poté je třeba si pohlídat znaménka a vyjde

$$s = \sqrt{\frac{1}{2} - \frac{\beta}{2}\sqrt{1+\beta^2}}, \quad c = \sqrt{\frac{1}{2} + \frac{\beta}{2}\sqrt{1+\beta^2}}.$$

Umíme tedy vynulovat jeden nediagonální prvek v matici A, problém je, že takto nedostaneme konečný algoritmus, protože při nulování jiných si tento zase pokazíme.

Mnohem lepší strategie tedy je postupně přesouvat velikosti prvků směrem k diagonále, ty již zmenšené se sice mohou zase zvětšit, ale při vhodně zvolené strategii to nebude moc. Vypovídá o tom tato věta:

Věta.
Nechť $B = G^{-1}AG$, kde $G = G(i, j, \theta)$. Pak

$$\sum_{i=1}^{n} b_{i,i}^2 = \sum_{i=1}^{n} a_{i,i}^2 + 2a_{i,j}^2 \text{ a } \sum_{i,j=1}^{n} b_{i,j}^2 = \sum_{i,j=1}^{n} a_{i,j}^2.$$

Vidíme tedy, že s každou takovouto iterací se suma diagonály zvětší a celková nezmění, tudíž suma prvků mimo diagonálu se musí zmenšovat. My se samozřejmě snažíme o co největší změnu, takže budeme vždy likvidovat největší nediagonální prvek.

Algoritmus (Jordan iteration method)**.**
0. $A_0 = A$.
1. Nechť $a_{i,j}$ je takový prvek, že $|a_{i,j}| = \max\{|a_{l,m}|;\ l \neq m\}$.
Označme $\beta = \frac{a_{i,i}-a_{j,j}}{2a_{i,j}}$, nechť G je Givensova $G(i, j, \theta)$ matice s konstantami $s = \sqrt{\frac{1}{2} - \frac{\beta}{2}\sqrt{1+\beta^2}}$, $c = \sqrt{\frac{1}{2} + \frac{\beta}{2}\sqrt{1+\beta^2}}$.
Definujeme $A_{k+1} = G^{-1}A_kG$.
Jestliže nejsme spokojeni, zvýšíme k o jedničku a jdeme zpět na krok **1**.
△

Pak $A_k \to D$, dále $G_1G_2 \cdots G_k \to V$.
Protože jsou matice G ortogonální, chyby se nenásobí, jde tedy o numericky stabilní proces.