

1 Omezení počítače: čas a paměť

Úloha 1. Mějme čtvercovou matici řádu n a vektor s n prvky. Spočítejte, kolik operací násobení a sčítání stojí operace násobení matice vektorem.

Úloha 2. Pomocí příkazů `tic` a `toc` (start a konec stopek) změřte násobení náhodné čtvercové matice vektorem s řádem $n = 100, 1000, 10\,000$. S využitím výpočtu v předchozí úloze přepočítejte výsledek jako počet elementárních operací (flopů) za vteřinu.

Úloha 3. Vyzkoušejte, jakou největší náhodnou čtvercovou matici dokážete v MATLABu vytvořit (a uložit). Pomocí příkazů `whos` zjistěte, kolik paměti zabírá.

2 LU rozklad

2.1 Opakování a LU rozklad v MATLABu

LU rozklad je maticový zápis Gaussovy eliminace, tedy převodu matice do odstupňovaného tvaru. Označíme-li E_k matici elementární transformace (vynásobení řádku nenulovým číslem, prohození dvou řádků, nebo přičtení násobku řádku k jinému), pak lze Gaussovu eliminaci schematicky zapsat jako

$$A \rightarrow A^{(1)} = E_1 A \rightarrow \dots \rightarrow U = A^{(n-1)} = \overbrace{E_{n-1} \dots E_1}^{L^{-1}} A, \quad (1)$$

kde U je matice v odstupňovaném tvaru a platí $LU = A$. (Připomínáme, že matice elementárních úprav jsou regulární.)

Pro tzv. silně regulární matice nepotřebujeme v Gaussově eliminaci prohazovat řádky. Pak je matice L dolní trojúhelníková, což vysvětluje název rozkladu (Lower–Upper). To, jestli je matice silně regulární, ale bohužel obvykle dopředu nevíme.

Úloha 4. Pro vestavěnou funkci `[L,U] = lu(A)`:

- Zvolte `A = wilkinson(6)` a spočítejte její LU rozklad. Ověřte, že U je horní trojúhelníková. Je L dolní trojúhelníková?
- Prohodte u matice A třetí a čtvrtý sloupec a zopakujte.
- (navíc) nastudujte si v nápovědě variantu volání vestavěné funkce jako `[L,U,P] = lu(A)`. Čemu odpovídá matice P ? Jak se změní vlastnosti L oproti `[L,U] = lu(A)`?

2.2 Řídké matice a LU rozklad

V řadě aplikací dostáváme matice, které mají na většině pozic nuly. Těmto maticím říkáme *řídké* a obvykle u nich ukládáme pouze nenulové prvky. (Matice, které nejsou řídké, označujeme jako *husté*.)

Úloha 5. • Vytvořte řídkou matici příkazem `B = gallery('poisson',5)`; Jaký má rozměr? Jaký je rozdíl zadáte-li do příkazové řádky bez středníku `A` (z předchozí úlohy), resp. `B`?

- Pomocí příkazu `whos` srovnajte paměťové nároky na uložení řídké matice `B = gallery('poisson',50)` a náhodné (husté) matice stejného řádu.

- (navíc) Jakou největší matici lze pomocí $B = \text{gallery}('poisson', N)$; sestavit a uložit? Odhadněte na základě několika voleb parametru N a ověřte.

Úloha 6. Zkonstruuje matici $B = \text{gallery}('poisson', 50)$; a matici C , která vznikne přidáním sloupcového vektoru samých jedniček (vhodné délky) zleva k matici B . Provedte pro obě LU rozklad. Pomocí příkazu `whos` srovnajte paměťové nároky pro matice B a C a pro jejich LU faktory. Pomocí příkazu `spy` srovnajte zaplnění (tj. počet nenulových prvků) faktorů L pro matice B a C .

Úloha 7. Zkonstruuje matici $B = \text{gallery}('poisson', 100)$; a spočítejte její LU rozklad. Jak dlouho to trvá? Kolik paměti faktory L a U zabírají? Zvládne MATLAB spočítat LU rozklad matice $B = \text{gallery}('poisson', 1000)$;? **Varování:** Výpočet tohoto rozkladu vám na několik minut v podstatě zastaví počítač. A pak to stejně skončí s chybovou hláškou "Out of memory".

3 Singulární rozklad a komprese dat

Nechť $A \in \mathbb{C}^{n \times m}$, $\text{rank}(A) = r$, pak matici A můžeme zapsat ve tvaru *singulárního rozkladu* (SVD):

$$A = U \Sigma V^*,$$

kde $U \in \mathbb{C}^{n \times n}$ a $V \in \mathbb{C}^{m \times m}$ jsou unitární matice a

$$\Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times m}, \quad \Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{R}^{r \times r},$$

$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$. Schematicky (včetně takzvaného ekonomického tvaru):

$$\begin{array}{c} A \\ \boxed{} \end{array} = \begin{array}{c} U \\ \boxed{} \end{array} \begin{array}{c} \Sigma \\ \boxed{} \end{array} \begin{array}{c} V^* \\ \boxed{} \end{array} = \begin{array}{c} U_r \\ \boxed{} \end{array} \begin{array}{c} \Sigma_r \\ \boxed{} \end{array} \begin{array}{c} V_r^* \\ \boxed{} \end{array}.$$

Díky singulárnímu rozkladu lze matici zapsat v tzv. *dyadickém rozvoji*

$$A = \sum_{j=1}^r \sigma_j u_j v_j^* = \sum_{j=1}^r A_j, \quad A_j \equiv \sigma_j u_j v_j^*.$$

Věta 1 (Eckart–Young–Mirsky). Nechť $A \in \mathbb{C}^{n \times m}$, $r := \text{rank } A$, a nechť $k < r$. Potom

$$\underset{\substack{X \in \mathbb{C}^{n \times m} \\ \text{rank}(X) \leq k}}{\text{argmin}} \|A - X\| = A^{(k)} \equiv \sum_{j=1}^k \sigma_j u_j v_j^*$$

a platí

$$\|A - A^{(k)}\| = \sigma_{k+1}.$$

Rovněž platí i

$$\underset{\substack{X \in \mathbb{C}^{n \times m} \\ \text{rank}(X) \leq k}}{\text{argmin}} \|A - X\|_F = A^{(k)}, \quad \|A - A^{(k)}\|_F^2 = \sum_{j=k+1}^r \sigma_j^2.$$

Úloha 8. Singulární rozklad a výše uvedenou větu využijeme ke kompresi obrazu.

1. Pomocí příkazu `doc` zjistíme informace o funkci `imread`. Zajímá nás především vstup a výstup. Vše důležité najdeme v dokumentaci v kapitole Examples.
2. Uložme si obrázek `beer.bmp` jako proměnnou typu `double`.
3. Použijme příkaz `svd` a výsledky si uložme. Pokud neznáme výstup funkce `svd`, můžeme opět použít příkaz `help`.
4. Nyní máme spočtený SVD rozklad. Můžeme se podívat na vlastnosti tohoto rozkladu. Pomocí příkazu `semilogy` můžeme vykreslit singulární čísla. Pokud nás zajímá chyba rozkladu, můžeme použít naši funkci `plotSVD`. U obou funkcí je dobré si zjistit podobu vstupu.
5. Vypočítejme matici pouze s polovinou (přibližně polovinou) singulárního tripletu.
6. Pomocí příkazů `imagesc` a `colormap` vykresleme zkomprimovaný snímek v černobíle škále.
7. Porovnejme obrázek s původním obrázkem tak, že pomocí příkazu `figure(3)` otevřeme třetí okno s obrázkem a opět pomocí příkazů `imagesc` a `colormap` vykreslíme původní obrázek.
8. Vyzkoušejme si vlastní volbu singulárního tripletu nebo si nahrajme vlastní obrázky a experimentujme.
9. Pomocí dema `runme.m` můžeme sledovat změnu velikosti obrázku a snižování kvality v závislosti na volbě singulárního tripletu.

Seznam připravených skriptů a funkcí

- `plotSVD.m`: Výpočet kvality SVD metody.
- `runme.m`: Skript na pozorování ztráty kvality obrazu.