Cvičení 3 - zadání a řešení úloh

Základy numerické matematiky - NMNM201 Verze z 5. října 2023

---

1 Omezení počítače: čas a paměť

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

Řešení. Potřebujeme celkem $2n^2 - n$ operací. (Výsledek má n prvků a každý spočítáme díky n násobení a $n-1$ sčítání.) □

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 přepočtěte výsledek jako počet elementárních operací (*flopů*) za vteřinu.*

Řešení.
```
n_all = [100 1000 10000];
for n = n_all
   A = rand(n);
   b = rand(n,1);
   tic;
   A*b;
   time = toc;
   flops = 2*n*n-n;
   flops/time
end
```
□

Současný nejvýkonnější superpočítač dosahuje teoretického výkonu přibližně 10^{18} flopů za vteřinu. I tak to pro některé aplikace nestačí.

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

Řešení. Tato velikost závisí na paměti RAM počítače a nastavení MATLABu (u mě to bylo například matice řádu 32 272, která zabrala 7.75 GB). Lze zjistit například pomocí metody bisekce. □

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

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

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?*
- *Prohoďte 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)`*?*

Řešení. V prvním případě L dolní trojúhelníková není.

Pro `[L,U] = lu(A(:,[1 2 4 3 5 6]))` už L dolní trojúhelníková je.

`[L,U,P] = lu(A)` odpovídá tomu, že elementární operace prohození řádků jsou uloženy v matici P, matice L je pak vždy dolní trojúhelníková. □

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` *srovnejte paměťové nároky na uložení řídké matice* `B = gallery('poisson',50);` *a náhodné (husté) matice stejného řádu.*
- *(navíc) Jakou největší matici lze pomocí* `B = gallery('poisson',N);` *sestavit a uložit? Odhadněte na základě několika voleb parametru* `N` *a ověřte.*

Řešení.

- U řídkých matic MATLAB vypisuje pozici nenulového prvku a jeho hodnotu; husté matice vypisuje po řádcích.
- Je to necelých 250 kB vs skoro 48 MB.

□

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

Řešení.
```
C = [ones(2500,1),B];
[LB,UB] = lu(B);
[LC,UC] = lu(C);
spy(LB);
spy(LC);
```
LC je plná dolní trojúhelníková matice s 3 126 250 nenulovými prvky, LB jich má jen 125 049. I tak je to víc než desetkrát více, než kolik je nenul v matici B! □

Úloha 7. *Zkonstruujte matici* `B = 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 = 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

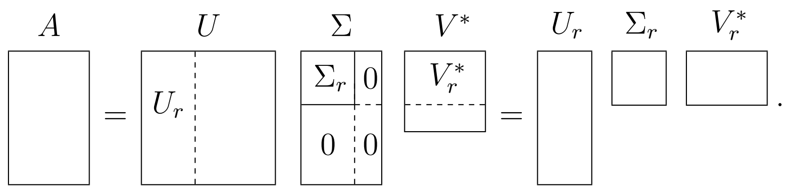
Nechť $A \in \mathbb{C}^{n\times m}$, $\operatorname{rank}(A) = r$, pak matici A můžeme zapsat ve tvaru *singulárního rozkladu* (SVD):

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

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

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

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



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 $A \in \mathbb{C}^{n\times m}$, $r := \operatorname{rank} A$, a nechť $k < r$. Potom*

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

a platí

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

Rovněž platí i

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

Pozn.: Tuto větu včetně důkazu lze nalézt ve skriptech LA, sekce 10.4.3.

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

1. *Pomocí příkazu* `doc` *zjistěme informace o funkci* `imread`*. Zajímá nás především vstup a výstup. Vše důležité najdeme v dokumentaci v kapitole* Examples.

   Řešení. `doc imread` □

2. *Uložme si obrázek* `beer.bmp` *jako proměnnou typu* `double`.

   Řešení. `A = double(imread('beer.bmp'));` □

3. *Použijme příkaz* `svd` *a výsledky si uložme. Pokud neznáme výstup funkce* `svd`*, můžeme opět použít příkaz* `help`.

Řešení. `[U,S,V] = svd(A);` □

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 naší funkci* `plotSVD`*. U obou funkcí je dobré si zjistit podobu vstupu.*

   Řešení. `semilogy(diag(S)); plotSVD(A,U,S,V)` □

5. *Vypočítejme matici pouze s polovinou (přibližně polovinou) singulárního tripletu.*

   Řešení. `B = U(:,1:300)*S(1:300,1:300)*(V(:,1:300))';` □

6. *Pomocí příkazů* `imagesc` *a* `colormap` *vykresleme zkomprimovaný snímek v černobíle škále.*

   Řešení. `imagesc(B) colormap(gray)` □

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.*

   Řešení. `figure(3) imagesc(A) colormap(gray)` □

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.