Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Maticové rozklady

Bakalářská práce

Autor: Kateřina Frončková

Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. RNDr. Pavel Pražák, Ph.D.

Hradec Králové duben 2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 27.4.2016 Kateřina Frončková

Poděkování

Ráda bych poděkovala doc. RNDr. Pavlu Pražákovi, Ph.D. za odborné vedení, ochotu a rady poskytnuté při vypracovávání této práce.

Anotace

Bakalářská práce se zabývá představením vybraných maticových rozkladů a jejich aplikacemi. Konkrétně je pozornost věnována LU rozkladu, Moore-Penroseově inverzi a SVD rozkladu. Každý z rozkladů je popsán na teoretické úrovni, následně je uveden algoritmus pro jeho výpočet a dále jsou pak představeny vybrané aplikace, jejichž použití je ukázáno na řešení různých příkladů a demonstračních úloh. Součástí práce je také implementace popsaných algoritmů a aplikačních úloh v prostředí MATLAB.

Annotation

Title: **Matrix factorization**

This bachelor thesis deals with the presentation of selected matrix factorizations and their applications. Specifically, attention is aimed at the LU factorization, the Moore-Penrose inverse and the singular value decomposition. Each of the factorizations is described on a theoretical level, subsequently, an algorithm for its computation is stated and then its selected applications are presented and demonstrated on solving different problems and examples. The described algorithms and applications are also implemented in the MATLAB environment.

Obsah

1 Úvod

Maticové rozklady tvoří jednu ze skupin maticových výpočtů, které jsou součástí oblasti matematiky nazývané numerická matematika či numerická analýza. Maticové rozklady obecně spočívají v rozložení matice na součin několika matic, které mají určité specifické charakteristiky (například se může jednat o matice diagonální, horní či dolní trojúhelníkové, ortogonální apod.), přičemž by toto mělo umožnit jednodušší řešení určitého problému či úlohy. Pro zajímavost, maticové rozklady se objevily i v seznamu 10 nejdůležitějších algoritmů 20. století dle [4].

Cílem práce je představit některé vybrané maticové rozklady, kromě teoretického popisu uvést i algoritmy pro jejich výpočet a zaměřit se na jejich praktické aplikace. Snahou je také popsané algoritmy implementovat v prostředí MATLAB a stručně nastínit jejich některé numerické aspekty. Cílem však není provést podrobnou numerickou analýzu těchto algoritmů. Posledním úkolem je předvést zmíněné aplikace rozkladů na řešení různých demonstračních příkladů a úloh, opět za využití prostředí MATLAB.

První část práce se věnuje připomenutí základních pojmů lineární algebry, které budou stěžejní pro celý následující text.

Druhá část již představuje jeden z nejznámějších maticových rozkladů – LU rozklad, spolu s ním popisuje Gaussovu eliminační metodu a dále zmiňuje jeho použití při řešení soustav lineárních algebraických rovnic, výpočtu inverzní matice a determinantu matice.

Další kapitola se zaměřuje na Moore-Penroseovu inverzi, která sice není rozkladem v pravém slova smyslu, ale úzce souvisí s popisovanou problematikou. Kapitola uvádí dvě metody jejího výpočtu a zabývá se také využitím při řešení problému nejmenších čtverců.

Poslední kapitola práce prezentuje SVD rozklad, který je řazen mezi nejdůležitější nástroje maticových výpočtů. V souvislosti s jeho výpočtem jsou představeny další dva maticové rozklady – spektrální rozklad a QR rozklad. Na závěr je vysvětleno použití SVD rozkladu například při výpočtu některých vlastností matic, aproximaci matice maticí nižší hodnosti, řešení úloh o nejmenších čtvercích a výpočtu Moore-Penroseovy inverze, řešení ill-posed problémů nebo při analýze hlavních komponent, některé aplikace jsou předvedeny na úlohách z oblasti zpracování obrazu.

2 Základní pojmy lineární algebry

Na začátek bude připomenuto několik základních pojmů lineární algebry, které se prolínají celým textem. Další pojmy budou vysvětlovány postupně, vždy na místě, kde s nimi bude v textu pracováno. Množství ostatních definic či vět včetně jejich odvození a důkazů lze nalézt v různých textech věnujících se lineární algebře, například v [1], [5], [8], [21], které posloužily jako hlavní zdroje pro tuto kapitolu.

2.1 Matice

2.1.1 Definice matice a vektoru

Definice 2.1.1. Obdélníkové schéma sestavené z reálných čísel

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

se nazývá (reálná) **matice** typu $m \times n$.

Matice budou značeny velkými písmeny, prvek matice $\mathbf{A}$ nacházející se na pozici ij bude zapisován jako a_{ij}. Pro označení i-tého řádku matice $\mathbf{A}$ bude použit symbol $\mathbf{A}_{i\bullet}$ a obdobně pro j-tý sloupec $\mathbf{A}_{\bullet j}$ popř. $\boldsymbol{a}_j$. Symbol $\mathbb{R}^{m\times n}$ označuje množinu všech reálných matic typu $m \times n$, text je omezen pouze na reálné matice, ačkoli většinu uvedeného by bylo možné zobecnit i na matice komplexní. Pokud $m = n$, jedná se o čtvercovou matici řádu n.

Definice 2.1.2. Matice typu $n \times 1$ se nazývá n-rozměrný aritmetický sloupcový **vektor**

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

řádkový vektor je matice typu $1 \times n$

$$\boldsymbol{x}^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}.$$

Vektory budou v textu označovány malými písmeny. Pro množinu všech reálných n-rozměrných vektorů je použit symbol $\mathbb{R}^n$.

2.1.2 Speciální typy matic, inverzní matice, determinant

Následující pojmy se týkají pouze čtvercových matic. Pojmy z definice 2.1.3 budou v kapitole 5 analogicky použity i pro matice obdélníkové tak, jak je tomu například v [21].

Definice 2.1.3. Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$ je čtvercová matice řádu n. Matice $\mathbf{A}$ se nazývá

1. **horní trojúhelníková**, je-li $a_{ij} = 0$ pro $i > j$,
2. **dolní trojúhelníková**, je-li $a_{ij} = 0$ pro $i < j$,
3. **diagonální**, je-li $a_{ij} = 0$ pro $i \neq j$,

$i, j = 1, \ldots, n$.

Dalším důležitým pojmem je regulární matice a s ní spojená inverzní matice. Možné definice těchto pojmů jsou následující:

Definice 2.1.4. Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$ je čtvercová matice řádu n. Matice $\mathbf{A}$ se nazývá **regulární**, jestliže existuje matice $\mathbf{B} \in \mathbb{R}^{n \times n}$ taková, že

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}, \tag{2.1}$$

kde $\mathbf{I}$ je jednotková matice řádu n.

Čtvercové matici, která není regulární, se říká **singulární** matice.

Definice 2.1.5. Matice $\mathbf{B} \in \mathbb{R}^{n\times n}$ z předchozí definice 2.1.4 se nazývá **inverzní matice** k matici $\mathbf{A}$ a značí se symbolem $\mathbf{A}^{-1}$. Vztah (2.1) lze přepsat do podoby

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}.$$

Některé další vlastnosti charakterizující regulární matice shrnuje věta 2.2.1 na konci této kapitoly.

Posledním pojmem vztahujícím se k maticím, který bude v této části definován, je determinant matice. Vlastní definici determinantu předchází zavedení pojmů permutace množiny a znaménko permutace.

Definice 2.1.6. Nechť M je konečná množina. **Permutací množiny** M se nazývá každé vzájemně jednoznačné zobrazení množiny M na množinu M.

Často se uvažuje konečná množina $M = \{1, 2, \ldots, n\}$, pak se permutace p zapisuje jako

$$p = \begin{pmatrix} 1 & 2 & \cdots & n \\ a_1 & a_2 & \cdots & a_n \end{pmatrix},$$

kde čísla $a_i \in M$ a pro všechna $i \in M$ platí $p(i) = a_i$.

Definice 2.1.7. Uspořádaná dvojice (a_i, a_j) se nazývá inverze v permutaci p, jestliže $i < j$ a zároveň $a_i > a_j$. Je-li $|p|$ počet všech inverzí v permutaci p, pak se číslo

$$\operatorname{sign} p = (-1)^{|p|}$$

nazývá **znaménko permutace** p.

Definice 2.1.8. Nechť $\mathbf{A} \in \mathbb{R}^{n\times n}$ je čtvercová matice řádu n. **Determinant** matice $\mathbf{A}$ se nazývá číslo

$$\det(\mathbf{A}) = \sum_p \operatorname{sign} p \; a_{1p_1} a_{2p_2} \cdots a_{np_n},$$

kde se sčítá přes všech $n!$ permutací $p = (p_1, p_2, \ldots, p_n)$ indexové množiny sloupců $\{1, 2, \ldots, n\}$.

2.1.3 Normy vektorů a matic

V textu budou použity tyto dvě normy:
Euklidovská norma vektoru $\boldsymbol{x} \in \mathbb{R}^n$

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$$

Frobeniova norma matice $\mathbf{A} \in \mathbb{R}^{m\times n}$

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

2.2 Soustavy lineárních algebraických rovnic

Definice 2.2.1. Soustava m lineárních algebraických rovnic o n neznámých je soustava ve tvaru

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m. \end{aligned}$$

Pro zápis soustavy lineárních algebraických rovnic se často používá maticový tvar

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

kde

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Matice $\mathbf{A}$ se nazývá matice soustavy, $\boldsymbol{x}$ vektor neznámých, $\boldsymbol{b}$ vektor pravých stran. Je-li $\boldsymbol{b} = \mathbf{0}$, hovoří se o homogenní soustavě, v opačném případě o soustavě nehomogenní.

Speciálním případem je soustava, kde matice $\mathbf{A}$ je čtvercová, tj. $m = n$, a navíc regulární. K řešení těchto soustav se vztahuje kapitola 3. Následující věta shrnuje některé charakteristiky, které platí pro regulární matice, viz [5, s. 34].

Věta 2.2.1. *Nechť $\mathbf{A} \in \mathbb{R}^{n\times n}$ je čtvercová matice řádu n. Potom následující tvrzení jsou ekvivalentní:*

1. $\mathbf{A}$ *je regulární;*
2. *řádky* $\mathbf{A}$ *jsou lineárně nezávislé;*
3. *sloupce* $\mathbf{A}$ *jsou lineárně nezávislé;*

4. $\det(\mathbf{A}) \neq 0$;

5. *existuje* $\mathbf{A}^{-1}$;

6. *pro každé* $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \mathbf{0}$, *je* $\mathbf{A}\boldsymbol{x} \neq \mathbf{0}$;

7. *pro každé* $\boldsymbol{b} \in \mathbb{R}^n$ *má soustava* $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ *právě jedno řešení.*

2.2.1 Přímá a zpětná substituce

V souvislosti s hledáním řešení soustav lineárních algebraických rovnic se často vyskytují pojmy přímé a zpětné substituce. Zpětnou substitucí lze snadno nalézt řešení soustavy s maticí v horním trojúhelníkovém tvaru. Nechť $\mathbf{U}\boldsymbol{x} = \boldsymbol{b}$ je soustava lineárních algebraických rovnic, kde $\mathbf{U} \in \mathbb{R}^{n\times n}$ je regulární horní trojúhelníková matice. Pro poslední složku vektoru řešení $\boldsymbol{x}$ platí $x_n = b_n/u_{nn}$ a dále lze využít vztah

$$x_k = \frac{b_k - \sum_{j=k+1}^{n} u_{kj}x_j}{u_{kk}},$$

pro $k = n-1, n-2, \ldots, 1$. Z regularity matice $\mathbf{U}$ plyne, že všechny diagonální prvky u_{kk} jsou nenulové.

Algoritmus zpětné substituce je shrnut v následujícím výpisu. Pro definici hodnot řídicí proměnné for cyklu je použito tzv. „dvojtečkové značení“ typické pro MATLAB. Konstrukce $a : b : c$ představuje vektor o složkách $a, a+b, a+2b, \ldots, c$, pokud b není specifikováno, je rovno 1.

Algoritmus 1 Zpětná substituce

Vstup: Horní trojúhelníková matice $\mathbf{U} \in \mathbb{R}^{n\times n}$, vektor $\boldsymbol{b} \in \mathbb{R}^n$

```
x_n := b_n/u_nn
for k := n − 1 : −1 : 1
    s := b_k
    for j := k + 1 : n
        s := s − u_kj x_j
    end
    x_k := s/u_kk
end
```

Podobně pro soustavu $\mathbf{L}\boldsymbol{x} = \boldsymbol{b}$ s regulární dolní trojúhelníkovou maticí $\mathbf{L}$ existuje algoritmus přímé substituce.

Algoritmus 2 Přímá substituce

Vstup: Dolní trojúhelníková matice $\mathbf{L} \in \mathbb{R}^{n \times n}$, vektor $\boldsymbol{b} \in \mathbb{R}^n$

$x_1 := b_1 / l_{11}$
for $k := 2 : n$
$s := b_k$
for $j := 1 : k - 1$
$s := s - l_{kj} x_j$
end
$x_k := s / l_{kk}$
end

3 Gaussova eliminační metoda a LU rozklad matice

Kapitola podrobněji popisuje využití Gaussovy eliminace pro převod regulární matice na matici v horním trojúhelníkovém tvaru, a tedy jako nástroj pro výpočet LU rozkladu. Nejprve je uveden algoritmus prosté Gaussovy eliminace, a poté algoritmus Gaussovy eliminace s částečnou pivotací. V aplikační části je ukázáno použití LU rozkladu pro řešení soustav lineárních algebraických rovnic, výpočet inverzní matice a determinantu matice. Teoretická část byla zpracována podle [5], [7], [10], [15] a [18], aplikační vychází z [5], [6] a [18].

3.1 Teoretická východiska a výpočet LU rozkladu

Gaussova eliminační metoda (GEM) je jednou z nejznámějších přímých metod řešení soustav lineárních algebraických rovnic. Její postup obecně spočívá v převedení původní soustavy na soustavu s maticí v řádkově odstupňovaném tvaru.

Definice 3.1.1. Matice $\mathbf{A} \in \mathbb{R}^{m \times n}$ je v **řádkově odstupňovaném tvaru**, jestliže platí:

1. obsahuje-li i-tý řádek samé nuly, pak všechny řádky pod i-tým řádkem jsou také nulové,
2. je-li v i-tém řádku první nenulový prvek v j-tém sloupci, tj. na pozici ij, pak všechny prvky, které leží v prvních j sloupcích a současně pod i-tým řádkem, se rovnají nule.

Poznámka 3.1.1. Odstupňovaný tvar matice bývá také označován jako REF (tvar) z anglického „row echelon form“.

Poznámka 3.1.2. Počet nenulových řádků matice $\mathbf{A}$ v odstupňovaném tvaru se nazývá hodnost matice a bude značen $\text{rank}(\mathbf{A})$.

K převodu matice do odstupňovaného tvaru slouží elementární řádkové úpravy. Matice získaná z původní matice elementárními řádkovými úpravami je s touto původní maticí řádkově ekvivalentní. Více v [5, s. 35].

Definice 3.1.2. Za **elementární řádkové úpravy** matice se považuje:

1. výměna dvou řádků matice,
2. vynásobení řádku matice nenulovým číslem,
3. přičtení násobku nějakého řádku matice k jinému řádku matice.

3.1.1 Prostá Gaussova eliminace a definice LU rozkladu

Nechť $\mathbf{A} \in \mathbb{R}^{n\times n}$ je regulární matice

$$\mathbf{A} = \mathbf{A}^{(0)} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \cdots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} & \cdots & a_{2n}^{(0)} \\ a_{31}^{(0)} & a_{32}^{(0)} & a_{33}^{(0)} & \cdots & a_{3n}^{(0)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & a_{n3}^{(0)} & \cdots & a_{nn}^{(0)} \end{pmatrix}.$$

Poté lze Gaussovu eliminaci provést v následujících krocích, přičemž bude prozatím využito pouze řádkových operací typu 3.
Číslo v horním indexu prvku a_{ij} vyjadřuje, pokolikáté je prvek v matici uložen na pozici ij.

Krok 1
Pokud $a_{11}^{(0)} \neq 0$, je možné eliminovat poddiagonální prvky v prvním sloupci matice $\mathbf{A}^{(0)}$ odečtením $m_{i1} = a_{i1}^{(0)}/a_{11}^{(0)}$ násobku prvního řádku od i-tého řádku, $i = 2, \ldots, n$. Výsledkem je matice

$$\mathbf{A}^{(1)} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix}.$$

Krok 2

Obdobně pokud $a_{22}^{(1)} \neq 0$, poddiagonální prvky druhého sloupce matice $\mathbf{A}^{(1)}$ lze eliminovat odečtením $m_{i2} = a_{i2}^{(1)}/a_{22}^{(1)}$ násobku druhého řádku této matice od i-tého řádku, $i = 3, \ldots, n$. Obdrží se matice

$$\mathbf{A}^{(2)} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}.$$

Krok k

Matice $\mathbf{A}^{(k-1)}$ získaná po $k-1$ krocích má tvar

$$\mathbf{A}^{(k-1)} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1k}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2k}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{pmatrix}.$$

Obecný k-tý krok spočívá v eliminaci poddiagonálních prvků k-tého sloupce matice $\mathbf{A}^{(k-1)}$ odečtením $m_{ik} = a_{ik}^{(k-1)}/a_{kk}^{(k-1)}$ násobku k-tého řádku od i-tého řádku, $i = k+1, \ldots, n$, opět za předpokladu, že $a_{kk}^{(k-1)} \neq 0$.

Poznámka 3.1.3. Prvek $a_{kk}^{(k-1)}$ na hlavní diagonále matice $\mathbf{A}^{(k-1)}$ se nazývá pivot. Číslu $m_{ik} = a_{ik}^{(k-1)}/a_{kk}^{(k-1)}$ se říká multiplikátor (násobitel).

Krok $n-1$

Po provedení $n-1$ kroků Gaussovy eliminace se matice nachází v horním trojúhelníkovém tvaru

$$\mathbf{A}^{(n-1)} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} \end{pmatrix}.$$

Dále bude pro tuto matici používáno označení $\mathbf{U}$ (z anglického „upper triangular“).

Každou řádkovou úpravu lze reprezentovat násobením příslušnou maticí zleva. Pokud $\mathbf{L}_1$ bude označení pro matici reprezentující úpravy provedené v prvním kroku Gaussovy eliminace, $\mathbf{L}_2$ bude odpovídat druhému kroku, až postupně $\mathbf{L}_{n-1}$ bude reprezentovat krok $n-1$, tedy krok poslední, výslednou horní trojúhelníkovou matici $\mathbf{U}$ je možné zapsat jako součin

$$\mathbf{U} = \mathbf{L}_{n-1} \cdots \mathbf{L}_2 \mathbf{L}_1 \mathbf{A}, \tag{3.1}$$

$\mathbf{L}_k$, $k = 1, \ldots, n-1$, je dolní trojúhelníková matice vyjadřující k-tý krok Gaussovy eliminace

$$\mathbf{L}_k = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1k} & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_{nk} & 0 & \cdots & 1 \end{pmatrix},$$

kde $m_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}$, $i = k+1, \ldots, n$.

Rovnici (3.1) je možné přepsat do tvaru

$$\mathbf{A} = \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \cdots \mathbf{L}_{n-1}^{-1} \mathbf{U},$$

a pokud bude zavedeno $\mathbf{L} = \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \cdots \mathbf{L}_{n-1}^{-1}$, získaná matice $\mathbf{L}$ bude v dolním trojúhelníkovém tvaru (označení $\mathbf{L}$ z anglického „lower triangular“), na hlavní diagonále bude obsahovat jedničky a pod diagonálou příslušné multiplikátory.

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{pmatrix}$$

Výsledný zápis rovnice (3.1) má tvar

$$\mathbf{A} = \mathbf{L}\mathbf{U}.$$

Podrobné odvození včetně důkazů se nachází v [5, s. 92].

Definice 3.1.3. Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$ je regulární matice. Rozklad tvaru

$$\mathbf{A} = \mathbf{L}\mathbf{U},$$

kde $\mathbf{L}$ je dolní trojúhelníková matice s jednotkovou diagonálou a $\mathbf{U}$ je horní trojúhelníková matice, se nazývá **LU rozklad matice A**.

3.1.2 Gaussova eliminace a LU rozklad s pivotací

Doposud bylo předpokládáno, že všechny naznačené kroky Gaussovy eliminace lze provést, tedy že v žádném z nich nebude $a_{kk}^{(k-1)} = 0$ pro $k = 1, \ldots, n-1$. Obecně však regularita matice splnění této podmínky nezaručuje. Pokud v průběhu eliminace tato situace nastane, řádek k není možné k eliminaci využít a je nutné nalézt řádek r, kde $a_{rk}^{(k-1)} \neq 0$ a $r > k$, a tyto řádky prohodit. Je-li matice regulární, lze takovýto řádek nalézt vždy.

K výpočtu již tedy nestačí pouze využití řádkových operací typu 3, ale je nezbytné využít i typ 1. Stejně jako v předchozí úvaze i výměnu dvou řádků matice lze reprezentovat pomocí násobení matice vhodnou maticí zleva. Takováto matice je popsána v následující definici.

Definice 3.1.4. Matice $\mathbf{P} \in \mathbb{R}^{n \times n}$, která vznikla z jednotkové matice změnou pořadí řádků (nebo sloupců), se nazývá **permutační matice**. Jedná se o matici, která v každém řádku a každém sloupci má právě jeden prvek rovný 1, ostatní prvky jsou nulové.

Věta 3.1.1. *Nechť $\mathbf{A} \in \mathbb{R}^{n \times n}$ je regulární matice. Potom existuje permutační matice $\mathbf{P}$, dolní trojúhelníková matice $\mathbf{L}$ a horní trojúhelníková matice $\mathbf{U}$ tak, že*

$$\mathbf{PA} = \mathbf{LU}.$$

Důkaz je uveden v [5, s. 102].

Výměna řádků matice nemusí být využita pouze k vyhnutí se situaci, kdy pivot $a_{kk}^{(k-1)}$ bude nulový. Čistě matematicky (v přesné aritmetice) nemá výměna řádků matice žádný vliv na obdržené výsledky. Z numerického hlediska (v konečné aritmetice) však vhodně zvolené pořadí řádků matice může vést k omezení šíření zaokrouhlovacích chyb a tím k zpřesnění výsledku. Snahou tedy je, aby multiplikátor $m_{ik} = a_{ik}^{(k-1)}/a_{kk}^{(k-1)}$ byl v absolutní hodnotě co nejmenší. Toho lze docílit tak, že v každém kroku výpočtu $k = 1, \ldots, n-1$ bude přeuspořádáno $n-k+1$ řádků účastnících se eliminace tak, aby byl jmenovatel zlomku $a_{kk}^{(k-1)}$ v absolutní hodnotě co největší. Je třeba nalézt index l takový, aby

$$|a_{lk}^{(k-1)}| = \max_{i=k,\ldots,n} |a_{ik}^{(k-1)}|,$$

k-tý a l-tý řádek matice $\mathbf{A}^{(k-1)}$ se vzájemně zamění a pokračuje se standardním postupem. Stejným způsobem je ovšem třeba prohodit řádky ve vznikající matici $\mathbf{L}$, do

níž se ukládají jednotlivé multiplikátory. Zároveň je zřejmé, že díky výběru pivota jsou všechny prvky matice **L** v absolutní hodnotě menší nebo rovny jedné.

Výše zmíněný postup popisuje **algoritmus Gaussovy eliminace (LU rozkladu) s částečnou pivotací** (někdy se hovoří též o sloupcovém výběru hlavního prvku). Pokud by byl pivot vybírán ze všech prvků $a_{ij}^{(k-1)}$ pro $i, j = k, \ldots, n$ a prohazovaly by se řádky i sloupce matice, jednalo by se o Gaussovu eliminaci resp. LU rozklad s úplnou pivotací.

Jen pro úplnost, LU rozklad lze uvažovat také pro singulární či obdélníkové matice, více například v [15].

Následující příklad ilustruje algoritmus Gaussovy eliminace resp. LU rozkladu s využitím částečné pivotace.

Příklad 3.1.1. Určeme LU rozklad matice

$$\mathbf{A} = \begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix}.$$

Řešení. Krok 1
Nejprve je v prvním sloupci matice **A** nalezen prvek (pivot) s největší absolutní hodnotou, kterým je číslo 6 v druhém řádku. Dojde tedy k výměně prvního a druhého řádku, což lze reprezentovat násobením permutační maticí

$$\mathbf{P}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Obdrží se matice

$$\mathbf{P}_1\mathbf{A} = \begin{pmatrix} 6 & 4 & 1 \\ 2 & 4 & -3 \\ 3 & 0 & -2 \end{pmatrix}.$$

Nyní lze spočítat příslušné multiplikátory a eliminovat poddiagonální prvky prvního sloupce této matice. Provedené úpravy jsou reprezentovány násobením maticí

$$\mathbf{L}_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{pmatrix}.$$

Výsledkem je matice

$$\mathbf{L}_1\mathbf{P}_1\mathbf{A} = \begin{pmatrix} 6 & 4 & 1 \\ 0 & \frac{8}{3} & -\frac{10}{3} \\ 0 & -2 & -\frac{5}{2} \end{pmatrix}.$$

Krok 2

Opět je nejprve mezi prvky druhého sloupce této matice ($\frac{8}{3}$ a -2) hledán pivot s největší absolutní hodnotou. Je jím číslo $\frac{8}{3}$ nacházející se v druhém řádku. Výměna tedy není nutná a lze rovnou vypočíst multiplikátor a provést eliminaci prvku pod pivotem. Úpravám odpovídá násobení maticí

$$\mathbf{L}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{3}{4} & 1 \end{pmatrix}.$$

Výsledná horní trojúhelníková matice je již hledanou maticí $\mathbf{U}$

$$\mathbf{L}_2\mathbf{L}_1\mathbf{P}_1\mathbf{A} = \begin{pmatrix} 6 & 4 & 1 \\ 0 & \frac{8}{3} & -\frac{10}{3} \\ 0 & 0 & -5 \end{pmatrix} = \mathbf{U}.$$

Pro matice $\mathbf{L}$ a $\mathbf{P}$ platí

$$\mathbf{L} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{1}{2} & -\frac{3}{4} & 1 \end{pmatrix},$$

$$\mathbf{P} = \mathbf{P}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Výsledný LU rozklad lze zapsat dle věty 3.1.1 ve tvaru $\mathbf{PA} = \mathbf{LU}$, tedy

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \frac{1}{2} & -\frac{3}{4} & 1 \end{pmatrix} \begin{pmatrix} 6 & 4 & 1 \\ 0 & \frac{8}{3} & -\frac{10}{3} \\ 0 & 0 & -5 \end{pmatrix}.$$

Numerický pohled na výpočet LU rozkladu

Jak již bylo naznačeno, numerické výpočty mají svá určitá specifika oproti „klasickým“ matematickým výpočtům v přesné aritmetice. Při tvorbě programu je nutné vzít v potaz otázku časové náročnosti výpočtu, paměťových nároků či přesnosti získaných výsledků. Obecnými charakteristikami a problémy, které provázejí numerické výpočty, se zabývá například [15], [29].

Analýza a popis numerických vlastností algoritmů Gaussovy eliminace a LU rozkladu bývají v odborných textech časté, v práci proto nebudou podrobněji uváděny, ale zastoupeny pouze odkazy na literaturu.

Pro výpočet LU rozkladu se v praxi obvykle používá Gaussova eliminace s částečnou pivotací. Úplná pivotace sice obecně vede k lepším (přesnějším) výsledkům, avšak za cenu vyšší časové náročnosti. Mimoto existují také třídy matic, pro které není pivotace nutná (permutační matice by byla maticí jednotkovou), a naopak matice, u kterých ani využití pivotace nepomůže získat přesnější výsledky. O problematice zaokrouhlovacích chyb a numerické stability Gaussovy eliminace, výpočetní složitosti či různých strategiích pivotace více pojednává například [10], [15].

Z pohledu hospodaření s pamětí není nutné ukládat zvlášť matici $\mathbf{L}$, $\mathbf{U}$ a $\mathbf{P}$, ale matice $\mathbf{U}$ postupně přepisuje prvky vstupní matice $\mathbf{A}$ a zároveň jsou do její poddiagonální části ukládány příslušné multiplikátory (prvky matice $\mathbf{L}$). Diagonála matice $\mathbf{L}$ je vždy jednotková, a tedy není třeba ji ukládat. Dále lze permutační matici $\mathbf{P}$ nahradit permutačním vektorem $\boldsymbol{p}$, který bude reprezentovat změnu pořadí řádků matice jako permutaci čísel $1, \ldots, n$ (n je řád matice $\mathbf{A}$). Toto v důsledku vede nejen ke snížení paměťové náročnosti, ale i časové. Uvedené úvahy shrnuje algoritmus 3.

Poslední poznámka k výpočtu LU rozkladu se týká pořadí cyklů v zápisu algoritmu. V algoritmu 3 jsou cykly uvedeny v pořadí kij, toto pořadí lze však přeuspořádat a odvodit tak dalších pět variant zápisu algoritmu LU rozkladu, které se liší svou efektivitou v závislosti na použité architektuře počítače. Všechny varianty jsou uvedeny v [7].

Algoritmus 3 LU rozklad s částečnou pivotací

Vstup: Regulární matice $\mathbf{A} \in \mathbb{R}^{n \times n}$

```
p := 1 : n
for k := 1 : n − 1
  Nalezni l tak, aby |a_lk| = max_{i=k,...,n} |a_ik|
  if a_lk = 0
    Chybové hlášení(Matice A je singulární.)
    Ukonči
  end
  A_k• ↔ A_l•
  p_k ↔ p_l
  for i := k + 1 : n
    m := a_ik / a_kk
    for j := k + 1 : n
      a_ij := a_ij − m · a_kj
    end
    a_ik := m
  end
end
if a_nn = 0
  Chybové hlášení(Matice A je singulární.)
  Ukonči
end
```

Implementace tohoto algoritmu v MATLABu je uvedena v příloze A.1.1.

3.2 Aplikace LU rozkladu

V následující části textu jsou představeny nejznámější aplikace LU rozkladu, zdrojové kódy jejich implementací lze nalézt v příloze A.1.2.

3.2.1 Řešení soustav lineárních algebraických rovnic

Hlavní využití LU rozkladu spočívá v řešení soustav lineárních algebraických rovnic. Nechť je zadána soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ a $\mathbf{A} \in \mathbb{R}^{n \times n}$ je regulární. Dle věty 3.1.1 lze nalézt vhodnou permutační matici $\mathbf{P}$ tak, že matice $\mathbf{PA}$ má LU rozklad. Pak soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$

je ekvivalentní soustavě $\mathbf{PA}\boldsymbol{x} = \mathbf{P}\boldsymbol{b}$ (změna pořadí rovnic soustavy je elementární operací, která zachovává množinu řešení). A protože $\mathbf{PA} = \mathbf{LU}$, soustavu lze přepsat do tvaru $\mathbf{LU}\boldsymbol{x} = \mathbf{P}\boldsymbol{b}$.

Jelikož $\mathbf{A}$ je regulární, soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ má právě jedno řešení (viz věta 2.2.1). Při známém LU rozkladu lze toto řešení $\boldsymbol{x}$ nalézt v následujících krocích:

1. Vyřešení soustavy $\mathbf{L}\boldsymbol{y} = \mathbf{P}\boldsymbol{b}$ pomocí přímé substituce (algoritmus 2).
2. Vyřešení soustavy $\mathbf{U}\boldsymbol{x} = \boldsymbol{y}$ pomocí zpětné substituce (algoritmus 1).

Oproti řešení soustav rovnic Gaussovou eliminací přináší LU rozklad výhodu v situaci, kdy je třeba vyřešit více soustav rovnic se stejnou maticí soustavy $\mathbf{A}$, ale pokaždé s jiným vektorem pravých stran $\boldsymbol{b}$. LU rozklad matice $\mathbf{A}$ stačí vypočítat jen jednou, a poté pro každý vektor $\boldsymbol{b}$ provést pouze dva výše uvedené kroky. Není tedy nutné provádět výpočetně náročnější eliminační proces pro každou soustavu zvlášť jako při použití klasické Gaussovy eliminace.

Řešení soustav rovnic je v praxi velmi časté. Jako demonstrace popsaného postupu řešení poslouží následující příklad, který pochází ze starověké čínské knihy Jiu zhang suan shu (Matematika v devíti kapitolách), a je tak považován za nejstarší zaznamenanou úlohu vedoucí k řešení soustavy rovnic.

Příklad 3.2.1. (převzato z [16]) „Mějme 3 snopy lepšího obilí, 2 snopy středního obilí a 1 snop horšího obilí, obsah je celkem 39 dou. Mějme 2 snopy lepšího, 3 snopy středního a 1 snop horšího, obsah je 34 dou. Mějme 1 snop lepšího, 2 snopy středního a 3 snopy horšího, obsah je 26 dou. Ptáme se, kolik je obsah 1 snopu lepšího, středního a horšího obilí? “

Řešení. Zadanou úlohu lze vyjádřit soustavou tří rovnic o třech neznámých, kde x_1 je obsah 1 snopu lepšího obilí, x_2 středního obilí a x_3 horšího obilí.

$$\begin{aligned} 3x_1 + 2x_2 + x_3 &= 39 \\ 2x_1 + 3x_2 + x_3 &= 34 \\ x_1 + 2x_2 + 3x_3 &= 26 \end{aligned}$$

Této soustavě odpovídá maticový zápis $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$.

$$\begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 1 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 39 \\ 34 \\ 26 \end{pmatrix}$$

Matice $\mathbf{A}$ má LU rozklad

$$\mathbf{A} = \mathbf{LU} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ \frac{1}{3} & \frac{4}{5} & 1 \end{pmatrix} \begin{pmatrix} 3 & 2 & 1 \\ 0 & \frac{5}{3} & \frac{1}{3} \\ 0 & 0 & \frac{12}{5} \end{pmatrix}.$$

Nejprve je pomocí algoritmu přímé substituce vyřešena soustava $\mathbf{L}\boldsymbol{y} = \boldsymbol{b}$. Výsledkem je vektor

$$\boldsymbol{y} = \begin{pmatrix} 39 \\ 8 \\ \frac{33}{5} \end{pmatrix}.$$

V druhém kroku je zpětnou substitucí nalezeno řešení soustavy $\mathbf{U}\boldsymbol{x} = \boldsymbol{y}$, které je hledaným řešením zadané úlohy.

$$\boldsymbol{x} = \begin{pmatrix} \frac{37}{4} \\ \frac{17}{4} \\ \frac{11}{4} \end{pmatrix}$$

Odpověď zní: Obsah 1 snopu lepšího obilí je $\frac{37}{4}$ dou, středního obilí $\frac{17}{4}$ dou a horšího obilí $\frac{11}{4}$ dou.

3.2.2 Výpočet inverzní matice

Metoda A

Jak již bylo uvedeno v definici 2.1.5, pro regulární matici $\mathbf{A} \in \mathbb{R}^{n\times n}$ a její inverzi $\mathbf{A}^{-1}$ platí, že $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. Tato rovnost platí právě tehdy, když

$$\mathbf{A}\mathbf{A}^{-1}_{\bullet j} = \boldsymbol{e}_j,$$

pro každé $j = 1, \ldots, n$. Tedy j-tý sloupec matice $\mathbf{A}^{-1}$ lze získat jako řešení soustavy $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, kde $\boldsymbol{b}$ je j-tý sloupec jednotkové matice $\mathbf{I}_{\bullet j}$ resp. $\boldsymbol{e}_j$.

Příklad 3.2.2. Nalezněme inverzní matici k matici $\mathbf{A}$ z příkladu 3.1.1,

$$\mathbf{A} = \begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix}.$$

Řešení. První sloupec inverzní matice $\mathbf{A}^{-1}$ je řešením $\boldsymbol{x}$ soustavy $\mathbf{A}\boldsymbol{x} = \boldsymbol{e}_1$, neboli

$$\begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Za využití výše popsaného postupu pro řešení soustavy rovnic a LU rozkladu matice $\mathbf{A}$ spočteného v příkladu 3.1.1 je získáno řešení

$$\boldsymbol{x} = \begin{pmatrix} -\frac{1}{10} \\ \frac{3}{16} \\ -\frac{3}{20} \end{pmatrix}.$$

Obdobně druhý sloupec inverzní matice lze získat vyřešením soustavy

$$\begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix} \boldsymbol{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Výsledkem je vektor řešení

$$\boldsymbol{y} = \begin{pmatrix} \frac{1}{10} \\ \frac{1}{16} \\ \frac{3}{20} \end{pmatrix}.$$

A nakonec třetí sloupec inverzní matice odpovídá řešení soustavy

$$\begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix} \boldsymbol{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

kterým je vektor

$$\boldsymbol{z} = \begin{pmatrix} \frac{1}{5} \\ -\frac{1}{4} \\ -\frac{1}{5} \end{pmatrix}.$$

Nyní už zbývá sestavit výslednou inverzní matici $\mathbf{A}^{-1}$

$$\mathbf{A}^{-1} = \begin{pmatrix} \boldsymbol{x} & \boldsymbol{y} & \boldsymbol{z} \end{pmatrix} = \begin{pmatrix} -\frac{1}{10} & \frac{1}{10} & \frac{1}{5} \\ \frac{3}{16} & \frac{1}{16} & -\frac{1}{4} \\ -\frac{3}{20} & \frac{3}{20} & -\frac{1}{5} \end{pmatrix}.$$

Kromě této metody uvádí Du Croz a Higham [6] další postupy výpočtu inverzní matice pomocí LU rozkladu. Pro zjednodušení bude v dalším výkladu uvažován LU rozklad bez pivotace, $\mathbf{A} = \mathbf{LU}$.

Metoda B

Z výchozího vztahu $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ lze odvodit maticovou rovnici

$$\mathbf{A}^{-1}\mathbf{L} = \mathbf{U}^{-1}$$

a inverzní matici $\mathbf{A}^{-1}$ získat jako její řešení.

Metoda C

Tato metoda vychází z rovnice $\mathbf{U}\mathbf{A}^{-1}\mathbf{L} = \mathbf{I}$. V každém kroku k (pro $k = n, n-1, \ldots, 1$) rekurzivního výpočtu je spočítána část k-tého sloupce inverzní matice $\mathbf{A}^{-1}_{k+1:n,k}$, část k-tého řádku $\mathbf{A}^{-1}_{k,k+1:n}$ a číslo $\mathbf{A}^{-1}_{kk}$.

Budiž předpokládána následující bloková reprezentace matic

$$\mathbf{A}^{-1} = \mathbf{X} = \begin{pmatrix} x_{11} & \boldsymbol{x}_{12}^T \\ \boldsymbol{x}_{21} & \mathbf{X}_{22} \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 \\ \boldsymbol{l}_{21} & \mathbf{L}_{22} \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} u_{11} & \boldsymbol{u}_{12}^T \\ 0 & \mathbf{U}_{22} \end{pmatrix}.$$

Při známé submatici $\mathbf{X}_{22}$ je zbytek matice $\mathbf{X}$ dopočítán takto

$$\begin{aligned} \boldsymbol{x}_{21} &= -\mathbf{X}_{22}\boldsymbol{l}_{21}, \\ \boldsymbol{x}_{12}^T &= -\boldsymbol{u}_{12}^T\mathbf{X}_{22}/u_{11}, \\ x_{11} &= 1/u_{11} - \boldsymbol{x}_{12}^T\boldsymbol{l}_{21}. \end{aligned}$$

Metoda D

Poslední metoda je opět založena na rovnici $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$, která je upravena do podoby

$$\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}.$$

Článek [6] obsahuje podrobnější popis a analýzu uvedených metod včetně jejich srovnání z numerického hlediska. Dále je zde uvedeno také několik algoritmů pro výpočet inverzní matice k matici v horním resp. dolním trojúhelníkovém tvaru, které je možné využít při implementaci metody B a D.

3.2.3 Výpočet determinantu matice

Další aplikací LU rozkladu je výpočet determinantu čtvercové matice $\mathbf{A} \in \mathbb{R}^{n\times n}$. Je-li $\mathbf{PA} = \mathbf{LU}$, pak pro determinanty těchto matic platí

$$\det(\mathbf{P})\det(\mathbf{A}) = \det(\mathbf{L})\det(\mathbf{U}).$$

Dále bude využito skutečnosti, že determinant trojúhelníkové matice je roven součinu prvků na hlavní diagonále této matice (viz [21, s. 127]). Z toho vyplývá, že $\det(\mathbf{L}) = 1$ a $\det(\mathbf{U}) = u_{11}u_{22}\cdots u_{nn}$. Determinant permutační matice $\mathbf{P}$ je buď $\det(\mathbf{P}) = 1$, pokud $\mathbf{P}$ vznikla z jednotkové matice sudým počtem výměn řádků, nebo $\det(\mathbf{P}) = -1$

v případě, že $\mathbf{P}$ vznikla z jednotkové matice lichým počtem výměn řádků.

Pro výpočet determinantu matice $\mathbf{A}$ lze psát vztah

$$\det(\mathbf{A}) = (-1)^r u_{11} u_{22} \cdots u_{nn}, \tag{3.2}$$

kde r je počet výměn řádků během LU rozkladu.

Tento způsob určení determinantu je použitelný i pro matice vyšších řádů, kdy výpočet podle definice 2.1.8 již není reálně proveditelný.

Příklad 3.2.3. Vyčísleme determinant matice $\mathbf{A}$ z příkladu 3.1.1,

$$\mathbf{A} = \begin{pmatrix} 2 & 4 & -3 \\ 6 & 4 & 1 \\ 3 & 0 & -2 \end{pmatrix}.$$

Řešení. Opět bude využito již spočteného LU rozkladu resp. matice $\mathbf{U}$,

$$\mathbf{U} = \begin{pmatrix} 6 & 4 & 1 \\ 0 & \frac{8}{3} & -\frac{10}{3} \\ 0 & 0 & -5 \end{pmatrix}$$

a faktu, že při výpočtu byla provedena jedna výměna řádků.

Příslušné hodnoty stačí dosadit do vztahu (3.2)

$$\det(\mathbf{A}) = (-1)^1 \cdot 6 \cdot \frac{8}{3} \cdot (-5) = 80.$$

4 Moore-Penroseova inverze matice

Kapitola představuje pojem Moore-Penroseovy inverze matice, zmiňuje některé její důležité charakteristiky a uvádí dvě metody výpočtu. Dále popisuje její uplatnění při řešení problému nejmenších čtverců. Část kapitoly věnující se definici a vlastnostem Moore-Penroseovy inverze čerpá z [3], [21] a [24], metody výpočtu byly zpracovány podle [3], [21] (metoda hodnostního rozkladu) a [11], [21], [22] (Grevillův algoritmus). Aplikační část vychází z [5] a [7].

4.1 Teoretická východiska a výpočet Moore-Penroseovy inverze

4.1.1 Definice a vlastnosti Moore-Penroseovy inverze

V předchozí kapitole o LU rozkladu bylo pracováno pouze s maticemi, které byly regulární. To, že je matice regulární, je ekvivalentní s tvrzením, že k dané matici existuje matice inverzní (viz věta 2.2.1). Singulární čtvercové matice či obdélníkové matice tedy inverzní matici nemají, ke každé matici však lze nalézt určitou matici, která splňuje vlastnosti charakterizované následující větou:

Věta 4.1.1. *Pro každou matici* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *existuje právě jedna matice* $\mathbf{A}^\dagger \in \mathbb{R}^{n\times m}$ *s těmito vlastnostmi:*

1. $\mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}$,
2. $\mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger$,
3. $(\mathbf{A}\mathbf{A}^\dagger)^T = \mathbf{A}\mathbf{A}^\dagger$,
4. $(\mathbf{A}^\dagger\mathbf{A})^T = \mathbf{A}^\dagger\mathbf{A}$.

Definice 4.1.1. Matice $\mathbf{A}^\dagger$ se nazývá **Moore-Penroseova inverze matice A**.

Poznámka 4.1.1. Místo označení Moore-Penroseova inverze matice se synonymně hovoří také o **pseudoinverzi matice**. Vlastnosti 1. – 4. z věty 4.1.1 se nazývají Penroseovy podmínky. Kromě Moore-Penroseovy inverze existují ještě další zobecněné inverze, které splňují některé z Penroseových podmínek, více o nich pojednává například [3].

Důkaz existence matice $\mathbf{A}^\dagger$ spočívá v ukázání, že pro matici $\mathbf{A}^\dagger$ platí Penroseovy podmínky, k tomu je možné využít hodnostní rozklad matice, který je podrobněji popsán dále v textu. Při důkazu jednoznačnosti matice $\mathbf{A}^\dagger$ se vychází z předpokladu, že existuje nějaká matice $\mathbf{A}^\#$, která také splňuje Penroseovy podmínky, a je dokázáno, že musí platit $\mathbf{A}^\dagger = \mathbf{A}^\#$. Provedení těchto důkazů je k nahlédnutí v [21, s. 42].

Využitím Penroseových podmínek lze dokázat, že Moore-Penroseova inverze splňuje určité vlastnosti (podrobněji v [21, s. 114]). Věta charakterizuje některé z nich.

Věta 4.1.2. *Moore-Penroseova inverze* $\mathbf{A}^\dagger$ *matice* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *má tyto vlastnosti:*

1. $\mathbf{A}^\dagger = \mathbf{A}^{-1}$ *je-li* $\mathbf{A}$ *čtvercová regulární,*
2. $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ *má-li* $\mathbf{A}$ *lineárně nezávislé sloupce,*
3. $\mathbf{A}^\dagger = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$ *má-li* $\mathbf{A}$ *lineárně nezávislé řádky,*
4. $(\mathbf{A}^\dagger)^\dagger = \mathbf{A}$.

4.1.2 Metody výpočtu Moore-Penroseovy inverze

Pro výpočet Moore-Penroseovy inverze existuje více metod. V této kapitole budou popsány dvě, první je založena na využití hodnostního rozkladu matice, druhou metodou je algoritmus, který publikoval T. N. E. Greville v [11]. Třetí způsob výpočtu využívající SVD rozklad je uveden v části 5.2.3. Pro příklady dalších metod výpočtu je možné nahlédnout do [3]. Implementace popsaných algoritmů jsou k vidění v příloze A.2.1.

Metoda hodnostního rozkladu matice

Ještě než bude definován vlastní hodnostní rozklad, nejprve bude zaveden pojem redukovaný odstupňovaný tvar matice. Při aplikaci Gaussovy eliminační metody na regulární matici $\mathbf{A} \in \mathbb{R}^{n\times n}$ je v každém kroku k pro $k = 1, \ldots, n-1$, pokud $a_{kk}^{(k-1)} = 0$, nutno nalézt řádek i, kde $a_{ik}^{(k-1)} \neq 0$ a $i > k$, a s k-tým řádkem ho vyměnit. Regularita matice zaručuje, že takovýto řádek lze nalézt vždy. Pokud je matice singulární, nastane situace, kdy $a_{ik}^{(k-1)} = 0$ pro všechna $i \geq k$. Pro singulární nebo obdélníkové

matice tedy Gaussovu eliminaci tak, jak byla popsána v předchozí kapitole, není možné použít. Algoritmus lze ale mírně upravit. Pokud se v k-tém sloupci nenachází nenulový pivot, bude pokračováno jeho hledáním v sloupci j, $j > k$. Dále bude místo Gaussovy eliminace použita Gauss-Jordanova eliminační metoda, která se od Gaussovy eliminace liší tím, že v každém kroku eliminuje i prvky příslušného sloupce nacházející se nad pivotem (algoritmus 4). Výstupem je pak matice v redukovaném odstupňovaném tvaru.

Definice 4.1.2. Matice $\mathbf{A} \in \mathbb{R}^{m\times n}$ je v **redukovaném odstupňovaném tvaru**, jestliže existují $0 \leq r \leq m$ a $1 \leq k_1 < k_2 < \cdots < k_r \leq n$ tak, že platí:

1. $\mathbf{A}_{i1} = \cdots = \mathbf{A}_{i,k_i-1} = \mathbf{0}$ a $\mathbf{A}_{\bullet k_i} = \boldsymbol{e}_i$ pro $i = 1, \ldots, r$,
2. $\mathbf{A}_{i\bullet} = \mathbf{0}^T$ pro $i = r+1, \ldots, m$.

Poznámka 4.1.2. Pro redukovaný odstupňovaný tvar matice se používá také označení RREF (tvar) z anglického „reduced row echelon form“.

Z definice vyplývá, že první nenulový prvek v řádku i pro $i = 1, \ldots, r$ je číslo jedna nacházející se v sloupci k_i a všechny ostatní prvky sloupce k_i jsou nulové. Zbývající řádky $r+1, \ldots, m$ jsou nulové.

Poznámka 4.1.3. Číslo r vyjadřuje hodnost matice, $r = \operatorname{rank}(\mathbf{A})$.

Algoritmus 4 Redukovaný odstupňovaný tvar matice

Vstup: Matice $\mathbf{A} \in \mathbb{R}^{m\times n}$

```
k := 1; j := 1
while k ≤ m and j ≤ n
  if a_il = 0 pro každé i ≥ k a l ≥ j
    Ukonči
  end
  Nalezni j := min{l; l ≥ j, a_il ≠ 0 pro jisté i ≥ k}
  Urči a_ij ≠ 0, i ≥ k a vyměň řádky A_k• ↔ A_i•
  A_k• := (1/a_kj) A_k•
  for all i ≠ k
    A_i• := A_i• − a_ij A_k•
  end
  k := k + 1; j := j + 1
end
```

Lze dokázat [21, s. 40], že matice v redukovaném odstupňovaném tvaru získaná popsaným postupem, je určena jednoznačně, a je tedy možné pro ni zavést označení $\mathbf{A}^R$.

Redukovaný odstupňovaný tvar matice je následně využit pro výpočet hodnostního rozkladu.

Věta 4.1.3. *Nechť $\mathbf{A}^R$ je redukovaný odstupňovaný tvar matice $\mathbf{0} \neq \mathbf{A} \in \mathbb{R}^{m\times n}$. Potom platí*

$$\mathbf{A} = \mathbf{BC}, \tag{4.1}$$

kde $\mathbf{B} \in \mathbb{R}^{m\times r}$ je matice o sloupcích $\mathbf{A}_{\bullet k_1}, \ldots, \mathbf{A}_{\bullet k_r}$ a matice $\mathbf{C} \in \mathbb{R}^{r\times n}$ sestává z prvních r řádků matice $\mathbf{A}^R$. Přitom $\mathbf{B}$ má lineárně nezávislé sloupce a $\mathbf{C}$ má lineárně nezávislé řádky.

Rozklad (4.1) se nazývá **hodnostní rozklad matice A**.

Důkaz lineární nezávislosti sloupců matice $\mathbf{B}$ a řádků matice $\mathbf{C}$ je proveden v [21, s. 41].

Věta 4.1.4. *Nechť $\mathbf{A} = \mathbf{BC}$ je hodnostní rozklad matice $\mathbf{A}$. Pro Moore-Penroseovu inverzi $\mathbf{A}^\dagger$ matice $\mathbf{A}$ platí vztah*

$$\mathbf{A}^\dagger = \mathbf{C}^T(\mathbf{B}^T\mathbf{AC}^T)^{-1}\mathbf{B}^T. \tag{4.2}$$

Jen pro doplnění, z lineární nezávislosti sloupců matice $\mathbf{B}$ a řádků matice $\mathbf{C}$ plyne, že matice $\mathbf{B}^T\mathbf{B}$ a $\mathbf{CC}^T$ jsou regulární, tedy i matice $\mathbf{B}^T\mathbf{AC}^T = \mathbf{B}^T\mathbf{BCC}^T$ je regulární a existuje její inverze $(\mathbf{B}^T\mathbf{AC}^T)^{-1}$. Podrobněji i s důkazy opět v [21].

Algoritmus 5 Metoda hodnostního rozkladu matice

Vstup: Matice $\mathbf{0} \neq \mathbf{A} \in \mathbb{R}^{m\times n}$

Vypočti redukovaný odstupňovaný tvar $\mathbf{A}^R$ matice $\mathbf{A}$

Sestav hodnostní rozklad matice $\mathbf{A}$, $\mathbf{A} = \mathbf{BC}$

$\mathbf{A}^\dagger = \mathbf{C}^T(\mathbf{B}^T\mathbf{AC}^T)^{-1}\mathbf{B}^T$

Moore-Penroseova inverze nulové matice $\mathbf{A} = \mathbf{0}$ je rovna její transpozici $\mathbf{A}^T$.

Uvedený postup výpočtu Moore-Penroseovy inverze je ukázán na následujícím příkladu.

Příklad 4.1.1. Určeme Moore-Penroseovu inverzi matice $\mathbf{A}$ za využití hodnostního rozkladu.

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 2 & -2 & -4 \end{pmatrix}$$

Řešení. Zadaná matice je singulární, pro soustavu $\mathbf{A}\boldsymbol{x} = \mathbf{0}$ lze nalézt vektor řešení $\boldsymbol{x} \neq \mathbf{0}$. Nemá tedy inverzní matici.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 2 & -2 & -4 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Při výpočtu Moore-Penroseovy inverze je nejprve zadaná matice převedena na redukovaný odstupňovaný tvar

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 2 & -2 & -4 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 0 \\ 0 & 3 & 3 \\ 0 & -4 & -4 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \mathbf{A}^R.$$

Dle věty 4.1.3 je pomocí redukovaného odstupňovaného tvaru vypočítán hodnostní rozklad matice $\mathbf{A}$

$$\mathbf{A} = \mathbf{BC} = \begin{pmatrix} 1 & 1 \\ 1 & 4 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Moore-Penroseova inverze $\mathbf{A}^\dagger$ je určena dosazením do vztahu (4.2)

$$\mathbf{A}^\dagger = \mathbf{C}^T(\mathbf{B}^T\mathbf{A}\mathbf{C}^T)^{-1}\mathbf{B}^T = \begin{pmatrix} \frac{3}{25} & \frac{19}{125} & \frac{74}{375} \\ \frac{2}{25} & \frac{21}{125} & \frac{16}{375} \\ -\frac{1}{25} & \frac{2}{125} & -\frac{58}{375} \end{pmatrix}.$$

Budiž poznamenáno, že explicitní vzorec (4.2) má své uplatnění při teoretickém popisu a důkazu Moore-Penroseovy inverze, pro její numerický výpočet se však nepoužívá. Nevýhodou je kromě jeho vyšší výpočetní složitosti také fakt, že vlivem zaokrouhlovacích chyb může nastat situace, kdy matice $\mathbf{B}^T\mathbf{A}\mathbf{C}^T$ nebude regulární, a tedy nebude možné určit její inverzi.

Grevillův algoritmus

Grevillův algoritmus pro výpočet Moore-Penroseovy inverze je rekurzivní algoritmus založený na postupném zpracovávání sloupců vstupní matice.

Nechť je vstupem algoritmu matice $\mathbf{A} \in \mathbb{R}^{m \times n}$. Vychází se z tvrzení, že pro Moore-Penroseovu inverzi $\mathbf{A}_1^\dagger$ matice $\mathbf{A}_1 = \boldsymbol{a}_1$ tvořenou prvním sloupcem $\boldsymbol{a}_1$ matice $\mathbf{A}$ (obecně pro matici typu $m \times 1$) platí

$$\mathbf{A}_1^\dagger = \begin{cases} \frac{\boldsymbol{a}_1^T}{\boldsymbol{a}_1^T \boldsymbol{a}_1} & \text{je-li } \boldsymbol{a}_1 \neq \mathbf{0} \\ \boldsymbol{a}_1^T & \text{je-li } \boldsymbol{a}_1 = \mathbf{0}. \end{cases}$$

Dále bude předpokládáno, že $\mathbf{A}_{k-1}$ je matice typu $m \times (k-1)$, která je tvořena prvními $k-1$ sloupci matice $\mathbf{A}$, a $\mathbf{A}_{k-1}^\dagger$ je její Moore-Penroseova inverze. Poté lze Moore-Penroseovu inverzi matice $\mathbf{A}_k = \begin{pmatrix} \mathbf{A}_{k-1} & \boldsymbol{a}_k \end{pmatrix}$, která vznikla z matice $\mathbf{A}_{k-1}$ přidáním k-tého sloupce $\boldsymbol{a}_k$ matice $\mathbf{A}$, spočítat dle vztahu

$$\mathbf{A}_k^\dagger = \begin{pmatrix} \mathbf{A}_{k-1}^\dagger - \boldsymbol{d}\boldsymbol{b}^T \\ \boldsymbol{b}^T \end{pmatrix}, \tag{4.3}$$

kde

$$\begin{aligned} \boldsymbol{d} &= \mathbf{A}_{k-1}^\dagger \boldsymbol{a}_k, \\ \boldsymbol{c} &= \boldsymbol{a}_k - \mathbf{A}_{k-1} \boldsymbol{d}, \\ \boldsymbol{b}^T &= \begin{cases} \frac{\boldsymbol{c}^T}{\boldsymbol{c}^T \boldsymbol{c}} & \text{je-li } \boldsymbol{c} \neq \mathbf{0} \\ \frac{\boldsymbol{d}^T \mathbf{A}_{k-1}^\dagger}{1 + \boldsymbol{d}^T \boldsymbol{d}} & \text{je-li } \boldsymbol{c} = \mathbf{0}. \end{cases} \end{aligned}$$

Důkaz založený na ověření platnosti Penroseových podmínek z věty 4.1.1 pro matice $\mathbf{A}_k$ a $\mathbf{A}_k^\dagger$ je uveden v [21, s. 44].

Po n opakováních je matice $\mathbf{A}_n^\dagger$ rovna Moore-Penroseově inverzi matice $\mathbf{A}$, symbolicky $\mathbf{A}_n^\dagger = \mathbf{A}^\dagger$.

Algoritmus 6 shrnuje výše popsaný postup. (Pro k-tý sloupec matice $\mathbf{A}$ je použito již dříve zavedené značení $\mathbf{A}_{\bullet k}$, symbol $\mathbf{A}_{\bullet,1:k-1}$ označuje matici tvořenou prvními $k-1$ sloupci matice $\mathbf{A}$.)

Algoritmus 6 Grevillův algoritmus

Vstup: Matice $\mathbf{A} \in \mathbb{R}^{m \times n}$

$\boldsymbol{a} := \mathbf{A}_{\bullet 1}$
if $\boldsymbol{a} = \mathbf{0}$
$\mathbf{X} := \boldsymbol{a}^T$
else
$\mathbf{X} := \frac{\boldsymbol{a}^T}{\boldsymbol{a}^T \boldsymbol{a}}$
end
for $k := 2 : n$
$\boldsymbol{d} := \mathbf{X}\mathbf{A}_{\bullet k}$
$\boldsymbol{c} := \mathbf{A}_{\bullet k} - \mathbf{A}_{\bullet, 1:k-1}\boldsymbol{d}$
if $\boldsymbol{c} = \mathbf{0}$
$\boldsymbol{b}^T := \frac{\boldsymbol{d}^T \mathbf{X}}{1 + \boldsymbol{d}^T \boldsymbol{d}}$
else
$\boldsymbol{b}^T := \frac{\boldsymbol{c}^T}{\boldsymbol{c}^T \boldsymbol{c}}$
end
$\mathbf{X} := \begin{pmatrix} \mathbf{X} - \boldsymbol{d}\boldsymbol{b}^T \\ \boldsymbol{b}^T \end{pmatrix}$
end

Příklad 4.1.2. Určeme Moore-Penroseovu inverzi matice $\mathbf{A}$ pomocí Grevillova algoritmu.

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 2 & -2 & -4 \end{pmatrix}$$

Řešení. Krok 1

Matice $\mathbf{A}_1$ je tvořena prvním sloupcem matice $\mathbf{A}$

$$\mathbf{A}_1 = \boldsymbol{a}_1 = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix},$$

a jelikož $\boldsymbol{a}_1 \neq \mathbf{0}$, je

$$\mathbf{A}_1^\dagger = \frac{\boldsymbol{a}_1^T}{\boldsymbol{a}_1^T \boldsymbol{a}_1} = \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{3} \end{pmatrix}.$$

Krok 2

Matice $\mathbf{A}_2$ vznikne z matice $\mathbf{A}_1$ přidáním druhého sloupce matice $\mathbf{A}$

$$\mathbf{A}_2 = \begin{pmatrix} \mathbf{A}_1 & \boldsymbol{a}_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 4 \\ 2 & -2 \end{pmatrix}.$$

Dále je dopočítáno $\boldsymbol{d}$, $\boldsymbol{c}$ a $\boldsymbol{b}^T$

$$\boldsymbol{d} = \mathbf{A}_1^\dagger \boldsymbol{a}_2 = \begin{pmatrix} \frac{1}{6} \end{pmatrix},$$

$$\boldsymbol{c} = \boldsymbol{a}_2 - \mathbf{A}_1 \boldsymbol{d} = \begin{pmatrix} \frac{5}{6} \\ \frac{23}{6} \\ -\frac{7}{3} \end{pmatrix},$$

$$\boldsymbol{b}^T = \frac{\boldsymbol{c}^T}{\boldsymbol{c}^T \boldsymbol{c}} = \begin{pmatrix} \frac{1}{25} & \frac{23}{125} & -\frac{14}{125} \end{pmatrix}.$$

Dosazením do vzorce (4.3) je určena Moore-Penroseova inverze $\mathbf{A}_2^\dagger$

$$\mathbf{A}_2^\dagger = \begin{pmatrix} \mathbf{A}_1^\dagger - \boldsymbol{d}\boldsymbol{b}^T \\ \boldsymbol{b}^T \end{pmatrix} = \begin{pmatrix} \frac{4}{25} & \frac{17}{125} & \frac{44}{125} \\ \frac{1}{25} & \frac{23}{125} & -\frac{14}{125} \end{pmatrix}.$$

Krok 3

K matici $\mathbf{A}_2$ je přidán třetí sloupec matice $\mathbf{A}$

$$\mathbf{A}_3 = \begin{pmatrix} \mathbf{A}_2 & \boldsymbol{a}_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 2 & -2 & -4 \end{pmatrix}.$$

Pro $\boldsymbol{d}$, $\boldsymbol{c}$ a $\boldsymbol{b}^T$ nyní platí

$$\boldsymbol{d} = \mathbf{A}_2^\dagger \boldsymbol{a}_3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix},$$

$$\boldsymbol{c} = \boldsymbol{a}_3 - \mathbf{A}_2 \boldsymbol{d} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\boldsymbol{b}^T = \frac{\boldsymbol{d}^T \mathbf{A}_2^\dagger}{1 + \boldsymbol{d}^T \boldsymbol{d}} = \begin{pmatrix} -\frac{1}{25} & \frac{2}{125} & -\frac{58}{375} \end{pmatrix}.$$

Získaná matice $\mathbf{A}_3^\dagger$ je rovna výsledné Moore-Penroseově inverzi $\mathbf{A}^\dagger$

$$\mathbf{A}_3^\dagger = \begin{pmatrix} \mathbf{A}_2^\dagger - \boldsymbol{d}\boldsymbol{b}^T \\ \boldsymbol{b}^T \end{pmatrix} = \begin{pmatrix} \frac{3}{25} & \frac{19}{125} & \frac{74}{375} \\ \frac{2}{25} & \frac{21}{125} & \frac{16}{375} \\ -\frac{1}{25} & \frac{2}{125} & -\frac{58}{375} \end{pmatrix} = \mathbf{A}^\dagger.$$

4.2 Aplikace Moore-Penroseovy inverze

Sekce popisuje využití Moore-Penroseovy inverze při řešení problému nejmenších čtverců, příklady dalších aplikací Moore-Penroseovy inverze jsou zmíněny v souvislosti s SVD rozkladem v části 5.2.

4.2.1 Problém nejmenších čtverců

V sekci 3.2 věnující se aplikacím LU rozkladu bylo zmíněno řešení soustav lineárních algebraických rovnic $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ omezené pouze pro případ, kdy je matice $\mathbf{A}$ regulární. Takováto soustava má vždy právě jedno řešení. V obecném případě, kdy je matice soustavy $\mathbf{A}$ obdélníková, nemusí řešení existovat, nebo nemusí být jednoznačné. Přesto může mít smysl hledat vektor $\boldsymbol{x}$ tak, aby platilo $\mathbf{A}\boldsymbol{x} \approx \boldsymbol{b}$. To vede k formulaci úlohy o nejmenších čtvercích.

Definice 4.2.1. Nechť $\mathbf{A} \in \mathbb{R}^{m\times n}$ a $\boldsymbol{b} \in \mathbb{R}^m$, pak $\boldsymbol{x} \in \mathbb{R}^n$ je **řešení úlohy o nejmenších čtvercích** pro $\mathbf{A}$, $\boldsymbol{b}$, jestliže $\boldsymbol{x}$ minimalizuje hodnotu

$$\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|.$$

Poznámka 4.2.1. $\|\cdot\|$ značí nějakou normu na $\mathbb{R}^m$. Obvykle se uvažuje euklidovská norma vektoru, která je definována (viz část 2.1.3) jako odmocnina ze součtu druhých mocnin složek vektoru. V úloze se tedy jedná o minimalizaci součtu čtverců, a proto název „úloha o nejmenších čtvercích“.

Pro zjednodušení bude v následujícím popisu vždy předpokládáno, že $m \geq n$. Řešení úlohy o nejmenších čtvercích je možné charakterizovat pomocí soustavy normálních rovnic (odvození a důkaz viz [7, s. 154]).

Věta 4.2.1. *Nechť $\mathbf{A} \in \mathbb{R}^{m\times n}$ a $\boldsymbol{b} \in \mathbb{R}^m$. Potom $\boldsymbol{x}$ je řešením úlohy o nejmenších čtvercích právě tehdy, je-li řešením soustavy normálních rovnic*

$$\mathbf{A}^T\mathbf{A}\boldsymbol{x} = \mathbf{A}^T\boldsymbol{b}. \tag{4.4}$$

Dále lze dokázat [5, s. 134], že v případě, kdy $\mathbf{A}$ má plnou sloupcovou hodnost ($\operatorname{rank}(\mathbf{A}) = n$, sloupce $\mathbf{A}$ jsou lineárně nezávislé), je matice $\mathbf{A}^T\mathbf{A}$ regulární a soustava (4.4) má právě jedno řešení

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

Toto řešení je pak možné dle věty 4.1.2 zapsat pomocí Moore-Penroseovy inverze

$$\boldsymbol{x} = \mathbf{A}^\dagger\boldsymbol{b}. \tag{4.5}$$

Je zřejmé, že pokud $m = n = \operatorname{rank}(\mathbf{A})$, je $\mathbf{A}^\dagger = \mathbf{A}^{-1}$.

Zápis řešení úlohy o nejmenších čtvercích (4.5) má obecnou platnost i v případě, kdy matice $\mathbf{A}$ nemá plnou sloupcovou hodnost ($\text{rank}(\mathbf{A}) < n$), tomuto se podrobněji věnuje sekce 5.2.3.

Vyjádření řešení pomocí Moore-Penroseovy inverze má však hlavně teoretický význam. V praxi se pro hledání řešení používají jiné metody, které jsou výpočetně méně náročné než nalezení Moore-Penroseovy inverze matice. Příkladem metod pro řešení úlohy o nejmenších čtvercích může být QR rozklad či SVD rozklad, jejich přiblížení spolu s ukázkami dalších metod nabízí [10], [15]. Další postup řešení úlohy o nejmenších čtvercích uvádí Greville v již zmiňovaném článku [11]. Jedná se o algoritmus, který umožňuje přímo rekurzivně vypočítat vektor $\mathbf{A}^\dagger \boldsymbol{b}$ bez nutnosti explicitního vyjádření $\mathbf{A}^\dagger$.

Typickým příkladem problémů, které vedou na řešení úlohy o nejmenších čtvercích (s maticí plné sloupcové hodnosti), je aproximace nějakých naměřených hodnot funkcí. Jako ilustrace poslouží následující příklad aproximace polynomem (polynomiální regrese).

Příklad 4.2.1. Aproximujme uvedené hodnoty polynomy stupně 1, 3, 5 a 6. (Data byla převzata z [29].)

x_i	-3	-2	-1	0	1	2	3
y_i	$-0,2774$	$0,8958$	$-1,5651$	$3,4565$	$3,0601$	$4,8568$	$3,8982$

Řešení. Polynom stupně (nejvýše) $n-1$ lze psát ve tvaru

$$y(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_n x^{n-1},\ a_n \neq 0.$$

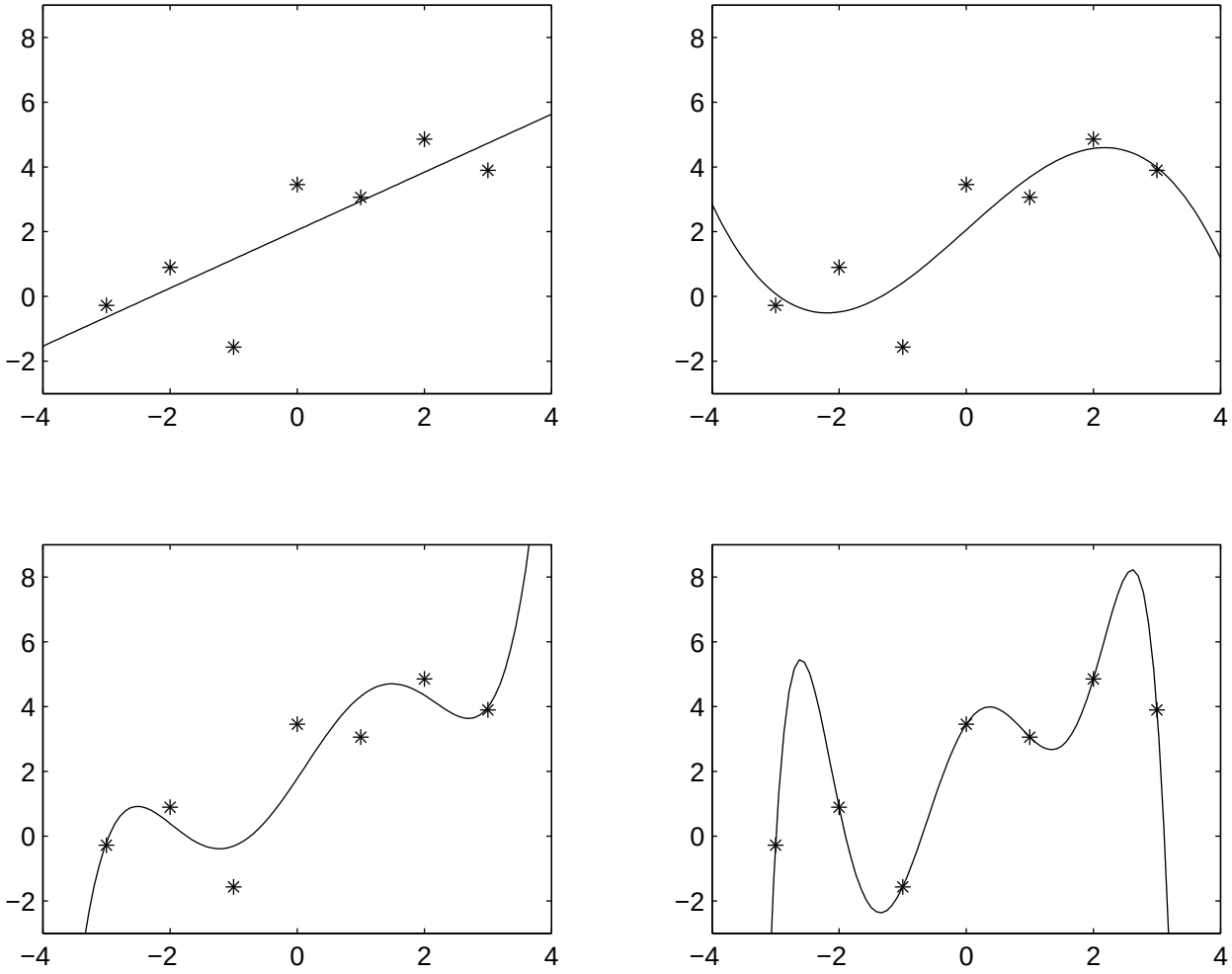
Koeficienty polynomu $a_1, a_2, \ldots, a_n$ je možné získat jako řešení soustavy $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, kde

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{n-1} \end{pmatrix},\ \boldsymbol{b} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix},\ \boldsymbol{x} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix},\ m \geq n.$$

Jelikož zadané hodnoty x_i a y_i byly zjištěny měřením, lze předpokládat, že soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ nemá řešení pro zadané stupně $p = 1, 3$ a 5 aproximačních polynomů. Cílem je tedy nalézt řešení ve smyslu nejmenších čtverců, $\boldsymbol{x} = \mathbf{A}^\dagger \boldsymbol{b}$. V případě polynomu stupně $p = 6$ již soustava $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ řešení mít bude, tento polynom tak bude polynomem interpolačním.

Za využití MATLABu byly vypočteny následující tvary hledaných polynomů (koeficienty jsou zaokrouhleny), jejich grafy znázorňuje obrázek 1.

$$p = 1: \quad y(x) = 2,0464 + 0,8955x$$
$$p = 3: \quad y(x) = 2,0563 + 1,7531x - 0,0025x^2 - 0,1225x^3$$
$$p = 5: \quad y(x) = 1,7769 + 2,9443x + 0,2576x^2 - 0,6795x^3 - 0,0272x^4 + 0,0477x^5$$
$$p = 6: \quad y(x) = 3,4565 + 2,9443x - 3,9948x^2 - 0,6795x^3 + 1,3935x^4 + 0,0477x^5 - \\ - 0,1078x^6$$



Obrázek 1: Výsledné aproximační polynomy stupně 1, 3, 5 a 6

Skript použitý k řešení tohoto příkladu je uveden v příloze A.2.2.

5 SVD rozklad matice

Poslední kapitola práce se věnuje SVD rozkladu matice (z anglického „singular value decomposition“, někdy označován také jako singulární rozklad). Po uvedení věty o existenci SVD rozkladu následuje představení algoritmu pro jeho výpočet. Druhá část kapitoly prezentuje vybrané aplikace SVD rozkladu nejen v matematice, ale například i v počítačové grafice. Teoretická východiska byla zpracována podle [5], [7], [21], [27] při popisu výpočtu bylo čerpáno z [9], [10], [15], [21] a [25]. Hlavními zdroji pro aplikační část byly [5], [7], [12], [21] a [25], poslední uvedená aplikace týkající se analýzy hlavních komponent vychází zejména z [14] a [28].

5.1 Teoretická východiska a výpočet SVD rozkladu

5.1.1 Zavedení SVD rozkladu

Vlastní formulaci SVD rozkladu předchází stručné uvedení pojmu ortogonální matice. Vše, co bude dále zmíněno, úzce souvisí s problematikou vektorových prostorů. Jedním z mnoha textů, ve kterých je zpracována, je [21] – v kapitole 2 lze nalézt definice základních pojmů týkajících se vektorových prostorů, kapitola 3 se podrobněji věnuje vektorovým prostorům se skalárním součinem a v souvislosti s nimi například pojmům ortogonální a ortonormální vektory, ortonormální báze či norma.

Definice 5.1.1. Matice $\mathbf{Q} \in \mathbb{R}^{n \times n}$ se nazývá **ortogonální**, jestliže

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I}.$$

Věta 5.1.1. *Následující tvrzení pro matici* $\mathbf{Q} \in \mathbb{R}^{n \times n}$ *jsou ekvivalentní:*

1. $\mathbf{Q}$ *je ortogonální,*
2. $\mathbf{Q}$ *je regulární a* $\mathbf{Q}^{-1} = \mathbf{Q}^T$*,*

3. $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$,

4. $\mathbf{Q}^T$ *je ortogonální,*

5. *řádky* $\mathbf{Q}$ *tvoří ortonormální bázi* $\mathbb{R}^n$,

6. *sloupce* $\mathbf{Q}$ *tvoří ortonormální bázi* $\mathbb{R}^n$.

Důkaz věty a více o vlastnostech ortogonálních matic v [21, s. 102].

Nyní již lze přistoupit k tvrzení, které je možné označit jako „**věta o existenci SVD rozkladu matice**“. Jedna z variant jejího poněkud rozsáhlejšího důkazu je uvedena v [21, s. 109].

Věta 5.1.2. *Nechť* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *a* $p = \min\{m, n\}$. *Potom existují ortogonální matice* $\mathbf{U} \in \mathbb{R}^{m\times m}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ *a diagonální matice* $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ *s diagonálními prvky* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ *tak, že platí*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{5.1}$$

Diagonální prvky $\sigma_{jj} = \sigma_j, j = 1, \ldots, p$ matice $\mathbf{\Sigma}$ se nazývají singulární čísla matice $\mathbf{A}$. Nechť dále $\boldsymbol{u}_j$ resp. $\boldsymbol{v}_j$ značí j-tý sloupec matice $\mathbf{U}$ resp. $\mathbf{V}$. Vektorům $\boldsymbol{u}_j, j = 1, \ldots, m$ se říká levé singulární vektory a vektorům $\boldsymbol{v}_j, j = 1, \ldots, n$ pravé singulární vektory matice $\mathbf{A}$.

Singulární čísla jsou určena jednoznačně, a pokud je jako ve větě 5.1.2 předpokládáno jejich nerostoucí uspořádání, je i matice $\mathbf{\Sigma}$ určena jednoznačně. Naopak levé a pravé singulární vektory, a tedy matice $\mathbf{U}$ a $\mathbf{V}$, jednoznačně určeny nejsou. Podrobněji k nejednoznačnosti SVD rozkladu v [7].

SVD rozklad matice lze zapsat několika způsoby, rozklad (5.1) má význam zejména pro teoretické účely, v praxi je však vhodnější využívat „úspornější“ varianty zápisu. Pokud je $m > n$, je posledních $m - n$ řádků matice $\mathbf{\Sigma}$ nulových, a tudíž v součinu (5.1) nehraje roli posledních $m - n$ sloupců matice $\mathbf{U}$. Budiž předpokládáno označení $\mathbf{U}_n = \mathbf{U}_{\bullet,1:n}$ a $\mathbf{\Sigma}_n = \mathbf{\Sigma}_{1:n,\bullet}$. Potom lze psát SVD rozklad matice $\mathbf{A} \in \mathbb{R}^{m\times n}$ jako

$$\mathbf{A} = \mathbf{U}_n\mathbf{\Sigma}_n\mathbf{V}^T, \tag{5.2}$$

kde $\mathbf{U}_n \in \mathbb{R}^{m\times n}$ má ortonormální sloupce, $\mathbf{\Sigma}_n \in \mathbb{R}^{n\times n}$ je diagonální matice s diagonálními prvky $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ a $\mathbf{V} \in \mathbb{R}^{n\times n}$ je ortogonální matice.

Rozklad tvaru (5.2) bývá nazýván **redukovaný** či **tenký SVD rozklad**.

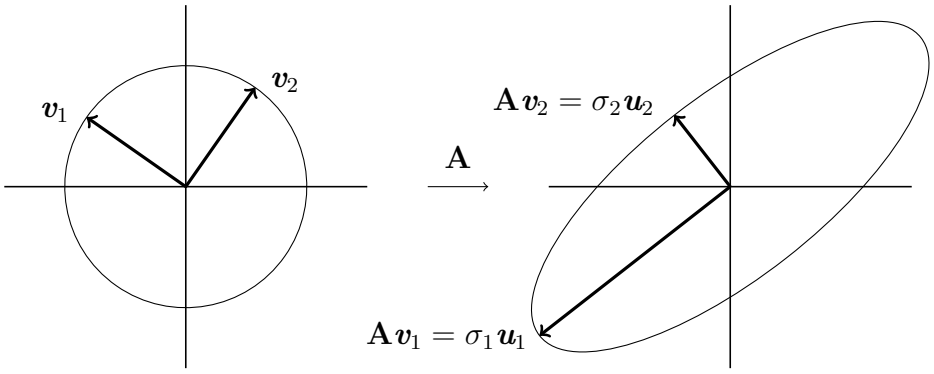
Dále nechť $r = \text{rank}(\mathbf{A})$ a $r < n$ (matice $\mathbf{A}$ nemá plnou hodnost). Pro singulární čísla platí $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$. Matice $\mathbf{\Sigma}_n$ má posledních $n - r$ řádků a posledních $n - r$ sloupců nulových, a tedy v součinu (5.2) nezáleží na posledních $n - r$ sloupcích matice $\mathbf{U}_n$ a posledních $n - r$ řádcích matice $\mathbf{V}^T$ (posledních $n - r$ sloupcích matice $\mathbf{V}$). Při označení $\mathbf{U}_r = \mathbf{U}_{n\ \bullet,1:r} = \mathbf{U}_{\bullet,1:r}$, $\mathbf{\Sigma}_r = \mathbf{\Sigma}_{n\ 1:r,1:r} = \mathbf{\Sigma}_{1:r,1:r}$ a $\mathbf{V}_r = \mathbf{V}_{\bullet,1:r}$ lze SVD rozklad matice $\mathbf{A} \in \mathbb{R}^{m\times n} \neq \mathbf{0}$ vyjádřit takto

$$\mathbf{A} = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^T, \tag{5.3}$$

matice $\mathbf{U}_r \in \mathbb{R}^{m\times r}$, $\mathbf{V}_r \in \mathbb{R}^{n\times r}$ mají ortonormální sloupce a $\mathbf{\Sigma}_r \in \mathbb{R}^{r\times r}$ je diagonální matice s diagonálními prvky $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

Zápisu (5.3) se říká **kondenzovaný SVD rozklad**. Budiž však poznamenáno, že terminologie v obou případech není v literatuře jednoznačná (viz např. [7]).

SVD rozklad lze interpretovat také geometricky. Na matici $\mathbf{A} \in \mathbb{R}^{m\times n}$ je možné nahlížet jako na transformační matici, která transformuje obecně jednotkovou sféru v $\mathbb{R}^n$ na hyperelipsoid v $\mathbb{R}^m$ (pro zjednodušení budiž předpokládáno, že $m \geq n$ a $\text{rank}(\mathbf{A}) = n$). Tomuto odpovídá upravený tvar zápisu SVD rozkladu $\mathbf{A}\boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j, j = 1, \ldots, n$. Levé singulární vektory $\boldsymbol{u}_j$ určují směr poloos hyperelipsoidu a singulární čísla σ_j odpovídají délce těchto poloos. Více o geometrické interpretaci SVD rozkladu pojednává např. [27, s. 25]. Obrázek 2 ilustruje předchozí uvedené pro případ matice $\mathbf{A} \in \mathbb{R}^{2\times 2}$.



Obrázek 2: Geometrická interpretace SVD rozkladu regulární matice $\mathbf{A} \in \mathbb{R}^{2\times 2}$, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ (vlastní zpracování dle [27]). Výslednou transformaci lze vnímat jako transformaci složenou ze tří základních: nejprve je provedena rotace (popř. reflexe) popsaná maticí $\mathbf{V}^T$, následuje změna měřítka reprezentovaná maticí $\mathbf{\Sigma}$ a na závěr dojde opět k rotaci (příp. reflexi) vyjádřené maticí $\mathbf{U}$

5.1.2 Výpočet SVD rozkladu

Pro výpočet SVD rozkladu existuje řada algoritmů. Zde bude naznačen algoritmus využívající spektrálního rozkladu a QR rozkladu matice. Uvedeny budou pouze vybrané definice, věty a základní algoritmy, související podrobnější výklad spolu s důkazy a odvozeními lze nalézt v textu [21], dle kterého byla tato část práce zpracována. Podrobnější informace k efektivním implementacím algoritmů pro výpočet spektrálního rozkladu a QR rozkladu nabízí [10]. Ukázky možných implementací uvedených algoritmů jsou k nahlédnutí v příloze A.3.1.

Spektrální rozklad

Definice 5.1.2 představuje pojem vlastního čísla a vlastního vektoru matice obecně pro komplexní matice. Dále budou v popředí zájmu pouze reálné symetrické matice ($\mathbf{A} = \mathbf{A}^T$), jejichž vlastní čísla jsou vždy reálná. Věta 5.1.3 popisuje **spektrální rozklad matice**, její důkaz viz [21, s. 154].

Definice 5.1.2. Nechť $\mathbf{A} \in \mathbb{C}^{n\times n}$. Jestliže platí

$$\mathbf{A}\boldsymbol{x} = \lambda\boldsymbol{x}$$

pro jisté $\lambda \in \mathbb{C}$ a jistý vektor $\boldsymbol{x} \in \mathbb{C}^n, \boldsymbol{x} \neq \mathbf{0}$, potom se číslo λ nazývá **vlastní číslo** matice $\mathbf{A}$ a vektor $\boldsymbol{x}$ **vlastní vektor** příslušný k tomuto vlastnímu číslu.

Věta 5.1.3. *Ke každé symetrické matici $\mathbf{A} \in \mathbb{R}^{n\times n}$ existuje ortogonální matice $\mathbf{X}$ taková, že platí*

$$\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^T,$$

kde $\mathbf{\Lambda}$ je diagonální s diagonálními prvky $\lambda_{jj} = \lambda_j(\mathbf{A}), j = 1, \ldots, n$, a pro každé j je $\mathbf{X}_{\bullet j}$ vlastní vektor příslušný k vlastnímu číslu $\lambda_j(\mathbf{A})$.

Poznámka 5.1.1. Bude předpokládáno nerostoucí uspořádání vlastních čísel $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.

Jednou z metod výpočtu vlastních čísel a vlastních vektorů symetrické matice $\mathbf{A} \in \mathbb{R}^{n\times n}$, a tedy spektrálního rozkladu, je tzv. Jacobiho metoda. Tato iterační metoda

spočívá v postupném snižování veličiny

$$\mathrm{off}(\mathbf{A}) = \left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}^2 \right)^{1/2}$$

využitím ortogonálních transformací zvaných Givensovy (či Jacobiho) rotace, které nemění vlastní čísla matice.

Obecný krok metody je založen na určení indexů p a q ($1 \leq p \leq q \leq n$) a výpočtu páru sinus-kosinus (s, c) tak, aby matice

$$\begin{pmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

byla diagonální ($b_{pq} = b_{qp} = 0$). Následně je matice $\mathbf{A}$ přepsána maticí $\mathbf{B} = \mathbf{G}^T\mathbf{A}\mathbf{G}$, kde $\mathbf{G}$ je Givensova matice

$$\mathbf{G} = \mathbf{G}(p, q, \theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{matrix} \\ \\ p \\ \\ q \\ \\ \\ \end{matrix}$$
$$\begin{matrix} & & p & & q & & \end{matrix}$$

$s = \sin(\theta), c = \cos(\theta)$.

V každém kroku (s každým přepsáním $\mathbf{A} := \mathbf{G}^T\mathbf{A}\mathbf{G}$) se matice $\mathbf{A}$ stále více blíží matici diagonální – snižuje se hodnota „normy“ nediagonálních prvků $\mathrm{off}(\mathbf{A})$. Po konečně mnoha krocích hodnota $\mathrm{off}(\mathbf{A})$ klesne pod zadanou mez δ a diagonální prvky této matice aproximují vlastní čísla původní matice $\mathbf{A}$ s přesností $\leq \delta$.

V algoritmu 7 je shrnut celý postup Jacobiho metody. Průběžná matice je značena $\mathbf{L}$ (místo $\mathbf{A}$). Indexy p, q jsou voleny tak, aby $|l_{pq}| = \max_{i<j} |l_{ij}|$. Následné hodnoty s, c jsou spočteny bez nutnosti vyjádření úhlu θ. Na závěr jsou diagonální prvky výsledné matice $\mathbf{L}$ seřazeny podle velikosti $l_{11} \geq l_{22} \geq \cdots \geq l_{nn}$. Pro výstupní matice platí $\mathbf{A} = \mathbf{X}\mathbf{L}\mathbf{X}^T$, kde $\mathbf{X}$ je ortogonální, $\mathrm{off}(\mathbf{L}) \leq \delta$ a $|\lambda_i(\mathbf{A}) - l_{ii}| \leq \delta$ pro každé $i = 1, \ldots, n$.

Algoritmus 7 Jacobiho metoda

Vstup: Symetrická matice $\mathbf{A} \in \mathbb{R}^{n\times n}$, požadovaná přesnost δ

$\mathbf{L} := \mathbf{A};\ \mathbf{X} := \mathbf{I}$
while $\mathrm{off}(\mathbf{L}) > \delta$
Nalezni $p < q$, pro které $|l_{pq}| = \max_{i<j} |l_{ij}|$
$\tau := (l_{qq} - l_{pp})/(2l_{pq})$
if $\tau \geq 0$
$t := 1/(\tau + \sqrt{1+\tau^2})$
else
$t := -1/(-\tau + \sqrt{1+\tau^2})$
end
$c := 1/\sqrt{1+t^2};\ s := tc$
$\mathbf{G} := \mathbf{I};\ \mathbf{G}_{pp} := c;\ \mathbf{G}_{qq} := c;\ \mathbf{G}_{pq} := s;\ \mathbf{G}_{qp} := -s$
$\mathbf{L} := \mathbf{G}^T\mathbf{L}\mathbf{G};\ \mathbf{X} := \mathbf{X}\mathbf{G}$
end
Seřaď diagonální prvky $\mathbf{L}$ dle velikosti $l_{k_1k_1} \geq l_{k_2k_2} \geq \cdots \geq l_{k_nk_n}$
Nechť $\mathbf{P}$ je matice, pro kterou $\mathbf{P}_{\bullet i} = \boldsymbol{e}_{k_i}$ pro každé i
$\mathbf{L} := \mathbf{P}^T\mathbf{L}\mathbf{P};\ \mathbf{X} := \mathbf{X}\mathbf{P}$

QR rozklad

Věta 5.1.4 obecně charakterizuje **QR rozklad**, další často využívaný rozklad matice. Důkaz je uveden například v [21, s. 105].

Věta 5.1.4. *Pro každou matici $\mathbf{A} \in \mathbb{R}^{m\times n}$ existuje ortogonální matice $\mathbf{Q} \in \mathbb{R}^{m\times m}$ a horní trojúhelníková matice $\mathbf{R} \in \mathbb{R}^{m\times n}$ tak, že platí*

$$\mathbf{A} = \mathbf{Q}\mathbf{R}.$$

Výpočet QR rozkladu lze opět provést několika způsoby. Metoda zde uvedená je založena na využití Householderových reflexí.

Věta 5.1.5. *Pro každý vektor $\mathbf{0} \neq \boldsymbol{q} \in \mathbb{R}^n$ je matice*

$$\mathbf{H} = \mathbf{I} - 2\frac{\boldsymbol{q}\boldsymbol{q}^T}{\boldsymbol{q}^T\boldsymbol{q}}$$

symetrická a ortogonální.

Jednoduchý důkaz věty je k nahlédnutí v [21, s. 104]. Matice $\mathbf{H}$ se nazývá matice Householderovy reflexe, vektor $\boldsymbol{q}$ nese název Householderův vektor.

Householderovy reflexe lze využít k nulování prvků vektoru (na rozdíl od Givensových rotací umožňují eliminovat i více prvků současně jednou reflexí), a tím k převodu matice na matici v horním trojúhelníkovém tvaru. Klíčem této myšlenky je tvrzení, že pro vektor $\boldsymbol{x} \in \mathbb{R}^n$ existuje Householderova reflexe reprezentovaná maticí $\mathbf{H}$ tak, že

$$\mathbf{H}\boldsymbol{x} = \pm\|\boldsymbol{x}\|_2 \boldsymbol{e}_1.$$

Matice $\mathbf{H}$ má tvar

$$\mathbf{H} = \mathbf{I} - 2\frac{\boldsymbol{q}\boldsymbol{q}^T}{\boldsymbol{q}^T\boldsymbol{q}},$$

kde $\boldsymbol{q} = \boldsymbol{x} \mp \|\boldsymbol{x}\|_2 \boldsymbol{e}_1$.

Postupnou eliminaci poddiagonálních prvků jednotlivých sloupců matice $\mathbf{A} \in \mathbb{R}^{m\times n}$ (buď $m \geq n$) je možné zapsat jako násobení ortogonálními maticemi $\mathbf{Q}_j, j = 1, \ldots, n$

$$\mathbf{Q}_j = \begin{pmatrix} \mathbf{I}_{(j-1)} & 0 \\ 0 & \mathbf{H}_j \end{pmatrix},$$

$\mathbf{I}_{(j-1)}$ značí jednotkovou matici řádu $j-1$, $\mathbf{H}_j$ je matice Householderovy reflexe nulující poddiagonální prvky j-tého sloupce matice $\mathbf{A}$.

Výsledkem je matice $\mathbf{R}$ v horním trojúhelníkovém tvaru

$$\mathbf{Q}_n\mathbf{Q}_{n-1}\cdots\mathbf{Q}_1\mathbf{A} = \mathbf{R}. \tag{5.4}$$

Při využití vlastností ortogonálních matic a označení $\mathbf{Q} = \mathbf{Q}_1^T \cdots \mathbf{Q}_n^T$ lze rovnost (5.4) přepsat do tvaru

$$\mathbf{A} = \mathbf{Q}\mathbf{R}.$$

Shrnutí postupu obsahuje algoritmus 8.

Algoritmus 8 QR rozklad užitím Householderových reflexí

Vstup: Matice $\mathbf{A} \in \mathbb{R}^{m \times n}$

$\mathbf{R} := \mathbf{A}$; $\mathbf{Q} := \mathbf{I}_{(m)}$
for $j := 1 : \min\{m, n\}$
$\boldsymbol{x} := \mathbf{R}_{j:m,j}$
if $x_1 \neq \|\boldsymbol{x}\|_2$
$x_1 := x_1 - \|\boldsymbol{x}\|_2$
$\mathbf{H}_j := \mathbf{I}_{(m-j+1)} - 2\frac{\boldsymbol{x}\boldsymbol{x}^T}{\boldsymbol{x}^T\boldsymbol{x}}$
$\mathbf{Q}_j := \begin{pmatrix} \mathbf{I}_{(j-1)} & 0 \\ 0 & \mathbf{H}_j \end{pmatrix}$
$\mathbf{R} := \mathbf{Q}_j\mathbf{R}$
$\mathbf{Q} := \mathbf{Q}_j\mathbf{Q}$
end
end
$\mathbf{Q} := \mathbf{Q}^T$

QR rozklad matice není určen jednoznačně. V důsledku volby znaménka $x_1 := x_1 - \|\boldsymbol{x}\|_2$ jsou všechny diagonální prvky matice $\mathbf{R}$ na výstupu algoritmu nezáporné. Obecně však tento způsob volby znaménka není numericky stabilní (v případě, že je x_1 kladné a blízké k $\|\boldsymbol{x}\|_2$, může při odečítání dojít k vyrušení platných číslic). Při implementaci algoritmu také není nutné v každé iteraci cyklu vyjadřovat matice $\mathbf{H}_j$ a $\mathbf{Q}_j$, ale veškeré výpočty lze provést pouze za využití Householderových vektorů. Ne vždy je třeba explicitně konstruovat i matici $\mathbf{Q}$. V praxi se pro úsporu paměti využívá přístup, kdy matice $\mathbf{R}$ postupně přepisuje matici $\mathbf{A}$ a do poddiagonální části této matice jsou ukládány jednotlivé Householderovy vektory. Podrobněji k tomuto i k volbě znaménka v [10], [15].

Kromě využití při výpočtu SVD rozkladu lze pomocí QR rozkladu řešit také soustavy rovnic s regulární maticí (v porovnání s LU rozkladem je tento přístup numericky stabilnější, avšak výpočetně dražší) a je i často používaným nástrojem pro řešení úloh o nejmenších čtvercích.

Zpět k výpočtu SVD rozkladu

Lze odvodit (více např. [7]), že existuje určitý vztah mezi SVD rozkladem matice $\mathbf{A} \in \mathbb{R}^{m \times n}$ a spektrálními rozklady matic $\mathbf{A}^T\mathbf{A}$ a $\mathbf{A}\mathbf{A}^T$ (budiž podotknuto, že pro každou matici $\mathbf{A}$ jsou matice $\mathbf{A}^T\mathbf{A}$ a $\mathbf{A}\mathbf{A}^T$ symetrické a dle věty 5.1.3 lze určit jejich spektrální rozklad). Nechť $r = \text{rank}(\mathbf{A})$, nenulová singulární čísla matice $\mathbf{A}$, $\sigma_1, \ldots, \sigma_r$,

jsou odmocninami nenulových (kladných) vlastních čísel matic $\mathbf{A}^T\mathbf{A}$ a $\mathbf{A}\mathbf{A}^T$. Levé singulární vektory matice $\mathbf{A}$ jsou příslušné vlastní vektory matice $\mathbf{A}\mathbf{A}^T$ a pravé singulární vektory matice $\mathbf{A}$ jsou příslušné vlastní vektory matice $\mathbf{A}^T\mathbf{A}$.

Jednu z možností, jak na základě výše uvedeného nalézt SVD rozklad matice, popisuje následující věta, důkaz viz [21, s. 162].

Věta 5.1.6. *Nechť* $\mathbf{A} \in \mathbb{R}^{m\times n}, m \geq n$. *Je-li*

$$\mathbf{A}^T\mathbf{A} = \mathbf{Y}\mathbf{\Lambda}\mathbf{Y}^T$$

spektrální rozklad matice $\mathbf{A}^T\mathbf{A}$ *a je-li*

$$\mathbf{A}\mathbf{Y} = \mathbf{Q}\mathbf{R}$$

QR rozklad matice $\mathbf{A}\mathbf{Y}$, *potom* $\mathbf{R}$ *je diagonální,* $\mathbf{Q}$, $\mathbf{Y}$ *jsou ortogonální a*

$$\mathbf{A} = \mathbf{Q}\mathbf{R}\mathbf{Y}^T$$

je SVD rozklad matice $\mathbf{A}$.

Poznámka 5.1.2. Pokud $m < n$, pak $\mathbf{A}^T$ splňuje předpoklady věty a je-li $\mathbf{A}^T = \mathbf{Q}\mathbf{R}\mathbf{Y}^T$ SVD rozklad matice $\mathbf{A}^T$, potom $\mathbf{A} = \mathbf{Y}\mathbf{R}^T\mathbf{Q}^T$ je SVD rozklad matice $\mathbf{A}$.

Algoritmus 9 SVD rozklad

Vstup: Matice $\mathbf{A} \in \mathbb{R}^{m\times n}, m \geq n$

Vypočti spektrální rozklad $\mathbf{A}^T\mathbf{A} = \mathbf{Y}\mathbf{\Lambda}\mathbf{Y}^T$ matice $\mathbf{A}^T\mathbf{A}$ (algoritmus 7)

Vypočti QR rozklad $\mathbf{A}\mathbf{Y} = \mathbf{Q}\mathbf{R}$ matice $\mathbf{A}\mathbf{Y}$ (algoritmus 8)

$\mathbf{A} = \mathbf{Q}\mathbf{R}\mathbf{Y}^T$ je SVD rozklad matice $\mathbf{A}$

Postup výpočtu SVD rozkladu shrnutý v algoritmu 9 zde byl uveden proto, že zároveň umožnil představit další dva často používané maticové rozklady a také ukázal souvislost SVD rozkladu s vlastními čísly a vlastními vektory matice. Z numerického hlediska má však své nevýhody. Ačkoli Jacobiho metoda umožňuje spočítat spektrální rozklad s relativně vysokou přesností, k určité ztrátě informace může dojít už samotnou konstrukcí matice $\mathbf{A}^T\mathbf{A}$ či $\mathbf{A}\mathbf{A}^T$ (komplikace nastávají, zejména je-li matice $\mathbf{A}$ špatně podmíněná, podrobněji v [10]). Standardně se tedy využívají přístupy, které nevyžadují explicitní vyjádření matice $\mathbf{A}^T\mathbf{A}$ či $\mathbf{A}\mathbf{A}^T$. Většina v praxi používaných algoritmů v prvním kroku transformuje matici $\mathbf{A}$ na bidiagonální tvar (tvar, ve kterém má matice obecně nenulovou pouze hlavní diagonálu a první naddiagonálu). Toto je nejčastěji provedeno pomocí

ortogonálních transformací (např. Householderových reflexí). V následujícím kroku je nalezen SVD rozklad bidiagonální matice, obvykle tzv. QR algoritmem (iterační algoritmus pro výpočet vlastních čísel založený na QR rozkladu). Jiný přístup k výpočtu zase nabízí modifikace Jacobiho metody pro SVD rozklad. Obecně umožňují algoritmy vypočíst SVD rozklad matice velmi přesně, na druhou stranu ale bývá nutné zaplatit vyšší časovou náročností. Podrobněji k výpočtu SVD rozkladu v [9], [10], [25].

Příklad 5.1.1. Nalezněme SVD rozklad matice $\mathbf{A}$ postupem uvedeným v algoritmu 9

$$\mathbf{A} = \begin{pmatrix} 2 & -3 & 4 \\ -1 & 5 & -2 \\ 8 & 1 & 1 \\ -6 & 3 & 5 \end{pmatrix}.$$

Řešení. Výpočet je proveden v MATLABu, výsledky jsou zaokrouhleny na 4 desetinná místa.

Nejprve je Jacobiho metodou (pro $\delta = 10^{-10}\|\mathbf{A}^T\mathbf{A}\|_F = 1,3964 \cdot 10^{-9}$) obdržen spektrální rozklad $\mathbf{A}^T\mathbf{A} = \mathbf{Y}\mathbf{\Lambda}\mathbf{Y}^T$ matice $\mathbf{A}^T\mathbf{A}$

$$\mathbf{Y} = \begin{pmatrix} 0,9499 & -0,0434 & 0,3095 \\ -0,2766 & -0,5776 & 0,7680 \\ -0,1455 & 0,8152 & 0,5607 \end{pmatrix},$$

$$\mathbf{\Lambda} = \begin{pmatrix} 112,9534 & 0 & -0,0000 \\ 0,0000 & 50,8902 & -0,0000 \\ -0,0000 & -0,0000 & 31,1564 \end{pmatrix}.$$

V druhém kroku je proveden QR rozklad matice $\mathbf{AY}$, $\mathbf{AY} = \mathbf{QR}$

$$\mathbf{Q} = \begin{pmatrix} 0,2021 & 0,6878 & 0,0999 & 0,6900 \\ -0,1922 & -0,6273 & 0,4316 & 0,6191 \\ 0,6753 & -0,0154 & 0,6817 & -0,2812 \\ -0,6828 & 0,3649 & 0,5823 & -0,2481 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 10,6280 & 0 & 0 \\ 0 & 7,1337 & 0 \\ 0 & 0 & 5,5818 \\ 0 & 0 & 0 \end{pmatrix}.$$

Výsledný SVD rozklad $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ má podobu

$$\mathbf{U} = \mathbf{Q} = \begin{pmatrix} 0,2021 & 0,6878 & 0,0999 & 0,6900 \\ -0,1922 & -0,6273 & 0,4316 & 0,6191 \\ 0,6753 & -0,0154 & 0,6817 & -0,2812 \\ -0,6828 & 0,3649 & 0,5823 & -0,2481 \end{pmatrix},$$

$$\mathbf{\Sigma} = \mathbf{R} = \begin{pmatrix} 10,6280 & 0 & 0 \\ 0 & 7,1337 & 0 \\ 0 & 0 & 5,5818 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{V} = \mathbf{Y} = \begin{pmatrix} 0,9499 & -0,0434 & 0,3095 \\ -0,2766 & -0,5776 & 0,7680 \\ -0,1455 & 0,8152 & 0,5607 \end{pmatrix}.$$

5.2 Aplikace SVD rozkladu

SVD rozklad je považován za jeden z nejuniverzálnějších a nejdůležitějších nástrojů maticových výpočtů, a nachází tak široké uplatnění nejen v numerické matematice. Výpočty využívající SVD rozklad jsou v důsledku použití ortogonálních transformací numericky stabilní, vlastní výpočet rozkladu však bývá časově náročnější. Na druhou stranu ale není vždy třeba počítat celý rozklad, některé aplikace například vyžadují znalost pouze singulárních čísel. V následujícím textu bude vždy předpokládán SVD rozklad matice $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ a nerostoucí uspořádání singulárních čísel dle věty 5.1.2. Zdrojové kódy funkcí a skriptů použitých při řešení demonstračních úloh jsou uvedeny v příloze A.3.2.

5.2.1 Vybrané vlastnosti matic

Hodnost a numerická hodnost matice

Při popisu kondenzovaného SVD rozkladu již bylo využito faktu, že hodnost r matice $\mathbf{A} \in \mathbb{R}^{m\times n}$ je rovna počtu jejích nenulových (kladných) singulárních čísel

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0,\ \sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_p = 0, \quad p = \min\{m, n\}.$$

Toto plyne z invariantnosti hodnosti vůči násobení regulární maticí (důkaz viz [21, s. 92]). Nechť $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ je SVD rozklad matice $\mathbf{A}$. Matice $\mathbf{U}$ a $\mathbf{V}$ jsou ortogonální,

tedy i regulární, a $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{\Sigma})$, jelikož je matice $\mathbf{\Sigma}$ v odstupňovaném tvaru, je její hodnost rovna počtu nenulových řádků (tj. počtu nenulových singulárních čísel).

Předchozí uvedené by však platilo pouze při použití exaktní aritmetiky. Při numerickém výpočtu na počítači lze předpokládat, že čísla $\sigma_{r+1}, \ldots, \sigma_p$ nebudou přesně rovna nule. Nemusí tak být vždy jasné, která singulární čísla mají skutečně být nulová a která jsou pouze blízko nuly. V tomto případě se zavádí pojem numerická hodnost matice. Matice má numerickou hodnost k, pokud má k singulárních čísel větších než zvolená tolerance $\delta > 0$, ostatní singulární čísla jsou považována za nulová. Určení hodnoty tolerance přitom není triviální, například MATLAB využívá výchozí hodnotu tolerance $\delta = \max\{m, n\}\ \varepsilon\ \|\mathbf{A}\|_2$, kde ε vyjadřuje strojovou přesnost a $\|\mathbf{A}\|_2$ značí spektrální normu (2-normu) matice $\mathbf{A}$ [20].

Frobeniova a spektrální norma matice

V části 2.1.3 byl uveden standardně používaný vztah pro výpočet Frobeniovy normy matice $\mathbf{A}$. Z věty o invariantnosti normy vůči ortogonálním transformacím (viz [21, s. 103]) vyplývá, že $\|\mathbf{A}\|_F = \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\|_F = \|\mathbf{\Sigma}\|_F$. Pro Frobeniovu normu matice $\mathbf{A}$ ($r = \text{rank}(\mathbf{A})$) lze tedy psát vztah

$$\|\mathbf{A}\|_F = \left(\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2\right)^{1/2}.$$

Další maticovou normou, kterou je možné zjistit na základě znalosti singulárních čísel, je 2-norma matice neboli spektrální norma. Tuto normu lze definovat pomocí euklidovské normy vektoru vztahem $\|\mathbf{A}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \|\mathbf{A}\boldsymbol{x}\|_2$. Stejně jako Frobeniova norma, je i spektrální norma invariantní vůči ortogonálním transformacím, tedy $\|\mathbf{A}\|_2 = \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\|_2 = \|\mathbf{\Sigma}\|_2$. Navíc platí, že spektrální norma diagonální matice je rovna v absolutní hodnotě největšímu diagonálnímu prvku této matice ([21, s. 90]). Při uspořádání singulárních čísel v matici $\mathbf{\Sigma}$ zavedeném ve větě o existenci SVD rozkladu je zřejmé, že tímto největším prvkem je číslo σ_1, z toho vyplývá, že

$$\|\mathbf{A}\|_2 = \sigma_1.$$

Jiná možnost odvození výše uvedeného vztahu vychází ze skutečnosti, že spektrální norma matice $\mathbf{A}$ je rovna odmocnině z největšího vlastního čísla matice $\mathbf{A}^T\mathbf{A}$, tj. číslu σ_1.

Číslo podmíněnosti matice

Číslo podmíněnosti regulární matice $\mathbf{A} \in \mathbb{R}^{n\times n}$ vzhledem k spektrální normě je definováno jako

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2. \tag{5.5}$$

Pokud má matice malé číslo podmíněnosti, je označována jako dobře podmíněná, v opačném případě jako špatně podmíněná. Pro singulární matici $\mathbf{A}$ se uvádí $\kappa_2(\mathbf{A}) = \infty$. O podmíněnosti se často hovoří v souvislosti s řešením soustav lineárních algebraických rovnic. Je-li úloha řešení soustavy $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, kde $\mathbf{A}$ je regulární, dobře podmíněná, pak „malé“ změně vstupních dat odpovídá „malá“ změna hodnot řešení na výstupu. Naopak u špatně podmíněné úlohy může i „malá“ změna vstupních dat způsobit „velkou“ změnu hodnot řešení (úloha je citlivá na perturbace vstupních dat). Číslo podmíněnosti regulární matice lze také interpretovat tak, že jeho převrácená hodnota vyjadřuje vzdálenost (relativní, ve spektrální normě) této matice od množiny singulárních matic.

Dle předchozí části lze vyjádřit spektrální normy ve vzorci (5.5) pomocí singulárních čísel matice $\mathbf{A}$, $\|\mathbf{A}\|_2 = \sigma_1$ a $\|\mathbf{A}^{-1}\|_2 = \sigma_n^{-1}$. Po dosazení pro číslo podmíněnosti platí

$$\kappa_2(\mathbf{A}) = \frac{\sigma_1}{\sigma_n}.$$

Další možnost vyjádření čísla podmíněnosti nabízí využití veličin

$$\operatorname{maxmag}(\mathbf{A}) \equiv \max_{\|\boldsymbol{x}\|_2=1} \|\mathbf{A}\boldsymbol{x}\|_2,$$

$$\operatorname{minmag}(\mathbf{A}) \equiv \min_{\|\boldsymbol{x}\|_2=1} \|\mathbf{A}\boldsymbol{x}\|_2.$$

Tímto způsobem lze rozšířit definici čísla podmíněnosti i obecně na obdélníkovou matici $\mathbf{A} \in \mathbb{R}^{m\times n}, m \geq n$ s plnou hodností ($\operatorname{rank}(\mathbf{A}) = n$) (podrobněji v [7])

$$\kappa_2(\mathbf{A}) = \frac{\operatorname{maxmag}(\mathbf{A})}{\operatorname{minmag}(\mathbf{A})} = \frac{\sigma_1}{\sigma_n}.$$

Číslo podmíněnosti obdélníkové matice s plnou hodností také někdy bývá popisováno vztahem využívajícím Moore-Penroseovu inverzi

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{\dagger}\|_2.$$

Ortonormální báze sloupcového a nulového prostoru matice

Pro každou matici $\mathbf{A} \in \mathbb{R}^{m\times n}$ lze definovat její sloupcový prostor jako

$$\mathcal{R}(\mathbf{A}) = \{\mathbf{A}\boldsymbol{x}; \boldsymbol{x} \in \mathbb{R}^n\} \subset \mathbb{R}^m$$

a její nulový prostor (jádro) jako

$$\mathcal{N}(\mathbf{A}) = \{\boldsymbol{x}; \mathbf{A}\boldsymbol{x} = \mathbf{0}\} \subset \mathbb{R}^n.$$

K nalezení ortonormálních bází těchto prostorů lze vhodně využít SVD rozklad. Platí následující věta, jejíž důkaz uvádí [21, s. 113].

Věta 5.2.1. *Je-li* $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ *SVD rozklad matice* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *a* $r = \mathrm{rank}(\mathbf{A})$*, potom:*

1. *prvních* r *sloupců matice* $\mathbf{U}$*,* $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$*, tvoří ortonormální bázi prostoru* $\mathcal{R}(\mathbf{A})$*,*
2. *posledních* $n-r$ *sloupců matice* $\mathbf{V}$*,* $\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n$*, tvoří ortonormální bázi prostoru* $\mathcal{N}(\mathbf{A})$*.*

Následující příklad shrnuje výpočet výše popsaných vlastností matic.

Příklad 5.2.1. Určeme hodnost, Frobeniovu a spektrální normu a číslo podmíněnosti matice $\mathbf{A}$ a ortonormální bázi $\mathcal{R}(\mathbf{A})$ a $\mathcal{N}(\mathbf{A})$,

$$\mathbf{A} = \begin{pmatrix} 1 & -4 & 3 & 6 \\ 2 & 5 & -1 & 1 \\ -4 & 2 & 7 & 0 \\ 3 & 5 & -2 & -6 \end{pmatrix}.$$

Řešení. Dle algoritmu 9 byl nalezen SVD rozklad matice $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, výsledky jsou zaokrouhleny na 4 desetinná místa,

$$\mathbf{U} = \begin{pmatrix} 0,6071 & -0,2103 & 0,4482 & 0,6215 \\ -0,2676 & 0,0629 & 0,8914 & -0,3602 \\ 0,2517 & 0,9655 & 0,0310 & 0,0584 \\ -0,7046 & 0,1398 & 0,0587 & 0,6932 \end{pmatrix},$$

$$\mathbf{\Sigma} = \begin{pmatrix} 11,8664 & 0 & 0 & 0 \\ 0 & 8,0248 & 0 & 0 \\ 0 & 0 & 5,0062 & 0 \\ 0 & 0 & 0 & 2,3934 \end{pmatrix},$$

$$\mathbf{V} = \begin{pmatrix} -0,2569 & -0,4395 & 0,4561 & 0,7299 \\ -0,5719 & 0,4718 & 0,6032 & -0,2941 \\ 0,4433 & 0,7209 & 0,1104 & 0,5211 \\ 0,6407 & -0,2539 & 0,6449 & -0,3304 \end{pmatrix}.$$

Hodnost matice $\mathbf{A}$ lze v tomto případě určit bez problémů, všechna její singulární čísla $\sigma_1, \ldots, \sigma_4$ jsou výrazně větší než nula, tedy

$$\operatorname{rank}(\mathbf{A}) = 4.$$

Pro zjištění Frobeniovy normy stačí dosadit do vztahu

$$\|\mathbf{A}\|_F = \left(\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_4^2\right)^{1/2} = 15,3623$$

a spektrální norma je rovna největšímu singulárnímu číslu

$$\|\mathbf{A}\|_2 = \sigma_1 = 11,8664.$$

Matice $\mathbf{A}$ je regulární a pro její číslo podmíněnosti vzhledem ke spektrální normě platí

$$\kappa_2(\mathbf{A}) = \frac{\sigma_1}{\sigma_n} = 4,9579.$$

Z regularity matice $\mathbf{A}$ také plyne, že soustava $\mathbf{A}\boldsymbol{x} = \mathbf{0}$ má jen triviální řešení $\boldsymbol{x} = \mathbf{0}$, tedy nulový prostor matice $\mathbf{A}$ obsahuje pouze nulový vektor, $\mathcal{N}(\mathbf{A}) = \{\mathbf{0}\}$, a jeho báze je prázdná množina.

Ortonormální báze sloupcového prostoru $\mathcal{R}(\mathbf{A})$ je tvořena sloupci $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_4$ matice $\mathbf{U}$, tj.

$$\boldsymbol{u}_1 = \begin{pmatrix} 0,6071 \\ -0,2676 \\ 0,2517 \\ -0,7046 \end{pmatrix}, \quad \boldsymbol{u}_2 = \begin{pmatrix} -0,2103 \\ 0,0629 \\ 0,9655 \\ 0,1398 \end{pmatrix}, \quad \boldsymbol{u}_3 = \begin{pmatrix} 0,4482 \\ 0,8914 \\ 0,0310 \\ 0,0587 \end{pmatrix}, \quad \boldsymbol{u}_4 = \begin{pmatrix} 0,6215 \\ -0,3602 \\ 0,0584 \\ 0,6932 \end{pmatrix}.$$

5.2.2 Aproximace maticí nižší hodnosti a komprese dat

V řadě aplikací může být užitečné k matici $\mathbf{A} \in \mathbb{R}^{m\times n}$ s hodností $\operatorname{rank}(\mathbf{A}) = r$ nalézt matici $\mathbf{A}_k \in \mathbb{R}^{m\times n}$ hodnosti $\operatorname{rank}(\mathbf{A}_k) = k < r$, která je v nějakém smyslu její nejlepší aproximací. Zde bude uvažována nejlepší aproximace ve smyslu minimalizace spektrální normy chyby $\mathbf{A} - \mathbf{A}_k$, tedy

$$\|\mathbf{A} - \mathbf{A}_k\|_2 = \min_{\substack{\mathbf{X}\in\mathbb{R}^{m\times n} \\ \operatorname{rank}(\mathbf{X})=k}} \|\mathbf{A} - \mathbf{X}\|_2. \tag{5.6}$$

Jak lze k vyřešení tohoto problému využít SVD rozklad, ukazuje věta 5.2.2, důkaz je uveden třeba v [7, s. 133]. K zápisu matice $\mathbf{A}_k$ je použit tzv. dyadický rozvoj. Pro ukázku, tato forma zápisu umožňuje například kondenzovaný SVD rozklad matice $\mathbf{A}$ vyjádřit jako

$$\mathbf{A} = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^T = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T. \tag{5.7}$$

Věta 5.2.2. *Nechť je dána matice $\mathbf{A} \in \mathbb{R}^{m\times n}$ hodnosti r a nechť k je přirozené číslo, $k < r$. Budiž uvažován SVD rozklad matice $\mathbf{A}$ zapsaný pomocí dyadického rozvoje (5.7). Potom je matice*

$$\mathbf{A}_k = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T$$

nejlepší aproximací matice $\mathbf{A}$ hodnosti k ve smyslu (5.6) a platí

$$\min_{\substack{\mathbf{X}\in\mathbb{R}^{m\times n} \\ \operatorname{rank}(\mathbf{X})=k}} \|\mathbf{A} - \mathbf{X}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}.$$

Jednou z aplikací, které mohou posloužit k demonstraci aproximace maticí nižší hodnosti, je komprese dat. Černobílý digitální obraz lze reprezentovat maticí typu $m \times n$, jejíž prvek na pozici ij odpovídá hodnotě intenzity ij-tého pixelu obrazu. Tato hodnota může být kódována například číslem z intervalu 0 (černá) až 1 (bílá) nebo celým číslem od 0 (černá) do 255 (bílá). Pro zjednodušení budou uvažovány pouze černobílé obrazy, v případě barevného obrazu by bylo možné analogicky aplikovat postup na každý z barevných kanálů zvlášť.

K ukázce byla využita vlastní fotografie Velkého náměstí v Hradci Králové (obrázek 3, vlevo nahoře), kterou lze reprezentovat maticí $\mathbf{A}$ typu 706×670. Dle věty 5.2.2 byly zkonstruovány matice $\mathbf{A}_{200}$, $\mathbf{A}_{80}$ a $\mathbf{A}_{20}$. Pro každou z těchto matic $\mathbf{A}_k$ stačí při využití kondenzovaného SVD rozkladu ukládat pouze hodnoty $\boldsymbol{u}_j$, $\boldsymbol{v}_j$ a σ_j pro $j = 1, \ldots, k$, což je $(m + n + 1)k$ hodnot. Pro původní matici $\mathbf{A}$ hodnosti r je nutné uchovávat $(m+n+1)r$ hodnot. Dochází tedy ke kompresi v poměru $\frac{r}{k}$. Paměťové nároky jednotlivých aproximací shrnuje tabulka 1.

Matice	**Co je ukládáno**	**Počet ukládaných čísel**	**Úspora paměti**
$\mathbf{A}$	$\mathbf{U}_{670}, \mathbf{V}_{670}, \sigma_{1,\ldots,670}$	$(706 + 670 + 1)670$	–
$\mathbf{A}_{200}$	$\mathbf{U}_{200}, \mathbf{V}_{200}, \sigma_{1,\ldots,200}$	$(706 + 670 + 1)200$	$70,15$ %
$\mathbf{A}_{80}$	$\mathbf{U}_{80}, \mathbf{V}_{80}, \sigma_{1,\ldots,80}$	$(706 + 670 + 1)80$	$88,06$ %
$\mathbf{A}_{20}$	$\mathbf{U}_{20}, \mathbf{V}_{20}, \sigma_{1,\ldots,20}$	$(706 + 670 + 1)20$	$97,01$ %

Tabulka 1: SVD komprese dvourozměrných dat

Z obrázku 3 je zřejmé, že s klesajícím k se však také postupně zhoršuje kvalita aproximací. Matici $\mathbf{A}_{200}$ lze z vizuálního hlediska považovat za „rozumnou“ aproximaci, kvalita aproximace $\mathbf{A}_{80}$ je již viditelně degradována a matice $\mathbf{A}_{20}$ připomíná původní fotografii poněkud vzdáleně, přesto však i při takto vysoké kompresi zůstávají výrazné struktury patrné.

Obrázek 3: Původní obraz a zkomprimované obrázky odpovídající aproximačním maticím pro $k = 200, 80, 20$

5.2.3 Návrat k Moore-Penroseově inverzi a problému nejmenších čtverců

V kapitole 4 byla Moore-Penroseova inverze počítána pomocí hodnostního rozkladu matice a podle Grevillova algoritmu. Další možnost, jak tuto zobecněnou inverzi matice nalézt, nabízí SVD rozklad.

Věta 5.2.3. *Nechť je dána matice* $\mathbf{A} \in \mathbb{R}^{m\times n}, r = \operatorname{rank}(\mathbf{A})$ *a nechť* $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^T$ *je její SVD rozklad resp. kondenzovaný SVD rozklad. Potom pro Moore-Penroseovu inverzi* $\mathbf{A}^\dagger$ *platí*

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T = \mathbf{V}_r\mathbf{\Sigma}_r^{-1}\mathbf{U}_r^T, \qquad \mathbf{\Sigma}^\dagger = \begin{pmatrix} \mathbf{\Sigma}_r^{-1} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n\times m}.$$

Není těžké dokázat (např. [7, s. 267]), že takto získaná Moore-Penroseova inverze splňuje všechny čtyři Penroseovy podmínky z věty 4.1.1. Matice $\mathbf{\Sigma}_r$ je regulární a její inverze má podobu

$$\mathbf{\Sigma}_r^{-1} = \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_r} \end{pmatrix}.$$

Příklad 5.2.2. Určeme Moore-Penroseovu inverzi matice $\mathbf{A}$ z příkladu 4.1.1 a 4.1.2 tentokrát pomocí SVD rozkladu,

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & 3 \\ 2 & -2 & -4 \end{pmatrix}.$$

Řešení. Matice $\mathbf{A}$ má SVD rozklad $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, výsledky jsou opět zaokrouhleny na 4 desetinná místa,

$$\mathbf{U} = \begin{pmatrix} 0,0882 & 0,4384 & -0,8944 \\ 0,7293 & 0,5832 & 0,3578 \\ -0,6785 & 0,6839 & 0,2683 \end{pmatrix},$$

$$\mathbf{\Sigma} = \begin{pmatrix} 6,5840 & 0 & 0 \\ 0 & 2,9412 & 0 \\ 0 & 0 & 0,0000 \end{pmatrix},$$

$$\mathbf{V} = \begin{pmatrix} -0,0819 & 0,8124 & -0,5774 \\ 0,6626 & 0,4771 & 0,5774 \\ 0,7445 & -0,3352 & -0,5774 \end{pmatrix}.$$

Například při stanovení hodnoty tolerance $\delta = \max\{m, n\}\ \varepsilon\ \|\mathbf{A}\|_2$ lze určit, že matice $\mathbf{A}$ má hodnost $r = 2$. Moore-Penroseova inverze $\mathbf{A}^\dagger$ je pak získána podle věty 5.2.3

$$\mathbf{A}^\dagger = \mathbf{V}_r \mathbf{\Sigma}_r^{-1} \mathbf{U}_r^T =$$

$$= \begin{pmatrix} -0,0819 & 0,8124 \\ 0,6626 & 0,4771 \\ 0,7445 & -0,3352 \end{pmatrix} \begin{pmatrix} 6,5840^{-1} & 0 \\ 0 & 2,9412^{-1} \end{pmatrix} \begin{pmatrix} 0,0882 & 0,4384 \\ 0,7293 & 0,5832 \\ -0,6785 & 0,6839 \end{pmatrix}^T =$$

$$= \begin{pmatrix} 0,1200 & 0,1520 & 0,1973 \\ 0,0800 & 0,1680 & 0,0427 \\ -0,0400 & 0,0160 & -0,1547 \end{pmatrix}.$$

Sekce 4.2 představila řešení úloh o nejmenších čtvercích v případě, že matice $\mathbf{A} \in \mathbb{R}^{m \times n}, m \geq n$, měla plnou sloupcovou hodnost. V opačném případě, kdy $\operatorname{rank}(\mathbf{A}) < n$, je matice $\mathbf{A}^T\mathbf{A}$ singulární a soustava normálních rovnic $\mathbf{A}^T\mathbf{A}\boldsymbol{x} = \mathbf{A}^T\boldsymbol{b}$, a tedy i příslušná úloha o nejmenších čtvercích, bude mít nekonečně mnoho řešení. V aplikacích však většinou bývá cílem získat pouze jedno řešení, a tak se obvykle přidává požadavek, aby samo toto řešení $\boldsymbol{x}$ mělo minimální euklidovskou normu. Formulace úlohy tak může být mírně upravena.

Nechť $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$. Úloha spočívá v určení $\boldsymbol{x} \in \mathbb{R}^n$ takového, aby hodnota

$$\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|_2$$

byla minimální a zároveň

$$\|\boldsymbol{x}\|_2$$

byla minimální (mezi všemi $\boldsymbol{x}$, která vyhovují předchozí podmínce).

Takto formulovanou úlohu o nejmenších čtvercích lze řešit následujícím způsobem. Budiž předpokládáno, že $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ je SVD rozklad matice $\mathbf{A}$ a stále platí $m \geq n$. Matice $\mathbf{U}$ a tedy i $\mathbf{U}^T$ je ortogonální a při využití faktu, že násobení ortogonální maticí nemění euklidovskou normu, platí

$$\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|_2 = \|\mathbf{U}^T(\boldsymbol{b} - \mathbf{A}\boldsymbol{x})\|_2 = \|\mathbf{U}^T\boldsymbol{b} - \mathbf{U}^T\mathbf{A}\boldsymbol{x}\|_2 = \|\mathbf{U}^T\boldsymbol{b} - \mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\boldsymbol{x}\|_2 =$$
$$= \|\mathbf{U}^T\boldsymbol{b} - \mathbf{\Sigma}\mathbf{V}^T\boldsymbol{x}\|_2.$$

Z toho vyplývá, že $\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|_2$ je minimální právě tehdy, když $\|\mathbf{U}^T\boldsymbol{b} - \boldsymbol{\Sigma}\mathbf{V}^T\boldsymbol{x}\|_2$ je minimální. Bude-li zavedeno označení $\boldsymbol{c} = \mathbf{U}^T\boldsymbol{b}$ a $\boldsymbol{y} = \mathbf{V}^T\boldsymbol{x}$, pak lze na problém pohlížet jako na řešení úlohy o nejmenších čtvercích pro $\boldsymbol{\Sigma}, \boldsymbol{c}$.
Jestliže $\boldsymbol{y}$ minimalizuje

$$\|\boldsymbol{c} - \boldsymbol{\Sigma}\boldsymbol{y}\|_2,$$

pak

$$\boldsymbol{x} = \mathbf{V}\boldsymbol{y}$$

minimalizuje

$$\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|_2.$$

Aby byla norma $\|\boldsymbol{c} - \boldsymbol{\Sigma}\boldsymbol{y}\|_2$ minimální, musí pro prvních $r = \mathrm{rank}(\mathbf{A})$ složek vektoru $\boldsymbol{y}$ platit

$$y_i = \frac{c_i}{\sigma_i} \qquad \text{pro } i = 1, \ldots, r.$$

Má-li matice $\mathbf{A}$ plnou sloupcovou hodnost ($r = n$), je již vektor $\boldsymbol{y}$ určen jednoznačně a stejně tak i řešení $\boldsymbol{x} = \mathbf{V}\boldsymbol{y}$ počáteční úlohy. Pokud matice $\mathbf{A}$ nemá plnou sloupcovou hodnost ($r < n$), lze složky $y_{r+1}, \ldots, y_n$ volit libovolně. Jestliže je však požadováno, aby řešení $\boldsymbol{x}$ mělo minimální euklidovskou normu, musí i vektor $\boldsymbol{y}$ mít tuto normu minimální (jelikož $\mathbf{V}$ je ortogonální, je $\|\boldsymbol{x}\|_2 = \|\mathbf{V}\boldsymbol{y}\|_2 = \|\boldsymbol{y}\|_2$). Je zřejmé, že toto nastane, je-li

$$y_{r+1} = y_{r+2} = \cdots = y_n = 0.$$

Celý postup řešení úlohy o nejmenších čtvercích v obecném případě je možné shrnout následovně: Nejprve je položeno

$$\boldsymbol{c} = \mathbf{U}^T\boldsymbol{b},$$

dále je zkonstruován vektor $\boldsymbol{y}$ tak, že

$$y_i = \frac{c_i}{\sigma_i} \quad \text{pro } i = 1, \ldots, r \qquad \text{a} \qquad y_i = 0 \quad \text{pro } i = r+1, \ldots, n,$$

hledané řešení $\boldsymbol{x}$ má tvar

$$\boldsymbol{x} = \mathbf{V}\boldsymbol{y}.$$

Bude-li využito Moore-Penroseovy inverze, lze vektor $\boldsymbol{y}$ vyjádřit jako $\boldsymbol{y} = \boldsymbol{\Sigma}^\dagger\boldsymbol{c}$. Poté lze řešení $\boldsymbol{x}$ úlohy o nejmenších čtvercích v obecném případě psát ve tvaru

$$\boldsymbol{x} = \mathbf{V}\boldsymbol{y} = \mathbf{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{c} = \mathbf{V}\boldsymbol{\Sigma}^\dagger\mathbf{U}^T\boldsymbol{b} = \mathbf{A}^\dagger\boldsymbol{b}.$$

Budiž poznamenáno, že celý výpočet je možné také provést pouze s využitím kondenzovaného SVD rozkladu, podrobné odvození postupu je k nahlédnutí v [5].

5.2.4 Ill-posed problémy a regularizace

Další v práci popsaná aplikace SVD rozkladu bude demonstrována na rekonstrukci rozmazaného obrazu. Opět bude předpokládán černobílý obraz reprezentovaný maticí tentokrát značenou $\mathbf{X} \in \mathbb{R}^{m \times n}$. Tato matice bude odpovídat ideálnímu ostrému obrazu, a tedy hledanému řešení při rekonstrukci (obrázek 5(a)). V praxi je však k dispozici pouze matice $\mathbf{B} \in \mathbb{R}^{m \times n}$, která představuje obraz degradovaný rozmazáním, například v důsledku pohybu či špatně zaostřené optické soustavy fotoaparátu (obrázek 5(b)). Pokud je rozmazání lineární a nezávislé po řádcích a sloupcích, lze jej reprezentovat maticovou rovnicí

$$\mathbf{A}_C \mathbf{X} \mathbf{A}_R^T = \mathbf{B}. \tag{5.8}$$

Matice $\mathbf{A}_C \in \mathbb{R}^{m \times m}$ představuje proces rozmazání po sloupcích a $\mathbf{A}_R \in \mathbb{R}^{n \times n}$ po řádcích.

Rovnici (5.8) lze také zapsat v obvyklé podobě soustavy lineárních algebraických rovnic

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \tag{5.9}$$

kde $\mathbf{A} = \mathbf{A}_R \otimes \mathbf{A}_C \in \mathbb{R}^{mn \times mn}$ je Kroneckerův součin matic $\mathbf{A}_R$ a $\mathbf{A}_C$. Vektor $\boldsymbol{x}$ vznikl umístěním sloupců matice $\mathbf{X}$ pod sebe, matematicky $\boldsymbol{x} = \mathrm{vec}(\mathbf{X}) = (\mathbf{X}_{\bullet 1}^T, \ldots, \mathbf{X}_{\bullet n}^T)^T \in \mathbb{R}^{mn}$ a obdobně $\boldsymbol{b} = \mathrm{vec}(\mathbf{B}) = (\mathbf{B}_{\bullet 1}^T, \ldots, \mathbf{B}_{\bullet n}^T)^T \in \mathbb{R}^{mn}$.

Obecně, je-li proces rozmazání lineární, lze ho reprezentovat zápisem (5.9) (ne vždy ale musí být toto rozmazání separovatelné po řádcích a sloupcích, a tedy někdy nemusí být možné ho přesně vyjádřit ve tvaru (5.8)). Na součin $\mathbf{A}\boldsymbol{x}$ lze také nahlížet jako na reprezentaci konvoluce ostrého obrazu $\mathbf{X}$ s tzv. rozptylovou funkcí bodu (point spread function, PSF) a je možné ukázat, že matice $\mathbf{A}$ může mít v závislosti na dalších podmínkách určitou specifickou strukturu, které lze využít např. při hledání jejího SVD rozkladu, podrobněji v [12]. V případě reálných úloh není skutečná matice $\mathbf{A}$ přesně známa, ale pracuje se s jejím odhadem odvozeným na základě matematického modelu popisujícího fyzikální proces, který rozmazání způsobil.

Při práci s reálnými (nejen obrazovými) daty se velice často stává, že vektor $\boldsymbol{b}$ obsahuje šum (nepřesnosti měření, zaokrouhlovací chyby, chyby diskretizace apod.). Tedy platí

$$\boldsymbol{b} = \boldsymbol{b}_{EXACT} + \boldsymbol{b}_{NOISE},$$

kde $\boldsymbol{b}_{EXACT}$ je přesná pravá strana soustavy a $\boldsymbol{b}_{NOISE}$ je vektor šumu. Bude předpokládáno, že $\|\boldsymbol{b}_{EXACT}\|_2 \gg \|\boldsymbol{b}_{NOISE}\|_2$.

Pokud $\text{rank}(\mathbf{A}_C) = m$ a $\text{rank}(\mathbf{A}_R) = n$, je matice $\mathbf{A}$ regulární a řešení soustavy (5.9), obvykle označované jako tzv. naivní, lze jednoduše zapsat jako $\boldsymbol{x}_{NAIVE} = \mathbf{A}^{-1}\boldsymbol{b}$. Toto řešení by odpovídalo požadovanému řešení $\boldsymbol{x}_{EXACT} = \mathbf{A}^{-1}\boldsymbol{b}_{EXACT}$, pokud by vektor $\boldsymbol{b}$ neobsahoval šum. V případě, že vektor $\boldsymbol{b}$ šum obsahuje, pro $\boldsymbol{x}_{NAIVE}$ platí

$$\boldsymbol{x}_{NAIVE} = \mathbf{A}^{-1}\boldsymbol{b} = \boldsymbol{x}_{EXACT} + \mathbf{A}^{-1}\boldsymbol{b}_{NOISE}. \tag{5.10}$$

Rekonstrukce rozmazaného obrazu je typickým příkladem tzv. ill-posed problému. Tyto problémy spočívají v řešení soustavy lineárních algebraických rovnic $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ (resp. úlohy o nejmenších čtvercích), kde matice soustavy $\mathbf{A}$ má vysoké číslo podmíněnosti a její singulární čísla postupně (bez výrazných mezer) klesají k nule. Vektor pravých stran $\boldsymbol{b}$ navíc obvykle bývá zatížen šumem, na který je soustava ze své podstaty velmi citlivá. S ill-posed problémy se lze při řešení reálných úloh setkat poměrně často.

Z obrázku 5(c) je patrné, že naivnímu řešení (5.10) dominuje inverzní šum $\mathbf{A}^{-1}\boldsymbol{b}_{NOISE}$ a zcela překrývá hledané řešení $\boldsymbol{x}_{EXACT}$. Řešení $\boldsymbol{x}_{NAIVE}$ lze pomocí SVD rozkladu matice $\mathbf{A}$ vyjádřeného v podobě dyadického rozvoje zapsat jako

$$\boldsymbol{x}_{NAIVE} = \sum_{j=1}^{mn} \frac{\boldsymbol{u}_j^T \boldsymbol{b}_{EXACT}}{\sigma_j}\boldsymbol{v}_j + \sum_{j=1}^{mn} \frac{\boldsymbol{u}_j^T \boldsymbol{b}_{NOISE}}{\sigma_j}\boldsymbol{v}_j. \tag{5.11}$$

Dále se předpokládá, že vektor $\boldsymbol{b}_{EXACT}$ splňuje tzv. diskrétní Picardovu podmínku, tedy zjednodušeně řečeno, že s rostoucím j velikost komponent $\boldsymbol{u}_j^T\boldsymbol{b}_{EXACT}$ klesá k nule v průměru rychleji než velikost odpovídajících singulárních čísel σ_j, více v [7]. Vektor šumu $\boldsymbol{b}_{NOISE}$ tuto podmínku splňovat nemusí. Hodnoty $\boldsymbol{u}_j^T\boldsymbol{b}_{NOISE}$ k nule obvykle neklesají a v důsledku dělení malými singulárními čísly dojde k zesílení jejich vlivu. V naivním řešení (5.11) tak dominuje druhá suma, to vysvětluje výsledek na obrázku 5(c).

Způsobem, jakým lze ill-posed problémy řešit, je regularizace. Cílem je nalézt nějakou aproximaci přesného řešení $\boldsymbol{x}_{EXACT}$ tak, aby byl potlačen vliv šumu. Jednou z klasických regularizačních metod je Truncated SVD (TSVD). TSVD regularizace je založena na nahrazení matice $\mathbf{A}$ její nejlepší aproximací nižší hodnosti $\mathbf{A}_k$ dle věty 5.2.2

$$\mathbf{A}_k = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \qquad k < mn.$$

Příslušné řešení problému (5.9) ve smyslu nejmenších čtverců má pak tvar

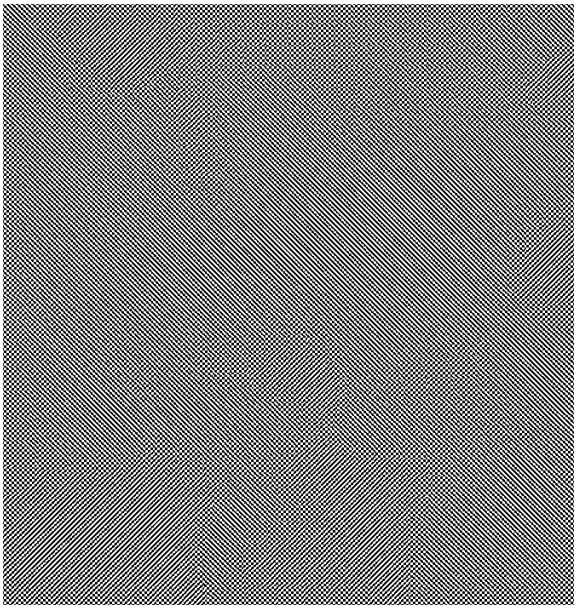
$$\boldsymbol{x}_{REG(k)} = \mathbf{A}_k^{\dagger}\boldsymbol{b} = \sum_{j=1}^{k} \frac{\boldsymbol{u}_j^T \boldsymbol{b}}{\sigma_j}\boldsymbol{v}_j.$$

Složky řešení nejvíce dominované šumem, odpovídající dělení malými singulárními čísly $\sigma_j, j = k+1, \ldots, mn$, jsou jednoduše odstraněny.

Otázkou zůstává volba vhodné hodnoty k, tzv. regularizačního parametru. S klesající hodnotou k je více potlačován vliv šumu, zároveň však dochází ke ztrátě části informace. V praxi se pro určení regularizačního parametru používají různé techniky, například princip diskrepance, generalized cross validation či L-křivky. Podrobněji k tomuto, k dalším metodám regularizace i obecně k problematice rekonstrukce rozmazaného obrazu v [12]. Obrázek 5(d) – (f) ukazuje řešení odpovídající různým volbám regularizačního parametru.

(a)

(b)



(c)

(d)

(e)

(f)

Obrázek 4: Rekonstrukce rozmazaného obrazu: (a) původní obraz (přesné řešení), (b) rozmazaný obraz s aditivním šumem, (c) naivní řešení, (d) řešení při vhodně zvoleném regularizačním parametru (k = 12 693), (e) podregularizované řešení s patrným vysokofrekvenčním šumem (k = 15 351), (f) přeregularizované řešení postrádající část vysokofrekvenční informace (šum, ale i detaily obrazu) (k = 3 602)

5.2.5 Analýza hlavních komponent

Analýza hlavních komponent (principal component analysis, PCA) je statistická metoda sloužící k redukci dimenze vícerozměrných dat a je také často používaným nástrojem pro prvotní poznání a pochopení dat předcházející vlastnímu řešení vícerozměrných úloh. Při statistických výzkumech může být počet sledovaných a zpracovávaných proměnných vysoký, cílem analýzy hlavních komponent je nalézt menší počet nových proměnných (tzv. hlavních komponent), kterými by bylo možné nahradit původní proměnné tak, aby nedošlo k výrazné ztrátě informace, přesněji, aby byla co nejvíce zachována variabilita původních dat. Nově nalezené hlavní komponenty jsou vzájemně nekorelované lineární kombinace původních proměnných.

Postup analýzy hlavních komponent a souvislost se spektrálním rozkladem a SVD rozkladem matice budou předvedeny na úloze rozpoznávání tváří (známé také jako „eigenfaces“), jak je představil Turk a Pentland v [28]. Vysvětlení použitých statistických pojmů lze nalézt například v [13], podrobnější popis analýzy hlavních komponent

uvádí [14]. K praktické demonstraci popsané metody budou použity fotografie z databáze ORL (Olivetti Research Laboratory) [23]. Databáze obsahuje 10 různých snímků tváře 40 jedinců, jednotlivé snímky téhož jedince se liší například výrazem tváře, osvětlením, přítomností brýlí apod.

Budiž předpokládána (tréninková) množina sestávající z m obrazů tváří různých jedinců $\mathbf{\Gamma} = \{\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, \ldots, \mathbf{\Gamma}_m\}$, kde je každý obraz o rozměrech $p \times q = n$ pixelů reprezentován vektorem $\mathbf{\Gamma}_j \in \mathbb{R}^n$, neboli jinak řečeno bodem v n-rozměrném euklidovském prostoru. Nejprve je vypočítán vektor průměru $\mathbf{\Psi}$ všech obrazů

$$\mathbf{\Psi} = \frac{1}{m} \sum_{j=1}^{m} \mathbf{\Gamma}_j,$$

a následně vektory $\mathbf{\Phi}_j$ vyjadřující rozdíl každého z obrazů od tohoto průměru

$$\mathbf{\Phi}_j = \mathbf{\Gamma}_j - \mathbf{\Psi}, \quad j = 1, \ldots, m.$$

Na obrázku 5 je příklad několika fotografií z tréninkové množiny a průměr všech obrazů tréninkové množiny.

Obrázek 5: Ukázka obrazů z tréninkové množiny a průměrný obraz této množiny $\mathbf{\Psi}$

Vektory $\mathbf{\Phi}_j$ jsou vstupem do analýzy hlavních komponent, která hledá množinu ortonormálních vektorů $\boldsymbol{u} \in \mathbb{R}^n$ co nejlépe popisujících rozložení variability vstupních dat. Vektor $\boldsymbol{u}_k$ je volen tak, aby hodnota

$$\lambda_k = \frac{1}{m} \sum_{j=1}^{m} (\boldsymbol{u}_k^T \mathbf{\Phi}_j)^2$$

byla maximální a zároveň

$$\boldsymbol{u}_l^T \boldsymbol{u}_k = \begin{cases} 1 & \text{je-li } l = k \\ 0 & \text{je-li } l \neq k. \end{cases}$$

Je možné ukázat, že vektory $\boldsymbol{u}_k$ a skaláry λ_k jsou vlastní vektory resp. vlastní čísla kovarianční matice $\mathbf{C}$, kterou lze pro tuto úlohu definovat jako

$$\mathbf{C} = \frac{1}{m}\sum_{j=1}^{m} \boldsymbol{\Phi}_j \boldsymbol{\Phi}_j^T = \mathbf{A}\mathbf{A}^T,$$

kde $\mathbf{A} = \begin{pmatrix} \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 & \cdots & \boldsymbol{\Phi}_m \end{pmatrix}$. Kovarianční matice je symetrická a pro určení jejích vlastních vektorů a vlastních čísel je možné využít spektrální rozklad dle věty 5.1.3, platí

$$\mathbf{C} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T,$$

kde $\boldsymbol{\Lambda}$ je matice, jejíž diagonální prvky jsou sestupně uspořádaná vlastní čísla λ_k kovarianční matice a sloupce matice $\mathbf{U}$ jsou vlastní vektory $\boldsymbol{u}_k$ příslušející k těmto vlastním číslům.

Kovarianční matice má rozměr $n \times n$, což například pro obrazy tváří o rozměru 100×100 představuje matici typu $10\,000 \times 10\,000$. Možností, jak se vyhnout nutnosti konstrukce kovarianční matice a výpočtu jejího spektrálního rozkladu, je využití již zmíněného faktu, že vlastní vektory této matice $\mathbf{C} = \mathbf{A}\mathbf{A}^T$ jsou zároveň levými singulárními vektory matice $\mathbf{A}$ a její vlastní čísla jsou druhými mocninami singulárních čísel matice $\mathbf{A}$. Pro určení hledaných vektorů $\boldsymbol{u}_k$ (resp. matice $\mathbf{U}$) tedy stačí vypočítat SVD rozklad matice $\mathbf{A}$

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T.$$

V článku [28] autoři předpokládají, že při řešení této úlohy bývá $m < n$, z toho vyplývá, že nejvýše prvních m vlastních čísel je nenulových, a tedy nemá význam dále pracovat s ostatními vlastními vektory příslušejícími k nulovým vlastním číslům. Dále z logiky analýzy hlavních komponent vyplývá, že vstupní množinu dat (obrazů) by bylo možné popsat i za využití menšího počtu $p < m$ vlastních vektorů (tedy pomocí p hlavních komponent místo původních n proměnných). Vlastní čísla λ_k vyjadřují variabilitu (rozptyl) k-té hlavní komponenty definované pomocí vlastního vektoru $\boldsymbol{u}_k$. Jelikož jsou vlastní čísla uspořádaná sestupně, je zřejmé, že první hlavní komponenta pokrývá největší část celkové variability dat a s rostoucím k se význam komponenty na celkové variabilitě snižuje. Jednou z možných technik, jak určit vhodné p, je požadovat, aby byla zachována určitá minimální část h variability původních dat neboli aby

$$\frac{\sum_{k=1}^{p} \lambda_k}{\sum_{k=1}^{m} \lambda_k} > h.$$

Dále už stačí pracovat pouze s prvními p vlastními vektory $\boldsymbol{u}_k$ $(k = 1, \ldots, p)$. Tyto vektory nazývané jako „eigenfaces“ (viz obrázek 6) tvoří bázi prostoru, tzv. prostoru

tváří, přičemž dimenze p tohoto prostoru je výrazně nižší než dimenze n prostoru původních dat.

Obrázek 6: Vizualizace prvních pěti vlastních vektorů („eigenfaces“) $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_5$

Následně jsou všechny vstupní obrazy $\boldsymbol{\Gamma}_j,\ j = 1, \ldots, m$, promítnuty do prostoru tváří a pro každý je vypočten vektor $\boldsymbol{\Omega}_j = \begin{pmatrix} \omega_{j_1} & \omega_{j_2} & \cdots & \omega_{j_p} \end{pmatrix}^T$, jehož složky

$$\omega_{j_k} = \boldsymbol{u}_k^T(\boldsymbol{\Gamma}_j - \boldsymbol{\Psi}) = \boldsymbol{u}_k^T \boldsymbol{\Phi}_j \tag{5.12}$$

představují jednotlivá komponentní skóre, která vyjadřují příspěvek k-tého vlastního vektoru k reprezentaci j-tého obrazu tváře.

Identifikace jedince je pak klasickou úlohou rozpoznávání. Pro nový testovací obraz $\boldsymbol{\Gamma}_t$ je podle vztahu (5.12) určen vektor $\boldsymbol{\Omega}_t$ a následně je vypočtena euklidovská vzdálenost tohoto vektoru od vektorů $\boldsymbol{\Omega}_j,\ j = 1, \ldots, m$, všech obrazů z tréninkové množiny

$$\epsilon_j = \|\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_j\|_2.$$

Obraz je přiřazen k jedinci i, pokud vzdálenost ϵ_i je minimální ze všech ϵ_j a zároveň ϵ_i je menší než nějaká předem stanovená hranice pro rozpoznání θ_ϵ.

Jelikož více různých obrazů, které ani nemusejí být obrazy tváří, může být promítnuto na stejný vektor $\boldsymbol{\Omega}$, je vhodné pro testovací obraz vypočíst jeho vzdálenost od prostoru tváří

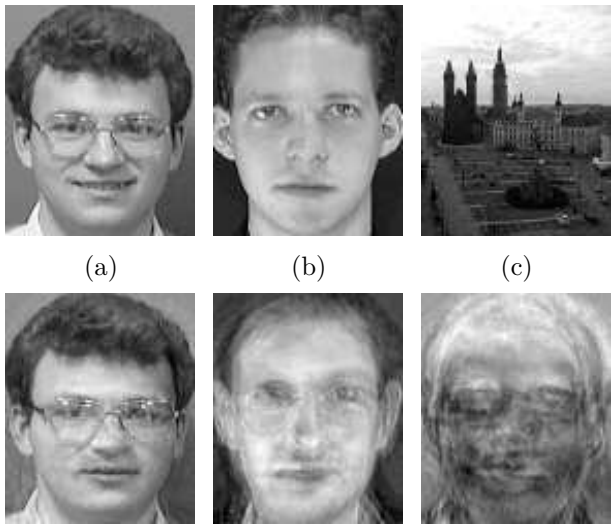
$$\delta = \|\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_f\|_2,$$

kde $\boldsymbol{\Phi}_t = \boldsymbol{\Gamma}_t - \boldsymbol{\Psi}$ a $\boldsymbol{\Phi}_f = \sum_{k=1}^{p} \omega_{t_k} \boldsymbol{u}_k$, a stanovit určitou hranici θ_δ, při které obraz již nebude považován za obraz tváře.

Nyní může nastat některá z následujících situací:

$\epsilon_i < \theta_\epsilon$ a $\delta < \theta_\delta$ obraz je identifikován jako patřící i-tému jedinci,
$\epsilon_i > \theta_\epsilon$ a $\delta < \theta_\delta$ jedná se o obraz tváře neznámého jedince,
$\delta > \theta_\delta$ nejedná se o obraz tváře.

V souvislosti s prvním a druhým případem je zřejmé, že s klesající hodnotou θ_ϵ bude růst přesnost rozpoznaných jedinců, ale zároveň bude více tváří zůstávat jako neznámých. Výpočet vzdálenosti od prostoru tváří a aplikace hranice θ_δ umožňuje tuto metodu vhodně použít nejen pro vlastní rozpoznávání tváří, ale i jako prostředek pro určení, zda je nějaký obraz obrazem tváře či nikoli. Hodnoty obou hranic θ_ϵ a θ_δ bývají určovány experimentálně. Příklady uvedených situací ilustruje obrázek 7.



Obrázek 7: Tři testovací obrazy a rekonstrukce jejich projekcí do prostoru tváří: (a) jedinec nacházející se v tréninkové množině, (b) neznámý jedinec, který nebyl v tréninkové množině, (c) fotografie, která není obrazem tváře

Tabulka 2 shrnuje hodnoty vzdáleností ϵ a δ pro tyto obrazy. Testovací obraz (a) byl správně identifikován jako jedinec z tréninkové množiny v experimentu označený číslem 5. Minimální vzdálenost ϵ obrazu (b) je relativně vyšší v porovnání s hodnotami správně rozpoznaných testovacích obrazů jedinců, kteří byly v tréninkové množině, při vhodně zvolené hranici θ_ϵ by tak byl tento obraz označen jako neznámý. Vysoká hodnota vzdálenosti δ obrazu (c) indikuje, že se nejedná o obraz tváře.

Testovací obraz t	$\epsilon = \min_{j=1,\ldots,m} \|\mathbf{\Omega}_t - \mathbf{\Omega}_j\|_2$	$\delta = \|\mathbf{\Phi}_t - \mathbf{\Phi}_f\|_2$
(a)	$3,6715$	$8,3403$
(b)	$11,7247$	$9,8888$
(c)	$25,1906$	$20,3478$

Tabulka 2: Hodnoty vzdáleností ϵ a δ pro testovací obrazy z obrázku 7 (zaokrouhleno na 4 desetinná místa)

Představená metoda pro rozpoznávání tváří byla v této práci otestována jednoduchým experimentem. Do tréninkové množiny bylo zařazeno 39 jedinců z databáze ORL a každý v ní byl reprezentován první z deseti fotografií. Testovací množina obsahovala vždy druhou z desítky fotografií každého jedince. Hranice θ_ϵ a θ_δ nebyly uvažovány, resp. byly považovány za nekonečné, neboť všechny obrazy v testovací množině bylo možné přiřadit k některému jedinci z tréninkové množiny. Výsledkem bylo 31 správně rozpoznaných jedinců z celkového počtu 39 obrazů v testovací množině, což odpovídá úspěšnosti přibližně 79,5 %.

Popsaná metoda rozpoznávání tváří založená na analýze hlavních komponent má určité nedostatky, které se projevily i v provedeném experimentu. Rozpoznávání je citlivé na výrazné změny světelných podmínek při pořizování fotografií, změny v pozici a natočení tváře, zakrytí části tváře například brýlemi apod. Způsobem, jak částečně potlačit či eliminovat důsledky působení těchto vlivů, a tím zvýšit přesnost obdržených výsledků, by mohlo být předzpracování (tedy jakási normalizace) všech obrazů před vlastním provedením metody nebo zahrnutí více odlišných fotografií každého jedince do tréninkové množiny [26]. „Eigenfaces“ jsou klasickou metodou pro rozpoznávání tváří, nicméně od roku 1991, kdy ji Turk a Pentland publikovali, bylo navrženo množství jejích vylepšení i metod nových.

FSVDR reprezentace tváří

Liu, Chen a Tan [17] navrhli způsob reprezentace obrazů tváří nazvaný FSVDR (fractional order singular value decomposition representation). Využitím této reprezentace tváří lze dosáhnout lepších výsledků rozpoznávání zejména v případech, kdy se fotografie v tréninkové a testovací množině liší podmínkami osvětlení nebo i částečným zakrytím tváře. FSVDR reprezentaci je možné použít kromě metod vycházejících z analýzy hlavních komponent i při jiných metodách rozpoznávání tváří.

Standardně je každá černobílá fotografie tváře o rozměru $p \times q$ reprezentována maticí obsahující hodnoty intenzit jednotlivých pixelů. Nechť je tato matice označena $\mathbf{G} \in \mathbb{R}^{p \times q}$ a nechť $\mathbf{G} = \mathbf{U}_G \mathbf{\Sigma}_G \mathbf{V}_G^T$ je její SVD rozklad. FSVDR spočívá v nahrazení matice $\mathbf{G}$ maticí

$$\mathbf{B} = \mathbf{U}_G \mathbf{\Sigma}_G^\alpha \mathbf{V}_G^T,$$

$0 \leq \alpha \leq 1$. Poté, co jsou takto upraveny všechny obrazy v tréninkové i testovací množině, je již úloha rozpoznávání řešena klasicky, například metodou „eigenfaces“. Podrobnější odvození FSVDR je uvedeno v [17].

Obrázek 8 ilustruje FSVDR reprezentace dvou obrazů tváře lišících se směrem osvětlení pro různé hodnoty parametru α. Vizuálně je patrné, že s klesající hodnotou α jsou rozdíly mezi obrazy méně znatelné. Fotografie pochází z databáze Yale [2].

$\alpha = 1$ $\alpha = 0,7$ $\alpha = 0,4$ $\alpha = 0,1$

Obrázek 8: Dva obrazy tváře s odlišným osvětlením a jejich FSVDR reprezentace ($\alpha = 1$ odpovídá původnímu obrazu)

Volba parametru α je klíčová. Hodnota obvykle bývá určována experimentálně pro potřeby konkrétní úlohy. Autoři v [17] uvádějí, že pro metody rozpoznávání založené na analýze hlavních komponent bývá obecně nejlepších výsledků dosaženo při $\alpha = 0,1$. Toto se potvrdilo i v níže popsaném experimentu.

Databáze Yale obsahuje fotografie tváří 15 jedinců, každý jedinec je zastoupen 11 různými fotografiemi. Pro účely zde provedeného experimentu byly z databáze vybrány pro každého jedince dvě fotografie lišící se světelnými podmínkami, přičemž jedna z nich byla zařazena do tréninkové množiny a druhá do testovací. Všechny fotografie byly oříznuty a bylo změněno jejich měřítko. V prvním pokusu byla ponechána standardní reprezentace fotografií a rozpoznávání bylo provedeno metodou „eigenfaces“ tak, jak byla představena v předchozí části textu. Správně byl identifikován pouze jeden jedinec. V druhém pokusu byla pro obrazy tváří použita FSVDR reprezentace (s parametrem $\alpha = 0,1$), ostatní postup zůstal stejný. Výsledkem druhého pokusu bylo 11 správně identifikovaných jedinců z celkového počtu 15, neboli úspěšnost přibližně 73,3 %.

6 Shrnutí výsledků

Práce představila vybrané maticové rozklady, předvedla možnosti jejich aplikace a zmínila i možné algoritmy pro jejich výpočet. Hlavní pozornost byla věnována LU rozkladu, Moore-Penroseově inverzi a SVD rozkladu, v souvislosti s nimi však byly popsány i další rozklady – hodnostní rozklad, spektrální rozklad a QR rozklad.

Pro každý z rozkladů byl uveden algoritmus jeho výpočtu, který byl následně implementován v MATLABu a otestován. Nutné je však podotknout, že pro každý rozklad obvykle existuje více různých postupů výpočtu, které se liší svými numerickými charakteristikami. V práci uvedené algoritmy jsou vhodné pro teoretický výklad rozkladů a ukázání souvislostí s dalšími matematickými koncepty, pro reálný výpočet však nemusejí být tou nejlepší variantou. Tyto skutečnosti byly nastíněny již ve vlastním textu a současně byla zmíněna i některá alternativní řešení. Příkladem může být výpočet SVD rozkladu využívající spektrální rozklad a QR rozklad. Algoritmus ukázal souvislost SVD rozkladu s vlastními čísly a vlastními vektory matice, pro praktické použití nicméně vhodný není, neboť výpočet spektrálního rozkladu Jacobiho metodou pro matice vyšších řádů a při požadování vysoké přesnosti trvá velmi dlouho a zároveň počáteční konstrukce matice $\mathbf{A}^T\mathbf{A}$ vede k určité ztrátě informace.

Dále bylo ukázáno, že maticové rozklady mohou mít široké možnosti využití. Kromě přímého použití v matematice nacházejí své uplatnění například ve statistice či počítačové grafice – jak bylo předvedeno na příkladu SVD rozkladu.

Vytvořené zdrojové kódy implementací představených algoritmů a skripty použité při řešení demonstračních úloh jsou k prohlédnutí v příloze práce: část A.1 obsahuje funkce související s výpočtem a aplikacemi LU rozkladu, část A.2 uvádí algoritmy týkající se Moore-Penroseovy inverze a část A.3 prezentuje zdrojové kódy spojené s SVD rozkladem.

7 Závěry a doporučení

I když práce ukázala pouze vybrané rozklady a jen některé jejich aplikace, je zřejmé, že maticové rozklady mají své nezastupitelné místo nejen v matematice, ale i v oblastech s ní souvisejících. Díky dostatečnému výpočetnímu výkonu, který je dnes k dispozici, není problém zkonstruovat rozklady i pro matice vyšších řádů, což otevírá cestu k dalším možnostem jejich využití. Na druhou stranu však použití maticových rozkladů nemusí být vždy tím nejvhodnějším přístupem pro řešení nějaké konkrétní úlohy, nutné je zvážit i alternativní možnosti řešení.

Práce poskytla pohled na teoretický výklad a praktické aplikace některých maticových rozkladů, mimo to nastínila i možnosti jejich výpočtu. Problematika výpočtu rozkladů však není zdaleka vyčerpána a zasloužila by si další pozornost. Jak již bylo zmíněno, pro výpočet uvedených rozkladů existuje množství algoritmů, které je možné analyzovat a porovnávat z numerického hlediska. Zajímavá je též otázka jejich efektivní implementace v určitém programovacím jazyce či třeba využití možností paralelních výpočtů, které jsou podporovány i prostředím MATLAB. Rovněž přehled možných aplikací uvedených rozkladů není kompletní. Například SVD rozklad nachází využití také při výpočtu polárního rozkladu, latentním sémantickém indexování, registraci dat a mnohé další, a je tak právem považován za jeden z nejdůležitějších nástrojů maticových výpočtů.

Literatura

[1] BEČVÁŘ, Jindřich. *Lineární algebra.* 4. vyd. Praha: Matfyzpress, 2010, 435 s. ISBN 978-80-7378-135-4.

[2] BELHUMEUR, Peter N., HESPANHA, João P., KRIEGMAN, David J. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997, vol. 19, no. 7, s. 711–720. ISSN 0162-8828.

[3] BEN-ISRAEL, Adi, GREVILLE, Thomas N. E. *Generalized inverses: theory and applications.* 2nd ed. New York: Springer, 2003, 420 s. CMS Books in Mathematics, 15. ISBN 0-387-00293-6.

[4] DONGARRA, Jack, SULLIVAN, Francis. Guest Editors Introduction to the top 10 algorithms. *Computing in Science & Engineering*, 2000, vol. 2, no. 1, s. 22–23. ISSN 1521-9615.

[5] DONT, Miroslav. *Elementy numerické lineární algebry.* 1. vyd. Praha: Vydavatelství ČVUT, 2004, 221 s. ISBN 80-01-02969-7.

[6] DU CROZ, Jeremy J., HIGHAM, Nicholas J. Stability of methods for matrix inversion. *IMA Journal of Numerical Analysis*, 1992, vol. 12, no. 1, s. 1–19. ISSN 0272-4979.

[7] DUINTJER TEBBENS, Jurjen et al. *Analýza metod pro maticové výpočty: základní metody.* 1. vyd. Praha: Matfyzpress, 2012, 308 s. ISBN 978-80-7378-201-6.

[8] GAVALCOVÁ, Tatiana, PRAŽÁK, Pavel. *Matematika 2.* 1. vyd. Hradec Králové: Gaudeamus, 2013, 271 s. ISBN 978-80-7435-247-8.

[9] GOLUB, Gene, KAHAN, William. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 1965, vol. 2, no. 2, s. 205–224. ISSN 0887-459X.

[10] GOLUB, Gene H., VAN LOAN, Charles F. *Matrix computations.* 3rd ed. Baltimore, Md.: Johns Hopkins University Press, 1996, 694 s. Johns Hopkins Studies in the Matematical Sciences. ISBN 0-8018-5414-8.

[11] GREVILLE, Thomas N. E. Some applications of the pseudoinverse of a matrix. *SIAM Review*, 1960, vol. 2, no. 1, s. 15–22. ISSN 0036-1445.

[12] HANSEN, Per Christian, NAGY, James G. O'LEARY, Dianne P. *Deblurring images: matrices, spectra, and filtering.* 1st ed. Philadelphia: Society for Industrial and Applied Mathematics, 2006, 130 s. ISBN 978-0-898716-18-4.

[13] HEBÁK, Petr, et al. *Vícerozměrné statistické metody (1).* 1. vyd. Praha: Informatorium, 2004, 239 s. ISBN 80-7333-025-3.

[14] HEBÁK, Petr, et al. *Vícerozměrné statistické metody (3).* 1. vyd. Praha: Informatorium, 2005, 255 s. ISBN 80-7333-039-3.

[15] HIGHAM, Nicholas J. *Accuracy and stability of numerical algorithms.* 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2002, 680 s. ISBN 0-89871-521-0.

[16] HUDEČEK, Jiří. *Matematika v devíti kapitolách: sbírka početních metod z doby Han s komentářem Liu Huie z doby Wei a Li Chunfega a dalších z doby Tang.* 1. vyd. Praha: Matfyzpress, 2008, 244 s. Dějiny matematiky, sv. 37. ISBN 978-80-7378-046-3.

[17] LIU, Jun, CHEN, Songcan, TAN, Xiaoyang. Fractional order singular value decomposition representation for face recognition. *Pattern Recognition*, 2008, vol. 41, no. 1, s. 378–395. ISSN 0031-3203.

[18] MATHEWS, John H., FINK Kurtis D. *Numerical methods using MATLAB.* 4th ed. Upper Saddle River, N.J.: Pearson, 2004, 680 s. ISBN 0-13-065248-2.

[19] The MathWorks, Inc. *MATLAB R2015b Documentation: Functions* [online]. 2015 [cit. 2016-02-08]. Dostupné z `http://www.mathworks.com/help/matlab/functionlist.html`

[20] The MathWorks, Inc. *MATLAB R2015b Documentation: Rank of matrix - MATLAB rank* [online]. 2015 [cit. 2015-09-29]. Dostupné z `http://www.mathworks.com/help/matlab/ref/rank.html`

[21] ROHN, Jiří. *Lineární algebra a optimalizace.* 1. vyd. Praha: Karolinum, 2004, 199 s. ISBN 80-246-0932-0.

[22] ROTELLA, Frédéric. *Observation* [online]. Tarbes: École Nationale d'Ingénieurs de Tarbes, 2003, Annexe C, s. 40–45 [cit. 2015-06-16]. Dostupné z `http://www.clubeea.org/documents/mediatheque/observateurs.pdf`

[23] SAMARIA, Ferdinando, HARTER, Andy. Parameterisation of a stochastic model for human face identification. In: *Proceedings of the Second IEEE Workshop on Applications of Computer Vision: December 5-7, 1994, Sarasota, Florida.* Los Alamitos, California: IEEE Computer Society Press, 1994, s. 138–142. ISBN 0-8186-6410-X.

[24] SKULA, Ladislav. Moore-Penroseova inverze matice a její aplikace. *Kvaternion* [online]. 2013, č. 1, s. 7–14 [cit. 2015-06-12]. ISSN 1805-1332. Dostupné z `http://kvaternion.fme.vutbr.cz/2013/kvat3_separaty/skula_final.pdf`

[25] STEWART, Gilbert W. *Matrix algorithms, Volume II: Eigensystems.* 1st ed. Philadelphia: Society for Industrial and Applied Mathematics, 2001, 469 s. ISBN 089871-503-2.

[26] TAN, Xiaoyang et al. Face recognition from a single image per person: A survey. *Pattern Recognition*, 2006, vol. 39, no. 9, s. 1725–1745. ISSN 0031-3203.

[27] TREFETHEN, Lloyd N, BAU, David. *Numerical linear algebra.* 1st ed. Philadelphia: Society for Industrial and Applied Mathematics, 1997, 361 s. ISBN 978-0-898713-61-9.

[28] TURK, Matthew, PENTLAND Alex. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 1991, vol. 3, no. 1, s. 71–86. ISSN 0898-929X.

[29] YANG, Won Yong et al. *Applied numerical methods using MATLAB.* 1st ed. Hoboken, N.J.: Wiley-Interscience, 2005, 509 s. ISBN 0-471-69833-4.

Použité značení

$\mathbb{R}$	Množina reálných čísel
$\mathbb{C}$	Množina komplexních čísel
$\boldsymbol{x}$	Sloupcový vektor
$\boldsymbol{x}^T$	Transponovaný vektor k vektoru $\boldsymbol{x}$ (řádkový vektor)
x_i	Prvek vektoru $\boldsymbol{x}$ na pozici i
$\mathbf{A}$	Matice
a_{ij}	Prvek matice $\mathbf{A}$ na pozici ij
$\mathbf{A}_{i\bullet}$	i-tý řádek matice $\mathbf{A}$
$\mathbf{A}_{\bullet j},\ \boldsymbol{a}_j$	j-tý sloupec matice $\mathbf{A}$
$\mathbf{A}_{1:i,1:j}$	Matice tvořená prvními i řádky a prvními j sloupci matice $\mathbf{A}$
$\mathbf{A}^T$	Transponovaná matice k matici $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverzní matice k regulární matici $\mathbf{A}$
$\mathbf{A}^\dagger$	Moore-Penroseova inverze matice $\mathbf{A}$
$\mathbf{I},\ \mathbf{I}_{(n)}$	Jednotková matice, jednotková matice řádu n
$\boldsymbol{e}_j$	j-tý sloupec jednotkové matice
$\det(\mathbf{A})$	Determinant matice $\mathbf{A}$
$\operatorname{rank}(\mathbf{A})$	Hodnost matice $\mathbf{A}$
$\mathcal{R}(\mathbf{A})$	Sloupcový prostor matice $\mathbf{A}$
$\mathcal{N}(\mathbf{A})$	Nulový prostor (jádro) matice $\mathbf{A}$
$\lambda,\ \lambda_i$	Vlastní čísla matice
$\sigma,\ \sigma_i$	Singulární čísla matice
$\|\boldsymbol{x}\|_2$	Euklidovská norma vektoru $\boldsymbol{x}$
$\|\mathbf{A}\|_2$	Spektrální norma matice $\mathbf{A}$
$\|\mathbf{A}\|_F$	Frobeniova norma matice $\mathbf{A}$
$\kappa(\mathbf{A})$	Číslo podmíněnosti matice $\mathbf{A}$

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Přílohy

A Zdrojové kódy funkcí a skriptů v MATLABu

Příloha obsahuje zdrojové kódy vytvořené v jazyce MATLAB, které ilustrují možné způsoby implementace v práci popsaných algoritmů a metod souvisejících s výpočtem a aplikacemi zmíněných maticových rozkladů. Jedná se o vlastní implementace, všechny uvedené funkce i skripty byly otestovány, při tvorbě bylo čerpáno zejména z [10] a [15]. Kódy byly vytvořeny v prostředí MATLAB R2014a a nevyžadují přítomnost žádného toolboxu.

Ve zdrojových kódech jsou využity následující předdefinované funkce MATLABu, podrobněji [19].

`abs(x)`	Vrací absolutní hodnotu prvků vektoru
`class(object)`	Určuje třídu objektu
`diag(A), diag(x)`	Vrací vektor diagonálních prvků matice $\mathbf{A}$ nebo diagonální matici s prvky vektoru $\boldsymbol{x}$ na diagonále
`double(A)`	Provádí konverzi na datový typ double
`eye(n)`	Vytváří jednotkovou matici řádu n
`imread(filename)`	Načítá obraz ze souboru
`intmax('classname')`	Vrací nejvyšší použitelnou hodnotu v zadané celočíselné třídě
`issymmetric(A)`	Ověří, zda je matice symetrická

`length(x)`	Vrací délku vektoru
`linspace(a, b)`	Vytváří vektor obsahující 100 čísel rovnoměrně rozložených mezi a a b
`max(x)`	Vrací největší prvek vektoru a případně i jeho pozici
`mean(A, 2)`	Vypočítá průměr prvků v jednotlivých řádcích matice
`min(x)`	Vrací nejmenší prvek vektoru a případně i jeho pozici
`norm(A), norm(A, 'fro')`	Vrací spektrální resp. Frobeniovu normu matice
`norm(x), norm(x, inf)`	Vrací euklidovskou resp. maximovou normu vektoru
`pinv(A)`	Vypočítá Moore-Penroseovu inverzi matice
`polyval(p, x)`	Vypočítá funkční hodnotu polynomu pro daná x
`prod(x)`	Vypočítá součin prvků vektoru
`reshape(A, [m n])`	Transformuje matici na matici o rozměrech $m \times n$
`size(A)`	Vrací rozměry matice
`sort(x, 'descend')`	Seřadí prvky vektoru v sestupném pořadí
`sqrt(x)`	Vypočítá druhou odmocninu
`strcat(s1, ..., sN)`	Spojuje textové řetězce
`strcmp(s1, s2)`	Porovnává shodnost textových řetězců
`sum(x)`	Vypočítá součet prvků vektoru
`svd(A)`	Vypočítá SVD rozklad matice
`tril(A)`	Vrací dolní trojúhelníkovou část matice
`triu(A)`	Vrací horní trojúhelníkovou část matice
`zeros(m, n)`	Vytváří nulovou matici o rozměrech $m \times n$

Budiž podotknuto, že i pro některé z implementovaných algoritmů lze v MATLABu najít ekvivalentní předdefinovanou funkci.

A.1 LU rozklad matice

A.1.1 Výpočet LU rozkladu

```
function [ varargout ] = luRozklad( A )
% LU rozklad matice s částečnou pivotací
%
%   [ Y, p ] = luRozklad( A )
%   Vstup:  A ... regulární matice n x n
%   Výstup: Y ... matice obsahující horní trojúhelníkovou matici U
%                 a dolní trojúhelníkovou matici L bez jednotkové
```

```
%                   diagonály, Y = U + L - eye(n)
%            p ... permutační vektor, A(p, :) = L * U
%    [ L, U, P ] = luRozklad( A )
%    Výstup: L ... dolní trojúhelníková matice
%            U ... horní trojúhelníková matice
%            P ... permutační matice, P * A = L * U
%
%    Datum poslední úpravy: 23.7.2015, KF

[n, n2] = size(A);
if n ~= n2
    error ('Zadaná matice není čtvercová.')
end
% Inicializace permutačního vektoru
p = 1 : n;
for k = 1 : n - 1
    % Nalezení maxima ve sloupci k (sloupcový výběr pivota)
    [maximum, imax] = max(abs(A(k : n, k)));
    if maximum < eps
        error ('LU:singularniMatice', 'Zadaná matice je singulární.')
    end
    r = imax + k - 1;
    % Výměna řádku k s řádkem r v matici A a zaznamenání informace
    % o výměně do permutačního vektoru
    if r ~= k
        A([r k], :) = A([k r], :);
        p([r k]) = p([k r]);
    end
    % Eliminace poddiagonálních prvků sloupce k
    for i = k + 1 : n
        A(i, k) = A(i, k) / A(k, k); % Výpočet multiplikátoru
        A(i, k + 1 : n) = A(i, k + 1 : n) - A(i, k) * A(k, k + 1 : n);
    end
end
if abs(A(n, n)) < eps
    error ('LU:singularniMatice', 'Zadaná matice je singulární.')
end
if nargout == 2
    varargout{1} = A;
    varargout{2} = p;
end
if nargout == 3
    I = eye(n);
    varargout{1} = I + tril(A, -1); % Matice L
    varargout{2} = triu(A);         % Matice U
    varargout{3} = I(p, :);         % Matice P
end
end
```

A.1.2 Aplikace LU rozkladu

Řešení soustav lineárních algebraických rovnic

```
function [ b ] = slar( A, b )
% Řešení soustavy lineárních algebraických rovnic s regulární maticí
% pomocí LU rozkladu s částečnou pivotací
```

```
%
%   [ x ] = slar( A, b )
%   Vstup:  A ... regulární matice soustavy n x n
%           b ... vektor pravých stran n x 1
%   Výstup: x ... vektor řešení
%
%   Datum poslední úpravy: 23.7.2015, KF

[n, ~] = size(A);
if any(size(b) ~= [n, 1])
    error ('Vektor pravých stran musí být rozměru n x 1.')
end
% LU rozklad matice A
[A, p] = luRozklad(A);
% Permutace prvků vektoru b
b = b(p);
% Řešení soustavy Ly = Pb pomocí přímé substituce
% (vektor y přepisuje vektor b)
for k = 2 : n
    b(k) = b(k) - A(k, 1 : k - 1) * b(1 : k - 1);
end
% Řešení soustavy Ux = y pomocí zpětné substituce
% (vektor x přepisuje vektor b)
b(n) = b(n) / A(n, n);
for k = n - 1 : -1 : 1
    b(k) = (b(k) - A(k, k + 1 : n) * b(k + 1 : n)) / A(k, k);
end
end
```

Výpočet inverzní matice

```
function [ X ] = inverzeA( A )
% Výpočet inverzní matice pomocí LU rozkladu s částečnou pivotací
% Inverze X je počítána po sloupcích postupným řešením soustav
% A * X(:, j) = I(:, j)
%
%   [ X ] = inverzeA( A )
%   Vstup:  A ... regulární matice n x n
%   Výstup: X ... inverzní matice A^-1
%
%   Datum poslední úpravy: 16.8.2015, KF

[A, p] = luRozklad(A);
[n, ~] = size(A);
X = eye(n);
X = X(p, :);
for j = 1 : n
    % Výpočet j-tého sloupce
    for k = 2 : n
        X(k, j) = X(k, j) - A(k, 1 : k - 1) * X(1 : k - 1, j);
    end
    X(n, j) = X(n, j) / A(n, n);
    for k = n - 1 : -1 : 1
        X(k, j) = (X(k, j) - A(k, k + 1 : n) * X(k + 1 : n, j)) / A(k,
            k);
    end
end
end
```

```
function [ A ] = inverzeB( A )
% Výpočet inverzní matice pomocí LU rozkladu s částečnou pivotací
% Inverze X je počítána jako řešení rovnice X * L = U^-1
%
%   [ X ] = inverzeB( A )
%   Vstup:  A ... regulární matice n x n
%   Výstup: X ... inverzní matice A^-1
%
%   Datum poslední úpravy: 16.8.2015, KF

[A, p] = luRozklad(A);
[n, ~] = size(A);
A = invU(triu(A)) / (eye(n) + tril(A, -1));
r(p) = 1 : n;
A = A(:, r);
end
```

```
function [ X ] = inverzeC( A )
% Výpočet inverzní matice pomocí LU rozkladu s částečnou pivotací
% Pro výpočet je využit algoritmus dle Du Croz, Higham: Stability of
% methods for matrix inversion, Method C
%
%   [ X ] = inverzeC( A )
%   Vstup:  A ... regulární matice n x n
%   Výstup: X ... inverzní matice A^-1
%
%   Datum poslední úpravy: 16.8.2015, KF

[A, p] = luRozklad(A);
[n, ~] = size(A);
X = eye(n);
for k = n : -1 : 1
    X(k + 1 : n, k) = -X(k + 1 : n, k + 1 : n) * A(k + 1 : n, k);
    X(k, k + 1 : n) = -A(k, k + 1 : n) * X(k + 1 : n, k + 1 : n) / A(k
        , k);
    X(k, k) = 1 / A(k, k) - X(k, k + 1 : n) * A(k + 1 : n, k);
end
r(p) = 1 : n;
X = X(:, r);
end
```

```
function [ A ] = inverzeD( A )
% Výpočet inverzní matice pomocí LU rozkladu s částečnou pivotací
% Inverze X je počítána podle vztahu X = U^-1 * L^-1
%
%   [ X ] = inverzeD( A )
%   Vstup:  A ... regulární matice n x n
%   Výstup: X ... inverzní matice A^-1
%
%   Datum poslední úpravy: 16.8.2015, KF

[A, p] = luRozklad(A);
[n, ~] = size(A);
A = invU(triu(A)) * invL(eye(n) + tril(A, -1));
r(p) = 1 : n;
```

```
A = A(:, r);
end
```

```
function [ L ] = invL( L )
% Výpočet inverze dolní trojúhelníkové matice
%
%   [ X ] = invL( L )
%   Vstup:  L ... dolní trojúhelníková matice n x n
%   Výstup: X ... inverzní matice L^-1
%
%   Datum poslední úpravy: 26.8.2015, KF

[n, ~] = size(L);
for j = n : -1 : 1
    L(j, j) = 1 / L(j, j);
    L(j + 1 : n, j) = L(j + 1 : n, j + 1 : n) * L(j + 1 : n, j);
    L(j + 1 : n, j) = -L(j, j) * L(j + 1 : n, j);
end
end
```

```
function [ U ] = invU( U )
% Výpočet inverze horní trojúhelníkové matice
%
%   [ X ] = invU( U )
%   Vstup:  U ... horní trojúhelníková matice n x n
%   Výstup: X ... inverzní matice U^-1
%
%   Datum poslední úpravy: 26.8.2015, KF

[n, ~] = size(U);
for j = 1 : n
    U(j, j) = 1 / U(j, j);
    U(1 : j - 1, j) = U(1 : j - 1, 1 : j - 1) * U(1 : j - 1, j);
    U(1 : j - 1, j) = -U(j, j) * U(1 : j - 1, j);
end
end
```

Výpočet determinantu matice

```
function [ det ] = determinant( A )
% Výpočet determinantu matice pomocí LU rozkladu s částečnou pivotací
%
%   [ det ] = determinant( A )
%   Vstup:  A ... regulární matice n x n
%   Výstup: det ... determinant matice A
%
%   Datum poslední úpravy: 23.7.2015, KF

try
[A, p] = luRozklad(A);
catch ME
    if (strcmp(ME.identifier, 'LU:singularniMatice'))
        det = 0; % Matice je singulární
        return
    else
        rethrow(ME)
```

```
    end
end
% Znaménko permutace
[n, ~] = size(A);
sign = 1;
for i = 1 : n - 1
    for j = i + 1 : n
        if p(i) > p(j)
            sign = -sign;
        end
    end
end
det = sign * prod(diag(A));
end
```

A.2 Moore-Penroseova inverze matice

A.2.1 Výpočet Moore-Penroseovy inverze

Následující funkce vyžadují jako vstupní argument hodnotu tolerance pro rozpoznání nuly. V případě výpočtu redukovaného odstupňovaného tvaru je možné toleranci určit automaticky využitím vztahu `tol = max(size(A)) * eps * norm(A, inf)` (tak, jak je tomu i u předdefinované funkce MATLABu `rref()`), pro Grevillův algoritmus však podobná možnost nebyla nalezena, a tak nezbývá, než vhodnou hodnotu tolerance určit pro konkrétní vstupní matici experimentálně.

Metoda hodnostního rozkladu

```
function [ X ] = pinvHR( A, tol )
% Moore-Penroseova inverze za využití hodnostního rozkladu matice
%
%   [ X ] = pinvHR( A, tol )
%   Vstup:  A ... matice m x n
%           tol ... tolerance pro rozpoznání nuly
%   Výstup: X ... Moore-Penroseova inverze matice A
%
%   Datum poslední úpravy: 8.8.2015, KF

try
[B, C] = hodRozklad(A, tol);
catch
    [m, n] = size(A);
    X = zeros(n, m);
    return
end
X = C' / (B' * A * C') * B';
end
```

```
function [ B, C ] = hodRozklad( A, tol )
% Hodnostní rozklad matice
%
%   [ B, C ] = hodRozklad( A, tol )
%   Vstup:  A ... matice m x n, A ~= 0
%           tol ... tolerance pro rozpoznání nuly
%   Výstup: B ... matice m x r
%           C ... matice r x n
%
%   Datum poslední úpravy: 8.8.2015, KF

[~, n] = size(A);
% Výpočet redukovaného odstupňovaného tvaru
[R, baz] = rrefTvar(A, tol);
% Hodnost matice A
r = length(baz);
if r == 0
    error('Zadaná matice je nulová.')
end
% Matice B je tvořena bázovými sloupci matice A
B = A(:, baz);
% Matice C odpovídá prvním r řádkům matice R
C = R(1 : r, 1 : n);
end
```

```
function [ A, baz ] = rrefTvar( A, tol )
% Výpočet redukovaného odstupňovaného tvaru matice (RREF)
% za využití Gauss-Jordanovy eliminace s částečnou pivotací
%
%   [ R, baz ] = rrefTvar( A, tol )
%   Vstup:  A ... matice m x n
%           tol ... tolerance pro rozpoznání nuly
%   Výstup: R ... matice v redukovaném odstupňovaném tvaru
%           baz ... vektor obsahující indexy bázových sloupců matice A
%
%   Datum poslední úpravy: 8.8.2015, KF

[m, n] = size(A);
i = 1;
j = 1;
baz = zeros(1, n);
while i <= m && j <= n
    % Nalezení maxima ve sloupci j (sloupcový výběr pivota)
    [maximum, imax] = max(abs(A(i : m, j)));
    if maximum <= tol
        % Prvky sloupce j jsou zanedbatelně malé a jsou vynulovány
        A(i : m, j) = zeros(m - i + 1, 1);
        % Hledání pivota pokračuje v sousedním sloupci
        j = j + 1;
    else
        k = imax + i - 1;
        % Výměna i-tého a k-tého řádku matice
        if i ~= k
            A([i k], j : n) = A([k i], j : n);
        end
        % Vydělení pivotního řádku hodnotou pivota
        A(i, j : n) = A(i, j : n) / A(i, j);
```

```
        % Eliminace prvků j-tého sloupce nacházejících se
        % nad a pod pivotem
        for k = [1 : i - 1 i + 1 : m]
            A(k, j + 1 : n) = A(k, j + 1 : n) - A(k, j) * A(i, j + 1 :
                n);
            A(k, j) = 0;
        end
        % Sloupec j je bázovým sloupcem
        baz(i) = j;
    i = i + 1;
    j = j + 1;
    end
end
baz = baz(1 : i - 1);
end
```

Grevillův algoritmus

```
function [ X ] = pinvGreville( A, tol )
% Moore-Penroseova inverze za využití Grevillova algoritmu
%
%   [ X ] = pinvGreville( A, tol )
%   Vstup:  A ... matice m x n
%           tol ... tolerance pro rozpoznání nulového vektoru
%   Výstup: X ... Moore-Penroseova inverze matice A
%
%   Datum poslední úpravy: 8.8.2015, KF

[m, n] = size(A);
% Pro první sloupec matice A
c = A(:, 1);
if norm(c, inf) <= tol
    % Prvky vektoru jsou zanedbatelně malé
    % a jsou tedy považovány za nulové
    X = zeros(1, m);
else
    X = c' / (c' * c);
end
% Pro druhý až n-tý sloupec
for j = 2 : n
    d = X * A(:, j);
    c = A(:, j) - A(:, 1 : j - 1) * d;
    if norm(c, inf) <= tol
        % Prvky vektoru jsou zanedbatelně malé
        % a jsou tedy považovány za nulové
        bt = (d' * X) / (1 + d' * d);
    else
        bt = c' / (c' * c);
    end
    % Grevillův vzorec
    X = [X - d * bt; bt];
end
end
```

A.2.2 Aplikace Moore-Penroseovy inverze

Problém nejmenších čtverců

```
% Aproximace polynomem

% Zadaná data
x = [-3; -2; -1; 0; 1; 2; 3];
y = [-0.2774; 0.8958; -1.5651; 3.4565; 3.0601; 4.8568; 3.8982];

% Výpočet koeficientů aproximačních polynomů stupně p = 1, 3, 5, 6
a1 = polynomFit(x, y, 1);
a3 = polynomFit(x, y, 3);
a5 = polynomFit(x, y, 5);
a6 = polynomFit(x, y, 6);

% Dopočítání bodů pro nakreslení grafů
xx = linspace(-4, 4);
yy1 = polyval(a1(end : -1 : 1), xx);
yy3 = polyval(a3(end : -1 : 1), xx);
yy5 = polyval(a5(end : -1 : 1), xx);
yy6 = polyval(a6(end : -1 : 1), xx);

% Zobrazení grafů nalezených polynomů
figure('Color', 'w')
subplot(2, 2, 1)
plot(x, y, 'k*')
hold on
plot(xx, yy1, 'k')
hold off
axis([-4 4 -3 9])

subplot(2, 2, 2)
plot(x, y, 'k*')
hold on
plot(xx, yy3, 'k')
hold off
axis([-4 4 -3 9])

subplot(2, 2, 3)
plot(x, y, 'k*')
hold on
plot(xx, yy5, 'k')
hold off
axis([-4 4 -3 9])

subplot(2, 2, 4)
plot(x, y, 'k*')
hold on
plot(xx, yy6, 'k')
hold off
axis([-4 4 -3 9])
```

```
function [ a ] = polynomFit( x, y, p )
% Aproximace polynomem
%
```

```
%   [ a ] = polynomFit( x, y, p )
%   Vstup:  x ... vektor souřadnic x vstupních dat
%           y ... vektor souřadnic y vstupních dat
%           p ... stupeň polynomu
%   Výstup: a ... koeficinty hledaného polynomu,
%                 a(1) + a(2) * x + a(3) * x^2 + ... + a(p + 1) * x^p
%
%   Datum poslední úpravy: 1.11.2015, KF

m = length(x);
n = length(y);
if m ~= n
    error('Vektory x a y musejí mít stejnou délku.')
end
if p >= m
    warning(['Zadaný stupeň polynomu je stejný nebo větší než ' ...
    'počet datových bodů, řešení nebude jednoznačné.'])
end
% Konstrukce matice A (Vandermondova matice)
A = zeros(m, p + 1);
for i = 1 : m
    A(i, 1 : p + 1) = x(i) .^ (0 : p);
end
% Řešení ve smyslu nejmenších čtverců
a = pinvSvd(A, max(size(A)) * norm(A) * eps) * y(:);
end
```

A.3 SVD rozklad matice

A.3.1 Výpočet SVD rozkladu

```
function [ U, S, V ] = svdRozklad( A, tol )
% SVD rozklad matice vypočtený pomocí spektrálního rozkladu
% a QR rozkladu
%
%   [ U, S, V ] = svdRozklad( A, tol )
%   Vstup:  A ... matice m x n
%           tol ... tolerance udávající požadovanou přesnost výsledků
%   Výstup: U ... ortogonální matice m x m, jejíž sloupce jsou tvořeny
%                 levými singulárními vektory matice A
%           S ... diagonální matice m x n, jejíž diagonální prvky jsou
%                 singulární čísla matice A
%           V ... ortogonální matice n x n, jejíž sloupce jsou tvořeny
%                 pravými singulárními vektory matice A,
%                 A = U * S * V'
%
%   Datum poslední úpravy: 1.11.2015, KF

[m, n] = size(A);
if m >= n
    [V, ~] = spektRozklad(A' * A, tol);
    [U, S] = qrRozklad(A * V);
    S = tril(S);
else
    [U, ~] = spektRozklad(A * A', tol);
```

```
    [V, S] = qrRozklad(A' * U);
    S = tril(S)';
end
end
```

```
function [ X, A ] = spektRozklad( A, tol )
% Spektrální rozklad matice pomocí Jacobiho metody
%
%   [ X, L ] = spektRozklad( A, tol )
%   Vstup:  A ... symetrická matice n x n
%           tol ... tolerance udávající požadovanou přesnost výsledků
%   Výstup: L ... diagonální matice n x n, jejíž diagonální prvky l_ii
%                  aproximují vlastní čísla lambda_i matice A,
%                  abs(lambda_i - l_ii) <= tol * norm(A, 'fro'),
%                  i = 1,...,n
%           X ... ortogonální matice n x n, jejíž sloupce jsou tvořeny
%                  příslušnými vlastními vektory matice A,
%                  A = X * L * X'
%
%   Datum poslední úpravy: 1.11.2015, KF

[n, n2] = size(A);
if n ~= n2
    error ('Zadaná matice není čtvercová.')
end
if ~issymmetric(A)
    error ('Zadaná matice není symetrická.')
end
X = eye(n);
delta = tol * norm(A, 'fro');
while off(A) > delta
    % Nalezení maxima z nediagonálních prvků
    p = 1;
    q = 2;
    for i = 1 : n
        for j = i + 1 : n
            if abs(A(i, j)) > abs(A(p, q))
                p = i;
                q = j;
            end
        end
    end
    % Výpočet hodnot sinus (s) a kosinus (c)
    if A(p, q) ~= 0
        tau = (A(q, q) - A(p, p)) / (2 * A(p, q));
        if tau >= 0
            t = 1 / (tau + sqrt(1 + tau ^ 2));
        else
            t = -1 / (-tau + sqrt(1 + tau ^ 2));
        end
        c = 1 / sqrt(1 + t ^ 2);
        s = t * c;
    else
        c = 1;
        s = 0;
    end
    % Aktualizace matic A (A = G' * A * G) a  X (X = X * G)
    A([p q], :) = [c s; -s c]' * A([p q], :);
```

```
    A(:, [p q]) = A(:, [p q]) * [c s; -s c];
    X(:, [p q]) = X(:, [p q]) * [c s; -s c];
end
% Seřazení diagonálních prvků výsledné matice A
[b, p] = sort(diag(A), 'descend');
A = diag(b);
X = X(:, p);
end
```

```
function [ s ] = off( A )
% Výpočet veličiny off(A)
% (pomocná funkce pro Jacobiho metodu)
%
%   [ off ] = off( A )
%   Vstup:  A ... matice m x n
%   Výstup: off ... odmocnina ze součtu druhých mocnin nediagonálních
%                   prvků matice A
%
%   Datum poslední úpravy: 23.10.2015, KF

[m, n] = size(A);
s = 0;
for i = 1 : m
    for j = 1 : n
        if i ~= j
            s = s + A(i, j) ^ 2;
        end
    end
end
s = sqrt(s);
end
```

```
function [ varargout ] = qrRozklad( A )
% QR rozklad matice užitím Householderových reflexí
%
%   [ Y ] = qrRozklad( A )
%   Vstup:  A ... matice m x n
%   Výstup: Y ... matice, pro kterou R = triu(Y) a v jejíž
%                 poddiagonální části jsou uloženy příslušné
%                 Householderovy vektory (kromě první složky
%                 vektoru, která je vždy rovna jedné)
%   [ Q, R ] = qrRozklad( A )
%   Vstup:  A ... matice m x n
%   Výstup: Q ... ortogonální matice m x m
%           R ... horní trojúhelníková matice m x n,
%                 A = Q * R
%
%   Datum poslední úpravy: 1.11.2015, KF

[m, n] = size(A);
beta = zeros(1, min(n, m));
for j = 1 : min(n, m - 1)
    % Výpočet j-tého Householderova vektoru
    x = A(j : m, j);
    sigma = x(2 : end)' * x(2 : end);
    v = [1; x(2 : end)];
    if sigma == 0 && x(1) >= 0
```

```
        beta(j) = 0;
    elseif sigma == 0 && x(1) < 0
        beta(j) = 2;
    else
        mi = sqrt(sigma + x(1) ^ 2);
        if x(1) < 0
            v(1) = x(1) - mi;
        else
            v(1) = -sigma / (x(1) + mi);
        end
        beta(j) = 2 * v(1) ^ 2 / (sigma + v(1) ^ 2);
        v = v / v(1);
    end
    % Aktualizace matice A (A = Q_j * A)
    A(j : m, j : n) = A(j : m, j : n) - (beta(j) * v) * (v' * A(j : m,
        j : n));
    % Uložení Householderova vektoru do poddiagonální části matice A
    A(j + 1 : m, j) = v(2 : end);
end
if nargout == 1
    varargout{1} = A;
end
if nargout == 2
    % Konstrukce matice Q
    Q = eye(m);
    for j = min(n, m - 1) : -1 : 1
        v = [1; A(j + 1 : m, j)];
        Q(j : m, j : m) = Q(j : m, j : m) - (beta(j) * v) * (v' * Q(j
            : m, j : m));
    end
    varargout{1} = Q;
    varargout{2} = triu(A); % Matice R
end
end
```

A.3.2 Aplikace SVD rozkladu

V následujících funkcích a skriptech byla pro výpočet SVD rozkladu použita předdefinovaná funkce MATLABu `svd()`, neboť zde implementovaný algoritmus jeho výpočtu není pro matice vyšších řádů reálně použitelný (viz kapitola 6).

Fotografie využité v aplikačních úlohách jsou k dispozici na přiloženém CD.

Aproximace maticí nižší hodnosti a komprese dat

```
% SVD komprese obrazových dat

% Načtení obrazu
A = imread('HK.jpg');
% Převod na odstíny šedi
A = 0.2989 * A(:, :, 1) + 0.5870 * A(:, :, 2) + 0.1140 * A(:, :, 3);
% Převod na hodnoty typu double <0; 1>
A = double(A) ./ double(intmax(class(A)));
```

```
% Zobrazení originálního nekomprimovaného obrazu
figure
imagesc(A, [0 1])
axis image
axis off
colormap(gray)

[U, S, V] = svd(A, 'econ');

% Konstrukce aproximačních matic A_k
[m, n] = size(A);
k = [200, 80, 20];
for i = 1 : length(k)
    A = zeros(m, n);
    for j = 1 : k(i)
        A = A + S(j, j) * U(:, j) * V(:, j)';
    end
    figure
    imagesc(A, [0 1])
    axis image
    axis off
    colormap(gray)
end
```

Návrat k Moore-Penroseově inverzi a problému nejmenších čtverců

```
function [ A ] = pinvSvd( A, tol )
% Moore-Penroseova inverze za využití SVD rozkladu
%
%   [ X ] = pinvSvd( A, tol )
%   Vstup:  A ... matice m x n
%           tol ... tolerance pro rozpoznání nulových singulárních
%                   čísel
%   Výstup: X ... Moore-Penroseova inverze matice A
%
%   Datum poslední úpravy: 1.11.2015, KF

[U, S, V] = svd(A, 'econ'); % Redukovaný SVD rozklad
s = diag(S);
r = sum(s > tol); % Hodnost matice S resp. A vzhledem k zadané
   toleranci
s = 1 ./ s(1 : r);
A = V(:, 1 : r) * diag(s) * U(:, 1 : r)';
end
```

Ill-posed problémy a regularizace

```
% TSVD regularizace a rekonstrukce rozmazaného obrazu

% Při tvorbě skriptu bylo čerpáno z knihy Hansen, Nagy, O'Leary:
% Deblurring images, odkud byly převzaty funkce psfGauss a kronDecomp

% Načtení obrazu (ideální nerozmazaný obraz)
X = imread('HK.jpg');
% Převod na odstíny šedi
X = 0.2989 * X(:, :, 1) + 0.5870 * X(:, :, 2) + 0.1140 * X(:, :, 3);
% Převod na hodnoty typu double <0; 1>
```

```
X = double(X) ./ double(intmax(class(X)));
% Zobrazení původního nerozmazaného obrazu
figure
imagesc(X, [0 1])
axis image
axis off
colormap(gray)

% Vytvoření PSF pro rozmazání funkcí Gaussova normálního rozdělení
% (směrodatná odchylka = 7)
[PSF, stred] = psfGauss(size(X), 7);
% Výpočet matic AR, AC Kronekerova součinu při reflexivních hraničních
% podmínkách
[AR, AC] = kronDecomp(PSF, stred, 'reflexive');
% Konstrukce rozmazaného obrazu
B = AC * X * AR';
% Přidání aditivního šumu (Gaussův bílý šum)
% (norm(B_NOISE, 'fro') / norm(B_EXACT, 'fro') = 1e-4)
E = randn(size(B));
E = E / norm(E, 'fro');
B = B + 1e-4 * norm(B, 'fro') * E;
figure
imagesc(B, [0 1])
axis image
axis off
colormap(gray)

% Výpočet naivního řešení
Xn = pinv(AC) * B * pinv(AR)';
figure
imagesc(Xn, [0 1])
axis image
axis off
colormap(gray)

% TSVD regularizace a výsledné řešení
% (regularizační parametr k odpovídá počtu singulárních čísel matice A
% větších než tolerance tol)
[UR, SR, VR] = svd(AR, 'econ');
[UC, SC, VC] = svd(AC, 'econ');
S = diag(SC) * diag(SR)';
tol = 2.8e-4; % Tolerance odpovídající vhodnému řešení (určeno
    experimentálně)
Phi = (abs(S) >= tol); % k = sum(Phi(:));
idx = (S ~= 0);
s = zeros(size(Phi));
s(idx) = Phi(idx) ./ S(idx);
Xf = VC * ((UC' * B * UR) .* s) * VR';
figure
imagesc(Xf, [0 1])
axis image
axis off
colormap(gray)

tol = 5e-5; % Podregularizované řešení
Phi = (abs(S) >= tol);
idx = (S ~= 0);
s = zeros(size(Phi));
```

```
s(idx) = Phi(idx) ./ S(idx);
Xf = VC * ((UC' * B * UR) .* s) * VR';
figure
imagesc(Xf, [0 1])
axis image
axis off
colormap(gray)

tol = 1e-1; % Přeregularizované řešení
Phi = (abs(S) >= tol);
idx = (S ~= 0);
s = zeros(size(Phi));
s(idx) = Phi(idx) ./ S(idx);
Xf = VC * ((UC' * B * UR) .* s) * VR';
figure
imagesc(Xf, [0 1])
axis image
axis off
colormap(gray)
```

Analýza hlavních komponent

```
% Rozpoznávání tváří metodou "eigenfaces"
% dle Turk, Pentland: Eigenfaces for recognition

% Fotografie použité pro experimenty pocházejí z databáze ORL,
% každý obraz má rozměry 112x92 a je uložen ve formátu pgm,
% obrazy tréninkové množiny jsou umístěny ve složce faces\training\,
% obrazy testovací množiny jsou umístěny ve složce faces\test\,
% obrazy jsou pojmenovány x.pgm, kde x představuje číslo jedince

n = 10304; % 112 * 92
m = 39; % Počet obrazů v tréninkové množině
% Načtení obrazů tréninkové množiny
X = zeros(n, m);
for i = 1 : m
    Xi = imread(strcat('faces\training\', int2str(i), '.pgm'));
    Xi = double(Xi) ./ double(intmax(class(Xi)));
    X(:, i) = reshape(Xi, [n 1]);
end
% Výpočet průměrného obrazu tréninkové množiny
prumer = mean(X, 2);
% Odečtení průměru od všech obrazů
for i = 1 : m
    X(:, i) = X(:, i) - prumer;
end
% SVD rozklad matice obrazů
[U, S, ~] = svd(X, 'econ');
% Určení počtu p vlastních čísel kovarianční matice, která odpovídají
% 90 % variability původních dat
vlCis = diag(S) .^ 2;
suma = sum(vlCis);
ksuma = 0;
i = 1;
while (ksuma / suma) < 0.9
    ksuma = ksuma + vlCis(i);
    i = i + 1;
end
```

```
p = i;
% Projekce obrazů tréninkové množiny do prostoru tváří, jehož bázi
% tvoří prvních p vlastních vektorů kovarianční matice
% (výpočet vektorů Omega)
U = U(:, 1 : p);
X = U' * X;

t = 41; % počet obrazů v testovací množině
% Načtení obrazů testovací množiny
Y = zeros(n, t);
for i = 1 : t
    Yi = imread(strcat('faces\test\', int2str(i), '.pgm'));
    Yi = double(Yi) ./ double(intmax(class(Yi)));
    Y(:, i) = reshape(Yi, [n 1]);
end
vysledky = zeros(3, t);
% Pro každý obraz testovací množiny
for i = 1 : t
    Y(:, i) = Y(:, i) - prumer; % Odečtení průměru
    Yomega = U' * Y(:, i); % Výpočet vektoru Omega
    % Nalezení nejbližšího obrazu tréninkové množiny
    e = zeros(1, m);
    for j = 1 : m
        e(j) = norm(Yomega - X(:, j));
    end
    [emin, jedinec] = min(e);
    d = norm(Y(:, i) - U * Yomega); % Vzdálenost od prostoru tváří
    vysledky(:, i) = [jedinec; emin; d];
end
```

```
% Rozpoznávání tváří metodou "eigenfaces" s využitím FSVDR
% reprezentace tváří
% dle Liu, Chen, Tan: Fractional order singular value decomposition
% representation for face recognition

% Fotografie použité pro experimenty pocházejí z databáze Yale,
% každý obraz má rozměry 143x100 a je uložen ve formátu pgm,
% obrazy tréninkové množiny jsou umístěny ve složce faces2\training\,
% obrazy testovací množiny jsou umístěny ve složce faces2\test\,
% obrazy jsou pojmenovány x.pgm, kde x představuje číslo jedince

alpha = 0.1; % Parametr pro FSVDR
n = 14300; % 143 * 100
m = 15;
X = zeros(n, m);
for i = 1 : m
    Xi = imread(strcat('faces2\training\', int2str(i), '.pgm'));
    Xi = double(Xi) ./ double(intmax(class(Xi)));
    % Nahrazení načteného obrazu z tréninkové množiny jeho FSVDR
    % reprezentací
    [Ui, Si, Vi] = svd(Xi, 'econ');
    Xi = Ui * Si^alpha * Vi';
    X(:, i) = reshape(Xi, [n 1]);
end
prumer = mean(X, 2);
for i = 1 : m
    X(:, i) = X(:, i) - prumer;
end
```

```
[U, S, ~] = svd(X, 'econ');
vlCis = diag(S) .^ 2;
suma = sum(vlCis);
ksuma = 0;
i = 1;
while (ksuma / suma) < 0.9
    ksuma = ksuma + vlCis(i);
    i = i + 1;
end
p = i;
U = U(:, 1 : p);
X = U' * X;

t = 15;
Y = zeros(n, t);
for i = 1 : t
    Yi = imread(strcat('faces2\test\', int2str(i), '.pgm'));
    Yi = double(Yi) ./ double(intmax(class(Yi)));
    % Nahrazení načteného obrazu z testovací množiny jeho FSVDR
    % reprezentací
    [Ui, Si, Vi] = svd(Yi, 'econ');
    Yi = Ui * Si^alpha * Vi';
    Y(:, i) = reshape(Yi, [n 1]);
end
vysledky = zeros(3, t);
for i = 1 : t
    Y(:, i) = Y(:, i) - prumer;
    Yomega = U' * Y(:, i);
    e = zeros(1, m);
    for j = 1 : m
        e(j) = norm(Yomega - X(:, j));
    end
    [emin, jedinec] = min(e);
    d = norm(Y(:, i) - U * Yomega);
    vysledky(:, i) = [jedinec; emin; d];
end
```

B Obsah přiloženého CD

Přiložené CD obsahuje zdrojové kódy z přílohy A v podobě souborů prostředí MATLAB `*.m`, dále fotografie použité při ukázce aplikačních úloh a elektronickou verzi této práce.

- /
  - MATLAB src
    - LU rozklad
      - luRozklad.m
      - slar.m
      - inverzeA.m
      - inverzeB.m
      - inverzeC.m
      - inverzeD.m
      - invL.m
      - invU.m
      - determinant.m
    - Moore-Penroseova inverze
      - pinvHR.m
      - hodRozklad.m
      - rrefTvar.m
      - pinvGreville.m
      - aproxPolynomy.m
      - polynomFit.m
    - SVD rozklad
      - faces ..... *fotografie tváří použité při ukázce metody eigenfaces* (`*.pgm`)
        - training
        - test
      - faces2 ..... *fotografie tváří použité při ukázce FSVDR reprezentace* (`*.pgm`)
        - training
        - test
      - HK.jpg
      - svdRozklad.m
      - spektRozklad.m
      - off.m
      - qrRozklad.m
      - kompreseSvd.m
      - pinvSvd.m
      - tsvd.m
      - psfGauss.m
      - kronDecomp.m
      - eigenfaces.m
      - eigenfacesFsvdr.m
  - BP_Fronckova_2016.pdf

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Frončková Kateřina	Edvarda Beneše 1433/1A, Hradec Králové - Nový Hradec Králové	I1301423

TÉMA ČESKY:

Maticové rozklady

TÉMA ANGLICKY:

Matrix factorization

VEDOUCÍ PRÁCE:

doc. RNDr. Pavel Pražák, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce:
Teoretický popis vybraných maticových rozkladů a představení jejich aplikací včetně implementace v prostředí MATLAB

Osnova práce:
Základní pojmy lineární algebry
LU rozklad matice
- Teoretická východiska a výpočet
- Aplikace (soustavy lineárních algebraických rovnic, inverzní matice, determinant)

Moore-Penroseova inverze matice
- Teoretická východiska a výpočet
- Aplikace (problém nejmenších čtverců)

SVD rozklad matice
- Teoretická východiska a výpočet
- Aplikace (vybrané vlastnosti matic, aproximace maticí nižší hodnosti, souvislost s řešením soustav rovnic)

SEZNAM DOPORUČENÉ LITERATURY:

DONT, Miroslav. Elementy numerické lineární algebry. 1. vyd. Praha: Vydavatelství ČVUT, 2004. ISBN 80-01-02969-7.
DUINTJER TEBBENS, Erik Jurjen. Analýza metod pro maticové výpočty: základní metody. 1. vyd. Praha: Matfyzpress, 2012. ISBN 978-80-7378-201-6.
GOLUB, Gene H, VAN LOAN, Charles F. Matrix computations. 3rd ed. Baltimore, Md.: Johns Hopkins University Press, 1996. Johns Hopkins studies in the matematical sciences. ISBN 0801854148.
ROHN, Jiří. Lineární algebra a optimalizace. 1. vyd. Praha: Karolinum, 2004. ISBN 80-246-0932-0.

Podpis studenta: Frončková ..... **Datum:** 9.10.2015

Podpis vedoucího práce: P. Pražák ..... **Datum:** 9.10.2015