

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Klančar

**Algoritmi za reševanje problema  
matričnih napolnitev**

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM  
PRVE STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Aljaž Zalar

Ljubljana, 2023

To delo je ponujeno pod licenco *Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija* (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavnine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Inštitutu za intelektualno lastnino, Strelška 1, 1000 Ljubljana.



Izvorna koda diplomskega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco GNU General Public License, različica 3 (ali novejša). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani <http://www.gnu.org/licenses/>.

*Besedilo je oblikovano z urejevalnikom besedil L<sup>A</sup>T<sub>E</sub>X.*

**Kandidat:** Matej Klančar

**Naslov:** Naslov diplomskega dela

**Vrsta naloge:** Diplomska naloga na univerzitetnem programu prve stopnje  
Računalništvo in informatika

**Mentor:** doc. dr. Aljaž Zalar

**Opis:**

V diplomskem delu predstavite nekaj različnih algoritmov za reševanje problema matričnih napolnitev. Seznanite se z uporabljenimi matematičnimi orodji. Algoritme implementirajte in testirajte na izbranem problemu uporabe. Primerjajte algoritme glede na kakovost napolnitev, časovno zahtevnost in pojasnite ugotovitve z navezavo na teoretično podlago algoritmov. Na koncu predlagajte možne nadaljnje izboljšave.

**Title:** Algorithms for solving matrix completion problem

**Description:**

In the thesis, present a few different algorithms for solving the matrix completion problem. Study the underlying mathematical methods. Implement the algorithms and test them on a selected applied problem. Compare the algorithms in terms of quality of the completion, computational cost, and explain the results referring to the theoretical background of the algorithms. Finally, suggest possible improvements to the methods presented.



*Na tem mestu zapišite, komu se zahvaljujete za pomoč pri izdelavi diplomske naloge oziroma pri vašem študiju nasploh. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.*



# Kazalo

## Povzetek

## Abstract

<b>1</b>	<b>Uvod</b>	<b>1</b>
1.1	Motivacija . . . . .	1
1.2	Cilji . . . . .	2
1.3	Struktura diplomskega dela . . . . .	2
<b>2</b>	<b>Pregled področja</b>	<b>3</b>
<b>3</b>	<b>Algoritmi</b>	<b>5</b>
3.1	Definicije in oznake . . . . .	5
3.2	Algoritem minimizacije nuklearne norme . . . . .	8
3.3	Algoritem praga singularnih vrednosti . . . . .	9
3.4	Algoritem minimizacije prirezane nuklearne norme . . . . .	14
3.5	Izmenjujoč gradientni spust . . . . .	19
3.6	LMaFit . . . . .	21
<b>4</b>	<b>Rezultati</b>	<b>23</b>
4.1	Velika črno-bela slika . . . . .	24
4.2	Vpliv kompleksnosti slik na rekonstrukcijo . . . . .	29
4.3	Rekonstrukcija barvnih slik . . . . .	35
4.4	Vpliv podatka o rangu na rezultate LMaFit in TNNM . . . . .	36
4.5	Rekonstrukcija slike z besedilom . . . . .	39

4.6	Primerjava rezultatov z algoritmom reševanja Laplaceove diferencialne enačbe . . . . .	41
4.7	Povzetek ugotovitev . . . . .	45
<b>5</b>	<b>Zaključek</b>	<b>49</b>

# Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>SVT</b>	Singular Value Thresholding	Prag singularnih vrednosti
<b>NNM</b>	Nuclear norm minimization	Minimizacija nuklearne norme
<b>TNNM</b>	Truncated nuclear norm minimization	Minimizacija prirezane nuklearne norme
<b>ASD</b>	Alternating Steepest Descent	Izmenjajoč gradientni spust
<b>NP</b>	Nondeterministic polynomial time	Nedeterministični polinomni čas
<b>SDP</b>	Semidefinite programming	Semidefinitno programiranje



# Povzetek

**Naslov:** Algoritmi za reševanje problema matričnih napolnitev

**Avtor:** Matej Klančar

Pri problemu matričnih napolnitev obravnavamo matriko, pri kateri nekaterih elementov ne poznamo. Cilj je poiskati take elemente, da bo rang napolnjene matrike minimalen. V diplomskem delu si pogledamo teoretično ozadje petih algoritmov, ki ta problem rešujejo (NNM, SVT, TNNM, ASD, LMaFit), ter jih testiramo. Pri testiranju algoritmov se osredotočamo na rekonstrukcijo slik, kjer vrednosti nekaterih pikslov ne poznamo. Med testiranjem poskušamo odgovarjati na različna vprašanja, ki se nam pri reševanju problema porodijo, ter rezultate interpretirati s predstavljenim matematičnim ozadjem. Prav tako rezultate primerjamo z bolj pogosto metodo rekonstrukcije slik - algoritmom reševanja Laplaceove diferencialne enačbe.

**Ključne besede:** matrične napolnitve, minimizacija ranga, rekonstrukcija slik.



# Abstract

**Title:** Algorithms for solving the matrix completion problem

**Author:** Matej Klančar

The matrix completion problem considers a matrix in which some elements are unknown. The goal is to find such elements, that the rank of the filled matrix will be minimal. In the bachelor's thesis, we look at the theoretical background of five algorithms used to solve this problem (NNM, SVT, TNNM, ASD, LMaFit) and test them. When testing algorithms, we focus on the reconstruction of images where the values of some pixels are unknown. During testing, we try to answer various questions that arise when solving the problem and interpret the results with the presented mathematical background. We also compare the results with a more common method of image reconstruction - the algorithm for solving the Laplace differential equations.

**Keywords:** matrix completion, rank minimization, image reconstruction.



# Poglavlje 1

## Uvod

### 1.1 Motivacija

*Problem matričnih napolnitev (PMN)* sprejme matriko, v kateri nekaterih elementov ne poznamo in sprašuje po določitvi vrednosti neznanih elementov. Napolnjeni matriki pravimo tudi *napolnitev*. Običajen kriterij za določanje neznanih vrednosti je, da bo rang napolnitve najmanjši možen. Ta kriterij je smiseln, ker želimo, da se napolnitev čim bolj prilega znanim podatkom. To pa dosežemo ravno z minimizacijo ranga, saj bo v tem primeru največ vrstic oz. stolpcev linearne odvisnih od preostalih. Ker pa je to zelo zahteven optimizacijski problem, ki spada v razred *NP-polnih* problemov [19], obstaja veliko različnih poenostavitev. Te poenostavitev rešujejo sorodne lažje probleme, pri čemer je cilj najti dovolj dober približek prave rešitve prvotnega problema.

PMN je zaradi uporabe na številnih področjih zelo popularen in dobro študiran, tako na področju matematike kot računalništva. Uporabe segajo na področje priporočilnih sistemih, rekonstrukcije signalov, rekonstrukcije in kompresije slik ter računanje razdalj med napravami. Ker gre pri različnih uporabah za probleme različnih dimenzij, med algoritmi ne obstaja tak, ki bi vedno dajal najbolj kakovostne rezultate in bi bil najhitrejši. Zato je izbira algoritma zelo odvisna tudi od problema samega.

## 1.2 Cilji

Cilj tega diplomskega dela se je seznaniti z nekaj različnimi algoritmi za reševanje PMN, pri čemer želimo spoznati čim več različnih matematičnih orodijih. Podrobno bi radi razumeli matematično ozadje predstavljenih algoritmov, saj samo tako lahko dobro interpretiramo opažanja pri testiranju na primerih. Študirane algoritme želimo implementirati in testirati njihovo delovanje na izbranih primerih uporabe. Po opravljenih testiranjih in interpretaciji ugotovitev bi hoteli predlagati možne izboljšave, ali nov algoritem, ki bi odpravil pomanjkljivosti, opažene pri testiranjih.

Glavni prispevki tega diplomskega dela so predstavitev matematičnega ozadja petih različnih algoritmov za reševanje PMN - s kraticami NNM, SVT, TNNM, LMaFit, ASD, implementacija teh algoritmov v programu Matlab, analiza izbire vhodnih parametrov v algoritmu in analiza kakovosti ter časovne zahtevnosti njihovega delovanja na problemu rekonstrukcije zašumljenih slik. Koda implementacij je na voljo na javnem GitHub repositoriju <https://github.com/selenci/Diploma-Matricne-Napolnitev>.

## 1.3 Struktura diplomskega dela

V poglavju 2 podamo pregled sorodnih dela o iz področja matričnih napolnitev. V poglavju 3 predstavimo matematično ozadje izbranih algoritmov in izpeljemo glavne korake, na katerih temeljijo ti algoritmi. V poglavju 4 predstavimo rezultate naše analize izbranih algoritmov, uporabljenih na problemu rekonstrukcije zašumljenih slikah. Poglavlje je razdeljeno na razdelke, v katerih poskušamo odgovoriti na različna vprašanja, ki se porajajo med rekonstrukcijo. V poglavju 5 diplomsko delo strnemo in navedemo nekaj idej za nadaljnje delo.

# Poglavlje 2

## Pregled področja

Raziskovanje problema matričnih napolnitev (PMN) je še vedno zelo aktivno, saj se pojavljajo številni članki z novimi pristopi in izboljšavami obstoječih metod. Splošnost problema in številne uporabe pojasnjuje motivacijo po iskanju še učinkovitejših algoritmov od mnogih že obstoječih.

Eno prvih del, ki obravnava PMN, je [16]. Delo opiše problem in njegovo naravo, poleg napolnjevanja s ciljem minimalnega ranga napolnitve pa se osredotoči še na druge zahteve za napolnitev. Obravnavane so tudi metode za napolnjevanje matrik s ciljem pozitivne semidefinitnosti napolnitve ali pa s ciljem maksimizacije determinante. Oba problema sta bila obsežno študirana v zadnjih dveh desetletjih prejšnjega 20. stoletja. [2, 7, 1]

Doktorska disertacija [12] podaja pomembne opise pretvorbe problema minimizacije ranga v problem iz področja semidefinitnega programiranja. Ideje in dokazi tega dela služijo kot temelj za algoritme, ki so se razvili v zadnjih letih. Članek [5] opisuje algoritem SVT, ki za svoje delovanje uvede operator praga, ki ignorira singularne vrednosti manjše od praga. Članek tudi pokaže, zakaj je tak način reševanja smiselen. Članek [15] opisuje algoritem TNNM, ki uporabi informacijo o rangu nezašumljene matrike, ter problem rešuje s pomočjo algoritma ADMM [24]. Vir [22] opisuje algoritem ASD, ki isče matriki  $X$  in  $Y$ , katerih produkt bo enak napolnjeni matriki. Matriki isče z uporabo gradientnega spusta, ter izračunom optimalnega koraka po

gradientu v vsaki iteraciji. Algoritem LMaFit, opisan v [23] podobno išče produkt dveh matrik, vendar ta za iskanje uporablja Moore-Penroseov inverz, ki v vsaki iteraciji najde rezultat z metodo najmanjših kvadratov. Obstajajo še številne tudi druge metode, ki slonijo na drugih idejah [17, 18, 8]. Zaradi obsežnosti, si teh v tem diplomskem delu ne bomo pogledali.

Glavna literatura pri nastajanju diplomske naloge je bil pregledni članek [19], ki opiše problem ter v grobem predstavi več algoritmov in jih primerja. Medtem ko opisi pogosto niso bili dovolj podrobni, da bi lahko začel algoritme implementirati, je članek ponujal dobro razumljive opise algoritmov, kot tudi navedel vire, ki so pomagali pri implementaciji. Prav tako je članek podal pomembno primerjavo rezultatov različnih algoritmov.

Kot smo videli zgoraj, obstajajo številni članki s področja PMN. Večina člankov predstavi neko novo metodo, razloži idejo v ozadju, izpelje nekaj konvergenčnih rezultatov in testiranj na izbranem problemu. Zelo malo pa je literature, ki različne metode primerja med seboj in poskuša klasificirati algoritme glede na to, za kateri problem so najprimernejši. Prav tako je navadno pomanjkljivo razloženo, zakaj je bilo testiranje narejeno ravno na izbranih podatkih. Ponekod gre za naključno generirane podatke iz točno določene porazdelitve s točno določeno strukturo, pri čemer od tod ni moč sklepati, kako bi se algoritem obnesel na nekoliko drugačnih podatkih, ki ne bi bili pridobljeni ravno na tovrsten način. V delu se zato želimo osredotočiti tudi na ta vidik, tj. na implementacijo algoritmov in testiranja na podatkih istega tipa.

# Poglavlje 3

## Algoritmi

V tem poglavju predstavimo teoretično ozadje algoritmov za reševanje problema matričnih napolnitev. V posameznih razdelkih predstavimo različne algoritme, povzamemo idejo algoritma za iskanje rešitve, ter razložimo, zakaj deluje. Vrstni red predstavitve algoritmov je postavljen tako, da kjer algoritom razširja idejo prejšnjega, tega predstavimo kasneje.

Razdelek 3.1 predstavi definicije in oznake, ki se čez celotno poglavje pojavljajo. Razdelek 3.2 na kratko predstavi algoritmom minimizacije nuklearne norme (NNM), ter predstavi, kako problem pretvorimo v problem semidefinitnega programiranja. Razdelek 3.3 opisuje algoritmom praga singularnih vrednosti (SVT), ter predstavi idejo za algoritmom. V razdelku 3.4 predstavimo algoritmom minimizacije prirezane nuklearne norme (TNNM), ter algoritmom ADMM, ki ga algoritmom TNNM uporablja. Razdelek 3.5 opisuje, kako algoritmom izmenjujočega gradientnega spusta (ASD) uporablja gradient in optimalni korak za iskanje primerne rešitve. V razdelku 3.6 pa opisemo algoritmom LMaFit, ki za svoje delovanje uporablja Moore-Penroseov inverz.

### 3.1 Definicije in oznake

V tem razdelku uvedemo definicije in oznake, ki se bodo pojavljale v preostanku dela.

1. Z  $\mathbb{R}^{n_1 \times n_2}$  označujemo množico  $n_1 \times n_2$  realnih matrik. Matriko  $A \in \mathbb{R}^{n_1 \times n_2}$  pišemo kot  $A = [a_{ij}]_{i,j}$ .
2.  $\Omega$  je podmnožica množice vseh urejenih parov  $\{(i, j) : i = 1, \dots, n_1, j = 1, \dots, n_2\}$ . V delu je imenujemo **množica znanih vrednosti**.
3. Z  $\mathcal{P}_\Omega : \mathbb{R}^{n_1 \times n_2} \rightarrow \mathbb{R}^{n_1 \times n_2}$  označimo preslikavo, definirano kot

$$[\mathcal{P}_\Omega(A)_{ij}]_{i,j} = \begin{cases} a_{ij}, & (i, j) \in \Omega, \\ 0, & \text{sicer.} \end{cases}$$

Z besedami, preslikava  $\mathcal{P}_\Omega$  ohrani vrednosti matrike na mestih iz množice  $\Omega$ , ostale pa postavi na 0.

4. Naj bo  $\tau > 0$  pozitivno realno število. **Operator praga** ([5])  $\mathcal{D}_\tau : \mathbb{R}^{n_1 \times n_2} \rightarrow \mathbb{R}^{n_1 \times n_2}$  je definiran kot

$$\begin{aligned} \mathcal{D}_\tau(A) &:= U\mathcal{D}_\tau(\Sigma)V^T, \\ \mathcal{D}_\tau(\Sigma) &= \text{diag} \left( \max(\sigma_1 - \tau, 0), \max(\sigma_2 - \tau, 0), \dots, \right. \\ &\quad \left. \max(\sigma_{\min(n_1, n_2)} - \tau, 0) \right), \end{aligned} \tag{3.1}$$

kjer je  $A = U\Sigma V^T$  singularni razcep matrike  $A$ , pri čemer sta  $U \in \mathbb{R}^{n_1 \times n_1}$  in  $V \in \mathbb{R}^{n_2 \times n_2}$  ortogonalni matriki,

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_{\min(n_1, n_2)}) \in \mathbb{R}^{n_1 \times n_2}$$

pa je diagonalna matrika singularnih vrednosti  $\sigma_i$  matrike  $A$ .

5. Matriko  $M \in \mathbb{R}^{n_1 \times n_2}$ , ki ima določene samo nekatere elemente, imenujemo **delno določena matrika**.
6. **Sled** kvadratne matrike  $A = [a_{ij}]_{i,j} \in \mathbb{R}^{n \times n}$  je vsota njenih diagonalnih elementov oz. s formulo

$$\text{Tr}(A) = \sum_{i=1}^n a_{ii}.$$

7. Skalarni produkt  $\langle \cdot, \cdot \rangle$  na vektorskem prostoru pravokotnih  $n_1 \times n_2$  matrik je definiran kot

$$\langle A, B \rangle = \text{Tr}(AB^T).$$

8. **Frobeniusova norma** matrike  $A \in \mathbb{R}^{n_1 \times n_2}$  izhaja iz zgornjega skalarnega produkta in je enaka

$$\|A\|_F = \sqrt{\langle A, A \rangle} = \sqrt{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} a_{ij}^2}.$$

9. **Nuklearna norma** matrike  $A \in \mathbb{R}^{n_1 \times n_2}$  je definirana kot

$$\|A\|_* = \sum_{i=1}^n \sigma_i(A),$$

pri čemer  $\sigma_i(A)$  označuje  $i$ -to največjo singularno vrednost matrike  $A$ . Dokaz, zakaj nuklearna norma upošteva potrebne pogoje norme je opisan v [14, teorem 5.6.18]. Definicija  $\|\cdot\|_1$  je ekvivalentna nuklearni normi.

10. Matrika  $A \in \mathbb{R}^{n \times n}$  je **pozitivno semidefinitna**, če velja  $x^T Ax \geq 0$  za vsak  $x \in \mathbb{R}^n$ . To lastnost matrike  $A$  označimo z  $A \succeq 0$ . Spomnimo se še, da je  $A \succeq 0$  ekvivalentno dejstvu, da so vse lastne vrednosti matrike  $A$  nenegativne.

11. Vektor  $c \in \mathbb{R}^n$  je **subgradient** konveksne funkcije  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  v točki  $x_0$ , če velja

$$\forall x \in \mathbb{R}^n : f(x) - f(x_0) \geq c^T(x - x_0).$$

Z besedami:  $c$  je subgradient funkcije  $f$  v točki  $x_0$ , če premica skozi točko  $(x_0, f(x_0))$  s smernim koeficientom  $c$  leži pod grafom funkcije  $f$  (in se ga dotika v točki  $(x_0, f(x_0))$ ).

12. Z oznako  $\arg \min_x f(x)$  označimo tisto točko iz domene funkcije  $f$ , za katero je vrednost funkcije najmanjša možna.

### 3.2 Algoritem minimizacije nuklearne norme

Ker je minimizacija ranga matrike NP-poln problem [12], algoritmi matričnih napolnitvev temeljijo na metodah, ki rešujejo sorodne, hkrati pa bistveno enostavnejše probleme. **Algoritem minimizacije nuklearne norme (NNM)** uporabi idejo, da je rang matrike tesno povezan z njeno nuklearno normo. Izkaže se [12], da je nuklearna norma konveksna ovojnica funkcije ranga. **Konveksna ovojnica funkcije**  $f : \mathbb{R}^{n_1 \times n_2} \rightarrow \mathbb{R}$  je definirana kot

$$\tilde{f}(A) = \sup\{g(A) \mid g : \mathbb{R}^{n_1 \times n_2} \rightarrow \mathbb{R} \text{ je konveksna funkcija in} \\ \forall A \in \mathbb{R}^{n_1 \times n_2} : g(A) \leq f(A)\}.$$

Problem minimizacije nuklearne norme je možno pretvoriti v optimizacijski problem s področja **semidefinitnega programiranja (SDP)**, ki ga lahko rešujemo z učinkovitimi orodji, npr. knjižnico SeDuMi [21] v Matlabu.

Naj bodo  $C, A_1, \dots, A_\ell \in \mathbb{R}^{n \times n}$  dane matrike in  $b_1, \dots, b_\ell \in \mathbb{R}$  dana števila. **Standardna oblika semidefinitnega programa** je

$$\begin{aligned} \min_{Y \in \mathbb{R}^{n \times n}} \quad & \langle C, Y \rangle, \\ \text{pri pogojih} \quad & \langle A_k, Y \rangle = b_k, \quad k = 1, \dots, \ell, \\ & Y \succeq 0 \end{aligned} \tag{3.2}$$

Vrnimo se k našemu problemu. Naj bo  $M \in \mathbb{R}^{n_1 \times n_2}$  hitrozašumljena matrika (tj. matrika z nekaterimi neznanimi vhodi),  $\Omega$  pa množica urejenih parov, ki predstavljajo mesta, kjer vrednosti matrike poznamo. Problem minimizacije nuklearne norme lahko zapišemo kot

$$\begin{aligned} \min_{\substack{X \in \mathbb{R}^{n_1 \times n_2}, \\ t \in \mathbb{R}}} \quad & t, \\ \text{pri pogojih} \quad & \|X\|_* \leq t, \\ & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{3.3}$$

Hitro se da videti [12, lema 2], da za matriko  $X \in \mathbb{R}^{n_1 \times n_2}$  in  $t \in \mathbb{R}$  velja

$$\|X\|_* \leq t \iff \exists W_1 \in \mathbb{R}^{n_1 \times n_1}, W_2 \in \mathbb{R}^{n_2 \times n_2} : \\ Y = \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix}, \quad Y \succeq 0, \quad \text{Tr}(Y) \leq 2t. \quad (3.4)$$

Minimizacijski problem (3.3) lahko z uporabo (3.4) zapišemo kot

$$\min_{\substack{Y \in \mathbb{R}^{(n_1+n_2) \times (n_1+n_2)}, \\ t \in \mathbb{R}}} \quad 2t, \\ \text{pri pogojih} \quad \text{Tr}(Y) \leq 2t, \quad (3.5) \\ Y \succeq 0, \\ \langle Y, A_{ab} \rangle = M_{ab}, \quad (a, b) \in \Omega,$$

kjer so  $A_{ab} \in \mathbb{R}^{(n_1+n_2) \times (n_1+n_2)}$  matrike, ki imajo vse elemente ničelne, razen tistega na mestu  $(a, n_1+b)$ , ki ima vrednost 1. Programi za reševanje SDP-jev pa lahko sprejmejo in rešujejo probleme v obliki (3.5) [19].

### 3.3 Algoritem praga singularnih vrednosti

**Algoritem praga singularnih vrednosti (SVT)** [5], uporabi idejo, da imajo matrike z majhnim rangom nekaj velikih singularnih vrednosti, ostale pa 0 ali pa vsaj blizu 0. Ključna parametra v SVT-ju sta *izbira premika* in *izbira praga*, algoritem pa temelji na iteraciji

$$X^{(k)} = \mathcal{D}_\tau(Y^{(k-1)}), \quad (3.6)$$

$$Y^{(k)} = Y^{(k-1)} + \delta_k \mathcal{P}_\Omega(M - X^{(k)}), \quad (3.7)$$

kjer so  $\tau > 0$  izbran prag,  $\delta_k$  izbran premik,  $X^{(0)} = 0 \in \mathbb{R}^{n_1 \times n_2}$  in  $Y^{(0)} = 0 \in \mathbb{R}^{n_1 \times n_2}$ . [5]

V nadaljevanju bomo opisali glavno idejo zgornje iteracije. V grobem pa temelji na uporabi metode za iskanje vezanih ekstremov, kjer elementi matrik  $Y^{(k)}$  predstavlja Lagrangove množitelje.

Uvedimo funkcijo

$$f_\tau(X) = \tau \|X\|_* + \frac{1}{2} \|X\|_F^2 \quad (3.8)$$

in optimizacijski problem

$$\begin{aligned} \min_{X \in \mathbb{R}^{n_1 \times n_2}} \quad & f_\tau(X), \\ \text{pri pogojih} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M). \end{aligned} \quad (3.9)$$

Opazimo lahko, da za velike vrednosti  $\tau$  velja  $f_\tau(X) \approx \tau \|X\|_*$ , kar pomeni, da bo s primerno izbranim  $\tau$ , optimizacijski problem minimiziral nuklearno normo.

Denimo, da želimo poiskati minimum funkcije  $f(x)$  pri pogojih  $g_1(x) = g_2(x) = \dots = g_k(x) = 0$ . V teoriji vezanih ekstremov se za tovrstne probleme uvede **Lagrangeovo funkcijo**

$$\mathcal{L}(x, \lambda_1, \lambda_2, \dots, \lambda_k) = f(x) + \lambda_1 g_1(x) + \lambda_2 g_2(x) + \dots + \lambda_k g_k(x),$$

nato pa išče ekstreme med njenimi stacionarnimi točkami. Problemu (3.9) lahko priredimo Lagrangeovo funkcijo

$$\mathcal{L}(X, Y) = f_\tau(X) + \langle Y, \mathcal{P}_\Omega(M - X) \rangle,$$

nato pa iščemo njene stacionarne točke. Zaradi velikega števila parametrov pa ta pristop navadno ni izvedljiv, zato se v SVT-ju uporabi za iskanje ekstremov  $\mathcal{L}(X, Y)$  t.i. *Uzawa algoritem* [5]. Ta ekstreme išče prek iterativnega postopka:

$$X^{(k)} = \arg \min_X \mathcal{L}(X^{(k)}, Y^{(k-1)}), \quad (3.10)$$

$$Y^{(k)} = Y^{(k-1)} + \delta_k \mathcal{P}_\Omega(M - X^{(k)}) \quad (3.11)$$

Izkaže se, da je rešitev (3.10) enaka  $\mathcal{D}_\tau(Y^{(k-1)})$ . To spodaj dokažemo, še prej pa izpeljimo pomožen rezultat.

**Trditev 3.1.** *Velja:*

$$\arg \min_X \mathcal{L}(X, Y) = \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \|X - Y\|_F^2 \right) \quad (3.12)$$

*Dokaz.* Trditev sledi iz krajsega računa:

$$\begin{aligned}
& \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \|X - Y\|_F^2 \right) \\
&= \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \langle X - Y, X - Y \rangle \right) \\
&= \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} (\|X\|_F^2 - 2 \langle X, Y \rangle + \|Y\|_F^2) \right) \\
&= \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \|X\|_F^2 - \langle X, \mathcal{P}_\Omega(Y) \rangle \right) \\
&= \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \|X\|_F^2 + \text{Tr}(-\mathcal{P}_\Omega(X)Y^T) + \text{Tr}(\mathcal{P}_\Omega(M)Y^T) \right) \\
&= \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \|X\|_F^2 + \text{Tr}(\mathcal{P}_\Omega(M - X)Y^T) \right) \\
&= \arg \min_X \left( \tau \|X\|_* + \frac{1}{2} \|X\|_F^2 + \langle Y, \mathcal{P}_\Omega(M - X) \rangle \right) \\
&= \arg \min_X \mathcal{L}(X, Y)
\end{aligned}$$

kjer smo v prvi enakosti uporabili definicijo Frobeniusove norme, v drugi bilinearnost skalarnega produkta, v tretji smo ignorirali konstanto  $\|Y\|_F^2$ , saj ne vpliva na rezultat, in upoštevali  $\mathcal{P}_\Omega(Y^{(k)}) = Y^{(k)}$  za vse  $k \in \mathbb{N}$ . Zadnje dejstvo sledi iz definicije (3.11) in  $Y^{(0)} = 0$ . V četrti enakosti smo upoštevali  $\langle X, \mathcal{P}_\Omega(Y) \rangle = \langle \mathcal{P}_\Omega(X), Y \rangle$ , kar je lahko videti, ko se spomnimo, da za skalarni produkt  $\langle A, B \rangle$  matrik  $A, B \in \mathbb{R}^{n_1 \times n_2}$  velja

$$\langle A, B \rangle = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} a_{ij} b_{ij}.$$

Prišteli smo tudi konstanto  $\text{Tr}(\mathcal{P}_\Omega(M)Y^T)$ , ki ne vpliva na rezultat. V peti enakosti smo upoštevali linearnost sledi in aditivnost operatorja  $\mathcal{P}_\Omega$ , v šesti definicijo skalarnega produkta in v zadnji definicijo Lagrangeove funkcije.  $\square$

**Izrek 3.2.** Za matriki  $X \in \mathbb{R}^{n_1 \times n_2}$  in  $Y \in \mathbb{R}^{n_1 \times n_2}$  velja:

$$\mathcal{D}_\tau(Y) = \arg \min_X \left( \frac{1}{2} \|X - Y\|_F^2 + \tau \|X\|_* \right) \quad (3.13)$$

*Dokaz.* Najprej se spomnimo definicije konveksne funkcije. Funkcija  $f$  je konveksna, če za katerikoli dve točki  $x_1, x_2$  v domeni funkcije  $f$  velja, da je

premica čez ti dve točki na odseku med tema dvema točkama nad grafom funkcije  $f$ . Funkcija  $h(X) := \frac{1}{2}\|X - Y\|_F^2 + \tau\|X\|_*$  je konveksna funkcija, saj potreben pogoj trikotniške neenakosti matričnih norm zagotavlja, da je matrična norma konveksna funkcija. Vsota konveksnih funkcij pa je prav tako konveksna funkcija. Zaradi konveksnosti funkcije  $f$  je subgradient v vsaki točki iz domene dobro definiran. Tega smo definirali v razdelku 3.1. Matrika  $Z \in \mathbb{R}^{n_1 \times n_2}$  je subgradient funkcije  $f$  v točki  $X_0 \in \mathbb{R}^{n_1 \times n_2}$ , če velja

$$\forall X \in \mathbb{R}^{n_1 \times n_2} : f(X) \geq f(X_0) + \langle Z, X - X_0 \rangle.$$

Iz definicije subgradienta sledi, da bo imela funkcija  $f$  minimum v točki  $X'$  natanko tedaj, ko bo ničelna matrika  $\mathbf{0}$  eden izmed subgradientov funkcije  $f$  v točki  $X_0$ .

Izkaže se [5], da je množica subgradientov nuklearne norme v točki  $X$  enaka

$$\partial\|X\|_* = \{UV^* + W : W \in \mathbb{R}^{n_1 \times n_2}, U^*W = \mathbf{0}, WV = \mathbf{0}, \|W\|_2 \leq 1\}. \quad (3.14)$$

kjer  $U\Sigma V^T$  predstavlja SVD razcep matrike  $X$ . S krajšim računom lahko preverimo, da za vsak par  $(i, j)$  velja  $\frac{\partial}{\partial X_{ij}}\|X - Y\|_F^2 = 2(X_{ij} - Y_{ij})$ . Če te parcialne odvode zložimo v matriko, dobimo  $2(X - Y)$ . Od tod sledi, da je  $X - Y$  eden od subgradientov funkcije  $h_1(X) := \frac{1}{2}\|X - Y\|_F^2$  v točki  $X$  [4, razdel. 3.1.3]. Iz teh dveh premislekov sledi, da bo  $X'$  minimum  $h$  natanko tedaj, ko velja

$$\mathbf{0} \in X' - Y + \tau\partial\|X'\|_*. \quad (3.15)$$

Trditev izreka bo sledila, če pokažemo, da velja  $X' = \mathcal{D}_\tau(Y)$ . Najprej razčlenimo SVD razcep matrike  $Y$  kot

$$Y = U_0\Sigma_0V_0^T + U_1\Sigma_1V_1^T$$

kjer  $U_0, \Sigma_0$  in  $V_0$  predstavljajo singularne vrednosti in pripadajoče singularne vektorje večje od  $\tau$ ,  $U_1, \Sigma_1$  in  $V_1$  pa tiste manjše od  $\tau$ . Pokazati želimo, da velja

$$X' = U_0(\Sigma_0 - \tau I)V_0^T,$$

kar je natanko enako  $\mathcal{D}_\tau(Y)$ . S pretvorbo (3.15) pridemo do zapisa

$$Y - X' \in \tau \partial \|X'\|_*. \quad (3.16)$$

Sedaj iščemo tako matriko  $W$ , da bodo zanjo držali potrebni pogoji, podani v (3.14), prav tako pa bo po (3.16) držalo  $Y - X' = \tau(U_0 V_0^T + W)$ . Primerna izbira za  $W$  je  $W = \tau^{-1} U_1 \Sigma_1 V_1^T$ . Zadošča, da pokažemo

$$\begin{aligned} Y - X' &= U_0 \Sigma_0 V_0^T + U_1 \Sigma_1 V_1^T - U_0 (\Sigma_0 - \tau I) V_0^T \\ &= U_0 (\Sigma_0 - \Sigma_0 + \tau I) V_0^T + U_1 \Sigma_1 V_1^T \\ &= \tau U_0 V_0^T + U_1 \Sigma_1 V_1^T \end{aligned}$$

in

$$\begin{aligned} \tau(U_0 V_0^T + W) &= \tau(U_0 V_0^T + \tau^{-1} U_1 \Sigma_1 V_1^T) \\ &= \tau U_0 V_0^T + U_1 \Sigma_1 V_1^T \end{aligned}$$

Sedaj je zgolj potrebno pokazati, da veljajo potrebne lastnosti matrike  $W$ . Po sami definiciji SVD vemo, da so vsi stolpci matrik  $U$  in  $V$  ortogonalni. Torej velja  $U_0^T W = 0$  in  $W V_0 = 0$ . Ker pa ima matrika  $\Sigma_1$  vse elemente manjše od  $\tau$  velja tudi  $\|W\|_2 \leq 1$ .  $\|A\|_2$  je namreč definirana kot največja singularna vrednost matrike  $A$ . S tem smo pokazali, da  $Y - X' \in \tau \partial \|X'\|_*$ .

□

Z uporabo (3.10), (3.11) in izreka 3.2 res pridemo do iteracije (3.6), ki jo uporablja algoritem SVT.

### 3.3.1 Nastavljanje parametrov $\tau$ in $\delta$

Opazimo lahko, da algoritem SVT potrebuje dva parametra,  $\tau$  in  $\delta$ , ki ju moramo izbrati že pred vstopom v algoritom.

Po priporočilih [5] sta primerni izbiri za  $\delta$  in  $\tau$  enaki

$$\delta = 1.2 \frac{n_1 n_2}{m} \quad \text{in} \quad \tau = 5n,$$

pri čemer je  $\tau$  naveden za kvadratne  $n \times n$  matrike. V poglavju z rezultati smo prvotno uporabljali ti dve konstanti, pri čemer smo zaradi pravokotnosti  $n_1 \times n_2$  matrik uporabljali  $\tau = 5\frac{n_1+n_2}{2}$ . V naših testiranjih se je izkazalo, da sta taka parametra dobra za večje matrike. V razdelku 4.2 pa bomo videli, da moramo pri manjših matrikah pogosto zmanjšati premik in povečati prag.

### 3.4 Algoritem minimizacije pritezane nuklearne norme

Kot nam že samo ime pove, je **algoritem minimizacije pritezane nuklearne norme (TNNM)** [15] soroden algoritmu NNM. Dodatna informacija, ki pa jo uporabi TNNM, je rang  $r$  originalne, nezašumljene matrike.

Osrednjo vlogo v algoritmu ima  **$r$ -pritezana nuklearna norma**, ki za dano matriko  $X \in \mathbb{R}^{n_1 \times n_2}$  vrne vsoto njenih  $\min(n_1, n_2) - r$  najmanjših singularnih vrednosti:

$$\|X\|_r = \sum_{i=r+1}^{\min(n_1, n_2)} \sigma_i(X).$$

TNNM rešuje optimizacijski problem

$$\begin{aligned} & \min_{X \in \mathbb{R}^{n_1 \times n_2}} \|X\|_r, \\ & \text{pri pogoju } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M). \end{aligned} \tag{3.17}$$

Cilj algoritma je torej čim bolj zmanjšati najmanjše singularne vrednosti, medtem ko velikih ne omejujemo. S tem problem minimizacije omilimo.

Problem (3.17) lahko zapišemo v ekvivalentni obliki

$$\begin{aligned} & \min_{X \in \mathbb{R}^{n_1 \times n_2}} \|X\|_* - \sum_{i=1}^r \sigma_i(X), \\ & \text{pri pogojih } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M). \end{aligned} \tag{3.18}$$

V izpeljavah, ki sledijo, bomo potrebovali naslednji izrek.

**Izrek 3.3.** Za matrike  $X \in \mathbb{R}^{n_1 \times n_2}$ ,  $A \in \mathbb{R}^{r \times n_1}$  in  $B \in \mathbb{R}^{r \times n_2}$ , ter naravno število  $r \in \mathbb{N}$ , ki zadoščajo pogojem  $r \leq \min(n_1, n_2)$ ,  $AA^T = I_r$  in  $BB^T = I_r$ , velja neenakost

$$\text{Tr}(AXB^T) \leq \sum_{i=1}^r \sigma_i(X) \quad (3.19)$$

*Dokaz.* Velja

$$\text{Tr}(AXB^T) = \text{Tr}(XB^TA) \leq \sum_{i=1}^{\min(n_1, n_2)} \sigma_i(X)\sigma_i(B^TA), \quad (3.20)$$

kjer smo v enakosti uporabili komutativnost  $\text{Tr}(ZW) = \text{Tr}(WZ)$  sledi, v neenakosti pa von Neumannovo neenakost za sled [15].

Po definiciji so singularne vrednosti matrike  $Y$  enake korenom lastnih vrednosti matrike  $Y^TY$ . Torej so singularne vrednosti matrike  $B^TA$  enake korenom lastnih vrednosti matrike  $A^TBB^TA = A^TI_rA = A^TA$ . Ker imata matriki  $XY$  in  $YX$  enake neničelne lastne vrednosti in po predpostavki velja  $AA^T = I_r$ , lahko od tod sklepamo, da ima produkt  $B^TA$   $r$  singularnih vrednosti enakih 1. Zato velja

$$\sum_{i=1}^{\min(n_1, n_2)} \sigma_i(X)\sigma_i(B^TA) = \sum_{i=1}^r \sigma_i(X),$$

kar skupaj z (3.20) dokaže neenakost (3.19) v izreku.  $\square$

**Izrek 3.4.** Naj bo  $X = U\Sigma V^T$  SVD razcep matrike  $X$ . Naj bosta  $A$  in  $B$  matriki, sestavljeni iz prvih  $r$  stolpcev matrik  $U$  in  $V$ . Velja

$$\text{Tr}(AXB^T) = \sum_{i=1}^r \sigma_i(X).$$

*Dokaz.* Označimo matriki  $A$  in  $B$  z  $A = (u_1, \dots, u_r)^T$  in  $B = (v_1, \dots, v_r)^T$ ,

kjer je  $u_i$   $i$ -ti stolpec matrike  $U$  ter  $v_i$   $i$ -ti stolpec matrike  $V$ .

$$\begin{aligned}
 \text{Tr}(AXB^T) &= \text{Tr}((u_1, \dots, u_r)^T X (u_1, \dots, u_r)^T) \\
 &= \text{Tr}((u_1, \dots, u_r)^T U \Sigma V^T (u_1, \dots, u_r)^T) \\
 &= \text{Tr}\left(\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \Sigma \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}\right) \\
 &= \text{Tr}\left(\begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & 0 \\ & & & \ddots & 0 \end{bmatrix}\right) \\
 &= \sum_{i=1}^r \sigma_i(X),
 \end{aligned}$$

kar dokaže trditev izreka.  $\square$

Po izrekih 3.3 in 3.4 velja

$$\max_{\substack{AA^T=I, \\ BB^T=I}} \text{Tr}(AXB^T) = \sum_{i=1}^r \sigma_i(X)$$

Torej je optimizacijski problem (3.18) ekvivalenten problemu

$$\begin{aligned}
 \min_{X \in \mathbb{R}^{n_1 \times n_2}} \quad & \|X\|_* - \max_{\substack{AA^T=I, \\ BB^T=I}} \text{Tr}(AXB^T), \\
 \text{pri pogoju} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M).
 \end{aligned}$$

Sedaj lahko opišemo idejo algoritma TNNM:

1. Izračunamo  $X^{(0)} = \mathcal{P}_\Omega(M)$ .
2. Za  $k = 0, 1, 2, \dots$  ponavljamo iteracijo:
  - (a) Izračunamo SVD razcep  $X^{(k)} = U^{(k)} \Sigma^{(k)} (V^{(k)})^T$ .
  - (b)  $A^{(k)}$  definiramo kot prvih  $r$  stolpcev matrike  $U^{(k)}$ ,  $B^{(k)}$  pa kot prvih  $r$  stolpcev matrike  $V^{(k)}$ .

(c)  $X^{(k+1)}$  je enak rešitvi optimizacijskega problema

$$\min_{X \in \mathbb{R}^{n_1 \times n_2}} \|X\|_* - \text{Tr}(A^{(k)} X (B^{(k)})^T), \quad (3.21)$$

pri pogoju  $\mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)$ .

Z nadaljnjam preoblikovanjem problema (3.21) v ekvivalentnega

$$\min_{X \in \mathbb{R}^{n_1 \times n_2}} \|X\|_* - \text{Tr}(A^{(k)} W (B^{(k)})^T), \quad (3.22)$$

pri pogojih  $W = X$ ,  $\mathcal{P}_\Omega(W) = \mathcal{P}_\Omega(M)$ ,

lahko za reševanje uporabimo **algoritem ADMM** [15].

Gre za reševanje problema vezanih ekstremov, ki ga lahko zapišemo s pomočjo Lagrangeove funkcije, pri čemer algoritom ADMM doda še člen, pomnožen z *regularizacijskim parametrom*  $\beta$ .

$$\mathcal{L}(X, Y, W, \beta) = \|X\|_* - \text{Tr}(AWB^T) + \frac{\beta}{2}\|X - W\|_F^2 + \text{Tr}(Y^T(X - W)).$$

Matriki  $A$  in  $B$  sta v okviru algoritma ADMM konstantni. Ti na začetku nastavimo na vrednost  $A^{(k)}$  in  $B^{(k)}$  iz prejšnje iteracije. Opazimo lahko, da funkcija upošteva zgolj pogoj  $X = W$ . Videli bomo, da algoritom pogoj  $\mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)$  definira posredno, s popravljanjem znanih vrednosti (korak (3.23) spodaj).

Matriko  $X^{(k+1)}$  definiramo kot

$$X^{(k+1)} = \arg \min_X \mathcal{L}(X, Y^{(k)}, W^{(k)}, \beta)$$

Z ignoriranjem konstantnih členov, pa lahko zapišemo

$$\begin{aligned} X^{(k+1)} &= \arg \min_X \left( \|X\|_* + \frac{\beta}{2} \langle X - W^{(k)}, X - W^{(k)} \rangle + \frac{\beta}{2} \langle \frac{2}{\beta} Y, X \rangle \right) \\ &= \arg \min_X \left( \|X\|_* + \frac{\beta}{2} \text{Tr} \left( X^T X - X^T W^{(k)} - (W^{(k)})^T X + (W^{(k)})^T W^{(k)} + \frac{2}{\beta} (Y^{(k)})^T X \right) \right). \end{aligned}$$

Z namenom faktorizacije dodamo konstantne člene

$$-(W^{(k)})^T \frac{1}{\beta} (Y^{(k)}) + \left( \frac{1}{\beta} (Y^{(k)}) \right)^T (-W^{(k)} + \frac{1}{\beta} (Y^{(k)}))$$

in dobimo

$$\begin{aligned}
& \arg \min_X \left( \|X\|_* + \frac{\beta}{2} \operatorname{Tr} \left( X^T (X - W^{(k)} + \frac{1}{\beta} Y^{(k)}) - (W^{(k)})^T (X - W^{(k)} + \frac{1}{\beta} Y^{(k)}) \right. \right. \\
& \quad \left. \left. + \frac{1}{\beta} (Y^{(k)})^T (X - W^{(k)} + \frac{1}{\beta} Y^{(k)}) \right) \right) \\
& = \arg \min_X \left( \|X\|_* + \frac{\beta}{2} \|X - W^{(k)} + \frac{1}{\beta} Y^{(k)}\|_F^2 \right) \\
& = \arg \min_X \left( \frac{1}{\beta} \|X\|_* + \frac{1}{2} \|X - (W^{(k)} - \frac{1}{\beta} Y^{(k)})\|_F^2 \right)
\end{aligned}$$

Po izreku 3.2 uporabljenem za  $X = X^{(k+1)}$ ,  $Y = W^{(k)} - \frac{1}{\beta} Y^{(k)}$  in  $\tau = \frac{1}{\beta}$  pa sledi

$$X^{(k+1)} = \mathcal{D}_{\frac{1}{\beta}}(W^{(k)} - \frac{1}{\beta} Y^{(k)}).$$

Matriko  $W^{(k+1)}$  podobno izračunamo kot

$$\begin{aligned}
W^{(k+1)} &= \arg \min_W \mathcal{L}(X^{(k+1)}, Y^{(k)}, W, \beta) \\
&= \arg \min_W \frac{\beta}{2} \|W - (X^{(k+1)} + \frac{1}{\beta} (A^T B + Y^{(k)}))\|_F^2
\end{aligned}$$

Očitno je

$$W^{(k+1)} = X^{(k+1)} + \frac{1}{\beta} (A^T B + Y^{(k)})$$

Sedaj uporabimo še pogoj  $\mathcal{P}_\Omega(W^{(k+1)}) = \mathcal{P}_\Omega(M)$  in tiste elemente, ki jih poznamo, popravimo.

$$W^{(k+1)} = W^{(k+1)} + \mathcal{P}_\Omega(M - W^{(k+1)}). \quad (3.23)$$

Z besedami, predpis (3.23) preprosto spremeni mesta  $W^{(k+1)}$ , kjer vrednosti poznamo, na znane vrednosti, ostalih pa ne spremeni.

Po algoritmu ADMM [24] na koncu iteracije preprosto posodobimo matriko  $Y$ , kot

$$Y^{(k+1)} = Y^{(k)} + \beta(X^{(k+1)} - W^{(k+1)}).$$

### 3.5 Izmenjujoč gradientni spust

Računanje SVD razcepa je zahtevna operacija, saj ima časovno zahtevnost  $O(n^3)$ , kar je lahko za veliko matrike zelo počasno. Zato je bilo izpeljanih nekaj algoritmov, ki za svoje delovanje ne potrebujejo SVD-ja. **Algoritem izmenjujočega gradientnega spusta (ASD)** [22] temelji na izračunu gradijeta in premiku v smeri tega za nek korak. Ideja algoritma je poiskati matriki  $X \in \mathbb{R}^{n_1 \times r}$  ter  $Y \in \mathbb{R}^{r \times n_2}$ , tako da velja  $\mathcal{P}_\Omega(M) = \mathcal{P}_\Omega(XY)$ . Tudi tu potrebujemo informacijo o rangu matrike, ki jo rekonstruiramo. Ker imata  $X$  in  $Y$  rang največ  $r$ , tudi njun produkt  $XY$  ne bo imel ranga večjega od  $r$ .

Cilj algoritma je rešiti optimizacijski problem

$$\min_{\substack{X \in \mathbb{R}^{n_1 \times r}, \\ Y \in \mathbb{R}^{r \times n_2}}} \frac{1}{2} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)\|_F^2$$

Algoritem problem minimizacije razdeli na dva optimizacijska podproblema, ki ju rešuje izmenično s pomočjo iterativnega postopka

$$X^{(k+1)} = \arg \min_X \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY^{(k)})\|_F^2, \quad (3.24)$$

$$Y^{(k+1)} = \arg \min_Y \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(X^{(k+1)}Y)\|_F^2 \quad (3.25)$$

Za uporabo algoritma ASD potrebujemo gradijeta funkcije  $f(X, Y) = \frac{1}{2} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)\|_F^2$  po obeh spremenljivkah, ki ju izračunamo za vsak element posebej. Za reševanje (3.24) so spremenljivke elementi matrike  $X$ , za reševanje (3.25) pa elementi matrike  $Y$ . Zaradi enostavnosti definiramo dve pomožni funkciji. Funkcija  $f_Y(X)$  je enaka funkciji  $f(X, Y)$ , le da vrednost  $Y$  jemljemo kot konstanto. Simetrično definiramo  $f_X(Y)$ , kjer jemljemo vrednost  $X$  funkcije  $f(X, Y)$  za konstantno. Sedaj preprosto poiščemo oba gradijeta pomožnih funkcij, označena z  $\nabla f_Y(X)$  in  $\nabla f_X(Y)$  ter najboljša premika po gradientih, označena s  $t_x$  in  $t_y$ . Sam algoritem se potem preprosto premika po gradientih za določen premik. Koraka algoritma bomo v vsaki iteraciji definirali kot

$$X^{(k+1)} = X^{(k)} - t_{x^{(k)}} \nabla f_{Y^{(k)}}(X^{(k)}), \quad (3.26)$$

$$Y^{(k+1)} = Y^{(k)} - t_{y^{(k)}} \nabla f_{X^{(k+1)}}(Y^{(k)}).$$

**Izrek 3.5.** *Velja*

$$\begin{aligned}\nabla f_Y(X) &= -(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY))Y^T, \\ \nabla f_X(Y) &= -X^T(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)), \\ t_x &= \frac{\|\nabla f_Y(X)\|_F^2}{\|\mathcal{P}_\Omega(\nabla f_Y(X)Y)\|_F^2}, \\ t_y &= \frac{\|\nabla f_X(Y)\|_F^2}{\|\mathcal{P}_\Omega(X\nabla f_X(Y))\|_F^2}.\end{aligned}$$

*Dokaz.* Pri tem dokazu nam bo prav prišla Diracova  $\Omega$  -  $\delta_{i,j}$ , definirana kot

$$\delta_{i,j} = \begin{cases} 1, & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega. \end{cases}$$

Za gradient funkcije  $f_Y(X)$  po spremenljivki  $X$ , označen z  $\nabla f_Y(X)$ , velja

$$\begin{aligned}\nabla f_Y(X)_{a,b} &= \frac{\partial}{\partial x_{a,b}} \frac{1}{2} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)\|_F^2 \\ &= \frac{\partial}{\partial x_{a,b}} \frac{1}{2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (\delta_{i,j} m_{i,j} - \delta_{i,j} \sum_{k=1}^r (x_{i,k} y_{k,j}))^2 \\ &= \sum_{j=1}^{n_2} (\delta_{a,j} m_{a,j} - \delta_{a,j} \sum_{k=1}^r (x_{a,k} y_{k,j})) (-y_{b,j}) \\ \implies \nabla f_Y(X) &= -(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY))Y^T\end{aligned}$$

Na podoben način lahko pokažemo še

$$\nabla f_X(Y) = -X^T(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY))$$

Poščimo še najboljši korak gradientnega spusta. Cilj je pokazati, da je premik po gradientu s korakom  $t_x$  najboljši. Sam premik smo zgoraj (3.26) definirali kot

$$X^{(k+1)} = X^{(k)} - t_{x^{(k)}} \nabla f_{Y^{(k)}}(X^{(k)})$$

Ker želimo, da bodo znane vrednosti produkta  $X^{(k+1)}Y^{(k)}$  kar se da podobne znanim vrednostim matrike  $M$ , nastavimo  $t_x$  kot

$$t_x = \arg \min_t g(t),$$

kjer je

$$g(t) = \frac{1}{2} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega((X - t\nabla f_Y(X))Y)\|_F^2.$$

Kot v [22] se izkaže, da velja

$$g'(t) = -\|\nabla f_Y(X)\|_F^2 + t\|\mathcal{P}_\Omega(f_Y(X)Y)\|_F^2.$$

Od tod sklepamo, da funkcija  $g(t)$  doseže minimum pri

$$t_x = \frac{\|\nabla f_Y(X)\|_F^2}{\|\mathcal{P}_\Omega(\nabla f_Y(X)Y)\|_F^2}$$

Podobno velja za korak v smeri gradientnega spusta matrike  $Y$ , kjer

$$t_y = \frac{\|\nabla f_X(Y)\|_F^2}{\|\mathcal{P}_\Omega(X\nabla f_X(Y))\|_F^2}$$

□

### 3.6 LMaFit

**LMaFit** [23] je algoritem, ki podobno kot ASD, rešuje problem matričnih napolnitev z uporabo dveh manjših matrik  $X^{n_1 \times r}$  in  $Y^{n_1 \times r}$ . Optimizacijski problem, ki ga rešuje LMaFit je

$$\min_{X,Y,Z} \quad \frac{1}{2} \|XY - Z\|_F^2,$$

pri pogoju  $\mathcal{P}_\Omega(M) = \mathcal{P}_\Omega(Z)$ .

LMaFit za reševanje tega problema uporabi Moore-Penroseov (MP) inverz. V nadaljevanju bomo MP inverz matrike  $A$  označevali z  $A^\dagger$ .

Očitno bo imela funkcija  $f(X, Y, Z) = \frac{1}{2} \|XY - Z\|_F^2$  najmanjšo vrednost natanko tedaj, ko bo  $XY = Z$ . LMaFit izmenično posodablja matrike  $X$ ,  $Y$  in  $Z$ , pri čemer eno spreminja, preostali dve pa fiksira. Matriko  $X$  posodobi po formuli

$$X^{i+1}Y^i = Z^i \quad \Rightarrow \quad X^{i+1} = Z^i(Y^i)^\dagger,$$

pri čemer uporablja dejstvo, da množenje z MP inverzom z desne da najboljši rezultat po metodi najmanjših kvadratov [9, pogl. 3]. Podobno posodobimo matriko  $Y$ .

$$X^{i+1}Y^{i+1} = Z^i \quad \Rightarrow \quad Y^{i+1} = (X^{i+1})^\dagger Z^i.$$

Na koncu le še posodobimo matriko  $Z$ , kot

$$Z^{i+1} = X^{i+1}Y^{i+1} + \mathcal{P}_\Omega(M - X^{i+1}Y^{i+1}).$$

Torej podobno kot prej poskušamo doseči  $XY = Z$ , vendar zaradi omejitve znanih vrednosti matrike  $M$ , na tistem mestu vrednosti popravimo.

# Poglavlje 4

## Rezultati

V tem poglavju opišemo rezultate, ki smo jih pridobili med testiranjem algoritmov. Kot smo omenili v uvodu, smo predstavljene algoritme za matrične napolnitve testirali na slikah z manjkajočimi pikslji. Pri surovih podatkih po-mena numeričnih vrednosti namreč ni tako enostavno interpretirati, zaradi česar težje ovrednotimo koristnost algoritmov.

Poleg točnosti rezultatov različnih metod merimo tudi čas njihovega izvajanja. Metode testiramo na različnih tipih zašumljenih slik, tj. na slikah, ki so zašumljene enakomerno, pa tudi slikah, pri katerih šum predstavlja besedilo, napisano čez sliko.

V tem odstavku opišemo strukturo poglavja. V razdelku 4.1 primerjamo algoritme na problemu rekonstrukcije zašumljene velike, črno-bele slike. V razdelku 4.2 raziščemo vpliv kompleksnosti motivov na slikah na rezultate algoritmov. V razdelku 4.3 preizkusimo in primerjamo dva načina rekonstrukcije podatkov, kjer so nekateri podatki med seboj neodvisni. Iz poglavja 3 vemo, da nekateri algoritmi (TNNM, ASD, LMaFit) za svoje delovanje potrebujejo informacijo o rangu nezašumljenih podatkov. V razdelku 4.4 raziščemo vpliv informacije o rangu na rezultate. V razdelku 4.5 preverjamo delovanje algoritmov na slikah, kjer so neznani podatki zgoščeni. Zgoščen šum je v našem primeru dodano besedilo na sliki, ki ga želimo odstraniti. Razdelek 4.6 pa primerja algoritme z drugačno metodo rekonstrukcije slik,

ki neznane piksle določa prek reševanja Poissonove parcialne diferencialne enačbe. Razdelek 4.7 strne ugotovitve, predstavljene v ostalih razdelkih.

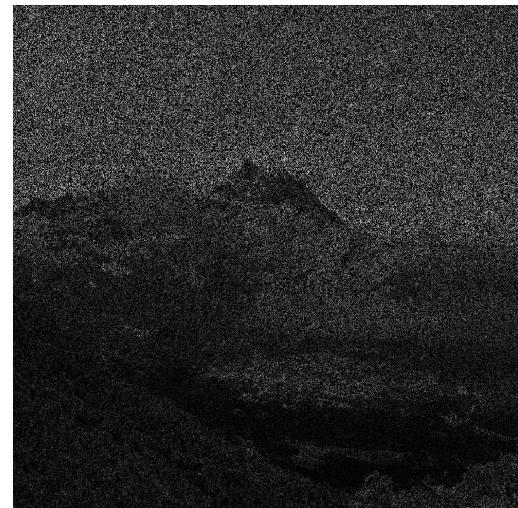
Vse slike si je mogoče v boljši kakovosti ogledati na povezavi <https://cutt.ly/matricne-napolnitve>. Sama implementacija algoritmov pa je dostopna na GitHub repozitoriju <https://github.com/selenci/Diploma-Matricne-Napolnitezv>.

## 4.1 Velika črno-bela slika

Algoritme najprej testiramo na veliki, črno-beli sliki, velikosti  $1000 \times 1000$  pikslov. Velika slika je bila izbrana zato, ker omogoča lažje vizualno ocenjevanje delovanja in pravilnosti algoritmov. Medtem ko se v sledečih razdelkih osredotočamo na bolj specifična vprašanja o kakovosti rekonstrukcij, je cilj tega razdelka pridobiti osnovno informacijo o delovanju algoritmov. To je potem tudi vodilo za vprašanja, ki jih naslovimo v naslednjih razdelkih, kjer se zaradi velike časovne zahtevnosti osredotočimo na manjše slike. Algoritme preizkušamo trikrat, na podatkih z deleži znanih vrednosti 0.35, 0.45 in 0.60.



(a) Originalna slika.



(b) Slika s 35% znanimi podatki.

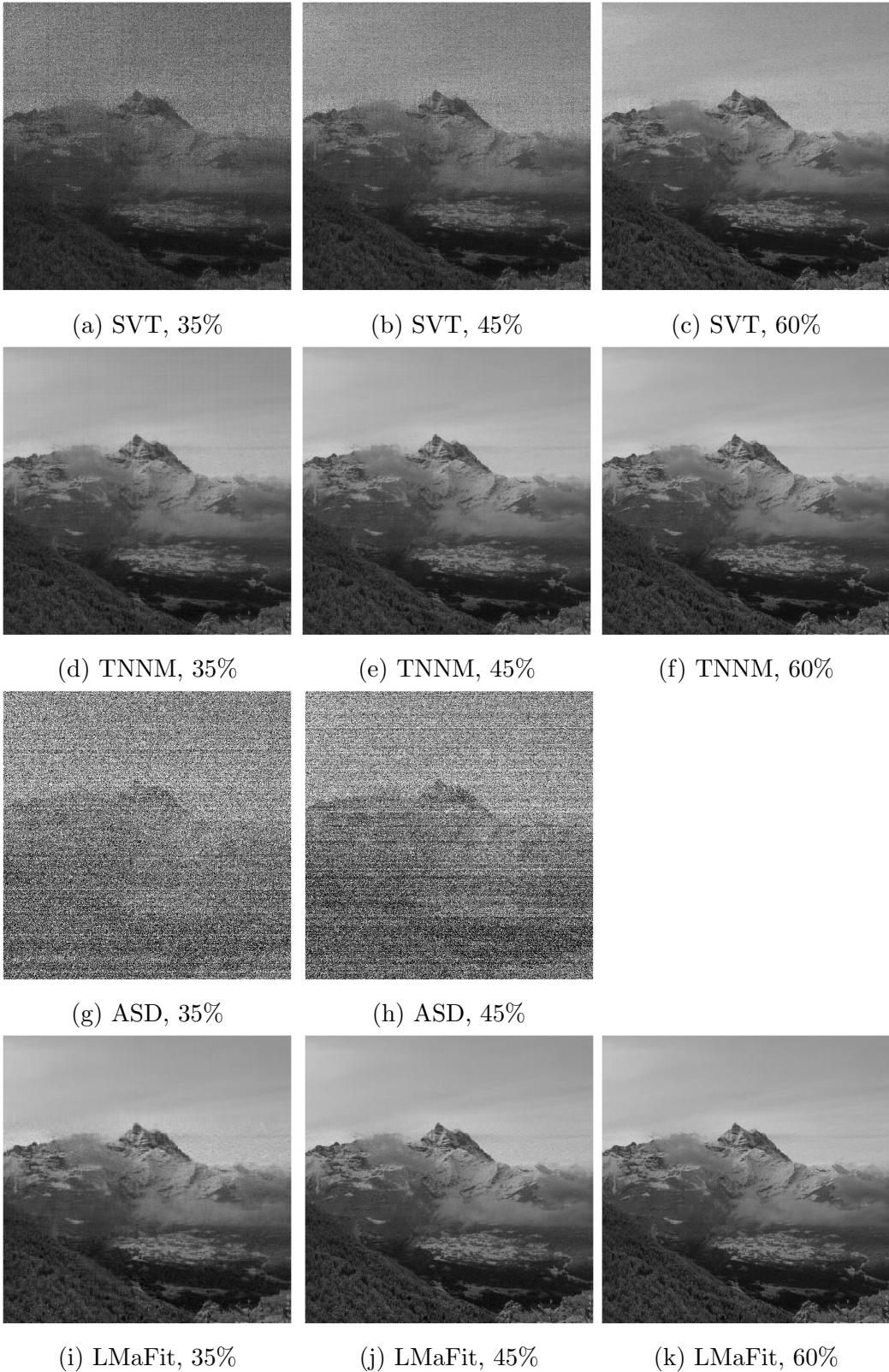


(c) Slika s 45% znanimi podatki.



(d) Slika s 60% znanimi podatki.

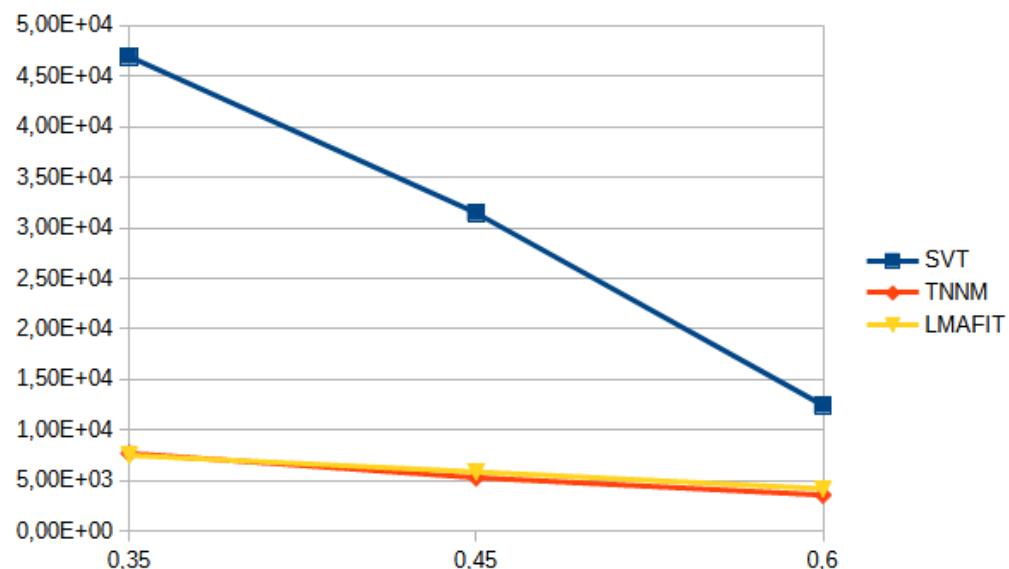
Slika 4.1: Slika uporabljena za rekonstrukcijo. Vir slike: [3].



Slika 4.2: Rekonstrukcija zašumljenih slik z uporabo različnih algoritmov in pri različnih odstotkih znanih vrednosti. Kratica pod sliko označuje uporabljen algoritem, odstotek za vejico pa odstotek znanih vrednosti.

OZP \ Algoritem	SVT	TNNM	LMAFIT	ASD
35%	$4.69 \times 10^4$	$7.70 \times 10^3$	$7.50 \times 10^3$	$3.9743 \times 10^7$
45%	$3.15 \times 10^4$	$5.30 \times 10^3$	$5.87 \times 10^3$	$6.0910 \times 10^7$
60%	$1.25 \times 10^4$	$3.58 \times 10^3$	$4.20 \times 10^3$	-

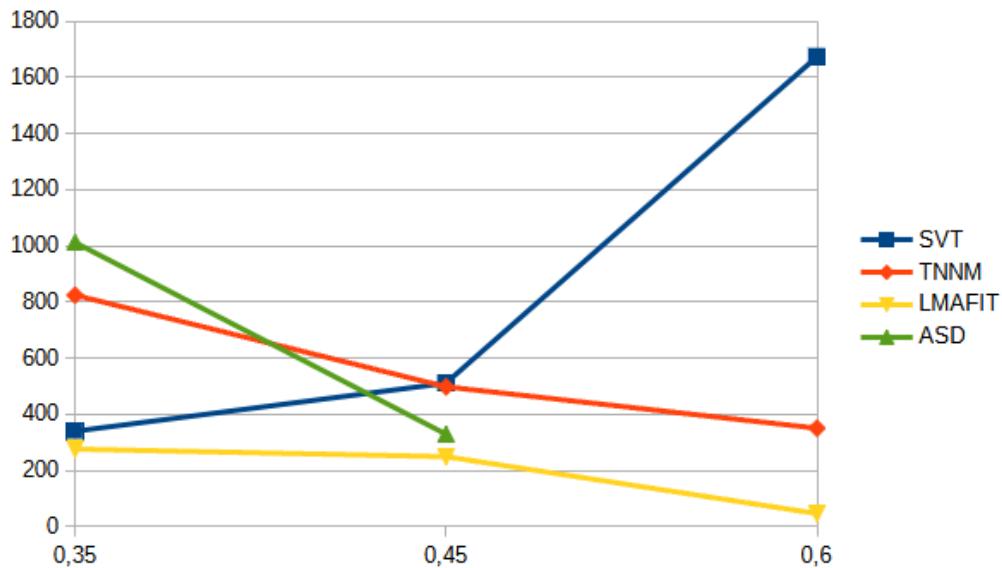
Tabela 4.1: Napake algoritmov, izračunane v Frobeniusovi normi. Kratica OZP stoji za odstotek znanih podatkov.



Slika 4.3: Napake algoritmov v Frobeniusovi normi. Na abscisni osi so deleži znanih podatkov slik.

OZP \ Algoritem	SVT	TNNM	LMAFIT	ASD
35%	338s	824s	275s	1012s
45%	510s	498s	248s	328s
60%	1674s	350s	45.6s	-

Tabela 4.2: Časi do dosega zaustavitvenega pogoja. Kratica OZP stoji za odstotek znanih podatkov.



Slika 4.4: Časi izvajanja algoritmov. Na abscisni osi so deleži znanih podatkov slik.

Opazimo lahko, da je med rezultati velika razlika. Že na pogled lahko zaključimo, da algoritma TNNM in LMaFit delujeta najbolje, SVT nekoliko slabše, medtem ko ASD vrne slabe rezultate. Te si lahko interpretiramo kot posledico lastnosti, da lahko algoritem konča v lokalnem minimumu. Prav tako algoritem ASD ni našel rešitve, ko je imel znanih 60% podatkov. Zato je ta algoritem smiselnouporabljati, kadar imamo manj poznanih vrednosti in dober začetni približek matrik  $X$  in  $Y$  (definirana v razdelku 3.5), ki vstopita v algoritem.

Algoritem NNM smo med rezultati izpustili, saj zaradi velikega števila parameterov pripadajočega SDP-ja problem za zagon potrebuje več pomnilnika, kot ga ima povprečen domač računalnik.. Ta algoritem je zato smiselno uporabljati, kadar imamo matrike manjših dimenzij (algoritem lahko še rešuje probleme, za matrike velikosti  $100 \times 100$  [6]). Ker ta algoritem vrača tiste rešitev, kjer je nuklearna norma najmanjša, bi sicer v teh primerih moral vrniti najboljše rezultate med vsemi algoritmi.

Zaradi zgornjih ugotovitev se v naslednjih razdelkih osredotočamo na algoritme SVT, TNNM in LMaFit.

Sicer bomo čase izvajanja algoritmov podrobneje analizirali v naslednjem razdelku, že sedaj pa lahko navedemo nekaj ugotovitev. Vidimo lahko, da je algoritem SVT veliko počasnejši, kadar je delež znanih vrednosti večji, medtem ko se čas izvajanja ostalih algoritmov z dodajanjem pikslov manjša. Prav tako časovna kompleksnost ni linearna, saj lahko opazimo, da je algoritmu SVT čas izvajanja veliko hitreje naraščal v preskoku iz 45% na 60% znanih vrednoti, kot pa pri preskoku iz 35% na 45% znanih vrednosti.

## 4.2 Vpliv kompleksnosti slik na rekonstrukcijo

Pomembno vprašanje pri študiju algoritmov matričnih napolnitev za rekonstrukcijo slik je vpliv kompleksnosti slik na točnost rezultatov. Ker slika, sestavljena iz naključnih vrednosti pikslov, vizualno ni smiselna, domevamo, da bodo slike s preprostimi motivi napolnjene bolje. Za namene testiranja je torej smiselno izbrati eno preprosto sliko in eno sliko z veliko različnimi motivi. V naših testiranjih uporabljamo sliko knjige in sliko mesta, ki sta velikosti  $300 \times 300$  pikslov.

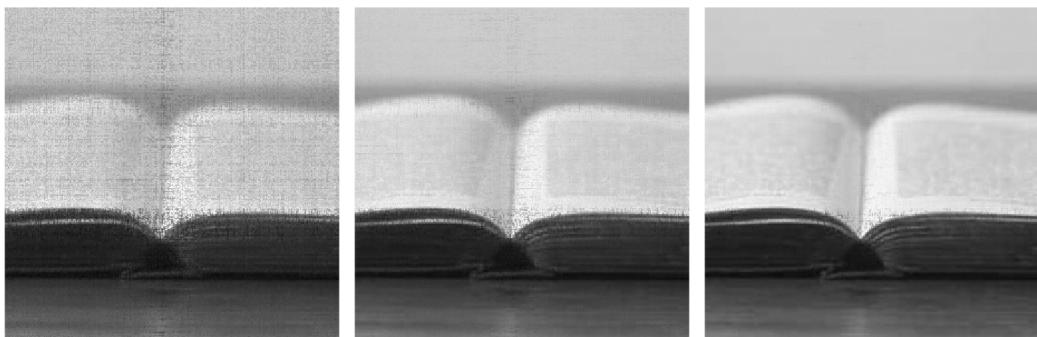


(a) Slika s preprostim motivom.



(b) Slika s kompleksnim motivom.

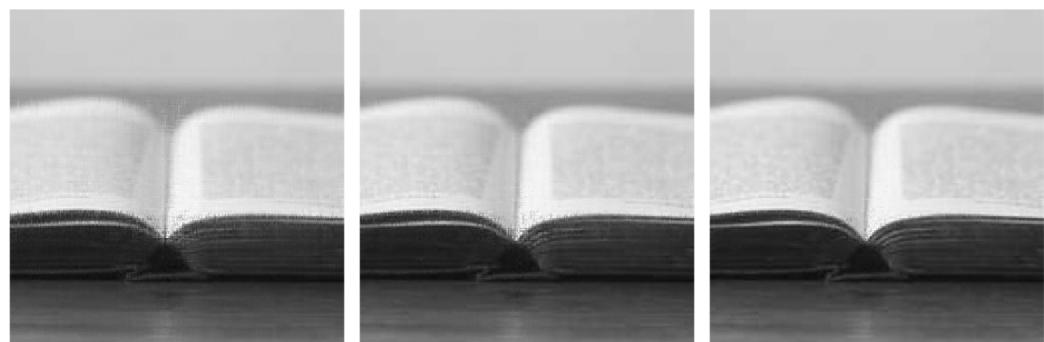
Slika 4.5: Vira slik: [11, 13].



Slika 4.6: Rekonstrukcija preprostega motiva z algoritmom SVT. Odstotek znanih vrednosti slik so bili 35% (leva), 45% (sredinska) in 60% (desna).



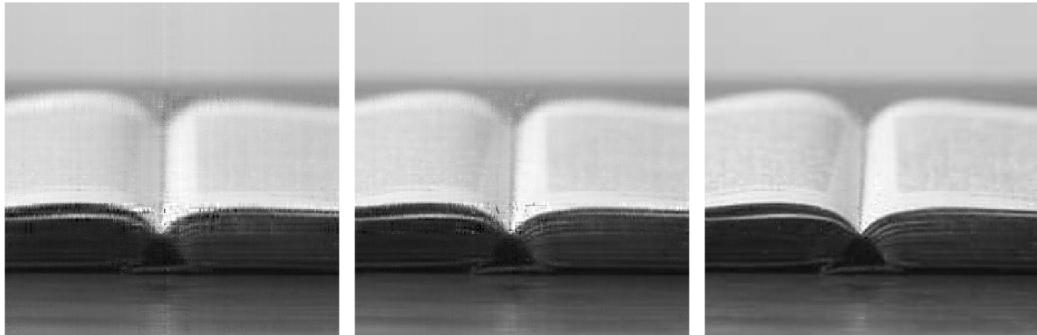
Slika 4.7: Rekonstrukcija kompleksnega motiva z algoritmom SVT. Odstotek znanih vrednosti slik so bili 35% (leva), 45% (sredinska) in 60% (desna).



Slika 4.8: Rekonstrukcija preprostega motiva z algoritmom TNNM. Odstotek znanih vrednosti slik so bili 35% (leva), 45% (sredinska) in 60% (desna).



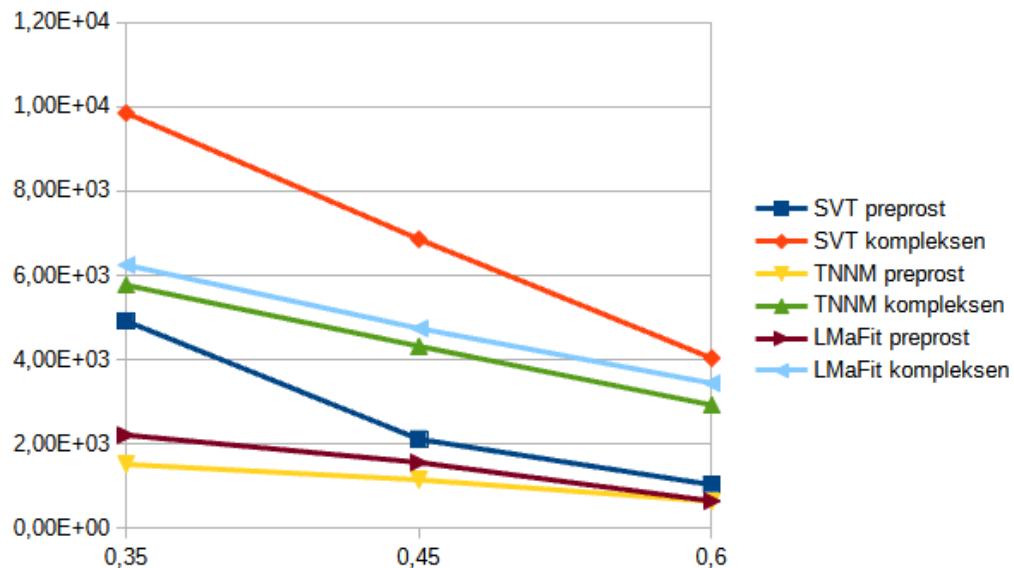
Slika 4.9: Rekonstrukcija kompleksnega motiva z algoritmom TNNM. Odstotek znanih vrednosti slik so bili 35% (leva), 45% (sredinska) in 60% (desna).



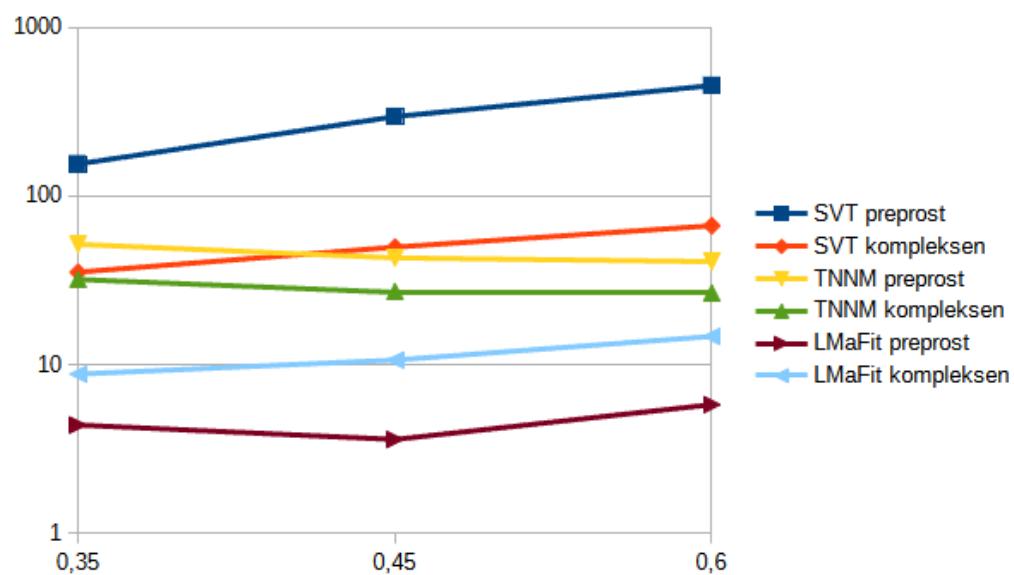
Slika 4.10: Rekonstrukcija preprostega motiva z algoritmom LMaFit. Odstotek znanih vrednosti slik so bili 35% (leva), 45% (sredinska) in 60% (desna).



Slika 4.11: Rekonstrukcija preprostega motiva z algoritmom LMaFit. Odstotek znanih vrednosti slik so bili 35% (leva), 45% (sredinska) in 60% (desna).



Slika 4.12: Graf napak algoritmov v Frobeniusovi normi. Na abscisni osi so deleži znanih podatkov slik.



Slika 4.13: Graf časov izvajanja algoritmov na logaritmični skali. Na abscisni osi so deleži znanih podatkov slik.

Kot smo pričakovali, so rezultati rekonstrukcije slike s preprostim motivom boljši. Prav tako lahko opazimo, da ima delež znanih vrednosti večji vpliv pri sliki s kompleksnim motivom. Napake z dodajanjem informacij torej hitreje padajo pri matrikah večjega ranga. Spomnimo se, da algoritma LMaFit in TNNM za svoje delovanje potrebujeta informacijo o rangu. Pri testiranju je bilo zato potrebno kompleksni sliki podati večjo vrednost ranga, da sta lahko algoritma prišla do dobrih rezultatov.

Točnost algoritmov pa ostaja zelo podobna rekonstrukciji velike slike, torej z najboljšimi rezultati pridobljenimi z algoritmom TNNM, nato z LMaFitom in z najslabšimi rezultati s SVT-jem.

Časi izvajanja pa tu niso tako intuitivni. Prva glavna ugotovitev je, da algoritom SVT potrebuje veliko več časa tako pri preprostem kot tudi pri kompleksnem motivu. V razdelku 3.3.1 smo omenili pripomoček parameter praga iz [5], vendar je v naših testiranjih ta (**v našem primeru bi bil prag nastavljen na 1500**) vrnil slabe rezultate. S preizkušanjem smo prag večali in prišli do vrednosti 7200, ki je dala primerljive rezultate z ostalimi algoritmi. Prav tako je bilo za slike z manj znanimi vrednostmi potrebno korak zmanjšati, sicer algoritom ni konvergiral. V primeru preprostega motiva lahko sicer pričakujemo, da bomo imeli manj zelo velikih singularnih vrednosti v primerjavi s kompleksnim motivom. Zaradi tega se algoritom počasneje premika in dolgo išče rešitev. Iz tega sledi ugotovitev, da je algoritom SVT bolj smiselno uporabljati za kompleksne motive. Pri preprostem motivu sicer prag res lahko nastavimo na manjše vrednosti, s čimer algoritom pospešimo, vendar s tem dobimo slabši rezultat.

Naslednja pomembna ugotovitev je, da delež znanih vrednosti različno vpliva na čas izvajanja. Tudi ta faktor je torej lahko pomemben pri izbiri algoritma za reševanje problema. Algoritom SVT potrebuje za rekonstrukcijo več časa, kadar ima poznanih več vrednosti, medtem ko se algoritmu TNNM z deležem znanih vrednosti čas izvajanja manjša. Algoritmu LMaFit težko določimo pravilo, saj je njegovo obnašanje odvisno od primera do primera. To je razvidno že iz slike 4.13, kjer je kompleksen motiv potreboval najmanj

časa za rekonstrukcijo slike s 35%, preprost pa s 45% znanih podatkov. Ker pa algoritom LMaFit začne iteracije z naključnima matrikama  $X$  in  $Y$ , se čas izvajanja zelo razlikuje ob različnih ponovitvah. Smiselno bi bilo razmisiliti o implementaciji algoritma, kjer bi začeli z več pari matrik  $X$  in  $Y$ , ter po nekaj iteracijah na vseh parih, algoritom nadaljevali zgolj na paru, ki najhitreje konvergira.

### 4.3 Rekonstrukcija barvnih slik

Naslednje smiselno vprašanje je, kako dobro algoritmi delujejo za rekonstrukcijo barvnih slik. Barvne slike so podane kot kombinacija barvnih kanalov rdeče, zelene in modre barve, pri čemer je vsak kanal predstavljen z matriko vrednosti pikslov. V tem razdelku nas bo zanimalo, ali je bolje napolnjevati matrike vsakega barvnega kanala posebej, ali je bolje matrike kanalov združiti v večjo pravokotno matriko in napolniti to večjo matriko. V prvem primeru algoritom uporabimo trikrat, medtem ko v drugem definiramo veliko matriko oblike

$$A = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

kjer  $R$ ,  $G$ ,  $B$  zaporedoma predstavljajo matrike vrednosti rdečega, zelenega in modrega kanala. Vsi testi v tej fazi so bili izvedeni na podatkih, kjer imamo poznanih 35% informacij.

Tip rekonstrukcije	Algoritem	SVT	TNNM	LMAFIT
Enojna rekonstrukcija	$1.50 \times 10^4$	$9.60 \times 10^3$	$1.10 \times 10^4$	
Trojna rekonstrukcija	$1.66 \times 10^4$	$9.79 \times 10^3$	$1.06 \times 10^4$	

Tabela 4.3: Napake algoritmov izračunane v Frobeniusovi normi.

Tip rekonstrukcije \ Algoritem	SVT	TNNM	LMAFIT
Enojna rekonstrukcija	352s	124s	66s
Trojna rekonstrukcija	112s	100s	46s

Tabela 4.4: Časi do dosega zaustavitvenega pogoja.

Iz rezultatov v tabelah 4.3 in 4.4 opazimo, da obe metodi vrnete približno enako dobre rezultate, prav tako tudi vizualno med rezultati težko opazimo razlike. Hkrati pa vidimo, da je rekonstrukcija pri vseh testiranih algoritmih hitrejša, če ločene barvne kanale rekonstruiramo posebej. Zaključimo lahko, da je smiselno med seboj neodvisne podatke ločiti in jih obravnavati samostojno. Rezultat je smiseln, saj nam iskanje podobnosti med nepodobnimi podatki poveča količino dela.

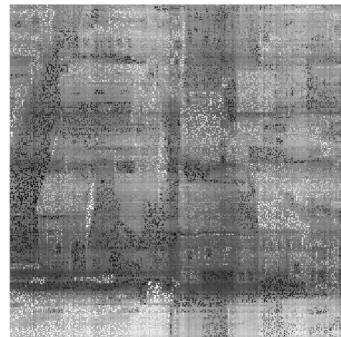
## 4.4 Vpliv podatka o rangu na rezultate LMaFit in TNNM

Spomnimo se, da algoritma TNNM in LMaFit za svoje izvajanje potrebujejo informacijo o rangu (v nadaljevanju jo bomo imenovali *parameter*). V tem podpoglavlju bomo poskušali odgovoriti na vprašanje, kako pri obeh algoritmih ta informacija vpliva na rezultate in hitrost izvajanja. Za namene testiranja smo algoritem na isti sliki pognali večkrat, pri čemer smo postopoma povečevali parameter. Ponovno smo teste izvajali na sliki mesta, z znanim deležem podatkov enakim 0.35.

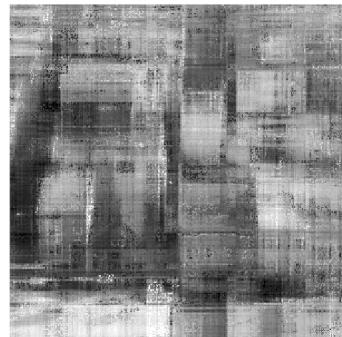
### 4.4.1 LMaFit

Algoritem LMaFit smo testirali šestkrat, s parametrom nastavljenim na 1, 5, 10, 22, 25 in 60. Vrednost 22 je bila izbrana, ker je ob večkratnem zagonu programa pri različnih parametrih vrnila najboljši rezultat. Posledično sta bili vrednosti 25 in 60 izbrani z namenom opazovanja, kakšne rezultate

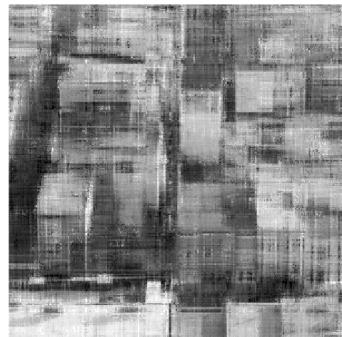
pridobimo z nadaljnjam povečanjem ranga rekonstruirane matrike. Tu pa se je vredno spomniti, da medtem ko parameter res določa rang rekonstrukcije neznanih elementov, pred vrnitvijo rezultatov algoritmom LMaFit znana mesta popravi, zaradi česar rang rezultata ni nujno enak parametru.



(a) Rekonstrukcija s para-  
metrom 1.



(b) Rekonstrukcija s para-  
metrom 5.



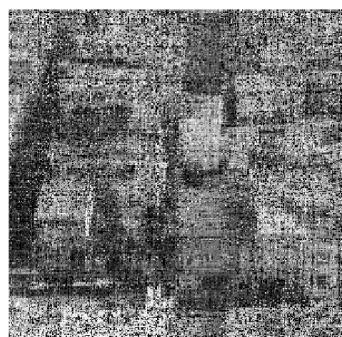
(c) Rekonstrukcija s para-  
metrom 10.



(d) Rekonstrukcija s para-  
metrom 22.



(e) Rekonstrukcija s para-  
metrom 25.



(f) Rekonstrukcija s para-  
metrom 60.

Parameter	Napaka (v $\ \cdot\ _F$ )	Čas izvajanja
1	$1.11 \times 10^4$	0.04s
5	$8.05 \times 10^3$	0.41s
10	$6.61 \times 10^3$	0.42s
22	$6.25 \times 10^3$	8.78s
25	$6.57 \times 10^3$	41.26s
60	$2.40 \times 10^4$	325.31s

Tabela 4.5: Rezultati rekonstrukcije algoritma LMaFit za različne parametre.

Na rekonstruiranih slikah lahko opazimo izboljševanje podrobnosti slike vse do parametra 22. Vidimo pa lahko tudi, da pride pri prevelikem parametru do preobrata. Rezultati se ponovno slabšajo, algoritem pa postane počasnejši. Od tod lahko zaključimo, da je pravilna izbira parametra ključna. Vredno je omeniti še, da za nekatere izbire ranga, algoritem ne konvergira. V primeru zgornje slike za izbiro ranga 30, algoritem LMaFit ni našel rešitve.

#### 4.4.2 TNNM

V tem razdelku predstavimo rezultate vpliva parametrov na delovanje algoritma TNNM. Zaradi večje časovne zahtevnosti, algoritem TNNM testiramo zgolj trikrat, na parametrih 1, 5 in 12. Za večje parametre je algoritem konvergiral zelo počasi, zaradi česar se testiranju teh izognemo. Tukaj se je pomembno spomniti, da parameter pri algoritmu TNNM ne določa samega ranga rezultata, vendar zgolj vpliva nanj. V algoritmu namreč omejimo samo ranga nastopajočih matrik  $A_l$  in  $B_l$ , prek katerih dobimo rezultat (algoritem 3.22). Zaradi tega je težje določiti pravilo, kateri parameter je najboljši.



(a) Rekonstrukcija s par- (b) Rekonstrukcija s par- (c) Rekonstrukcija s para-  
metrom 1. metrom 5. metrom 12.

Parameter	Napaka ( $v \ \cdot\ _F$ )	Čas izvajanja
1	$5.70 \times 10^3$	35.9s
5	$5.46 \times 10^3$	481s
12	$5.27 \times 10^3$	930s

Slika 4.16: Rezultati rekonstrukcije algoritma TNNM za različne parametre.

Sami rezultati so vizualno med seboj delujejo podobni. Numerični izračun napak sicer pokaže, da se napaka počasi zmanjšuje s povečevanjem ranga, vendar pa se čas reševanja zelo hitro povečuje. Zato se je potrebno odločiti, kako dober rezultat potrebujemo in ali smo točnost pripravljeni kompenzirati z bistveno večjo časovno zahtevnostjo rekonstrukcije. Seveda pa velja povedati, da je že pri parametru 1 TNNM dosegel najboljše rezultate izmed vseh algoritmov, obravnavanih v tej diplomski nalogi.

## 4.5 Rekonstrukcija slike z besedilom

V tem razdelku bomo preizkušali učinkovitost algoritmov na slikah, kjer se želimo znebiti nekega besedila na sliki. Gre za drugačno vrsto šuma, kjer so namesto enakomerne razporeditve neznani podatki zgoščeni na določenem delu slike. V našem primeru je bil delež znanih podatkov enak 92%, kar je bistveno več kot v primerih, ki smo si jih ogledali doslej.



Slika 4.17: Slika z besedilom.

(a) Rekonstrukcija z algo-  
ritmom SVT.(b) Rekonstrukcija z algo-  
ritmom TNNM.(c) Rekonstrukcija z algo-  
ritmom LMaFit.

	Napaka ( $v \parallel \cdot \parallel_F$ )	Čas izvajanja (s)
SVT	$4.64 \times 10^3$	674s
TNNM	$2.64 \times 10^3$	83.6s
LMaFit	$3.17 \times 10^3$	0.95s

Tabela 4.6: Rezultati rekonstrukcije različnih algoritmov.

Ponovno lahko opazimo, da je najboljši rezultat vrnil algoritmom TNNM. Pri algoritmu SVT lahko opazimo sledi besedila, zaradi česar je tudi sama napaka pri tem algoritmu večja. Algoritmom LMaFit ni skonvergiral za vrednosti parametra ranga večje od 16, zaradi česar je sam rezultat končnega produkta matrik  $X$  in  $Y$  slab. Opazimo lahko, da po rekonstrukciji manjkajočih vrednosti večina slike izgleda pravilno, še vedno pa je mogoče opaziti

obriše besedila. Rezultati teh testov nam pokažejo pomembnost vrste šuma, saj kljub velikemu deležu znanih podatkov, algoritmi večine podatkov ne morejo kakovostno uporabiti.

## 4.6 Primerjava rezultatov z algoritmom reševanja Laplaceove diferencialne enačbe

Diplomsko delo [10] se ukvarja z rekonstrukcijo slik prek reševanja sistemov Laplaceovih diferencialnih enačb. Ta način rekonstrukcije je v praksi pogosto uporabljen, še posebej na zašumljenih slikah in pri odstranjevanju motivov s slik. Zato je smiselno je primerjati rezultate takšne rekonstrukcije z metodami rekonstrukcij, ki temeljijo na problemu matričnih napolnitev. Laplaceova enačba je definirana kot

$$-\frac{\partial^2 v(x, y)}{x^2} - \frac{\partial^2 v(x, y)}{y^2} = 0.$$

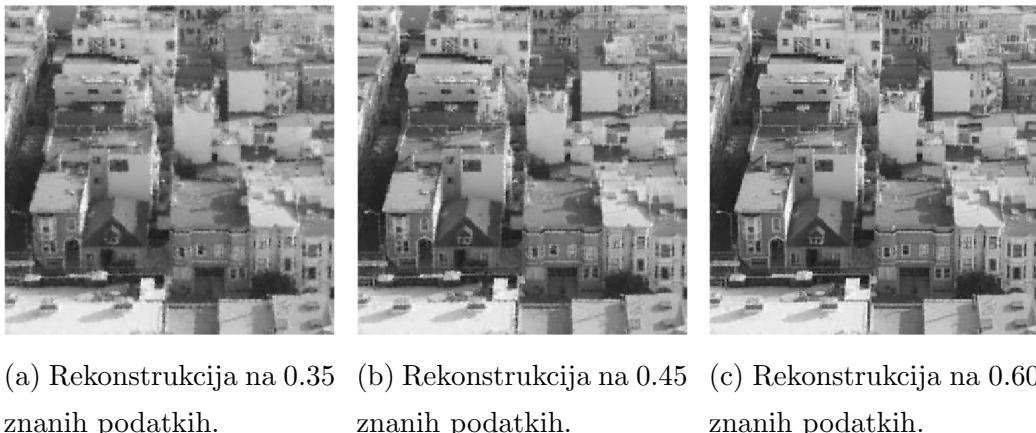
kjer je  $v(x, y)$  vrednost piksla v točki  $(x, y)$ . To je parcialna diferencialna enačba, ki se jo običajno rešuje z uporabo metode *končnih differenc*. Pri tej metodi odvode aproksimiramo z uporabo diferenčnega kvocienta kot

$$\begin{aligned} -\frac{\partial^2 v(x, y)}{x^2} &\approx \frac{2v_{i,j} - v_{i-1,j} - v_{i+1,j}}{h^2}, \\ -\frac{\partial^2 v(x, y)}{y^2} &\approx \frac{2v_{i,j} - v_{i,j-1} - v_{i,j+1}}{h^2}. \end{aligned}$$

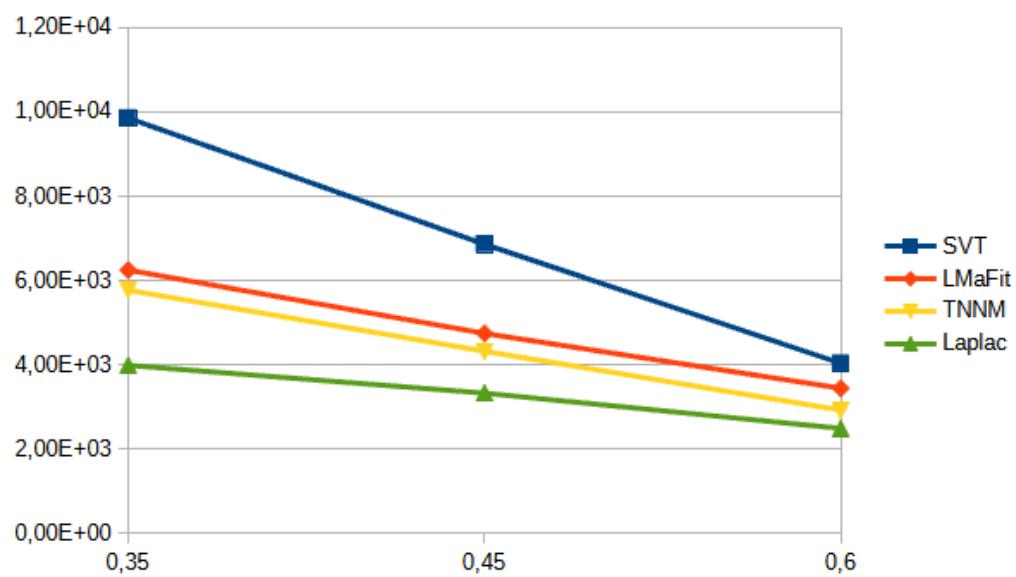
Z uporabo *Jacobijeve iteracije* [9, pogl. 6] lahko problem rešujemo iterativno, tako da neznane vrednosti v vsaki iteraciji posodobimo po formuli

$$u_{i,j}^{(k+1)} = \frac{1}{4}(u_{i-1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}).$$

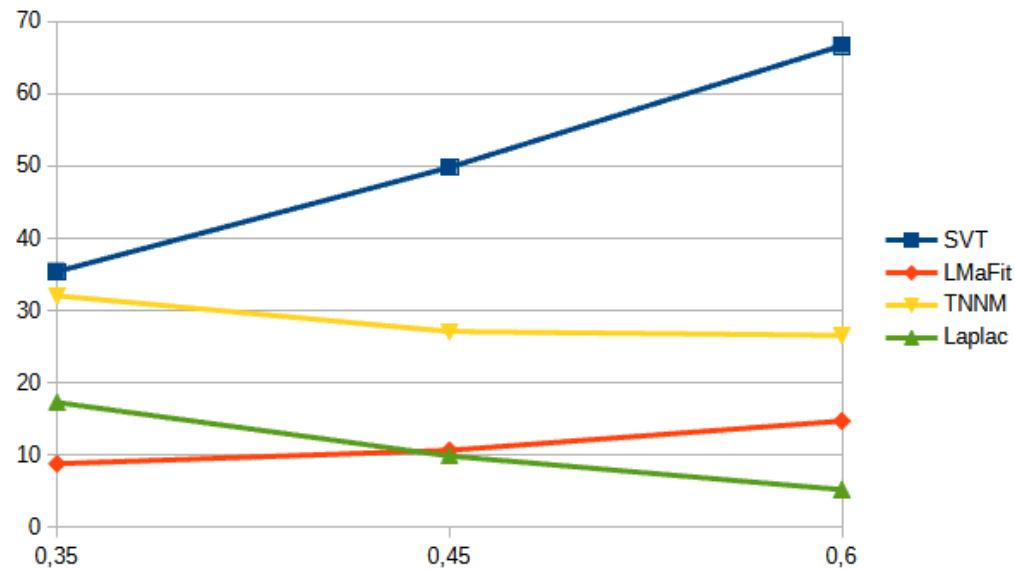
Zaradi lastnosti vrstične diagonalne dominantnosti matrike Jacobijeve iteracije velja, da bo iteracija konvergirala. Spodaj si lahko ogledamo rezultate rekonstrukcij zašumljene slike mesta.



(a) Rekonstrukcija na 0.35 znanih podatkih.  
 (b) Rekonstrukcija na 0.45 znanih podatkih.  
 (c) Rekonstrukcija na 0.60 znanih podatkih.

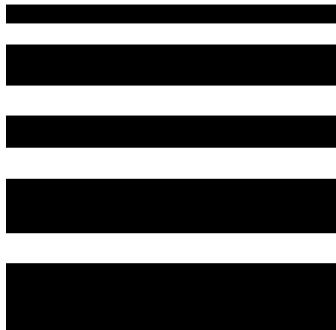


Slika 4.20: Napaka rekonstrukcij slike mesta v Frobeniusovi normi. Na abscisni osi so deleži znanih podatkov slik.

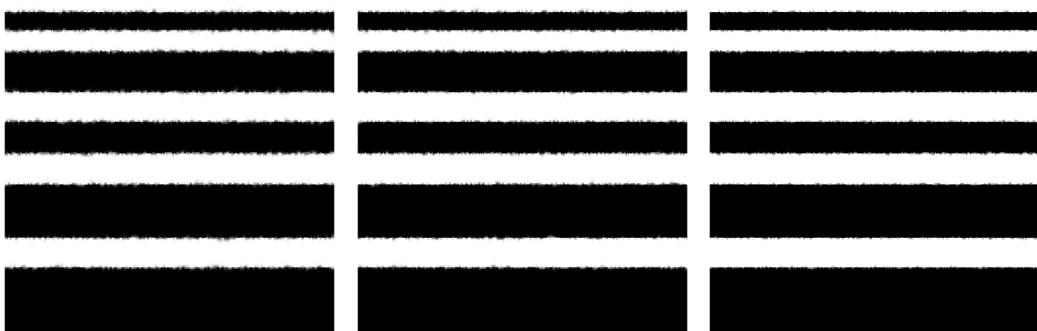


Slika 4.21: Čas izvajanja rekonstrukcije slike mesta. Na abscisni osi so deleži znanih podatkov slik.

Vidimo, da je algoritem tako hitrejši, kot bolj točen. Vendar je pri primerjavi potrebno upoštevati, da se tak algoritem zanaša na lokalno podobnost podatkov, tj. sosedne točke imajo podobne vrednosti barvnih kanalov. Pri problemu minimizacije ranga matrik pa se na take podobnosti ne moremo zanašati. Omenili smo že, da je pomembna uporaba algoritmov matričnih napolnitev v priporočilnih sistemih. V takem primeru sosednost nima pomena, saj imata lahko uporabnika v sosednjih vrsticah povsem različne preference. Prav tako si je lahko zamisliti sliko, kjer bi reševanje Laplaceove enačbe vrnilo slab rezultat. Tak primer je lahko preprosta dvobarvna slika, sestavljena iz več pasov. Očitno je, da ima originalna slika rang 1.

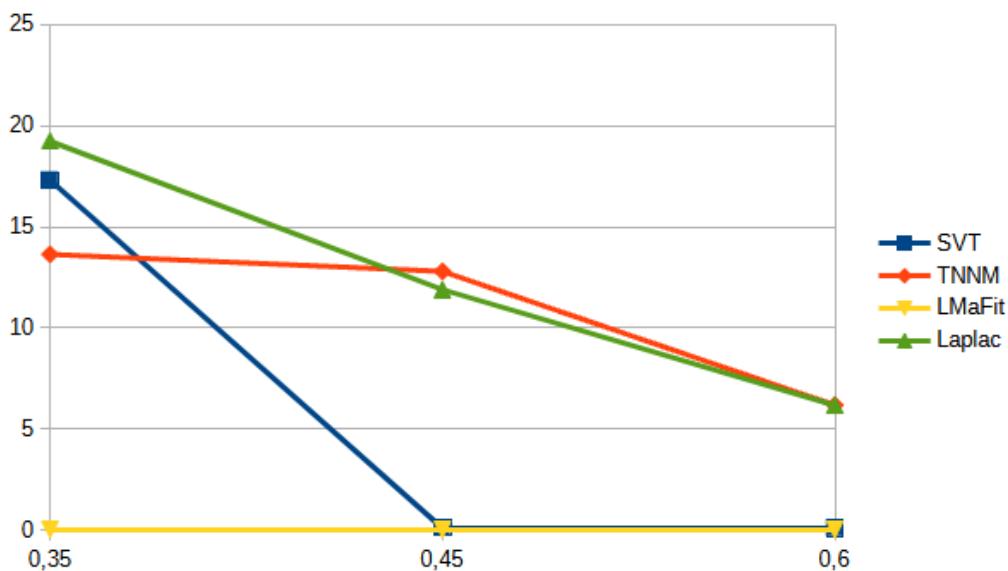


Slika 4.22: Dvobarvna slika.



(a) Rekonstrukcija na 35% znanih podatkih. (b) Rekonstrukcija na 45% znanih podatkih. (c) Rekonstrukcija na 60% znanih podatkih.

Slika 4.23: Rekonstrukcija z uporabo algoritma prek reševanja sistema Laplaceovih diferencialnih enačb.



Slika 4.24: Čas izvajanja rekonstrukcije dvobarvne slike. Na abscisni osi so deleži znanih podatkov slik.

Medtem ko so algoritmi SVT, TNNM in LMaFit sliko rekonstruirali točno, je algoritem prek reševanja Laplaceove enačbe, kot pričakovano, tu imel več težav. Prav tako je v večini primerov potreboval več časa.

## 4.7 Povzetek ugotovitev

Tekom poglavja smo predstavili in interpretirali rezultate številnih testov algoritmov matričnih napolnitev ter analizirali različna vprašanja o rekonstrukciji slik. Postopoma smo predstavljali ugotovitve, ki jih v tem razdelku povzamemo.

Najprej smo algoritme testirali na veliki sliki, ter ugotovili, da algoritem ASD na zašumljenih slikah ne deluje najbolje. V razdelku 4.2 smo testirali razliko v rekonstrukciji preproste in kompleksne slike. Ugotovili smo, da je rekonstrukcija preprostega motiva pri vseh algoritmih točnejša. Videli smo tudi, da pri algoritmu SVT pomembna izbira praga, saj je ob istem pragu preprost motiv potreboval veliko več časa do konvergencije. V istem razdelku

smo opisali tudi, kako odstotek znanih podatkov vpliva na čas izvajanja. Videli smo, da algoritem TNNM konvergira hitreje, kadar ima več poznanih podatkov, medtem ko se algoritem SVT upočasni. Ugotovili smo tudi, da je algoritmu LMaFit težko določiti pravilo o času izvajanja.

V razdelku 4.3 smo preverili, kako rekonstruirati barvne slike. Ugotovili smo, da je zaradi časa izvajanja bolje ločiti neodvisne podatke (različne barvne kanale) in jih rekonstruirati posebej. Preverili smo tudi pomembnost parametra, povezanega z rangom algoritmov (razdelek 4.4). Ta parameter uporabljata algoritma TNNM in LMaFit. Videli smo, da je za dobre rezultate dobra izbira parametra ključna. Prednost algoritma SVT je ta, da ne zahteva izbiranja parametra. Če o rangu nezašumljene matrike ne vemo ničesar, je torej smiselno razmisljiti o uporabi tega algoritma. V okviru razdelka 4.5 smo preizkusili še drugačno vrsto šuma, saj smo iz slike odstranjevali besedilo. Videli smo, da kljub visokemu deležu znanih podatkov, rezultati niso bili preveč dobri, saj je del besedila po rekonstrukciji še vedno viden. S tem smo pokazali, da tudi sama vrsta šuma vpliva na rekonstrukcijo.

Poglavlje smo zaključili z razdelkom 4.6, kjer smo primerjali rekonstrukcijo naših algoritmов z algoritmom prek reševanja Laplaceovih enačb. Opazili smo, da je ta metoda boljša, vendar pokazali tudi primer, kjer zanašanje na lokalno podobnost vrne slab rezultat. Rekonstrukcija slik je bila v okviru diplomskega dela izbrana iz več razlogov. Glavni razlog je ta, da smo lahko rezultate ocenili ne le numerično prek vrednosti napak v neki normi, pač pa tudi vizualno. Tudi v originalnih člankih s predstavljenimi algoritmi je velikokrat testiranje narejeno na primerih rekonstrukcij slik. Seveda so opisani algoritmi bolj kot na sami rekonstrukciji slik uporabni na drugih področjih, npr. priporočilnih sistemih, kjer so lahko podatki v sosednjih vrsticah povsem neodvisni. Testiranje na takih podatkih je bistveno težje, saj bi potrebovali ogromne baze podatkov, kjer bi poznali vse podatke v neki podmatriki, jih zašumili, nato pa rekonstruirali in računali napake. Obstajajo sicer velike baze podatkov za priporočilne sisteme (npr. [20]), vendar ti podatki praviloma nimajo velikih povsem napolnjenih podmatrik, tako da rezultatov naših

algoritmov ne bi mogli primerjati s pravimi vrednostmi. To bi šlo zgolj na majhnih matrikah. V člankih zato algoritme velikokrat testirajo kar na naključno generiranih matrikah iz neke porazdelitve (npr. vsak vhod pride iz enakomerne porazdelitve na intervalu) ali pa produktih naključno generiranih matrik. Če bi se odločili za ta pristop, ne bi mogli vizualno ocenjevati rezultatov, pa tudi same razlike v normah napak numeričnih rezultatov ne bi mogli tako lepo interpretirati.



# Poglavlje 5

## Zaključek

V tem diplomskem delu smo si ogledali pet različnih algoritmov - NNM, SVT, TNNM, LMaFit, ASD - za reševanje problema matričnih napolnitev, pri čemer vsak temelji na drugačnih matematičnih orodjih. NNM ključno uporablja orodja iz semidefinitne optimizacije, SVT in TNNM se opreta na lastnosti singularnega razcepa matrik, LMaFit in ASD pa uporabita reševanje minimizacijskega problema po metodi najmanjših kvadratov, pri čemer prvi rešuje eksaktno, drugi pa s pomočjo verzije gradientnega spusta. V poglavju 3 smo podrobno predstavili matematično ozadje vsakega izmed algoritmov in dokazali nekatere pomembne trditve, na katerih temeljijo ideje teh algoritmov. Nato smo se v poglavju 4 osredotočili na testiranje in primerjavo delovanja algoritmov na primeru rekonstrukcije slik. Zanimala nas je kakovost rekonstruiranih slik, čas za rekonstrukcijo in vpliv ter težavnost izbire parametrov, ki so vhodni podatki v posamezni algoritmu. Analizirali smo več vidikov rekonstrukcije, pri čemer smo se osredotočali na vprašanja, ki so pri rekonstrukciji pomembna. Na koncu smo algoritme primerjali s precej bolj uveljavljenim algoritmom za rekonstrukcijo, ki temelji na reševanju Laplaceove diferencialne enačbe.

Glavni prispevki tega diplomskega dela so predstavitev matematičnega ozadja različnih algoritmov za reševanje problema matričnih napolnitev, implementacija teh algoritmov v programu Matlab in raziskava kakovosti ter

časovne zahtevnosti njihovega delovanja na problemu rekonstrukcije zašumljenih slik.

Področje matričnih napolnitev je zelo obsežno, zato bi v prihodnosti ugotovitve tega dela lahko razširili v veliko smeri. Delovanje algoritmov bi lahko preizkusili še na drugih področjih njihove uporabe, npr. rekonstrukcija zašumljenih signalov, [kompresija](#), priporočilni sistemi in problemi napolnitev matrik razdalje [19]. Ker so podatki v slikah lokalno podobni, česar problem matričnih napolnitev ne more upoštevati, pričakujemo, da bi algoritmi delovali še veliko bolje na tistih področjih uporabe, kjer lokalna podobnost podatkov ni pomembna (npr. priporočilni sistemi). Predstavljeni algoritmi TNNM, LMaFit in ASD imajo še alternativne verzije, ki posamezne korake algoritma izvedejo na alternativne načine. Lahko bi testirali delovanje teh alternativnih različic. Poleg tega obstajajo tudi številne izpeljanke teh algoritmov, pa tudi algoritmi, ki uporabljajo povsem drugačna matematična orodja. Vir [19] navaja še algoritme IRLS, ADMiRA in algoritom optimizacije na gladki Riemannovi mnogoterosti. Kot smo navedli v razdelku 4.2, bi lahko izpeljali tudi svoje različice algoritmov, kjer bi kakšen korak naredili na nov način. LMaFit bi zaradi velike odvisnosti od začetnega približka lahko razširili tako, da bi začeli z več naključno generiranimi začetnimi približki in po nekaj prvotnih korakih izmed njih izbrali tistega, ki bi najhitreje konvergiral. Zanimivo bi bilo klasificirati vhodne probleme glede na to, katerega od algoritmom bi bilo smiselno najprej uporabiti za napolnitev in kako nastaviti vhodne parametre. Seveda je nemogoče pričakovati, da bi bila takšna klasifikacija eksaktna, kar kažejo tudi naša testiranja, kljub vsemu pa bi lahko generirali neko lestvico priporočil, ki bi uporabniku omogočala smiselno zaporedje izbiranja algoritmov in parametrov v njih.

# Literatura

- [1] Mihály Bakonyi in Hugo J. Woerdeman. *Matrix Completions, Moments, and Sums of Hermitian Squares*. Princeton University Press, 2011. ISBN: 9780691128894.
- [2] Wayne W. Barrett, Charles R. Johnson in Michael Lundquist. “Determinantal formulae for matrix completions associated with chordal graphs”. V: *Linear Algebra and its Applications* 121 (1989), str. 265–289. ISSN: 0024-3795. DOI: [https://doi.org/10.1016/0024-3795\(89\)90706-4](https://doi.org/10.1016/0024-3795(89)90706-4). URL: <https://www.sciencedirect.com/science/article/pii/0024379589907064>.
- [3] Jonathan Bean. *Unsplash*. URL: <https://unsplash.com/photos/5ulmc8IHdLc> (pridobljeno 9. 2. 2023).
- [4] Stephen P Boyd in Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [5] Jian-Feng Cai, Emmanuel J. Candès in Zuowei Shen. *A Singular Value Thresholding Algorithm for Matrix Completion*. 2010. DOI: [10.1137/080738970](https://doi.org/10.1137/080738970). eprint: <https://doi.org/10.1137/080738970>. URL: <https://doi.org/10.1137/080738970>.
- [6] Emmanuel J. Candès in Benjamin Recht. “Exact Matrix Completion via Convex Optimization”. V: *Foundations of Computational Mathematics* 9.6 (dec. 2009), str. 717–772. ISSN: 1615-3383. DOI: [10.1007/s10208-009-9045-5](https://doi.org/10.1007/s10208-009-9045-5). URL: <https://doi.org/10.1007/s10208-009-9045-5>.

- [7] Nir Cohen in Jerome Dancis. “Inertias of Block Band Matrix Completions”. V: *SIAM Journal on Matrix Analysis and Applications* 19.3 (1998), str. 583–612. DOI: 10.1137/S0895479895296471. URL: <https://doi.org/10.1137/S0895479895296471>.
- [8] Wei Dai in Olgica Milenkovic. “SET: An algorithm for consistent matrix completion”. V: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, str. 3646–3649. DOI: 10.1109/ICASSP.2010.5495899.
- [9] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, jan. 1997. DOI: 10.1137/1.9781611971446.
- [10] Blaž Erzar. “Inkrementalna rekonstrukcija slike iz razpršenih podatkov”. Diplomsko delo. Univerza v Ljubljani, 2023. URL: <https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=144588>.
- [11] Alejandro Escamilla. *Unsplash*. URL: <https://unsplash.com/photos/cZhUxIQjILg> (pridobljeno 9. 2. 2023).
- [12] Maryam Fazel. “Matrix rank minimization with applications”. Doktorska disertacija. Stanford, CA: Stanford University, mar. 2002.
- [13] Gabe. *Unsplash*. URL: [https://unsplash.com/photos/bIZrEK-Z\\_cI](https://unsplash.com/photos/bIZrEK-Z_cI) (pridobljeno 9. 2. 2023).
- [14] Roger A. Horn in Charles R. Johnson. *Matrix Analysis*. 2nd. Cambridge; New York: Cambridge University Press, 2013. ISBN: 9780521839402.
- [15] Yao Hu in sod. “Fast and Accurate Matrix Completion via Truncated Nuclear Norm Regularization”. V: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.9 (2013), str. 2117–2130. DOI: 10.1109/TPAMI.2012.271.
- [16] Charles R Johnson. “Matrix completion problems: a survey”. V: *Matrix theory and applications*. Zv. 40. 1990, str. 171–198.

- [17] Kiryung Lee in Yoram Bresler. “ADMiRA: Atomic Decomposition for Minimum Rank Approximation”. V: *IEEE Transactions on Information Theory* 56.9 (2010), str. 4402–4416. DOI: [10.1109/TIT.2010.2054251](https://doi.org/10.1109/TIT.2010.2054251).
- [18] Bamdev Mishra in sod. “Fixed-rank matrix factorizations and Riemannian low-rank optimization”. V: *Computational Statistics* 29 (2014), str. 591–621.
- [19] Luong Trung Nguyen, Junhan Kim in Byonghyo Shim. “Low-Rank Matrix Completion: A Contemporary Survey”. V: *IEEE Access* 7 (2019), str. 94215–94237. DOI: [10.1109/ACCESS.2019.2928130](https://doi.org/10.1109/ACCESS.2019.2928130).
- [20] Ryan A. Rossi in Nesreen K. Ahmed. “The Network Data Repository with Interactive Graph Analytics and Visualization”. V: *AAAI*. 2015. URL: <https://networkrepository.com>.
- [21] Jos F. Sturm. “Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones”. V: *Optimization Methods and Software* 11.1-4 (1999), str. 625–653. DOI: [10.1080/10556789908805766](https://doi.org/10.1080/10556789908805766). URL: <https://doi.org/10.1080/10556789908805766>.
- [22] Jared Tanner in Ke Wei. “Low rank matrix completion by alternating steepest descent methods”. V: *Applied and Computational Harmonic Analysis* 40.2 (2016), str. 417–429. ISSN: 1063-5203. DOI: <https://doi.org/10.1016/j.acha.2015.08.003>. URL: <https://www.sciencedirect.com/science/article/pii/S1063520315001062>.
- [23] Zaiwen Wen, Wotao Yin in Yin Zhang. “Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm”. V: *Mathematical Programming Computation* 4 (dec. 2012). DOI: [10.1007/s12532-012-0044-1](https://doi.org/10.1007/s12532-012-0044-1).
- [24] Junfeng Yang in Xiaoming Yuan. “Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization”. V: *Mathematics of computation* 82.281 (2013), str. 301–329.