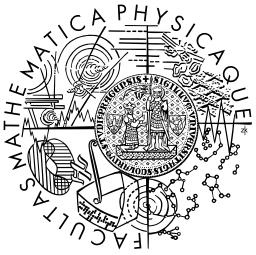


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

**DIPLOMOVÁ PRÁCE**

Markéta Machalová

# Analýza krylovovských regularizačních metod pro úlohy zaostřování obrazu

Katedra numerické matematiky

Vedoucí diplomové práce: doc. RNDr. Iveta Hnětynková, Ph.D.
Studijní program: Matematika pro informační technologie

Praha 2023

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....  .....

Podpis autora

Ráda bych na tomto místě poděkovala doc. RNDr. Ivetě Hnětynkové, Ph.D., za cenné rady, připomínky a především za ochotu, trpělivost a čas, který mi v průběhu psaní práce věnovala.

V neposlední řadě děkuji Bohu, rodině a přátelům za veškerou podporu během celého studia.

Název práce: Analýza krylovovských regularizačních metod pro úlohy zaostřování obrazu

Autor: Markéta Machalová

Katedra: Katedra numerické matematiky

Vedoucí diplomové práce: doc. RNDr. Iveta Hnětynková, Ph.D.

Abstrakt: Diplomová práce se zabývá konstrukcí a vlastnostmi úloh zaostřování obrazu spolu s přístupy k jejich řešení. Zaměřujeme se na krylovovské metody LSQR, GMRES a RRGMRES, jež jsou známy svými regularizačními vlastnostmi. Analyzujeme konvergenční chování metod, časovou efektivitu a kvalitu aproximovaného řešení. Dále představujeme blokové krylovovské metody, v oblasti zpracování obrazu nepříliš probádané. Tyto metody řeší soustavu lineárních rovnic s násobnou pravou stranou a vznikly zobecněním krylovovských metod, které slouží pro řešení lineárních rovnic s vektorovou pravou stranou. V závěru pak provádíme numerické experimenty zkoumající vliv různých faktorů na výsledky zaostřování obrazu a časovou náročnost jednotlivých metod a porovnáváme blokové a neblokové metody.

Klíčová slova: inverzní problém, šum, regularizace, Krylovův prostor

Title: Analysis of Krylov regularization methods for image deblurring problems

Author: Markéta Machalová

Department: Department of Numerical Mathematics

Supervisor: doc. RNDr. Iveta Hnětynková, Ph.D.

Abstract: The diploma thesis deals with the construction and properties of image deblurring problems along with approaches to their solution. We focus on Krylov subspace methods LSQR, GMRES and RRGMRES, which are known for their regularization properties. We analyze the convergence behavior of the methods, the time efficiency and the quality of the approximate solution. Next, we present block Krylov subspace methods, which are not well explored in the field of image processing. These methods solve a system of linear equations with a multiple right-hand side and were created by the generalizing Krylov subspace methods, which are used for solving linear equations with a vector right-hand side. Finally, we perform numerical experiments investigating the influence of various factors on the results of image deblurring and the time complexity of individual methods, and we compare block and non-block methods.

Keywords: inverse problem, noise, regularization, Krylov subspace

# Obsah

**Úvod** **2**

**1 Úlohy zaostřování obrazu** **4**
1.1 Numerická reprezentace obrazu . . . . . 4
1.2 Lineární model rozmazání obrazu . . . . . 5
1.2.1 Sestavení maticového modelu . . . . . 6
1.2.2 Obecný lineární model . . . . . 8
1.2.3 Naivní řešení . . . . . 10
1.2.4 Vlastnosti matice A . . . . . 11
1.2.5 Struktura modelu . . . . . 13
1.3 Algoritmus rychlého násobení matice s vektorem . . . . . 16

**2 Krylovovské regularizační metody** **18**
2.1 LSQR . . . . . 18
2.2 GMRES . . . . . 20
2.3 RRGMRES . . . . . 22
2.4 Konvergenční chování . . . . . 23

**3 Blokové krylovovské regularizační metody** **26**
3.1 Blokový LSQR . . . . . 28
3.2 Blokový GMRES a RRGMRES . . . . . 30
3.3 Konvergenční chování . . . . . 32

**4 Numerické experimenty** **33**
4.1 Vliv okrajových podmínek . . . . . 34
4.2 Časová náročnost . . . . . 36
4.3 Princip diskrepance . . . . . 37
4.4 Porovnání blokových a neblokových metod . . . . . 39
4.4.1 Obrázek ve stupních šedi s různým zašuměním . . . . . 39
4.4.2 Barevný obrázek . . . . . 41

**Závěr** **43**

**Seznam použité literatury** **44**

**Seznam algoritmů** **47**

# Úvod

V dnešní době se lidé setkávají se zpracováním obrazové informace na každém kroku – při platbě QR kódem, při přihlašování otiskem prstu nebo třeba při skenování státní poznávací značky na parkovištích. Pro zpracování obrazových dat je stěžejní mít k dispozici data v dostatečné kvalitě, především co se týče ostrosti obrazu. Jen tak je možné obraz správně segmentovat či v něm detekovat objekty. Zatímco v některých případech je postačující mít k dispozici kvalitní fotoaparát a správně nastavenou čočku, jindy není z technických či finančních důvodů možné dostatečně ostrý obraz získat. Lze však proces rozmazání obrazu modelovat a zpětně pak s pomocí numerických metod hledat co nejvíce ostrou podobu rozmazaného obrazu, viz například [1, 2].

Tuto úlohu zaostřování obrazu lze zařadit mezi *inverzní problémy*, které můžeme definovat jako problémy, jež se snaží rekonstruovat vstupní data, vlastnosti nebo parametry systému na základě pozorovaných dat [2]. Zatímco u běžných problémů známe vstupní data a hledáme odpovídající výstup, u inverzních problémů vstupy neznáme a snažíme se získat jejich aproximaci. Numericky lze úlohu chápat například jako soustavu lineárních algebraických rovnic

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

kde $\mathbf{A}$ označuje matici rozmazání, $\mathbf{b}$ reprezentuje zaznamenaný rozmazaný obraz a $\mathbf{x}$ odpovídá hledanému zaostřenému obrazu. Obecně v inverzních úlohách může být $\mathbf{A}$ obdélníková, v našem případě však bude vždy reálná čtvercová. Matice rozmazání má zhlazující vlastnost, tedy přenásobení vektoru maticí $\mathbf{A}$ způsobuje tlumení vysokých frekvencí.

Vlivem různých okolností je vektor $\mathbf{b}$ zatížen šumem. Zároveň je matice $\mathbf{A}$ typicky špatně podmíněná, tedy i malé chyby ve vstupních datech mohou přispět k velmi nepřesnému řešení. Ačkoliv matematicky může být $\mathbf{A}$ regulární, numericky je kvůli malým singulárním číslům blízko singulární matici. Pokud bychom řešili úlohu přímými metodami, například metodou nejmenších čtverců, došlo by k zesílení šumu. Proto se pro řešení diskrétních inverzních úloh využívají takzvané *regularizační metody*, které implicitně tlumí vliv šumu a malých singulárních čísel [1, 2, 3].

Regularizačních metod existuje celá řada a pro každý typ úloh se hodí jiná. V úlohách zaostřování obrazu je matice $\mathbf{A}$ hustá a velkých rozměrů. Na druhou stranu její specifická struktura umožňuje rychlé násobení. Nabízí se tedy využití metod, které vyžadují pouze násobení s maticí $\mathbf{A}$. Takovou skupinou jsou například *krylovovské regularizační metody*, ve kterých navíc lze díky zhlazující vlastnosti $\mathbf{A}$ generovat Krylovovy prostory s dominantními hladkými vektory. Pro základní principy krylovovských metod viz [4], pro jejich regularizační vlastnosti viz [2, 5, 6].

V první kapitole se seznámíme s modelem rozmazání obrazu. Ukážeme si, jak se model sestavuje, jaké má vlastnosti, jak velkou roli hraje přítomnost šumu, a proč k jeho řešení potřebujeme sofistikovanější přístup. Zároveň se podíváme na to, jak strukturu matice rozmazání obrazu ovlivňuje volba okrajové podmínky a jak lze využít těchto struktur při násobení matice s vektorem [1].

Druhá kapitola pojednává o krylovovských regularizačních metodách. Představíme si metody LSQR [7], GMRES [8] a RRGMRES [9]. Tyto metody patří mezi známé přístupy pro řešení diskrétních inverzních úloh a o jejich aplikaci pojednává řada publikací [2, 3, 5]. Ukážeme si jejich algoritmy, vlastnosti a konvergenční chování.

Ve třetí kapitole se přesuneme k blokovým krylovovským metodám, které byly původně navrženy v [10, 11] k řešení klasických soustav lineárních algebraických rovnic s násobnou pravou stranou. Před časem byla v [12] prezentována myšlenka, jak lze těchto metod využít při řešení úloh zaostřování barevného obrázku, kdy jedna pravá strana odpovídá jednomu barevnému kanálu. Proto se dále zaměříme na rozšíření blokových metod pro řešení inverzních úloh. Představíme si blokový LSQR, o jehož využití při regularizaci pojednává právě [12]. Jakožto rozšíření této idey odvodíme a popíšeme blokový GMRES a budeme diskutovat jeho úpravu na blokový RRGMRES.

Na závěr přistoupíme k numerickým experimentům s využitím prostředí MATLAB, vybraných toolboxů a vlastních implementací všech diskutovaných krylovovských metod. Podíváme se na konvergenční chování představených algoritmů v závislosti na hladině šumu a vybrané okrajové podmínce. Zhodnotíme jejich časovou náročnost a vzájemně porovnáme blokové a neblokové krylovovské metody.

# 1. Úlohy zaostřování obrazu

Abychom mohli regularizační metody pro úlohy zaostřování obrazu analyzovat, je třeba porozumět tomu, jak vůbec počítač s obrázky pracuje.

Zatímco teoretický koncept obrazu je spojitý, počítač dokáže pracovat jen s konečným počtem čísel. Nezbytným krokem je proto **digitalizace**. Tento pojem označuje převod spojitého analogového signálu na signál digitální. Tím získáme digitální obraz, který nyní můžeme numericky reprezentovat.

## 1.1 Numerická reprezentace obrazu

Digitální obraz se skládá z pixelů. Hodnota jednoho pixelu odpovídá jeho intenzitě. Uvážíme-li obrázek ve stupních šedi (viz obrázek 1.1), intenzita koresponduje s množstvím vyzářeného bílého světla. Nejnižší značí černou barvou, nejvyšší bílou barvou. V našem případě jeden pixel zabírá 8 bitů paměti, tedy intenzita nabývá hodnoty z množiny $\{0, \ldots, 255\}$.

Obraz ukládáme ve formě nezáporné matice, kterou si nyní definujeme.

**Definice 1** (Nezáporná matice a matice obrazu)**.** *Matice* $\mathbf{X} = (X_{i,j})_{m,n} \in \mathbb{R}^{m \times n}$ *se nazývá nezáporná, pokud pro všechna* $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ *platí:*

$$X_{i,j} \geq 0.$$

*Maticí obrazu nazveme nezápornou matici z* $\mathbb{R}^{n \times m}$*, ve které je jeden pixel obrazu reprezentován jedním prvkem matice.*

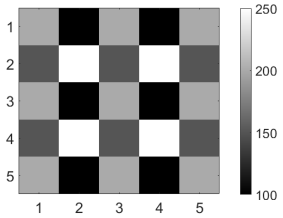
Při práci s RGB obrázkem ukládáme informaci do trojice matic (resp. trojrozměrné matice). Každému kanálu přísluší jedna matice, jeden pixel nyní zabere 3 bajty paměti (viz obrázek 1.2). Hodnota pixelu odpovídá intenzitě červeného, zeleného, resp. modrého světla. Mimo RGB existuje i řada dalších barevných formátů, reprezentace obrazu však bude analogická.

Na závěr této sekce ještě poznamenejme, že existuje více možných formátů pro reprezentaci jednoho pixelu. Velmi často se můžeme setkat s reprezentací v intervalu [0,1]. Pro ilustraci si ukážeme přeškálování prvků matice obrazu z množiny {0,1,...,255} do intervalu [0,1] [1].

$$\begin{pmatrix} 0 & 255 & 0 & 255 & 0 \\ 255 & 102 & 255 & 102 & 255 \\ 0 & 255 & 0 & 255 & 0 \\ 255 & 102 & 255 & 102 & 255 \\ 0 & 255 & 0 & 255 & 0 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0{,}4 & 1 & 0{,}4 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0{,}4 & 1 & 0{,}4 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

[1] Více informací naleznete v kapitole 4.

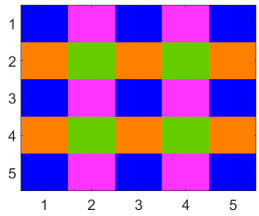
$$X = \begin{pmatrix} 200 & 100 & 200 & 100 & 200 \\ 150 & 250 & 150 & 250 & 150 \\ 200 & 100 & 200 & 100 & 200 \\ 150 & 250 & 150 & 250 & 150 \\ 200 & 100 & 200 & 100 & 200 \end{pmatrix}$$




Obrázek 1.1: Matice obrázku $5 \times 5$ pixelů ve stupních šedi.

$$X_R = \begin{pmatrix} 0 & 255 & 0 & 255 & 0 \\ 255 & 102 & 255 & 102 & 255 \\ 0 & 255 & 0 & 255 & 0 \\ 255 & 102 & 255 & 102 & 255 \\ 0 & 255 & 0 & 255 & 0 \end{pmatrix} \quad X_G = \begin{pmatrix} 0 & 51 & 0 & 51 & 0 \\ 128 & 204 & 128 & 204 & 128 \\ 0 & 51 & 0 & 51 & 0 \\ 128 & 204 & 128 & 204 & 128 \\ 0 & 51 & 0 & 51 & 0 \end{pmatrix}$$

$$X_B = \begin{pmatrix} 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$




Obrázek 1.2: Matice barevného obrázku $5 \times 5$ pixelů reprezentovaného v RGB.

## 1.2 Lineární model rozmazání obrazu

Ve chvíli, kdy víme, jak reprezentovat diskrétní obraz v numerické podobě, se můžeme přesunout k interpretaci úlohy zaostřování obrazu – nejprve ve stupních šedi, později RGB. Motivací nám bude následující příklad:

*Příklad* 1. Máme obrázek ve stupních šedi zachycený běžným fotoaparátem. Vlivem špatně zaostřené čočky je však rozmazaný a zašuměný. Úlohou je ke vstupnímu obrázku nalézt matici obrazu, která reprezentuje zaostřený obraz (viz obrázek 1.3). Obrázek má $594 \times 594$ pixelů a je zašuměný gaussovským bílým šumem s hladinou šumu $10^{-1}$. O šumu pojednává sekce 1.2.1.

Obrázek 1.3: Vlevo zaznamenaný obrázek, vpravo přesné řešení příkladu 1.

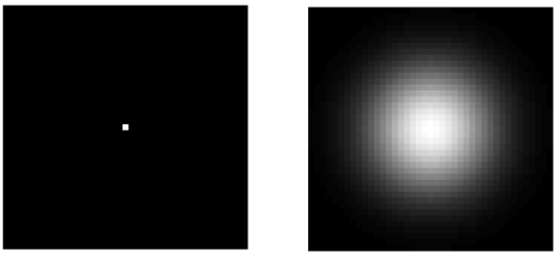
### 1.2.1 Sestavení maticového modelu

Pro nalezení aproximace neznámého ostrého obrazu nyní sestavme matematický model procesu rozmazání. Nechť $\mathbf{B} \in \mathbb{R}^{m \times n}$ označuje matici našeho rozmazaného obrázku a $\mathbf{X} \in \mathbb{R}^{m \times n}$ matici hledaného zaostřeného obrázku. Předpokládejme apriorní znalost modelu rozmazání, který je navíc *separabilní*. To znamená, že v modelu je rozmazání řádků a sloupců na sobě nezávislé. Potom existují matice $\mathbf{A}_c \in \mathbb{R}^{m \times m}$ a $\mathbf{A}_r \in \mathbb{R}^{n \times n}$, které určují vztah mezi ostrým a rozmazaným obrázkem [1]:

$$\mathbf{A}_c \mathbf{X} \mathbf{A}_r^T = \mathbf{B}. \tag{1.1}$$

#### Point spread funkce

Rozmazání obrázku lze modelovat **point spread funkcí (PSF)**, která popisuje, jak je bodový zdroj světla upraven a zkreslen optickým systémem. Jinými slovy: PSF popisuje, jak je světlo z jednoho pixelu v obrázku $\mathbf{X}$ rozprostřeno na určitý počet pixelů v rozmazaném obrázku $\mathbf{B}$ (viz obrázek 1.4).



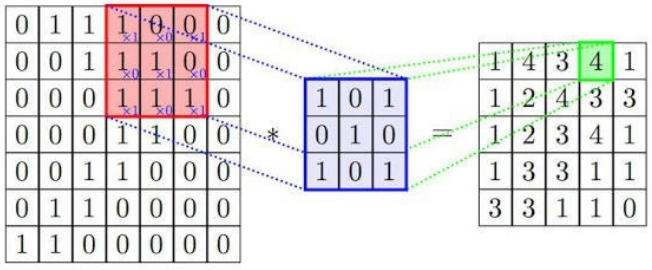
Obrázek 1.4: Point spread funkce. Vlevo bodový zdroj světla, vpravo jeho out-of-focus blur.

Matematicky lze tento proces popsat pomocí *diskrétní 2D konvoluce* [2]:

$$B_{i,j} = \sum_{k=1}^{(m-1)/2} \sum_{l=1}^{(n-1)/2} P_{i-k,j-l} X_{k,l}, \tag{1.2}$$

kde $B_{i,j}$, $\forall i, \in \{1, 2, \ldots, m\}$, $\forall j \in \{1, 2, \ldots, m\}$, označuje prvek matice obrazu $\mathbf{B}$ na pozici $i$,$j$. Pro vizualizaci viz obrázek 1.5. V praxi má point spread funkce menší rozměry než příslušný obraz, proto pro sestavení matice $\mathbf{P}$ je potřeba ji doplnit nulami tak, aby PSF byla vycentrovaná. Zároveň prostřední prvek odpovídá $P_{0,0}$. Bez újmy na obecnosti předpokládáme, že $\mathbf{B}$ a $\mathbf{P}$ jsou rozměru alespoň $3 \times 3$ a $m$, $n$ jsou liché [2].

[2] V opačném případě by matice PSF nemohla být vycentrována. Jde tedy o předpoklad pro zjednodušený zápis, prakticky může být velikost obrázku libovolná.



Obrázek 1.5: Vizualizace diskrétní konvoluce [13].

Známe-li způsob, jak k rozmazání obrazu došlo, je možné sestrojit PSF explicitně. V příkladu 1 došlo k rozmazání kvůli špatnému zaostření čočky. Tento jev je označován jako *out-of-focus blur* a lze jej modelovat předpisem

$$P_{i,j} = \begin{cases} \frac{1}{\pi r^2}, & i^2 + j^2 \leq r^2 \\ 0, & i^2 + j^2 > r^2, \end{cases}$$

kde $r$ označuje poloměr PSF. Dalším častým způsobem je rozmazání pohybem, případně atmosférickou turbulencí. O těchto a dalších způsobech se lze více dočíst v [1].

Jak již bylo zmíněno, naším předpokladem je, že rozmazání je separabilní, tedy existují vektory $\mathbf{c} \in \mathbb{R}^m$, $\mathbf{r} \in \mathbb{R}^n$ splňující

$$\mathbf{P} = \mathbf{c}\mathbf{r}^T = \begin{pmatrix} c_{-(m-1)/2} \\ \vdots \\ c_{(m-1)/2} \end{pmatrix} (r_{-(n-1)/2} \cdots r_{(n-1)/2}). \tag{1.3}$$

Využijeme-li tohoto rozkladu v rovnici (1.2), dostaneme

$$B_{i,j} = \sum_{k=1}^{(m-1)/2} \sum_{l=1}^{(n-1)/2} c_{i-k} r_{j-l} X_{k,l} = (c_{i-1}, \ldots, c_{i-(m-1)/2}) \cdot \mathbf{X} \cdot \begin{pmatrix} r_{j-1} \\ \vdots \\ r_{j-(n-1)/2} \end{pmatrix}.$$

Z tohoto vyjádření a rovnice (1.1) je již možné sestavit matice $\mathbf{A}_c$ a $\mathbf{A}_r$. Konkrétní podobu matic rozebereme v sekci 1.2.5.

### Šum

V dosavadní podobě se úloha zdá být poměrně jednoduchá, jenže zachycený obrázek je nejen rozmazaný, ale zároveň obsahuje i šum. Jeho zdrojem je především již zmíněná digitalizace, při které nelze naměřené hodnoty pixelů ukládat přesně a dochází k zaokrouhlování. Šum může však být i jiného původu v závislosti na úloze, kterou modelujeme. Budeme uvažovat pouze aditivní šum $\mathbf{E}$, v řeči matic jde o matici splňující rovnost:

$$\mathbf{B} = \mathbf{B}_{exact} + \mathbf{E}, \text{ kde} \tag{1.4}$$

$$\mathbf{B}_{exact} = \mathbf{A}_c \mathbf{X}_{exact} \mathbf{A}_r^T.$$

$\mathbf{B}_{exact}$ označuje matici obrazu bez šumu, kterou ale ve skutečnosti neznáme.

Zároveň se v našich experimentech omezíme na **gaussovský bílý šum**, což znamená, že všechny prvky matice $\mathbf{E} = (E_{ij})_{m,n}$ budou náležet stejnému Gaussovu normálnímu rozdělení s nulovou střední hodnotou a rozptylem $\eta^2$, tedy

$$E_{i,j} \sim \mathcal{N}(0, \eta^2) \qquad \forall\, i \in \{1, \ldots, m\},\, j \in \{1, \ldots, n\}.$$

Podstatné je, že množství šumu je nezávislé na konkrétním pixelu. Pro interpretaci množství šumu v obrázku se nejvíce využívají funkce *signal-to-noise ratio* a *hladina šumu*:

**Definice 2** (Signal-to-noise ratio)**.** *Mějme matici* $\mathbf{B} = (B_{i,j})_{m,n} \in \mathbb{R}^{m\times n}$ *označující matici obrazu a* $\mathbf{E} = (E_{i,j})_{m,n} \in \mathbb{R}^{m\times n}$ *označující matici šumu. Potom*

$$\mathrm{SNR}(\mathbf{B},\mathbf{E}) = \frac{\|\mathbf{B}\|_F^2}{\|\mathbf{E}\|_F^2} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}|B_{i,j}|^2}{\sum_{i=1}^{m}\sum_{j=1}^{n}|E_{i,j}|^2}$$

*nazveme signal-to-noise ratio (SNR).*

**Definice 3** (Hladina šumu)**.** *Mějme matici* $\mathbf{B} \in \mathbb{R}^{m\times n}$ *označující matici obrazu a* $\mathbf{E} \in \mathbb{R}^{m\times n}$ *označující matici šumu. Potom*

$$\mu(\mathbf{B},\mathbf{E}) = \frac{\|\mathbf{E}\|_F}{\|\mathbf{B}\|_F}$$

*nazveme hladinou šumu.*

Obě výše definované funkce jsou závislé pouze na Frobeniově normě matice obrazu a matice šumu. Je tedy postačující pracovat dále s jednou interpretací množství šumu, v našem případě s hladinou šumu.

### 1.2.2 Obecný lineární model

Nyní si představíme druhý způsob, jak modelovat úlohu rozmazání obrazu. Vychází ze soustavy rovnic

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.5}$$

kde $\mathbf{A} \in \mathbb{R}^{mn\times mn}$ je čtvercová matice rozmazání a $\mathbf{x} \in \mathbb{R}^{mn}$, $\mathbf{b} \in \mathbb{R}^{mn}$ jsou *vektorizace matic* $\mathbf{X}$ a $\mathbf{B}$ z předchozího modelu. Vektorizace matice je definována následovně:

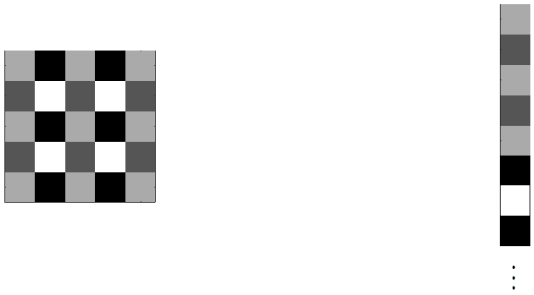
**Definice 4** (Vektorizace matice)**.** *Mějme matici*

$$\mathbf{X} = (\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_n) \in \mathbb{R}^{m\times n},$$

*kde* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^m$*. Potom*

$$\mathrm{vec}(\mathbf{X}) = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{mn}$$

*nazveme vektorizací matice* $\mathbf{X}$*.*



Obrázek 1.6: Maticová reprezentace versus vektorová reprezentace obrazu.

V praxi tedy vektorizace matice obrazu vznikne „rozřezáním“ obrázku po sloupcích $m \times 1$ pixel a následným „slepením“ (viz obrázek 1.6). Matici $\mathbf{A}$ získáme z modelu (1.1) skrze Kroneckerův součin matic, který se řídí definicí:

**Definice 5** (Kroneckerův součin matic)**.** *Mějme matice* $\mathbf{A}_r = (A^r_{i,j})_{n,n} \in \mathbb{R}^{n\times n}$ *a* $\mathbf{A}_c \in \mathbb{R}^{m\times m}$*. Potom Kroneckerův součin matic* $\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c$ *odpovídá [14, kapitola 2.1]*

$$\mathbf{A}_r \otimes \mathbf{A}_c = \begin{pmatrix} A^r_{1,1}\mathbf{A}_c & A^r_{1,2}\mathbf{A}_c & \cdots & A^r_{1,n}\mathbf{A}_c \\ A^r_{2,1}\mathbf{A}_c & A^r_{2,2}\mathbf{A}_c & \cdots & A^r_{2,n}\mathbf{A}_c \\ \vdots & \vdots & \ddots & \vdots \\ A^r_{n,1}\mathbf{A}_c & A^r_{n,2}\mathbf{A}_c & \cdots & A^r_{n,n}\mathbf{A}_c \end{pmatrix} \in \mathbb{R}^{nm\times nm}.$$

Využitím těchto poznatků převedeme model (1.1) na model (1.5):

$$(\mathbf{A}_r \otimes \mathbf{A}_c) \cdot \mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{B}). \tag{1.6}$$

Stejně jako u předchozího modelu nesmíme opomenout přítomný šum v obrázku, proto

$$\mathbf{b} = \mathbf{b}_{exact} + \mathbf{e}, \tag{1.7}$$

kde

$$\begin{aligned} \mathbf{b}_{exact} &= \mathrm{vec}(\mathbf{B}_{exact}) \in \mathbb{R}^{mn}, \\ \mathbf{e} &= \mathrm{vec}(\mathbf{E}) \in \mathbb{R}^{mn} \end{aligned}$$

a $\mathbf{B}_{exact}$, $\mathbf{E}$ značí matice z rovnice (1.4). Obecnost tohoto modelu tkví v širším využití – mimo modelování úlohy zaostřování obrazu lze tento model využít například při určování gravimetrického profilu či při rekonstrukci signálu z počítačové tomografie [2].

### Rozmazání barevného obrazu

Oba výše zmíněné modely lze poupravit pro zaostřování barevného obrazu reprezentovaného RGB barevným modelem. Uvážíme-li obecný lineární model (1.5), příslušná úloha zaostřování barevného obrazu odpovídá

$$\mathbf{AX} = \mathbf{B}, \tag{1.8}$$

kde

$$\begin{aligned}\mathbf{A} &\in \mathbb{R}^{mn\times mn},\\ \mathbf{X} &= \begin{pmatrix}\mathbf{x}_R & \mathbf{x}_G & \mathbf{x}_B\end{pmatrix} \in \mathbb{R}^{mn\times 3},\\ \mathbf{B} &= \begin{pmatrix}\mathbf{b}_R & \mathbf{b}_G & \mathbf{b}_B\end{pmatrix} \in \mathbb{R}^{mn\times 3},\end{aligned}$$

přičemž $\mathbf{b}_R, \mathbf{b}_G, \mathbf{b}_B$ a $\mathbf{x}_R, \mathbf{x}_G, \mathbf{x}_B$ označují odpovídající vektorizace matic obrazu, které udávají informaci o intenzitě červeného, zeleného a modrého světla. Je vhodné zmínit, že tento model předpokládá identické rozmazání všech tří barevných kanálů. V opačném případě bychom museli definovat mnohem složitější model. Tento jev však není až tak častý, proto zůstaneme u tohoto modelu.

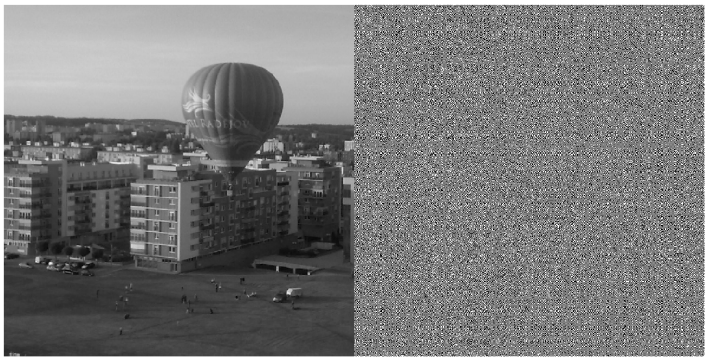
Obrázek 1.7: Příklad 1 s barevným obrázkem.

### 1.2.3 Naivní řešení

Předpokládejme, že matice $\mathbf{A}$ je regulární[3]. Potom soustava $\mathbf{Ax} = \mathbf{b}$ má *naivní řešení*

$$\mathbf{x}_{naive} = \mathbf{A}^{-1}\mathbf{b}, \tag{1.9}$$

jež by mohlo působit jako vektorizace matice hledaného zaostřeného obrázku. Jak ale ukazuje obrázek 1.8, toto naivní řešení má k reálnému původnímu obrázku daleko. Proč tomu tak je, nám ukáže analýza vlastností matice rozmazání.



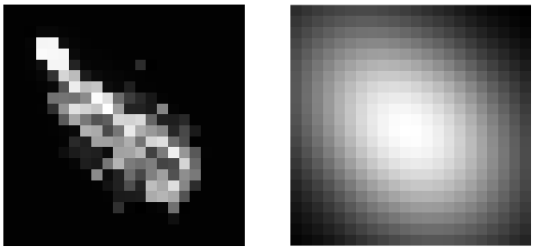
Obrázek 1.8: Naivní řešení. Vlevo přesné řešení příkladu 1, vpravo odpovídající naivní řešení.

[3]V praxi je tento předpoklad často splněn.

### 1.2.4 Vlastnosti matice A

Nejprve zaveďme pro tuto sekci příklad 2, jenž pracuje s řádově menším obrázkem než předchozí příklad 1:

*Příklad* 2. Máme obrázek $22 \times 22$ pixelů vygenerovaný funkcí `PRBlurGauss(22)` z toolboxu `IRTools` v Matlabu (viz [15]). K němu přidáme aditivní Gaussův šum s $\mu = 10^{-4}$ s out-of-focus blur s $r = 17$ (viz obrázek 1.9).



Obrázek 1.9: Vlevo původní obrázek vygenerovaný `IRTools`, vpravo rozmazaný a zašuměný obrázek.

Pro samotnou analýzu matice $\mathbf{A}$ budeme potřebovat větu o singulárním rozkladu:

**Věta 1** (Singulární rozklad)**.** *Nechť* $\mathbf{A} \in \mathbb{R}^{\mathtt{m}\times\mathtt{n}}$ *je matice hodnosti* $r$*. Potom existují ortogonální matice* $\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_\mathtt{m}) \in \mathbb{R}^{\mathtt{m}\times\mathtt{m}}$ *a* $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_\mathtt{n}) \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$*, které splňují*

$$\mathbf{A} = \mathbf{U}\begin{pmatrix}\mathbf{\Sigma}_r & 0\\ 0 & 0\end{pmatrix}\mathbf{V}^T = \sum_{i=1}^{r}\sigma_i\mathbf{u}_i\mathbf{v}_i^T,$$

*kde* $\mathbf{\Sigma}_r \in \mathbb{R}^{r\times r}$ *je diagonální matice s* $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$ *na diagonále. Vektory* $\mathbf{u}_1, \mathbf{u}_2, \ldots$ *se nazývají levé singulární vektory,* $\mathbf{v}_1, \mathbf{v}_2, \ldots$ *pravé singulární vektory a skaláry* $\sigma_1, \sigma_2, \ldots$ *singulární čísla [16, kapitola 2.5].*

S využitím tohoto rozkladu můžeme nyní vyjádřit naivní řešení (1.9):

$$\begin{aligned}\mathbf{x}_{\text{naive}} &= \mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{b} = \sum_{i=1}^{mn}\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\cdot\mathbf{v}_i\\ \mathbf{x}_{\text{naive}} &= \sum_{i=1}^{mn}\frac{\mathbf{u}_i^T(\mathbf{b}_{exact}+\mathbf{e})}{\sigma_i}\mathbf{v}_i = \sum_{i=1}^{mn}\frac{\mathbf{u}_i^T\mathbf{b}_{\text{exact}}}{\sigma_i}\cdot\mathbf{v}_i + \sum_{i=1}^{mn}\frac{\mathbf{u}_i^T\mathbf{e}}{\sigma_i}\cdot\mathbf{v}_i.\end{aligned} \tag{1.10}$$

Ve vyjádření (1.10) dostáváme dvě sumy. První z nich koresponduje s přesným řešením $\mathbf{x}_{exact}$, druhá s $\mathbf{A}^{-1}\mathbf{e}$ označovaným jako *inverzní šum*. Je tedy na místě prozkoumat podíl tohoto inverzního šumu v jednotlivých komponentách naivního řešení v závislosti na rostoucím indexu $i$. Zaveďme nejprve definice, které poslouží pro potřeby analýzy.

**Definice 6** (Diskrétní Picardova podmínka)**.** *Nechť* $\tau$ *označuje úroveň strojové přesnosti. Potom řekneme, že diskrétní Picardova podmínka (DPC) je splněna, pokud pro každou singulární hodnotu větší než* $\tau$ *platí, že příslušné koeficienty* $\left|u_i^T b_{exact}\right|$ *klesají k nule (v průměru) rychleji než* $\sigma_i$ *[2].*

**Definice 7** (Kovarianční matice **K**). *Nechť* $\mathbf{s} \in \mathbb{R}^{\mathrm{n}} = (s_1, s_2, \ldots, s_{\mathrm{n}})^T$ *je náhodný vektor a* $s_1, s_2, \ldots s_{\mathrm{n}}$ *jsou náhodné veličiny s konečnou střední hodnotou a rozptylem. Potom kovarianční matice* $\mathbf{K}(\mathbf{s})$ *je čtvercová matice, která má na pozici* $(i, j)$ *kovarianci* $s_i$ *a* $s_j$*,*

$$K(\mathbf{s})_{i,j} = \mathrm{cov}(s_i, s_j) = \mathbf{E}[(s_i - \mathbf{E}(s_i))(s_j - \mathbf{E}(s_j))],$$

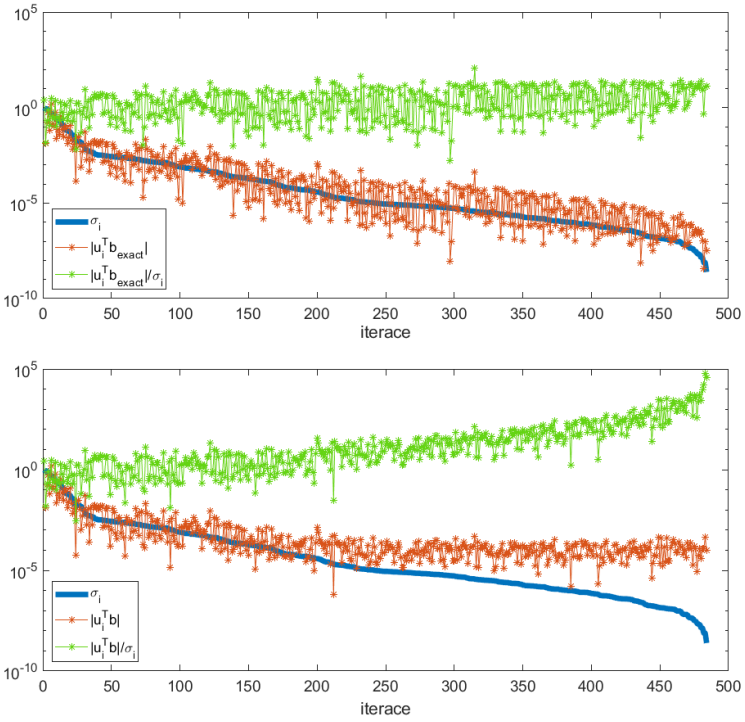
*kde* $\mathbf{E}$ *značí střední hodnotu [17, kapitola 2.3].*

Matice $\mathbf{A}$ má zhlazující efekt, neboť rozmazáním se v obrázku redukují vysoké frekvence, což se projevuje eliminací ostrých hran a detailů. Zhlazujícím efektem máme na mysli vlastnost, kdy přenásobení maticí $\mathbf{A}$ tlumí ve vektoru vysoké frekvence a vektor $\mathbf{Ax}$ je tak výrazně hladší než vektor $\mathbf{x}$. To má vliv i na hladkost vektoru $\mathbf{b}_{exact}$, který tak splňuje *diskrétní Picardovu podmínku* [3].

Hodnota $\left|\mathbf{u}_i^T \mathbf{b}_{\mathrm{exact}}\right|$ tedy s rostoucím $i$ klesá v průměru k nule. Naproti tomu hodnota $\left|\mathbf{u}_i^T \mathbf{e}\right|$ je na indexu nezávislá a pohybuje se kolem hodnoty $\eta$ (směrodatná odchylka $\mathbf{e}$), což je možné ukázat na kovarianční matici vektoru $\mathbf{U}^T\mathbf{e}$ [2]:

$$\mathbf{K}(\mathbf{U}^T\mathbf{e}) = \mathbf{U}^T\mathbf{K}(\mathbf{e})\mathbf{U} = \mathbf{U}^T\mathbf{E}(\mathbf{e}^T\mathbf{e})\mathbf{U} = \eta^2\mathbf{U}^T\mathbf{U} = \eta^2\mathbf{I},$$

kde $\eta^2$ značí rozptyl složek vektoru $\mathbf{e}$. Chování vektoru $\mathbf{U}^T\mathbf{e}$ tedy odpovídá gaussovskému bílému šumu. Vzhledem k tomu, že $\sigma_i$ klesají k nule, koeficient $\frac{\left|\mathbf{u}_i^T\mathbf{e}\right|}{\sigma_i}$ roste a se zvyšujícím se indexem se postupně stává dominantní složkou. V celkovém součtu proto hodnota koeficientu $\zeta_i = \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}$ od jistého indexu diverguje (viz obrázek 1.10 reprezentující takzvaný *Picardův graf* používaný standardně k analýze inverzní úlohy).

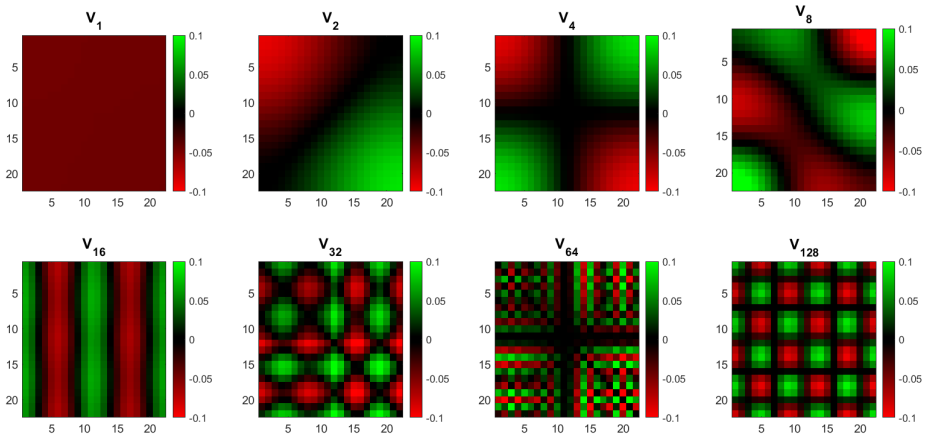



Obrázek 1.10: Picardův graf. Horní graf odpovídá obrázku z příkladu 2 bez přidaného šumu, druhý graf pak obrázku se šumem.

Nyní ještě proveďme analýzu naivního řešení maticového modelu pro lepší porozumění chování jeho komponent. Analogicky k (1.10) můžeme vyjádřit $\mathbf{X}_{naive}$ následovně:

$$\mathbf{X}_{\text{naive}} = \sum_{i=1}^{mn} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{V}_i, \tag{1.11}$$

kde $\mathbf{V}_i \in \mathbb{R}^{m \times n}$ označují *singular images*, tedy matice, jež vznikly z pravých singulárních vektorů „přerovnáním“ z rozměru $mn \times 1$ do $m \times n$ (neboli platí $\text{vec}(\mathbf{V}_i) = \mathbf{v}_i$). Každý pravý singulární vektor nám dává informaci o určitě frekvenční oblasti. Vektory odpovídající vyšším singulárním číslům odpovídají nižším frekvencím a obdobně nižší singulární čísla dávají informaci o vyšší frekvenci. To vyplývá z toho, že singulární vektory se chovají jako Fourierova báze [1] (viz obrázek 1.11).




Obrázek 1.11: Singular images $\mathbf{V}_i$ z příkladu 2. Zeleně jsou znázorněny kladné prvky vektoru, červeně záporné prvky. Se zvyšujícím se indexem $i$ lze pozorovat častější střídání znaménka.

Koeficient $\zeta_i$ v rozvoji $\sum_{i=1}^{mn} \zeta_i \mathbf{V}_i$ udává množství obrazové informace z patřičné frekvence. Jelikož od jistého $i$ tento koeficient diverguje, v celkovém součtu budou převládat vysokofrekvenční singular images. To vysvětluje podobu naivního řešení na obrázku 1.8.

Naivní řešení je v úlohách zaostřování obrazu nepoužitelné. Proto se ve druhé kapitole zaměříme na možnosti, jak úlohy řešit lépe. Ještě předtím se podíváme na konkrétní strukturu modelu.

### 1.2.5 Struktura modelu

V sekci 1.2.1 jsme ukázali, že proces rozmazání lze často modelovat diskrétní 2D konvolucí, viz rovnice (1.2). Tuto rovnici lze přeformulovat tak, že výsledná intenzita pixelu reprezentovaného $B_{ij}$ odpovídá váženému součtu intenzit okolních pixelů. Problém tkví v tom, že není jednoznačné, jak počítat konvoluci v krajních bodech obrázků, kde máme omezený počet sousedních pixelů.

Existují však tři základní **okrajové podmínky**, které nám dávají řešení tohoto problému. Nejjednodušší je *nulová* okrajová podmínka, při které předpokládáme, že vnější okraj obrazu je černý, pixely mají nulovou intenzitu. Pokud

jsou však okraje našeho obrázku světlé, je třeba zvážit využití *periodické* nebo *reflexivní* podmínky. Podobu okrajových podmínek shrnuje obrázek 1.12.

Přesuňme se nyní ke struktuře matice $\mathbf{A}$. Abychom pochopili, jak bude matice vypadat při různých okrajových podmínkách, omezíme se z ilustrativních důvodů na nejmenší možný model (pro větší modely je konstrukce analogická). Mějme model (1.5) s $\mathbf{X}$, $\mathbf{B} \in \mathbb{R}^{3\times 3}$ a point spread funkci $\mathbf{P} = (P_{i,j})_{3,3}$. Potom při užití **nulové** okrajové podmínky dostáváme následující vyjádření matice rozmazání (prázdné pozice odpovídají nulám):

$$\mathbf{A} = \left(\begin{array}{ccc|ccc|ccc} P_{0,0} & P_{0,1} & & P_{1,0} & P_{1,1} & & & & \\ P_{0,-1} & P_{0,0} & P_{0,1} & P_{1,-1} & P_{1,0} & P_{1,1} & & & \\ & P_{0,-1} & P_{0,0} & & P_{1,-1} & P_{1,0} & & & \\ \hline P_{-1,0} & P_{-1,1} & & P_{0,0} & P_{0,1} & & P_{1,0} & P_{1,1} & \\ P_{-1,-1} & P_{-1,0} & P_{-1,1} & P_{0,-1} & P_{0,0} & P_{0,1} & P_{1,-1} & P_{1,0} & P_{1,1} \\ & P_{-1,-1} & P_{-1,0} & & P_{0,-1} & P_{0,0} & & P_{1,-1} & P_{1,0} \\ \hline & & & P_{-1,0} & P_{-1,1} & & P_{0,0} & P_{0,1} & \\ & & & P_{-1,-1} & P_{-1,0} & P_{-1,1} & P_{0,-1} & P_{0,0} & P_{0,1} \\ & & & & P_{-1,-1} & P_{-1,0} & & P_{0,-1} & P_{0,0} \end{array}\right). \tag{1.12}$$

Tato matice je *toeplitzovská* bloková, přičemž jednotlivé bloky jsou také toeplitzovské matice. Tato struktura je paměťově velmi výhodná, neboť každou toeplitzovskou matici je možné plně popsat jejím prvním sloupcem a řádkem.

Pro **periodickou** podmínku dostáváme $\mathbf{A}$ v podobě

$$\left(\begin{array}{ccc|ccc|ccc} P_{0,0} & P_{0,1} & P_{0,-1} & P_{1,0} & P_{1,1} & P_{1,-1} & P_{-1,0} & P_{-1,1} & P_{-1,-1} \\ P_{0,-1} & P_{0,0} & P_{0,1} & P_{1,-1} & P_{1,0} & P_{1,1} & P_{-1,-1} & P_{-1,0} & P_{-1,1} \\ P_{0,1} & P_{0,-1} & P_{0,0} & P_{1,1} & P_{1,-1} & P_{1,0} & P_{-1,1} & P_{-1,-1} & P_{-1,0} \\ \hline P_{-1,0} & P_{-1,1} & P_{-1,-1} & P_{0,0} & P_{0,1} & P_{0,-1} & P_{1,0} & P_{1,1} & P_{1,-1} \\ P_{-1,-1} & P_{-1,0} & P_{-1,1} & P_{0,-1} & P_{0,0} & P_{0,1} & P_{1,-1} & P_{1,0} & P_{1,1} \\ P_{-1,1} & P_{-1,-1} & P_{-1,0} & P_{0,1} & P_{0,-1} & P_{0,0} & P_{1,1} & P_{1,-1} & P_{1,0} \\ \hline P_{1,0} & P_{1,1} & P_{1,-1} & P_{-1,0} & P_{-1,1} & P_{-1,-1} & P_{0,0} & P_{0,1} & P_{0,-1} \\ P_{1,-1} & P_{1,0} & P_{1,1} & P_{-1,-1} & P_{-1,0} & P_{-1,1} & P_{0,-1} & P_{0,0} & P_{0,1} \\ P_{1,1} & P_{1,-1} & P_{1,0} & P_{-1,1} & P_{-1,-1} & P_{-1,0} & P_{0,1} & P_{0,-1} & P_{0,0} \end{array}\right), \tag{1.13}$$

tedy $\mathbf{A}$ je *cirkulantní* bloková matice s *cirkulantními* bloky. K uložení cirkulantní matice postačí dokonce jeden vektor reprezentující její první sloupec.

Pro model (1.1) získáme matice $\mathbf{A}_c$ a $\mathbf{A}_r$ využitím rovnice (1.3) a definice Kroneckerova součinu. Pro **nulovou** okrajovou podmínku získáme matice

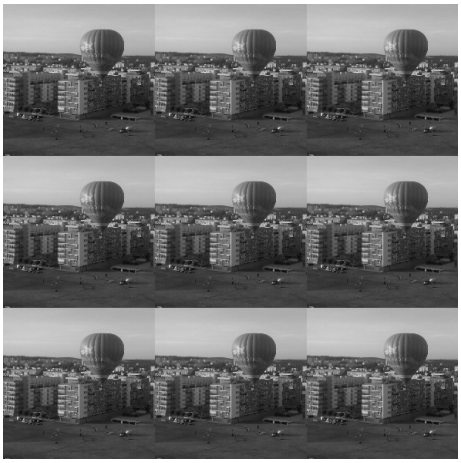
$$\mathbf{A}_r = \begin{pmatrix} r_0 & r_1 & \\ r_{-1} & r_0 & r_1 \\ & r_{-1} & r_0 \end{pmatrix} \mathbf{A}_c = \begin{pmatrix} c_0 & c_1 & \\ c_{-1} & c_0 & c_1 \\ & c_{-1} & c_0 \end{pmatrix}$$

a obdobně pro **periodickou** okrajovou podmínku

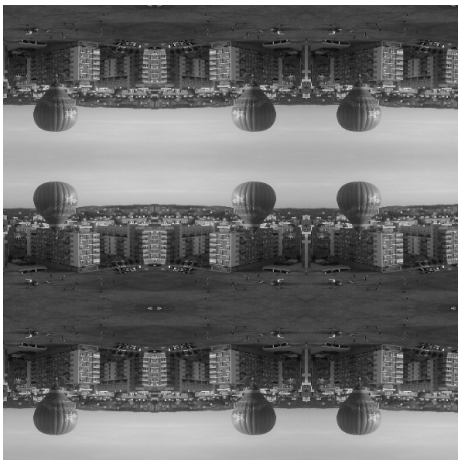
$$\mathbf{A}_r = \begin{pmatrix} r_0 & r_1 & r_{-1} \\ r_{-1} & r_0 & r_1 \\ r_1 & r_{-1} & r_0 \end{pmatrix} \mathbf{A}_c = \begin{pmatrix} c_0 & c_1 & c_{-1} \\ c_{-1} & c_0 & c_1 \\ c_1 & c_{-1} & c_0 \end{pmatrix}.$$

Reflexivní okrajová podmínka vytváří matici s poněkud náročnější strukturou (viz [1]), ale vlastnosti matice $\mathbf{A}$ (respektive $\mathbf{A}_c$, $\mathbf{A}_r$) umožňují u všech okrajových podmínek rychlé násobení s vektorem a paměťovou efektivitu.

$$\mathbf{X}_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbf{X} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$



$$\mathbf{X}_P = \begin{pmatrix} \mathbf{X} & \mathbf{X} & \mathbf{X} \\ \mathbf{X} & \mathbf{X} & \mathbf{X} \\ \mathbf{X} & \mathbf{X} & \mathbf{X} \end{pmatrix}$$



$$\mathbf{X}_R = \begin{pmatrix} \mathbf{X}_{hv} & \mathbf{X}_h & \mathbf{X}_{hv} \\ \mathbf{X}_v & \mathbf{X} & \mathbf{X}_v \\ \mathbf{X}_{hv} & \mathbf{X}_h & \mathbf{X}_{hv} \end{pmatrix}$$

Obrázek 1.12: Okrajové podmínky. Shora: nulová, periodická a reflexivní okrajová podmínka. $\mathbf{X}$ značí matici původního obrázku, $\mathbf{X}_h$ matici horizontálně převráceného, $\mathbf{X}_v$ vertikálně převráceného a $\mathbf{X}_{hv}$ horizontálně i vertikálně převráceného obrázku.

## 1.3 Algoritmus rychlého násobení matice s vektorem

Klasické algoritmy pro násobení matice rozměru $\mathfrak{n} \times \mathfrak{n}$ s vektorem délky $\mathfrak{n}$ vyžadují $\mathcal{O}(\mathfrak{n}^2)$ operací. Protože jsou matice námi uvažovaných modelů blokově toeplitzovské (nebo dokonce blokově cirkulantní) se strukturovanými bloky, při vhodně zvoleném algoritmu lze redukovat složitost na $\mathcal{O}(\mathfrak{n} \log \mathfrak{n})$. Příkladem takového algoritmu je násobení pomocí **rychlé Fourierovy transformace** (Fast Fourier Transform, dále FFT), které nyní popíšeme. Začněme klíčovou definicí diskrétní Fourierovy transformace.

**Definice 8** (Diskrétní Fourierova transformace)**.** *Nechť $\mathbb{T}$ je těleso a $\omega \in \mathbb{T}$. Diskrétní Fourierova transformace (DFT) s parametrem $\omega$ je definována jako zobrazení mezi vektorovými prostory $\mathbb{T}^{\mathfrak{n}} \to \mathbb{T}^{\mathfrak{n}}$, které vektoru koeficientů přiřadí vektor hodnot v bodech $1, \omega, \omega^2, \ldots, \omega^{\mathfrak{n}-1}$ [18, kapitola 8.1].*

Pro naši aplikaci budeme dále uvažovat $\mathbb{T} = \mathbb{R}$ a předpokládat $\mathfrak{n} = 2^l$ pro $l \in \mathbb{N}$. Dále si představme algoritmus FFT a rychlé násobení polynomů, jež tvoří základní stavební kámen finálního algoritmu násobení matice s vektorem. Jejich využitím dochází k redukci počtu jednotkových operací.

**Algoritmus 1:** FFT($\omega$) [18, kapitola 8.2]

**Input:** $(a_0, a_1, \ldots, a_{\mathfrak{n}-1})$
**Output:** $DFT(\omega)(a_0, a_1, \ldots, a_{\mathfrak{n}-1}) = (d_0, \ldots, d_{\mathfrak{n}-1})$
**if** *n=1* **then**
  return $a_0$
**end**
$(b_0, \ldots, b_{\mathfrak{n}/2-1})$ =FFT$(\omega^2)(a_0, a_2, \ldots, a_{\mathfrak{n}-2})$;
$(c_0, \ldots, c_{\mathfrak{n}/2-1})$ =FFT$(\omega^2)(a_1, a_3, \ldots, a_{\mathfrak{n}-1})$;
**for** $i = 0, \ldots \frac{\mathfrak{n}}{2} - 1$ **do**
  $d_i = b_i + \omega^i c_i$;
  $d_{i+\mathfrak{n}/2} = b_i - \omega^i c_i$;
**end**

**Algoritmus 2:** Rychlé násobení polynomů [18, kapitola 9]

**Input:** $p = \sum_{i=0}^{\mathfrak{n}} a_i x^i$, $q = \sum_{i=0}^{\mathfrak{m}} b_i x^i \in \mathbb{R}[x]$
**Output:** $p \cdot q = \sum_{i=0}^{\mathfrak{m}+\mathfrak{n}} f_i x^i$
Zvol $N = 2^k > \mathfrak{m} + \mathfrak{n}$ a $\omega = \pm 1$;
$\mathbf{c}$ =FFT$(\omega)(a_0, a_1, \ldots, a_{\mathfrak{n}}, 0, \ldots, 0)$;
$\mathbf{d}$ =FFT$(\omega)(b_0, b_1, \ldots, b_{\mathfrak{m}}, 0, \ldots, 0)$;
$\mathbf{e} = \mathbf{c} \cdot \mathbf{d} = (c_0 d_0, \ldots, c_{N-1} d_{N-1})$;
$\mathbf{f} = \frac{1}{N}$FFT$(\omega^{-1})(e_0, \ldots, e_{N-1})$;

FFT je rekurzivní algoritmus, jenž převádí vektor reálných čísel do DFT reprezentace a jehož základním stavebním kamenem je vyhodnocení polynomu stupně $s \leq \mathfrak{n}$ v bodě $\alpha$ skrze vyhodnocení polynomů stupňů $s/2$ v bodě $\alpha^2$. Rychlé násobení polynomů převádí polynomy do DFT reprezentace, vynásobí je mezi sebou

a výsledný součin převede inverzní diskrétní Fourierovou transformací zpět do původní reprezentace.

Nyní můžeme přistoupit k finálnímu algoritmu rychlého násobení matice s vektorem. Uvažujme toeplitzovskou blokovou matici **A** z (1.12) a vektor

$$\mathbf{v} = \begin{pmatrix} v_{11} & v_{12} & v_{13} & v_{21} & \cdots v_{33} \end{pmatrix}^T.$$

Potom rychlé násobení matice s vektorem pomocí FFT vypadá následovně:

**Algoritmus 3:** Rychlé násobení Toeplitzovské blokové matice s vektorem [19]

**Input:** $\mathbf{A}, \mathbf{v}$
**Output:** $\mathbf{Av} = \mathbf{D}$

$k = (\mathfrak{n} - 1)/2$;
**for** $i,j = 1,2,\ldots,\mathfrak{n}$ **do**
  $\lambda_{i-k-1,j-k-1} = (\mathfrak{n} + i - k - 2)(2\mathfrak{n} - 1) + (\mathfrak{n} + j - k - 2)$;
  $\mu_{i,j} = 2\mathfrak{n}(\mathfrak{n} - 1) - (i - 1)(2\mathfrak{n} - 1) - (j - 1)$;
**end**
$f(t) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} P_{ij} t^{\lambda_{i,j}}$;
$g(t) = \sum_{i=1}^{\mathfrak{n}} \sum_{j=1}^{\mathfrak{n}} v_{ij} t^{\mu_{i,j}}$;
$h(t)$ =Rychlé násobení polynomů$(f(t), g(t))$;
**for** $i,j = 1,2,\ldots,\mathfrak{n}$ **do**
  $\xi_{i,j} = 2\mathfrak{n}(2\mathfrak{n} - 1) - i(2\mathfrak{n} - 1) - j$;
  $d_{i,j} = \mathrm{coef}(h(t^{\xi_{i,j}}))$
**end**
$\mathbf{D} = (\mathbf{d}_{\mathrm{i,j}})_{\mathrm{n,n}}$

Podstatou metody je převedení matice a vektoru na polynomy $f(t)$ a $g(t)$ a využití rychlého násobení polynomů. Výsledný vektor $h(t)$, respektive koeficienty členů stupňů $\xi_{i,j}$, pak odpovídá hledanému součinu **Av**. Více informací naleznete v [19], o algoritmu rychlého násobení polynomů a FFT pak v [18, kapitola 8–9] nebo v [20, kapitola 8].

# 2. Krylovovské regularizační metody

V této kapitole si představíme iterační metody pro řešení problému (1.5), jež přibližné řešení hledají v **Krylovově prostoru** a zároveň mají regularizační vlastnosti. Nejprve přistupme k samotné definici Krylovova prostoru, k definici stupně vektoru vzhledem k matici a k lemmatu o dimenzi Krylovova prostoru.

**Definice 9** (Krylovův prostor)**.** *Nechť* $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}$*,* $\mathbf{r}_0 \in \mathbb{R}^{\mathfrak{n}}$ *a* $\mathfrak{n} \geq \mathfrak{m} \geq 1$*. Potom*

$$\mathcal{K}_{\mathfrak{m}}(\mathbf{A},\mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{\mathfrak{m}-1}\mathbf{r}_0\}$$

*nazveme* $\mathfrak{m}$*-tým Krylovovým prostorem matice* $\mathbf{A}$ *a vektoru* $\mathbf{r}_0$ *[4].*

**Definice 10** (Stupeň vektoru $\mathbf{r}_0$ vzhledem k matici $\mathbf{A}$)**.** *Mějme matici* $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}$ *a* $\mathbf{r}_0 \in \mathbb{R}^{\mathfrak{n}}$*. Potom*

$$d(\mathbf{A}, \mathbf{r}_0) = \min\{k \in \mathbb{N} \mid dim(\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)) = dim(\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0))\}$$

*nazveme stupněm vektoru* $\mathbf{r}_0$ *vzhledem k matici* $\mathbf{A}$*.*

**Lemma 1** (Dimenze Krylovova prostoru)**.** *Nechť* $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}$*,* $\mathbf{r}_0 \in \mathbb{R}^{\mathfrak{n}}$ *a* $k \leq \mathfrak{n}$*. Potom pro dimenzi Krylovova prostoru* $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ *platí*

$$dim(\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)) = \begin{cases} k & k \leq d(\mathbf{A}, \mathbf{r}_0) \\ d(\mathbf{A}, \mathbf{r}_0) & k > d(\mathbf{A}, \mathbf{r}_0). \end{cases} \tag{2.1}$$

Z (2.1) tedy plyne, že $\dim(\mathcal{K}_{\mathfrak{m}}(\mathbf{A},\mathbf{r}_0)) \leq \mathfrak{m}$. Dále v našich odvozeních předpokládejme plnou dimenzi zkonstruovaných Krylovových prostorů, tj. přesně $\mathfrak{m}$.

Velkou výhodou krylovovských metod je, že nevyžadují explicitní sestavení matice $\mathbf{A}$, namísto toho pracují pouze s násobením matice modelu s vektory a lze tudíž využít algoritmů rychlého násobení matice $\mathbf{A}$ s vektorem (viz sekce 1.3).

Krylovovských regularizačních metod je celá řada. Principem jejich fungování je projekce původního problému do Krylovova prostoru menší dimenze, přičemž tuto dimenzi iteračně zvětšujeme. Prostor je konstruován tak, aby mu dominovaly hladké vektory a aby tak docházelo k potlačení vlivu šumu z původních dat. Jelikož projektovaná úloha je velmi malé dimenze, je možné ji poměrně snadno řešit. Toto řešení je pak projektováno do původní dimenze, čímž získáváme regularizovanou aproximaci hledaného řešení $\mathbf{x}_{exact}$.

Jako zástupce zde popíšeme nejpoužívanější algoritmy – metodu LSQR a metodu GMRES spolu s její variantou RRGMRES. Vysvětlíme, jak pracují a zmíníme se o jejich konvergenci.

## 2.1 LSQR

LSQR [7] je metoda pro řešení řídkých lineárních soustav, lineárních aproximačních problémů či inverzních problémů. Zakládá se na aproximaci řešení pro

projektovanou úlohu pomocí nejmenších čtverců a hlavním cílem je minimalizovat normu rezidua.

Označme $\mathbf{x}_0$ počáteční aproximaci (často se volí nulový vektor) a

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$$

počáteční reziduum. Projekční prostor v LSQR odpovídá $\mathcal{K}_\mathfrak{m}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{r}_0)$. Počáteční vektor zde představuje zhlazené počáteční reziduum. Každý další vektor generující projekční prostor je získán násobením s $\mathbf{A}^T\mathbf{A}$, což zaručuje hladkost. Z důvodu numerické stability však nelze pracovat přímo s generujícími vektory. Budeme proto konstruovat bázi ortonormální. To nám také umožní snadné vyjádření projektované úlohy.

**Algoritmus 4:** LSQR [7]

**Input:** $\mathbf{A}, \mathbf{b}, \mathbf{x}_0$
**Output:** $\mathbf{x}_\mathfrak{m} \in \mathbf{x}_0 + \mathcal{K}_\mathfrak{m}(\mathbf{A}^T\mathbf{A},\mathbf{A}^T\mathbf{r}_0)$: $\mathbf{A}\mathbf{x}_\mathfrak{m} \approx \mathbf{b}$

1 $\mathbf{v}_0 = 0,\ \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0,\ \beta_1 = \|\mathbf{r}_0\|_2,\ \mathbf{u}_1 = \mathbf{r}_0/\beta_1$;
2 **for** $i = 1,2,\ldots$ **do**
3 $\quad \alpha_i\mathbf{v}_i = \mathbf{A}^T\mathbf{u}_i - \beta_i\mathbf{v}_{i-1}, \quad \|\mathbf{v}_i\|_2 = 1$;
4 $\quad \beta_{i+1}\mathbf{u}_{i+1} = \mathbf{A}\mathbf{v}_i - \alpha_i\mathbf{u}_i, \quad \|\mathbf{u}_{i+1}\|_2 = 1$;
5 $\quad$ **if** $\alpha_i = 0 \vee \beta_{i+1} = 0$ **then**
6 $\quad\quad \mathfrak{m} = i$;
7 $\quad\quad$ přejdi na řádek 10;
8 $\quad$ **end**
9 **end**
10 $\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}} = \mathrm{diag}([\alpha_1,\ldots,\alpha_\mathfrak{m}]) + \mathrm{diag}([\beta_2,\ldots,\beta_{\mathfrak{m}+1}], 1)$;
11 $\mathbf{V}_\mathfrak{m} = (\mathbf{v}_1,\ldots,\mathbf{v}_\mathfrak{m})$;
12 $\mathbf{y}_\mathfrak{m} = \underset{y\in\mathbb{R}^\mathfrak{m}}{\mathrm{argmin}}\ \|\beta_1\mathbf{e}_1 - \mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}\|_2$;
13 $\mathbf{x}_\mathfrak{m} = \mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m}$

Algoritmus 4 začíná hledáním ortonormální báze $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathfrak{m}$ Krylovova prostoru $\mathcal{K}_\mathfrak{m}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{r}_0)$ a pomocné báze $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_\mathfrak{m}$ prostoru $\mathcal{K}_\mathfrak{m}(\mathbf{A}\mathbf{A}^T,\mathbf{r}_0)$. K tomu využívá *Golub-Kahanovu bidiagonalizaci* [21], která probíhá na řádcích 1–9. Kromě zmíněných ortonormálních bází jsou výstupem bidiagonalizace i kladné skaláry $\alpha_1,\ldots,\alpha_\mathfrak{m}$, $\beta_1,\ldots,\beta_{\mathfrak{m}+1}$, které tvoří bidiagonální matici

$$\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}} = \begin{pmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \alpha_3 & & \\ & & \ddots & \ddots & \\ & & & \beta_\mathfrak{m} & \alpha_\mathfrak{m} \\ & & & & \beta_{\mathfrak{m}+1} \end{pmatrix} \in \mathbb{R}^{(\mathfrak{m}+1)\times\mathfrak{m}}$$

splňující

$$\begin{aligned} \mathbf{A}\mathbf{V}_\mathfrak{m} &= \mathbf{U}_{\mathfrak{m}+1}\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}} \\ \mathbf{A}^T\mathbf{U}_\mathfrak{m} &= \mathbf{V}_\mathfrak{m}\mathbf{T}_\mathfrak{m}^T, \end{aligned} \tag{2.2}$$

kde $\mathbf{U}_\mathfrak{m}, \mathbf{U}_{\mathfrak{m}+1}, \mathbf{V}_\mathfrak{m}$ označují matice s příslušnými ortonormálními vektory. Je vhodné zmínit, že celá konstrukce báze probíhá výpočetně úsporně s využitím krátkých rekurencí. To ve výsledku znamená, že pro získání dalšího bázového vektoru potřebujeme ortogonalizovat pouze proti dvěma předchozím bázovým vektorům. Díky tomu je výpočetní náročnost v každé iteraci konstantní.

Na zbylých řádcích hledá algoritmus v prostoru $\mathbf{x}_0 + \mathcal{K}_\mathfrak{m}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{r}_0)$ přibližné řešení soustavy (1.5) s minimální normou rezidua. Vektor $\mathbf{x}_\mathfrak{m}$ můžeme vyjádřit pomocí báze $\mathbf{V}_\mathfrak{m}$ jako

$$\mathbf{x}_\mathfrak{m} = \mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m} \tag{2.3}$$

a s využitím rovnice (2.2) nalezneme vhodnou volbu $\mathbf{y}_\mathfrak{m}$ pro minimalizaci normy rezidua:

$$\begin{aligned}
\|\mathbf{b} - \mathbf{A}\mathbf{x}_\mathfrak{m}\|_2 &= \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m})\|_2 \\
&= \|\mathbf{r}_0 - \mathbf{U}_{\mathfrak{m}+1}\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m}\|_2 \\
&= \|\beta\mathbf{v}_1 - \mathbf{U}_{\mathfrak{m}+1}\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m}\|_2 \\
&= \|\mathbf{U}_{\mathfrak{m}+1}(\beta\mathbf{e}_1 - \mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m})\|_2 \\
&= \|\beta\mathbf{e}_1 - \mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m}\|_2.
\end{aligned}$$

Z toho vyplývá, že přibližným řešením je vektor

$$\mathbf{x}_\mathfrak{m} = \mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m},$$

kde

$$\mathbf{y}_\mathfrak{m} = \underset{\mathbf{y}\in\mathbb{R}^\mathfrak{m}}{\operatorname{argmin}} \|\beta\mathbf{e}_1 - \mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}\|_2.$$

Projektovaná úloha je tvaru

$$\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y} \approx \beta\mathbf{e}_1$$

a její řešení hledáme ve smyslu nejmenších čtverců. Protože $\mathfrak{m}$ je typicky malé, lze k tomu využít jako vnitřní řešič přímou metodu založenou na *QR rozkladu* pomocí *Givensových rotací* [16, kapitola 5.1].

## 2.2 GMRES

Druhou metodou je GMRES (General Minimal RESidual) [8]. Jak název napovídá, i u ní je hlavním cílem minimalizace normy rezidua $\mathbf{b} - \mathbf{A}\mathbf{x}$. Projekce se však provádí jiným způsobem a do jiného Krylovova prostoru, konkrétně do $\mathcal{K}_\mathfrak{m}(\mathbf{A}, \mathbf{r}_0)$. To s sebou nese různé výhody a nevýhody.

Podstatným rozdílem je, že na rozdíl od LSQR lze tuto metodu použít pouze pro modely se čtvercovou maticí. Naše úloha zaostřování obrazu však tento předpoklad splňuje. Dále pak výpočet báze Krylovova prostoru probíhá v rámci *Arnoldiho algoritmu* [22], který nevyžaduje násobení s $\mathbf{A}^T$, ale na druhou stranu pracuje s dlouhými rekurencemi. To má za následek, že výpočetní náročnost každé další iterace roste a není tak efektivní provádět větší množství iterací.

**Algoritmus 5:** GMRES [8]

**Input:** $\mathbf{A}, \mathbf{b}, \mathbf{x}_0, \mathfrak{m}$
**Output:** $\mathbf{x}_\mathfrak{m} \in \mathbf{x}_0 + \mathcal{K}_\mathfrak{m}(\mathbf{A},\mathbf{r}_0)$: $\mathbf{A}\mathbf{x}_\mathfrak{m} \approx \mathbf{b}$

1 $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta = \|\mathbf{r}_0\|_2$, $\mathbf{v}_1 = \mathbf{r}_0/\beta$;
2 **for** $j = 1,2,\ldots,\mathfrak{m}$ **do**
3  $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$;
4  **for** $i = 1,\ldots,j$ **do**
5  $h_{i,j} = (\mathbf{w}_j, \mathbf{v}_i)$;
6  $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$;
7  **end**
8  $h_{j+1,j} = \|\mathbf{w}_j\|_2$;
9  **if** $h_{j+1,j} = 0$ **then**
10  $\mathfrak{m} = j$;
11  přejdi na řádek 15
12  **end**
13  $\mathbf{v}_{j+1} = \mathbf{w}_j/\|\mathbf{w}_j\|_2$
14 **end**
15 $\mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}} = (h_{i,j})_{\mathfrak{m}+1,\mathfrak{m}}$;
16 $\mathbf{V}_\mathfrak{m} = (\mathbf{v}_1,\ldots,\mathbf{v}_\mathfrak{m})$;
17 $\mathbf{y}_\mathfrak{m} = \underset{\mathbf{y}\in\mathbb{R}^\mathfrak{m}}{\operatorname{argmin}} \|\beta\mathbf{e}_1 - \mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}\|_2$;
18 $\mathbf{x}_\mathfrak{m} = \mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m}$

Do řádku 14 v algoritmu 5 probíhá Arnoldiho proces hledající ortonormální bázi $(\mathbf{v}_1,\ldots,\mathbf{v}_{\mathfrak{m}+1})$ Krylovova prostoru $\mathcal{K}_\mathfrak{m}(\mathbf{A},\mathbf{r}_0)$. Ortogonalizační a normalizační koeficienty se ukládají do horní Hessenbergovy matice $\mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}} \in \mathbb{R}^{(\mathfrak{m}+1)\times\mathfrak{m}}$. Její první poddiagonála obsahuje pouze kladné prvky dané normami pomocných vektorů $\mathbf{w}_j$. Výsledné matice pak splňují

$$\mathbf{A}\mathbf{V}_\mathfrak{m} \;=\; \mathbf{V}_{\mathfrak{m}+1}\mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}.$$

Obdobně jako u LSQR je zbytek algoritmu věnován hledání vhodného přibližného řešení soustavy (1.5), v tomto případě v prostoru $\mathbf{x}_0+\mathcal{K}_\mathfrak{m}(\mathbf{A},\mathbf{r}_0)$. Vyjádřením $\mathbf{x}_\mathfrak{m}$ jako (2.3) a úpravou normy rezidua nalezneme minimalizátor $\mathbf{y}_\mathfrak{m}$.

$$\begin{aligned}
\|\mathbf{b} - \mathbf{A}\mathbf{x}_\mathfrak{m}\|_2 &= \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m})\|_2 \\
&= \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m}\|_2 \\
&= \|\beta\mathbf{v}_1 - \mathbf{V}_{\mathfrak{m}+1}\mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m}\|_2 \\
&= \|\mathbf{V}_{\mathfrak{m}+1}(\beta\mathbf{e}_1 - \mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m})\|_2 \\
&= \|\beta\mathbf{e}_1 - \mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}_\mathfrak{m}\|_2
\end{aligned}$$

Přibližným řešením soustavy pro GMRES je tedy

$$\mathbf{x}_\mathfrak{m} = \mathbf{x}_0 + \mathbf{V}_\mathfrak{m}\mathbf{y}_\mathfrak{m},$$

kde

$$\mathbf{y}_\mathfrak{m} = \underset{\mathbf{y}\in\mathbb{R}^\mathfrak{m}}{\operatorname{argmin}} \|\beta\mathbf{e}_1 - \mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y}\|_2.$$

Projektovaná úloha je tentokrát ve tvaru

$$\mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{y} \approx \beta\mathbf{e}_1.$$

Opět ji lze řešit efektivně QR rozkladem za využití Givensových rotací.

## 2.3 RRGMRES

**Algoritmus 6:** RRGMRES [9]

**Input:** $\mathbf{A}, \mathbf{b}, \mathbf{x}_0, \mathfrak{m}$
**Output:** $\mathbf{x}_\mathfrak{m} \in \mathbf{x}_0 + \mathcal{K}_\mathfrak{m}(\mathbf{A},\mathbf{A}\mathbf{r}_0)$: $\mathbf{A}\mathbf{x}_\mathfrak{m} \approx \mathbf{b}$
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0,\ \beta = \|\mathbf{A}\mathbf{r}_0\|_2,\ \mathbf{v}_1 = \mathbf{A}\mathbf{r}_0/\beta,\ f_1 = (\mathbf{v}_1, \mathbf{r}_0)$;
**for** $j = 1,2,\ldots,\mathfrak{m}$ **do**
  $\mathbf{w} = \mathbf{A}\mathbf{v}_j$;
  **for** $i = 1,\ldots,j$ **do**
    $d_i = (\mathbf{w}, \mathbf{v}_i)$;
    $\mathbf{w} = \mathbf{w} - d_i\mathbf{v}_i$;
  **end**
  $\alpha = \|\mathbf{w}\|_2$;
  $\mathbf{v}_{j+1} = \mathbf{w}/\alpha$;
  $f_{j+1} = (\mathbf{v}_{j+1},\mathbf{r}_0)$;
  $\mathbf{H}(1:j,j) = \mathbf{Q}(1:j,1:j)^T\mathbf{d}(1:j)$;
  **if** $\alpha = 0$ **then**
    $\tau = 1$;
    $\sigma = 0$
  **end**
  **else if** $\alpha > |H(j,j)|$ **then**
    $\mu = -H(j,j)/\alpha$;
    $\sigma = 1/\sqrt{1+\mu^2}$;
    $\tau = \sigma\mu$;
  **end**
  **else**
    $\mu = -\alpha/H(j,j)$;
    $\tau = 1/\sqrt{1+\mu^2}$;
    $\sigma = \tau\mu$;
  **end**
  $H(j,j) = \tau H(j,j) - \sigma\alpha$;
  $\mathbf{Q}(1:j,j:j+1) = \mathbf{Q}(1:j,j) \cdot \begin{pmatrix}\tau & \sigma\end{pmatrix}$;
  $\mathbf{Q}(j+1,j:j+1) = \begin{pmatrix}-\sigma & \tau\end{pmatrix}$;
  $\mathbf{W}(:,j) = (\mathbf{v}_j - \mathbf{W}(:,1:j-1)^T\mathbf{H}(1:j-1,j))/H(j,j)$;
  $\mathbf{x}_j = \mathbf{x}_{j-1} + (\mathbf{Q}(1:j+1,j),\mathbf{f}) \cdot \mathbf{W}(:,j)$;
**end**

Alternativou k GMRES je využití obdobné metody – RRGMRES (*Range Restricted GMRES*) [9]. Výsledné řešení soustavy (1.5) se nachází v takzvaném *shiftovaném Krylovově prostoru*, jenž je definován takto:

**Definice 11** (Shiftovaný Krylovův prostor)**.** *Nechť* $\mathbf{A} \in \mathbb{R}^{\mathfrak{m}\times\mathfrak{n}}$ *je regulární, vektor* $\mathbf{r}_0 \in \mathbb{R}^{\mathfrak{n}}$ *je nenulový a* $\mathfrak{n} \geq \mathfrak{m} \geq 1$*. Potom*

$$\overrightarrow{\mathcal{K}}_{\mathfrak{m}}(\mathbf{A},\mathbf{r}_0) = \operatorname{span}\{\mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots \mathbf{A}^{\mathfrak{m}}\mathbf{r}_0\} = \mathcal{K}_{\mathfrak{m}}(\mathbf{A},\mathbf{A}\mathbf{r}_0)$$

*nazveme* $\mathfrak{m}$*-tým shiftovaným Krylovovým prostorem matice* $\mathbf{A}$ *a vektoru* $\mathbf{r}_0$ *[2].*

Využití tohoto prostoru přináší lepší regularizační vlastnosti, neboť šum z $\mathbf{r}_0$ je zhlazen maticí $\mathbf{A}$ ještě předtím, než začneme konstruovat projekční prostor, a tak jsou redukovány vysoké frekvence šumu [2].

Prvních 9 řádků algoritmu 6 odpovídá Arnoldiho procesu, zbylá část se však na první pohled liší od GMRES. V algoritmu 5 vidíme na 17. řádku hledání argumentu minima rozdílu dvou vektorů, což vede na *problém nejmenších čtverců.* Ten ve skutečnosti řešíme i v algoritmu 6 od 10. řádku, avšak zde již uvádíme variantu řešení pomocí transformace Givensovými rotacemi.

Maticové vztahy a tvar projektované úlohy pro RRGMRES zde již neuvádíme. Odvození bychom provedli obdobně jako u GMRES, avšak s využitím shiftovaného Krylovova prostoru, viz [9]. Pro úplnost uveďme, že RRGMRES používá dlouhé rekurence, neboť báze se počítá modifikací Arnoldiho procesu se startovním vektorem $\mathbf{A}\mathbf{r}_0$.

## 2.4 Konvergenční chování

Všechny výše uvedené algoritmy ze své podstaty minimalizují normu rezidua. Můžeme pro ně tedy vyslovit následující lemma:

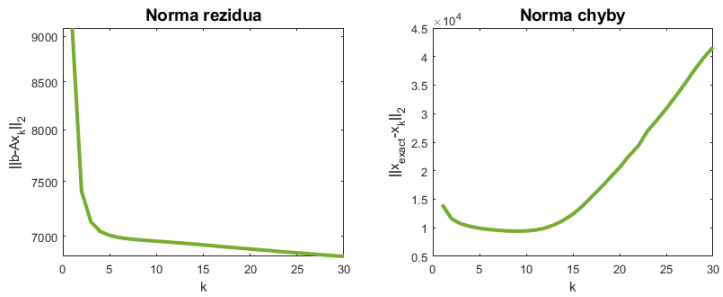
**Lemma 2.** *Nechť* $\mathbf{x}_k^L$ *označuje řešení soustavy (1.5) metodou LSQR,* $\mathbf{x}_k^G$ *metodou GMRES a* $\mathbf{x}_k^R$ *metodou RRGMRES v* $k$*-té iteraci. Potom pro všechna* $k \in \{0, 1, \ldots\}$ *platí*

$$\begin{aligned}
\left\|\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}^L\right\|_2 &\leq \left\|\mathbf{b} - \mathbf{A}\mathbf{x}_k^L\right\|_2 \\
\left\|\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}^G\right\|_2 &\leq \left\|\mathbf{b} - \mathbf{A}\mathbf{x}_k^G\right\|_2 \\
\left\|\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}^R\right\|_2 &\leq \left\|\mathbf{b} - \mathbf{A}\mathbf{x}_k^R\right\|_2.
\end{aligned}$$

Uvědomme si, že minimalizaci normy rezidua provádíme na projekčních prostorech rostoucí dimenze. Lemma 2 implikuje, že pro regulární matici $\mathbf{A}$ existuje $k < \infty$, pro které $\mathbf{A}\mathbf{x}_k = \mathbf{b}$. Problém však tkví v tom, že takové $\mathbf{x}_k$ odpovídá naivnímu řešení (1.9), o němž víme, že není vhodnou aproximací. Podívejme se proto na skutečnou chybu $\|\mathbf{x}_{exact} - \mathbf{x}_k\|$, abychom ukázali, jak „vzdálený“ je vektor $\mathbf{x}_k$ od přesného řešení $\mathbf{x}_{exact}$ [1].

Levý graf v obrázku 2.1 potvrzuje očekávaný pokles normy rezidua v LSQR pro příklad 1. Pravý graf však ukazuje, že skutečná chyba aproximace řešení nejprve klesá a od jistého indexu roste. Tomuto chování se říká *semikonvergence.* Příčinou tohoto chování je fakt, že s přibývajícím počtem iterací dochází postupně

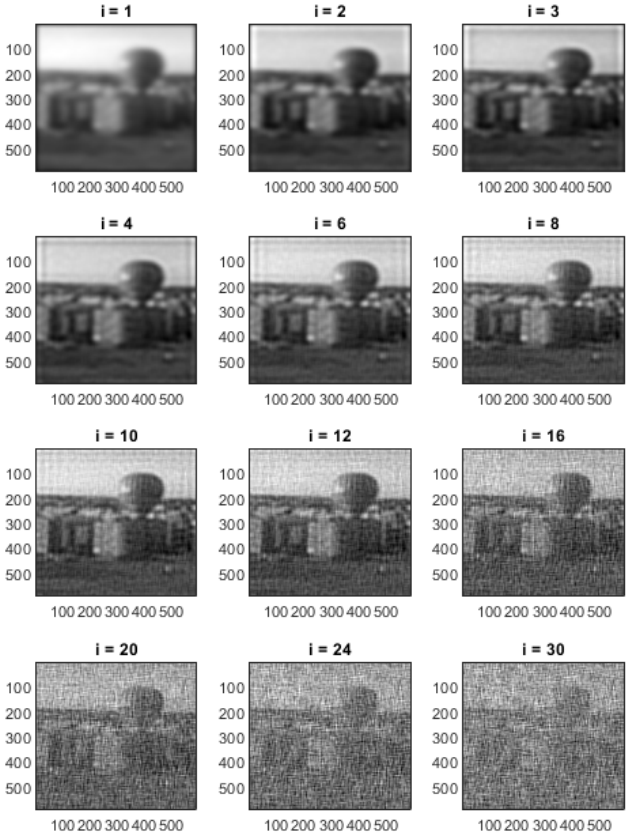
[1] V praxi $\mathbf{x}_{exact}$ neznáme, takže tuto analýzu lze provést pouze s uměle vytvořenými příklady.




Obrázek 2.1: Norma rezidua a norma chyby pro LSQR použité na příkladu 1.

k pronikání šumu z původních dat do projekčního prostoru a projektované úlohy. Spočtená aproximace řešení pak začíná být zanesena invertovaným šumem.

Obrázek 2.2 nám ukazuje, že obrázek s vektorizací matice obrazu $\mathbf{x}_k$ je ze začátku příliš hladký, postupně se zvýrazňují hrany, ale pak v obrazu začíná převládat šum. „Nejlepší“ aproximací se zdá být $\mathbf{x}_k$ pro $k = 9$, což koresponduje i s minimem funkce chyby. Jenže jak takové $k$ nalézt, když nemůžeme procházet všechny obrázky korespondující s $\mathbf{x}_k$ a hledat mezi nimi ten nejlepší?[2]




Obrázek 2.2: Přibližná řešení inverzní úlohy z příkladu 1 při využití LSQR. Obrázky odpovídají aproximacím $\mathbf{x}_i$.

[2]Poznamenejme, že obecně při řešení inverzních úloh nelze dosáhnout příliš malé normy chyby. Přesnost je vždy limitována množstvím přítomného šumu.

Metod hledání vhodného parametru $k$ je celá řada a nelze s jistotou označit jednu jako nejlepší. Vždy záleží na konkrétním typu úlohy. V praxi se lze nejčastěji setkat s *L-křivkou*, *zobecněnou cross validací*, *normalizovaným kumulativním periodogramem* a *principem diskrepance* [5]. Poslední zmíněný si nyní představíme.

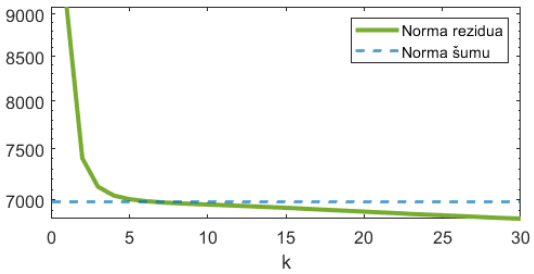
### Princip diskrepance

Princip diskrepance stojí na dvou základních předpokladech – známosti odhadu normy šumu v obrazu a použití iterační regularizační metody, kde je norma rezidua nerostoucí. Pro metody, které tento předpoklad nesplňují, lze princip diskrepance také použít, avšak implementace a konvergence takového postupu může být obtížná.

Označme $q$ jako odhad normy šumu $\mathbf{e}$. Norma šumu odpovídá normě rezidua pro přesné řešení:

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_{exact}\|_2 = \|\mathbf{b}_{exact} + \mathbf{e} - \mathbf{A}\mathbf{x}_{exact}\|_2 = \|\mathbf{e}\|_2.$$

Chtěli bychom tedy nalézt takovou aproximaci $\mathbf{x}_k$, aby odpovídající norma rezidua byla rovna přibližně $q$. Podmínka nerostoucí normy rezidua v iterační metodě zaručuje existenci parametru $k$, pro který je $\|\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}\|_2 \approx \|\mathbf{e}\|_2 \approx q$ (viz obrázek 2.3). Tato podmínka je v případě využití našich regularizačních metod splněna díky lemmatu 2.




Obrázek 2.3: Princip diskrepance. Graf ukazuje normu rezidua pro rostoucí počet iterací $k$ u metody LSQR aplikované na příklad 1.

Hledaný parametr $k$ odpovídá [23]:

$$k = \max\{l \mid \|\mathbf{b} - \mathbf{A}\mathbf{x}_l\|_2 \geq \nu_{DP} q > \|\mathbf{b} - \mathbf{A}\mathbf{x}_{l+1}\|_2\}, \tag{2.4}$$

kde $\nu_{DP} \geq 1$ označuje „bezpečnostní faktor“. Čím větší $\nu_{DP}$, tím silnější regularizace. Spolehlivost výběru regularizačního parametru budeme diskutovat v kapitole věnované numerickým experimentům.

# 3. Blokové krylovovské regularizační metody

Doposud jsme se zabývali ryze černobílými obrázky, tedy obrázky s dvojrozměrnou maticí obrazu. V předchozí kapitole jsme v rámci úlohy zaostřování obrazu na jejich obecný lineární model (1.5) aplikovali krylovovské regularizační metody. U barevného obrazu s modelem (1.8) lze zaostřený obraz hledat obdobně. Stačí model přeformulovat jako soustavu tří lineárních rovnic

$$\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$$

pro $i \in \{1, 2, 3\}$, kde každá rovnice reprezentuje rozmazání jednoho barevného kanálu. Každou z nich je možno řešit samostatně a z výsledků sestavit celkovou aproximaci barevného obrazu. Nabízí se však způsob, který daný model bere jako celek, čímž se nevytrácí souvislost mezi jednotlivými kanály.

Tím způsobem je využití *blokových krylovovských regularizačních metod*, jež si podrobněji popíšeme v této kapitole. Začněme definicí blokového Krylovova prostoru.

**Definice 12** (Blokový Krylovův prostor)**.** *Nechť* $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}$ *a* $\mathbf{R}_0 \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}$ *a* $\mathfrak{m} \geq 1$*. Potom*

$$\mathcal{B}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0) = \text{block span}\{\mathbf{R}_0, \mathbf{A}\mathbf{R}_0, \mathbf{A}^2\mathbf{R}_0, \dots \mathbf{A}^{\mathfrak{m}-1}\mathbf{R}_0\} \subseteq \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}, \tag{3.1}$$

*kde*

$$\text{block span}\{\mathbf{R}_0, \mathbf{A}\mathbf{R}_0, \mathbf{A}^2\mathbf{R}_0, \dots, \mathbf{A}^{\mathfrak{m}-1}\mathbf{R}_0\} = \left\{ \sum_{j=0}^{\mathfrak{m}-1} \mathbf{A}^j\mathbf{R}_0\mathbf{S}_j, \;\; \mathbf{S}_j \in \mathbb{R}^{\mathfrak{p}\times\mathfrak{p}} \right\}, \tag{3.2}$$

*nazveme* $\mathfrak{m}$*-tým blokovým Krylovovým prostorem matice* $\mathbf{A}$ *[24].*

V případě modelu (1.8) uvažujeme $\mathfrak{p} = 3$. Blokové krylovovské regularizační metody však mohou nalézt uplatnění i jinde než u RGB obrázků, proto model

$$\mathbf{A}\mathbf{X} = \mathbf{B} \tag{3.3}$$

nyní zobecněme pro $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}, \mathbf{X}, \mathbf{B} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}$.

Každý sloupec matice, která náleží prostoru $\mathcal{B}_\mathfrak{m}(\mathbf{A}, \mathbf{R}_0)$, náleží podprostoru

$$\mathcal{C}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0) = \text{span}\{\mathbf{r}_1, \dots, \mathbf{r}_\mathfrak{p}, \mathbf{A}\mathbf{r}_1, \dots, \mathbf{A}\mathbf{r}_\mathfrak{p}, \dots, \mathbf{A}^{\mathfrak{m}-1}\mathbf{r}_1, \dots, \mathbf{A}^{\mathfrak{m}}\mathbf{r}_\mathfrak{p}\} \subseteq \mathbb{R}^{\mathfrak{n}}, \tag{3.4}$$

kde $\mathbf{R}_0 = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \dots & \mathbf{r}_\mathfrak{p} \end{pmatrix}$. Tato množina je ekvivalentní spojením $\mathfrak{m}$-tých Krylovových prostorů matice $\mathbf{A}$ a $\mathbf{r}_i$ pro $i = \{1, \dots, \mathfrak{p}\}$, neboli

$$\mathcal{C}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0) = \mathcal{K}_\mathfrak{m}(\mathbf{A}, \mathbf{r}_1) + \dots + \mathcal{K}_\mathfrak{m}(\mathbf{A}, \mathbf{r}_\mathfrak{p}) = \text{span}\left\{ \bigcup_{i=1}^{p} \mathcal{K}_\mathfrak{m}(\mathbf{A}, \mathbf{r}_i) \right\}. \tag{3.5}$$

Blokový Krylovův prostor je tedy kartézským součinem $\mathcal{C}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0)$:

$$\mathcal{B}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0) = \underbrace{\mathcal{C}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0) \times \dots \times \mathcal{C}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0)}_{\mathfrak{p}\text{-krát}}. \tag{3.6}$$

Pomocí $\mathcal{C}_\mathfrak{m}(\mathbf{A}, \mathbf{R}_0)$ nyní můžeme definovat stupeň matice $\mathbf{R}_0$ vzhledem k matici $\mathbf{A}$ a vyslovit lemma o dimenzi.

**Definice 13** (Stupeň matice $\mathbf{R}_0$ vzhledem k matici $\mathbf{A}$)**.** *Mějme matici* $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}$ *a* $\mathbf{R}_0 \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}$*. Potom*

$$\nu(\mathbf{A},\mathbf{R}_0) = \min\{k \in \mathbb{N} \mid \mathcal{C}_k(\mathbf{A},\mathbf{R}_0) = \mathcal{C}_{k+1}(\mathbf{A},\mathbf{R}_0)\}$$

*nazveme stupněm matice* $\mathbf{R}_0$ *vzhledem k matici* $\mathbf{A}$*.*

**Lemma 3.** *Nechť* $\mathbf{A} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{n}}$*,* $\mathbf{R}_0 \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}$ *a* $k \leq \mathfrak{n}$*. Potom pro podprostor* $\mathcal{C}_k(\mathbf{A},\mathbf{R}_0)$ *a* $\mathcal{B}_k(\mathbf{A},\mathbf{R}_0)$ *platí [24]*

1. $dim(\mathcal{C}_k(\mathbf{A},\mathbf{R}_0)) \leq k\mathfrak{p}, \qquad k = 1,\ldots,\nu(\mathbf{A},\mathbf{R}_0)$
2. $\mathcal{C}_k(\mathbf{A},\mathbf{R}_0) = \mathcal{C}_{\nu(\mathbf{A},\mathbf{R}_0)}(\mathbf{A},\mathbf{R}_0), \qquad k = \nu(\mathbf{A},\mathbf{R}_0),\nu(\mathbf{A},\mathbf{R}_0)+1,\ldots$
3. $\mathcal{B}_k(\mathbf{A},\mathbf{R}_0) = \mathcal{B}_{\nu(\mathbf{A},\mathbf{R}_0)}(\mathbf{A},\mathbf{R}_0), \qquad k = \nu(\mathbf{A},\mathbf{R}_0),\nu(\mathbf{A},\mathbf{R}_0)+1,\ldots$

U neblokových Krylovových prostorů byla dimenze buď plná, nebo již došlo k dosažení maximální dimenze (viz lemma 1). U blokových je situace jiná. Z (3.4) je patrné, že může nastat situace, kdy vektor $\mathbf{A}^j\mathbf{r}_i$ je lineárně závislý na předchozích vektorech, zatímco zbylé vektory $\mathbf{A}^j\mathbf{r}_k$, $i \neq k$, jsou na nich lineárně nezávislé. V takovém případě vektory $\mathbf{A}^{j+1}\mathbf{r}_i, \mathbf{A}^{j+2}\mathbf{r}_i, \ldots$ nepřináší žádné nové informace do Krylovova prostoru a nezpůsobují nárůst jeho dimenze. Při zvětšování indexu $k$ již nedochází k růstu dimenze blokového Krylovova prostoru o celé $\mathfrak{p}$, ale jen o číslo menší dané počtem nově přidaných lineárně nezávislých vektorů [25]. Tomuto jevu se říká *deflace* a podrobněji se o něm lze dočíst v [26] a [27].

K redukci dimenze Krylovova prostoru tedy může dojít postupně a po dosažení stupně matice $\mathbf{R}_0$ vzhledem k matici $\mathbf{A}$ se růst zastaví zcela. Jelikož k deflaci dochází typicky v pozdějších iteracích, kterých v inverzních úlohách z důvodu semikonvergence nedosahujeme, budeme uvažovat pouze problémy, ve kterých k deflaci nedochází.

Máme-li definovány prostory, můžeme je využít ke konstrukci aproximací problému (3.3). Samotný princip fungování blokových metod se od neblokových metod zvláště neliší. I u nich je základem projektování úlohy do prostoru menší dimenze s dominancí hladkých vektorů a projektování zde získané aproximace řešení do původní dimenze.

Stejně jako v předchozí kapitole si nyní představíme konkrétní metody – blokový LSQR, blokový GMRES a blokový RRGMRES. Zaměříme se na jejich strukturu a konvergenční vlastnosti. Než přejdeme k jednotlivým metodám, zaveďme definici *blokové ortonormality.*

**Definice 14** (Bloková ortonormalita)**.** *Nechť* $\mathbf{G} = \left(\begin{array}{c|c|c} \mathbf{G}_1 & \ldots & \mathbf{G}_{\mathfrak{m}} \end{array}\right) \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{m}\mathfrak{p}}$ *je bloková matice. Potom řekneme, že* $\mathbf{G}$ *je blokově ortonormální, pokud pro všechna* $i,j \in \{1,\ldots\mathfrak{m}\}$

$$\mathbf{G}_i^T\mathbf{G}_j = \begin{cases} \mathbf{I}_{\mathfrak{p}} & i = j \\ \mathbf{O}_{\mathfrak{p}} & i \neq j. \end{cases},$$

*kde* $\mathbf{I}_{\mathfrak{p}} \in \mathbb{R}^{\mathfrak{p}\times\mathfrak{p}}$ *značí jednotkovou matici a* $\mathbf{O}_{\mathfrak{p}} \in \mathbb{R}^{\mathfrak{p}\times\mathfrak{p}}$ *nulovou matici.*

## 3.1 Blokový LSQR

Blokový LSQR byl představen dvojicí Karimi a Toutounian v roce 2006, a to pro obecně obdélníkovou matici $\mathbf{A}$ [10]. O jeho aplikaci v inverzních úlohách se lze dočíst v publikacích představených teprve nedávno [12], případně v [6]. Metoda projektuje úlohu (3.3) do prostoru $\mathcal{B}_\mathfrak{m}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{R}_0)$, kde

$$\mathbf{R}_0 = \mathbf{B} - \mathbf{A}\mathbf{X}_0$$

označuje počáteční reziduum a $\mathbf{X}_0$ počáteční aproximaci. Obdobně jako v neblokové metodě je obvyklou volbou $\mathbf{X}_0$ nulová matice. Všimněme si, že $i$-tý sloupec $\mathbf{X}_0$ reprezentuje vždy počáteční aproximaci pro $i$-tý sloupec matice $\mathbf{X}$, tedy platí

$$\mathbf{r}_i = \mathbf{b}_i - \mathbf{A}\mathbf{X}_0\mathbf{e}_i.$$

**Algoritmus 7:** Blokový LSQR [10]

**Input:** $\mathbf{A}, \mathbf{B}, \mathbf{X}_0$
**Output:** $\mathbf{X}_\mathfrak{m} \in \mathbf{X}_0 + \mathcal{B}_\mathfrak{m}(\mathbf{A}^T\mathbf{A},\mathbf{A}^T\mathbf{R}_0)$: $\mathbf{A}\mathbf{X}_\mathfrak{m} \approx \mathbf{B}$

1 $\mathbf{R}_0 = \mathbf{B} - \mathbf{A}\mathbf{X}_0$;
2 $[\mathbf{U}_1, \mathbf{R}_1] = QR(\mathbf{R}_0)$;
3 $[\mathbf{V}_1, \mathbf{L}_1^T] = QR(\mathbf{A}^T\mathbf{U}_1)$;
4 **for** $k = 1,2,\ldots,\mathfrak{m}$ **do**
5 $\quad [\mathbf{U}_{k+1},\mathbf{R}_{k+1}] = QR(\mathbf{A}\mathbf{V}_k - \mathbf{U}_k\mathbf{L}_k)$ ;
6 $\quad [\mathbf{V}_{k+1}, \mathbf{L}_{k+1}^T] = QR(\mathbf{A}^T\mathbf{U}_{k+1} - \mathbf{V}_k\mathbf{R}_{k+1}^T)$;
7 **end**
8 $\hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}} = \mathrm{diag}([\mathbf{L}_1,\ldots,\mathbf{L}_\mathfrak{m}]) + \mathrm{diag}([\mathbf{R}_2,\ldots,\mathbf{R}_{m+1}],1)$;
9 $\hat{\mathbf{V}}_\mathfrak{m} = \left(\begin{array}{c|c|c|c} \mathbf{V}_1 & \mathbf{V}_2 & \cdots & \mathbf{V}_\mathfrak{m} \end{array}\right)$;
10 $\mathbf{Y}_\mathfrak{m} = \underset{\mathbf{Y}\in\mathbb{R}^{\mathfrak{p}\mathfrak{m}\times\mathfrak{p}}}{\mathrm{argmin}} \left\|\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}\right\|_F$;
11 $\mathbf{X}_\mathfrak{m} = \mathbf{X}_0 + \hat{\mathbf{V}}_\mathfrak{m}\mathbf{Y}_\mathfrak{m}$

Algoritmus 7 začíná *Golub-Kahanovou blokovou bidiagonalizací*. Výstupem je blokově ortonormální báze $\mathbf{V}_1,\ldots,\mathbf{V}_\mathfrak{m}$ prostoru $\mathcal{B}_\mathfrak{m}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{R}_0)$ a blokově ortonormální báze $\mathbf{U}_1,\ldots,\mathbf{U}_\mathfrak{m} \in \mathcal{B}_\mathfrak{m}(\mathbf{A}\mathbf{A}^T, \mathbf{R}_0)$, jež je pomocná. Tento postup odpovídající řádkům 1–7 uvedeného algoritmu si nyní podrobněji vysvětlíme.

Pro konstrukci $\mathbf{U}_1 \in \mathcal{B}_\mathfrak{m}(\mathbf{A}\mathbf{A}^T, \mathbf{R}_0)$ potřebujeme nalézt ortonormální bázi oboru hodnot $\mathbf{R}_0$. K tomu je možné využít ekonomického QR rozkladu, jak je tomu na řádku 2 (QR značí výpočet rozkladu). Obecnou matici $\mathbf{M} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}$ rozkládá na ortonormální matici $\mathbf{Q} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}}$ a horní trojúhelníkovou matici $\mathbf{R} \in \mathbb{R}^{\mathfrak{p}\times\mathfrak{p}}$ [16, kapitola 5.2]. Analogicky probíhá konstrukce $\mathbf{V}_1 \in \mathcal{B}_\mathfrak{m}(\mathbf{A}\mathbf{A}^T, \mathbf{R}_0)$ rozkladem matice $\mathbf{A}^T\mathbf{U}_1$ na řádku 3. Kromě hledaných bází získáme také trojúhelníkové matice $\mathbf{R}_1$ a $\mathbf{L}_1$, které jsou regulární (dle předpokladu, že nedochází k deflaci). Vidíme, v čem je algoritmus komplikovanější než původní verze pro jeden startovní vektor, který stačilo normalizovat (viz algoritmus 4). Poznamenejme, že jelikož je $\mathfrak{p}$ typicky malé ($\mathfrak{p} << \mathfrak{n}$), QR rozklad lze počítat zpětně stabilně s malými výpočetními náklady, například pomocí *Householderových reflexí* [28], [16, kapitola 5.1].

Zbylé bloky bází generujeme ve `for cyklu` na řádcích 4–7 jako

$$\begin{aligned} [\mathbf{U}_{k+1}, \mathbf{R}_{k+1}] &= QR(\mathbf{A}\mathbf{V}_k - \mathbf{U}_k\mathbf{L}_k) \\ [\mathbf{V}_{k+1}, \mathbf{L}_{k+1}^T] &= QR(\mathbf{A}^T\mathbf{U}_{k+1} - \mathbf{V}_k\mathbf{R}_{k+1}^T) \end{aligned} \tag{3.7}$$

pro $k \in \{1, \ldots, \mathfrak{m}\}$. QR rozklady jsou opět ekonomické, výpočetně nákladnější operací zůstává násobení s maticí $\mathbf{A}$ a $\mathbf{A}^T$, které provádíme pro úlohy se strukturovanou maticí z naší aplikace pomocí FFT analogicky jako pro vektorový případ.

Matice $\mathbf{L}_k, \mathbf{R}_k$ dohromady tvoří blokově bidiagonální matici

$$\mathbf{T}_{\mathfrak{m}+1,\mathfrak{m}} = \begin{pmatrix} \mathbf{L}_1 & & & & \\ \mathbf{R}_2 & \mathbf{L}_2 & & & \\ & \mathbf{R}_3 & \mathbf{L}_3 & & \\ & & \ddots & \ddots & \\ & & & \mathbf{R}_{\mathfrak{m}} & \mathbf{L}_{\mathfrak{m}} \\ & & & & \mathbf{R}_{\mathfrak{m}+1} \end{pmatrix} \in \mathbb{R}^{(\mathfrak{m}+1)\mathfrak{p} \times \mathfrak{m}\mathfrak{p}},$$

která je zároveň $(\mathfrak{p}+1)$-diagonální[1] a zobecňuje tak bidiagonální matici z klasické neblokové bidiagonalizace. Z popisu konstrukce je patrné využití krátkých rekurencí, které přináší výpočetní úsporu.

Rekurenční vztahy (3.7) lze přepsat i jako

$$\begin{aligned} \hat{\mathbf{U}}_{\mathfrak{m}+1}\mathbf{R}_1\mathbf{E}_1 &= \mathbf{R}_0 \\ \mathbf{A}\hat{\mathbf{V}}_{\mathfrak{m}} &= \hat{\mathbf{U}}_{\mathfrak{m}+1}\hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}} \\ \mathbf{A}^T\hat{\mathbf{U}}_{\mathfrak{m}+1} &= \mathbf{V}_{\mathfrak{m}}\hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}^T + \mathbf{V}_{\mathfrak{m}+1}\mathbf{L}_{\mathfrak{m}+1}^T\mathbf{E}_{\mathfrak{m}+1}^T, \end{aligned} \tag{3.8}$$

kde $\mathbf{E}_1$ odpovídá matici rozměru $\mathfrak{n} \times \mathfrak{p}$, která má na prvních $\mathfrak{p}$ řádcích jednotkovou matici $\mathfrak{p} \times \mathfrak{p}$ a zbylé prvky nulové,

$$\hat{\mathbf{V}}_{\mathfrak{m}} = \left( \mathbf{V}_1 \,\middle|\, \mathbf{V}_2 \,\middle|\, \cdots \,\middle|\, \mathbf{V}_{\mathfrak{m}} \right) \in \mathbb{R}^{n \times \mathfrak{m}\mathfrak{p}}$$

a

$$\hat{\mathbf{U}}_{\mathfrak{m}+1} = \left( \mathbf{U}_1 \,\middle|\, \mathbf{U}_2 \,\middle|\, \cdots \,\middle|\, \mathbf{U}_{\mathfrak{m}+1} \right) \in \mathbb{R}^{n \times (\mathfrak{m}+1)\mathfrak{p}}.$$

Díky blokové ortonormalitě jednotlivých bloků a skutečnosti, že každý tento blok pochází z QR rozkladu, mají matice $\hat{\mathbf{V}}_{\mathfrak{m}}$ a $\hat{\mathbf{U}}_{\mathfrak{m}+1}$ ortonormální sloupce.

Matici $\mathbf{X}_m$ můžeme v prostoru $\mathbf{X}_0 + \mathcal{B}_{\mathfrak{m}}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{R}_0)$ vyjádřit jako

$$\mathbf{X}_{\mathfrak{m}} = \mathbf{X}_0 + \hat{\mathbf{V}}_{\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}}. \tag{3.9}$$

Využitím (3.9) a (3.8) dostáváme

$$\begin{aligned} \|\mathbf{B} - \mathbf{A}\mathbf{X}_{\mathfrak{m}}\|_F &= \left\|\mathbf{B} - \mathbf{A}(\mathbf{X}_0 + \hat{\mathbf{V}}_{\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}})\right\|_F \\ &= \left\|\mathbf{R}_0 - \hat{\mathbf{U}}_{\mathfrak{m}+1}\hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}}\right\|_F \\ &= \left\|\hat{\mathbf{U}}_{\mathfrak{m}+1}(\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}})\right\|_F \\ &= \left\|\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}}\right\|_F, \end{aligned}$$

přičemž poslední rovnost platí díky ortonormalitě sloupců $\hat{\mathbf{U}}_{\mathfrak{m}+1}$. Aproximované řešení s nejmenší normou rezidua odpovídá

$$\mathbf{X}_{\mathfrak{m}} = \mathbf{X}_0 + \hat{\mathbf{V}}_{\mathfrak{m}}\mathbf{Y}_m,$$

[1] $(\mathfrak{p}+1)$-diagonální matice se definuje podobně jako diagonální (neboli „1-diagonální“) a bidiagonální („2-diagonální“).

kde

$$\mathbf{Y}_\mathfrak{m} = \underset{\mathbf{Y}\in\mathbb{R}^{\mathfrak{p}\mathfrak{m}\times\mathfrak{p}}}{\operatorname{argmin}} \left\|\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}\right\|_F.$$

Jinými slovy,

$$\hat{\mathbf{T}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y} \approx \mathbf{R}_1\mathbf{E}_1$$

je projekcí úlohy (3.3) do prostoru $\mathcal{B}_\mathfrak{m}(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{R}_0)$, řešení úlohy hledáme ve smyslu nejmenších čtverců. Využívá se QR rozkladu pomocí Givensových rotací. Oproti neblokové variantě metody LSQR, kde byla třeba jedna Givensova rotace v každé iteraci, zde budeme potřebovat obecně až $\mathfrak{p}$ rotací, tedy celkem $\mathfrak{m}\cdot\mathfrak{p}$ po $\mathfrak{m}$ iteracích.

## 3.2 Blokový GMRES a RRGMRES

Druhou blokovou metodou, kterou zde popíšeme, je blokový GMRES. Gallopoulos a Simoncini jej představili v základním tvaru pro řešení klasických soustav lineárních algebraických rovnic v roce 1996 [11]. Rozšíření pro řešení diskrétních inverzních úloh nebylo doposud představeno a testováno. Proto nejprve metodu zformulujeme a popíšeme. Následně budeme studovat její numerické chování. Projekčním prostorem je $\mathcal{B}_\mathfrak{m}(\mathbf{A}, \mathbf{R}_0)$ a hledání jeho báze je postaveno na Arnoldiho algoritmu [29].

Stejně jako u blokového LSQR do hry vstupuje QR rozklad zaručující ortonormalitu báze. První bázovou matici $\mathbf{V}_1$ a její blokový koeficient $\mathbf{R}_1$ dostaneme z QR rozkladu počátečního rezidua $\mathbf{R}_0$, tedy

$$[\mathbf{V}_1, \mathbf{R}_1] = QR(\mathbf{R}_0).$$

**Algoritmus 8:** Blokový GMRES [30]

**Input:** $\mathbf{A}, \mathbf{B}, \mathbf{X}_0$
**Output:** $\mathbf{X}_\mathfrak{m} \in \mathbf{X}_0 + \mathcal{B}_\mathfrak{m}(\mathbf{A},\mathbf{R}_0)$: $\mathbf{A}\mathbf{X}_\mathfrak{m} \approx \mathbf{B}$

$\mathbf{R}_0 = \mathbf{B} - \mathbf{A}\mathbf{X}_0$;
$[\mathbf{V}_1,\mathbf{R}_1] = QR(\mathbf{R}_0)$;
**for** $k = 1,2,\ldots,\mathfrak{m}$ **do**
  $\mathbf{W}_k = \mathbf{A}\mathbf{V}_k$;
  **for** $i = 1,2,\ldots,k$ **do**
    $\mathbf{H}_{i,k} = \mathbf{V}_i^T \cdot \mathbf{W}_k$;
    $\mathbf{W}_k = \mathbf{W}_k - \mathbf{V}_i\mathbf{H}_{i,k}$;
  **end**
  $[\mathbf{V}_{k+1},\mathbf{H}_{k+1,k}] = QR(\mathbf{W}_k)$;
**end**

$$\hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}} = \left(\begin{array}{c|c|c|c} \mathbf{H}_{1,1} & \mathbf{H}_{1,2} & \cdots & \mathbf{H}_{1,\mathfrak{m}} \\ \hline \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \cdots & \mathbf{H}_{2,\mathfrak{m}} \\ \hline 0 & \mathbf{H}_{3,2} & \cdots & \mathbf{H}_{3,\mathfrak{m}} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & \mathbf{H}_{\mathfrak{m}+1,\mathfrak{m}} \end{array}\right);$$

$$\hat{\mathbf{V}}_\mathfrak{m} = \left(\begin{array}{c|c|c|c} \mathbf{V}_1 & \mathbf{V}_2 & \cdots & \mathbf{V}_\mathfrak{m} \end{array}\right);$$

$$\mathbf{Y}_\mathfrak{m} = \underset{\mathbf{Y}\in\mathbb{R}^{\mathfrak{p}\mathfrak{m}\times\mathfrak{p}}}{\operatorname{argmin}} \left\|\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}\right\|_F;$$

$\mathbf{X}_\mathfrak{m} = \mathbf{X}_0 + \hat{\mathbf{V}}_\mathfrak{m}\mathbf{Y}_\mathfrak{m}$

Zbytek báze konstruujeme jako

$$[\mathbf{V}_{k+1},\mathbf{H}_{k+1,k}] = QR(\mathbf{A}\mathbf{V}_k - \sum_{i=1}^{k} \mathbf{V}_i\mathbf{H}_{i,k}), \tag{3.10}$$

kde $k = 1,\ldots,\mathfrak{m}$, $i = 1,\ldots,k$ a

$$\mathbf{H}_{i,k} = \mathbf{V}_i^T\mathbf{A}\mathbf{V}_k.$$

Bloky $\mathbf{H}_{i,k}$ jsou základem matice $\hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}} \in \mathbb{R}^{\mathfrak{p}(\mathfrak{m}+1)\times\mathfrak{p}\mathfrak{m}}$, přičemž $\mathbf{H}_{k+1,k}$ jsou z definice QR rokladu horní trojúhelníkové. Navíc jsou regulární, neboť vylučujeme deflaci. Výsledné bázové matice $\mathbf{V}_1,\ldots,\mathbf{V}_{\mathfrak{m}+1}$ jsou blokově ortonormální a tvoří matici $\hat{\mathbf{V}}_{\mathfrak{m}+1} \in \mathbb{R}^{\mathfrak{n}\times\mathfrak{p}(\mathfrak{m}+1)}$, pro kterou platí

$$\mathbf{A}\hat{\mathbf{V}}_{\mathfrak{m}} = \hat{\mathbf{V}}_{\mathfrak{m}+1}\hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}}.$$

Matice $\hat{\mathbf{V}}_{\mathfrak{m}+1}$ má navíc ortonormální všechny sloupce, jelikož jednotlivé bloky jsou výstupy QR rozkladu.

Vyjádřením normy rezidua jako

$$\begin{aligned}
\|\mathbf{B} - \mathbf{A}\mathbf{X}_{\mathfrak{m}}\|_F &= \left\|\mathbf{B} - \mathbf{A}(\mathbf{X}_0 + \hat{\mathbf{V}}_{\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}})\right\|_F \\
&= \left\|\mathbf{R}_0 - \mathbf{A}\hat{\mathbf{V}}_{\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}}\right\|_F \\
&= \left\|\hat{\mathbf{V}}_{\mathfrak{m}+1}(\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}})\right\|_F \\
&= \left\|\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}}\right\|_F
\end{aligned}$$

dostáváme přibližné řešení soustavy (3.3) v prostoru $\mathbf{X}_0 + \mathcal{B}_{\mathfrak{m}}(\mathbf{A},\mathbf{R}_0)$ s minimální normou. Tím je

$$\mathbf{X}_{\mathfrak{m}} = \mathbf{X}_0 + \hat{\mathbf{V}}_{\mathfrak{m}}\mathbf{Y}_{\mathfrak{m}},$$

kde

$$\mathbf{Y}_{\mathfrak{m}} = \underset{\mathbf{Y}\in\mathbb{R}^{\mathfrak{p}\mathfrak{m}\times\mathfrak{p}}}{\operatorname{argmin}} \left\|\mathbf{R}_1\mathbf{E}_1 - \hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y}\right\|_F.$$

Projektovaná úloha je ve tvaru

$$\hat{\mathbf{H}}_{\mathfrak{m}+1,\mathfrak{m}}\mathbf{Y} \approx \mathbf{R}_1\mathbf{E}_1$$

a její řešení hledáme QR rozkladem pomocí $\mathfrak{m}\cdot\mathfrak{p}^2$ Givensových rotací, případně pomocí $\mathfrak{m}\cdot\mathfrak{p}$ Householderových reflexí [26]. Konstrukce je postavena na dlouhých rekurencích, což s sebou přináší stejně jako u neblokového GMRES problém v podobě rychle rostoucích výpočetních nákladů s každou novou iterací.

Blokový RRGMRES je alternativou klasického blokového GMRES, představen byl teprve v roce 2023 [31]. Tato varianta je známá jako l-shifted BGMRES pro $l \geq 1$, přičemž 1-shifted GMRES odpovídá blokovému RRGMRES. Jako projekční prostor využívá *shiftovaný blokový Krylovův prostor* $\overrightarrow{\mathcal{B}}_{\mathfrak{m}}(\mathbf{A},\mathbf{R}_0) = \mathcal{B}_{\mathfrak{m}}(\mathbf{A},\mathbf{A}\mathbf{R}_0)$. Tento prostor umožňuje zlepšit regularizační vlastnosti tím, že zhlazuje šum v počátečním vektoru $\mathbf{R}_0$ maticí A již před konstrukcí projekčního prostoru. Tím dochází ke snížení vysokofrekvenčních složek šumu.

Stejně jako blokový GMRES hledá ortonormální bázi tohoto prostoru Arnoldiho procesem, tentokrát však se startovní maticí $\mathbf{A}\mathbf{R}_0$. To znamená, že i zde vstupují do hry dlouhé rekurence a s nimi spojený problém s výpočetními náklady. Minimalizace normy rezidua se řeší ve smyslu nejmenších čtverců, k čemuž se využívá QR rozklad pomocí Givensových rotací.

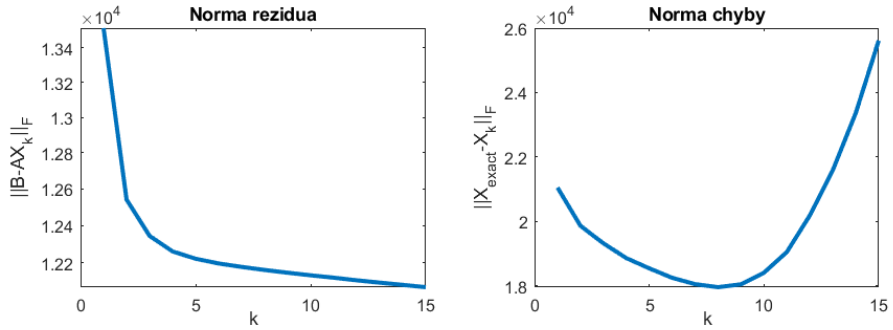
## 3.3 Konvergenční chování

Konvergenční chování blokových metod LSQR, GMRES a RRGMRES je analogické k jejich neblokovým verzím. Chování normy rezidua shrnuje následující lemma:

**Lemma 4.** *Nechť $\mathbf{X}_k^L$ označuje řešení soustavy (3.3) blokové metody LSQR, $\mathbf{X}_k^G$ blokové metody GMRES a $\mathbf{X}_k^R$ blokové metody RRGMRES v $k$-té iteraci. Potom pro všechna $k \in \{0, 1, \ldots\}$ platí*

$$\begin{aligned} \left\|\mathbf{B} - \mathbf{A}\mathbf{X}_{k+1}^L\right\|_F &\leq \left\|\mathbf{B} - \mathbf{A}\mathbf{X}_k^L\right\|_F \\ \left\|\mathbf{B} - \mathbf{A}\mathbf{X}_{k+1}^G\right\|_F &\leq \left\|\mathbf{B} - \mathbf{A}\mathbf{X}_k^G\right\|_F \\ \left\|\mathbf{B} - \mathbf{A}\mathbf{X}_{k+1}^R\right\|_F &\leq \left\|\mathbf{B} - \mathbf{A}\mathbf{X}_k^R\right\|_F. \end{aligned}$$

Všechny diskutované blokové metody opět minimalizují (tentokrát maticovou) normu rezidua. Důležité však je se podívat na vývoj skutečné chyby řešení. Levý graf v obrázku 3.1 znázorňuje normu rezidua v blokovém LSQR pro příklad 1 s barevným obrázkem, což koresponduje s lemmatem 4. Pravý graf pak ukazuje chování normy chyby, která obdobně jako v neblokovém případě *semikonverguje*. Konvergenční chování blokové metody LSQR bylo diskutováno pro soustavy klasických lineárních algebraických rovnic v [32].




Obrázek 3.1: Norma rezidua a norma chyby pro blokový LSQR použitý na příkladu 1 s barevným obrázkem.

Na závěr sekce se ještě zastavme u zastavovacích kritérií. I u blokových metod je potřeba podle počtu iterací zvolit odpovídající strategii výběru regularizačního parametru, pro který je skutečná chyba $\|X_{exact} - X_k\|_F$ co nejmenší (bez znalosti $\mathbf{X}_{exact}$). Nabízí se možnost zobecnit již představený **princip diskrepance** z vektorové varianty na maticovou. V případě znalosti odhadu Frobeniovy normy $\hat{q}$ matice šumu $\mathbf{E}$ lze zvolit vhodný počet iterací jako

$$k = \max\{l \mid \|\mathbf{B} - \mathbf{A}\mathbf{X}_l\|_F \geq \nu_{DP}\hat{q} > \|\mathbf{B} - \mathbf{A}\mathbf{X}_{l+1}\|\},$$

kde $\nu_{DP} \geq 1$ odpovídá „bezpečnostnímu faktoru“, jenž reguluje míru regularizace. Tuto strategii budeme testovat dále v numerických experimentech. Podobně by bylo možné zobecnit i další metody výběru regularizačního parametru pro blokové regularizační algoritmy, jako například GCV a L-křivku. Jejich odvození se však již věnovat nebudeme.

# 4. Numerické experimenty

V této kapitole se dostáváme k numerickým experimentům. Experimenty byly provedeny na čtyřjádrovém procesoru Intel Core i7-10510U se základní frekvencí 2,3 GHz a s 16 GB paměti RAM. Implementace proběhla v Matlabu R2021a s Regularizations Tools [33] a IR Tools [15]. Zmíněné toolboxy však byly využity pouze pro generování příkladů 2 a 4. V experimentální části jsme implementovali funkce a skripty výhradně s využitím základních vestavěných funkcí.

Obrázek reprezentujeme nezápornou maticí s prvky z množiny

$$M = \{0, \dots, 255\}$$

(více viz sekce 1.1). Když ale matici obrazu vynásobíme maticí (rozmažeme) a přičteme matici (přidáme šum), může se snadno stát, že některé prvky této matice rozmazaného obrázku nebudou v $M$. Tento problém je potřeba ošetřit, zvolíme tedy přístup, jak tuto matici transformovat, aby odpovídala obrazové reprezentaci.

Prvním přístupem, který se používá v praxi především, je prvky z matice rozmazaného obrázku zaokrouhlit na celé číslo, zároveň všechny záporné hodnoty převést na nulu a hodnoty větší než 255 převést na 255. Formálněji řečeno, všechny prvky $b_{i,j}$ matice $\mathbf{B}$ transformujeme funkcí

$$f(b_{i,j}) = \lfloor \min\{1, \max\{0, b_{i,j}\}\} \rceil, \tag{4.1}$$

kde $\lfloor \cdot \rceil$ značí zaokrouhlování na celá čísla. Alternativou je pak přeškálování – označíme-li maximum matice $\mathbf{B}$ jako $b_{max}$ a minimum jako $b_{min}$, přeškálování odpovídá funkci

$$f(b_{i,j}) = \left\lfloor 255 \cdot \frac{b_{i,j} - b_{min}}{b_{max} - b_{min}} \right\rceil. \tag{4.2}$$

Algoritmy LSQR, GMRES, RRGMRES a jejich blokové varianty jsme implementovali dle algoritmů 4, 5, 6, 7 a 8. Rozdíl byl pouze ve vstupu, neboť namísto matice $\mathbf{A}$ byla vstupem matice point spread funkce $\mathbf{P}$. Tu jsme ve všech příkladech generovali funkcí `fspecial` s parametry *'gaussian'*, *hsize* = [17 17] a $\sigma = 4$. To odpovídá jádru s maticí rozměru $17 \times 17$ s předpisem

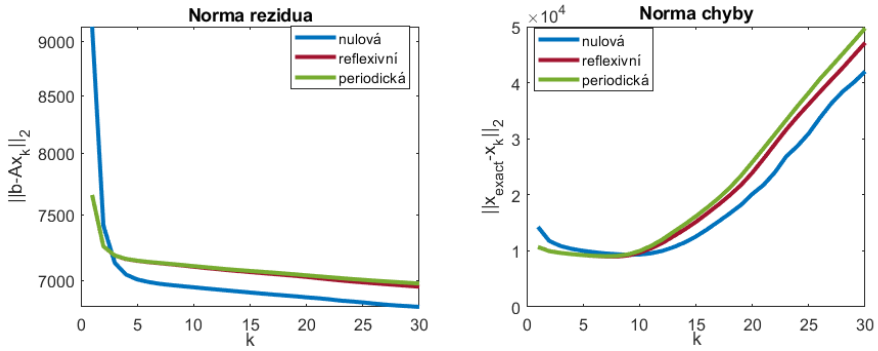
$$p_{i,j} = \frac{1}{2 \cdot 4^2 \pi} \exp\left( \frac{-i^2 + j^2}{2 \cdot 4^2} \right). \tag{4.3}$$

Násobení matice s vektorem jsme realizovali jako konvoluci s výše popsanou maticí PSF za použití funkce `conv2`, respektive `imfilter` (při použití jiné než nulové okrajové podmínky). V případě blokových metod jsme se museli vypořádat s násobením dvou matic. Jelikož jsme $\mathbf{A}$ explicitně nesestavovali, algoritmus pro násobení dvou matic obsahuje cyklus s již implementovaným násobením matice s vektorem.

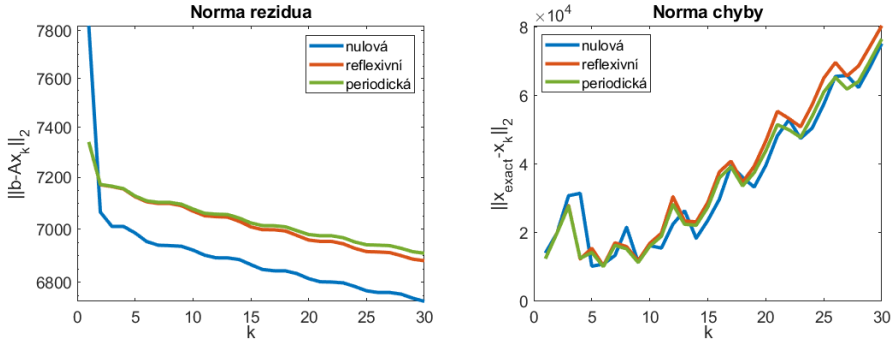
## 4.1 Vliv okrajových podmínek

V prvním experimentu se vrátíme k sekci 1.2.5, ve které jsme si představili okrajové podmínky. Porovnáme normu rezidua a normu chyby u aproximovaných řešení příkladu 1 pro metody LSQR, GMRES a RRGMRES při využití nulové, reflexivní a periodické okrajové podmínky.

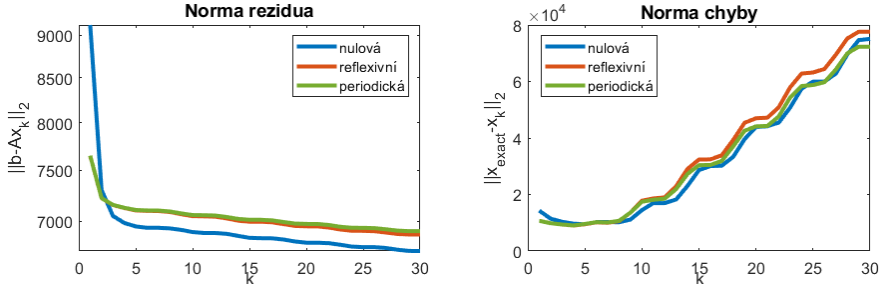
I když během prvních iterací je norma rezidua výrazně vyšší při využití nulové okrajové podmínky, po přibližně třech iteracích se ve všech metodách dostává pod úroveň norem při využití zbývajících dvou podmínek (viz obrázky 4.1–4.3). Na základě těchto výsledků by se mohlo zdát, že volbou nulové podmínky lze rychleji získat dobrou aproximaci. Když se však zaměříme na normu chyby, zjistíme, že tomu tak není.




Obrázek 4.1: Norma rezidua a norma chyby LSQR pro různé okrajové podmínky.

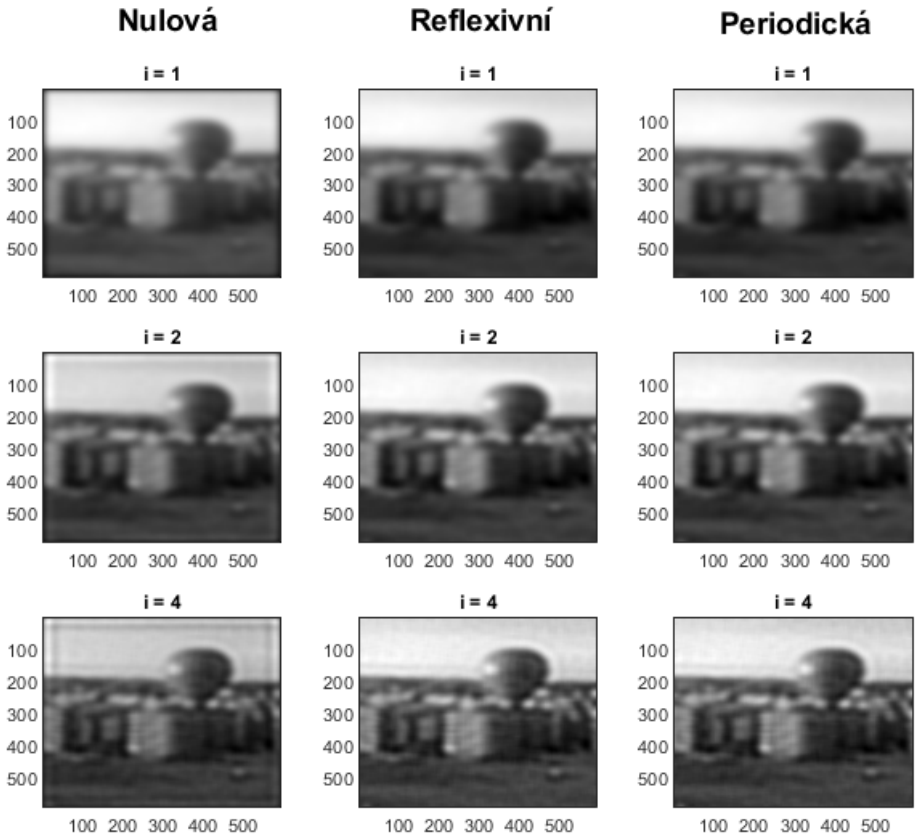



Obrázek 4.2: Norma rezidua a norma chyby GMRES pro různé okrajové podmínky.




Obrázek 4.3: Norma rezidua a norma chyby RRGMRES pro různé okrajové podmínky.

Chování normy chyby vykazuje více odlišností napříč metodami, naopak mezi jednotlivými okrajovými podmínkami je rozdíl nepatrný. Normu chyby nejlépe ilustruje obrázek 4.4 znázorňující aproximovaná řešení po jednotlivých iteracích. Zatímco reflexivní a periodická podmínka dávají podobné výsledky, u nulové podmínky je nejprve patrné tmavé zvýraznění okrajů, jež postupně snižuje intenzitu světlých částí a zanechává další viditelné stopy po uměle vytvořených okrajích.

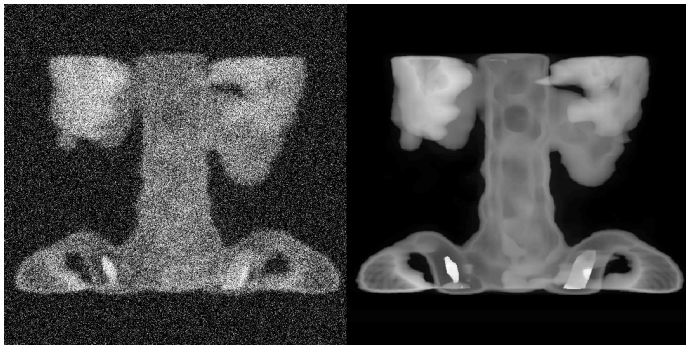



Obrázek 4.4: Přibližná řešení soustavy při využití LSQR s různými okrajovými podmínkami. Obrázky odpovídají vektorovým aproximacím $x_i$ převedeným do maticového tvaru.

Z těchto pozorování by šlo poměrně snadno vyvodit závěr, že nejvýhodnější je volit reflexivní, případně periodickou okrajovou podmínku, a že nulová okrajová podmínka nedává ideální výsledky. Pokud bychom však měli na vstupu obrázek s tmavými okraji, byla by situace trochu jiná. Podívejme se na následující příklad:

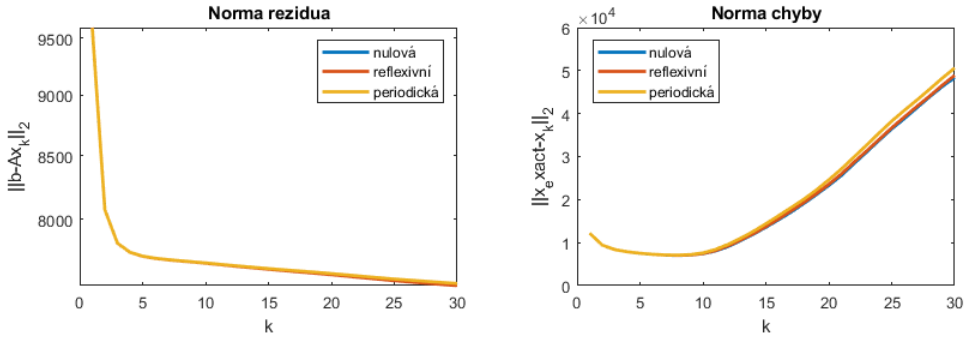
*Příklad* 3. Máme obrázek `spine.mat` velikosti $975 \times 975$ pixelů z knihovny obrázků Matlabu. K němu byl přidán aditivní Gaussův šum s $\mu = 10^{-1}$ a rozmazání odpovídá vzorci 4.3 (viz obrázek 4.5).

Jak ukazují grafy na obrázku 4.6, norma rezidua i norma chyby se zde chovají pro všechny podmínky téměř identicky. Zaostřovaný obraz totiž obsahuje výrazné černé okraje, což odpovídá nulám na okrajových pozicích v matici obrazu $\mathbf{B}$. Znamená to tedy, že zde je potřeba zvolit vhodnou okrajovou podmínku na základě jiných parametrů.



Obrázek 4.5: Vlevo zaznamenaný obrázek, vpravo přesné řešení příkladu 3.

Obecně je lepší pracovat s reflexivní či periodickou podmínku, neboť jsou nezávislé na intenzitě okrajových částí obrazu. Výhodou periodické okrajové podmínky je možnost reprezentovat matici **A** velmi úsporně (viz 1.13), což má za důsledek optimalizaci časové náročnosti. Naopak reflexivní okrajová podmínka se hodí v případě, kdy krajní prvky matice obrazu nabývají značně rozdílných hodnot – tedy když jeden nebo více okrajů obrázku jsou tmavé a ostatní světlé.




Obrázek 4.6: Norma rezidua a norma chyby LSQR pro příklad 3.

## 4.2 Časová náročnost

Kromě kvality výsledného zaostřeného obrazu a počtu iterací k tomu potřebných je žádoucí analyzovat i časovou náročnost jednotlivých algoritmů. Porovnáme algoritmy LSQR, GMRES a RRGMRES při hledání aproximovaného řešení příkladu 1 s reflexivní okrajovou podmínkou.

Tabulka 4.1 a graf 4.7 ukazují, že s každou novou iterací se zvětšuje rozdíl mezi jednotlivými metodami. Zatímco při prvních deseti iteracích je čas potřebný k výpočtu srovnatelný, pro 100 iterací vyžaduje RRGMRES dvakrát více času než LSQR. Tento rozdíl se v dalších iteracích ještě zvyšoval.

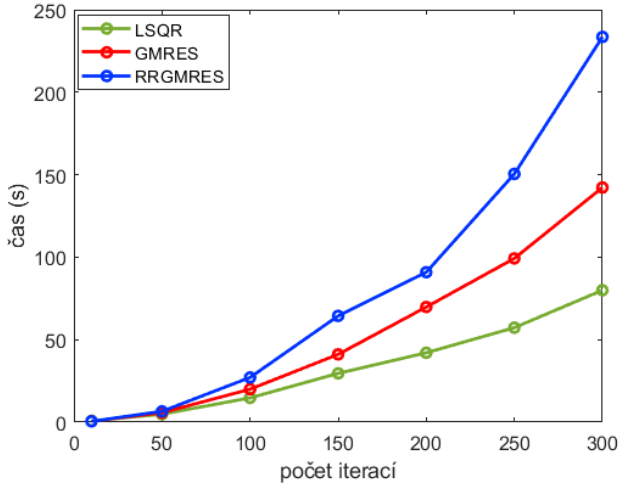
Jeden z důvodů, proč tomu tak je, je částečně rozebrán ve druhé kapitole. U LSQR se využívá krátkých rekurencí, neboť nový vektor báze Krylovova prostoru se stabilně počítá ze dvou vektorů. Naproti tomu GMRES a RRGMRES využívají ke konstrukci báze Arnoldiho algoritmus, který k výpočtu používá všechny předchozí prvky báze. To znamená, že zatímco u LSQR je čas potřebný pro výpočet každé nové aproximace přibližně stejný, u metod pracujících s Arnoldiho algoritmem vyžaduje každý nový vektor báze více výpočetního času.

Poměrně překvapivý rozdíl je mezi GMRES a RRGMRES, neboť algoritmy pracují téměř totožně, tedy časová náročnost by měla být přibližně stejná. Tento fakt pravděpodobně plyne z rozdílné implementace metod. V případě GMRES jsme řešení projektovaných úloh hledali s pomocí vestavěné funkce '\' pro řešení problému nejmenších čtverců $\mathbf{Mx} \approx \mathbf{y}$. Tato funkce je implementována tak, že konkrétní algoritmus vybírá na základě struktury matice $\mathbf{M}$. Například pokud je matice hermitovská, používá LU rozklad, pokud je obdélníková, automaticky volí QR rozklad. Implementace rozkladů matice bude rovněž efektivnější než naše „obyčejné“ využití Givensových rotací u RRGMRES.

Důležité je však zmínit, že při výběru vhodné metody záleží především na počtu iterací. Pokud úloha vyžaduje pouze pár desítek iterací (což při úloze rozmazání obrazu typicky bývá), rozdíl mezi časovou náročností jednotlivých metod je zanedbatelný. GMRES a RRGMRES ze začátku nemají příliš dlouhé rekurence, převládá tedy výhoda pouze jednoho násobení matice s vektorem oproti LSQR, kde v každé iteraci takto násobíme dvakrát.

| | **10** | **50** | **100** | **150** | **200** | **250** | **300** |
|---|---|---|---|---|---|---|---|
| **LSQR** | 0.79 | 4.86 | 14.72 | 29.58 | 42.06 | 57.29 | 79.96 |
| **GMRES** | 0.58 | 5.73 | 19.97 | 41.10 | 69.80 | 99.25 | 142.26 |
| **RRGMRES** | 0.51 | 6.57 | 27.09 | 64.41 | 90.87 | 150.47 | 233.49 |

Tabulka 4.1: Časová náročnost regularizace v závislosti na počtu iterací a algoritmu (v sekundách). Hodnoty odpovídají aplikaci LSQR, GMRES a RRGMRES na příklad 1 s reflexivní okrajovou podmínkou.




Obrázek 4.7: Grafické znázornění tabulky 4.1.

## 4.3 Princip diskrepance

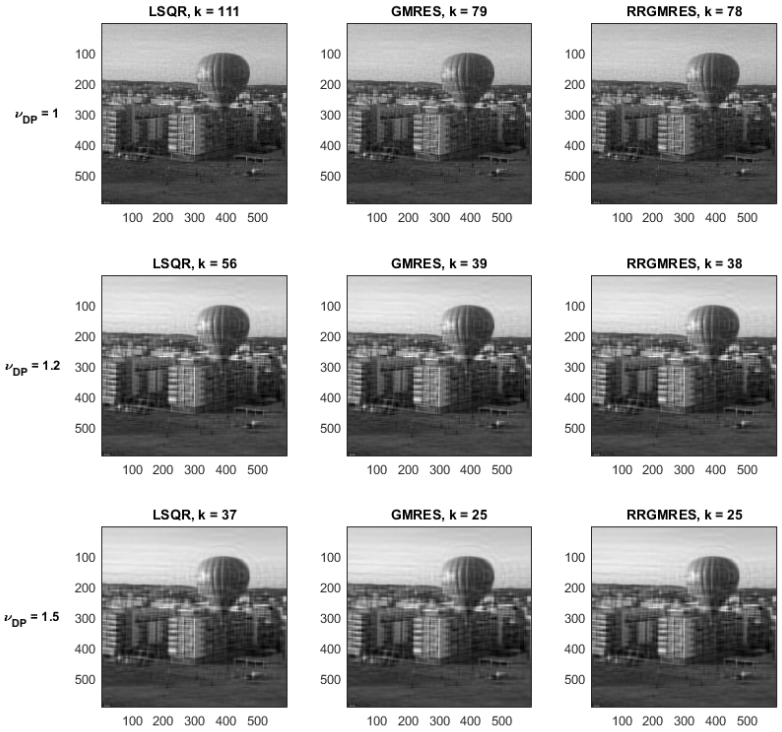
V této části se zaměříme na hledání regularizačního parametru $k$. V sekcích 2.4, 3.3 jsme se seznámili s principem diskrepance, jenž parametr $k$ hledá na základě odhadu normy šumu $q$ a „bezpečnostního“ faktoru $\nu_{DP} \geq 1$. Podívejme

se nejprve na hodnoty parametru $k$ pro příklad 1 při využití principu diskrepance s $\nu_{DP} = 1$. V našich experimentech uvažujeme $q = \|\mathbf{e}\|$.

| | **0.1** | **0.01** | **0.001** |
|---|---|---|---|
| **LSQR** | 6 | 25 | 111 |
| **GMRES** | 5 | 19 | 79 |
| **RRGMRES** | 4 | 18 | 78 |

Tabulka 4.2: Regularizační parametr $k$ pro příklad 1 v závislosti na využité metodě a hladině šumu.

Tabulka 4.2 ukazuje, že při nižší hladině šumu dostáváme z principu diskrepance vyšší $k$. Zatímco pro $\mu = 10^{-1}$ nacházíme řešení s využitím metody LSQR během 6 iterací, pro $\mu = 10^{-3}$ k tomu potřebujeme 111 iterací. Nabízí se však otázka, zda takové množství iterací je skutečně potřebné pro získání vhodné aproximace. Vyzkoušejme proto i jiné nastavené $\nu_{DP}$, jež zajistí dřívější zastavení procesu hledání aproximovaného řešení.




Obrázek 4.8: Řešení příkladu 1 s $\mu = 0.1$ nalezené při $k$-té iteraci. K výběru iterace byl použit princip diskrepance s různými parametry $\nu_{DP}$.

Obrázek 4.8 ukazuje jednotlivé aproximované zaostřené obrázky v závislosti na vybrané metodě a bezpečnostním faktoru $\nu_{DP}$ pro $\mu = 10^{-3}$. Na horním řádku jsou obrázky nejvíce ostré a zároveň bez patrného šumu. Tento řádek odpovídá volbě $\nu_{DP} = 1$. S vyšší hodnotou bezpečnostního faktoru mají výsledné aproximace nižší ostrost. Pokud však řešení úlohy nevyžaduje nejjemnější detaily, je možné volbou většího $\nu_{DP}$ optimalizovat časovou náročnost.

V příkladu 1 s nižší hladinou šumu by bylo vhodnější zvolit $\nu_{DP} = 1{,}2$, které redukuje počet iterací na téměř polovinu při současném zachování dostatečné ostrosti.

## 4.4 Porovnání blokových a neblokových metod

V následujících experimentech se zaměříme na porovnání blokových a neblokových metod. Provedeme experiment se zaostřováním obrázku s různými hladinami šumu a na závěr analyzujeme úlohu s barevným obrázkem.

### 4.4.1 Obrázek ve stupních šedi s různým zašuměním

V první části využijeme příkladu `tomo` z Regularization Tools, který vytváří jednoduchou 2D tomografickou testovací úlohu. Vygenerovaný obrázek není barevný, řešení této úlohy tedy spočívá v řešení soustavy s jednou pravou stranou. Můžeme však úlohu pozměnit a k vygenerovanému obrázku přičíst šum s různými $\mu$.

*Příklad* 4. Vygenerujeme pomocí funkce `tomo` matici $\mathbf{A} \in \mathbb{R}^{400\times 400}$ a vektory $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{400}$. Vytvoříme matici $\mathbf{B} \in \mathbb{R}^{400\times 3}$ s vektory $\mathbf{b} + \mathbf{e}_i$ pro $i = 1, 2, 3$, kde $\mathbf{e}_i$ je náhodně generovaný gaussovský šum s $\mu = 10^k$ pro $k = -4, -3, -2$ (viz obrázek 4.9).

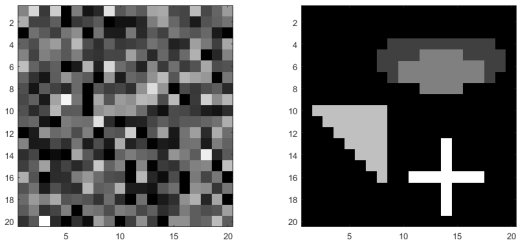
Příklad 4 nyní odpovídá úloze (3.3) s $\mathtt{p} = 3$. Můžeme tedy aplikovat blokové metody představené v kapitole 3. Začněme s metodou LSQR a blokovým LSQR (BLSQR). Z grafů na obrázku 4.10 je patrné, že pro danou úlohu klesá norma rezidua při využití BLSQR rychleji. Norma chyby v počátečních iteracích také nabývá nižších hodnot. Opačný trend nastává až v pozdějších iteracích, které pro nás nejsou relevantní.

Zajímavější rozdíl je u optimálních hodnot regularizačního parametru $k$ dle principu diskrepance. Pokud použijeme neblokový LSQR, musíme řešit jednotlivé úlohy separátně, a jelikož každá z nich má jinou hladinu šumu, lze očekávat různé hodnoty parametrů $k_1, k_2, k_3$. Jak ukazuje tabulka 4.3, rozdíl mezi hodnotami je v tomto případě značný. Zároveň je patrné, že využití blokové metody zde přináší výraznou časovou úsporu. Z vlastností principu diskrepance plyne, že B-LSQR obecně nabízí efektivnější řešení jedné úlohy zaostřování obrazu s různými hladinami šumu [1].

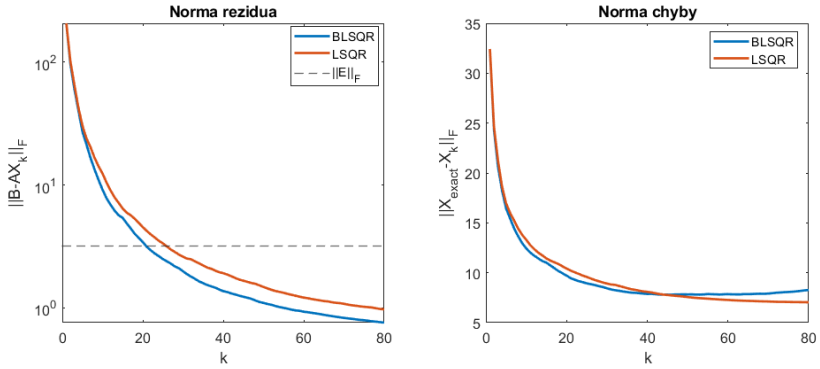
| | $k_1$ | $k_2$ | $k_3$ | $k_B$ |
|---|---|---|---|---|
| **LSQR** | 20 | 102 | 443 | 22 |
| **GMRES** | 377 | 397 | 399 | 127 |

Tabulka 4.3: Regularizační parametr dle principu diskrepance. $k_1, k_2, k_3$ značí výsledné parametry při použití neblokové metody, $k_B$ značí výsledek blokového principu diskrepance.

[1] V případném rozšíření práce by bylo užitečné otestovat i další zastavovací kritéria.

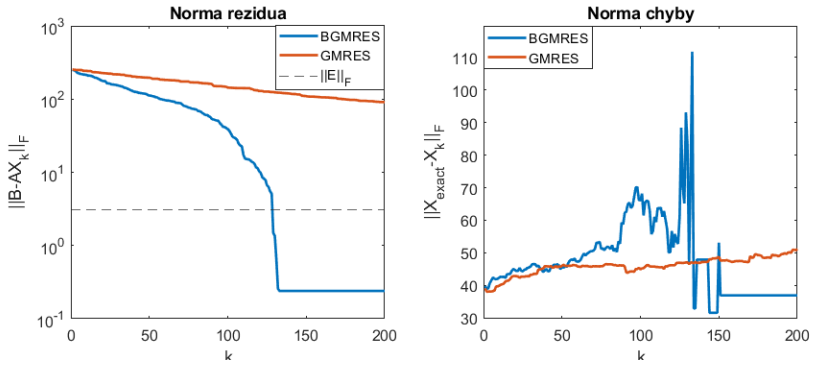


Obrázek 4.9: Vlevo obrázek odpovídající **b**, vpravo přesné řešení **x** pro úlohu `tomo`, oba velikosti $20 \times 20$ pixelů.




Obrázek 4.10: Norma rezidua a norma chyby LSQR a BLSQR pro příklad 4.

Výsledky pro blokový GMRES (BGMRES) naopak nevypadají moc příznivě. Norma rezidua sice klesá podstatně rychleji, nás ale zajímá především norma chyby, která prezentuje kvalitu aproximovaného řešení. Norma chyby se chová nepředvídatelně, nejedná se o semikonvergenci (viz obrázek 4.11). Optimálním regularizačním parametrem $k_B$ je dle principu diskrepance $k_B = 127$, což odpovídá šestinásobku iterací oproti BLSQR. Navíc norma chyby při této iteraci je osmkrát vyšší než u BLSQR.




Obrázek 4.11: Norma rezidua a norma chyby GMRES a BGMRES pro příklad 4.

V tento moment se ukazuje metoda BGMRES jako nepříliš vhodná pro inverzní úlohy a regularizaci. Podívejme se však na zaostřování barevného obrazu se shodným zašuměním všech kanálů.

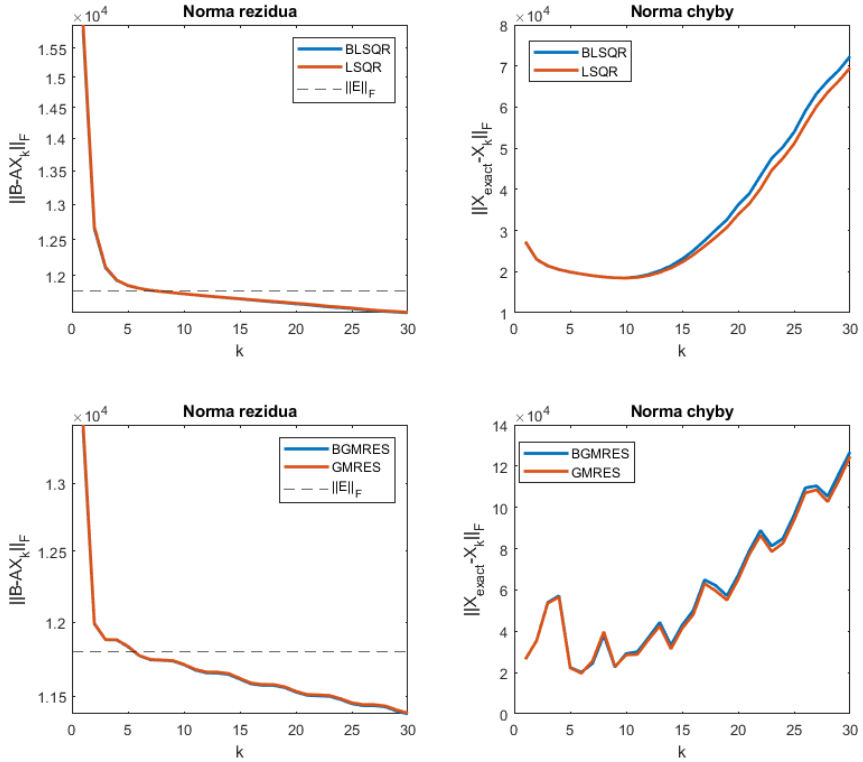
### 4.4.2 Barevný obrázek

V závěrečném experimentu se vrátíme k příkladu 1, tentokrát s barevným obrázkem. Pro hledání zaostřeného řešení použijeme metody LSQR, GMRES, BLSQR a BGMRES.

Grafy na obrázku 4.12 ukazují, že bloková varianta GMRES funguje v tomto experimentu poměrně dobře. Norma chyby sice vykazuje jisté výkyvy, ale její chování přibližně připomíná semikonvergenci. Pro optimální regularizační parametr $k_B$ (dle principu diskrepance) je $\|\mathbf{X}_{exact} - \mathbf{X}_k\|_F = 20720$, což představuje v porovnání s normou chyby u BLSQR pouze 10% nárůst a oproti GMRES rozdíl menší než 0,25 % (viz tabulka 4.4). Tato pozorování ukazují potenciál využití BGMRES v dalších inverzních úlohách a regularizaci.

| | LSQR | BLSQR | GMRES | BGMRES |
|---|---|---|---|---|
| $\|\mathbf{X}_{exact} - \mathbf{X}_k\|_F$ | 18710 | 18726 | 20066 | 20720 |

Tabulka 4.4: Norma chyby při iteraci $k$ odpovídající optimálnímu regularizačnímu parametru v závislosti na použité metodě.




Obrázek 4.12: Norma rezidua a norma chyby LSQR, BLSQR, GMRES a BGMRES pro příklad 1 s barevným obrázkem.

V této fázi by bylo namístě ještě ověřit časovou náročnost blokových algoritmů a porovnat ji s výsledky pro neblokové algoritmy. Jak ale ukázal experiment v sekci 4.2, bez znalosti přesných struktur zabudovaných algoritmů v Matlabu není možné porovnávat mezi sebou algoritmy, které používají pouze zabudované

funkce, a algoritmy, které pracují s námi vytvořenými funkcemi. Pro další prozkoumání by bylo potřeba vytvořit alternativy k zabudovaným funkcím, jež by nám umožnily komplexní vhled do struktury kódu.

# Závěr

Cílem práce bylo detailně popsat úlohy zaostřování obrazu a analyzovat krylovovské regularizační metody aplikované na tyto úlohy. Na úvod jsme představili maticový a obecný lineární model rozmazání obrazu a detailně popsali jednotlivé komponenty a parametry těchto modelů. Na základě vlastností matice rozmazání a přítomnosti šumu v obrazové informaci jsme dokázali, proč je pro úlohu zaostřování obrazu potřeba využít regularizačních metod.

V druhé kapitole jsme představili známé krylovovské regularizační metody. Rozebrali jsme základní kroky algoritmů LSQR, GMRES, RRGMRES a popsali jejich výhody a nevýhody. Zároveň jsme ukázali, že norma rezidua je pro zmíněné algoritmy nerostoucí a norma chyby během prvních iterací klesá a následně roste. Z těchto důvodů je při řešení úloh zaostřování obrazu nutné využít zastavovacích kritérií, které na základě jistých parametrů vyberou iteraci s co nejmenší normou chyby. Z těchto kritérií jsme vybrali *princip diskrepance*, který vhodný regularizační parametr volí dle odhadu normy chyby,

Třetí kapitola byla věnována méně známým blokovým krylovovským metodám. Definovali jsme blokový Krylovův prostor, popsali jeho základní vlastnosti a problémy spojené s dimenzí prostoru. Následně jsme představili blokový LSQR a ukázali odlišnosti při konstrukci ortonormální báze prostoru spočívající ve využití QR rozkladu. Obdobně jsme studovali blokový GMRES a diskutovali úpravu na blokový RRGMRES. Zobecnili jsme princip diskrepance pro blokové metody.

Poslední část byla věnována numerickým experimentům. Ukázali jsme, že volba okrajové podmínky ovlivňuje normu rezidua a také normu chyby v počátečních iteracích krylovovských metod. Nejhorších výsledků dosáhla nulová okrajová podmínka. Zároveň však rozdíl při volbě okrajových podmínek závisí na vstupním obrázku. Analyzovali jsme vyšší časovou náročnost pro metody pracující s dlouhými rekurencemi.

V dalším experimentu jsme prozkoumali princip diskrepance pro různá nastavení bezpečnostního faktoru. Ukázali jsme, že s nižší hladinou šumu roste hodnota výsledného regularizačního parametru. Při vyšších hodnotách bezpečnostního faktoru $\nu_{DP}$ dochází k větší regularizaci, což má pozitivní dopad na časovou náročnost, ale zároveň výsledný zaostřený obrázek neobsahuje dostatek detailů. Proto je potřeba volit $\nu_{DP}$ individuálně pro konkrétní typ úlohy.

Zbylé experimenty byly věnovány porovnání blokových a neblokových metod. V první fázi jsme se zaměřili na úlohu s obrázkem ve stupních šedi, ke kterému byl přičten šum s různými hladinami. Porovnali jsme řešení úlohy jakožto řešení separátních problémů pomocí neblokových krylovovských regularizačních metod a řešení úlohy pomocí blokových metod. Využití blokového LSQR s principem diskrepance zde přináší významnou časovou úsporu při zachování nízké normy chyby. V druhé fázi jsme se pokusili řešit úlohu zaostřování obrazu s barevným obrázkem, kde jeden sloupec matic $\mathbf{B}, \mathbf{X}$ odpovídá jednomu barevnému kanálu z RGB. Zde jsme zaznamenali poměrně vyrovnané výsledky mezi blokovými a neblokovými metodami.

# Seznam použité literatury


[1] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring images: Matrices, Spectra and Filtering, Fundamentals of Alghoritms.* SIAM, Philadelphia, 2006.

[2] P. C. Hansen, *Discrete Inverse Problems: Insight and Algorithms.* SIAM, 2010.

[3] J. Chung, S. Knepper, and J. G. Nagy, "Large-Scale Inverse Problems in Imaging," *Handbook of Mathematical Methods in Imaging*, pp. 43–86, 2011.

[4] J. Liesen and Z. Strakoš, *Krylov Subspace Methods: Principles and Analysis.* OUP Oxford, 2013.

[5] S. Gazzola and M. S. Landman, "Krylov methods for inverse problems: Surveying classical, and introducing new, algorithmic approaches," *Surveys for Applied Mathematics and Mechanics*, vol. 43, no. 4, 2020.

[6] A. Alqahtani, S. Gazzola, L. Reichel, and G. Rodriguez, "On the block Lanczos and block Golub–Kahan reduction methods applied to discrete ill-posed problems," *Numerical Linear Algebra with Applications*, vol. 28, p. e2376, 2021.

[7] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM Trans. Math. Software*, no. 8, pp. 49–51, 1982.

[8] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[9] A. Neuman, L. Reichel, and H. Sadok, "Implementations of range restricted iterative methods for linear discrete ill-posed problems," *Linear Algebra and its Applications*, vol. 436, pp. 3974–3990, 2012.

[10] S. Karimi and F. Toutounian, "The block least squares method for solving nonsymmetric linear systems with multiple right-hand sides," *Applied Mathematics and Computation*, vol. 177, no. 2, pp. 852–862, 2006.

[11] V. Simoncini and E. Gallopoulos, "Convergence properties of block GMRES and matrix polynomials," *Linear Algebra and its Applications*, vol. 247, pp. 97–119, 1996.

[12] A. H. Bentbib, M. El Guide, K. Jbilou, E. Onuwor, and L. Reichel, "Solution methods for linear discrete ill-posed problems for color image restoration," *BIT Numerical Mathematics*, vol. 58, pp. 555–567, 2018.

[13] A. Hossain and S. A. Sajib, "Classification of Image using Convolutional Neural Network (CNN)," *Global Journal of Computer Science and Technology*, vol. 19, no. 2, pp. 12–18, 2019.

[14] W.-H. Steeb, *Kronecker Product of Matrices and Applications.* BI-Wissenschaftsverlag, 1991.

[15] S. Gazzola, P. C. Hansen, and J. G. Nagy, "IR tools: A MATLAB Package of Iterative Regularization Methods and Large-Scale Test Problems," *Numerical Algorithms*, vol. 81, pp. 773–811, 2019.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, 1996.

[17] W. D. Penny, *Lecture notes in Signal Processing Course.* UCL Queen Square Institute of Neurology, University College London, 2020.

[18] L. Barto and D. Stanovský, *Počítačová algebra.* MatfyzPress, 2017.

[19] D. Lee, "Fast Multiplication of a Recursive Block Toeplitz Matrix by a Vector and its Application," *Journal of Complexity*, vol. 2, pp. 295–305, 1986.

[20] E. O. Brigham, *The Fast Fourier Transform and Its Applications.* Prentice-Hall, Inc., 1988.

[21] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *Journal of the Society for Industrial and Applied Mathematics*, vol. 2, no. 2, pp. 205–224, 1965.

[22] W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," *Quarterly of Applied Mathematics*, vol. 9, no. 1, pp. 17–29, 1951.

[23] V. A. Morozov, *Methods for Solving Incorrectly Posed Problems.* Springer Verlag, 1984.

[24] M. H. Gutknecht and T. Schmelzer, "The block grade of a block krylov space," *Linear Algebra and its applications*, pp. 174–186, 2009.

[25] Q. Niu and L. Lu, "Deflated block Krylov subspace methods for large scale eigenvalue problems," *Journal of Computational and Applied Mathematics*, vol. 234, pp. 636–648, 2010.

[26] M. H. Gutknecht, "Block Krylov space methods for linear systems with multiple right-hand sides," *Modern mathematical models, methods and algorithms for real world systems*, pp. 420–447, 2007.

[27] M. Rapavý, "Globálne krylovovské metódy pre riešenie lineárnych algebraických problémov s maticovým pozorovaním," *Diplomová práce, Univerzita Karlova*, 2019.

[28] A. S. Householder, "A class of methods for inverting matrices," *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 2, pp. 189–195, 1958.

[29] D. Boley and G. Golub, "The lanczos-arnoldi algorithm and controllability," *Systems & Control Letters*, vol. 4, no. 6, pp. 317–324, 1984.

[30] L. Onisk, L. Reichel, and H. Sadok, "Numerical considerations of block GMRES methods when applied to linear discrete ill-posed problems," *Journal of Computational and Applied Mathematics*, vol. 430, 2023.

[31] A. Buccini, L. Onisk, and L. Reichel, "Range restricted iterative methods for linear discrete ill-posed problems," *Electronic Transactions on Numerical Analysis*, vol. 58, pp. 348–377, 2023.

[32] S. Karimi and F. Toutounian, "On the convergence of the Bl-LSQR algorithm for solving matrix equations," *International Journal of Compuer Mathematics*, vol. 88, pp. 171–182, 2011.

[33] P. C. Hansen, "Regularization Tools Version 4.0 for MATLAB 7.3," *Numerical Algorithms*, vol. 46, pp. 189–194, 2007.

[34] C. Moler and J. Little, "A History of MATLAB," *Proceedings of the ACM on Programming Languages*, vol. 4, no. 81, pp. 1–67, 2020.

[35] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Society for Industrial and Applied Mathematics, 1999.

# Seznam algoritmů


1 FFT . . . . . 16
2 Rychlé násobení polynomů . . . . . 16
3 Rychlé násobení toeplitzovské blokové matice s vektorem . . . . . 17

4 LSQR . . . . . 19
5 GMRES . . . . . 21
6 RRGMRES . . . . . 22

7 Blokový LSQR . . . . . 28
8 Blokový GMRES . . . . . 30