Numerical Optimization and Multigrid Computational Methods with Applications

Tao Hong

Numerical Optimization and Multigrid Computational Methods with Applications

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Tao Hong

Submitted to the Senate
of the Technion — Israel Institute of Technology
Elul 5781 Haifa August 2021

This research was carried out under the supervision of Prof. Irad Yavneh and Prof. Michael Zibulevsky, in the Faculty of Computer Science.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author's doctoral research period, the most up-to-date versions of which being:

Tao Hong, Xiao Li, Zhihui Zhu, and Qiuwei Li. Optimized structured sparse sensing matrices for compressive sensing. *Signal Processing*, 159:119–129, 2019.

Tao Hong, Yaniv Romano, and Michael Elad. Acceleration of RED via vector extrapolation. *Journal of Visual Communication and Image Representation*, 63:102575, 2019.

Tao Hong, Irad Yavneh, and Michael Zibulevsky. Solving RED with weighted proximal methods. *IEEE Signal Processing Letters*, 27:501–505, 2020.

Tao Hong and Zhihui Zhu. An efficient method for robust projection matrix design. *Signal Processing*, 143:200–210, 2018.

Tao Hong and Zhihui Zhu. Online learning sensing matrix and sparsifying dictionary simultaneously for compressive sensing. *Signal Processing*, 153:188–196, 2018.

An additional paper, **Tao Hong** and Irad Yavneh, "On adapting Nesterov's scheme to accelerate iterative methods for linear problems" is under consideration for the journal *Numerical Linear Algebra with Applications*, after a "minor revision" decision in the first round.

ACKNOWLEDGEMENTS

I would like to thank my advisors Prof. Irad Yavneh and Prof. Michael Zibulevsky for their helpful guidance and patience during the course of my PhD studies. Irad's and Michael's optimism, openness, passion, and clarity in research teach me how to overcome the difficulties I have met in research and daily life. Working with Irad and Michael is truly enjoyable and I hope to keep working with you in the future. Special thanks to Irad who not only teaches me knowledge but also guides me to think critically. Moreover, I really want to share one sentence learnt from Irad—*If there is a question you cannot answer, there must be a sub-question you can answer and your first task is to find it*—which helps me to realize the importance of understanding and guides me the way to solve challenging problems in research.

Studying at Technion is a wonderful journey and I want to thank many friends and collaborators I have met. Special thanks to those with whom I had the privilege to learn and to collaborate: Sanketh Vedula, Prof. Vardan Papyan, Prof. Yaniv Romano, and Prof. Michael Elad—I hope we can continue to work together. I also want to thank all of my other friends for their continued encouragement and support: Xiao, Zhihui, Shuo, Qiucheng and everyone else who helped me along the way. Special thanks to Prof. Michael Unser who hosted me during my visit to EPFL. Working with the members of the Biomedical Imaging Group at EPFL is a wonderful journey and I hope we can continue to work together.

Last, but not least, I would like to thank my parents and young sister for their unconditional love and support throughout these years. This thesis is dedicated to them.

The generous financial help of the Technion is gratefully acknowledged.

Contents

Abstract

Optimization problems can be found naturally in many different fields. In many engineering problems, we get a thirst to organize things in their best way which defines an optimization problem of a certain formulation. Numerical optimization focuses on the discovery of fast numerical algorithms for such an optimization problem. Multigrid computational methods study the way to solve differential problems efficiently by using a hierarchy of discretizations. In this thesis, we mainly study numerical optimization and multigrid computational methods as well as their applications.

Regularization by denoising (RED) is a recently developed framework to construct advanced priors through state-of-the-art image denosising algorithms. We begin with the study of applying vector extrapolation to accelerating the existing solvers in RED. Following, we propose a general framework named weighted proximal methods (WPMs) for RED. We show that the previous two solvers in RED, namely the fixed point and accelerated proximal methods, are two special cases of WPMs. By setting a simple weighting, we show that WPMs converge faster than previous solvers.

In the second part of this thesis, we first adapt Nesterov's scheme to accelerate iterative methods for linear problems. In this work, we propose an analytic solution to obtain the optimal parameter inside Nesterov's scheme and discuss the robustness of Nesterov's scheme. Our second work in this part merges multigrid optimization with sequential subspace optimization formulating a new accelerated scheme, dubbed SESOP-MG, which inherits the merits of these two methods. Additionally, we study the asymptotic convergence factor of the two-grid version of SESOP-MG for linear problems and propose a fixed-stepsize version for linear problems to save computation of SESOP-MG further.

Compressive sensing (CS) is a signal processing technique used to acquire and reconstruct signals of interest efficiently. Sparsity and incoherence are two conditions in which the recovery in CS is possible. Sparsity means that the signal can be represented sparsely in some domain which can be realized by learning a dictionary or choosing a prescribed one for particular signals. Incoherence is satisfied by designing a sensing matrix which also plays a role in acquiring signals. Our first work in this direction presents a model to learn the dictionary and sensing matrix simultaneously. The resulting CS system yields a higher signal reconstruction accuracy than previous CS systems. Our second work considers the efficiency of acquiring signals with a structured sensing matrix consisting of a sparse matrix and a dense matrix which allows fast implementation.

List of Figures

List of Tables

Chapter 1

Introduction

Computational methods naturally find applications in all fields of engineering and sciences, e.g., biological, physical, social, and business, etc. Since the growth in computing power has revolutionized in the past decades, researchers can build progressively more accurate models, to better understand the real world. These models are large-scale and complicated, requiring efficient computational methods even for today's powerful computers. This thesis concentrates on efficient computational methods—particularly numerical optimization and multigrid computational methods—and their applications to signal processing and computational imaging. The main contribution of this thesis is divided into three parts: (i) two efficient solvers for regularization by denoising (RED); (ii) two accelerated schemes for optimization and linear equations; (iii) the design of robust compressive sensing (CS) systems.

1.1 Two Efficient Solvers for Regularization by Denoising (RED)

1.1.1 First Contribution - Acceleration of RED via Vector Extrapolation

Inverse problems are one of the major problems in signal processing and its mission is to recover signals of interest from some degraded measurements. By modelling the noisy and degraded measurements $\boldsymbol{y} \in \mathbb{R}^M$ as $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$, where $\boldsymbol{H} \in \mathbb{R}^{M \times N}$ represents the degraded operator and $\boldsymbol{n} \in \mathbb{R}^M$ is the additive noise, our target is to recover $\boldsymbol{x}$ given $\boldsymbol{y}$. Typically, $\boldsymbol{n}$ is assumed to be the i.i.d. white Gaussian noise with mean 0 and variance σ^2. Using the maximum a posterior probability (MAP) and Bayes's rule, the reconstruction of $\boldsymbol{x}$ is equivalent to solving the following minimization problem

$$\min_{\boldsymbol{x}} \underbrace{\frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|_2^2}_{F(\boldsymbol{x})} + \lambda R(\boldsymbol{x}),$$

where $F(\boldsymbol{x})$ refers to the data-fidelity term measuring the discrepancy between the estimated signal and the measurements and $R(\cdot)$ refers to the prior used to describe the distribution of $\boldsymbol{x}$, e.g., total variation [1], wavelets [2], dictionary learning [3], or convolutional neural network [4]. $\lambda > 0$ is a trade-off parameter to balance $F(\boldsymbol{x})$ and $R(\cdot)$. We note that finding an effective $R(\boldsymbol{x})$ to describe the distribution of signals of interest is a core subject in inverse problems. Recently, Romano et al. introduced a general framework called regularization by denoising (RED) [5] to construct priors through state-of-the-art image denoising algorithms that allows us to adapt modern denoisers to establish priors. In RED, Romano et al. showed

that solving inverse problems amounts to an iterated denoising process. Since the complexity of denoising algorithms is generally high, RED may lead to an overall slow algorithm hindering practical use. Our first contribution proposes an accelerated technique based on vector extrapolation (VE) to speed-up existing RED solvers [6].

1.1.2 Second Contribution - Solving RED with Weighted Proximal Methods

By using VE to speed-up the solvers in RED, one can recover an image by calling the denoisers $30-50$ times. However, it is still slow for realistic use. Our second contribution presents a general framework called weighted proximal methods (WPMs) for RED [7]. We first show that two recently introduced RED solvers (the fixed point and accelerated proximal gradient methods) are particular cases of WPMs. Then we verify by numerical experiments that slightly more sophisticated variants of WPMs can lead to reduced run times for RED by requiring a significantly smaller number of calls, typically $10-20$ times, to the denoiser.

1.2 Two Accelerated Schemes for Optimization and Linear Equations

1.2.1 Third Contribution - On Adapting Nesterov's Scheme to Accelerate Iterative Methods for Linear Problems

In many scientific applications, we need to solve linear systems of equations,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f},$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a sparse, large-scale, ill-conditioned matrix. In practice, the size of $\boldsymbol{A}$ is often so large that direct methods are inapplicable and the use of iterative methods becomes more attractive, e.g., Jacobi, Gauss-Seidel, and multigrid methods. Classic iterative methods may become inefficient for difficult problems, and we therefore combine them with acceleration techniques, e.g., Krylov subspace acceleration methods, to obtain efficient hybrid methods.

Our third contribution studies the adaption of Nesterov's well-known scheme to accelerating stationary iterative solvers for linear problems [8]. Compared with Krylov subspace acceleration methods, the proposed scheme requires more iterations, but it is trivial to implement and retains essentially the same computational cost as the unaccelerated method. An explicit formula for a fixed optimal parameter is derived in the case where the stationary iteration matrix has only real eigenvalues, based only on the smallest and largest eigenvalues. The fixed parameter and corresponding convergence factor are shown to maintain their optimality when the iteration matrix also has complex eigenvalues that are contained within an explicitly defined disk in the complex plane. A comparison to Chebyshev acceleration based on the same information of the smallest and largest real eigenvalues (dubbed Restricted Information Chebyshev acceleration) demonstrates that Nesterov's scheme is more robust in the sense that it remains optimal over a larger domain when the iteration matrix does have some complex eigenvalues.

1.2.2 Fourth Contribution - Merging Multigrid Optimization with SESOP

Multigrid (MG) methods are widely considered to be an efficient approach for solving elliptic partial differential equations (PDEs) and systems, as well as other problems which can be effectively represented

on a hierarchy of grids or levels [9, 10, 11, 12]. However, it is often challenging to design efficient stand-alone MG methods for difficult problems, and therefore MG methods are often used in combination with acceleration techniques (e.g., [13]). Our fourth contribution [14] of this thesis is to treat MG as an optimization framework and seek robust solution methods by merging this approach with so-called SEquential Subspace OPtimization (SESOP) [15], dubbed SESOP-MG. Our idea is to add the coarse grid correction (CGC) generated from MG methods to the search subspace of SESOP which contains preconditioned gradient and the search directions produced by the previous iterates, called history. Furthermore, we also study the asymptotic convergence factor (ACF) of the two-level version of SESOP-MG for quadratic optimization problems. Interestingly, we show that if the problem is quadratic and the subspace only contains three directions, namely preconditioned gradient, CGC, and one history, then we are able to use fixed stepsizes for each direction and thus eliminate the computation required for calculating the stepsizes at each iteration. Our numerical experiments demonstrate the effectiveness of such a merger and the relevance of our theoretic study.

1.3 The Design of Robust Compressive Sensing Systems

1.3.1 Fifth Contribution - Online Learning Sensing Matrix and Sparsifying Dictionary Simultaneously for Compressive Sensing

We call a signal $\boldsymbol{x} \in \mathbb{R}^N$ sparse if $\|\boldsymbol{x}\|_0 \ll N$ or $\boldsymbol{x}$ can be represented sparsely by a prescribed sparsifying dictionary $\boldsymbol{\Psi} \in \mathbb{R}^{N\times L}$, $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{\theta} + \boldsymbol{e}$ with $\|\boldsymbol{\theta}\|_0 \ll L$, where $\|\cdot\|_0$ denotes the number of non-zeros. $\boldsymbol{e} \neq 0$ refers to the sparse representation error. By defining a sensing matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ with $M < N$, the theory of CS claims that one can recover $\boldsymbol{x}$ uniquely from the measurements $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}$ if $\boldsymbol{x}$ is sparse and $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ satisfy the Restricted Isometry Property (RIP) [16]. In practice, many signals can be represented sparsely, e.g., natural images are sparse under wavelet transforms [2]. Furthermore, one can also learn a $\boldsymbol{\Psi}$ [17] to represent $\boldsymbol{x}$ sparsely. Note that the RIP is satisfied with high probability if $\boldsymbol{\Phi}$ is set to be a random matrix.

Our fifth contribution considers the problem of simultaneously learning the sensing matrix $\boldsymbol{\Phi}$ and sparsifying dictionary $\boldsymbol{\Psi}$ (SMSD) on a large training dataset yielding a CS system which has a higher signal reconstruction accuracy than choosing $\boldsymbol{\Phi}$ randomly and using a prescribed $\boldsymbol{\Psi}$. To address the formulated joint learning problem, we propose an online algorithm that consists of a closed-form solution to optimize $\boldsymbol{\Phi}$ and a stochastic method to learn $\boldsymbol{\Psi}$ on a large dataset. Moreover, we also study the convergence of our proposed method.

1.3.2 Sixth Contribution - Optimized Structured Sparse Sensing Matrices for Compressive Sensing

Our sixth contribution addresses the efficiency of acquiring signals in CS systems. We propose a robust structured sparse sensing matrix consisting of a sparse matrix with a few non-zero entries per row and a dense base matrix which allows fast implementation. The robust structured sparse sensing matrix is obtained through minimizing the distance between the Gram matrix of the equivalent dictionary $\boldsymbol{\Phi}\boldsymbol{\Psi}$, and the target Gram matrix to reduce the averaged mutual coherence (cf. (7.1)). Moreover, a regularization is added to enforce the robustness of the optimized structured sparse sensing matrix to the case $\boldsymbol{e} \neq 0$. An alternating minimization algorithm with a global sequence convergence analysis is presented for the formulated optimization problem.

1.4 Thesis Structure

The thesis is organized as follows. In Chapter 2, we present the use of vector extrapolation techniques to speed-up the existing solvers in RED. We introduce our second work in RED, weighted proximal methods, in Chapter 3. Chapters 2 and 3 formulate the first part of this thesis. In Chapter 4, we explore the adaption of Nesterov's scheme for accelerating iterative methods for linear problems. Chapter 5 describes the merger of multigrid optimization and SESOP as well as our theoretical study of this new accelerated scheme. These two chapters consist of the second part of this dissertation. In Chapter 6, we discuss the design of sensing matrix and sparsifying dictionary simultaneously for CS systems. Chapter 7 considers the efficiency of acquiring a signal that we propose a way to design a structured sparse sensing matrix. These two chapters compose the third part of this thesis. Finally, we conclude this thesis in Chapter 8 and show some interesting future directions.

1.5 Notation

This thesis is a summary of the author's publications, often treating different fields and applications. However, the use of notation throughout this document is consistent in that, in general, we use bold lowercase and uppercase letters to refer to vectors and matrices, respectively. The non-bold letters are used to denote scalar quantities. Moreover, the specific meaning of the use of notation in each chapter will be specified clearly as needed.

Chapter 2

Acceleration of RED via Vector Extrapolation

In this chapter, we report our first work about applying vector extrapolation to accelerating REgularization by Denoising (RED). This chapter is based on the following published paper.

- **Tao Hong**, Yaniv Romano, and Michael Elad, *Acceleration of RED via Vector Extrapolation,* Journal of Visual Communication and Image Representation, vol. 63, Aug. 2019.

2.1 Introduction

Inverse problems in imaging science address the reconstruction of clean images from their corrupted versions. The corruption can be a blur, loss of samples, downscale or a more complicated operator (e.g., CT and MRI), accompanied by a noise contamination. Roughly speaking, inverse problems are characterized by two main parts: the first is called the forward model, which formulates the relation between the noisy measurement and the desired signal, and the second is the prior, describing the log-probability of the destination signal.

In recent years, we have witnessed a massive advancement in a basic inverse problem referred to as image denoising [18, 19, 20, 21, 22, 23]. Indeed, recent work goes as far as speculating that the performance obtained by leading image denoising algorithms is getting very close to the possible ceiling [24, 25, 26]. This motivated researchers to seek ways to exploit this progress in order to address general inverse problems. Successful attempts, as in [27, 28, 29], suggested an exhaustive manual adaptation of existing denoising algorithms, or the priors used in them, to treat specific alternative missions. This line of work has a clear limitation, as it does not offer a flexible and general scheme for incorporating various image denoising achievements for tackling other advanced image processing tasks. This led to the following natural question: is it possible to suggest a general framework that uses the abundance of high-performance image denoising algorithms for addressing general inverse problems? Venkatakrishnan et al. gave a positive answer to this question, proposing a framework called Plug-and-Play Priors (P^3) method [30, 31, 32]. Formulating the inverse problem as an optimization task and handling it via the Alternating Direction Method of Multipliers (ADMM) scheme [33], P^3 shows that the whole problem is decomposed into a sequence of image denoising sub-problems, coupled with simpler computational steps. The P^3 scheme provides a constructive answer to the desire to use denoisers within inverse problems,

but it suffers from several key disadvantages: (a) P^3 does not define a clear objective function, since the regularization used is implicit; (b) Tuning the parameters in P^3 is extremely delicate; (c) since P^3 is tightly coupled with the ADMM, it has no flexibility with respect to the numerical scheme.

A novel framework named REgularization by Denoising (RED) [5] proposes an appealing alternative while overcoming all these flaws. The core idea in RED is the use of the given denoiser within an expression of regularization that generalizes a Laplacian smoothness term. The work in [5] carefully shows that the gradient of this regularization is in fact the denoising residual. This, in turn, leads to several iterative algorithms, all guaranteed to converge to the global minimum of the inverse problem's penalty function, while using a denoising step in each iteration.

The idea of using a state-of-the-art denoising algorithm for constructing an advanced prior for general inverse problems is very appealing because: (a) it enables using the vast progress in image denoising for solving challenging inverse problems as explained above; (b) it allows to usie the denoiser as a black-box. However, a fundamental problem is the high complexity of typical denoising algorithms, which are required to be activated many times in such a recovery process. Indeed, the evidence from the numerical experiments posed in [5] clearly exposes this problem, in all the three methods proposed, namely the steepest descent, the fixed-point (FP) strategy, and the ADMM scheme. Note that the FP method is parameter free and the most efficient among the three, and yet this approach too requires the activation of the denoising algorithms dozens of times for a completion of the recovery algorithm.

Our main contribution in this chapter is to address these difficulties by applying vector extrapolation (VE) [34, 35, 36] to accelerating the FP algorithm shown in [5]. Our simulations illustrate the effectiveness of VE for acceleration, saving more than 50% of the overall computations involved compared with the native FP method.

The rest of this chapter is organized as follows. We review RED and the FP method in Section 2.2. Section 2.3 recalls the VE acceleration idea. Several experiments on image deblurring and super-resolution, which follows the ones given in [5], are carried out to exam the performance of VE, and these are brought in Section 2.4. We conclude this chapter in Section 2.5.

2.2 REgularization by Denoising (RED)

This section reviews the framework of RED which uses denoising algorithms as image priors [5]. We also describe its original solver based on the Fixed-Point (FP) method.

2.2.1 Inverse Problems as Optimization Tasks

From an estimation point of view, the signal $\boldsymbol{x} \in \mathbb{R}^N$ is to be recovered from its measurements $\boldsymbol{y}$ using the posterior conditional probability $P(\boldsymbol{x}|\boldsymbol{y})$. Using maximum a posterior probability (MAP) and the Bayes rule, the estimation task is formulated as:

$$\boldsymbol{x}^*_{MAP} = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}|\boldsymbol{y}) = \arg\max_{\boldsymbol{x}} \frac{P(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{x})}{P(\boldsymbol{y})} = \arg\max_{\boldsymbol{x}} P(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} -\log\{P(\boldsymbol{y}|\boldsymbol{x})\} - \log P(\boldsymbol{x}).$$

The third equality is obtained by the fact that $P(\boldsymbol{y})$ does not depend on $\boldsymbol{x}$. The term $-\log\{P(\boldsymbol{y}|\boldsymbol{x})\}$ is known as the log-likelihood $\ell(\boldsymbol{y},\boldsymbol{x})$. A typical example is

$$\ell(\boldsymbol{y},\boldsymbol{x}) \triangleq -\log\{P(\boldsymbol{y}|\boldsymbol{x})\} = \frac{1}{2\sigma^2}\|\boldsymbol{H}\boldsymbol{x}-\boldsymbol{y}\|_2^2, \tag{2.1}$$

referring to the case $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}+\boldsymbol{n}$, where $\boldsymbol{H}$ is any linear degradation operator and $\boldsymbol{n}$ is a white mean zero Gaussian noise with variance σ^2. Now, we can write the MAP optimization problem as

$$\boldsymbol{x}^*_{MAP} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \ell(\boldsymbol{y},\boldsymbol{x}) + \alpha R(\boldsymbol{x}), \tag{2.2}$$

where $\alpha > 0$ is a trade-off parameter to balance $\ell(\boldsymbol{y},\boldsymbol{x})$ and $R(\boldsymbol{x})$. $R(\boldsymbol{x}) \triangleq -\log P(\boldsymbol{x})$ refers to the prior that describes the statistical nature of $\boldsymbol{x}$. This term is typically referred to as the regularization, as it is used to stabilize the inversion by emphasizing the features of the recovered signal. In the following, we will describe how RED activates denoising algorithms for composing $R(\boldsymbol{x})$. Note that (2.2) defines a wide family of inverse problems including, but not limited to, inpainting, deblurring, super-resolution, [37] and more.

2.2.2 RED and the Fixed-Point Method

Define $f(\boldsymbol{x})$ as an abstract and differentiable denoiser which admits a noisy image $\boldsymbol{x}$ and removes additive Gaussian noise from it with assuming a prescribed noise energy. RED suggests applying the following form as the prior:

$$R(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathcal{T}}\left(\boldsymbol{x}-f(\boldsymbol{x})\right), \tag{2.3}$$

where $\mathcal{T}$ denotes the transpose operator. The term $\boldsymbol{x}^{\mathcal{T}}\left(\boldsymbol{x}-f(\boldsymbol{x})\right)$ is an image-adaptive Laplacian regularizer, which favors either a small residual $\boldsymbol{x}-f(\boldsymbol{x})$, or a small inner product between $\boldsymbol{x}$ and the residual [5]. Plugging (2.3) into (2.2) leads to the following minimization task:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} E(\boldsymbol{x}) \triangleq \ell(\boldsymbol{y},\boldsymbol{x}) + \alpha\frac{1}{2}\boldsymbol{x}^{\mathcal{T}}\left(\boldsymbol{x}-f(\boldsymbol{x})\right). \tag{2.4}$$

The prior $R(\boldsymbol{x})$ of RED is a convex function and differentiable if the following two conditions are met:

- Local homogeneity: For any scalar c arbitrarily close to 1, we have $f(c\boldsymbol{x}) = cf(\boldsymbol{x})$.
- Strong passivity: The Jacobian $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ is stable in the sense that its spectral radius is upper bounded by one, $\rho(\nabla_{\boldsymbol{x}} f(\boldsymbol{x})) \leq 1$.

With these two conditions, the gradient of $E(\boldsymbol{x})$ is given by

$$\nabla_{\boldsymbol{x}} E(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}\ell(\boldsymbol{y},\boldsymbol{x}) + \alpha\left(\boldsymbol{x}-f(\boldsymbol{x})\right). \tag{2.5}$$

As discussed experimentally and theoretically in [5, Section 3.2], many state-of-the-art denoising algorithms satisfy the aforementioned two conditions, and thus the gradient of (2.4) is simply evaluated through (2.5). As a consequence, $E(\boldsymbol{x})$ in (2.4) is a convex function if $\ell(\boldsymbol{x},\boldsymbol{y})$ is convex, such as in the case of (2.1). In such cases any gradient-based algorithm can be utilized to address (2.4) leading to its global minimum.

Note that evaluating the gradient of $E(\boldsymbol{x})$ calls for one denoising activation, resulting in an expensive operation as the complexity of effective denoising algorithms is typically high. Because of the slow convergence speed of steepest descent and the high complexity of ADMM, the work reported in [5] suggested using the FP method to handle the minimization task posed in (2.4). The development of the FP method is rather simple, relying on the fact that the global minimum of (2.4) should satisfy the first-order optimality condition, i.e., $\nabla_{\boldsymbol{x}}\ell(\boldsymbol{y},\boldsymbol{x})+\alpha(\boldsymbol{x}-f(\boldsymbol{x}))=\boldsymbol{0}$. For the FP method, we use the following iterative formula to solve this equation:

$$\nabla_{\boldsymbol{x}}\ell(\boldsymbol{y},\boldsymbol{x}_{k+1})+\alpha(\boldsymbol{x}_{k+1}-f(\boldsymbol{x}_k))=\boldsymbol{0}. \tag{2.6}$$

The explicit expression of (2.6) for $\ell(\boldsymbol{y},\boldsymbol{x})=\frac{1}{2\sigma^2}\|\boldsymbol{H}\boldsymbol{x}-\boldsymbol{y}\|_2^2$ is

$$\boldsymbol{x}_{k+1}=\left[\frac{1}{\sigma^2}\boldsymbol{H}^T\boldsymbol{H}+\alpha\boldsymbol{I}\right]^{-1}\left[\frac{1}{\sigma^2}\boldsymbol{H}^T\boldsymbol{y}+\alpha f(\boldsymbol{x}_k)\right], \tag{2.7}$$

where $\boldsymbol{I}$ represents the identity matrix. We note that the matrix inversion here is calculated in the Fourier domain for block-circulant $\boldsymbol{H}$ or using iterative methods for the more general cases. The convergence of the FP method is guaranteed since

$$\rho\left(\left[\frac{1}{\sigma^2}\boldsymbol{H}^T\boldsymbol{H}+\alpha\boldsymbol{I}\right]^{-1}\alpha\nabla_{\boldsymbol{x}}f(\boldsymbol{x}_k)\right)<1,$$

where $\rho(\cdot)$ denotes the spectral radius.

Although the FP method is more efficient than steepest descent and ADMM, it still needs hundreds of iterations, which means hundreds of denoising activations, to reach the desired minimum. This results in a high complexity algorithm which we aim to address in this work. In the next section, we introduce an accelerated technique called Vector Extrapolation (VE) to substantially reduce the amount of iterations in the FP method.

2.3 Proposed Method via Vector Extrapolation

We begin this section by introducing the philosophy of vector extrapolation (VE) in linear and nonlinear systems and then discuss three variants of VE, i.e., Minimal Polynomial Extrapolation (MPE), Reduced Rank Extrapolation (RRE) and Singular Value Decomposition Minimal Polynomial Extrapolation (SVD-MPE) [38, 36]. Efficient implementation of these three variants is also discussed. We refer the reader to [36] and the references therein to explore further the VE technique. We end this section by embedding VE in the FP method for RED, obtaining an acceleration of this scheme. Finally, we discuss the convergence and stability properties of VE.

2.3.1 Vector Extrapolation in Linear and Nonlinear Systems

Consider a vector set $\{\boldsymbol{x}_i\in\mathbb{R}^N\}$ generated via a linear process,

$$\boldsymbol{x}_{i+1}=\boldsymbol{A}\boldsymbol{x}_i+\boldsymbol{b},\ \ i=0,1,\cdots, \tag{2.8}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{b} \in \mathbb{R}^N$ and $\boldsymbol{x}_0$ is the initial vector. If $\rho(\boldsymbol{A}) < 1$, a limit point $\boldsymbol{x}^*$ exists, being the FP of (2.8), $\boldsymbol{x}^* = \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{b}$. We next describe how VE works on such linear systems [39]. Define the defect vector $\boldsymbol{e}_i$ as

$$\boldsymbol{e}_i = \boldsymbol{x}_i - \boldsymbol{x}^*, \;\; i = 0, 1, \cdots. \tag{2.9}$$

Subtracting $\boldsymbol{x}^*$ from both sides of (2.8) and using the fact that $\boldsymbol{x}^*$ is the FP, we have $\boldsymbol{e}_{i+1} = \boldsymbol{A}\boldsymbol{e}_i$ resulting in

$$\boldsymbol{e}_{i+1} = \boldsymbol{A}^{i+1}\boldsymbol{e}_0. \tag{2.10}$$

We define a new extrapolated vector $\boldsymbol{x}_{(m,\kappa)}$ as a weighted average of the form

$$\boldsymbol{x}_{(m,\kappa)} = \sum_{i=0}^{\kappa} \gamma_i \boldsymbol{x}_{m+i}, \tag{2.11}$$

where $\sum\limits_{i=0}^{\kappa} \gamma_i = 1$. Substituting (2.9) into (2.11) and using (2.10) and $\sum_{i=0}^{\kappa} \gamma_i = 1$, we have

$$\boldsymbol{x}_{(m,\kappa)} = \sum_{i=0}^{\kappa} \gamma_i \left(\boldsymbol{x}^* + \boldsymbol{e}_{m+i}\right) = \boldsymbol{x}^* + \sum_{i=0}^{\kappa} \gamma_i \boldsymbol{e}_{m+i} = \boldsymbol{x}^* + \sum_{i=0}^{\kappa} \gamma_i \boldsymbol{A}^i \boldsymbol{e}_m. \tag{2.12}$$

Note that the optimal $\{\gamma_i\}$ and κ should be chosen so as to force $\sum\limits_{i=0}^{\kappa} \gamma_i \boldsymbol{A}^i \boldsymbol{e}_m = 0$. This way, we attain the FP through only one extrapolation step.

More broadly speaking, given a nonzero matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ and an arbitrary nonzero vector $\boldsymbol{u} \in \mathbb{R}^N$, we can find a unique polynomial $P(z)$ with smallest degree to yield $P(\boldsymbol{B})\boldsymbol{u} = \boldsymbol{0}$. Such a $P(z)$ is called the minimal polynomial of $\boldsymbol{B}$ with respect to the vector $\boldsymbol{u}$. Notice that the zeros of $P(z)$ are the eigenvalues of $\boldsymbol{B}$. Thus, assume that the minimal polynomial of $\boldsymbol{A}$ with respect to $\boldsymbol{e}_m$ can be represented as

$$P(z) = \sum_{i=0}^{\kappa} c_i z^i, \;\; c_\kappa = 1 \tag{2.13}$$

resulting in $P(\boldsymbol{A})\boldsymbol{e}_m = \boldsymbol{0}$. So, we have

$$\sum_{i=0}^{\kappa} c_i \boldsymbol{A}^i \boldsymbol{e}_m = \sum_{i=0}^{\kappa} c_i \boldsymbol{e}_{m+i} = \boldsymbol{0}. \tag{2.14}$$

Multiplying both sides of (2.14) by $\boldsymbol{A}$ results in $\sum_{i=0}^{\kappa} c_i \boldsymbol{A}\boldsymbol{e}_{m+i} = \sum_{i=0}^{\kappa} c_i \boldsymbol{e}_{m+i+1} = \boldsymbol{0}$, and thus we receive

$$\sum_{i=0}^{\kappa} c_i \boldsymbol{e}_{m+i} = \sum_{i=0}^{\kappa} c_i \boldsymbol{e}_{m+i+1} = \boldsymbol{0}. \tag{2.15}$$

Subtracting these expressions gives

$$\sum_{i=0}^{\kappa} c_i \left(\boldsymbol{e}_{m+i+1} - \boldsymbol{e}_{m+i}\right) = \sum_{i=0}^{\kappa} c_i \left(\boldsymbol{x}_{m+i+1} - \boldsymbol{x}_{m+i}\right) = \sum_{i=0}^{\kappa} c_i \boldsymbol{u}_{m+i} = \boldsymbol{0}, \tag{2.16}$$

where $\boldsymbol{u}_{m+i} = \boldsymbol{x}_{m+i+1} - \boldsymbol{x}_{m+i}$, $i = 0, \cdots, \kappa$. This suggests that $\{c_i\}$ can be determined by solving the linear equations posed in (2.16). Once obtaining $\{c_i\}$, $\{\gamma_i\}$ are calculated by $\gamma_i = \frac{c_i}{\sum_{j=0}^{\kappa} c_j}$. Note that $\sum_{j=0}^{\kappa} c_j \neq 0$ if $\boldsymbol{I} - \boldsymbol{A}$ is not singular yielding $\sum_{j=0}^{\kappa} c_j = P(1) \neq 0$. Assuming κ is the degree of the minimal polynomial

of $\boldsymbol{A}$ with respect to $\boldsymbol{e}_m$, we can find a set of $\{\gamma_i\}$ to satisfy $\sum_{i=0}^{\kappa}\gamma_i = 1$ resulting in $\sum_{i=0}^{\kappa}\gamma_i \boldsymbol{x}_{m+i} = \boldsymbol{x}^*$. However, the degree of the minimal polynomial of $\boldsymbol{A}$ can be as large as N, which in our case is very high. Moreover, we also do not have a way to obtain this degree with an easy algorithm. Because of these two difficulties, some approximate methods are developed to extrapolate the next vector via the previous ones and we will discuss them in Section 2.3.2.

Turning to the nonlinear case, denote by $\boldsymbol{F}$ the FP function to evaluate the next vector,

$$\boldsymbol{x}_{i+1} = \boldsymbol{F}(\boldsymbol{x}_i),\ i = 0, 1, \cdots, \tag{2.17}$$

where $\boldsymbol{F}$ is an N-dimensional vector-valued function, $\boldsymbol{F} : \mathbb{R}^N \to \mathbb{R}^N$. We say $\boldsymbol{x}^*$ is a FP of $\boldsymbol{F}$ if $\boldsymbol{x}^* = \boldsymbol{F}(\boldsymbol{x}^*)$. Expanding $\boldsymbol{F}(\boldsymbol{x})$ in its Taylor series yields

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}^*) + \boldsymbol{F}'(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*) + O(\|\boldsymbol{x} - \boldsymbol{x}^*\|^2) \ \text{ as } \ \boldsymbol{x} \to \boldsymbol{x}^*,$$

where $\boldsymbol{F}'(\cdot)$ is the Jacobian matrix of $\boldsymbol{F}(\cdot)$. Recalling $\boldsymbol{F}(\boldsymbol{x}^*) = \boldsymbol{x}^*$, we have

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x}^* + \boldsymbol{F}'(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*) + O(\|\boldsymbol{x} - \boldsymbol{x}^*\|^2) \ \text{ as } \ \boldsymbol{x} \to \boldsymbol{x}^*.$$

Assuming the sequence $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ converges to $\boldsymbol{x}^*$ (if $\rho(\boldsymbol{F}'(\boldsymbol{x})) < 1$), it follows that $\boldsymbol{x}_i$ will be close enough to $\boldsymbol{x}^*$ for all large i, and hence

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}^* + \boldsymbol{F}'(\boldsymbol{x}^*)(\boldsymbol{x}_i - \boldsymbol{x}^*) + O(\|\boldsymbol{x}_i - \boldsymbol{x}^*\|^2), \ \text{ as } \ i \to \infty.$$

Then, we rewrite this in the form

$$\boldsymbol{x}_{i+1} - \boldsymbol{x}^* = \boldsymbol{F}'(\boldsymbol{x}^*)(\boldsymbol{x}_i - \boldsymbol{x}^*) + O(\|\boldsymbol{x}_i - \boldsymbol{x}^*\|^2), \ \text{ as } \ i \to \infty.$$

For large i, the vectors $\{\boldsymbol{x}_i\}$ behave as in the linear system of the form $(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b}$ through

$$\boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}, \ \ i = 0, 1, \cdots,$$

where $\boldsymbol{A} = \boldsymbol{F}'(\boldsymbol{x}^*)$, $\boldsymbol{b} = [\boldsymbol{I} - \boldsymbol{F}'(\boldsymbol{x}^*)]\boldsymbol{x}^*$. This implies that the nonlinear system yields the same formula as the linear one and motivates us to extrapolate the next vector by the previous ones as in linear systems. Indeed, such an extension has been shown to be successful in various areas of science and engineering, e.g., computational fluid dynamics, semiconductor research, tomography, and geometrical image processing [35, 40, 36].

2.3.2 Derivations of MPE, RRE, and SVD-MPE

We next discuss an approximate way to obtain the next vector by extrapolating the previous ones. Due to the fact that the degree of the minimal polynomial can be as large as N and we cannot obtain it, an arbitrary positive number is set as the degree, being much smaller than the true one. With such a replacement, the linear equations in (2.16) become inconsistent and there does not exist a solution for $\{c_i\}, c_\kappa = 1$ in the

ordinary sense. Alternatively, we solve instead

$$\min_{\boldsymbol{c}} \|\boldsymbol{U}_{\kappa}^{m}\boldsymbol{c}\|_2^2, \ \text{s.t.} \ c_{\kappa} = 1, \tag{2.18}$$

where $\boldsymbol{c} = \begin{bmatrix} c_0 & \cdots & c_{\kappa} \end{bmatrix}^{\mathcal{T}}$ and $\boldsymbol{U}_{\kappa}^{m} = \begin{bmatrix} \boldsymbol{u}_m & \cdots & \boldsymbol{u}_{m+\kappa} \end{bmatrix}$. Then evaluating γ_i through $c_i / \left(\sum_{i=0}^{i=\kappa} c_i\right)$ results in the next vector $\boldsymbol{x}_{(m,\kappa)} = \sum_{i=0}^{\kappa} \gamma_i \boldsymbol{x}_{m+i}$ as a new approximation. This method is known as Minimal Polynomial Extrapolation (MPE) [41].

The detailed steps to obtain the next vector through MPE are shown in Algorithm 2.1. To solve the constrained problem in (2.18), we suggest utilizing QR decomposition with the modified Gram-Schmidt (MGS) [41, 42]. The MGS procedure for the matrix $\boldsymbol{U}_{\kappa}^{m}$ is shown in Algorithm 2.2.

Algorithm 2.1 Minimal Polynomial Extrapolation (MPE)

Input: A sequence of vectors $\{\boldsymbol{x}_m, \boldsymbol{x}_{m+1}, \boldsymbol{x}_{m+2}, \cdots, \boldsymbol{x}_{m+\kappa+1}\}$ is produced by the baseline algorithm (FP in our case).

Output: A new vector $\boldsymbol{x}_{(m,\kappa)}$.

1: Construct the matrix

$$\boldsymbol{U}_{\kappa}^{m} = \begin{bmatrix} \boldsymbol{x}_{m+1} - \boldsymbol{x}_m, \cdots, \boldsymbol{x}_{m+\kappa+1} - \boldsymbol{x}_{m+\kappa} \end{bmatrix} \in \mathbb{R}^{N \times (\kappa+1)},$$

and then compute its QR factorization via Algorithm 2.2, $\boldsymbol{U}_{\kappa}^{m} = \boldsymbol{Q}_{\kappa}\boldsymbol{R}_{\kappa}$.

2: Denote $\boldsymbol{r}_{\kappa+1}$ as the $\kappa+1$th column of $\boldsymbol{R}_{\kappa}$ without the last row and solve the following $\kappa \times \kappa$ upper triangular system

$$\boldsymbol{R}_{\kappa-1}\boldsymbol{c}' = -r_{\kappa+1}, \quad \boldsymbol{c}' = \begin{bmatrix} c_0, c_1, \cdots, c_{\kappa-1} \end{bmatrix}^{\mathcal{T}},$$

where $\boldsymbol{R}_{\kappa-1}$ is the previous κ columns of $\boldsymbol{R}_{\kappa}$ without the last row. Finally, evaluate $\{\gamma_i\}$ through $\left\{ \frac{c_i}{\sum_{i=0}^{k} c_i} \right\}$.

3: Compute $\boldsymbol{\xi} = \begin{bmatrix} \xi_0, \xi_1, \cdots, \xi_{\kappa-1} \end{bmatrix}^{\mathcal{T}}$ through

$$\xi_0 = 1 - \gamma_0, \ \ \xi_j = \xi_{j-1} - \gamma_j, \ \ j = 1, \cdots, \kappa - 1.$$

4: Compute $\boldsymbol{\eta} = \begin{bmatrix} \eta_0, \eta_1, \cdots, \eta_{\kappa-1} \end{bmatrix}^{\mathcal{T}} = \boldsymbol{R}_{\kappa-1}\boldsymbol{\xi}$. Then we attain $\boldsymbol{x}_{(m,\kappa)} = \boldsymbol{x}_m + \boldsymbol{Q}_{\kappa-1}\boldsymbol{\eta}$ as the new initial vector where $\boldsymbol{Q}_{\kappa-1}$ represents the previous κ columns of $\boldsymbol{Q}_{\kappa}$.

Algorithm 2.2 Modified Gram-Schmidt (MGS)

Output: $\boldsymbol{Q}_{\kappa}$ and $\boldsymbol{R}_{\kappa}$ (r_{ij} denotes the (i,j)th element of $\boldsymbol{R}_{\kappa}$ and $\boldsymbol{q}_i$ and $\boldsymbol{u}_i$ represent the ith column of $\boldsymbol{Q}_{\kappa}$ and $\boldsymbol{U}_{\kappa}^{m}$, respectively.).

1: Compute $r_{11} = \|\boldsymbol{u}_1\|_2$ and $\boldsymbol{q}_1 = \boldsymbol{u}_1 / r_{11}$.
2: **for** $i = 2, \cdots, \kappa+1$ **do**
3: $\quad \boldsymbol{u}_i^{(1)} \leftarrow \boldsymbol{u}_i$.
4: $\quad$ **for** $j = 1, \cdots, i-1$ **do**
5: $\quad\quad r_{ji} \leftarrow \boldsymbol{q}_j^{\mathcal{T}} \boldsymbol{u}_i^{(j)}$ and $\boldsymbol{u}_i^{(j+1)} \leftarrow \boldsymbol{u}_i^{(j)} - r_{ji}\boldsymbol{q}_j$.
6: $\quad$ **end for**
7: $\quad r_{ii} \leftarrow \|\boldsymbol{u}_i^{(i)}\|_2$.
8: $\quad \boldsymbol{q}_i \leftarrow \boldsymbol{u}_i^{(i)} / r_{ii}$.
9: **end for**

Now we discuss the other two variants of VE, i.e., Reduced Rank Extrapolation (RRE) [41] and SVD-MPE [38]. The main differences between RRE, MPE, and SVD-MPE is at Step 2 in Algorithm 2.1

regarding the evaluation of $\{\gamma_i\}$. In RRE and SVD-MPE, we utilize the following methods to obtain $\{\gamma_i\}$:

RRE: Solving $\boldsymbol{R}_\kappa^{\mathcal{T}}\boldsymbol{R}_\kappa\boldsymbol{d}=\mathbf{1}$ through forward and backward substitution, we obtain $\boldsymbol{\gamma}$ through $\frac{\boldsymbol{d}}{\sum_i d_i}$ where d_i is the ith element of $\boldsymbol{d}$. Actually, such a formulation of $\boldsymbol{\gamma}$ is the solution of:

$$\min_{\boldsymbol{\gamma}} \|\boldsymbol{U}_\kappa^m\boldsymbol{\gamma}\|_2^2, \ \text{ s.t. } \sum_i \gamma_i = 1. \tag{2.19}$$

SVD-MPE: Computing the SVD decomposition of $\boldsymbol{R}_\kappa=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathcal{T}}$, we have $\boldsymbol{\gamma}=\frac{\boldsymbol{v}_{\kappa+1}}{\sum_i \boldsymbol{v}_{i,\kappa+1}}$ where $\boldsymbol{v}_{\kappa+1}$ and $\boldsymbol{v}_{i,\kappa+1}$ represent the last column and the $(i,\kappa+1)$th element of matrix $\boldsymbol{V}$, respectively.

Remark 2.1.

- Observing the derivations of MPE, SVD-MPE and RRE, we notice that RRE's solution must exist unconditionally, while MPE and SVD-MPE may not exist because the sum of $\{c_i\}$ and $\{v_{i,\kappa+1}\}$ in MPE and SVD-MPE may become zero. Thus RRE may be more robust in practice [40]. However, MPE and RRE are related, as revealed in [43]. Specifically, if MPE does not exist, we have $\boldsymbol{x}_{(m,\kappa)}^{RRE}=\boldsymbol{x}_{(m,\kappa-1)}^{RRE}$. Otherwise, the following holds

  $$\mu_\kappa\boldsymbol{x}_{(m,\kappa)}^{RRE}=\mu_{\kappa-1}\boldsymbol{x}_{(m,\kappa-1)}^{RRE}+v_\kappa\boldsymbol{x}_{(m,\kappa)}^{MPE}, \ \ \mu_\kappa=\mu_{\kappa-1}+v_\kappa,$$

  where μ_κ, $\mu_{\kappa-1}$, and v_κ are positive scalars depending only on $\boldsymbol{x}_{(m,\kappa)}^{RRE}$, $\boldsymbol{x}_{(m,\kappa-1)}^{RRE}$ and $\boldsymbol{x}_{(m,\kappa)}^{MPE}$, respectively. Furthermore, the performance of MPE and RRE is similar – both of the methods either perform well or work poorly [36].

- Observe that we only need to store $\kappa+2$ vectors in memory at all steps in Algorithm 2.2. Specifically, formulating the matrix $\boldsymbol{U}_m^\kappa$, we overwrite the vector $\boldsymbol{x}_{m+i}$ with $\boldsymbol{u}_{m+i}=\boldsymbol{x}_{m+i}-\boldsymbol{x}_{m+i-1}$ when the latter is computed and only $\boldsymbol{x}_m$ is always in the memory. Next, $\boldsymbol{u}_{m+i}$ is overwritten by $\boldsymbol{q}_i$, $i=1,\cdots,\kappa+1$ in computing the matrix $\boldsymbol{Q}_\kappa$. Thus, we do not need to save the vectors $\boldsymbol{x}_{m+1},\cdots,\boldsymbol{x}_{m+\kappa+1}$, which implies that no additional memory is required in running Algorithm 2.2.

2.3.3 Embedding Vector Extrapolation in the Baseline Algorithm

In this part, we introduce VE in its cycling formulation for practical use. One cycle means we activate the baseline algorithm to produce $\{\boldsymbol{x}_i\}$ and then utilize VE once to evaluate the new vector as a novel initial value. Naturally, we repeat such a cycle many times. The steps of utilizing VE in its cycling mode are shown in Algorithm 2.3. A few comments are in order:

- In practice, we utilize VE in its cycling formulation. Specifically, the iterative form shown in Algorithm 2.3 is called full cycling [36]. To save the complexity of computing $\{\gamma_i\}$, one may reuse the previous $\{\gamma_i\}$, a method known as cycling with frozen γ_i. Parallel VE can also be developed if more machines are available. Details of the last two strategies are outside the scope of this chapter. We refer the reader to [36] for more information.

- Numerical experience also indicates that cycling with even moderately large $m>0$ will avoid stalling from happening [44]. Moreover, we also recommend setting $m>0$ when the problem becomes challenging to solve.

- In our case, the termination criterion in Algorithm 2.3 can be the number of total iterations (the number of calls to the baseline algorithm) or the difference between consecutive two vectors. Furthermore, we also recommend additional iterations to activate the baseline algorithm after terminating the VE, which can stabilize the accelerated algorithm in practice.

Algorithm 2.3 Baseline Algorithm + Vector Extrapolation

Input: Choose nonnegative integers m and κ and an initial vector $\boldsymbol{x}_0$. The baseline algorithm is the FP method, as given in (2.7).

Output: $\boldsymbol{x}^*$.

1: **while** 1 **do**
2: Obtain the series of $\boldsymbol{x}_i$ through the baseline algorithm where $1 \le i \le m+\kappa+1$, and save $\boldsymbol{x}_{m+i}$ for $0 \le i \le \kappa+1$ to formulate $\boldsymbol{U}_m^\kappa$.
3: Call Algorithm 2.1 to obtain $\boldsymbol{x}_{(m,\kappa)}$.
4: If the termination of the algorithm is satisfied, set $\boldsymbol{x}^* = \boldsymbol{x}_{(m,\kappa)}$ and break, otherwise, set $\boldsymbol{x}_{(m,\kappa)}$ as the new initial value $\boldsymbol{x}_0$ and go to Line 2.
5: **end while**

2.3.4 Convergence and Stability Properties

We mention existing results regarding the convergence and stability properties of VE for understanding this technique better. A rich literature has examined the convergence and stability properties of RRE, MPE, and SVD-MPE in linear systems [45, 46]. Assuming the matrix $\boldsymbol{A}$ is diagonalizable, then in the kth iteration $\boldsymbol{x}_k$ should have the form $\boldsymbol{x}_k = \boldsymbol{x}^* + \sum_{i=1}^{\kappa} \boldsymbol{v}_i \lambda_i^k$ where $(\lambda_i, \boldsymbol{v}_i)$ are some or all of the eigenvalues and corresponding eigenvectors of $\boldsymbol{A}$, with distinct nonzero eigenvalues. By ordering λ_i as $|\lambda_1| \ge |\lambda_2| \ge \cdots$, the following asymptotic performance holds for all of the three variants of VE when $|\lambda_k| > |\lambda_{k+1}|$:

$$\boldsymbol{x}_{(m,\kappa)} - \boldsymbol{x}^* = O(\lambda_{\kappa+1}^m) \ \text{ as } \ m \to \infty. \tag{2.20}$$

This implies that the sequence $\{\boldsymbol{x}_{(m,\kappa)}\}_{m=0}^{\infty}$ converges to $\boldsymbol{x}^*$ faster than the original sequence $\{\boldsymbol{x}_k\}$.

As shown in (2.20), for a large m, (2.8) reduces the contributions of the smaller λ_i to the error $\boldsymbol{x}_{(m,\kappa)} - \boldsymbol{x}^*$, while VE eliminates the contributions of the κ largest λ_i. This indicates that $\boldsymbol{x}_{(m,\kappa)} - \boldsymbol{x}^*$ is smaller than each of the errors $\boldsymbol{x}_{m+i} - \boldsymbol{x}^*$, $i = 0, 1, \cdots, \kappa$, when m is large enough. We mention another observation that an increasing κ generally results in a faster convergence of VE. However, a large κ has to increase the storage requirements and also requires a much higher computational cost. Numerical experiments indicate that a moderate κ can already work well in practice.

If the following condition is held, we say VE is stable:

$$\sup_m \sum_{i=0}^{\kappa} |\gamma_i^{(m,\kappa)}| < \infty. \tag{2.21}$$

Here, we denote $\{\gamma_i\}$ by $\{\gamma_i^{(m,\kappa)}\}$ to show their dependence on m and κ. If (2.21) holds true, the error in $\boldsymbol{x}_i$ will not magnify severely. As shown in [45, 46, 47], MPE and RRE obey such a stability property.

For nonlinear systems, the analysis of convergence and stability becomes extremely challenging. One of the main results is the quadratic convergence theorem [48, 39, 49]. This theorem is built on one special assumption that κ is set to be the degree of the minimal polynomial of $\boldsymbol{F}'(\boldsymbol{x}^*)$. The proof of the quadratic

convergence was shown in [48]. In subsequent work, Smith, Ford and Sidi noticed that there exists a gap in the previous proof [39]. Jbilou et al. suggested two more conditions in order to close the gap [49]:

- The matrix $\boldsymbol{F}'(\boldsymbol{x}^*) - \boldsymbol{I}$ is nonsingular.
- $\boldsymbol{F}'(\cdot)$ satisfies the following Lipschitz condition:

$$\|\boldsymbol{F}'(\boldsymbol{x}) - \boldsymbol{F}'(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \;\; L > 0.$$

Surprisingly, these two conditions are met by the RED scheme. The first condition is satisfied by the fact

$$\rho\left(\left[\frac{1}{\sigma^2}\boldsymbol{H}^T\boldsymbol{H} + \alpha\boldsymbol{I}\right]^{-1}\alpha\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\right) < 1.$$

The second one is also true, due to the assumption in RED that the denoiser $f(\boldsymbol{x})$ is differentiable. So we claim that it is possible for VE to solve RED with quadratic convergence rate.

Although VE can lead to a quadratic convergence rate, trying to achieve such a rate may not be realistic because κ may need to be as large as N. However, we may obtain a linear but fast convergence in practice with even moderate values of m and κ, which is also demonstrated in the following numerical experiments.

2.4 Experimental Results

We follow the same experiments of image deblurring and super-resolution as presented in [5] to investigate the performance of VE in acceleration. The trainable nonlinear reaction diffusion (TNRD) method [23] is chosen as the denoising engine. Mainly, we choose the FP method as our baseline algorithm. For a fair comparison, the same parameters suggested in [5] for different image processing tasks are used in our experiments. In [5], the authors compared RED with other algorithms in image deblurring and super-resolution tasks, showing its superiority. As the main purpose in this chapter is to present the acceleration of our method for solving RED, we omit the comparisons with other popular algorithms. In the following, we mainly show the acceleration of applying MPE with FP for solving RED first and then discuss the choice of parameters in VE, i.e., m and κ. In addition, we compare our method with three other methods, steepest descent (SD), Nesterov's acceleration [50], and Limited-memory BFGS (L-BFGS) [51]. Note that we need to determine a proper stepsize for the above methods [51]. However, evaluating the objective value or gradient in RED is expensive implying that any line-search method becomes prohibitive. Note that, in contrast, in our framework as described in Algorithm 2.3 does not suffer from such a problem. In the following, we manually choose a fixed stepsize for getting a good convergence behavior. Finally, we compare the differences between RRE, MPE, and SVD-MPE. All of the experiments are conducted on a workstation with Intel(R) Xeon(R) CPU E5-2699 @2.20GHz.

2.4.1 Image Deblurring

In this experiment, we degrade the test images by convolving with two different point spread functions (PSFs), i.e., 9×9 uniform blur and a Gaussian blur with a standard derivation of 1.6. In both of these cases, we add an additive Gaussian noise with $\sigma = \sqrt{2}$ to the blurred images. The parameters m and κ in

Algorithm 2.3 are set to 0 and 5 for the image deblurring task. Additionally, we apply VE to the case where the baseline algorithm is SD, called SD-MPE, with the parameters $m = 0$ and $\kappa = 8$. The value of the cost function and peak signal to noise ratio (PSNR) versus iteration or CPU time are given in Figures 2.2 and 2.3[1]. These correspond to both a uniform and a Gaussian blur kernels, all tested on the "Starfish" image. Clearly, we observe that SD is the slowest algorithm. Surprisingly, SD-MPE and Nesterov's method yield almost the same convergence speed, despite their totally different scheme. Moreover, We note that FP is faster than L-BFGS, Nesterov's method, and SD-MPE. Undoubtedly, FP-MPE is the fastest one, both in terms of iterations and CPU time, which indicates the effectiveness of MPE's acceleration. To provide a visual effect, we show the change in reconstructed quality of different algorithms in Figure 2.1. Clearly, the fourth column of FP-MPE achieves the best reconstruction faster, while other methods need more iterations to obtain a comparable result.

Nine additional test images suggested in [5] are also included in our experiments, in order to investigate the performance of VE further. In this experiment we focus on the comparison between FP-MPE and FP for the additional images. We run the native FP method 200 iterations first and denote the final image by $\boldsymbol{x}^*$. Clearly, the corresponding cost-value is $E(\boldsymbol{x}^*)$. We activate Algorithm 2.3 with the same initial value as used in the FP method to examine how many iterations are needed to attain the same or lower objective value than $E(\boldsymbol{x}^*)$. The final number of iterations with different images are given in Table 2.1. Clearly, an acceleration is observed in all the test images in the image deblurring task.

2.4.2 Image Super-resolution

We generate a low resolution image by blurring the ground truth one with a 7×7 Gaussian kernel with standard derivation 1.6 and then downsample by a factor of 3. Afterwards, an additive Gaussian noise with $\sigma = 5$ is added to the resulting image. The same parameters m and κ used in the deblurring task for FP-MPE are adopted here. For SD-MPE, the parameters m and κ are set to 1 and 10, respectively. We choose "Plants" as our test image because it needs more iterations for FP-MPE to converge. As observed from Figure 2.4, while L-BFGS and the Nesterov's method are faster than the FP method, our acceleration method (FP-MPE) is quite competitive with both. Furthermore, we investigate all of the test images as shown in [5] to see how many iterations are needed for MPE to achieve the same or lower cost compared with the FP method. The results are shown in Table 2.2. As can be seen, MPE works better than the FP method indicating an effective acceleration for solving RED.

2.4.3 The Choice of the Parameters and the Differences between RRE, MPE, and SVD-MPE

We conclude by discussing the robustness with respect to the choice of parameters m and κ for the MPE algorithm. To this end, the single image super-resolution task is chosen as our study. Furthermore, we choose to demonstrate this robustness on the "Plants" image since it required the largest number of iterations in the MPE recovery process. As seen from Figures 2.5(a) to 2.5(c), MPE always converges faster than the regular FP method. Moreover, we also observe that a lower objective value is attained by

[1]The goal of this chapter is to investigate the performance of solving RED with VE rather than the restoration results. Therefore, we present the recovered PSNR versus iteration or running time. One can utilize some no-reference quality metrics like NFERM [52] and ARISMc [53] to further examine the restoration results.

(a) SD: 22.56dB (input) → 25.65dB → 26.47dB → 26.97dB → 27.35dB → original.

(b) SD-MPE: 22.56dB (input) → 27.46dB → 28.62dB → 29.29dB → 29.69dB → original.

(c) Nesterov: 22.56dB (input) → 27.44dB → 28.78dB → 29.43dB → 29.79dB → original.

(d) LBFGS: 22.56dB (input) → 27.80dB → 29.52dB → 30.14dB → 30.40dB → original.

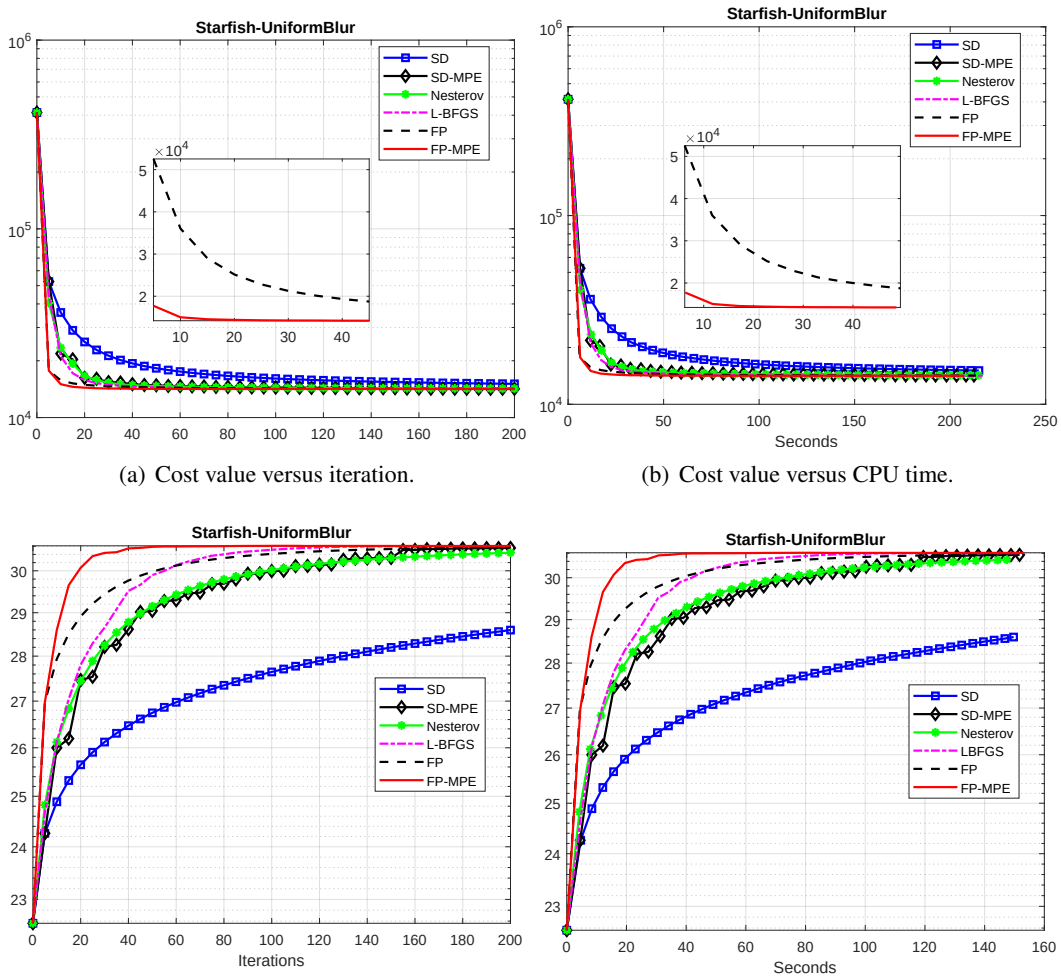
(e) FP: 22.56dB (input) → 28.91dB → 29.76dB → 30.13dB → 30.31dB → original.

(f) FP-MPE: 22.56dB (input) → 30.07dB → 30.55dB → 30.60dB → 30.60dB → original.

Figure 2.1: Reconstruction of different algorithms in various iterations for the Image deblurring with uniform kernel. From left to right: Blurred one → #20 → #40 → #60 → #80 → Ground truth.

MPE. Notice that MPE has some oscillations because it is not a monotonically accelerated technique. However, we still see a lower cost is achieved if additional iterations are given.

In panel (d) of Figure 2.5, an optimal pair of m and κ is chosen for MPE, RRE, and SVD-MPE for the single image super-resolution task with the "Plants" image. The optimal m and κ are obtained by



(a) Cost value versus iteration.

(b) Cost value versus CPU time.

(c) PSNR versus iteration.

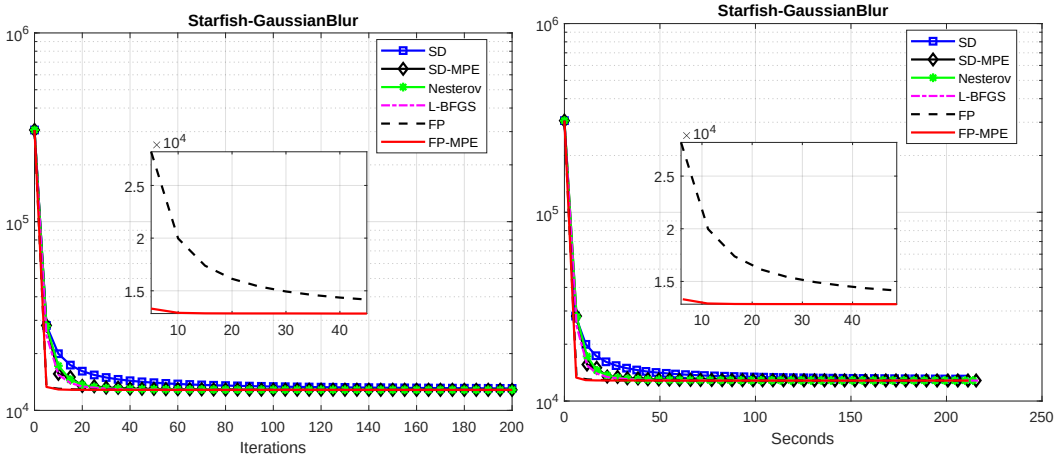
(d) PSNR versus CPU time.

Figure 2.2: Image deblurring - uniform kernel, "Starfish" image.

searching in the range $[0, 10]$ with $\kappa \geq 2$, seeking the fastest convergence for these three methods. We see that all three methods yield an acceleration and a lower cost, demonstrating the effectiveness of the variants of VE. Moreover, we see that SVD-MPE converges faster at the beginning, but MPE yields the lowest final cost.

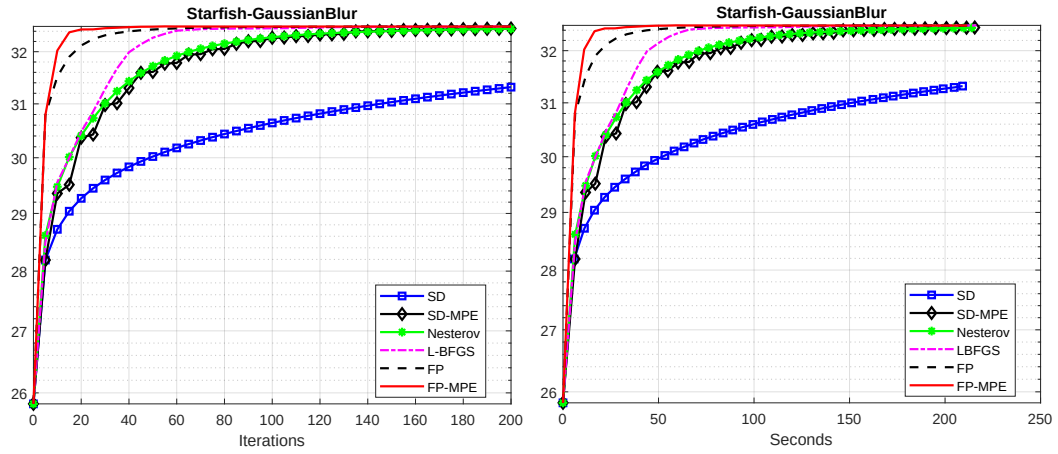
2.5 Conclusion

The work reported in [5] introduced RED – a flexible framework for using arbitrary image denoising algorithms as priors for general inverse problems. This scheme amounts to iterative algorithms in which the denoiser is called repeatedly. While appealing and quite practical, there is one major weakness to the RED scheme – the complexity of denoising algorithms is typically high which implies that the use of RED is likely to be costly in run-time. This work aims at deploying RED efficiently, alleviating the above described shortcoming. An accelerated technique is proposed in this chapter, based on the Vector Extrapolation (VE) methodology. The proposed algorithms are demonstrated to substantially reduce the number of iterations required for the overall recovery process. We also observe that the choice of the parameters in the VE scheme is robust.



(a) Cost value versus iteration.

(b) Cost value versus CPU time.



(c) PSNR versus iteration

(d) PSNR versus iteration

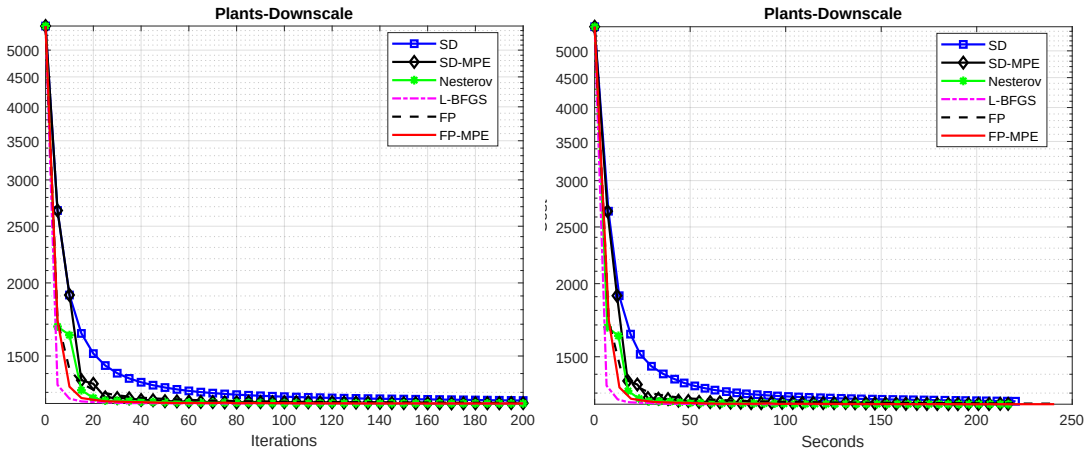
Figure 2.3: Image deblurring - Gaussian kernel, “Starfish” image.

Table 2.1: The number of iterations with different images for FP and FP-MPE in image deblurring task to attain the same cost.

Image	Butterfly	Boats	C. Man	House	Parrot	Lena	Barbara	Starfish	Peppers	Leaves
Deblurring: Uniform kernel, $\sigma = \sqrt{2}$										
RED: FP-TNRD	200	200	200	200	200	200	200	200	200	200
RED: FP-MPE-TNRD	60	55	55	80	50	50	55	50	55	55
Deblurring: Gaussian kernel, $\sigma = \sqrt{2}$										
RED: FP-TNRD	200	200	200	200	200	200	200	200	200	200
RED: FP-MPE-TNRD	70	65	55	65	55	80	45	55	95	80

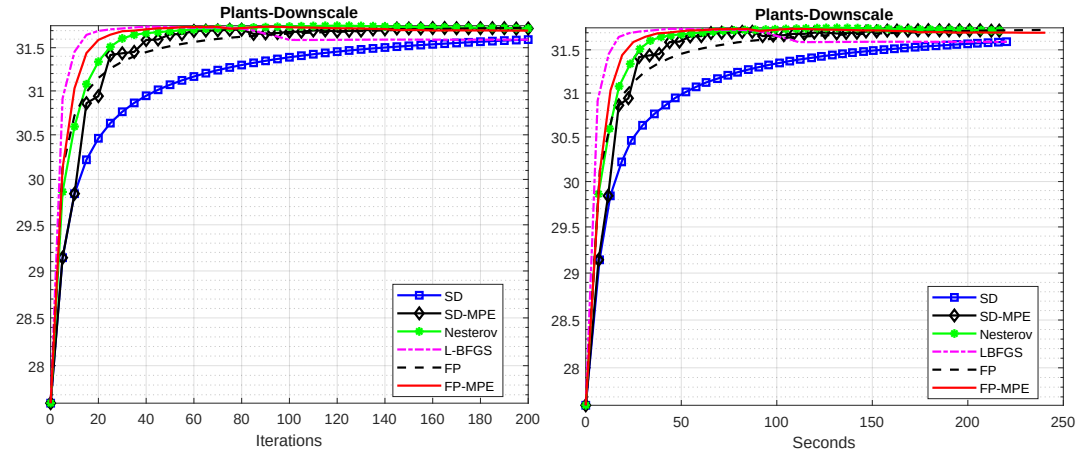
Table 2.2: The number of iterations with different images for FP and FP-MPE in image super-resolution task to attain the same cost.

Super-Resolution, scaling = 3, $\sigma = 5$									
Image	Butterfly	Flower	Girl	Parth.	Parrot	Raccoon	Bike	Hat	Plants
RED: FP-TNRD	200	200	200	200	200	200	200	200	200
RED: FP-MPE-TNRD	60	65	50	70	55	60	60	50	70



(a) Cost value versus iteration.

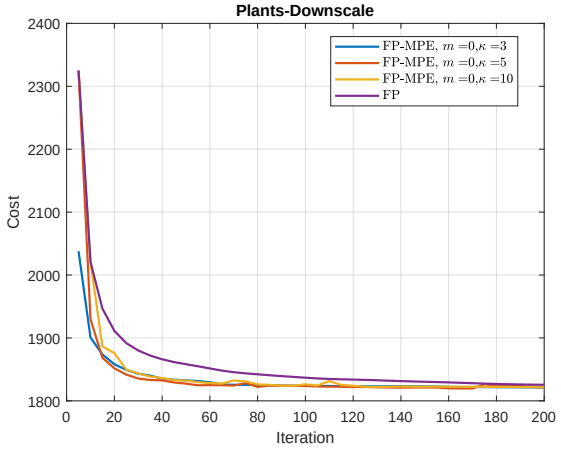
(b) Cost value versus CPU time.



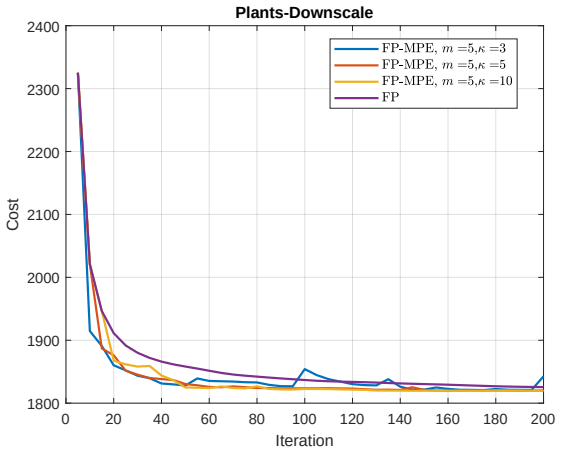
(c) PSNR versus iteration.

(d) PSNR versus CPU time.

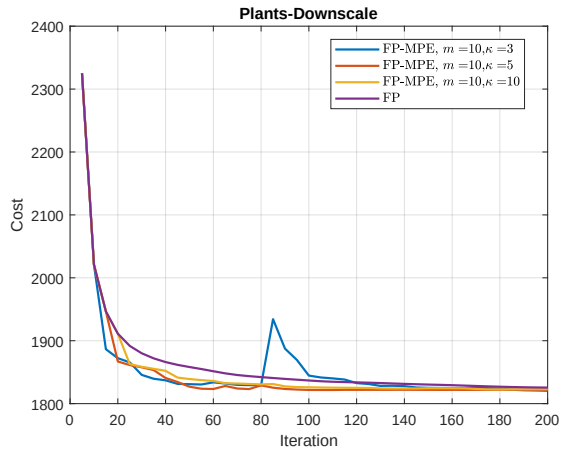
Figure 2.4: Image super-resolution, “Plant” image.



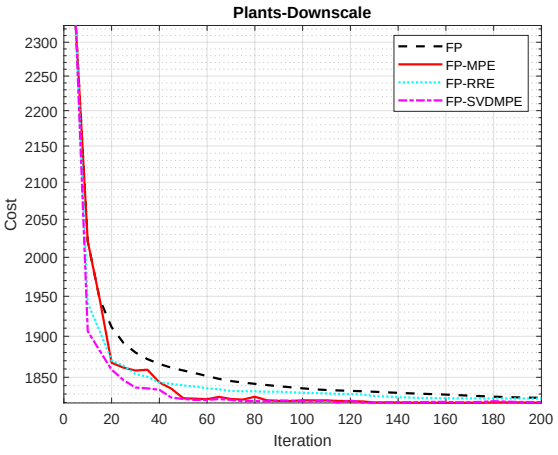
(a) $m = 0$ and $\kappa = 3, 5, 10$.



(b) $m = 5$ and $\kappa = 3, 5, 10$.



(c) $m = 10$ and $\kappa = 3, 5, 10$.



(d) Comparison between MPE, RRE and SVD-MPE.

Figure 2.5: Exploring the robustness to the choice of the parameters in MPE, and the differences between the three VE schemes. All these graphs correspond to the test image "Plants" for the single image super-resolution task.

Chapter 3

Solving RED with Weighted Proximal Methods

In this chapter, we report our study on employing weighted proximal methods for RED, which is faster than the technique described in Chapter 2. This chapter is summarized in the following published paper:

- **Tao Hong**, Irad Yavneh, and Michael Zibulevsky, *Solving RED with Weighted Proximal Methods,* IEEE Signal Processing Letters, vol. 27, pp. 501-505, Mar. 2020.

Since the work presented in this chapter solves the same problem as shown in Chapter 2, we omit the introduction and the problem formulation and discuss our technique directly. Note that the notation used in Chapter 2 is adapted in this chapter as well.

3.1 Weighted Proximal Methods

Consider the following composite problem and assume its solution set is nonempty,

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathcal{F}(\boldsymbol{x}) \triangleq g(\boldsymbol{x}) + h(\boldsymbol{x}), \tag{3.1}$$

where g and h are convex and differentiable. Denote the proximal operator by

$$\text{prox}_{h,\boldsymbol{B}}(\hat{\boldsymbol{x}}) = \arg\min_{\boldsymbol{u}\in\mathbb{R}^N} \left\{ h(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u}-\hat{\boldsymbol{x}}\|_{\boldsymbol{B}}^2 \right\}, \tag{3.2}$$

where $\boldsymbol{B}$ is a symmetric positive definite matrix called the weighting and $\|\cdot\|_{\boldsymbol{B}}$ denotes the $\boldsymbol{B}$-norm, $\|\boldsymbol{q}\|_{\boldsymbol{B}} = \sqrt{\boldsymbol{q}^T\boldsymbol{B}\boldsymbol{q}}$. With these, we describe the explicit form of WPMs for (3.1) in Algorithm 3.1 [54, Chap. 10.7.5]. Note that by setting $\boldsymbol{B} = \beta\boldsymbol{I}$ with $\beta > 0$, we recover the proximal gradient (PG) method. Usually, PG is used for (3.1) when h is nonsmooth [55], whereas here we use it even though h is differentiable. We do this for computational efficiency, knowing that applying the denoiser is the most expensive part of the solution process.

To apply Algorithm 3.1 to RED, we set $g(\boldsymbol{x}) = \alpha R(\boldsymbol{x}) = \frac{\alpha}{2}\boldsymbol{x}^T(\boldsymbol{x} - f(\boldsymbol{x}))$ and $h(\boldsymbol{x}) = \frac{1}{2\sigma^2}\|\boldsymbol{H}\boldsymbol{x}-\boldsymbol{y}\|_2^2$. If $h(\boldsymbol{x})$ is convex, solving (3.2) is equivalent to satisfying the first-order optimality condition,

$$\nabla_{\boldsymbol{u}} h(\boldsymbol{u}) + \boldsymbol{B}(\boldsymbol{u}-\hat{\boldsymbol{x}}) = \boldsymbol{0}. \tag{3.3}$$

Algorithm 3.1 Weighed Proximal Methods (WPMs)

Input: $\boldsymbol{x}_0$.

Output: $\boldsymbol{x}^*$.

1: **for** $k = 0, 1, \ldots$ **do**
2: pick the step-size a_k and the weighting $\boldsymbol{B}_k$.
3: $\boldsymbol{x}_{k+1} \leftarrow \mathrm{prox}_{a_k h, \boldsymbol{B}_k}\left(\boldsymbol{x}_k - a_k \boldsymbol{B}_k^{-1} \nabla_{\boldsymbol{x}} g(\boldsymbol{x}_k)\right)$.
4: **end for**

Substituting $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_k - a_k \boldsymbol{B}_k^{-1} \nabla_{\boldsymbol{x}} g(\boldsymbol{x}_k)$, $h(\boldsymbol{u}) \leftarrow a_k h(\boldsymbol{u})$ and $\boldsymbol{u} \leftarrow \boldsymbol{x}_{k+1}$, at the kth iteration into (3.3) and rearranging, we obtain

$$\left(\frac{a_k}{\sigma^2}\boldsymbol{H}^{\mathcal{T}}\boldsymbol{H} + \boldsymbol{B}_k\right)\boldsymbol{x}_{k+1} = \frac{a_k}{\sigma^2}\boldsymbol{H}^{\mathcal{T}}\boldsymbol{y} + \boldsymbol{B}_k\boldsymbol{x}_k - a_k\alpha\left(\boldsymbol{x}_k - f(\boldsymbol{x}_k)\right). \tag{3.4}$$

In this chapter, we use the conjugate gradient (CG) method to approximately solve (3.4) for $\boldsymbol{x}_{k+1}$.

Next we discuss possible practical choices for the weighting $\boldsymbol{B}_k$. Note first that if we set $\boldsymbol{B}_k = \alpha\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix, and select the step-size $a_k = 1$, (3.4) is reduced to the PG method and we recover the FP method. Moreover, by using the accelerated version of Algorithm 3.1 (cf. [54, Chap. 10.7.5]) we get APG [56]. We now propose a more elaborate approach of choosing some approximation to the Hessian of $\alpha R(\boldsymbol{x})$ as the weighting. (Because of the abstract denoiser in $R(\boldsymbol{x})$, the exact Hessian is not computable.) Specifically, we choose the *symmetric-rank-one* (SR1) approximation to the Hessian [51, Chap. 6.2], as is used in quasi-Newton methods. The SR1 approximation is described in Algorithm 3.2. This choice yields faster convergence in our experiments than either FP or APG, as shown below. We henceforth use WPM to denote Algorithm 3.1 with the weighting chosen by Algorithm 3.2.

Algorithm 3.2 SR1 updating

Input: $k = 1$, $\gamma = 1.25$, $\delta = 10^{-8}$, $\boldsymbol{x}_k$, $\boldsymbol{x}_{k-1}$, $\nabla g(\boldsymbol{x}_k)$, $\nabla g(\boldsymbol{x}_{k-1})$.

Output: $\boldsymbol{B}_k$.

1: **if** $k = 1$ **then**
2: $\boldsymbol{B}_k \leftarrow \alpha\boldsymbol{I}$.
3: **else**
4: Set $\boldsymbol{s}_k \leftarrow \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ and $\boldsymbol{m}_k \leftarrow \nabla g(\boldsymbol{x}_k) - \nabla g(\boldsymbol{x}_{k-1})$.
5: Calculate $\tau \leftarrow \gamma \frac{\|\boldsymbol{m}_k\|_2^2}{\langle \boldsymbol{s}_k, \boldsymbol{m}_k \rangle}$.
6: **if** $\tau < 0$ **then**
7: $\boldsymbol{B}_k \leftarrow \alpha\boldsymbol{I}$.
8: **else**
9: $\boldsymbol{H}_0 \leftarrow \tau\boldsymbol{I}$.
10: **if** $|\langle \boldsymbol{m}_k - \boldsymbol{H}_0\boldsymbol{s}_k, \boldsymbol{s}_k \rangle| \leq \delta \|\boldsymbol{s}_k\|_2 \|\boldsymbol{m}_k - \boldsymbol{H}_0\boldsymbol{s}_k\|_2$ **then**
11: $\boldsymbol{u}_k \leftarrow \boldsymbol{0}$.
12: **else**
13: $\boldsymbol{u}_k \leftarrow \frac{\boldsymbol{m}_k - \boldsymbol{H}_0\boldsymbol{s}_k}{\sqrt{\langle \boldsymbol{m}_k - \boldsymbol{H}_0\boldsymbol{s}_k, \boldsymbol{s}_k \rangle}}$.
14: **end if**
15: $\boldsymbol{B}_k \leftarrow \boldsymbol{H}_0 + \boldsymbol{u}_k\boldsymbol{u}_k^{\mathcal{T}}$.
16: **end if**
17: **end if**
18: **Return**

Unlike the traditional SR1, we formulate each $\boldsymbol{B}_k$ from the initial $\boldsymbol{H}_0$ rather the previous iterate $\boldsymbol{B}_{k-1}$ [51]. Moreover, we scale $\boldsymbol{H}_0$ by $\gamma > 1$ as suggested in [57], which we found useful in practice. In the

practical implementation of Algorithm 3.2, we efficiently represent $\boldsymbol{B}_k$ as a matrix-vector multiplication operator rather than as an explicit matrix.

In general, the step-size a_k in Algorithm 3.1 needs to be chosen by some line search process to guarantee monotonically decreasing objective values at each iteration. However, because evaluating the objective value in RED requires calling the denoiser, standard line search methods may dramatically increase the complexity of the algorithm. To maintain a low computational cost, we fix $a_k = 1$ and reduce the step-size by half only if the objective value exhibits a relative growth above some threshold, i.e., $E(\boldsymbol{x}_{k+1}) - E(\boldsymbol{x}_k) > \varepsilon E(\boldsymbol{x}_{k+1})$, where we use $\varepsilon = 10^{-2}$ in all our experiments. In practice, we found that we never needed to reduce the step-size.

In this chapter we only investigate the SR1 approximation to the Hessian of $\alpha R(\boldsymbol{x})$. We acknowledge that a more accurate Hessian estimate may prove to be even more cost-effective for RED, but leave such investigation to future work. Because we use an approximate Hessian for the weighting, our algorithm is equivalent to a quasi-newton proximal method. It follows that if both $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are strongly convex and their gradients are Lipschitz continuous, WPM with SR1 estimation, an appropriate step-size a_k, and exact solution of (3.2), converges linearly; see details in [58]. Because we depart from these strict requirements for efficiency, we cannot claim provable convergence in our implementation. However, in all our experiments WPM converged. Finally, we note that [56] challenges the validity in practice of the underlying assumptions of RED for most denoisers, concluding that (2.5) is not truly the gradient of (2.4). Nevertheless, setting (2.5) to zero, as is the objective of all the algorithms we discuss here, remains a most attractive method for signal recovery.

3.2 Numerical Experiments

In this section we investigate the performance of solvers for RED. Following [5], we perform our tests on image deblurring and super-resolution tasks and use the trainable nonlinear reaction diffusion (TNRD) [23] method as the abstract denoiser. We remark that one can adopt deep denoising techniques instead of TNRD, since the differentiability requirement of the denoiser is not mandatory in practice [56]. This may possibly lead to improved results in practice, but we do not investigate such options here. Also, since the authors in [5] already show the superiority of RED for image deblurring and super-resolution tasks compared with other popular algorithms, we largely omit such comparisons in this chapter and concentrate on computational efficiency. Moreover, the experiments conducted in [6] demonstrated that the FP method converges faster than LBFGS and Nesterov's acceleration for RED. Therefore, we only compare WPM to FP [5], FP-MPE [6], and APG [56]. All of the experiments are carried out on a laptop with Intel i7 − 6500U CPU @2.50GHz and 8GB RAM.

For image deblurring, the image is degraded by convolving with a point spread function (PSF), 9×9 uniform blur or a Gaussian blur with a standard derivation 1.6, and then adding Gaussian noise with mean zero and $\sigma = \sqrt{2}$. The recovered peak-signal-to-noise ratio (PSNR) versus the number of denoiser evaluations (left column) and running time (right column) when using RED for the "Starfish" image are shown in Figure 3.2. We find that the performances of FP-MPE and APG are similar, whereas WPM is more efficient than both, requiring less denoiser evaluations and running time to achieve a comparable PSNR. These results also indicate that indeed the denoiser dominates the complexity of solving RED.

Next, we test the algorithms on image super-resolution. A low resolution image is generated by

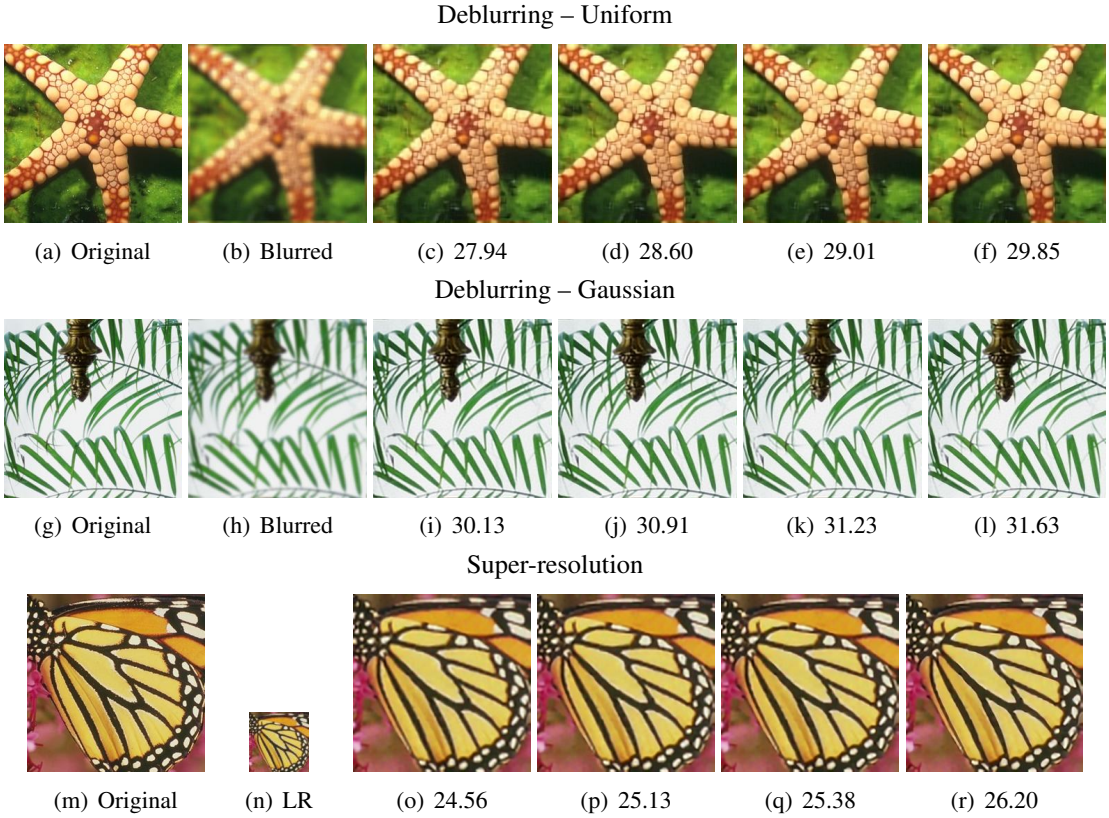


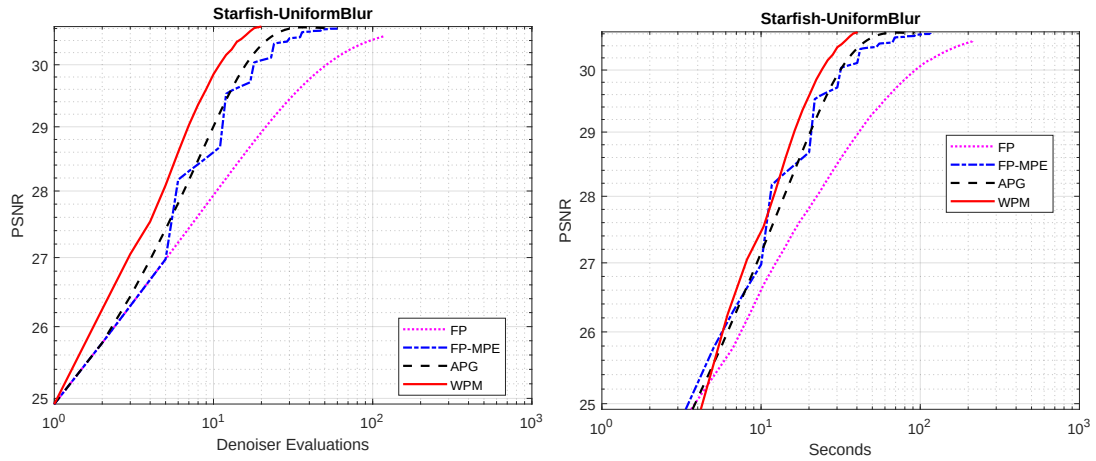
Figure 3.1: PSNR (dB) of the image recovered by, from left to right, FP, FP-MPE, APG, and WPM, after 10 denoiser evaluations. LR stands for low-resolution.

blurring a high-resolution image with a 7×7 Gaussian kernel with standard derivation 1.6, and then downscaling by a factor of 3. To the resulting image we add Gaussian noise with mean zero and $\sigma = 5$, resulting in our deteriorated image. The PSNR of the recovered fine-resolution image versus the number of denoiser evaluations (left) and running time (right) for the "Plants" image are presented in Figure 3.3. Again, we observe that WPM requires less denoiser evaluations and running time to achieve a comparable PSNR.

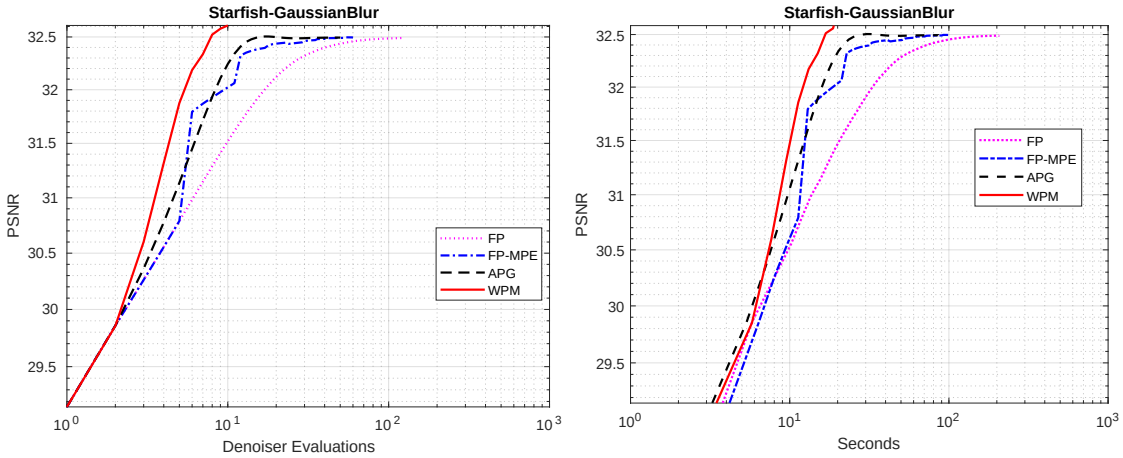
Examining the performance of the algorithms further, we run them on eight additional images tested in [5]. For each image, we run the FP method with 200 denoiser evaluations and take the final PSNR as a benchmark. Then we examine how many denoiser evaluations are needed for APG, FP-MPE, and WPM, to achieve a similar PSNR. The results are listed in Table 3.1. Evidently, with the exception of "Boats" and "House" in the deblurring task, we observe that WPM requires the smallest number of denoiser evaluations to achieve a comparable PSNR, demonstrating its efficiency for solving RED. Additionally, we present the recovered results of the "Starfish" and "Leaves" images from deblurring with uniform and Gaussian blurs, respectively, and the "Butterfly" image from super-resolution in Figure 3.1 to visualize the effectiveness of RED solved by WPM.

3.3 Conclusion

In this chapter, we propose a general framework for RED called weighted proximal methods (WPMs). By setting $\boldsymbol{B}_k = \alpha \boldsymbol{I}$ and $a_k = 1$, we retrieve the FP and APG methods. However, by choosing the weighting to be an approximation to the Hessian of $\alpha R(\boldsymbol{x})$, we obtain a more efficient algorithm. The experiments



(a) Deblurring with uniform blur.



(b) Deblurring with Gaussian blur.

Figure 3.2: PSNR versus denoiser evaluations (left column) and CPU time (right column) for deblurring the "Starfish" image.

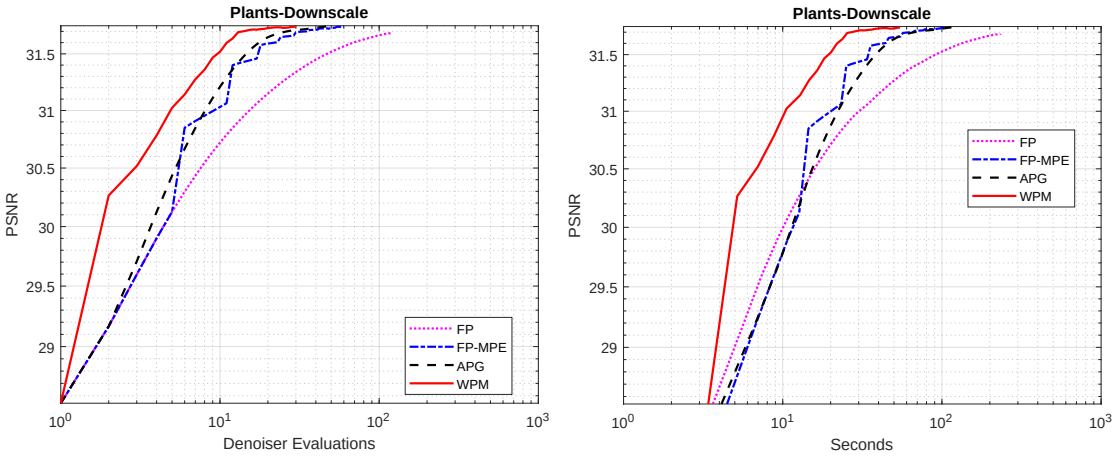


Figure 3.3: PSNR versus denoiser evaluations (left) and CPU time (right) for super-resolution of the "Plants" image.

on image deblurring and super-resolution tasks demonstrate that WPM with a simple and inexpensive approximation to the Hessian can substantially reduce the overall number of denoiser evaluations in the

Table 3.1: Denoiser evaluations required to attain a similar PSNR. The first and second rows per each image refer to image deblurring and the third row refers to super-resolution. The minimal number of denoiser evaluations is marked in bold.

	FP-MPE	APG	WPM
Butterfly	54	34	**25**
	54	26	**17**
	80	51	**26**
Boats	24	**20**	21
	60	34	**22**
	36	20	**12**
House	24	**18**	19
	62	26	**25**
	18	15	**10**
Parrot	39	30	**20**
	52	40	**36**
	49	31	**28**
Lena	48	34	**29**
	47	16	**15**
	37	26	**18**
Barbara	14	12	**11**
	48	23	**16**
	17	15	**11**
Peppers	42	29	**22**
	41	40	**34**
	38	30	**28**
Leaves	50	41	**34**
	36	18	**14**
	60	41	**12**

recovery process, usually resulting in significant speedup. In future work we aim to design better Hessian approximations in order to accelerate the computation further.

Chapter 4

On Adapting Nesterov's Scheme to Accelerate Iterative Methods for Linear Problems

In this chapter, we report our work on adapting Nesterov's scheme to accelerate iterative methods for linear problems, based on the following paper submitted for publication:

- **Tao Hong** and Irad Yavneh, *On Adapting Nesterov's Scheme to Accelerate Iterative Methods for Linear Problems,* arXiv:2102.09239, Submitted to Numerical Linear Algebra with Applications.

4.1 Introduction

Many scientific computing applications require solving linear systems of equations of the form [9, 59]:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}, \tag{4.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a sparse, large-scale, ill-conditioned matrix. For example, $\boldsymbol{A}$ may be a discretization of an elliptic partial differential equation (PDE) or system. Because direct solvers are relatively expensive, especially for 3D problems, iterative methods are often preferred, e.g., successive over-relaxation or multigrid. These are very often used to advantage as preconditioners for Krylov subspace acceleration methods. The LOPCG method for eigenvalue problems[60, 61] is an alternative acceleration method, which uses a linear combination of two consecutive iterates, together with a preconditioned residual, to construct the next iterate such that the residual norm at the current step is minimized. Motivated by Nesterov's scheme developed in the framework of convex optimization, we consider adapting this approach to accelerating iterative methods for linear problems, using a fixed optimal scalar parameter for which we derive an explicit formula.

Nesterov's well-known scheme for accelerating gradient descent for convex optimization problems [62, 50] has attracted much attention in the optimization community over the years. Given an unconstrained optimization problem,

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{R}^N} F(\boldsymbol{x}),$$

the kth iteration of Nesterov's scheme is defined as follows:

$$\begin{aligned} \boldsymbol{x}_{x+1} &= \boldsymbol{B}(\boldsymbol{y}_k), \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_{k+1} + c_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \end{aligned} \tag{4.2}$$

where $\boldsymbol{y}_k \in \mathbb{R}^N$ is an intermediate iterate and $\boldsymbol{B}(\cdot)$ represents some iterative method, e.g., gradient descent. In the classical method, an optimal analytical sequence c_k is introduced for convex problems with $\boldsymbol{B}(\cdot)$ defined by gradient descent [50]. We refer the reader to [63, 64] and references therein for further discussion of Nesterov's scheme and the choice of c_k with some restarting strategies. Over the years, Nesterov's scheme has been extended to more general $\boldsymbol{B}(\cdot)$ operators, including the proximal operator [65], coordinate descent [66], alternating least squares [67], second-order methods [68], and stochastic methods [69].

In this chapter, we focus on linear systems (4.1) and adapt (4.2) to acceleration of stationary iterative methods. Since the problem of interest is linear, we simplify (4.2) as follows:

$$\begin{aligned} \boldsymbol{x}_{x+1} &= \boldsymbol{B}\boldsymbol{y}_k + \text{Constant}, \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_{k+1} + c_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \end{aligned} \tag{4.3}$$

where we use the fixed matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ (called iteration matrix) plus an appropriate constant vector to represent the operator $\boldsymbol{B}(\cdot)$. Below, we show that if all the eigenvalues of $\boldsymbol{B}$ are real, then an optimal fixed c_k can be found analytically, depending on the smallest and largest eigenvalues of $\boldsymbol{B}$. We note that a similar analysis was performed in [70] for the restricted case of gradient descent and the assumption that all eigenvalues of $\boldsymbol{B}$ are real and of the same sign. Furthermore, [70] also studied the complex eigenvalues case and proved a lower and upper ACF bound when all complex eigenvalues lie in a prescribed rectangular region. Since the additional computations in (4.3) are negligible, we suggest that (4.3) may in some cases prove to be competitive for accelerating stationary iterative methods. Because c_k will be fixed, we discard the subscript k in the rest of the chapter. Interestingly, we find that the results developed for $\boldsymbol{B}$ whose eigenvalues are real are also valid for $\boldsymbol{B}$ which has complex eigenvalues within a relatively large domain. Moreover, a "valid" region in the complex plane, defined as a region where existence of complex eigenvalues of $\boldsymbol{B}$ does not influence the optimal c or the asymptotic convergence factor (ACF), is explicitly identified, dependent on the smallest and largest real eigenvalues. Furthermore, we compare Nesterov's scheme to a "restricted-information" (RI) Chebyshev acceleration, where we choose the optimal parameters based on the same information as required for Nesterov's scheme, i.e., the smallest and largest real eigenvalues of $\boldsymbol{B}$ [71]. Our comparison indicates that Nesterov's scheme is more robust than RI Chebyshev acceleration with respect to the existence of complex eigenvalues for $\boldsymbol{B}$.

The rest of this chapter is organized as follows. The analytical derivation of c for $\boldsymbol{B}$ with only real eigenvalues is given in Section 4.2. The robustness of these results in cases where $\boldsymbol{B}$ also has complex eigenvalues is studied in Section 4.3, including a comparison to RI Chebyshev acceleration. In Section 4.4 we demonstrate the usefulness of the proposed method in accelerating multigrid solution of some second-order elliptic boundary value problems, and conclusions and future work are summarized in Section 4.5.

4.2 Optimal Acceleration

Representing $\boldsymbol{y}_k$ by the previous iterates $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$, we obtain

$$\boldsymbol{x}_{k+1} = \boldsymbol{B}\left(\boldsymbol{x}_k + c\left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\right)\right) + \text{Constant}. \tag{4.4}$$

Denote $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$, where $\boldsymbol{x}^*$ is the sought solution. Subtracting $\boldsymbol{x}^*$ from both sides of (4.4) and substituting $\boldsymbol{B}\boldsymbol{x}^* + \text{Constant} = \boldsymbol{x}^*$, we arrive at

$$\boldsymbol{e}_{k+1} = \boldsymbol{B}\left(\left(1+c\right)\boldsymbol{e}_k - c\boldsymbol{e}_{k-1}\right). \tag{4.5}$$

Denote $\boldsymbol{E}_k = \begin{bmatrix} \boldsymbol{e}_k \\ \boldsymbol{e}_{k-1} \end{bmatrix}$ and rewrite (4.5) as $\boldsymbol{E}_k = \begin{bmatrix} (1+c)\boldsymbol{B} & -c\boldsymbol{B} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{E}_{k-1}$. Then, the asymptotic convergence factor ACF of (4.3) is given by $\rho(\boldsymbol{\Gamma})$, the spectral radius of $\boldsymbol{\Gamma} \triangleq \begin{bmatrix} (1+c)\boldsymbol{B} & -c\boldsymbol{B} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$. Evidently, if there is a c yielding $\rho(\boldsymbol{\Gamma}) < \rho(\boldsymbol{B})$, then (4.3) provides acceleration. The following derivation produces a c which minimizes $\rho(\boldsymbol{\Gamma})$, and this optimal c can easily be calculated analytically if all the eigenvalues of $\boldsymbol{B}$ are real and its smallest and largest eigenvalues are known. We note that the cost of obtaining the required information of the smallest and largest eigenvalues of $\boldsymbol{B}$ may be low for certain specific linear problems and stationary iterative methods, as discussed in Section 4.4.

Denote by λ and $\begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix}$ an eigenvalue and corresponding eigenvector of $\boldsymbol{\Gamma}$. We have $\begin{bmatrix} (1+c)\boldsymbol{B} & -c\boldsymbol{B} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix}$, yielding

$$\left(1 + c - \frac{c}{\lambda}\right)\boldsymbol{B}\boldsymbol{v}_1 = \lambda \boldsymbol{v}_1, \tag{4.6}$$

where the trivial case, $\lambda = 0$, is omitted. Let $b \in \mathbb{R}$ denote some eigenvalue of $\boldsymbol{B}$ corresponding to λ. From (4.6), we have $\left(1 + c - \frac{c}{\lambda}\right) b = \lambda$, yielding $\lambda^2 - (1+c)b\lambda + cb = 0$, with solutions $\lambda_1(c,b) = \frac{1}{2}\left[(1+c)b + \sqrt{(1+c)^2 b^2 - 4cb}\right]$, and $\lambda_2(c,b) = \frac{1}{2}\left[(1+c)b - \sqrt{(1+c)^2 b^2 - 4cb}\right]$. We use

$$b_{cr} = \frac{4c}{(1+c)^2} \tag{4.7}$$

to denote the "critical" value of b for which the square root term in $\lambda_{1,2}$ vanishes for a given c.

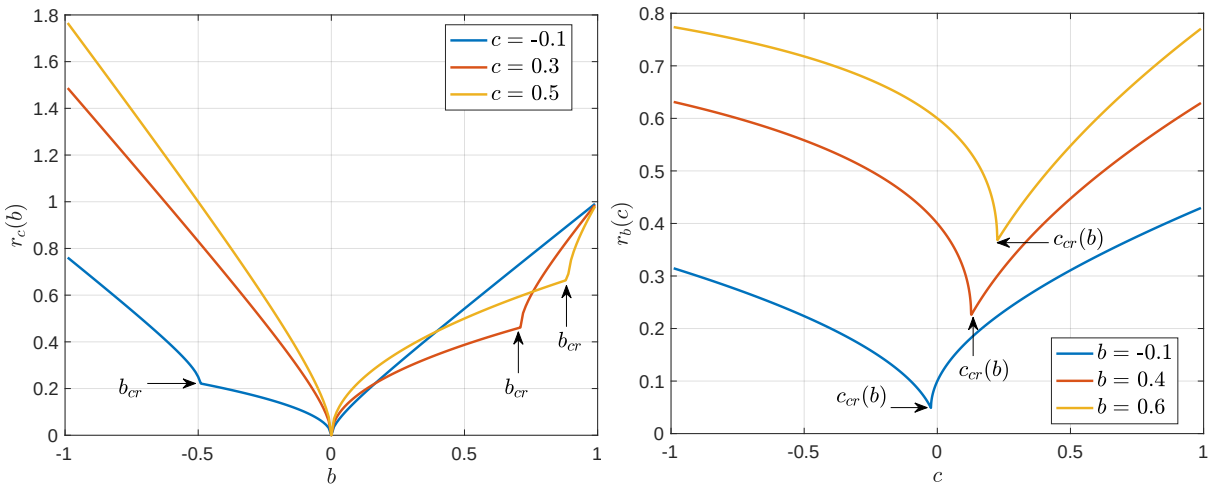
Remark 4.1. $\lambda_{1,2}$ are complex if and only if b and c are of the same sign and $0 < |b| < |b_{cr}|$.

Denote

$$r(c,b) \triangleq \max\left\{\left|\lambda_1(c,b)\right|, \left|\lambda_2(c,b)\right|\right\}. \tag{4.8}$$

Without loss of generality, given that $\boldsymbol{B}$ represents the iteration matrix of a convergent method with real eigenvalues, we assume that its eigenvalues are ordered as $-1 < b_1 \leq b_2 \leq \cdots \leq b_N < 1$. The optimal c, given by $c^* = \arg\min_c \max_b r(c,b)$, is derived next. To simplify the presentation, we assume $b \neq 0$ throughout, since in this case $r(c,0) = 0$, so a vanishing b has no influence on c^*.

Lemma 4.2.1. $|c^*| < 1$.



(a) (b)

Figure 4.1: (a): The value of $r_c(b)$ as a function of c and $b \in (-1,1)$. (b): The value of $r_b(c)$ as a function of b and $c \in (-1,1)$.

Proof. We prove this lemma by showing that $r(c,b) \triangleq \max\{|\lambda_1(c,b)|, |\lambda_2(c,b)|\} > |b|$ for any nonzero b if $|c| \geq 1$, so such a c slows down convergence.

For $c > 1$, the first term in $\lambda_{1,2}$ already satisfies $\left|\frac{1}{2}(1+c)b\right| > |b|$ so either λ_1 or λ_2 (or both) must be larger than b. In the borderline case of $c = 1$, the first term equals b, but in this case the second term cannot vanish except for $b = 0$, which is excluded, so again either λ_1 or λ_2 must be strictly larger than b.

For $c \leq -1$ and $b > 0$, $r(c,b) = -\lambda_2(c,b)$ and the square root term is real. To show that $-\lambda_2(c,b) > b$, we need to prove $\sqrt{(1+c)^2b^2 - 4cb} > (3+c)b$. Squaring both sides of this inequality, and simplifying, results in $(2+c)b < -c$, which is satisfied, because the left side is smaller than one in this regime, while the right side is at least one.

For $c \leq -1$ and $b < 0$, if $b > b_{cr}$ then the square root term is imaginary by Remark 4.1, resulting by Remark 4.2 in $r(c,b) = \sqrt{cb} \geq \sqrt{-b} > |b|$. Otherwise, the square root is real and $r(c,b) = \lambda_1(c,b)$. To show that $\lambda_1(c,b) > |b|$, we need to prove that $\sqrt{(1+c)^2b^2 - 4cb} > (-c-3)b$. This is indeed satisfied, due to the fact that the right side is negative because $-1 < b \leq b_{cr}$ implies $c < -3 - 2\sqrt{2}$. □

In light of Lemma 4.2.1, we henceforth restrict $|c|$ to be smaller than 1. This and the fact that b is real imply

$$r(c,b) = \frac{1}{2}\left|(1+c)b + sgn(b)\sqrt{(1+c)^2b^2 - 4cb}\right|, \tag{4.9}$$

where $sgn(\cdot)$ is the sign function.

Remark 4.2. When the square root term is imaginary (see Remark 4.1), we obtain $r(c,b) = |\lambda_1| = |\lambda_2| = \sqrt{cb}$.

For convenience, we henceforth use $r_c(b)$ (respectively, $r_b(c)$) to denote $r(c,b)$ considered as a single-variable function with a fixed c (respectively, fixed b). First we show that the maximal $r_c(b)$ depends only on the extreme eigenvalues of b.

Lemma 4.2.2. *For $|c| < 1$, $r_c(b)$ is maximized at either b_1 or b_N and has no local maximum.*

Proof. To prove that $r_c(b)$ has no local maximum for any $|c| < 1$, we first observe by Remarks 4.1 and 4.2 that in the range where $\lambda_{1,2}$ are complex, $r_c(b) = \sqrt{cb}$ is strictly increasing (respectively, decreasing) for positive (respectively, negative) b, and therefore has no local maximum in this range. Outside this range, the derivative of $r_c(b)$ is discontinuous (only) at $b = 0$ and $b = b_{cr}$. Elsewhere, it is given by

$$r_c'(b) = \frac{1}{2}\left(sgn(b)(1+c) + \frac{(1+c)^2 b - 2c}{\sqrt{(1+c)^2 b^2 - 4cb}} \right). \tag{4.10}$$

Note that for $b = b_{cr}$, the numerator in the second term is given by $2c$. It follows that $sgn(r_c'(b_{cr}^-)) = sgn(r_c'(b_{cr}^+))$ regardless of whether c (hence also b_{cr}) is positive or negative, so b_{cr} cannot be a local maximum. It remains to show that $r_c'(b)$ cannot vanish. We show this by comparing the squares of the first and second terms in the brackets on the right side of (4.10):

$$(1+c)^2 - \left(\frac{(1+c)^2 b - 2c}{\sqrt{(1+c)^2 b^2 - 4cb}} \right)^2 = -\frac{4c^2}{(1+c)^2 b^2 - 4cb}.$$

This expression only vanishes for $c = 0$, for which (4.10) implies that $r_c'(b) \neq 0$. Hence, $r_c'(b)$ cannot vanish for any $b \in (-1,1)$. In Figure 4.1(a), we numerically show the value of $r_c(b)$ as a function of c and $b \in (-1,1)$ that we clearly see the discontinuous at $b = 0$ and $b = b_{cr}$. □

Next, we fix b and minimize $r_b(c)$. To this end we first identify the range of c for which $\lambda_{1,2}$ are real.

Remark 4.3. Define the critical $c \in (-1,1)$, for which the square-root term in (4.9) vanishes, by

$$c_{cr}(b) = \frac{1 - \sqrt{1-b}}{1 + \sqrt{1-b}}.$$

Then, by the solution of (4.6), $\lambda_{1,2}$ are complex if and only if $\{c > c_{cr}(b) \text{ and } b > 0\}$ or $\{c < c_{cr}(b) \text{ and } b < 0\}$.

We note that $c_{cr}(b)$ is continuous ($\lim_{b\to 0} c_{cr}(b) = 0$) and monotonically increasing on $b \in (-1,1)$, with $c_{cr}(b) \in (-3+2\sqrt{2}, 1)$. We next show that c_{cr} is the optimal value of c for minimizing $r_b(c)$.

Lemma 4.2.3.

$$c_{cr}(b) = \operatorname*{argmin}_c r_b(c).$$

Proof. Consider first the case $0 < b < 1$. In this regime, by Remark 4.3, $r_b(c) = \sqrt{cb}$ for $c > c_{cr}(b) > 0$, so $r_b'(c) > 0$. On the other hand, for $c < c_{cr}(b)$ we have $r_b(c) = \frac{1}{2}\left((1+c)b + \sqrt{(1+c)^2 b^2 - 4cb}\right)$, hence,

$$\begin{aligned} r_b'(c) &= \frac{1}{2}\left(b + \frac{(1+c)b^2 - 2b}{\sqrt{(1+c)^2 b^2 - 4bc}} \right) \\ &= \frac{b}{2}\left(1 - \sqrt{1 + \frac{4(1-b)}{(1+c)^2 b^2 - 4bc}} \right) < 0. \end{aligned}$$

It follows that $r_b(c)$ is minimized in this regime for $c = c_{cr}(b)$. The derivation for $-1 < b < 0$ is analogous and we omit the details. In Figure 4.1(b), we numerically show the value of $r_b(c)$ as a function of c and $b \in (-1,1)$ that $c_{cr}(b)$ is the minimizer. □

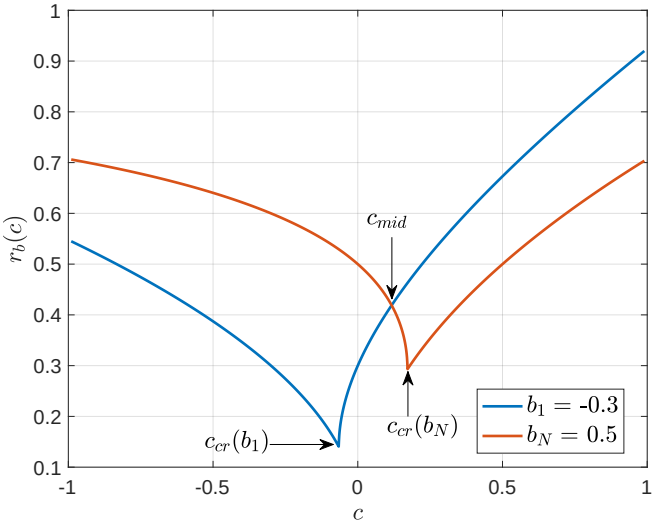


Figure 4.2: The curves of $r_b(c)$ as a function of c with $b = b_1 = -0.3$ and $b = b_N = 0.5$.

Summarizing, we observe that, by Lemma 4.2.2, c^* depends only on b_1 and b_N, and therefore, by Lemma 4.2.3, there are only three possible values of c to consider as candidates for c^*: $c_{cr}(b_1)$, $c_{cr}(b_N)$, and the value of c which minimizes $r(c,b)$ subject to $r(c,b_1) = r(c,b_N)$. We map out the regions where each of these three options yields the optimal c in the following theorem.

Theorem 4.1. *Let $-1 < b_1 \le b_N < 1$. Then, the optimal coefficient c^* is given by*

$$c^* = c_{cr}(g(b_1, b_N)),$$

where

$$g(b_1,b_N) = \begin{cases} b_N, & b_N \ge -3b_1, \\ -\frac{8b_N b_1(b_1+b_N)}{(b_1-b_N)^2}, & -\frac{1}{3}b_1 < b_N < -3b_1, \\ b_1, & b_N \le -\frac{1}{3}b_1, \end{cases}$$

yielding the corresponding asymptotic convergence factor

$$r^* = \begin{cases} 1-\sqrt{1-b_N}, & b_N \ge -3b_1, \\ r(c^*,b_1) = r(c^*,b_N), & -\frac{1}{3}b_1 < b_N < -3b_1, \\ \sqrt{1-b_1}-1, & b_N \le -\frac{1}{3}b_1. \end{cases}$$

Proof. In light of Lemmas 4.2.2 and 4.2.3, c^* must satisfy one of the following three conditions: ① $c^* = c_{cr}(b_N)$, ② $r(c^*,b_1) = r(c^*,b_N)$, or ③ $c^* = c_{cr}(b_1)$, denoted c_{top}, c_{mid} and c_{bot}, respectively. The choice is determined by the maximal r obtained amongst the three. We assume for simplicity $|b_1| \le b_N$, and remark that the complementary case, $|b_N| < -b_1$, is obtained analogously so the details are omitted.

Note first that for $c < c_{cr}(b_N)$ we have $r_c(b_N) = \frac{1}{2}\left[(1+c)b_N + \sqrt{(1+c)^2 b_N^2 - 4cb_N}\right]$, which is larger than b_N if $c < 0$, so it follows that $c^* \in [0,1)$. We first consider the case $b_1 \ge 0$. For $c < c_{cr}(b)$, we have for any b,

$$r_c'(b) = \frac{1}{2}\left[(1+c) + \frac{(1+c)^2 b - 2c}{\sqrt{(1+c)^2 b^2 - 4cb}}\right] > 0.$$

On the other hand, for $c > c_{cr}(b)$, we have $r_c(b) = \sqrt{cb}$ so again $r_c'(b) > 0$. The monotonicity of $r_c(b)$

implies $r_c(b_N) \geq r_c(b_1)$, with equality achieved only for $b_1 = b_N$. It follows that $c^* = \text{argmin}_c\, r_{b_N}(c)$ and case ① holds, i.e., $c^* = c_{top} \triangleq c_{cr}(b_N)$. The corresponding ACF is $r_{c_{top}}(b_N) = 1 - \sqrt{1-b_N} = \frac{2c_{top}}{1+c_{top}}$.

It remains to consider the case $b_1 < 0$, whereby $r_c(b_1) = -\frac{1}{2}\left[(1+c)b_1 - \sqrt{(1+c)^2 b_1^2 - 4cb_1}\right]$. First, we identify the subrange in $b_1 < 0$ yielding $r_{c_{top}}(b_N) \geq r_{c_{top}}(b_1)$, leading us again to case ①. Denote for clarity $r_{top} = r_{c_{top}}(b_N)$. Then case ① holds if b_1 satisfies the following inequality,

$$\begin{aligned}
r_{top} &\geq -\tfrac{1}{2}\left[(1+c_{top})b_1 - \sqrt{(1+c_{top})^2 b_1^2 - 4c_{top}b_1}\right], \\
2r_{top} + (1+c_{top})b_1 &\geq \sqrt{(1+c_{top})^2 b_1^2 - 4c_{top}b_1} \quad \text{(square both sides and divide by 4)}, \\
r_{top}^2 + r_{top}(1+c_{top})b_1 &\geq -c_{top}b_1, \\
b_1 &\geq -\tfrac{r_{top}^2}{r_{top}(1+c_{top})+c_{top}}.
\end{aligned}$$

Using $r_{top} = \frac{2c_{top}}{1+c_{top}}$ and $b_N = \frac{4c_{top}}{(1+c_{top})^2}$, we obtain the condition $b_1 \geq -\frac{1}{3}b_N$.

Finally, we discuss the remaining range of $b_1 < 0$, i.e., $b_1 \in [-b_N, -\frac{1}{3}b_N)$, where case ① does not hold. The fact that $c^* \in [0,1)$ rules out case ③, leaving case ②, that is,

$$-(1+c^*)b_1 + \sqrt{(1+c^*)^2 b_1^2 - 4c^* b_1} \quad = \quad (1+c^*)b_N + \sqrt{(1+c^*)^2 b_N^2 - 4c^* b_N}.$$

Simplifying by repeatedly putting the square root terms on one side and squaring both sides, we get

$$\left[2b_1 b_N (b_1+b_N)\right](c^*)^2 + \left[4b_1 b_N (b_1+b_N) + (b_1-b_N)^2\right] c^* + 2b_1 b_N (b_1+b_N) = 0.$$

Using $c^* \in [0,1)$, the valid solution of this quadratic equation is

$$c^* = c_{mid} \quad \triangleq \quad \tfrac{1-\sqrt{1-g(b_1,b_N)}}{1+\sqrt{1-g(b_1,b_N)}},$$

where $g(b_1,b_N) = -\frac{8b_1 b_N (b_1+b_N)}{(b_1-b_N)^2}$. In Figure 4.2, we numerically show the curves of $r_b(c)$ as a function of c and b corresponding to ② that we clearly see c_{mid} is the solution for $\min_c \max\{r_{b_1}(c), r_{b_N}(c)\}$.

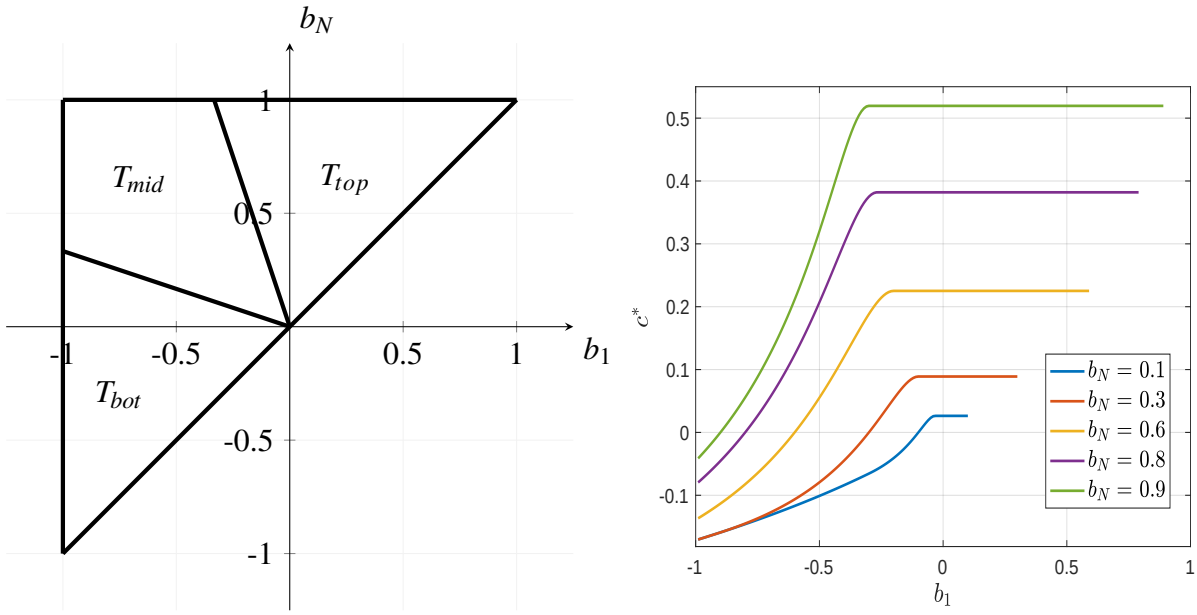
As noted above, the derivation for $(b):\ |b_N| \leq -b_1$ is analogous and we omit the details. □

Remark 4.4. Nesterov's scheme can converge even for some $\boldsymbol{B}$ whose spectral radii are larger than 1. In our setting, we can relax our assumption from $-1 < b_1 \leq b_N < 1$ to $-3 < b_1 \leq b_N < 1$, and Theorems 4.1 and 4.2 remain valid.

Henceforth use c_{top}, c_{mid}, and c_{bot}, as defined in the proof of Theorem 4.1, to denote the optimal coefficient c^* corresponding to regime T_{top} ($b_N \geq -3b_1$), T_{mid} ($b_N \in (-\frac{1}{3}b_1, -3b_1)$) and T_{bot} ($b_N \leq -\frac{1}{3}b_1$) of Theorem 4.1, respectively, (see Figure 4.3(a)). Also, we numerically show the value of c^* as a function of b_1 and b_N in Figure 4.3(b). The flat parts of the curves in Figure 4.3(b) imply that c^* only depends on b_N in this regime, and the coincidence of the curves corresponding to $b_N = 0.1$ and $b_N = 0.3$ for $b_1 \to -1$ indicates that c^* only depends on b_1 in that regime.

This concludes our derivation for the case where the eigenvalues of $\boldsymbol{B}$ are all real. In the next section, we extend our results to certain cases where some of the eigenvalues of $\boldsymbol{B}$ are complex.

To conclude this section, we numerically evaluate the savings in computations that are provided by employing Nesterov's scheme (4.3). Without loss of generality, we assume $b_N \geq -b_1$. Then, the



(a) (b)

Figure 4.3: (a): Three regimes for determining c^* and r^*. (b): The value of c^* as a function of b_1 and b_N.

Acceleration Ratio (AR), defined as the ratio of the number of iterations required without acceleration to the number of iterations required with acceleration, in order to reach the same accuracy, is asymptotically given by

$$AR = \frac{\log r^*}{\log b_N}. \tag{4.11}$$

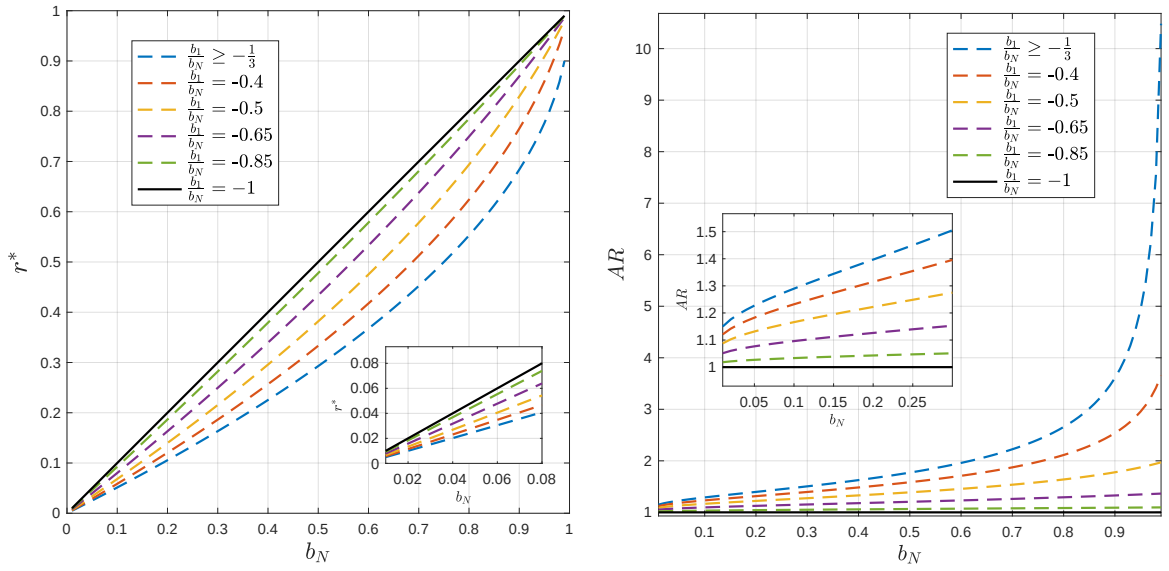
Figure 4.4 shows the relation between r^* and b_N in panel (a), and the acceleration ratio AR in panel (b), for a range of b_1 and b_N values. In each curve we fix the ratio b_1/b_N and vary $b_N \in (0,1)$. For $b_1/b_N = -1$, there is no acceleration, as discussed above, but for larger ratios we observe acceleration which improves as b_1/b_N increases, and rapidly improves as $b_N \to 1$ (that is, when the unaccelerated iterations converge slowly). Note that all values of r^* and AR are identical for $b_1/b_N \geq -1/3$.

4.3 Complex Eigenvalues

Theorem 4.1 is formulated under the assumption that all the eigenvalues of $\boldsymbol{B}$ are real. We next state sufficient conditions under which Theorem 4.1 continues to hold even though some of the eigenvalues of $\boldsymbol{B}$ are complex.

Theorem 4.2. *Assume that, in addition to the real eigenvalues $-1 < b_1 \leq \ldots \leq b_N < 1$ of $\boldsymbol{B}$ as assumed in Theorem 4.1, $\boldsymbol{B}$ also has complex eigenvalues. Denote the complex eigenvalues generically by $b^c = \bar{b}^c e^{j\theta}$, where j is the imaginary unit, $\bar{b}^c$ is the modulus, and $\theta \in (-\pi, \pi]$ is the argument. Then, c^* and r^* of Theorem 4.1 remain valid if for every one of the complex eigenvalues of $\boldsymbol{B}$ the modulus satisfies*

$$\bar{b}^c \leq \begin{array}{ll} \frac{1}{3}b_N & c^* = c_{top} \\ \min(|b_1|, |b_N|) & c^* = c_{mid} \\ -\frac{1}{3}b_1 & c^* = c_{bot} \end{array}.$$



(a) (b)

Figure 4.4: (a): The ACF achieved by Nesterov's scheme as a function of b_N and the ratio $\frac{b_1}{b_N}$. (b): The acceleration ratio AR (cf. (4.11)) as a function of b_N and the ratio $\frac{b_1}{b_N}$.

Proof. As in the proof of Theorem 4.1, we assume $|b_1| \leq b_N$ and remark that the complement, $|b_N| < -b_1$, is proved analogously, omitting the details.

A complex eigenvalue b^c does not influence the ACF, when c^* is chosen according to Theorem 4.1, if $r(c^*, b^c) \leq r^*$, where $r^* = r_{c^*}(b_N)$, with $c^* = c_{top}$ or c_{mid} as dictated by Theorem 4.1. This yields the explicit condition:

$$\frac{1}{2}\left|(1+c^*)b^c \pm \sqrt{(1+c^*)^2(b^c)^2 - 4c^* b^c}\right| \leq r^*. \tag{4.12}$$

Substituting $b^c = \bar{b}^c e^{j\theta}$ into (4.12) and simplifying, we obtain

$$\frac{1+c^*}{2}\bar{b}^c \left|1 \pm \sqrt{1+\bar{b}e^{j\bar{\theta}}}\right| \leq r^*, \tag{4.13}$$

where we have introduced the notation $\bar{b} = \frac{4c^*}{(1+c^*)^2 \bar{b}^c} \geq 0$ and $\bar{\theta} = \pi - \theta$. Since the term of the square root is complex and the sign of its real part is always positive, (4.13) becomes

$$\frac{1+c^*}{2}\bar{b}^c \left|1 + \sqrt{1+\bar{b}e^{j\bar{\theta}}}\right| \leq r^*. \tag{4.14}$$

Denote $\phi(\bar{\theta}) = \left|1 + \sqrt{1+\bar{b}e^{j\bar{\theta}}}\right|$. To bound the left-hand side from above, we derive a monotonicity property for $\phi(\bar{\theta})$ in the period $\bar{\theta} \in (-\pi, \pi]$. With Euler's formula, we can rewrite $\phi(\bar{\theta})$ as $\sqrt{1 + 2\chi\cos\frac{\vartheta}{2} + \chi^2}$, where $\chi = \left(1 + 2\bar{b}\cos\bar{\theta} + \bar{b}^2\right)^{\frac{1}{4}}$ and $\vartheta \in (-\frac{\pi}{2}, \frac{\pi}{2}) := \arctan\frac{\bar{b}\sin\bar{\theta}}{1+\bar{b}\cos\bar{\theta}}$, and then it is evident that $\phi(-\bar{\theta}) = \phi(\bar{\theta})$. Since $\phi(\bar{\theta})$ is an even function, we consider its monotonicity in the half period $\bar{\theta} \in (0, \pi)$. Using $\sqrt{x+jy} = \sqrt{\frac{x+\sqrt{x^2+y^2}}{2}} + sgn(y)\, j\sqrt{\frac{-x+\sqrt{x^2+y^2}}{2}}$ and $\phi(\bar{\theta}) = \left|1 + \sqrt{1 + \bar{b}\cos\bar{\theta} + j\bar{b}\sin\bar{\theta}}\right|$, we get $\phi(\bar{\theta}) = \left|1 + \sqrt{\frac{1+\bar{b}\cos\bar{\theta}+\chi^2}{2}} + j\sqrt{\frac{-(1+\bar{b}\cos\bar{\theta})+\chi^2}{2}}\right|$. For convenience, we consider the monotonicity of $\phi^2(\bar{\theta}) =$

$1+\sqrt{2+2\bar{b}\cos\bar{\theta}+2\chi^2}+\chi^2$ instead of $\phi(\bar{\theta})$. Since

$$\frac{\partial\phi^2(\bar{\theta})}{\partial\bar{\theta}} = 2\chi\frac{\partial\chi}{\partial\bar{\theta}} + \frac{-\bar{b}\sin\bar{\theta}+2\chi\frac{\partial\chi}{\partial\bar{\theta}}}{\sqrt{2(1+\bar{b}\cos\bar{\theta})+2\chi^2}} < 0, \quad \bar{\theta}\in(0,\pi)$$

(using $\frac{\partial\chi}{\partial\bar{\theta}}<0$, $\bar{b}\sin\bar{\theta}\geq 0$, and $\chi\geq 0$), we find that $\phi(\bar{\theta})$ is monotonically decreasing in $\bar{\theta}\in(0,\pi)$. Together with the fact that $\phi(\theta)$ is an even function, we have $r(c^*,\bar{b}^c)\leq r(c^*,b^c)\leq r(c^*,-\bar{b}^c)$.

From Theorem 4.1, we know that $r^* = r_{c_{top}}(b_N)\geq r_{c_{top}}(b_i)$ for any real $b_i\geq -\frac{1}{3}b_N$ when T_{top} is the relevant domain. Thus, for any complex eigenvalue with $\bar{b}^c\leq\frac{1}{3}b_N$, we obtain $r(c_{top},b^c)\leq r^*$. Since $\phi(\theta)$ is monotonically increasing (respectively, decreasing) in $\theta\in(0,\pi)$ (respectively, $\theta\in(-\pi,0)$), the upper bound on $r(c_{top},b^c)$ becomes tight when the argument is close to π (respectively, $-\pi$). This means that if the modulus is larger than $\frac{1}{3}b_N$ then the argument should be close to 0 to enforce $r(c_{top},b^c)\leq r^*$. Evidently, the conclusion drawn here is valid also for $c^*=c_{mid}$, as it is implied by the monotonicity properties of $\phi(\theta)$.

As noted above, the proof for the complementary domain $|b_N|<-b_1$ is analogous and so the details are omitted. □

The sufficient robustness conditions of Theorem 4.2 are tight for $|\theta|\to\pi$ when $b_N>|b_1|$, and can be relaxed increasingly as $|\theta|$ decreases towards 0. Similarly (and in a symmetrical manner), the conditions are tight for $\theta\to 0$ when $b_N<|b_1|$, and can be relaxed increasingly as $|\theta|$ increases towards π. This follows from the proof, and it is demonstrated by numerical examples in Figure 4.5. The blue region marks the domain where $r_{c^*}(b^c)\leq r^*$, that is, the results of Theorem 4.1 hold so long as all the complex eigenvalues of $\boldsymbol{B}$ lie within this region. The disk enclosed by the red circle is the subdomain covered by Theorem 4.2.

Comparison of Nesterov's Scheme to RI Chebyshev Acceleration

A classical approach to accelerating convergence of stationary iteration schemes is by polynomial acceleration, whereby successive iterates are combined linearly with skillfully selected coefficients [71]:

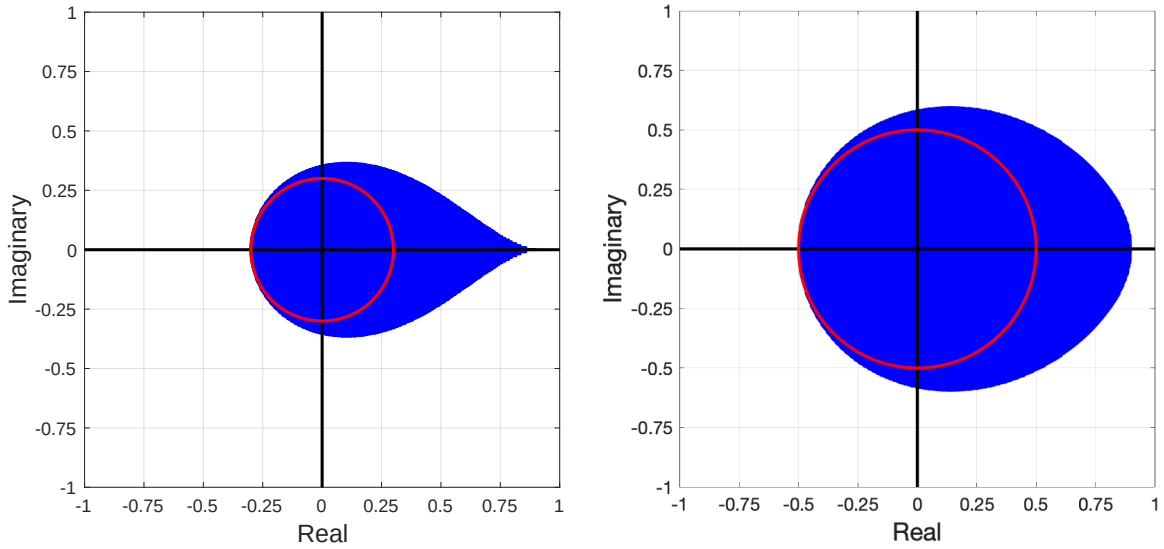
$$\bar{\boldsymbol{x}}_k = \sum_{n=0}^{k}\alpha_{k,n}\boldsymbol{x}_n, \tag{4.15}$$

where $\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{x}_{n-1}+\text{Constant}$ and $\sum_{n=0}^{k}\alpha_{k,n}=1$. Denoting $\bar{\boldsymbol{e}}_k=\bar{\boldsymbol{x}}_k-\boldsymbol{x}^*$, we obtain

$$\bar{\boldsymbol{e}}_k = \left(\sum_{n=0}^{k}\alpha_{k,n}\boldsymbol{B}^n\right)\boldsymbol{e}_0\,,$$

where $\boldsymbol{e}_0=\boldsymbol{x}_0-\boldsymbol{x}^*$. The objective of minimizing the spectral radius $\rho\left(\sum_{n=0}^{k}\alpha_{k,n}\boldsymbol{B}^n\right)$ (which yields the asymptotic convergence factor ACF), is achieved by using the well-known Chebyshev polynomial. Applied in recursive form, this yields a scheme of the following form:

$$\begin{aligned}\boldsymbol{x}_1 &= \gamma(\boldsymbol{B}\boldsymbol{x}_0+\text{Constant})+(1-\gamma)\boldsymbol{x}_0,\\ \boldsymbol{x}_{k+1} &= \beta_{k+1}\left\{\gamma(\boldsymbol{B}\boldsymbol{x}_k+\text{Constant})+(1-\gamma)\boldsymbol{x}_k\right\}+(1-\beta_{k+1})\boldsymbol{x}_{k-1}.\end{aligned} \tag{4.16}$$



(a) $b_1 = -0.3$ and $b_N = 0.9$.

(b) $b_1 = -0.5$ and $b_N = 0.9$.

(c) $b_1 = -0.9$ and $b_N = 0.3$.

(d) $b_1 = -0.9$ and $b_N = 0.5$.

Figure 4.5: The complex domains defined in Theorem 4.2. The red circle has radius $|b_1|$ in panels (a) and (b) and $|b_N|$ in (c) and (d).

The optimal values of γ and β_{k+1} depend on the eigenvalues of $\boldsymbol{B}$, see details in [71]. Often, such information is not readily available. This motivates us to consider a "Restricted Information" scenario, where we assume that we are given only the smallest and largest real eigenvalues of $\boldsymbol{B}$, b_1 and b_N, as in the previous section. We refer to the scheme (4.16) that is based solely on this information as Restricted Information (RI) Chebyshev acceleration. Clearly, RI Chebyshev acceleration is optimal in the case where all the eigenvalues of $\boldsymbol{B}$ are real, because in that case we have all the required information. Indeed, in this case (4.16) converges as fast as Preconditioned Conjugate Gradients (PCG) with preconditioner $\boldsymbol{B}$, but requires less computation than PCG once b_1 and b_N are known [71]. In this case, Chebyshev acceleration is more efficient than Nesterov's scheme. However, it is interesting to note that we only need to know b_N for applying Nesterov's scheme if $b_1 \geq -\frac{1}{3}b_N$ (or, by a symmetric argument, we only need to know b_1 if $b_N \leq -\frac{1}{3}b_1$), so in these regimes less information is required—just the spectral radius of $\boldsymbol{B}$ which is often easy to compute approximately.

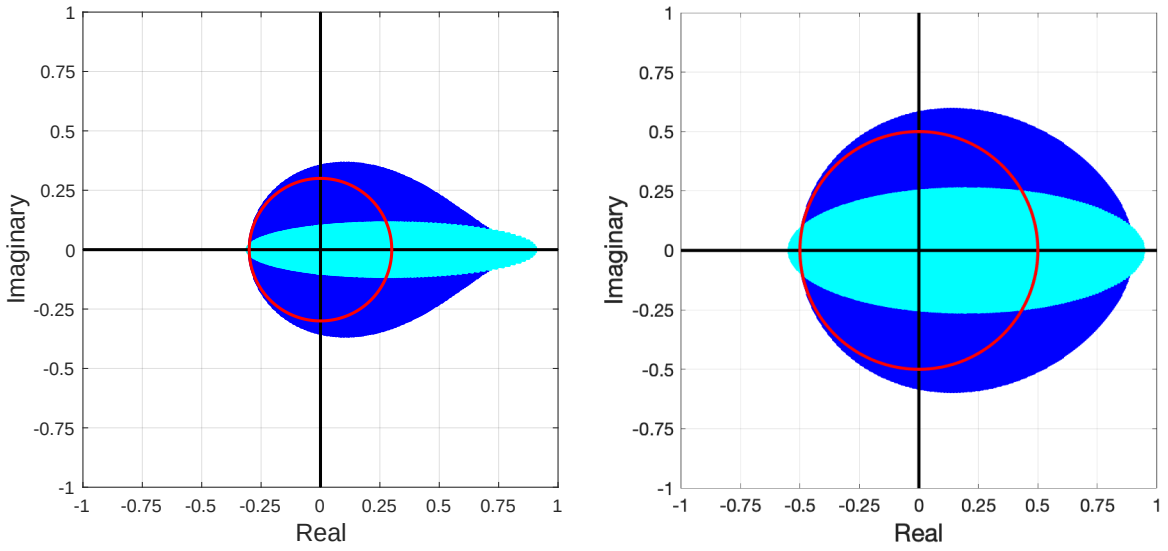
Next, we study and compare the performance of Nesterov's scheme and RI Chebyshev acceleration in cases where $\boldsymbol{B}$ has complex eigenvalues[2]. As noted above, if $\boldsymbol{B}$ has no complex eigenvalues then (4.16) is always faster than Nesterov's scheme [63], as we also demonstrate later in our numerical tests. However, first we show in Figure 4.6 that Nesterov's scheme can be faster than RI Chebyshev acceleration in a significant regime when $\boldsymbol{B}$ does have complex eigenvalues. This figure compares Nesterov's scheme and RI Chebyshev for two cases of given b_1 and b_N values. We use r^*, the ACF of Theorem 4.1, as a benchmark, and show the numerically computed range of eigenvalues in the complex plane for which RI Chebyshev (cyan) and Nesterov's scheme (blue) yield convergence factors which do not exceed r^*. Evidently, Nesterov's scheme remains robust for a significantly broader range of complex eigenvalues, and it converges faster than the RI Chebyshev acceleration scheme if there is at least one complex eigenvalue in the blue sub-domain. We demonstrate numerical examples of accelerated multigrid solvers where this behavior is relevant and yields an advantage to Nesterov's scheme.

Finally, we select several complex eigenvalues with a given $\bar{b}^c$ and argument varying from 0 to π, to show how the argument affects $r_{c^*}(b^c)$. The results are shown in Figure 4.7. In general, we see that RI Chebyshev acceleration is adversely affected more strongly by change of θ than Nesterov's scheme, again demonstrating that the latter is more robust with respect to the existence of complex eigenvalues.

4.4 Numerical Tests

Nesterov's scheme is evidently easy to implement in practice for any stationary iteration method, since it only requires one additional step to combine the current iterate with the previous one. Furthermore, the additional computation is negligible and so acceleration is obtained almost for free. The only significant cost is the memory, since we need to store one previous iterate. In particular, compared with common acceleration techniques such as Krylov subspace methods [59], the cost of (4.3) is smaller. The drawback of course is the requirement to know b_1 and b_N, since they are needed for computing the optimal parameter c^*. In practice, as noted earlier, we may only need to know the spectral radius of $\boldsymbol{B}$. For example, if $b_1 \geq 0$ (e.g., if we iterate with $\boldsymbol{B}$ twice before successive Nesterov steps), then we only need to evaluate b_N, which can be done approximately by the power method [72], or by running the unaccelerated iteration

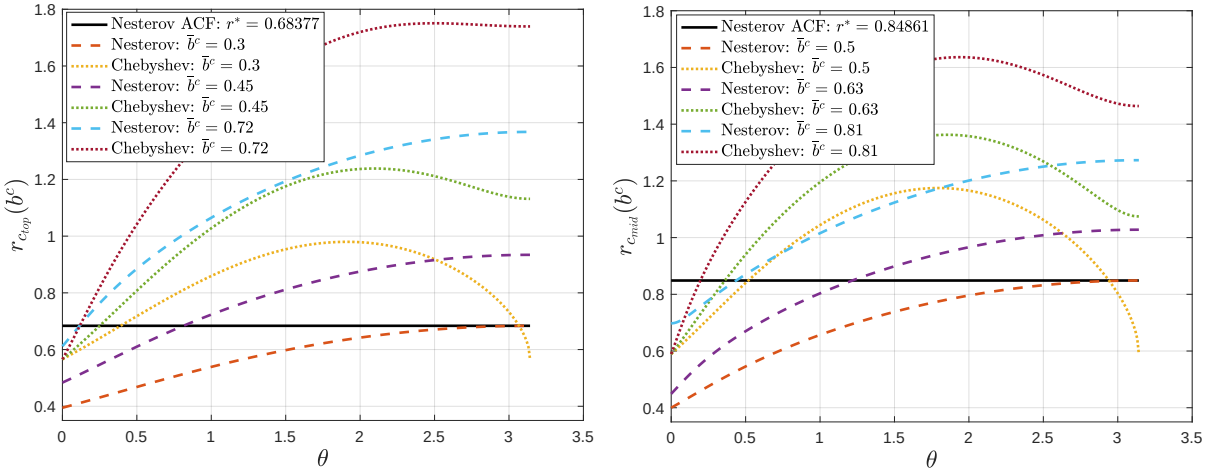
[2]Optimal Chebyshev acceleration for $\boldsymbol{B}$ with complex eigenvalues requires knowing an ellipse in the complex plane which contains all the eigenvalues, which may be hard to approximate in practice.



(a) $b_1 = -0.3$ and $b_N = 0.9$.

(b) $b_1 = -0.5$ and $b_N = 0.9$.

Figure 4.6: Cyan: complex eigenvalues of $\boldsymbol{B}$ for which RI Chebyshev acceleration yields a convergence factor smaller than or equal to r^* of Theorem 4.1; Blue: complex eigenvalues of $\boldsymbol{B}$ for which Nesterov's scheme yields a convergence factor smaller than or equal to r^* of Theorem 4.1. The red circle is of radius $|b_1|$.



(a) $b_1 = -0.3$ and $b_N = 0.9$.

(b) $b_1 = -0.5$ and $b_N = 0.9$.

Figure 4.7: ACF of particular complex eigenvalues b^c with fixed $\bar{b}^c$ and varying $\theta \in [0, \pi]$ for Nesterov's scheme and RI Chebyshev acceleration. The black line represents r^* of Nesterov's scheme.

for several steps. Since in many applications we need to solve (4.1) many times with different $\boldsymbol{f}$, we argue that the amount of computation required to approximate b_1 and b_N is often acceptable [59].

In this section we focus on accelerating multigrid V-cycle iterations [9, 10, 73] for elliptic boundary value problems. In some applications, the so-called smoothing factor, which is obtained by Fourier smoothing analysis, may provide a cheap yet sufficiently accurate approximation to b_1 and b_N [74, 75, 76, 77, 78].

We compare acceleration schemes for a multigrid solver for the two-dimensional diffusion equation on the unit square,

$$-\nabla(\sigma(x,y)\nabla u(x,y)) = f(x,y), \tag{4.17}$$

discretized on a 1024×1024 uniform grid, yielding a linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}\,, \tag{4.18}$$

with $\boldsymbol{A} = -\nabla^h(\boldsymbol{\sigma}\nabla^h\boldsymbol{u})$ a second-order finite-difference or bilinear finite element discretization of the diffusion equation with Dirichlet boundary conditions. In the first set of tests, the elements of the diffusion coefficient vector are all equal to 1, yielding a discretized Poisson equation, while in the second and third sets the elements are sampled from a log-normal distribution and from a uniform distribution in $(0,1)$, respectively, following [79]. All the tests in this chapter are implemented on a laptop with 2.3GHz Intel Core i9.

For the Poisson problem we employ the standard five-point finite difference discretization, damped Jacobi or Red-Black Gauss-Seidel relaxation, full-weighted residual transfers and bilinear interpolation, and the coarse-grid operators are defined by rediscretization on all coarse grids with the five-point discretization stencil [74]. We run standard $\mathrm{V}(\nu_1,\nu_2)$ cycles with $\nu_1 = 1$ pre-relaxation sweep and $\nu_2 = 0$ or 1 post-relaxation sweeps. We compare acceleration by Nesterov's scheme to Preconditioned Conjugate Gradients (PCG) with $\mathrm{V}(\nu_1,\nu_2)$ as the preconditioner, denoted PCG-$\mathrm{V}(\nu_1,\nu_2)$ and to RI Chebyshev acceleration of the $\mathrm{V}(\nu_1,\nu_2)$ cycles, denoted Chebyshev-$\mathrm{V}(\nu_1,\nu_2)$.

Let $\boldsymbol{r}_k = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}_k$ denote the residual vector at the end of the kth iteration. Then the convergence factor CF at the kth iteration is given by the ratio of the successive residual norms, $||\boldsymbol{r}_k||_2/||\boldsymbol{r}_{k-1}||_2$. We estimate the ACF by running sufficiently many iterations such that the CF no longer changes significantly.

In our first test we employ V(1,0) cycles for the Poisson problem, using Jacobi relaxation with the theoretical optimal damping factor 0.8 (obtained by Fourier smoothing analysis), yielding a smoothing factor (and correspondingly, an ACF) of 0.6. For this choice, Nesterov's scheme cannot provide acceleration, because $b_N = -b_1 = 0.6$. However, it is possible to improve the convergence factor by choosing a damping factor that is not optimal for stand-alone multigrid. We do this by increasing both b_N and b_1, and of course applying Nesterov acceleration. To demonstrate the potential gain, we show in Figure 4.8 the ACF of $\mathrm{V}(1,0)$ and Nesterov-accelerated $\mathrm{V}(1,0)$ cycles, for varying damping factors of the Jacobi relaxation. We find that the optimal choice is to reduce the damping factor to $\frac{8}{13}$, yielding $b_1 = -\frac{1}{3}b_N$. Evidently, these theoretical results are matched well by the numerical results achieved in practice. The effect is also seen in the first row of Figure 4.9. We find that optimizing the Jacobi damping factor for Nesterov acceleration yields a non-negligible reduction in the ACF (from 0.6 to 0.45) and in the run-time. Moreover, although PCG-$\mathrm{V}(1,0)$ is faster than Nesterov-$\mathrm{V}(1,0)$ in terms of iteration count, as expected, Nesterov acceleration yields a shorter run-time. As expected, the winner is Chebyshev-$\mathrm{V}(1,0)$, which is as fast as PCG-$\mathrm{V}(1,0)$ in terms of iteration count, but faster than all its competitors in terms of CPU time. Rather similar conclusions are obtained for $\mathrm{V}(1,1)$ cycles, as seen in the second row of Figure 4.9.

We next test accelerated multigrid cycles with Red-Black (RB) relaxation, which costs about the same as Jacobi and provides better smoothing, hence faster convergence. In this case, some of the eigenvalues of $\boldsymbol{B}$ are complex, and yet we select c^* based only on b_1 and b_N using Theorem 4.2. Again, we test both $\mathrm{V}(1,0)$ and $\mathrm{V}(1,1)$ cycles with acceleration. The results are shown in Figure 4.10. Note that we use GMRES without restart [59], which is faster than restarted GMRES, instead of PCG in this experiment, because $\mathrm{V}(1,0)$ and $\mathrm{V}(1,1)$ are not symmetric. The alternative of using symmetric Gauss-Seidel for the relaxation instead of RB retains the symmetry, allowing the use of PCG. However, in our numerical tests with $\nu_1 = \nu_2 = 1$ we found that using symmetric GS yields an ACF that is similar to that of RB but

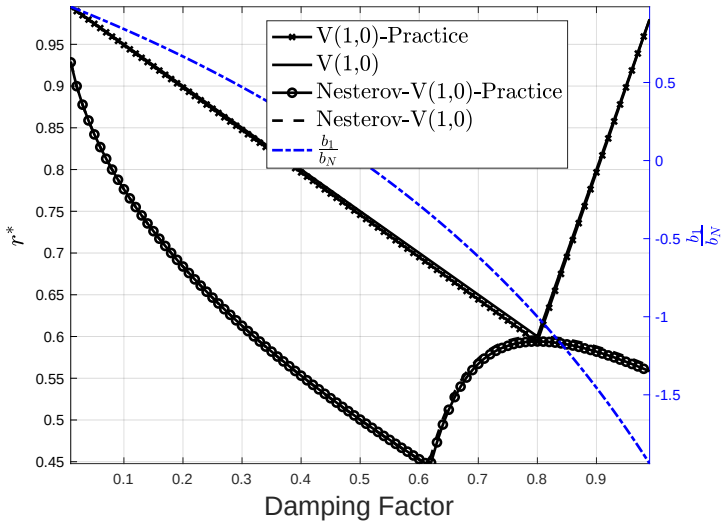


Figure 4.8: The ACF achieved by $\mathrm{V}(1,0)$ cycles, with and without Nesterov acceleration, as a function of the Jacobi relaxation damping factor. The extreme eigenvalues used for determining c^*, b_1 and b_N, are estimated by Fourier smoothing analysis, and the ratio $\frac{b_1}{b_N}$ is also shown. "Practice" refers to results achieved in practice by numerical computations on a 256×256 grid, terminated when the residual norm is smaller than 10^{-8}. The ACF is then estimated by the geometric mean of the last 5 iterations.

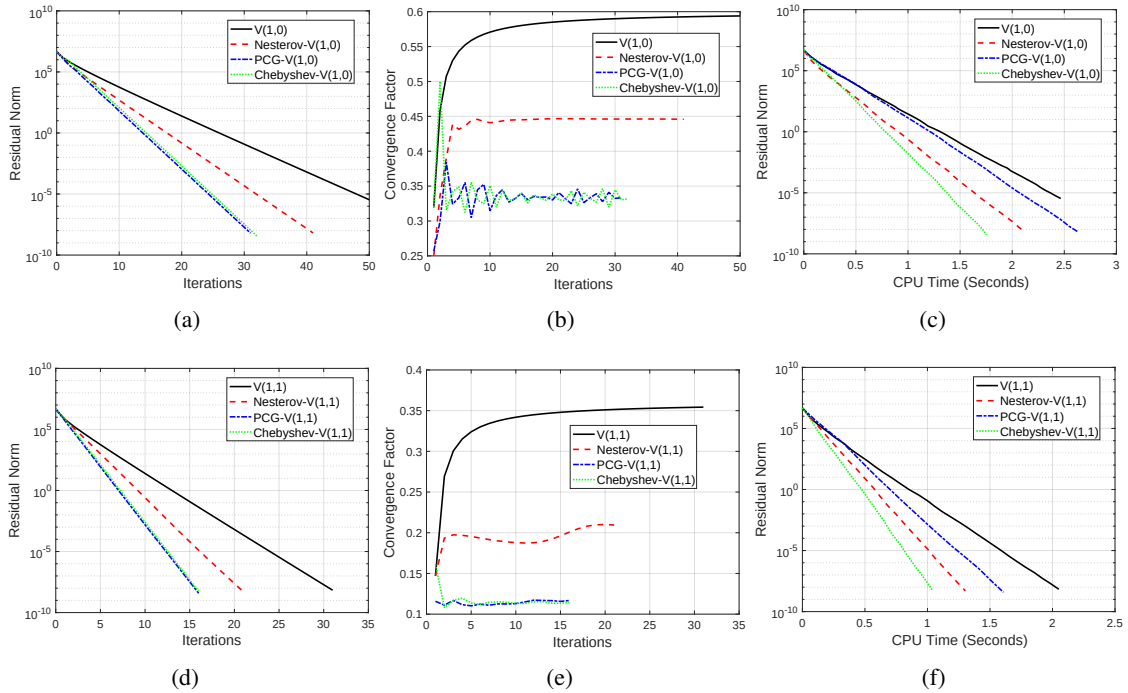


Figure 4.9: Comparison of acceleration methods for the Poisson problem. Optimally damped Jacobi relaxation is used in all the tests except for Nesterov-$\mathrm{V}(1,0)$, which uses a damping coefficient of $\frac{8}{13}$. First row: Accelerated $\mathrm{V}(1,0)$ cycles. Second row: Accelerated $\mathrm{V}(1,1)$ cycles.

requires twice the number of operations per cycle. Moreover, RB has an advantage in parallel computation. instead of PCG in this experiment, because $\mathrm{V}(1,0)$ and $\mathrm{V}(1,1)$ are not symmetric. In Figure 4.10, we see that Nesterov's scheme is faster than RI-Chebyshev acceleration, both in terms of iterations and CPU time. Moreover, we observe that GMRES with $\mathrm{V}(1,0)$ or $\mathrm{V}(1,1)$ as the preconditioner is fastest in terms

of iterations. However, Nesterov's scheme is fastest in terms of CPU time. Furthermore, we note that GMRES needs much more memory than Nesterov's scheme, and its implementation is not as trivial.

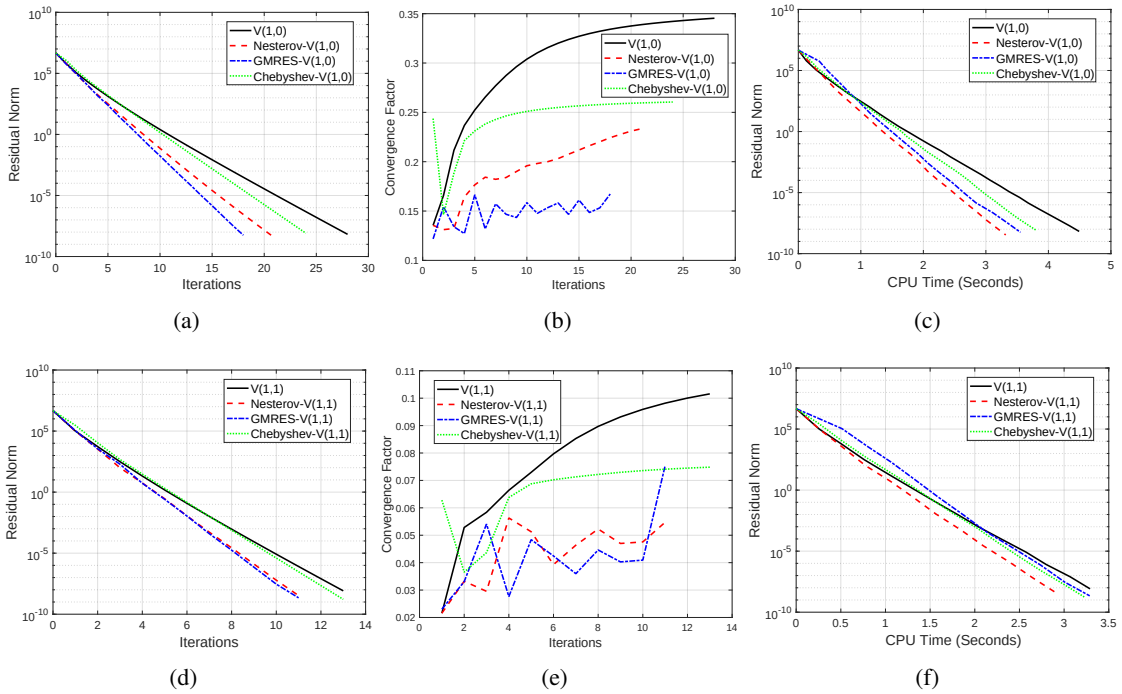


Figure 4.10: Comparison of acceleration methods for the Poisson problem with Red-Black relaxation. First row: Accelerated $\mathrm{V}(1,0)$ cycles. Second row: Accelerated $\mathrm{V}(1,1)$ cycles.

Finally, we test (4.17), (4.18) with a random diffusion coefficient vector $\boldsymbol{\sigma}$, sampled from a log-normal or uniform distribution, following [79]. Due to the discontinuous coefficients, we use the classical Black Box Multigrid algorithm [80], employing operator-dependent prolongation and Galerkin coarsening. For relaxation we employ Gauss-Seidel in natural (lexicographic) ordering. Following [79], we use bilinear finite element discretization. Here, we cannot use Fourier smoothing analysis, and b_N is estimated by running V cycles with no acceleration, and we assume $b_1 = 0$. The results are shown in Figure 4.11. We find that Nesterov's scheme is competitive in terms of iteration count, and it is the fastest method in terms of CPU time in these experiments.

4.5 Conclusion

In this chapter, we adapt Nesterov's scheme to accelerate stationary iterative methods for linear problems. Under the assumption that the eigenvalues of the iteration matrix are real, we derive a closed-form solution for the optimal scalar coefficient c used in Nesterov's scheme. Numerical tests with accelerated multigrid cycles demonstrate the advantages of this approach. Moreover, we also study the robustness of Nesterov's scheme for cases where some of the eigenvalue of the iteration matrix $\boldsymbol{B}$ are complex, identifying an explicit disk in the complex plane where the existence of complex eigenvalues does not degrade the rate of convergence. Our numerical results also demonstrate an advantage of Nesterov's acceleration scheme in such cases. In future work we plan to study more general cases of complex eigenvalues not covered in this chapter, and to extend our approach to nonlinear problems.

We note that the lower bound on the asymptotic convergence factor ACF shown in [70] is tight if the

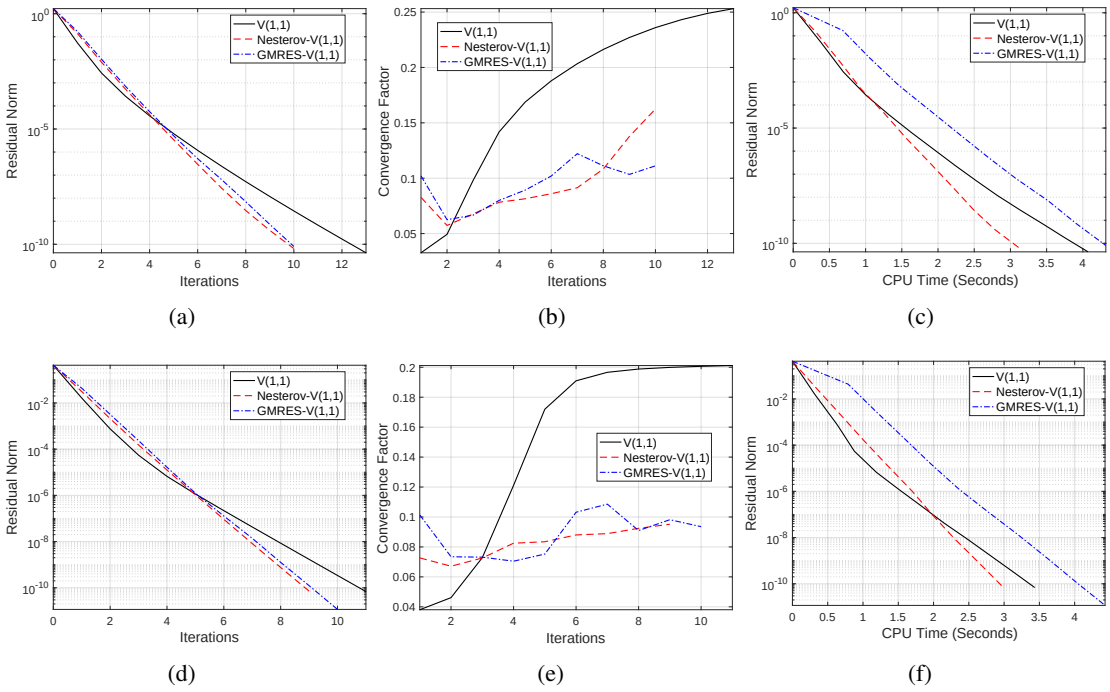


Figure 4.11: Comparison of accelerated Black Box Multigrid V$(1,1)$ cycles for the diffusion problem with log-normal (first row) and uniform (second row) distributions of the diffusion coefficient vector $\boldsymbol{\sigma}$.

regime of the complex eigenvalues meets the disk defined in Theorem 4.2. Following the idea in [70], the authors in [81] quantified the improvement of the ACF of Anderson acceleration applied to Alternating Direction Method of Multipliers for nonlinear problems. From [81], we see that the Jacobians of ADMM at the fixed point always have complex eigenvalues and find that our results are still relevant to some nonlinear examples considered in [81]. We also note that PCG and restricted-information Chebyshev are optimal when $\boldsymbol{B}$ only has real eigenvalues and are faster than Nesterov's scheme. However, one may benefit from the multi-step acceleration described in [70] to reduce such a gap. In future work we plan to study more general cases of complex eigenvalues not covered in this paper, to explore multi-step acceleration with optimal acceleration coefficients, and to try to extend our approach to other methods for nonlinear problems with analytic coefficients c_k.

Chapter 5

Merging Multigrid Optimization with SESOP

In this chapter, we report our work on merging multigrid optimization with sequential subspace optimization. This chapter is based on the following arXiv paper:

- **Tao Hong**, Irad Yavneh, and Michael Zibulevsky, *Merging Multigrid Optimization with SESOP,* arXiv:1812.06896.

5.1 Introduction

Multigrid (MG) methods are widely considered to be an efficient approach for solving elliptic partial differential equations (PDEs) and systems, as well as other problems which can be effectively represented on a hierarchy of grids or levels [9, 10, 11, 12]. However, it is often challenging to design efficient stand-alone MG methods for difficult problems, and therefore MG methods are often used in combination with acceleration techniques (e.g., [13]).

In this chapter we study MG in an optimization framework and seek robust solution methods by merging this approach with so-called SEquential Subspace OPtimization (SESOP) [15]. SESOP is a general framework for iteratively solving large-scale optimization problems, as described in the next section. The combined framework is called SESOP-MG, and its two-grid (TG) version is called SESOP-TG. We then analyze the asymptotic convergence factor (ACF) of a fixed-stepsize version of SESOP-TG for quadratic optimization problems, and estimate the expected acceleration due to SESOP by means of the so-called h-ellipticity measure [82, 74]. Resorting to local Fourier analysis (LFA) [82, 75], we propose two methods to estimate optimal fixed stepsizes of SESOP-TG for quadratic optimization problems cheaply in cases where LFA is applicable. Numerical tests demonstrate the relevance of the theoretical analysis in practice.

The chapter is organized as follows. The standard TG and SESOP algorithms are briefly described in the remainder of this section. The merged SESOP-TG/MG algorithm is proposed and tested in Section 5.2. An analysis of the ACF for a fixed-stepsize version of the SESOP-TG method for quadratic problems is presented in Section 5.3. There, we also show how to estimate the optimal fixed stepsizes cheaply for some specific problems. Numerical tests validate our analysis and the effectiveness of the proposed method for estimating the optimal fixed stepsizes. Conclusions are drawn in Section 5.4.

We adopt the following notation. $\boldsymbol{x}$ denotes the unknown solution vector, and $F(\boldsymbol{x})$ a convex function we aim to minimize. In the two-grid case, we use superscripts h and H to denote the fine and coarse grid, e.g., $F^h(\boldsymbol{x}^h)$ and $F^H(\boldsymbol{x}^H)$ denote the fine and coarse functions, respectively. In general, we use boldface font to denote vectors and matrices, and $^\mathcal{T}$ denotes the transpose operator.

5.1.1 Multigrid (MG)

We consider MG as a method for convex optimization. For an extensive review of MG for PDE optimization, see Borzì and Schulz [83] and references therein. Several authors developed MG optimization for specific problems in the 1990's, and at the end of the decade Nash [84] formulated MG as a general optimization framework called MG/OPT based directly on the well-known full approximation storage (FAS) scheme of Brandt [9]. Following this, Lewis and Nash applied this framework to systems governed by differential equations [85]. In similar vein, Wen and Goldfard proposed a line search MG method to solve unconstrained convex and nonconvex problems [86]. Toint et al. merged MG optimization with the trust region approach, applying MG to the series of linear subproblems arising in each step of trust region methods. This was applied to nonlinear convex or nonconvex problems, including bound constraints [87, 88, 89, 90]. Recently, Calandra et al. applied MG optimization to reduce the cost of step computation in high-order optimization [91].

Consider the following unconstrained problem defined on the fine-grid:

$$\boldsymbol{x}_*^h = \underset{\boldsymbol{x}^h \in \mathbb{R}^N}{\operatorname{argmin}} F^h(\boldsymbol{x}^h), \tag{5.1}$$

where $F^h(\boldsymbol{x}^h) : \mathbb{R}^N \to \mathbb{R}$ is a convex and differentiable function and the solution set of (5.1) is not empty. We describe a single iteration of the two-grid algorithm next, with MG obtained by straightforward recursion. Denote by $\boldsymbol{x}_k^h$ the approximation to the fine-grid solution $\boldsymbol{x}_*^h$ after the kth iteration. Assume that we have defined two full-rank operators, a *restriction* $\boldsymbol{I}_h^H : \mathbb{R}^N \to \mathbb{R}^{N_c}$, and a *prolongation* $\boldsymbol{I}_H^h : \mathbb{R}^{N_c} \to \mathbb{R}^N$, where N_c is the size of the coarse-grid. Furthermore, let $\boldsymbol{x}_k^H$ denote a coarse-grid approximation to $\boldsymbol{x}_k^h$ (for example, we may use $\boldsymbol{x}_k^H = \boldsymbol{I}_h^H \boldsymbol{x}_k^h$, but other choices may be used as well). The coarse-grid problem is then defined as follows:

$$\boldsymbol{x}_*^H = \underset{\boldsymbol{x}^H \in \mathbb{R}^{N_c}}{\operatorname{argmin}} F^H(\boldsymbol{x}^H) - \boldsymbol{v}_k^\mathcal{T} \boldsymbol{x}^H, \tag{5.2}$$

where F^H is a coarse approximation to F^h, and $\boldsymbol{v}_k = \nabla F^H(\boldsymbol{x}_k^H) - \boldsymbol{I}_h^H \nabla F^h(\boldsymbol{x}_k^h)$. The correction term $\boldsymbol{v}_k$, adapted from FAS, is used to enforce the same first-order optimality condition [84] on the fine and coarse levels. After solving (5.2) (exactly in TG, and approximately and recursively in MG), the coarse-grid correction CGC direction is defined by

$$\boldsymbol{d}_k^h = \boldsymbol{I}_H^h (\boldsymbol{x}_*^H - \boldsymbol{x}_k^H). \tag{5.3}$$

Finally, the CGC is added to the current fine-grid approximation:

$$\boldsymbol{x}_k^h \leftarrow \boldsymbol{x}_k^h + \boldsymbol{d}_k^h. \tag{5.4}$$

This may be followed by additional relaxation steps, yielding the $k+1$st approximation $\boldsymbol{x}_{k+1}^h$. The two-grid algorithm is written in Algorithm 5.1.

Algorithm 5.1 Two-Grid (TG)

Input: Initial value $\boldsymbol{x}_0^h$, convergence criterion ε, the number of maximal iterations Max_Iter, ν_1, ν_2 – the number of pre- and post-relaxation steps, $k = 1$.

Output: Solution $\boldsymbol{x}_*^h$.

1: **while** $k \leq Max\_Iter$ **do**
2: $\quad \boldsymbol{x}_k^h \leftarrow Relaxation(\boldsymbol{x}_{k-1}^h, \nu_1)$.
3: $\quad$ Evaluate the gradient $\nabla F^h(\boldsymbol{x}_k^h)$ and formulate $\boldsymbol{v}_k$.
4: $\quad$ **if** $|\nabla F^h(\boldsymbol{x}_k^h)| \leq \varepsilon$ **then**
5: $\quad\quad \boldsymbol{x}_*^h \leftarrow \boldsymbol{x}_k^h$.
6: $\quad\quad$ **Return**
7: $\quad$ **end if**
8: $\quad$ Solve the coarse problem (5.2) to get $\boldsymbol{x}_*^H$ and then $\boldsymbol{d}_k^h \leftarrow \boldsymbol{I}_H^h \left(\boldsymbol{x}_*^H - \boldsymbol{I}_h^H \boldsymbol{x}_k^h\right)$.
9: $\quad$ Update $\boldsymbol{x}_k^h \leftarrow \boldsymbol{x}_k^h + \boldsymbol{d}_k^h$.
10: $\quad \boldsymbol{x}_k^h \leftarrow Relaxation(\boldsymbol{x}_k^h, \nu_2)$.
11: $\quad k \leftarrow k+1$
12: **end while**
13: $\boldsymbol{x}_*^h \leftarrow \boldsymbol{x}_k^h$.
14: **Return**

5.1.2 SEquential Subspace OPtimization (SESOP)

SESOP [92, 93, 94, 15] is a framework for solving smooth large-scale unconstrained minimization problems such as (5.1), by sequential optimization over affine subspaces $\boldsymbol{M}_k$, spanned by the current descent direction (typically preconditioned gradient) and m previous propagation directions of the method. If $F^h(\boldsymbol{x}^h)$ is convex, SESOP yields the optimal worst-case convergence factor of $O(\frac{1}{k^2})$ [92], while achieving efficiency of the quadratic Conjugate Gradients (CG) method when the problem is close to quadratic or in the vicinity of the solution.

The affine subspace at the kth iteration is defined by

$$\boldsymbol{M}_k^h = \left\{ \boldsymbol{x}_k^h + \boldsymbol{P}_k^h \boldsymbol{\alpha} : \boldsymbol{\alpha} \in \mathbb{R}^{m+1} \right\},$$

where $\boldsymbol{x}_k^h$ is the kth iterate, the matrix $\boldsymbol{P}_k^h$ contains the spanning directions in its columns, the preconditioned gradient $\boldsymbol{P}\nabla F^h(\boldsymbol{x}_k^h)$ and m last steps $\boldsymbol{\delta}_i^h = \boldsymbol{x}_i^h - \boldsymbol{x}_{i-1}^h$,

$$\boldsymbol{P}_k^h = \left[\boldsymbol{P}\nabla F^h(\boldsymbol{x}_k^h), \boldsymbol{\delta}_k^h, \boldsymbol{\delta}_{k-1}^h, \cdots, \boldsymbol{\delta}_{k-m+1}^h \right], \ m \geq 0. \tag{5.5}$$

The new iterate $\boldsymbol{x}_{k+1}^h$ is obtained via optimization of $F^h(\cdot)$ over the current subspace $\boldsymbol{M}_k^h$,

$$\begin{aligned} \boldsymbol{\alpha}_k^* &= \operatorname{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^{m+1}} F^h(\boldsymbol{x}_k^h + \boldsymbol{P}_k^h \boldsymbol{\alpha}), \\ \boldsymbol{x}_{k+1}^h &= \boldsymbol{x}_k^h + \boldsymbol{P}_k^h \boldsymbol{\alpha}_k^*. \end{aligned} \tag{5.6}$$

By keeping the dimension of $\boldsymbol{\alpha}$ low, (5.6) can be solved efficiently with a Newton-type method. SESOP is relatively efficient if (i) the objective function can be represented as $F^h(\boldsymbol{A}\boldsymbol{x}^h)$; (ii) the evaluation of $F^h(\cdot)$ is cheap, but calculating $\boldsymbol{A}\boldsymbol{x}^h$ is expensive. In this case, we can avoid re-computation of $\boldsymbol{A}\boldsymbol{x}_k^h$ during the subspace minimization by storing $\boldsymbol{A}\boldsymbol{P}_k^h$ and $\boldsymbol{A}\boldsymbol{x}_k^h$. A typical class of problems of this type is the well-known $\ell_1 - \ell_2$ optimization [94].

SESOP can yield faster convergence if more efficient descent directions are added to the subspace

$\boldsymbol{M}_k^h$. This may include a cumulative or parallel coordinate descent step, a separable surrogate function or expectation-maximization step, Newton-type steps and other methods [93, 94, 15]. In this work we suggest adding the CGC direction provided by the MG framework.

5.2 Merging Multigrid with SESOP

5.2.1 SESOP-TG

We begin this section by introducing a two-grid version of our scheme, SESOP-TG, and later extend it to the multilevel version, SESOP-MG. In its basic form, the idea is to add the CGC $\boldsymbol{d}_k^h$ of (5.3) into the affine subspace $\boldsymbol{M}_k$ by replacing $\boldsymbol{P}_k^h$ with an augmented subspace $\bar{\boldsymbol{P}}_k^h = \begin{bmatrix} \boldsymbol{d}_k^h & \boldsymbol{P}_k^h \end{bmatrix}$ and computing the locally optimal $\boldsymbol{\alpha}$ to obtain the next iterate, $\boldsymbol{x}_{k+1}^h$. Our intention is to combine the efficiency of MG that results from fast reduction of smooth error by the CGC, with the robustness of SESOP that results from the local optimization over the subspace. That is, even in difficult cases where the CGC is inadequate, the algorithm should still converge at least as fast as the standard SESOP, because an inefficient direction simply results in small (or conceivably even negative) coefficient. SESOP-TG is presented in Algorithm 5.2. Note that we add the option of two additional steps to the usual SESOP algorithm, the pre- and post-relaxation at Lines 2 and 13 in Algorithm 5.2, commonly applied in MG algorithms. This allows us to advance the solution with low computational expense whenever the coarse-grid direction is inefficient (due the the fact that previous use of the CGC direction greatly reduced the smooth error). For the relaxation we typically use nonlinear Jacobi or Gauss-Seidel applied to the gradient equation. The flowchart of Algorithm 5.2 is presented in Figure 5.1.

Algorithm 5.2 SESOP-TG-m

Input: Initial value $\bar{\boldsymbol{x}}_0^h = \boldsymbol{x}_0^h$, convergence criterion ε, the number of maximal iterations Max_Iter, the preconditioner $\boldsymbol{P}$ and ν_1, ν_2 – the number of pre- and post-relaxation steps, $k = 1$.

Output: $\boldsymbol{x}_*^h$.

1: **while** $k \leq Max\_Iter$ **do**
2: $\quad \bar{\bar{\boldsymbol{x}}}_k^h \leftarrow Relaxation(\bar{\boldsymbol{x}}_{k-1}^h, \nu_1)$.
3: $\quad$ Evaluate the gradient $\nabla F^h(\bar{\bar{\boldsymbol{x}}}_k^h)$.
4: $\quad$ **if** $|\nabla F^h(\bar{\bar{\boldsymbol{x}}}_k^h)| \leq \varepsilon$ **then**
5: $\quad\quad \boldsymbol{x}_*^h \leftarrow \bar{\bar{\boldsymbol{x}}}_k^h$.
6: $\quad\quad$ **Return**
7: $\quad$ **else**
8: $\quad\quad \boldsymbol{P}_k^h \leftarrow \left[\boldsymbol{P}\nabla F^h(\bar{\bar{\boldsymbol{x}}}_k^h) \quad \bar{\bar{\boldsymbol{x}}}_k^h - \boldsymbol{x}_{k-2}^h \quad \cdots \quad \bar{\bar{\boldsymbol{x}}}_{k-m+1}^h - \boldsymbol{x}_{k-m-1}^h\right]$.
9: $\quad$ **end if**
10: $\quad$ Solve (5.2) with initial $\boldsymbol{x}_k^H$ to get $\boldsymbol{x}_*^H$ and then $\boldsymbol{d}_k^h \leftarrow \boldsymbol{I}_H^h\left(\boldsymbol{x}_*^H - \boldsymbol{x}_k^H\right)$.
11: $\quad$ Formulate the augmented $\bar{\boldsymbol{P}}_k^h \leftarrow \left[\boldsymbol{d}_k^h \quad \boldsymbol{P}_k^h\right]$.
12: $\quad$ Solve approximately (5.6) on $\bar{\boldsymbol{P}}_k^h$ and update $\boldsymbol{x}_k^h \leftarrow \bar{\bar{\boldsymbol{x}}}_k^h + \bar{\boldsymbol{P}}_k^h \boldsymbol{\alpha}_k^*$.
13: $\quad \bar{\boldsymbol{x}}_k^h \leftarrow Relaxation(\boldsymbol{x}_k^h, \nu_2)$.
14: **end while**
15: $\boldsymbol{x}_*^h \leftarrow \bar{\boldsymbol{x}}_k^h$.
16: **Return**

Remark 5.1. Evidently, if we select $\bar{\boldsymbol{P}}_k^h$ to contain only $\boldsymbol{d}_k^h$, with $\boldsymbol{\alpha}_k^* = 1$, then SESOP-TG-$m$ reduces to TG. Note that in Algorithm 5.1, we need to valuate $\nabla F^h(\boldsymbol{x}_k^h)$ to formulate $\boldsymbol{v}_k$ but we do not use it further. Here,

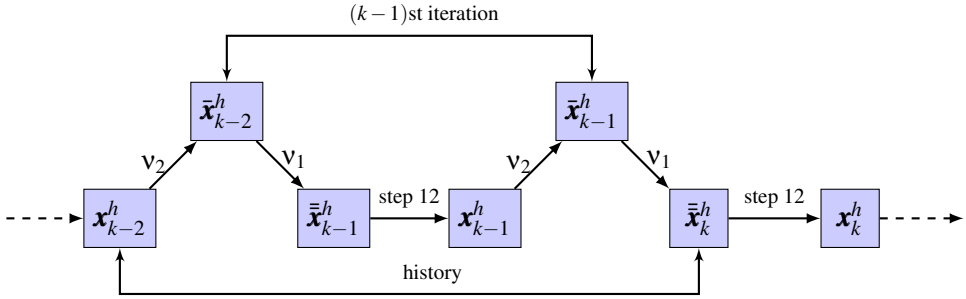


Figure 5.1: Flowchart of the iterates of Algorithm 5.2.

however, we put $\nabla F^h(\boldsymbol{x}^h_k)$ together with the CGC direction in the augmented subspace $\bar{\boldsymbol{P}}^h_k$ for further use. The main drawback of Algorithm 5.2 is of course the need to solve the subspace minimization problem for $\boldsymbol{\alpha}^*_k$ at each iteration. However, in certain cases of problems with special structure or large size, the cost of the subspace minimization may become negligible; see numerical tests in Section 5.2.2.

Remark 5.2. Similarly to standard multigrid methods, the multilevel algorithm called SESOP-MG-*m* is derived by recursively treating (5.2) at step 10 of Algorithm 5.2. Compared with the subspace $\bar{\boldsymbol{P}}^k_h$ in (5.2), the subspaces in the coarse problems only contain two directions: the preconditioned gradient and the coarse grid correction from the coarser level.

5.2.2 Numerical Tests

We demonstrate the SESOP-MG-*m* algorithm performance for two nonlinear problems and compare it with steepest descent (SD), Nesterov acceleration [50], and limited-memory BFGS (LBFGS) [51]. The MG method proposed in [86]—denoted "MG-Line"—is adapted for this comparison, to illustrate the effectiveness of using history. We use SD as the relaxation for "MG-Line". For SESOP-MG-*m*, we employ Newton's method for the subspace minimization at each iteration. All the tests are performed on a laptop with 2.3GHz Intel Core i9.

Example I

Consider the following variational problem on a uniform grid [86],

$$\begin{cases} \min_{u(x,y)} F\left(u(x,y)\right) \equiv \int_\Omega \frac{1}{2}\left|\nabla u(x,y)\right|^2 + \gamma\left(u(x,y)e^{u(x,y)} - e^{u(x,y)}\right), \\ \qquad\qquad -f(x,y)u(x,y)dxdy \\ \text{such that} \quad u(x,y) = 0 \quad \text{on} \quad \partial\Omega, \end{cases} \tag{5.7}$$

where ∇ is the gradient, $\gamma = 10$, $\Omega = [0,1]\times[0,1]$, and

$$f(x,y) = \left(\left(9\pi^2 + \gamma e^{(x^2-x^3)\sin(3\pi y)}\right)\left(x^2-x^3\right) + 6x - 2\right)\sin\left(3\pi y\right).$$

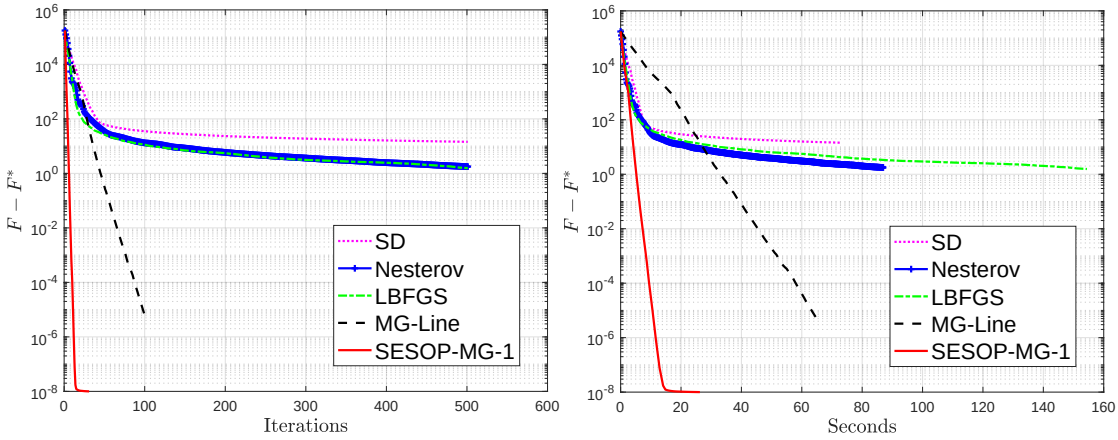
Through the Euler-Lagrange equation, the PDE formulation of (5.7) reads

$$\begin{cases} -\Delta u(x,y) + \gamma u(x,y)e^{u(x,y)} = f(x,y) & \text{on } \Omega, \\ u(x,y) = 0 & \text{on } \partial\Omega, \end{cases} \tag{5.8}$$

where Δ denotes the Laplacian.

Equation (5.7) is discretized by finite differences, using first order forward differences for ∇ in (5.7), resulting in a five-point stencil for Δ in (5.8). The finest grid size is 1024×1024, and we employ 7 levels, coming down to grids of size 8×8. The coarse problems are defined by rediscretization, and full-weighted residual transfers and bilinear interpolation are used as the restriction and prolongation. The relaxation sweep parameters are $\nu_1 = 1$ and $\nu_2 = 0$ for MG-Line. BFGS with up to ten iterations is used for solving the problem on the coarsest level. The minFunc toolbox [95] is used for BFGS and Newton's method. The maximal number of iterations of SESOP-MG-1 and MG-Line are set to be 30 and 100, respectively. The other methods are allowed to use 500 iterations. For clarity of display, we denote by $F^* + 10^{-8}$ the minimal objective value over all methods after running the maximal allowed number of iterations.

From Figure 5.2, we clearly observe that SESOP-MG-1 is the fastest algorithm in terms of both iteration count and CPU time. Moreover, we note that SD, Nesterov, and LBFGS converge fast initially, but then slow down. This well-known phenomenon is a result of the fact that these methods cannot efficiently eliminate low-frequency error, unlike MG methods which use CGC. Note also that SESOP-MG-1 is significantly faster than MG-Line, demonstrating the effectiveness of introducing history for acceleration.



(a) (b)

Figure 5.2: Comparison of different methods for (5.7) on 1024×1024 grids.

Example II

Our second example is the p-Laplacian,

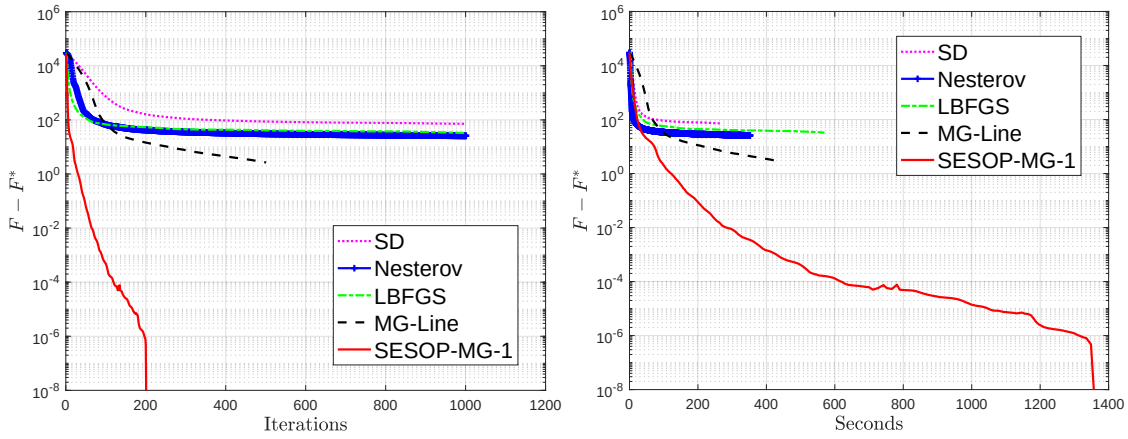
$$\begin{cases} \min_{u(x,y)} F\left(u(x,y)\right) \equiv \int_\Omega |\nabla u(x,y) + \xi|^p - f(x,y)u(x,y)dxdy, \\ \quad \text{such that} \quad u(x,y) = 0 \quad \text{on} \quad \partial\Omega, \end{cases} \tag{5.9}$$

where $p \in (1,2)$. The corresponding PDE of (5.9) is:

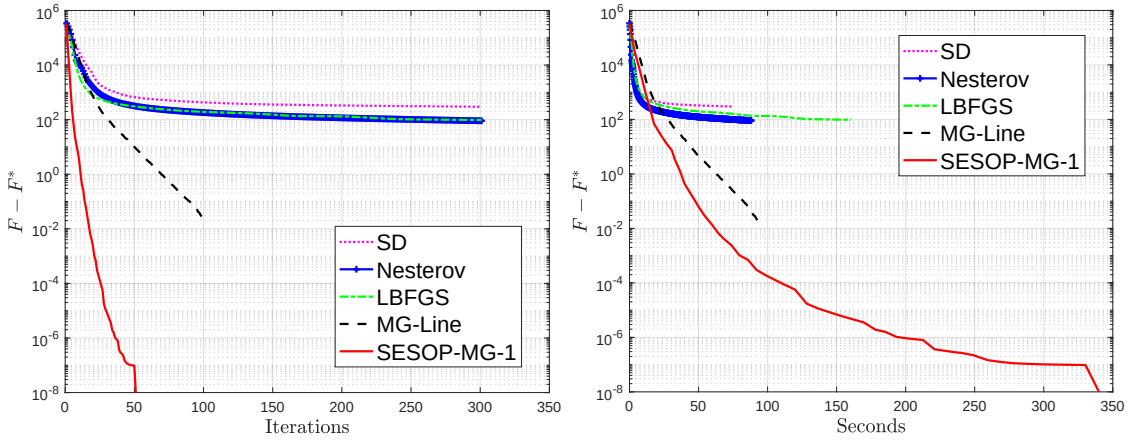
$$\begin{cases} -\nabla \cdot \left(|\nabla u + \xi|^{p-2} \nabla u \right) = f & \text{on } \Omega \\ u(x,y) = 0 & \text{on } \partial\Omega. \end{cases} \tag{5.10}$$

The parameter $\xi = 10^{-6}$ is introduced to maintain differentiability (hence a positive denominator). The function $f(x,y)$ is defined by substituting $u(x,y) = (x^2 - x^3)\sin(3\pi y)$ into (5.10). Note that solving (5.9)

becomes especially challenging when p is close to 1. We choose $p = 1.3$ and 1.6 to study the performance of our approach. The experimental setting is the same as that of *Example I*, except the maximal number of iterations allowed. As seen in Figure Figure 5.3, SESOP-MG-1 is the fastest of the five methods, followed by MG-Line, as in *Example I*.



(a) $p = 1.3$. (b) $p = 1.3$.



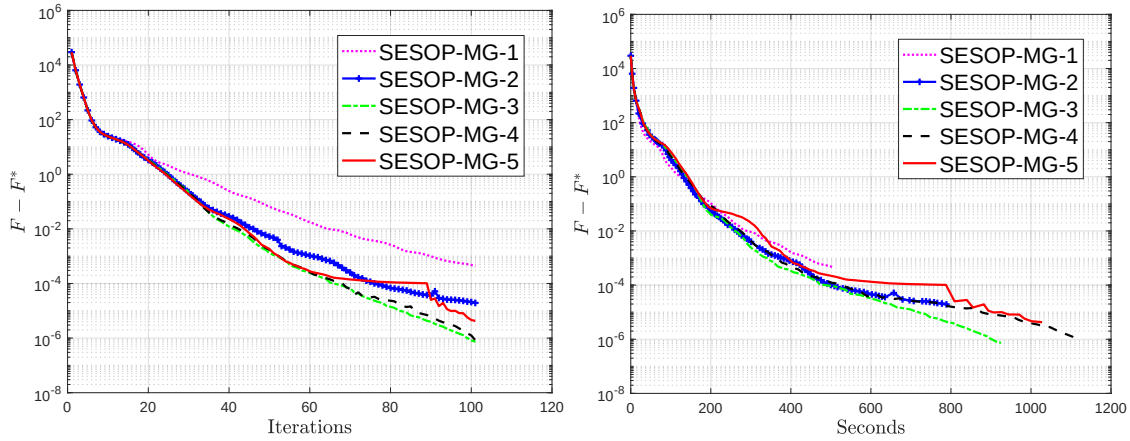
(c) $p = 1.6$. (d) $p = 1.6$.

Figure 5.3: Comparison of different methods for (5.9) on 1024×1024 grids.

Dependence of SESOP-MG-m on m

Next we show how the choice of the search-space size m affects the performance of SESOP-MG-m. To this end, we apply SESOP-MG-m with different m to (5.9) with $p = 1.3$. In Figure 5.4(a), we see that using a larger m yields faster convergence rates. However, a larger m also increases the complexity of the subspace minimization results in higher CPU times per iteration. From Figure 5.4(b), we find that $m = 3$ is a good compromise for achieving low CPU times for this test. We also see that $m = 5$ results in higher times than $m = 3, 4$ because, in practice, the larger size of the subspace introduces some numerical difficulties.

The numerical tests demonstrate the potential advantage of SESOP-TG/MG-m for such types of optimization problems. However, SESOP-TG/MG-m comes with the cost of solving a subspace minimization problem at each iteration. From Figure 5.4, we observe that the performance of SESOP-TG/MG-m deteriorates for large m. In the case of quadratic optimization problems, we may be able to avoid the



(a) Objective value versus iterations. $p = 1.3$.

(b) Objective value versus CPU time. $p = 1.3$.

Figure 5.4: Comparison of different m for (5.9) on 1024×1024 grids.

subspace minimization, by using fixed nearly-optimal stepsizes for SESOP-TG-1. Indeed, we derive such fixed stepsizes, and show that they yield ACF's that are comparable to those obtained by subspace minimization. In such cases, we get the acceleration nearly for free, provided that we can estimate the fixed parameters efficiently. To this end, we propose two heuristic methods, based on local Fourier analysis (LFA) and smoothing analysis, to estimate the optimal fixed stepsizes cheaply.

5.3 Convergence Factor Analysis of SESOP-TG-1 for Quadratic Problems

To gain insight, we analyze SESOP-TG-1 for quadratic optimization problems, which are equivalent to the solution of linear systems. We first derive a fairly general formulation for SESOP-TG-1. Then we explore, under certain simplifying assumptions, a fixed-stepsize variant of SESOP-TG-1. In this analysis we assume for simplicity no pre- or post-relaxation steps.

Consider the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}, \tag{5.11}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a symmetric positive-definite (SPD) matrix, and we omit h superscripts for notational simplicity. Evidently, solving (5.11) is equivalent to the following quadratic minimization problem:

$$\boldsymbol{x}_* = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^N} F^h(\boldsymbol{x}) \triangleq \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{f}^T\boldsymbol{x}. \tag{5.12}$$

Given iterates $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_{k-2}$, the next iterate produced by SESOP-TG-1 is given by

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + c_1(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}) + c_2\boldsymbol{P}(\boldsymbol{f} - \boldsymbol{A}\boldsymbol{x}_{k-1}) + c_3\boldsymbol{I}_H^h\boldsymbol{A}_H^{-1}\boldsymbol{I}_h^H(\boldsymbol{f} - \boldsymbol{A}\boldsymbol{x}_{k-1}), \tag{5.13}$$

where c_1, c_2, c_3 are the optimal weights associated with the three directions comprising $\bar{\boldsymbol{P}}_k^h$: c_1 multiplies the so-called history, that is, the difference between the last two iterates; c_2 multiplies the preconditioned gradient; c_3 multiplies the CGC direction $\boldsymbol{d}_{k-1}^h$. Here, $\boldsymbol{A}_H$ represents the coarse-grid matrix approximating $\boldsymbol{A}$, which is most commonly defined by the Galerkin formula, $\boldsymbol{A}_H = \boldsymbol{I}_h^H\boldsymbol{A}\boldsymbol{I}_H^h$, or simply by rediscretization

on the coarse-grid in the case where $\boldsymbol{A}$ is the discretization of an elliptic PDE on the fine-grid. Subtracting $\boldsymbol{x}_*$ from both sides of (5.13), and denoting the error by $\boldsymbol{e}_k = \boldsymbol{x}_* - \boldsymbol{x}_k$, we get

$$\boldsymbol{e}_k = \boldsymbol{e}_{k-1} + c_1(\boldsymbol{e}_{k-1} - \boldsymbol{e}_{k-2}) - c_2\boldsymbol{P}\boldsymbol{A}\boldsymbol{e}_{k-1} - c_3\boldsymbol{I}_H^h\boldsymbol{A}_H^{-1}\boldsymbol{I}_h^H\boldsymbol{A}\boldsymbol{e}_{k-1}. \tag{5.14}$$

Rearranging (5.14) yields

$$\boldsymbol{e}_k = \boldsymbol{B}\boldsymbol{e}_{k-1} - c_1\boldsymbol{e}_{k-2}, \tag{5.15}$$

where

$$\boldsymbol{B} = (1+c_1)\boldsymbol{I} - (c_2\boldsymbol{P} + c_3 I_H^h \boldsymbol{A}_H^{-1} I_h^H)\boldsymbol{A}, \tag{5.16}$$

and $\boldsymbol{I}$ denotes the identity matrix. Define the vector $\boldsymbol{E}_k = \begin{bmatrix} \boldsymbol{e}_k \\ \boldsymbol{e}_{k-1} \end{bmatrix}$. Then, (5.15) implies the following relation:

$$\boldsymbol{E}_k = \boldsymbol{\Upsilon}\boldsymbol{E}_{k-1}, \;\; \boldsymbol{\Upsilon} \triangleq \begin{bmatrix} \boldsymbol{B} & -c_1\boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \tag{5.17}$$

and the asymptotic convergence factor (ACF) of SESOP-TG-1 is given by the spectral radius of $\boldsymbol{\Upsilon}$.

To analyze the ACF of SESOP-TG-1, we continue under the assumption that the coefficients c_j, $j = 1,2,3$, are fixed, and compute the optimal coefficients, yielding the smallest ACF. Denote by r an eigenvalue of $\boldsymbol{\Upsilon}$ with eigenvector $\boldsymbol{v} = [\boldsymbol{v}_1, \boldsymbol{v}_2]^{\mathcal{T}}$,

$$\begin{bmatrix} \boldsymbol{B} & -c_1\boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix} = r \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix}.$$

Hence, $\boldsymbol{v}_1 = r\boldsymbol{v}_2$ and $\boldsymbol{B}\boldsymbol{v}_1 - c_1\boldsymbol{v}_2 = r\boldsymbol{v}_1$. This yields

$$\boldsymbol{B}\boldsymbol{v}_1 = \left(r + \frac{c_1}{r}\right)\boldsymbol{v}_1.$$

Thus, $\boldsymbol{v}_1$ is an eigenvector of $\boldsymbol{B}$ with eigenvalue $b \triangleq r + \frac{c_1}{r}$ leading to $r^2 - br + c_1 = 0$, with solutions $r_1(b,c_1) = \frac{1}{2}\left(b + \sqrt{b^2 - 4c_1}\right)$ and $r_2(b,c_1) = \frac{1}{2}\left(b - \sqrt{b^2 - 4c_1}\right)$. The spectral radius of $\boldsymbol{\Upsilon}$ is therefore

$$\rho(\boldsymbol{\Upsilon}) = \frac{1}{2}\max_b \left| b + \mathrm{sgn}(b)\sqrt{b^2 - 4c_1} \right|,$$

where b runs over the eigenvalues of $\boldsymbol{B}$ and $\mathrm{sgn}(\cdot)$ is a sign function which is defined to be 1 for non-negative arguments and -1 for negative arguments. For the remainder of our analysis we focus on the case where the eigenvalues b of $\boldsymbol{B}$ are all real.

5.3.1 The Case of Real b

In many practical cases, the eigenvalues of $\boldsymbol{B}$ are all real, which simplifies the analysis and yields insight. We focus next on a common situation where this is indeed the case.

Lemma 5.3.1. *Assume:*

1. *The prolongation has full column-rank and the restriction is the adjoint of the prolongation:* $I_h^H = (I_H^h)^{\mathcal{T}}$.

2. *The coarse-grid operator $\boldsymbol{A}_H$ is SPD.*

3. *The preconditioner $\boldsymbol{P}$ is SPD.*

Then the eignevalues of $\boldsymbol{B} = (1+c_1)\boldsymbol{I} - (c_2\boldsymbol{P} + c_3\boldsymbol{I}_H^h\boldsymbol{A}_H^{-1}\boldsymbol{I}_h^H)\boldsymbol{A}$ are all real.

Proof. Because $\boldsymbol{A}$ is SPD, there exists a SPD matrix $\boldsymbol{A}^{\frac{1}{2}}$ such that $\boldsymbol{A}^{\frac{1}{2}}\boldsymbol{A}^{\frac{1}{2}} = \boldsymbol{A}$. The matrix

$$\boldsymbol{A}^{\frac{1}{2}}\boldsymbol{B}\boldsymbol{A}^{-\frac{1}{2}} = (1+c_1)\boldsymbol{I} - \boldsymbol{A}^{\frac{1}{2}}(c_2\boldsymbol{P} + c_3\boldsymbol{I}_H^h\boldsymbol{A}_H^{-1}\boldsymbol{I}_h^H)\boldsymbol{A}^{\frac{1}{2}}$$

is similar to $\boldsymbol{B}$ so they have the same eigenvalues. Moreover, $\boldsymbol{A}^{\frac{1}{2}}\boldsymbol{B}\boldsymbol{A}^{-\frac{1}{2}}$ is evidently symmetric, so its eigenvalues are all real. □

The first two assumptions of this lemma are satisfied commonly, including of course both the case where $\boldsymbol{A}_H$ is defined by rediscretization on the coarse grid and the case of Galerkin coarsening. The preconditioner $\boldsymbol{P}$ is SPD for some commonly used MG relaxation methods, including Richardson (where $\boldsymbol{P}$ is the identity matrix), Jacobi (where $\boldsymbol{P}$ is the inverse of the diagonal of $\boldsymbol{A}$), and symmetric Gauss-Seidel.

We next adopt the change of variables $c_{23} = c_2 + c_3$, $\alpha = c_2/c_{23}$. This yields

$$\boldsymbol{B} = (1+c_1)\boldsymbol{I} - c_{23}\boldsymbol{A}_\alpha, \tag{5.18}$$

with

$$\boldsymbol{A}_\alpha = \left(\alpha\boldsymbol{P} + (1-\alpha)\boldsymbol{I}_H^h\boldsymbol{A}_H^{-1}\boldsymbol{I}_h^H\right)\boldsymbol{A}. \tag{5.19}$$

Lemma 5.3.2. *The eigenvalues of $\boldsymbol{A}_\alpha$ are real and positive for any $\alpha \in (0,1]$.*

Proof. This follows from the fact that $\boldsymbol{A}$, $\boldsymbol{P}$ and $I_H^h\boldsymbol{A}_H^{-1}I_h^H$ are SPD. □

We henceforth denote the eigenvalues of $\boldsymbol{A}_\alpha$ by a_α, and assume $\alpha \in (0,1]$ (that is, c_2 and c_3 are of the same sign), so the a_α's are all real and positive. We then proceed by fixing α and optimizing c_1 and c_{23} so as to minimize the spectral radius of $\boldsymbol{\Upsilon}$. In light of Lemma 5.3.1, we have $b \equiv b(c_1, c_{23}, a_\alpha) = 1 + c_1 - c_{23}a_\alpha$, and the ACF as a function of $c_1,\ c_{23}$ is given by:

$$\bar{r}(c_1, c_{23}) \triangleq \rho(\boldsymbol{\Upsilon}) = \max_{a_\alpha} \hat{r}(c_1, c_{23}, a_\alpha), \tag{5.20}$$

where $\hat{r}(c_1, c_{23}, a_\alpha) \triangleq \frac{1}{2}\left|b + \mathrm{sgn}(b)\sqrt{b^2 - 4c_1}\right|$. For convenience, we use $\bar{r}_{c_1}(c_{23})$ (respectively, $\bar{r}_{c_{23}}(c_1)$) to denote $\bar{r}(c_1, c_{23})$ considered as a single-variable function with a fixed c_1 (respectively, c_{23}). Our parameter optimization problem is now defined by

$$(c_1^*, c_{23}^*) = \underset{(c_1, c_{23})}{\mathrm{argmin}}\, \bar{r}(c_1, c_{23}).$$

The following three lemmas provide us with a closed-form solution of this minimization problem, if the smallest and largest eigenvalues of $\boldsymbol{A}_\alpha$ are given.

Lemma 5.3.3. *For any fixed c_1 and c_{23}, $\hat{r}(c_1, c_{23}, a_\alpha)$ is maximized at either a_α^{min} or a_α^{max}, the smallest and largest eigenvalues of $\boldsymbol{A}_\alpha$, respectively.*

Proof. Consider $\hat{r}(c_1, c_{23}, a_\alpha)$ in (5.20) as a continuous function of a positive variable $a_\alpha \in \left[a_\alpha^{min}, a_\alpha^{max}\right]$, with fixed c_1 and c_{23}:

$$\hat{r}(c_1, c_{23}, a_\alpha) = \frac{1}{2}\left| b(c_1, c_{23}, a_\alpha) + \operatorname{sgn}\left(b(c_1, c_{23}, a_\alpha)\right)\sqrt{b^2(c_1, c_{23}, a_\alpha) - 4c_1}\right|.$$

We distinguish between two regimes: (I): $b^2(c_1, c_{23}, a_\alpha) < 4c_1$, the square-root term is imaginary, and we simply get $\hat{r}(c_1, c_{23}, a_\alpha) = \sqrt{c_1}$; (II): $b^2(c_1, c_{23}, a_\alpha) \geq 4c_1$,the square-root term is real and the derivative of $\hat{r}(c_1, c_{23}, a_\alpha)$ with respect to a_α is

$$\frac{\partial \hat{r}(c_1, c_{23}, a_\alpha)}{\partial a_\alpha} = -\frac{c_{23}}{2}\operatorname{sgn}\left(b(c_1, c_{23}, a_\alpha)\right)\left[1 + \frac{|b(c_1, c_{23}, a_\alpha)|}{\sqrt{b^2(c_1, c_{23}, a_\alpha) - 4c_1}}\right].$$

We ignore the irrelevant choice $c_{23} = 0$, for which the method is obviously not convergent. Using $b(c_1, c_{23}, a_\alpha) = -c_{23}\left(a_\alpha - \frac{1+c_1}{c_{23}}\right)$, we get

$$\frac{\partial \hat{r}(c_1, c_{23}, a_\alpha)}{\partial a_\alpha} = \frac{|c_{23}|}{2}\operatorname{sgn}\left(a_\alpha - \frac{1+c_1}{c_{23}}\right)\left[1 + \frac{|b(c_1, c_{23}, a_\alpha)|}{\sqrt{b^2(c_1, c_{23}, a_\alpha) - 4c_1}}\right].$$

Notice that $\hat{r}(c_1, c_{23}, a_\alpha)$ is a symmetric function of $a_\alpha - (1+c_1)/c_{23}$ and it is furthermore convex because its derivative is strictly negative for $a_\alpha < (1+c_1)/c_{23}$ and positive for $a_\alpha > (1+c_1)/c_{23}$. It follows that, regardless of the sign of $b^2(c_1, c_{23}, a_\alpha) - 4c_1$ throughout the regime $a_\alpha \in [a_\alpha^{min}, a_\alpha^{max}]$, there exists no local maximum of $\hat{r}(c_1, c_{23}, a_\alpha)$. Hence, $\bar{r}(c_1, c_{23}) = \max\left(\hat{r}(c_1, c_{23}, a_\alpha^{min}), \hat{r}(c_1, c_{23}, a_\alpha^{max})\right)$. □

Lemma 5.3.4. *For any fixed c_1, $c_{23}^* = 2(1+c_1)/(a_\alpha^{max} + a_\alpha^{min})$.*

Proof. Consider $\hat{r}(c_1, c_{23}, a_\alpha)$ as a continuous function of a real variable c_{23} with c_1 and a_α fixed. For $b^2(c_1, c_{23}, a_\alpha) \geq 4c_1$, the derivative of $\hat{r}(c_1, c_{23}, a_\alpha)$ with respect to c_{23} is

$$\frac{\partial \hat{r}(c_1, c_{23}, a_\alpha)}{\partial c_{23}} = \frac{|a_\alpha|}{2}\operatorname{sgn}\left(c_{23} - \frac{1+c_1}{a_\alpha}\right)\left[1 + \frac{|b(c_1, c_{23}, a_\alpha)|}{\sqrt{b^2(c_1, c_{23}, a_\alpha) - 4c_1}}\right].$$

As in Lemma 5.3.3, the meeting point of $\hat{r}(c_1, c_{23}, a_\alpha^{max})$ and $\hat{r}(c_1, c_{23}, a_\alpha^{min})$, which lies between $(1+c_1)/a_\alpha^{max}$ and $(1+c_1)/a_\alpha^{min}$, is where $\hat{r}(c_1, c_{23}, a_\alpha)$ is minimized with respect to c_{23} and then the optimal c_{23} enforces $\hat{r}(c_1, c_{23}, a_\alpha^{max}) = \hat{r}(c_1, c_{23}, a_\alpha^{min})$ resulting in $1 + c_1 - c_{23}a_\alpha^{max} = -(1 + c_1 - c_{23}a_\alpha^{min})$, leading to $c_{23}^* = \frac{2(1+c_1)}{a_\alpha^{min} + a_\alpha^{max}}$. For $b^2(c_1, c_{23}, a_\alpha) < 4c_1$, we have $\hat{r}(c_1, c_{23}, a_\alpha) = \sqrt{c_1}$ which is irrelevant to c_{23}. □

Plugging c_{23}^* into (5.20) yields

$$\bar{r}_{c_{23}^*}(c_1) = \frac{1}{2}\left|\mu(1+c_1) + \sqrt{\mu^2(1+c_1)^2 - 4c_1}\right|, \tag{5.21}$$

where $\mu = (\kappa - 1)/(\kappa + 1) \in (0, 1)$ with $\kappa = \frac{a_\alpha^{max}}{a_\alpha^{min}} > 1$ the condition number of $\boldsymbol{A}_\alpha$. Lemma 5.3.5 provides the optimal c_1, which minimizes $\bar{r}_{c_{23}^*}(c_1)$ in (5.21).

Lemma 5.3.5. $c_1^* = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2$.

Proof. Consider two regimes: (I) $\mu^2(1+c_1)^2 \le 4c_1$; (II) $\mu^2(1+c_1)^2 \ge 4c_1$. For (I), we have $\bar{r}_{c_{23}^*}(c_1) = \sqrt{c_1}$ and $c_1 \in \left[c_1^-, c_1^+\right]$ where $c_1^{\pm} = \left(2-\mu^2 \pm \sqrt{4-4\mu^2}\right)/\mu^2$ are the two solutions of $\mu^2(1+c_1)^2 = 4c_1$. Evidently, c_1^- is the solution to minimize $r_{c_{23}^*}(c_1)$.

For (II), we have $c_1 \le c_1^-$ or $c_1 \ge c_1^+$. We first note that

$$\mu(1+c_1) + \sqrt{\mu^2(1+c_1)^2 - 4c_1} > 0.$$

and then $\bar{r}_{c_{23}^*}(c_1) = \frac{1}{2}\left[\mu(1+c_1) + \sqrt{\mu^2(1+c_1)^2 - 4c_1}\right]$. Evidently, the derivative of $\bar{r}_{c_{23}^*}(c_1)$ with respect to c_1 is

$$\frac{\partial \bar{r}_{c_{23}^*}(c_1)}{\partial c_1} = \frac{1}{2}\left(\mu + \frac{\mu^2(1+c_1)-2}{\sqrt{\mu^2(1+c_1)^2-4c_1}}\right). \tag{5.22}$$

Note that $\frac{\mu^2(1+c_1)-2}{\sqrt{\mu^2(1+c_1)^2-4c_1}}$ is positive for $c_1 > \frac{2}{\mu^2} - 1$ and negative for $c_1 < \frac{2}{\mu^2} - 1$, hence, (5.22) is positive for $c_1 > c_1^+$, whereas for $c_1 < c_1^-$ we have

$$\begin{aligned} \frac{\partial \bar{r}_{c_{23}^*}(c_1)}{\partial c_1} &= \frac{1}{2}\left(\mu - \sqrt{\frac{(\mu^2(1+c_1)-2)^2}{\mu^2(1+c_1)^2-4c_1}}\right) = \frac{\mu}{2}\left(1 - \sqrt{\frac{\mu^4(1+c_1)^2-4\mu^2(1+c_1)+4}{\mu^4(1+c_1)^2-4\mu^2 c_1}}\right) \\ &= \frac{\mu}{2}\left(1 - \sqrt{1 + \frac{4(1-\mu^2)}{\mu^4(1+c_1)^2-4\mu^2 c_1}}\right) < 0 \end{aligned}$$

The last inequality is due to the fact that that $0 < \mu < 1$ and $\mu^2(1+c_1)^2 - 4c_1 > 0$. Evidently, the optimal c_1 to minimize $\bar{r}_{c_{23}^*}(c_1)$ is either c_1^- or c_1^+. However, $\bar{r}_{c_{23}^*}(c_1) > 1$ for c_1^+ that $c_1^- = \left(2-\mu^2-\sqrt{4-4\mu^2}\right)/\mu^2$ is the solution. Substituting the expression of μ into c_1^-, we get the desired result. □

Substituting the expression of c_1^* into (5.21), we get the worst-case convergence factor r^* with optimal c_1^* and c_{23}^*

$$\bar{r}^* \triangleq \bar{r}(c_1^*, c_{23}^*) = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}. \tag{5.23}$$

Note that $c_1^* = (\bar{r}^*)^2$ indicates that the history term becomes significant if the problem is ill-conditioned.

Remark 5.3. The optimal ACF of the fixed-coefficient variant of SESOP-TG-1 is found to be $\bar{r}^* = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, where κ is the condition number of $\boldsymbol{A}_\alpha$, optimized over $\alpha \in (0,1]$. (The optimal choice of α will be discussed further below.) By using the definition of κ, the optimal coefficient for the preconditioned gradient term as specified in Lemma 5.3.4 can be written as

$$c_{23}^* = \frac{4}{a_\alpha^{min}(\sqrt{\kappa}+1)^2}. \tag{5.24}$$

Comparing SESOP-TG-1 to SESOP-TG-0. By setting $c_1 = 0$, we reduce to SESOP-TG-0 and the optimal c_{23} becomes

$$c_{23}^* = \frac{2}{a_\alpha^{min}(\kappa+1)}. \tag{5.25}$$

Furthermore, by (5.21), the asymptotic convergence factor of SESOP-TG-0 is given by

$$\bar{r}^* = \frac{\kappa-1}{\kappa+1}. \tag{5.26}$$

Comparing (5.23) with (5.26), we see the significant improvement provided by the use of a single history direction, with the condition number replaced by the square root. This result is reminiscent of the conjugate gradients (CG) method compared to steepest descent (SD) for quadratic problems, and indeed there is an equivalence between SESOP and CG (respectively, SD) for the case of single history (respectively, no history) with $\alpha = 1$. We note that the condition number in our scheme is that of $\boldsymbol{A}_\alpha$, not $\boldsymbol{A}$, which differs from the case where we do not use the CGC direction. The advantage of SESOP is in allowing the addition of various search directions, at the cost of optimizing the coefficients. This study is partly aimed at reducing this cost.

5.3.2 Towards Optimizing the Condition Number of $\boldsymbol{A}_\alpha$

The previous analysis indicates that we should aim to minimize κ, the condition number of $\boldsymbol{A}_\alpha$. In certain cases, particularly when $\boldsymbol{A}$ is a circulant matrix (typically the discretization of an elliptic PDE with constant coefficients on a rectangular or infinite domain), this can be done by means of Fourier analysis [75], as discussed in Section 5.3.3. Here, we begin with a more general discussion to gain insight into this matter. We consider the case of Galerkin coarsening, $\boldsymbol{A}_H = (\boldsymbol{I}_H^h)^{\mathcal{T}} \boldsymbol{A} \boldsymbol{I}_H^h$, and $\boldsymbol{P} = \boldsymbol{I}$. Guided by [96], we consider the case where the columns of the prolongation matrix $\boldsymbol{I}_H^h$ are comprised of a subset of the eigenvectors of $\boldsymbol{A}$.

Lemma 5.3.6. *Let $\{a_i, \boldsymbol{a}_i\}$, $i = 1, \ldots, N$, denote the eigenvalues and eigenvectors of $\boldsymbol{A}$. Assume that the columns of the prolongation matrix $\boldsymbol{I}_H^h$ are comprised of a subset of the eigenectors of $\boldsymbol{A}$, and denote by $\mathcal{R}(\boldsymbol{I}_H^h)$ the range of the prolongation, that is, the subspace spanned by the columns of $\boldsymbol{I}_H^h$. Denote*

$$\begin{aligned} a_{fmax} &= \max_{i:\boldsymbol{a}_i \notin \mathcal{R}(\boldsymbol{I}_H^h)} a_i, \qquad & a_{fmin} &= \min_{i:\boldsymbol{a}_i \notin \mathcal{R}(\boldsymbol{I}_H^h)} a_i, \\ a_{cmax} &= \max_{i:\boldsymbol{a}_i \in \mathcal{R}(\boldsymbol{I}_H^h)} a_i, \qquad & a_{cmin} &= \min_{i:\boldsymbol{a}_i \in \mathcal{R}(\boldsymbol{I}_H^h)} a_i. \end{aligned}$$

Then, the condition number of $\boldsymbol{A}_\alpha$ in (5.19) *with $\boldsymbol{P} = \boldsymbol{I}$ is given by*

$$\kappa = \frac{\max(\alpha a_{fmax},\ \alpha a_{cmax} + 1 - \alpha)}{\min(\alpha a_{fmin},\ \alpha a_{cmin} + 1 - \alpha)}. \tag{5.27}$$

Proof. Any eigenvector $\boldsymbol{a}_i \in R(\boldsymbol{I}_H^h)$ (respectively, $\boldsymbol{a}_i \notin \mathcal{R}(\boldsymbol{I}_H^h)$)) is an eigenvector of the CGC matrix $\boldsymbol{I}_H^h \boldsymbol{A}_H^{-1} (\boldsymbol{I}_H^h)^{\mathcal{T}} \boldsymbol{A}$, with eigenvalue 1 (respectively, 0), because if $\boldsymbol{a}_i \in \mathcal{R}(\boldsymbol{I}_H^h)$ then it can be written as $\boldsymbol{a}_i = \boldsymbol{I}_H^h \boldsymbol{e}_j$ (that is, it is the jth column for some j), so

$$\left[\boldsymbol{I}_H^h \boldsymbol{A}_H^{-1} (\boldsymbol{I}_H^h)^{\mathcal{T}} \boldsymbol{A}\right] \boldsymbol{a}_i = \boldsymbol{I}_H^h \left[(\boldsymbol{I}_H^h)^{\mathcal{T}} \boldsymbol{A} \boldsymbol{I}_H^h\right]^{-1} \left[(\boldsymbol{I}_H^h)^{\mathcal{T}} \boldsymbol{A} \boldsymbol{I}_H^h\right] \boldsymbol{e}_j = \boldsymbol{I}_H^h \boldsymbol{e}_j = \boldsymbol{a}_i,$$

whereas if $\boldsymbol{a}_i \notin \mathcal{R}(\boldsymbol{I}_H^h)$ then it is orthogonal to the columns of $\boldsymbol{I}_H^h$, so

$$\left[\boldsymbol{I}_H^h \boldsymbol{A}_H^{-1} (\boldsymbol{I}_H^h)^{\mathcal{T}} \boldsymbol{A}\right] \boldsymbol{a}_i = \left[\boldsymbol{I}_H^h \boldsymbol{A}_H^{-1} (\boldsymbol{I}_H^h)^{\mathcal{T}}\right] a_i \boldsymbol{a}_i = \boldsymbol{0}.$$

It follows that the eigenvectors of $\boldsymbol{A}_\alpha$ are $\boldsymbol{a}_i$, with eigenvalues given by

$$a_\alpha^i = \begin{cases} \alpha a_i + 1 - \alpha & \text{if } \boldsymbol{a}_i \in \mathcal{R}(\boldsymbol{I}_H^h), \\ \alpha a_i & \text{otherwise.} \end{cases}$$

By using the definition of κ, a_{fmax}, a_{cmax}, a_{fmin}, and a_{cmin}, the desired result is derived. $\square$

We see that the second term in $\boldsymbol{A}_\alpha$, which corresponds to the direction provided by the CGC, increases the eigenvalues associated with the columns of the prolongation by $1-\alpha$. It thus follows from (5.27) that, to obtain any advantage at all from the CGC direction in reducing κ, the eigenvector associated with the smallest a_i must be included amongst the columns of the prolongation, and therefore $a_{cmin} \leq a_{fmin}$. Similarly, considering the numerator in (5.27), it is clearly advantageous that the eigenvector corresponding to the largest a_i not be included in the range of the prolongation. With these assumptions, we obtain the following result for the optimal α which minimizes κ.

Theorem 5.1. *Assume $a_{cmin} \leq a_{fmin}$ and $a_{cmax} \leq a_{fmax}$. Then, the condition number κ is minimized by choosing*

$$\alpha_{opt} \triangleq \frac{1}{1+a_{fmin}-a_{cmin}} \leq 1,$$

resulting in the optimal κ,

$$\kappa_{opt} \triangleq \begin{cases} \frac{a_{fmax}}{a_{fmin}} & \text{if } a_{fmax}-a_{fmin} \geq a_{cmax}-a_{cmin}, \\ 1+\frac{a_{cmax}-a_{cmin}}{a_{fmin}} & \text{otherwise.} \end{cases} \tag{5.28}$$

Proof. To minimize κ, the optimal α should minimize the numerator and maximize the denominator. Note that $\alpha_{top} = \frac{1}{1+a_{fmax}-a_{cmax}}$ and $\alpha_{bot} = \frac{1}{1+a_{fmin}-a_{cmin}}$ are the ones to minimize the numerator and maximize the denominator, respectively. For $a_{fmax}-a_{fmin} \geq a_{cmax}-a_{cmin}$, we have $\alpha_{top} \leq \alpha_{bot}$ and then:

1. $\alpha \in (0, \alpha_{top}) \Rightarrow \kappa = \frac{\alpha a_{cmax}+1-\alpha}{\alpha a_{fmin}}$, $\frac{d\kappa}{d\alpha} = -\frac{1}{\alpha^2 a_{fmin}} < 0$, choosing $\alpha = \alpha_{top}$.
2. $\alpha \in (\alpha_{top}, \alpha_{bot}) \Rightarrow \kappa = \frac{a_{fmax}}{a_{fmin}}$, choosing any $\alpha \in (\alpha_{top}, \alpha_{bot})$.
3. $\alpha \in (\alpha_{bot}, 1) \Rightarrow \kappa = \frac{\alpha a_{fmax}}{\alpha a_{cmin}+1-\alpha}$, $\frac{d\kappa}{d\alpha} = \frac{a_{fmax}}{(\alpha a_{cmin}+1-\alpha)^2} > 0$, choosing $\alpha = \alpha_{bot}$.

Evidently, the optimal α can be any one between α_{top} and α_{bot} yielding $\kappa_{opt} = \frac{a_{fmax}}{a_{fmin}}$. By applying the same reasoning to $a_{fmax}-a_{fmin} < a_{cmax}-a_{cmin}$, the optimal α achieves at $\alpha_{opt} \triangleq \alpha_{bot}$. Summarizing, α_{bot} is the optimal solution for both cases. Substituting $\alpha_{opt} = \alpha_{bot}$ into (5.27), we get the desired result.$\square$

Remark 5.4. Notice that κ_{opt} is either equal to a_{fmax}/a_{fmin} or at most "slightly larger" because

$$1+\frac{a_{cmax}-a_{cmin}}{a_{fmin}} = \frac{a_{fmax}}{a_{fmin}} + 1 - \frac{a_{fmax}-a_{cmax}+a_{cmin}}{a_{fmin}} < \frac{a_{fmax}}{a_{fmin}} + 1\,.$$

That is, even in the regime $a_{fmax}-a_{fmin} < a_{cmax}-a_{cmin}$, the optimal condition number κ_{opt} is increased by less than 1. Note that $\kappa = a_{fmax}/a_{fmin}$ yields a convergence factor (with no history) of $\mu = \frac{\kappa-1}{\kappa+1}$, which matches that of the classical TG algorithm with optimally weighted Richardson relaxation followed by CGC. In Section 5.3.4, we will show the connection of κ_{opt} presented here with the so-called *h*-ellipticity measure, which is a qualitative criterion for the existence of local smoothers for a given elliptic PDE [74]. Finally, we note that the optimal prolongation is obtained by choosing the columns of I_H^h to be the eigenvectors associated with the smallest eigenvalues of $\boldsymbol{A}$ (similarly to the classical TG case [96]). This clearly minimizes the ratio a_{fmax}/a_{fmin}. Furthermore, this choice yields $a_{cmax} \leq a_{fmin}$, so if $a_{fmax}-a_{fmin} < a_{cmax}-a_{cmin}$ (so $\kappa_{opt} > a_{fmax}/a_{fmin}$), then we obtain $\kappa_{opt} = 1+\frac{a_{cmax}-a_{cmin}}{a_{fmin}} < 2$.

5.3.3 Optimizing the Condition Number of A_α in Practice

As our analysis indicates, optimizing α so as to minimize κ is a crucial step towards obtaining an optimal convergence factor. In general, the complexity of minimizing κ is very high and the explicit results of Section 5.3.2 are only valid when the prolongation is comprised of eigenvectors of $\boldsymbol{A}$, which is impractical, because the prolongation needs to be very sparse for efficiency. In certain cases, such as when $\boldsymbol{A}$ results from discretizing elliptic PDEs with constant or slowly varying coefficients, Local Fourier Analysis (LFA) [9, 75] can be used in conjunction with the analysis of Section 5.3.2 to yield effective approximate results.

LFA, also called Local Mode Analysis, is a useful quantitative tool for estimating the asymptotic convergence factor ACF of MG methods for elliptic PDEs with constant coefficients. We describe LFA here briefly, and for further details refer the reader to standard multigrid textbooks or to [75], a comprehensive book on LFA for MG. We consider the two-dimensional case for simplicity. For $\boldsymbol{A}$ that is a discretization of a constant-coefficient elliptic PDE on an infinite or doubly periodic domain of uniform mesh-size h, grid-based functions of the form $\psi(\boldsymbol{\theta},x,y) = e^{\imath\theta_1 x/h}e^{\imath\theta_2 y/h}$ with $\boldsymbol{\theta} = (\theta_1,\theta_2) \in [-\pi,\pi)^2$ and $\imath = \sqrt{-1}$, are eigenfunctions of $\boldsymbol{A}$. Furthermore, if standard shift-invariant prolongation is used, such as bilinear or bicubic interpolation, then appropriate subsets of dimension four of the eigenfunctions $\psi(\boldsymbol{\theta},x,y)$ form subspaces that are invariant under multiplication by $\boldsymbol{A}_\alpha$ (as in standard two-level analysis). The upshot is that we can compute the eigenvalues of $\boldsymbol{A}_\alpha$ at a fairly moderate cost, and use linesearch to find α which optimizes the condition number of $\boldsymbol{A}_\alpha$. This approach is demonstrated later in numerical examples, including the option of saving computational cost by optimizing α on relatively coarse grids.

We can furthermore obtain a very cheap approximation to the optimal α and κ by making the simplifying assumptions that are commonly used in computing the so-called smoothing factor of relaxation (known as smoothing analysis) as follows. Denote by $a(\boldsymbol{\theta})$ the eigenvalue of $\boldsymbol{A}$ associated with the grid-function $\psi(\boldsymbol{\theta},x,y)$, and partition the $\boldsymbol{\theta}$ domain into low- and high-frequencies as in Definition 5.3.7.

Definition 5.3.7 (Low- and High-frequency components)**.**

$$\begin{aligned} a(\boldsymbol{\theta}) \text{ is a low-frequency component} &\iff \boldsymbol{\theta} \in T^{\text{low}} := \left[-\tfrac{\pi}{2},\tfrac{\pi}{2}\right)^2, \\ a(\boldsymbol{\theta}) \text{ is a high-frequency component} &\iff \boldsymbol{\theta} \in T^{\text{high}} := [-\pi,\pi)^2 \setminus \left[-\tfrac{\pi}{2},\tfrac{\pi}{2}\right)^2. \end{aligned}$$

Smoothing analysis simplifies by assuming that the CGC acts as a projection onto the high-frequency subspace, that is, it has no effect on high-frequency error components, while it eliminates exactly low-frequency error components. Under these simplifying assumptions, we obtain

$$\begin{aligned} a_{fmax} &= \max_{\boldsymbol{\theta}\in T^{\text{high}}} a(\boldsymbol{\theta}), \quad a_{fmin} = \min_{\boldsymbol{\theta}\in T^{\text{high}}} a(\boldsymbol{\theta}), \\ a_{cmax} &= \max_{\boldsymbol{\theta}\in T^{\text{low}}} a(\boldsymbol{\theta}), \quad a_{cmin} = \min_{\boldsymbol{\theta}\in T^{\text{low}}} a(\boldsymbol{\theta}). \end{aligned} \tag{5.29}$$

Moreover, the eigenvalues of the second term in $\boldsymbol{A}_\alpha$ are given by $\gamma(\boldsymbol{\theta}) \triangleq \frac{a(\boldsymbol{\theta})}{a_H(2\boldsymbol{\theta})}$ with $a_H(2\boldsymbol{\theta})$ the eigenvalues of $\boldsymbol{A}_H$ for $\boldsymbol{\theta} \in T^{\text{low}}$ [97]. Now, we can write κ as:

$$\kappa = \frac{\max\left(\alpha a_{fmax},\ \alpha a_{cmax} + (1-\alpha)\,\gamma_{max}(\boldsymbol{\theta})\right)}{\min\left(\alpha a_{fmin},\ \alpha a_{cmin} + (1-\alpha)\,\gamma_{min}(\boldsymbol{\theta})\right)}, \tag{5.30}$$

where $\gamma_{max}(\boldsymbol{\theta})$ and $\gamma_{min}(\boldsymbol{\theta})$ denote the maximum and minimum value of $\gamma(\boldsymbol{\theta})$, respectively. Similarly to

Theorem 5.1, the optimal α is given by

$$\alpha_{opt} = \frac{1}{1 + \frac{a_{fmin} - a_{cmin}}{\gamma_{min}(\boldsymbol{\theta})}}. \tag{5.31}$$

We note that, for elliptic PDEs, the accuracy of using the approximations (5.29) and $\gamma(\boldsymbol{\theta})$ depend on the order of intergrid transfers [98, 97]. In the following numerical tests, we evaluate experimentally the accuracy of using (5.31) to estimate the fixed stepsizes in practice.

5.3.4 A Connection with the h-ellipticity Measure

Using Definition 5.3.7, we define an "idealized" MG method as a two-grid method that affects high-frequency error components only on the fine grid and eliminates all low-frequency error components via the CGC, corresponding to (5.29). Then, the ACF of an idealized SESOP-TG-1 is $\frac{\sqrt{\kappa_{opt}}-1}{\sqrt{\kappa_{opt}}+1}$ with $\kappa_{opt} = \frac{a_{fmax}}{a_{fmin}}$, corresponding to the discussion of Section 5.3.2 for ill-conditioned problems. Alternatively, we can express this idealized ACF in terms of $E_h(\boldsymbol{A}) = \frac{1}{\kappa_{opt}}$, which is known in the literature as the h-ellipticity measure, obtaining $r = \frac{1-\sqrt{E_h}}{1+\sqrt{E_h}}$. For SESOP-TG-0, in contrast, the ACF becomes $r = \frac{1-E_h}{1+E_h}$, which is the well-known smoothing factor of optimally damped Jacobi relaxation for symmetric problems [74].

5.3.5 Numerical Tests—Continued

In this section, we first test the accuracy of using (5.17) to predict the ACF of SESOP-TG-1 with fixed stepsizes. Then, compare the ACF with minimization over the subspace to that obtained with optimized fixed stepsizes. Finally, we examine the performance of the two proposed heuristic methods (cf. Section 5.3.3) for estimating the fixed stepsizes. The rotated anisotropic diffusion problem is chosen as the test problem,

$$u_{ss} + \varepsilon u_{tt} = f, \tag{5.32}$$

where u_{ss} and u_{tt} denote the second partial derivatives of u in the (s,t) coordinate system. Denote by ϕ the angle between (s,t) and the grid-aligned coordinate system, (x,y). We rewrite (5.32) as

$$(C^2 + \varepsilon S^2)u_{xx} + 2(1-\varepsilon)CSu_{xy} + (\varepsilon C^2 + S^2)u_{yy} = f, \tag{5.33}$$

where $C = \cos\phi$ and $S = \sin\phi$. Using a nine-point stencil,

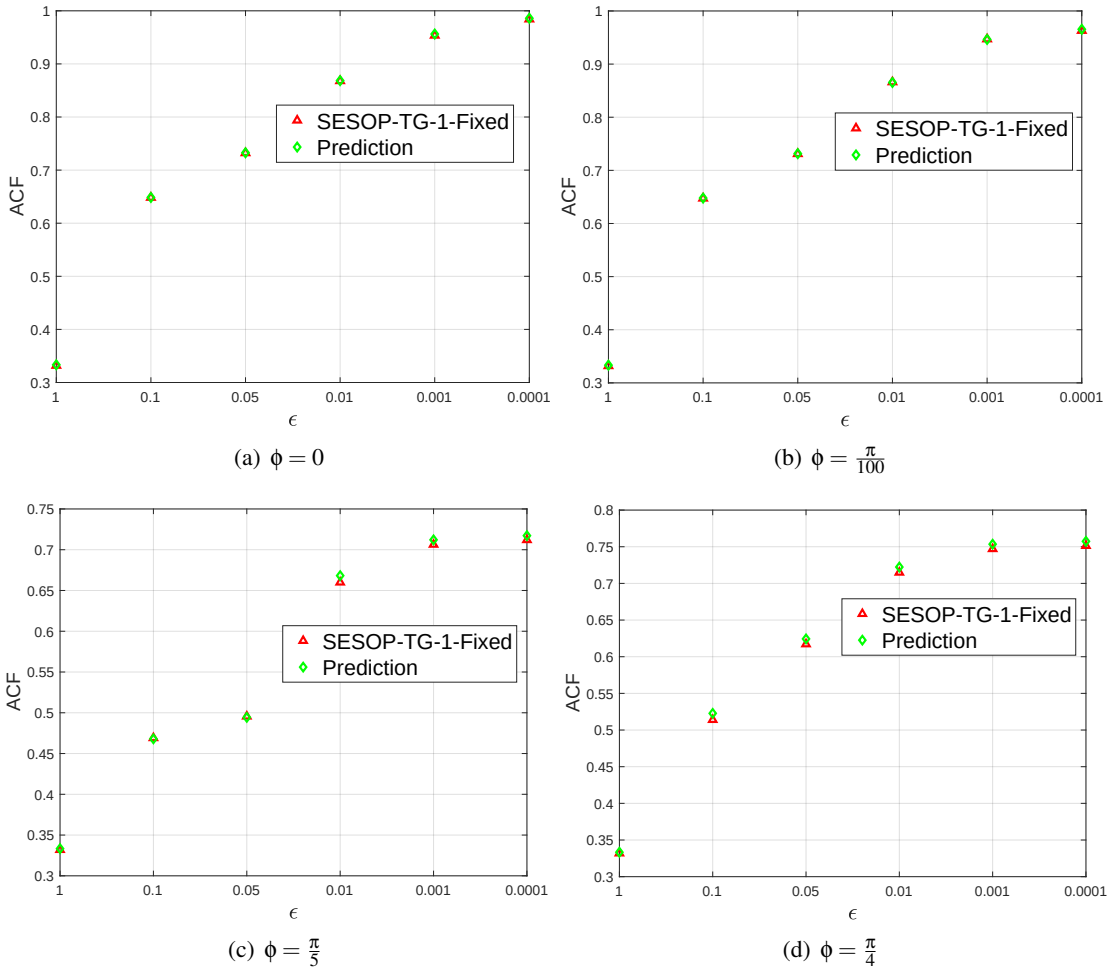
$$\begin{bmatrix} -\frac{1}{2}(1-\varepsilon)CS & \varepsilon C^2 + S^2 & \frac{1}{2}(1-\varepsilon)CS \\ C^2 + \varepsilon S^2 & -2(1+\varepsilon) & C^2 + \varepsilon S^2 \\ \frac{1}{2}(1-\varepsilon)CS & \varepsilon C^2 + S^2 & -\frac{1}{2}(1-\varepsilon)CS \end{bmatrix},$$

to discretize (5.33) on a uniform grid with mesh-size h and prescribed boundary conditions, we get a linear system of the form

$$\boldsymbol{A}^h \boldsymbol{u}^h = \boldsymbol{f}^h. \tag{5.34}$$

ACF Prediction

In our first test, we simply choose $c_3 = 1$, and $\kappa = \frac{1}{E_h}$, and then use Lemmas 5.3.4 and 5.3.5 to compute c_1 and c_2. The resulting algorithm is referred to as "SESOP-TG-1-Fixed". We discretize (5.33) on a 256×256 grid, imposing Dirichlet boundary conditions, and we employ bilinear prolongation and full-weighted residual transfers. The coarse-grid operators are defined by direct rediscretization. Finally, we set $\mathcal{P} = \boldsymbol{I}$, i.e., no preconditioning. The ACF achieved in practice by SESOP-TG-1-Fixed is evaluated as the geometric mean of the convergence factor per iteration in the last 5 iterations, which are terminated at 500 iterations or when the residual norm is smaller than 10^{-8}, whichever comes first. The convergence factor at the kth iteration is defined by the ratio of the successive residual norms $\|\boldsymbol{r}_k^h\|_2/\|\boldsymbol{r}_{k-1}^h\|_2$, where $\boldsymbol{r}_k^h = \boldsymbol{f}^h - \boldsymbol{A}^h\boldsymbol{u}_k^h$ and $\boldsymbol{u}_k^h$ is the approximate solution at the kth iteration. In Figure 5.5, we see that the ACF predicted by the spectral radius of $\boldsymbol{\Upsilon}$ in (5.17) matches the practical results well.



(a) $\phi = 0$ (b) $\phi = \frac{\pi}{100}$

(c) $\phi = \frac{\pi}{5}$ (d) $\phi = \frac{\pi}{4}$

Figure 5.5: Comparison of the predicted ACF to the convergence factor achieved in practice.

Comparison between Subspace Minimization and Optimal Fixed Stepsizes

Next, we compare the ACFs of three different approaches to determining the fixed stepsizes: 1) SESOP-TG-1-Fixed defined above; 2) subspace minimization (classical SESOP); 3) optimized fixed stepsizes. The optimized stepsizes are obtained by minimizing the true condition number κ of $\boldsymbol{A}_\alpha$ in (5.19) by employing linesearch over α. The test problem remains unchanged, except that we test both bilinear and bicubic

Table 5.1: Comparison of the ACF of SESOP-TG-1-Fixed (TG-1), classical subspace minimization (SESOP), and optimized stepsizes (Opt). The idealized convergence factor of Section 5.3.4, based on the h-ellipticity measure, is included as a benchmark (Idealized).

ϕ	ε	Bilinear			Bicubic			Idealized
		TG-1	SESOP	Opt	TG-1	SESOP	Opt	
0	1	0.333	0.332	0.332	0.333	0.333	0.331	0.333
$\frac{\pi}{6}$	10^{-3}	0.669	0.561	0.563	0.587	0.533	0.532	0.587
$\frac{\pi}{6}$	10^{-4}	0.676	0.563	0.565	0.588	0.534	0.533	0.588
$\frac{\pi}{4}$	10^{-3}	0.753	0.500	0.535	0.653	0.454	0.443	0.446
$\frac{\pi}{4}$	10^{-4}	0.757	0.502	0.537	0.658	0.451	0.445	0.446

prolongations and use 64×64 grids. The ACF achieved in practice is estimated as above by the geometric mean of the last 5 iterations when the algorithm reaches 500 iterations or the residual norm is smaller than 10^{-8}. Note also that κ_{opt} of Section 5.3.4, i.e., one over the h-ellipticity measure, is used here to represent the idealized ACF for comparison.

From Table 5.1, we clearly see that optimized fixed stepsizes yield a lower ACF than SESOP-TG-1-Fixed for the rotated anisotropic diffusion problem, and in fact its ACF is at least as good as that obtained by subspace minimization. This suggests that, if we can estimate the optimal stepsizes efficiently, then we can significantly reduce computation time because the subspace minimization, in general, is expensive. Also, we see that bicubic prolongation yields a lower ACF than bilinear prolongation, consistent with [98].

The relative disadvantage of SESOP-TG-1-Fixed for rotated anisotropic diffusion stems mainly from fixing $c_3 = 1$, whereas the optimal c_3 is higher for this problem, consistent with the analysis of [97].

Optimizing κ of $\boldsymbol{A}_\alpha$ in Practice

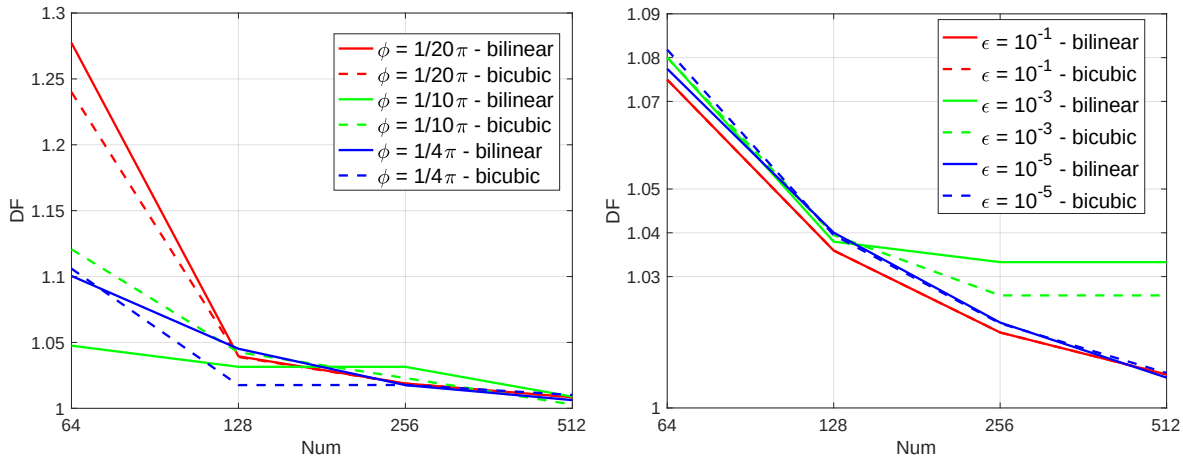
We next study the efficacy of the two heuristic methods proposed in Section 5.3.3—estimating κ_{opt} on a coarse grid or using (5.31)—to select the fixed stepsizes.

In the following tests we use 1024×1024 grids to discretize (5.33). Denote by r_{Num}^{opt} the ACF obtained by optimizing κ of $\boldsymbol{A}_\alpha$ on a $Num \times Num$ grid. Then, we define the "Deterioration Factor" (DF),

$$DF(Num) \triangleq \frac{\log r_{1024}^{opt}}{\log r_{Num}^{opt}},$$

to measure the deterioration of the ACF incurred by optimizing κ on a coarser grid. For example, $DF(Num) = 2$ means that, asymptotically, it takes twice as many iterations of the algorithm which uses fixed stepsizes optimized on coarser grids to achieve the same error reduction as a single iteration with the true optimal stepsize. Additionally, we compare between bilinear and bicubic interpolation [98]. In Figure 5.6, we see $DF(Num) \approx 1.05$ for $Num \geq 128$, which means that determining the stepsizes on 128×128 grids results in just a $\sim 5\%$ asymptotic increase in the required iterations. Moreover, we observe that the use of higher order interpolation tends to reduce the DF.

Now, we study the accuracy of using (5.31) for selecting the fixed stepsizes. From [97, Equation (14)], we know that the accuracy of (5.30) for approximating κ of $\boldsymbol{A}_\alpha$ depends on the order of intergrid transfers.



(a) $\varepsilon = 10^{-3}$. (b) $\phi = \frac{\pi}{30}$.

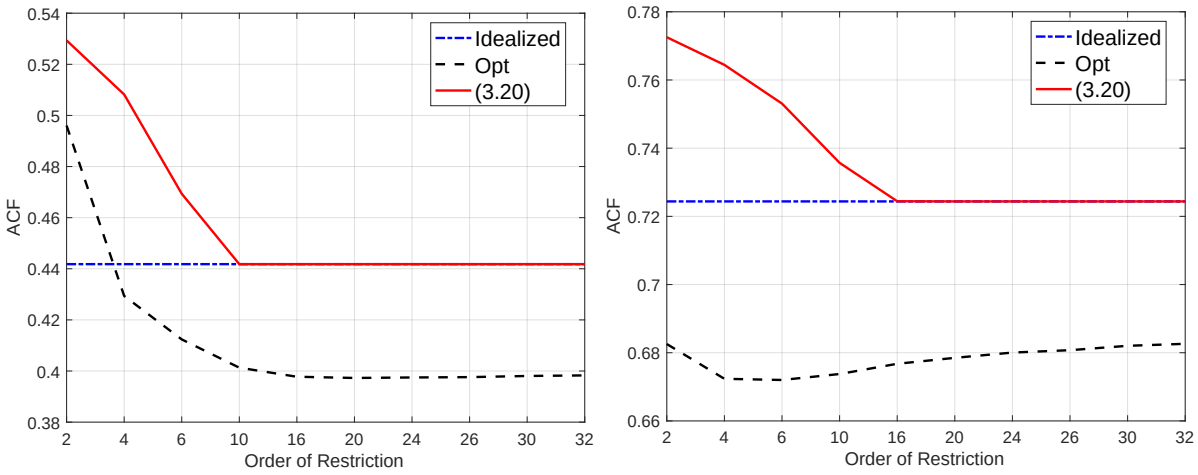
Figure 5.6: $DF(Num)$ for various ε and ϕ.

In Figure 5.7, we show a comparison of ACFs with different orders of intergrid transfers. Note that the intergrid transfers used here have the same order for low and high-frequency modes. From Figure 5.7, we find that (5.31) becomes more accurate and very close to the idealized estimate when the order of restriction increases. This is due to the fact that higher order intergrid transfer operators filter out the high-frequency modes, and therefore the idealized assumptions are more closely satisfied. Note that the optimized algorithm is even better than the the idealized one, as also observed in Table 5.1, because the optimized algorithm takes into account all the modes globally when minimizing κ. Moreover, for the optimized version we see that increasing the order of the restriction begins increase the ACF slightly, bringing it closer to the idealized value. We finally note that the purpose of this test is to academically study the relationship between the accuracy of (5.31) and the use of intergrid transfers and, in practice, it is not cost-effective to use very high order intergrid transfers.

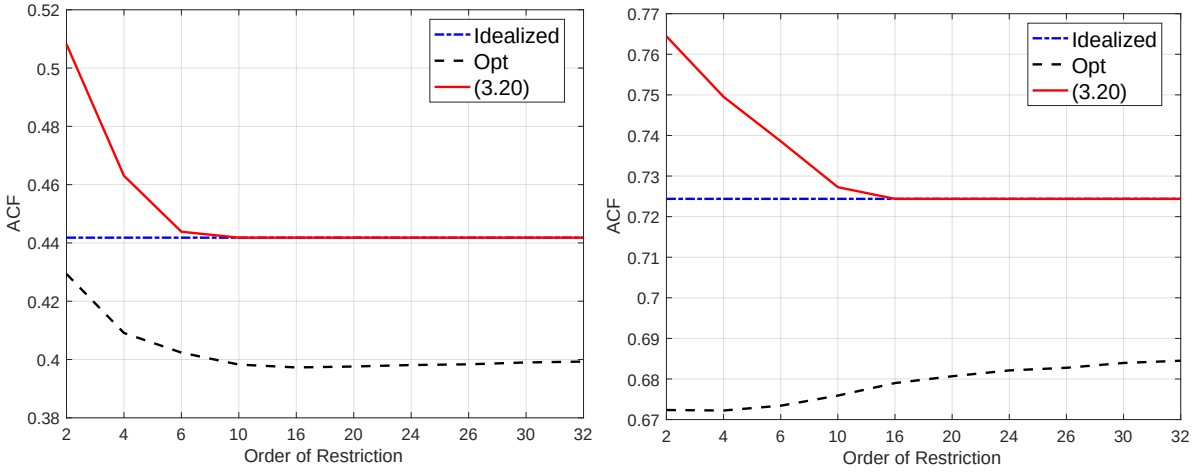
Remark 5.5. In this part we have examined the accuracy of two heuristic methods for determining the fixed stepsizes. In practice, the method shown in Figure 5.6 may be more attractive because the increase of iterations due to computing the stepsizes on a coarse grid is modest. However, for some practical problems, (5.31) is also attractive because its computation is cheaper than working on a coarse grid. Moreover, for many practical problems, we need to solve (5.34) with multiple $\boldsymbol{f}^h$, and then the additional computation for selecting the stepsizes is negligible.

Practical Tests

Now we study the performance of applying the multilevel version (V-cycle) of the proposed scheme to (5.33). The multilevel version of Algorithm 5.1 is denoted by "MG". Using preconditioned conjugate gradients (PCG) with MG as the preconditioner is denoted by "PCG-MG". The fixed-stepsize version of SESOP-MG-1 is denoted by SESOP-MG-1-Opt when α_{opt} is computed on a coarse grid and by SESOP-MG-1-(5.31) when (5.31) α_{opt} is used. We use 1024×1024 grids to discretize (5.33) and obtain α_{opt} for SESOP-MG-1-Opt on 128×128 grids. The coarse problems are obtained by direct rediscretization, and the bilinear prolongation and full-weighted residual transfers are employed. For coarse problems, we use Jacobi relaxation with optimal damping factor as estimated in [78] and $\nu_1 = 2$ and $\nu_2 = 1$. Note that on



(a) $\varepsilon = 10^{-3}$, $\phi = \frac{\pi}{4}$ (b) $\varepsilon = 10^{-3}$, $\phi = \frac{2\pi}{5}$

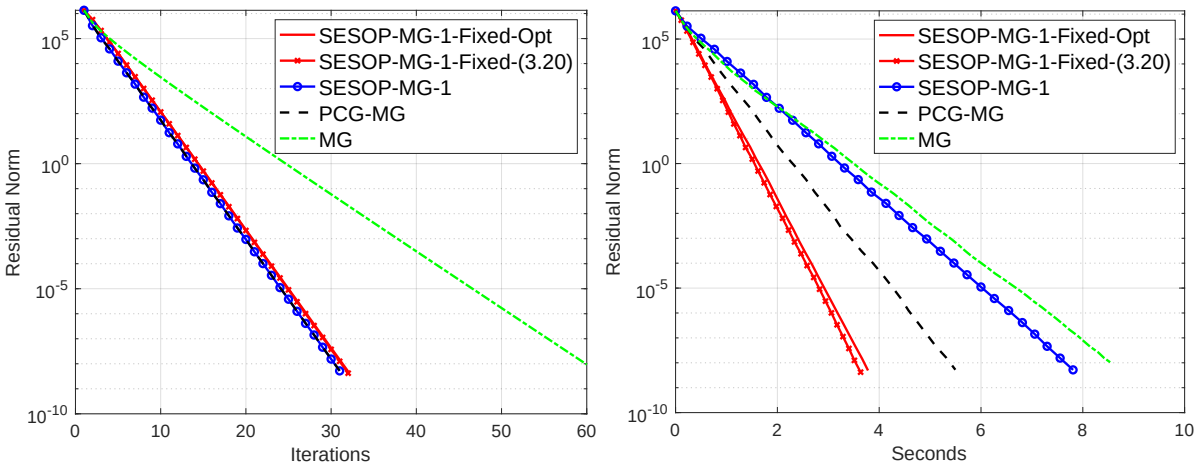


(c) $\varepsilon = 10^{-3}$, $\phi = \frac{\pi}{4}$ (d) $\varepsilon = 10^{-3}$, $\phi = \frac{2\pi}{5}$

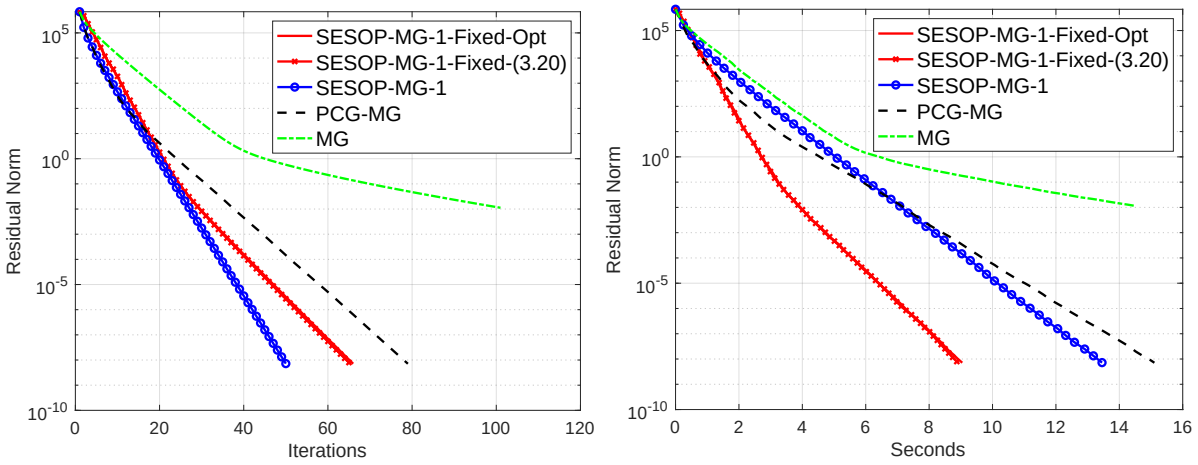
Figure 5.7: Comparison of ACFs for varying order of intergrid transfers, using 512×512 grids and Galerkin coarsening. First row: bilinear interpolation; Second row: bicubic interpolation. "Idealized" and "Opt" refer to Table 5.1.

the finest level, we set $\nu_1 = 0$ and $\nu_2 = 0$ and only one evaluation of the gradient is allowed.

In Figure 5.8(a) ($\varepsilon = 1, \phi = 0$), we see SESOP-MG-1 and PCG-MG perform identically. However, in Figure 5.8(c) ($\varepsilon = 10^{-3}, \phi = \frac{\pi}{4}$), we find that SESOP-MG-1 becomes faster than PCG-MG. Indeed, in this case, it helps to scale the residual before restricting to the coarse for MG (also see the deterioration of MG in Figure 5.8(c)), but SESOP-MG-1 can automatically scale the residual via the subspace minimization [82]. Moreover, from Figures 5.8(b) and 5.8(d), we observe that SESOP-MG-1-Opt and SESOP-MG-1-(5.31) are the fastest methods in terms of CPU time because these two methods inherit the effectiveness of SESOP-MG-1 but need much less CPU time. It is interesting to note that SESOP-MG-1-Opt and SESOP-MG-1-(5.31) work almost the same in these two tests, demonstrating the effectivenss of the proposed strategies for determining the stepsizes.



(a) $\varepsilon = 1, \phi = 0$ (b) $\varepsilon = 1, \phi = 0$



(c) $\varepsilon = 10^{-3}, \phi = \frac{\pi}{4}$ (d) $\varepsilon = 10^{-3}, \phi = \frac{\pi}{4}$

Figure 5.8: Comparison of different methods for (5.33). Test on 1024×1024 grids and use the V-cycle.

5.4 Conclusion

In this chapter, we merge multigrid (MG) optimization with SESOP optimization. The numerical experiments on linear and nonlinear problems illustrate the effectiveness and robustness of our scheme. Moreover, for linear problems, if only one history is used, we derive optimal fixed stepsizes that allow us to avoid the expensive subspace minimization and save computation. Specifically, for elliptic PDEs with constant coefficients, we propose two heuristic methods to estimate the fixed stepsizes based on LFA and smoothing analysis and the efficiency of these two methods is demonstrated in numerical tests.

Chapter 6

Online Learning Sensing Matrix and Sparsifying Dictionary Simultaneously for Compressive Sensing

In this chapter, we describe our work on learning sensing matrix and sparsifying dictionary simultaneously for compressive sensing. This chapter is based on the following published paper:

- **Tao Hong** and Zhihui Zhu, *Online Learning Sensing Matrix and Sparsifying Dictionary Simultaneously for Compressive Sensing,* Signal Processing, vol. 153, pp. 188-196, Dec. 2018.

6.1 Introduction

Sparse representation (Sparseland) has led to numerous successful applications, e.g., image processing, machine learning, pattern recognition, and compressive sensing (CS) [2, 99, 3, 100, 17, 16] etc. Sparseland assumes that a signal $\boldsymbol{x} \in \mathbb{R}^N$ can be represented as a linear combination of a few columns of a matrix $\boldsymbol{\Psi} \in \mathbb{R}^{N\times L}$ (also called dictionary):

$$\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{\theta} + \boldsymbol{e}, \tag{6.1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^L$ is the representation coefficient of $\boldsymbol{x}$ over $\boldsymbol{\Psi}$ which only has a few non-zeros and $\boldsymbol{e} \neq \boldsymbol{0} \in \mathbb{R}^N$ refers to the sparse representation error (SRE). The signal $\boldsymbol{x}$ is called K-sparse in $\boldsymbol{\Psi}$ if $\|\boldsymbol{\theta}\|_0 \leq K$ where $\|\boldsymbol{\theta}\|_0$ is used to count the number of non-zeros in $\boldsymbol{\theta}$.

The choice of dictionary $\boldsymbol{\Psi}$ depends on specific applications and can be a prescribed one, e.g., discrete cosine transform (DCT), wavelets transform or a multiband modulated discrete prolate spheroidal sequences dictionary [101] etc. Moreover, one can also learn a dictionary $\boldsymbol{\Psi}$ (called dictionary learning), such that a set of P training signals $\{\boldsymbol{x}_k, k = 1, 2, \cdots, P\}$ is sparsely represented by optimizing $\boldsymbol{\Psi}$. There exist many efficient algorithms to learn such a dictionary [3], e.g., the method of optimal directions (MOD) [100] and the K-singular value decomposition (KSVD) algorithm [17].

CS is an emerging framework that enables us to exactly recover the signal $\boldsymbol{x}$, which is sparse or sparsely represented by a dictionary $\boldsymbol{\Psi}$, from a number of linear measurements that is considerably lower than the size of samples required by the Shannon-Nyquist theorem [16]. Using a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ called

sensing matrix (a.k.a projection matrix), we obtain the linear measurements $\boldsymbol{y}$ through

$$\boldsymbol{y} = \boldsymbol{\Phi x} = \boldsymbol{\Phi \Psi \theta} + \boldsymbol{\Phi e}, \tag{6.2}$$

where $M \ll N$. Many efforts have been devoted to optimize the sensing matrix with a prescribed dictionary resulting in a CS system that outperforms the standard one (random matrix) in various cases [102, 103, 104, 105, 106, 107, 108].

Recently, researchers have found that optimizing the sensing matrix and dictionary simultaneously for CS yields higher signal reconstruction accuracy than the classical approach of optimizing the sensing matrix for a prescribed dictionary [107, 108]. The main idea underlying [107, 108] is to consider the influence of SRE in optimizing the sensing matrix and learning the dictionary (see Section 6.3 for the formal problem). In contrast to [107], closed-form solutions for updating the sensing matrix and the dictionary are derived in [108], yielding better performance in terms of signal recovery accuracy. However the method in [108] requires many singular value decompositions (SVDs) hindering the efficiency.

Although the methods proposed in [107, 108] for jointly optimizing $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ work well for a small-scale training dataset (e.g., $N = 64$ and $P = 10^4$), they become inefficient (and even impractical) if the dimension of the dictionary is high or the size of training dataset is very large (e.g., $P > 10^6$) or for the case involving dynamic data like video streams. It is easy to see that the methods in [107, 108] require heavy memory and computations to address such a large scale problem because they have to sweep all of the training data in each dictionary updating procedure. Inspired by [109, 110], an *online* algorithm with less complexity and memory is introduced to address the same learning problem shown in [107, 108] but on a large dataset. Note that, in this chapter, large or large-scale dataset means a dataset that contains a large amount of training data, i.e., P is very large. We use a toy example to briefly explain the benefit of training on a large-scale dataset. Assume that the dimension of the dictionary is 64×100 and the number of non-zeros in the sparse vector $\boldsymbol{\theta}$ is 4. Then the number of subspaces in this dictionary is $\binom{100}{4} \approx 3.9 \times 10^6$. Thus, we see that such a dictionary provides a rich number of subspaces which motivates us to train the dictionary on a large-scale dataset to explore the dictionary to represent the signals of interest better. One can still imagine that along with the increase of the dimension of the dictionary, the number of subspaces will become much richer and we can expect such a dictionary to yield many interesting properties. Indeed, the benefit of learning a dictionary on a large dataset or a high dimension (without training the sensing matrix) has been experimentally demonstrated in [111, 112, 113]. Moreover, the simulations shown in this chapter also indicate the advantage of learning the CS system (both the dictionary and the sensing matrix) on a large-scale dataset.

Note that, at each step, the sensing matrix is either updated with an iterative algorithm in [107] or an alternating-minimization method in [108], both requiring many SVD computations. To avoid this, we present an efficient method to optimize the sensing matrix which is robust with respect to the SRE. The proposed method is inspired by the recent results in [104, 114, 106] for robust sensing matrices, but it differs from these works in which there is no need to tune the trade-off parameter and hence it is more suitable for online learning and dynamic data. The experiments on natural images demonstrate that jointly optimizing the Sensing Matrix and Sparsifying Dictionary (SMSD) on a large dataset has much better performance in terms of signal recovery accuracy than that of [107, 108]. Notice that in this chapter we are interested in designing a CS system for natural images.

The rest of this chapter is organized as follows. In Section 6.2, a novel model is proposed to design

the sensing matrix to reduce the mutual coherence (cf. (7.1)) between each two columns in $\boldsymbol{\Phi\Psi}$ and to overcome the influence of SRE. Moreover, a closed-form solution is derived to obtain the optimized sensing matrix which is parameter free and then it is suitable for the following joint learning SMSD method. A joint optimization algorithm for learning SMSD on a large dataset is suggested in Section 6.3. To learn the sparsifying dictionary on a large dataset efficiently, an online method is introduced. Note that the online algorithm is applied to update the dictionary because the training data is only involved in the dictionary learning procedure (6.13). For brevity, we still call the whole joint algorithm online SMSD. Numerical experiments on natural images are carried out in Section 6.4 to demonstrate the effectiveness of the proposed algorithm and the advantage of training on a large dataset compared with other methods. Conclusion and future work are given in Section 6.5.

Notation: MATLAB notation is adopted in this chapter. For a vector, $\boldsymbol{v}(m)$ denotes the mth component of $\boldsymbol{v}$. For a matrix, $\boldsymbol{Q}(m,n)$ means the (m,n)th element of matrix $\boldsymbol{Q}$, while $\boldsymbol{Q}(m,:)$ and $\boldsymbol{Q}(:,m)$ indicate the mth row and column vector of $\boldsymbol{Q}$, respectively.

6.2 An Efficient Method for Robust Sensing Matrix Design

Following the terminology used in [104, 106], a robust sensing matrix refers to a sensing matrix which yields robust performance for signals with $\boldsymbol{e} \neq 0$. We note that one of the major purposes in optimizing the sensing matrix is to reduce the coherence between all pairs of columns of the equivalent dictionary $\boldsymbol{\Phi\Psi}$ leading to [102, 103, 115, 116] which demonstrate that the optimized sensing matrix yields much better performance than the one with a random sensing matrix for the signals with $\boldsymbol{e} = \boldsymbol{0}$. However, it was recently realized [104, 114, 106] that such a sensing matrix is not robust with respect to $\boldsymbol{e} \neq 0$ and thus the corresponding CS system results in poor performance. For natural images, the SRE always exists even when we represent the images with a well designed dictionary. Moreover, the average mutual coherence (i.e., the coherence measured with ℓ_2 norm instead of the infinity norm) rather than the exact mutual coherence is suggested in [104, 114] for designing an optimized robust sensing matrix. Specifically, a robust sensing matrix is attained by solving [104, 114]:

$$\min_{\boldsymbol{\Phi}} \ \|\boldsymbol{I}_L - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\|_F^2 + \lambda\|\boldsymbol{\Phi E}\|_F^2, \tag{6.3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\boldsymbol{I}_L$ represents an identity matrix with dimension L, $\boldsymbol{E}(:,k) = \boldsymbol{e}_k$, $k = 1, \cdots, P$, and λ is a trade-off parameter that balances the coherence of the equivalent dictionary and the robustness of the sensing matrix with respect to the SRE. In [106], it is suggested to replace the penalty $\|\boldsymbol{\Phi E}\|_F^2$ by $\|\boldsymbol{\Phi}\|_F^2$ (which is independent of the training data) since $\|\boldsymbol{\Phi}\|_F^2$ has the same effectiveness as $\|\boldsymbol{\Phi E}\|_F^2$ when the SRE is modelled as Gaussian noise and $P \to \infty$. Thus, the robust sensing matrix is optimized following [106] by

$$\min_{\boldsymbol{\Phi}} \ f(\boldsymbol{\Phi}) = \|\boldsymbol{I}_L - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\|_F^2 + \lambda\|\boldsymbol{\Phi}\|_F^2. \tag{6.4}$$

Numerical experiments with natural images show that the optimized sensing matrix obtained by solving (6.4) with a well-chosen λ yields state-of-the-art performance in CS-based image compression [106]. However, we note that it is nontrivial to choose an optimal λ for (6.4) since the two terms $\|\boldsymbol{I}_L - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\|_F^2$ and $\|\boldsymbol{\Phi}\|_F^2$ have different physical meanings: the former represents the average

mutual coherence of the equivalent dictionary $\boldsymbol{\Phi\Psi}$, while the latter is the energy of the sensing matrix $\boldsymbol{\Phi}$. For off-line applications when the dictionary is fixed, it is suggested to choose a λ by a grid search method [104, 114, 106]. However, this strategy becomes very inefficient for online applications when the dictionary $\boldsymbol{\Psi}$ is evolving, which is the case we consider in this chapter. To avoid tuning the parameter λ, we suggest designing the robust sensing matrix with the following two steps: (i) find a set of solutions which minimize $\|\boldsymbol{I}_L - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\|_F^2$ (i.e., solve (6.4) without the term $\|\boldsymbol{\Phi}\|_F^2$); (ii) then choose a $\boldsymbol{\Phi}$ that has the smallest energy. Thus, we consider the following optimization problem to design the sensing matrix:

$$\begin{aligned} &\min_{\boldsymbol{\Phi}\in\mathcal{S}} \quad \|\boldsymbol{\Phi}\|_F^2 \\ &\mathcal{S} = \quad \arg\min_{\boldsymbol{\Phi}\in\mathbb{R}^{M\times N}} \; g(\boldsymbol{\Phi}) = \|\boldsymbol{I}_L - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\|_F^2. \end{aligned} \tag{6.5}$$

Let $\mathcal{U}_{M,\bar{N}} := \left\{\boldsymbol{U}_{M,\bar{N}} : \boldsymbol{U}_{M,\bar{N}}^{\mathcal{T}}\boldsymbol{U}_{M,\bar{N}} = \boldsymbol{I}_{\bar{N}}\right\}$ denote the set of $M\times\bar{N}$ orthonormal matrices for $\bar{N}\le M$. When $\bar{N} = M$, to simplify the notation, we use $\mathcal{U}_M$ to denote the set of $M\times M$ orthonormal matrices. The following result establishes a set of closed-form solutions to (6.5):

Theorem 6.1. *Let $\boldsymbol{\Psi} = \boldsymbol{U}_{\boldsymbol{\Psi}}\boldsymbol{\Lambda}\boldsymbol{V}_{\boldsymbol{\Psi}}^{\mathcal{T}}$ be an SVD of $\boldsymbol{\Psi}$, where Rank$(\boldsymbol{\Psi}) = \bar{N}\le N$, $\boldsymbol{\Lambda} = diag(\lambda_1,\lambda_2,\cdots,\lambda_{\bar{N}}) > 0$ with $\lambda_1\ge\lambda_2\ge\cdots\ge\lambda_{\bar{N}}$, and $\boldsymbol{U}_{\boldsymbol{\Psi}}$ and $\boldsymbol{V}_{\boldsymbol{\Psi}}$ are $N\times\bar{N}$ and $L\times\bar{N}$ orthonormal matrices, respectively. When $\bar{N}\ge M$, a set of optimal solutions for* (6.5) *is given by*

$$\mathcal{W}_1 := \left\{\boldsymbol{\Phi} : \boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{U}_M & \boldsymbol{0}\end{bmatrix}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}, \boldsymbol{U}_M\in\mathcal{U}_M\right\}. \tag{6.6}$$

On the other hand, when $\bar{N} < M$, a set of optimal solutions for (6.5) *is given by*

$$\mathcal{W}_2 := \left\{\boldsymbol{\Phi} : \boldsymbol{\Phi} = \boldsymbol{U}_{M,\bar{N}}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}, \boldsymbol{U}_{M,\bar{N}}\in\mathcal{U}_{M\times\bar{N}}\right\}. \tag{6.7}$$

Proof. We first rewrite $g(\boldsymbol{\Phi})$ as

$$\begin{aligned} g(\boldsymbol{\Phi}) &= \|\boldsymbol{I}_L - \boldsymbol{V}_{\boldsymbol{\Psi}}\boldsymbol{\Lambda}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{U}_{\boldsymbol{\Psi}}\boldsymbol{\Lambda}\boldsymbol{V}_{\boldsymbol{\Psi}}^{\mathcal{T}}\|_F^2 \\ &= \underbrace{\|\boldsymbol{I}_{\bar{N}} - \boldsymbol{\Lambda}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{U}_{\boldsymbol{\Psi}}\boldsymbol{\Lambda}\|_F^2}_{h(\boldsymbol{\Phi})} + L - \bar{N}. \end{aligned}$$

Thus, minimizing $g(\boldsymbol{\Phi})$ is equivalent to $\min_{\boldsymbol{\Phi}} h(\boldsymbol{\Phi})$. We proceed by considering the following two cases.

Case I: $\bar{N}\ge M$. Noting that $\text{rank}(\boldsymbol{\Lambda}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{U}_{\boldsymbol{\Psi}}\boldsymbol{\Lambda}) = M\le\bar{N}$ and using the Eckart-Young-Mirsky theorem [117], we have $h(\boldsymbol{\Phi})\ge\bar{N}-M$ which achieves its minimum when

$$\boldsymbol{\Lambda}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{U}_{\boldsymbol{\Psi}}\boldsymbol{\Lambda} = \boldsymbol{U}_{\bar{N},M}\boldsymbol{U}_{\bar{N},M}^{\mathcal{T}}, \tag{6.8}$$

where $\boldsymbol{U}_{\bar{N},M}\in\mathcal{U}_{\bar{N},M}$. Through (6.8), we get the set of $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi}\in\mathcal{S} = \left\{\boldsymbol{U}_{\bar{N},M}^{\mathcal{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}} : \boldsymbol{U}_{\bar{N},M}\in\in\mathcal{U}_{\bar{N},M}\right\}. \tag{6.9}$$

Now we identify a $\boldsymbol{\Phi}$ which has the smallest $\|\boldsymbol{\Phi}\|_F^2$. Note that

$$\|\boldsymbol{\Phi}\|_F^2 = \text{Tr}\left(\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\right) = \text{Tr}\left(\boldsymbol{U}_{\bar{N},M}\boldsymbol{U}_{\bar{N},M}^{\mathcal{T}}\boldsymbol{\Lambda}^{-2}\right) = \sum_{i=1}^{\bar{N}}\frac{\alpha_i}{\lambda_i^2},$$

where $\alpha_i \in [0,1]$ is the ith diagonal element of $\boldsymbol{U}_{\bar{N},M}\boldsymbol{U}_{\bar{N},M}^{\mathcal{T}}$ and $\sum_{i=1}^{\bar{N}} \alpha_i = M$ because $\boldsymbol{U}_{\bar{N},M}$ is an $\bar{N} \times M$ orthonormal matrix. Therefore, we have $\sum_{i=1}^{\bar{N}} \frac{\alpha_i}{\lambda_i^2} \geq \sum_{i=1}^{M} \frac{1}{\lambda_i^2}$ and the minimal value is achieved when $\alpha_1 = \cdots = \alpha_M = 1$ and $\alpha_{M+1} = \cdots = \alpha_{\bar{N}} = 0$ which implies $\boldsymbol{U}_{\bar{N},M} = \begin{bmatrix} \boldsymbol{U}_M \\ \boldsymbol{0} \end{bmatrix}$ with $\boldsymbol{U}_M \in \mathcal{U}_M$.

Case II: $\bar{N} < M$. We first note that $h(\boldsymbol{\Phi}) \geq 0$ and it achieves its minimum when

$$\boldsymbol{\Phi} \in \mathcal{S} = \left\{ \boldsymbol{U}_{M,\bar{N}}^{\mathcal{T}} \boldsymbol{\Lambda}^{-1} \boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}} : \boldsymbol{U}_{M,\bar{N}}^{\mathcal{T}} \boldsymbol{U}_{\bar{N},M} = \boldsymbol{I}_{\bar{N}} \right\}.$$

where $\boldsymbol{U}_{M,\bar{N}} \in \mathcal{U}_{M,\bar{N}}$. For such $\boldsymbol{\Phi}$, we have

$$\|\boldsymbol{\Phi}\|_F^2 = \mathrm{Tr}\left(\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\right) = \mathrm{Tr}\left(\boldsymbol{\Lambda}^{-2}\right) = \sum_{i=1}^{\bar{N}} \frac{1}{\lambda_i^2},$$

which implies that all $\boldsymbol{\Phi}$ have the same energy. □

We remark that [103, Theorem 2] also gives a set of optimal solutions to minimize $g(\boldsymbol{\Phi})$ for $\mathrm{Rank}(\boldsymbol{\Psi}) = \bar{N} \geq M$, i.e., $\mathcal{S} = \left\{ \begin{bmatrix} \boldsymbol{U}_M & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda}^{-1} \boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}, \boldsymbol{U}_M \in \mathcal{U}_M \right\}$ coinciding with (6.9) because $\begin{bmatrix} \boldsymbol{U}_M^{\mathcal{T}} \\ \boldsymbol{0} \end{bmatrix} \in \mathcal{U}_{\bar{N},M}$. When $\mathrm{Rank}(\boldsymbol{\Psi}) = \bar{N} \geq M$ (which is true for most of applications), (6.6) gives a set of optimal solutions for (6.5) and implies that there may exist some degrees of freedom in choosing $\boldsymbol{U}_M$. However, the following result expresses an equivalent performance of the sensing matrices with different $\boldsymbol{U}_M$.

Lemma 6.2.1. *Compressive sensing systems with the same dictionary $\boldsymbol{\Psi}$ and different $\boldsymbol{\Phi} \in \mathcal{W}_1$ (which is defined in* (6.6)*) have the same performance.*

Proof. Suppose we have two compressive sensing systems with the same dictionary $\boldsymbol{\Psi}$ and the sensing matrices $\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{U}_M & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda}^{-1} \boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}$ and $\overline{\boldsymbol{\Phi}} = \begin{bmatrix} \overline{\boldsymbol{U}}_M & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda}^{-1} \boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}$, respectively, where $\boldsymbol{U}_M, \overline{\boldsymbol{U}}_M \in \mathcal{U}_M$. For any $\boldsymbol{x} \in \mathbb{R}^N$, the two CS systems obtain the measurements $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}$, $\overline{\boldsymbol{y}} = \overline{\boldsymbol{\Phi}}\boldsymbol{x}$, and respectively attempt to recover $\boldsymbol{x}$ via

$$\min_{\theta} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}\|^2, \ \text{ s.t. } \ \|\boldsymbol{\theta}\|_0 \leq K,$$

and

$$\min_{\theta} \|\overline{\boldsymbol{y}} - \overline{\boldsymbol{\Phi}}\boldsymbol{\Psi}\boldsymbol{\theta}\|^2, \ \text{ s.t. } \ \|\boldsymbol{\theta}\|_0 \leq K.$$

The proof is completed by noting that the above two equations have the same solution since

$$\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}\|^2 = \|\overline{\boldsymbol{U}}_M \boldsymbol{U}_M^{\mathcal{T}} (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta})\|^2 = \|\overline{\boldsymbol{y}} - \overline{\boldsymbol{\Phi}}\boldsymbol{\Psi}\boldsymbol{\theta}\|^2.$$

Lemma 6.2.1 implies that one can choose $\boldsymbol{U}_M$ as an identity matrix which has the same performance as other $\boldsymbol{\Phi} \in \mathcal{W}_1$ and then the solution for (6.5) becomes

$$\boldsymbol{\Phi} \triangleq \phi(\boldsymbol{\Psi}) = \boldsymbol{\Lambda}_M^{-1} \boldsymbol{U}_{\boldsymbol{\Psi}}(:, 1:M)^{\mathcal{T}}, \tag{6.10}$$

where $\boldsymbol{\Lambda}_M = \boldsymbol{\Lambda}(1:M, 1:M)$. Notice that $\boldsymbol{\Lambda}_M$ is a diagonal matrix so the calculation of its inversion is cheap. In effect, one SVD of the dictionary $\boldsymbol{\Psi}$ dominates the complexity in the sensing matrix updating procedure. However, we only need the M largest singular values and the corresponding left orthogonal matrices that the computation can be reduced further by using the power method. Compared with the

methods shown in [107, 108] which need to perform eigenvalue decompositions or SVD many times, our method saves significant computation.

We end this section by comparing (6.10) with gradient descent for solving (6.4) in terms of the computational complexity, though our main purpose of using (6.10) is to avoid tuning the parameter λ in (6.4). We note that although (6.4) is nonconvex, recent work on low-rank optimization [118] indicates that gradient descent can converge to the global solution for a set of low-rank matrix optimization. The convergence is also experimentally verified for (6.4) in [106], though there is no theoretical guarantee about the convergence rate (i.e., how fast it converges to the global solution). Note that the gradient of $f(\boldsymbol{\Phi})$ is

$$\nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}) = 2\lambda\boldsymbol{\Phi} - 4\boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathcal{T}} + 4\boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathcal{T}}.$$

Suppose that $\boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathcal{T}}$ is precomputed, and then evaluating $\nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi})$ requires $O(MN^2)$ computations. Thus, using gradient descent has at least $O(MN^2)$ computational complexity but we do not know how fast of the convergence. From (6.9), our closed-form solution only needs to compute the largest M eigenvalues and corresponding eigenvectors of $\boldsymbol{\Psi}\boldsymbol{\Psi}^T$ requiring $O(MN^2)$ computational complexity in total. We also note that in the dictionary updating procedure (see (6.18) in Section 6.3), it is required to compute $(\boldsymbol{I}_N + \frac{1}{\gamma}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi})^{-1}$, which can be directly obtained through (6.10). If we use the gradient descent method to update $\boldsymbol{\Phi}$, we still need to compute such an inversion for updating the dictionary but this can be saved if we use (6.10).

6.3 Online Learning SMSD Simultaneously

We consider the problem of jointly optimizing the sensing matrix and sparsifying dictionary (SMSD) on a very large training dataset in this section. To reduce the complexity of learning a dictionary on a large training data, we propose an online algorithm with the consideration of the projected sparse representation error (SRE) $\|\boldsymbol{\Phi e}\|_2$. Moreover, we develop an alternating-minimization based approach to consider the corresponding joint optimization problem.

6.3.1 Online Joint SMSD Optimization

Given a set of P training signals $\boldsymbol{X}(:,k) = \boldsymbol{x}_k$, $k = 1, 2, \cdots, P$, our purpose is to jointly design the SMSD. Classical dictionary learning attempts to minimize the SRE:

$$\min_{\boldsymbol{\Psi}\in\mathcal{C},\boldsymbol{\Theta}} \|\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta}\|_F^2, \ \text{ s.t. } \ \|\boldsymbol{\theta}_k\|_0 \leq K, \ \forall k, \tag{6.11}$$

where $\boldsymbol{\Theta}(:,k) = \boldsymbol{\theta}_k$, $\forall k$ contains the sparse coefficient vectors and $\mathcal{C}$ is a constraint set to avoid trivial solutions. Note that in CS, we obtain the linear measurements $\boldsymbol{y}$ as in (6.2) and then recover the signal from $\boldsymbol{y}$ by first recovering the sparse coefficients $\boldsymbol{\theta}$ and then obtain $\boldsymbol{x}$ through $\boldsymbol{\Psi\theta}$. Therefore, a smaller $\boldsymbol{\Phi e}$ is also preferred. This implies that besides reducing $\|\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta}\|_F^2$, given a sensing matrix $\boldsymbol{\Phi}$, the dictionary is also expected to reduce $\|\boldsymbol{\Phi}(\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta})\|_F^2$ [107, 108, 119, 120]. So the sensing matrix and the sparsifying dictionary (SMSD) are jointly optimized through [107, 108]

$$\begin{aligned} \min_{\boldsymbol{\Psi}\in\mathcal{C},\boldsymbol{\Theta},\boldsymbol{\Phi}} \quad & \gamma\|\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta}\|_F^2 + \|\boldsymbol{\Phi}\boldsymbol{X} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Theta}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{\Phi} = \phi(\boldsymbol{\Psi}), \|\boldsymbol{\theta}_k\|_0 \leq K, \ \forall k, \end{aligned} \tag{6.12}$$

where $\phi(\boldsymbol{\Psi})$ is given in (6.10) and $\gamma \in [0,1]$ is a trade-off parameter to balance these two terms. The value of γ can be determined through grid search to receive the highest signal recovery accuracy on the testing dataset.

Remark 6.1.

- We first note that the projected SRE $\|\boldsymbol{\Phi}(\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta})\|_F^2$ also contains the sensing matrix $\boldsymbol{\Phi}$ and hence this term should be considered in designing the sensing matrix. As discussed in Section 6.2, the sensing matrix $\phi(\boldsymbol{\Psi})$ given in (6.10) already considers the projected SRE implying the advantage of our proposed method for designing the sensing matrix compared with the ones used in [107, 108] where the influence of $\|\boldsymbol{\Phi}(\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta})\|_F^2$ is not considered.
- Compared with a separate approach that (usually) first learns the dictionary by (6.11) and then designs the sensing matrix with the learned dictionary, jointly learning the SMSD through (6.12) is expected to yield a better CS system, as the projected SRE is also minimized sequentially. We refer the reader to [107, 108] for more discussions about the advantages of this joint approach.

Similarly to [107, 108], we use the alternating-minimization based method to solve (6.12). The main idea is to alternatively update the sensing matrix (when the dictionary and sparse coefficients are fixed) by (6.10) and then update the dictionary and the sparse coefficients (when the sensing matrix is fixed) by solving

$$\begin{aligned} \min_{\boldsymbol{\Psi} \in C, \boldsymbol{\Theta}} \quad & \sigma(\boldsymbol{\Psi}, \boldsymbol{\Theta}) \triangleq \gamma \|\boldsymbol{X} - \boldsymbol{\Psi}\boldsymbol{\Theta}\|_F^2 + \|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Theta}\|_F^2, \\ \text{s.t.} \quad & \|\boldsymbol{\theta}_k\|_0 \leq K, \ \forall k, \end{aligned} \tag{6.13}$$

where $\boldsymbol{Y} = \boldsymbol{\Phi}\boldsymbol{X}$. Since we are interested in learning a dictionary on a large training dataset, we suggest an online algorithm in the next subsection to fit such a large dataset. The detailed steps for solving (6.12) are summarized in Algorithm 6.1. Compared with the methods in [107, 108], Algorithm 6.1 is designed to work on a large training dataset that it uses (6.10) for optimizing the sensing matrix with an online method (in the next subsection), which is independent to the size of the training dataset, for learning the dictionary. We note that the methods described in [107, 108] for solving (6.13) need to sweep all of the training data at each iteration requiring high computation and large memory if the training dataset is large. Moreover, the methods proposed in [107, 108] for updating the sensing matrix are iterative algorithms which require many SVDs. The simulation results in the next section demonstrate the effectiveness of Algorithm 6.1.

Algorithm 6.1 Online Joint Optimization of SMSD

Input: Initial dictionary $\boldsymbol{\Psi}_0$ and maximal iterations Max_Iter_{sendic}.
Output: The sensing matrix $\boldsymbol{\Phi}$ and the sparsifying dictionary $\boldsymbol{\Psi}$.

1: **for** Iter $= 1$ **to** Max_Iter_{sendic} **do**
2: Update the sensing matrix $\boldsymbol{\Phi}_{\text{Iter}}$ with fixed $\boldsymbol{\Psi} = \boldsymbol{\Psi}_{\text{Iter}-1}$ through (6.10).
3: Update the dictionary through (6.13) with Algorithm 6.2 for fixed $\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\text{Iter}}$.
4: **end for**

6.3.2 Online Dictionary Learning with Projected SRE

In this subsection, we present an online algorithm (Algorithm 6.2) to address (6.13) when the dataset is large. The online method for (6.13) comprises two main stages: (i) the sparse coefficient vectors in $\boldsymbol{\Theta}$

are computed with a fixed $\boldsymbol{\Psi}$; (ii) the sparsifying dictionary $\boldsymbol{\Psi}$ is updated with a fixed $\boldsymbol{\Theta}$. Note that only random sample of the training data is used at each iteration in our case which is different from the methods shown in [107, 108]. The detailed steps of the online algorithm are summarized in Algorithm 6.2.

Algorithm 6.2 Online Dictionary Learning with Projected SRE

Input: Training data $\boldsymbol{X} \in \mathbb{R}^{N\times P}$, trade-off parameter γ, sensing matrix $\boldsymbol{\Phi}$ and initial dictionary $\boldsymbol{\Psi}_0$, batch size $\eta \geq 1$, sparsity level K, forgetting parameter ρ, and maximal iterations Max_Iter_{dic}.

Output: Dictionary $\boldsymbol{\Psi}$.

$\boldsymbol{A}_0 \leftarrow \boldsymbol{0}, \boldsymbol{B}_0 \leftarrow \boldsymbol{0}, k \leftarrow 1.$

for $t = 1$ **to** Max_Iter_{dic} **do**

  if $k+\eta \leq P$ **then**

    $\boldsymbol{X}_t \leftarrow \boldsymbol{X}(:, k:k+\eta-1), \boldsymbol{Y}_t \leftarrow \boldsymbol{\Phi}\boldsymbol{X}_t.$

    $k \leftarrow k+\eta.$

  else

    Shuffle $\boldsymbol{X}$, $k \leftarrow 1$.

    $\boldsymbol{X}_t \leftarrow \boldsymbol{X}(:, k:k+\eta-1), \boldsymbol{Y}_t \leftarrow \boldsymbol{\Phi}\boldsymbol{X}_t.$

    $k \leftarrow k+\eta.$

  end if

  Sparse coding

$$\begin{aligned} \boldsymbol{\Theta}_t \leftarrow \quad & \arg\min_{\tilde{\boldsymbol{\Theta}}_t} \left\| \begin{bmatrix} \sqrt{\gamma}\boldsymbol{X}_t \\ \boldsymbol{Y}_t \end{bmatrix} - \begin{bmatrix} \sqrt{\gamma}\,\boldsymbol{\Psi}_{t-1} \\ \boldsymbol{\Phi}\boldsymbol{\Psi}_{t-1} \end{bmatrix} \tilde{\boldsymbol{\Theta}}_t \right\|_F^2, \\ & \text{s.t. } \|\tilde{\boldsymbol{\Theta}}_t(:,k)\|_0 \leq K, \forall k. \end{aligned} \tag{6.14}$$

  $\boldsymbol{A}_t \leftarrow (1-\frac{1}{t})^{\rho}\boldsymbol{A}_{t-1} + \frac{1}{\eta}\boldsymbol{\Theta}_t\boldsymbol{\Theta}_t^{\mathcal{T}}.$

  $\boldsymbol{B}_t \leftarrow (1-\frac{1}{t})^{\rho}\boldsymbol{B}_{t-1} + \frac{1}{\eta}\boldsymbol{X}_t\boldsymbol{\Theta}_t^{\mathcal{T}}.$

  Compute $\boldsymbol{\Psi}_t$ using Algorithm 6.3 with $\boldsymbol{\Psi}_{t-1}$ as the initial value, so that

$$\boldsymbol{\Psi}_t = \arg\min_{\boldsymbol{\Psi}\in\mathcal{C}} \hat{\sigma}_t(\boldsymbol{\Psi}).$$

end for

For simplicity, similarly to [107, 108], Algorithm 6.2 utilizes Orthogonal Matching Pursuit (OMP) for the sparse coding problem (6.14) to update $\boldsymbol{\Theta}$. At tth iteration of the dictionary updating procedure, we suggest minimizing a surrogate function $\sigma_t(\boldsymbol{\Psi})$ instead of considering (6.13) directly, i.e.,

$$\min_{\boldsymbol{\Psi}} \sigma_t(\boldsymbol{\Psi}) \triangleq \frac{1}{2}\sum_{i=1}^{t} \left(\gamma\|\boldsymbol{X}_i - \boldsymbol{\Psi}\boldsymbol{\Theta}_i\|_F^2 + \|\boldsymbol{Y}_i - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Theta}_i\|_F^2\right), \tag{6.15}$$

which is equivalent to

$$\min_{\boldsymbol{\Psi}} \hat{\sigma}_t(\boldsymbol{\Psi}) \triangleq \frac{1}{2}\text{Tr}\left(\boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Omega}\boldsymbol{\Psi}\boldsymbol{A}_t\right) - \text{Tr}\left(\boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Omega}\boldsymbol{B}_t\right), \tag{6.16}$$

where $\boldsymbol{A}_t = \sum_{i=1}^{t}\boldsymbol{\Theta}_i\boldsymbol{\Theta}_i^{\mathcal{T}}$, $\boldsymbol{B}_t = \sum_{i=1}^{t}\boldsymbol{X}_i\boldsymbol{\Theta}_i^{\mathcal{T}}$, $\boldsymbol{\Omega} = \boldsymbol{I}_N + \frac{1}{\gamma}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}$ and $\text{Tr}(\cdot)$ denotes the trace operator. To solve (6.16), we use the block-coordinate descent algorithm that updates the dictionary column by column. By choosing block-coordinate descent, tuning learning rate, which is required by stochsatic gradient descent, and calculating the inversion of any matrices, which is required by Newton-type methods for (6.16), are avoided [121]. The gradient of (6.16) with respect to the jth column of $\boldsymbol{\Psi}$ is

$$\frac{\partial\hat{\sigma}_t(\boldsymbol{\Psi})}{\partial\boldsymbol{\psi}_j} = \boldsymbol{\Psi}\boldsymbol{a}_j - \boldsymbol{b}_j + \frac{1}{\gamma}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{a}_j - \frac{1}{\gamma}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}\boldsymbol{b}_j, \tag{6.17}$$

where $\boldsymbol{\psi}_j$, $\boldsymbol{a}_j$ and $\boldsymbol{b}_j$ are the jth column of the matrices $\boldsymbol{\Psi}$, $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$, respectively. The optimal $\boldsymbol{\psi}_j$ is obtained by enforcing (6.17) to be zero that $\boldsymbol{\psi}_j$ should be updated as (6.19) while keeping the others fixed. Following each column of the dictionary is normalized to have a unit ℓ_2 norm to avoid the trivial solution. The detailed steps for solving (6.16) are summarized in Algorithm 6.3.

The matrices $\boldsymbol{\Xi}_1$ and $\boldsymbol{\Xi}_2$ in Algorithm 6.3 are equivalent to $\boldsymbol{\Omega}^{-1}$, and $\boldsymbol{\Omega}^{-1}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi}$, respectively. Because of the special structure of $\boldsymbol{\Phi}$ shown in (6.10), the matrices $\boldsymbol{\Xi}_1$ and $\boldsymbol{\Xi}_2$ can be evaluated simply by:

$$\begin{aligned}\boldsymbol{\Xi}_1 = &\ \boldsymbol{U}_{\boldsymbol{\Psi}}\begin{bmatrix}\left(\gamma^{-1}\boldsymbol{\Lambda}_M^{-2}+\boldsymbol{I}_M\right)^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}_{N-M}\end{bmatrix}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}},\\ \boldsymbol{\Xi}_2 = &\ \boldsymbol{U}_{\boldsymbol{\Psi}}\begin{bmatrix}\left(\gamma^{-1}\boldsymbol{I}_M+\boldsymbol{\Lambda}_M^{2}\right)^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\boldsymbol{U}_{\boldsymbol{\Psi}}^{\mathcal{T}}.\end{aligned} \tag{6.18}$$

Although we have to compute the inversion of matrices in (6.18), the computational burden is essentially low because the related matrices are diagonal. Following, we present several remarks which are useful in practice to improve the performance of Algorithm 6.2.

Remark 6.2.

- When the training dataset has finite size (though it maybe very large), we suggest simulating the random sampling of the data by cycling over a randomly permuted dataset, i.e., Lines 4 and 8 in Algorithm 6.2.
- As introduced before, we sample one example from the training dataset instead of swapping all of the data during each iteration. A typical strategy which can be used to accelerate the algorithm is to sample a relatively large set of examples instead of only one example ($\eta > 1$). This corresponds to a classical heuristic strategy in the stochastic gradient descent method [122] called mini-batch which is also useful in our case. Another strategy to accelerate the algorithm is to add the history into $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ as we have already shown in the formulation of $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ through the accumulation of $\boldsymbol{\Theta}_t$ and $\boldsymbol{X}_t$. By assuming the dictionary will approach a stationary point after a sufficient number of iterations, which is compatible with our following convergence analysis, the latest $\boldsymbol{\Theta}_t$ is more important than the old one. Hence, a forgetting factor is added in $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ to de-emphasize the older information in $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$. We set the forgetting factor to be $(1-\frac{1}{t})^{\rho}$ with $\rho > 1$ in updating $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$. The detailed formulation can be found at Lines 12 and 13 in Algorithm 6.2.
- In practical situations, the dictionary learning technique will lead to a dictionary whose columns are never (or very seldom) used in sparse coding procedures, which happens typically with a poor initialization. If we encounter such a phenomenon, one training example is randomly sampled to replace such a column in this chapter.

6.3.3 Convergence Analysis

Although the logic in Algorithm 6.1 is relatively simple, it is nontrivial to prove the convergence of Algorithm 6.3 because of its stochastic nature, the non-convexity and two different objective functions ((6.5) and (6.13)). In what follows, we provide the convergence analysis for each step in Algorithm 6.3. Notice that Algorithm 6.3 contains two parts: optimizing the sensing matrix and learning the sparsifying

Algorithm 6.3 Dictionary Update

Input: $\boldsymbol{A}_t = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_L]$, $\boldsymbol{B}_t = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_L]$, $\boldsymbol{\Xi}_1$, $\boldsymbol{\Xi}_2$, $\boldsymbol{\Psi}_{t-1} = [\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_L]$.

Output: Dictionary $\boldsymbol{\Psi}_t$.

1: **repeat**

2: **for** $j = 1$ **to** L **do**

3: Update the jth column of $\boldsymbol{\Psi}_t$:

$$\begin{aligned} \boldsymbol{u}_j &\leftarrow \boldsymbol{\Xi}_1 \left[\frac{\boldsymbol{b}_j - \boldsymbol{\Psi}_{t-1}\boldsymbol{a}_j}{\boldsymbol{A}_{t-1}(j,j)} + \boldsymbol{\psi}_j \right] + \\ &\quad \boldsymbol{\Xi}_2 \left[\frac{\boldsymbol{b}_j}{\boldsymbol{A}_{t-1}(j,j)\gamma} + \frac{1}{\gamma}\boldsymbol{\psi}_j - \frac{\boldsymbol{\Psi}_{t-1}\boldsymbol{a}_j}{\boldsymbol{A}_{t-1}(j,j)} \right]. \\ \boldsymbol{\Psi}_t(:,j) &\leftarrow \frac{\boldsymbol{u}_j}{\|\boldsymbol{u}_j\|_2}. \end{aligned} \tag{6.19}$$

4: **end for**

5: **until convergence**

dictionary to decrease the coherence of $\boldsymbol{\Phi\Psi}$ and to minimize the sparse representation error, respectively. Separately, we claim both of these two steps are convergent. Our simulation result shown in the next section also indicates the convergence of Algorithm 6.3. However, we leave the complete proof to future work. For the sensing matrix updating procedure, we attain the minimum with one step because of the closed-form solution. Following, we need to investigate whether the updating procedure for the dictionary is also convergent. In fact, this holds by the following assumptions and propositions which are taken from [109, 110].

Assumption 6.3.1.

1. *The data admits a distribution with compact support K.*
2. *The quadratic surrogate functions* $\hat{\sigma}_t$ *(defined in* (6.16)*) are strictly convex with lower-bounded Hessians*. Assume that the matrix $\boldsymbol{A}_t$ is positive definite. In fact, this hypothesis is verified experimentally after a few iterations of the algorithm with a proper initial dictionary. Specifically, all columns will be chosen at least once in the sparse coding procedure during the iterations. The Hessian matrix of $\hat{\sigma}_t$ is $\boldsymbol{\Omega} \otimes 2\boldsymbol{A}_t$ where $\otimes$ represents the kronecker product. Clearly, the eigenvalues of $\boldsymbol{\Omega} \otimes 2\boldsymbol{A}_t$ are the product of $\boldsymbol{\Omega}$ and $\boldsymbol{A}_t$'s eigenvalues. This indicates that the Hessian matrix of $\hat{\sigma}_t$ is positive definite because $\boldsymbol{\Omega}$ is a positive definite matrix which results in the fact that $\hat{\sigma}_t$ is a strictly convex function.
3. *A particular sufficient condition for the uniqueness of the sparse coding solution is satisfied.* Considering our sparse coding mission (6.14), we see that it exactly shares the same structure as in [109, 110]. So this assumption is also satisfied in our case.

Proposition 6.3.2. *[110, Propostion 2] Assume the assumptions (1) to (3) are satisfied, then we have*

1. $\hat{\sigma}_t(\boldsymbol{\Psi}_t)$ *convergences almost surely;*
2. $\sigma(\boldsymbol{\Psi}_t) - \hat{\sigma}_t(\boldsymbol{\Psi}_t)$ *converges almost surely to* 0*;*
3. $\sigma(\boldsymbol{\Psi}_t)$ *converges almost surely.*

Proposition 6.3.3. *[110, Propostion 3] Under assumptions (1) to (3), the distance between* $\boldsymbol{\Psi}_t$ *and the set of stationary points of the dictionary learning problem converges almost surely to 0 when t tends to infinity.*

Since Assumption 6.3.1 is held in our case, we conclude that the dictionary updating procedure in our case is also convergent. This verifies what we argue at the second term in Remark 6.2, that the dictionary will approach a stationary point after sufficient iterations. Though we have not rigorously proved the convergence of Algorithm 6.1, the convergence of the two parts in Algorithm 6.1 implies that the proposed algorithm at least is stable because both of these two steps (updating the sensing matrix and the dictionary) are convergent and decrease the value of the corresponding objective functions. The experiment in the following section also demonstrates this. We note that such a convergence analysis is not discussed in [107, 108], where the sensing matrix is updated with an iterative algorithm rather than as here with a closed-form solution. The only convergence analysis we are aware of jointly designing sensing matrix and dictionary is in [119], where both the sensing matrix and the dictionary are optimized in the same framework, but the training algorithm is not customized for large-scale applications. As for the convergence of Algorithm 6.1, we defer this to future work.

6.4 Numerical Experiments

In this section we compare our method (denoted by CS_{Alg3}) with the ones given in [107, 108] (denoted by CS_{S-DCS} and CS_{BL}, respectively) which share the same framework with ours but are based on the batch method (sweep the whole training data in each iteration) on natural images. The training and testing data are extracted from the LabelMe database [123]. All of the experiments are carried out on a laptop with Intel(R) i7-6500 CPU @ 2.5GHz and RAM 8G.

The signal reconstruction accuracy is evaluated in terms of Peak Signal to Noise Ratio (PSNR) given by [99]:

$$\rho_{psnr} \triangleq 10 \times \log 10 \left[\frac{(2^r - 1)^2}{\rho_{mse}} \right] dB$$

with $r = 8$ bits per pixel and $\rho_{mse} \triangleq \frac{1}{N \times P} \sum_{k=1}^{P} \|\tilde{\boldsymbol{x}}_k - \boldsymbol{x}_k\|_2^2$, where $\boldsymbol{x}_k$ is the original signal, $\tilde{\boldsymbol{x}}_k = \boldsymbol{\Psi}\tilde{\boldsymbol{\theta}}_k$ stands for the recovered signal and P is the number of patches in an image or testing data. The training and testing data are obtained through the following method.

Training data A set of 8×8 non-overlapping patches is obtained by randomly extracting 400 patches from each of the images in the whole LabelMe training dataset, with each patch of 8×8 arranged as a vector of size 64 resulting in totally $400 \times 2920 = 1.168 \times 10^6$ training samples for training.

Testing data The testing data is extracted from the LabelMe testing dataset. Here, we randomly extract 15 patches from 400 images and each sample is an 8×8 non-overlapping patch. Finally, we obtain 6000 test samples.

8×10^4 and 6×10^3 patches are randomly chosen from the 1.168×10^6 *Training data* for CS_{S-DCS} and CS_{BL}, respectively, because these two methods cannot stand too large training patches. To show the advantage of designing the SMSD on a large training dataset, the same 6×10^3 patches which are prepared for CS_{BL} are also used by CS_{S-DCS} and called $CS_{S-DCS} - small$. The parameters in these two methods are chosen as recommended in the corresponding papers that M, L and K are set to 20, 256 and 4 in CS_{Alg3}, respectively. The parameters γ, η, Max_Iter_{dic}, and Max_Iter_{sendic} are set to $\frac{1}{32}$, 128, 1000, and 10 in Algorithm 6.1. The initial sensing matrix and dictionary for [107, 108] are a random Gaussian matrix and the DCT dictionary, respectively. The initial sparsifying dictionary in the proposed algorithm

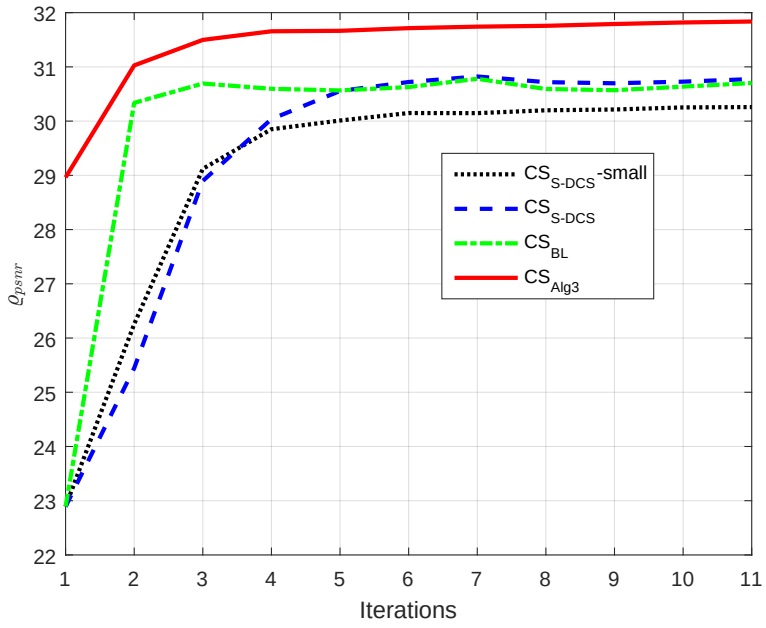


Figure 6.1: σ_{psnr} of the four different CS systems on testing data.

Table 6.1: CPU time (Seconds) of the four different CS dystems.

$CS_{S-DCS}-small$	CS_{S-DCS}	CS_{BL}	CS_{Alg3}
2.79×10^{1}	1.32×10^{3}	4.33×10^{4}	1.54×10^{2}

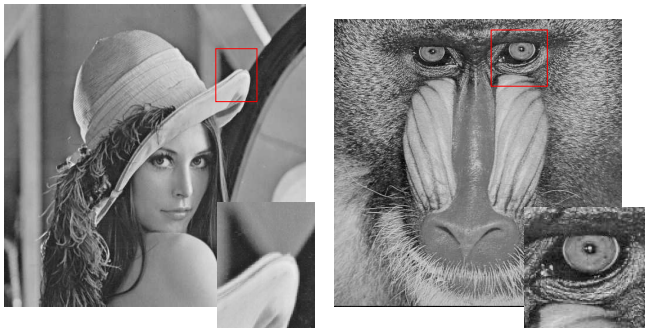
is randomly chosen from the training data and the corresponding sensing matrix is obtained through the method shown in Section 6.2. According to our experiments, using the initial value in such a case yields a slightly better performance compared with the initial settings suggested in [107, 108].

The signal recovery accuracy of the aforementioned methods on *testing data* and the corresponding CPU time in seconds are shown in Table 6.1 and Figure 6.1, respectively. Evidently, we see that CS_{S-DCS} yields better performance in terms of σ_{psnr} than $CS_{S-DCS}-small$ implying the benefit of working on a large training dataset. Compared with $CS_{S-DCS}-small$, CS_{BL} has a higher ρ_{psnr} which agrees with the observation shown in [108], but such an advantage disappears if we enlarge the size of the training dataset; see CS_{S-DCS}. Moreover, it is hard to extend CS_{BL} to large training datasets because it requires many SVDs which can also be observed in Table 6.1. From Figure 6.1, we observe that CS_{S-DCS} performs similarly to CS_{BL}, but requires less training time; see Table 6.1. Evidently, CS_{Alg3} yields the best performance in terms of ρ_{psnr}. Meantime, CS_{Alg3} needs a relatively less training time even the training dataset is largest than others implying that Algorithm 6.1 belongs to a good choice, which takes both efficiency and effectiveness into account simultaneously, for training the SMSD on a large training dataset.

Additionally, we investigate the performance of the four different CS systems mentioned in this chapter on ten natural images. The Structural Similarity Index (SSIM) [124] is also involved in comparing the recovered natural images obtained by other methods. In Table 6.2, we see that CS_{alg3} yields the highest PSNR and SSIM. Compared with $CS_{S-DCS}-small$, CS_{S-DCS} has a higher PSNR and SSIM in all of the ten test natural images that agrees with the argument in this chapter—one can benefit from using a large-scale training dataset. Similarly to test on the testing data, we observe that CS_{BL} works better than

Table 6.2: Performance evaluation of four different CS systems shown in this chapter. (Left: PSNR, Right: SSIM. The highest is marked in bold.)

	$CS_{S-DCS}-small$		CS_{S-DCS}		CS_{BL}		CS_{Alg3}	
Lena	33.0566	0.9089	33.7859	0.9184	33.3059	0.9111	**34.6557**	**0.9281**
Elaine	32.3990	0.8073	32.6903	0.8145	32.4076	0.8043	**33.1756**	**0.8244**
Man	31.1978	0.8738	31.8509	0.8866	31.4686	0.8782	**32.5941**	**0.8999**
Mandrill	23.4291	0.7598	23.8411	0.7823	23.8221	0.7746	**24.3753**	**0.8007**
Peppers	28.9462	0.8877	29.6975	0.9005	29.4145	0.8925	**30.6859**	**0.9169**
Boat	29.7350	0.8561	30.3488	0.8679	30.1027	0.8580	**31.2858**	**0.8837**
House	31.5166	0.8842	32.0602	0.8985	32.0707	0.8956	**33.0146**	**0.9158**
Cameraman	26.2240	0.8581	26.8272	0.8716	26.4545	0.8673	**27.4254**	**0.8877**
Barbara	25.6148	0.8239	25.9153	0.8316	25.5165	0.8168	**26.0835**	**0.8393**
Tank	30.7233	0.8252	30.8210	0.8369	31.1403	0.8361	**31.7818**	**0.8576**
Averaged	29.2842	0.8485	29.7838	0.8609	29.5703	0.8534	**30.5078**	**0.8754**

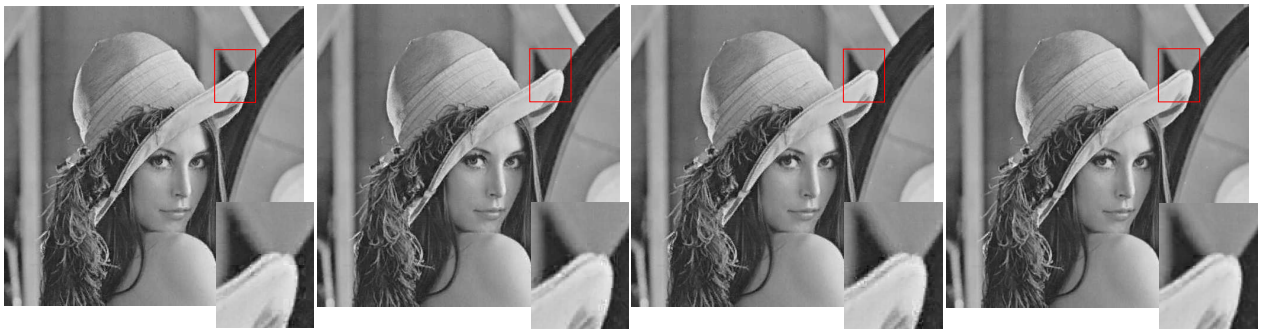


(a) 'Lena' (b) 'Mandrill'

Figure 6.2: The original test images.

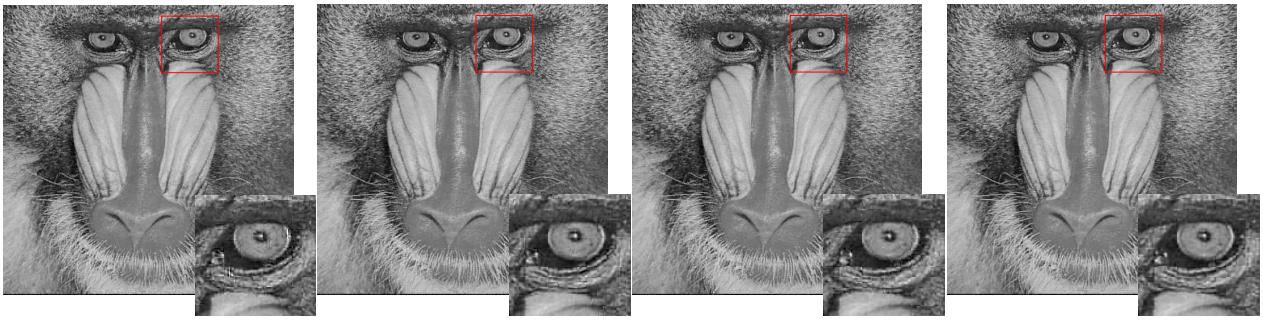
$CS_{S-DCS}-small$ but becomes worse than CS_{S-DCS}. All of these imply the merit of training the sensing matrix and the sparsifying dictionary on a large dataset and Algorithm 6.3 is a good choice for such a mission. To examine the visual effect, the reconstruction of two natural images—'Lena' and 'Mandril' of Figure 6.2—is shown in Figures 6.3 and 6.4.

Now, we experimentally study the convergence of Algorithm 6.3. In this part, we run Algorithm 6.2 again with sufficient iterations after calling Algorithm 6.1, to see whether the dictionary can be improved further. Although we train $\mathbf{\Phi}$ and $\mathbf{\Psi}$ on training data, we prefer to see the objective value on the testing data, on which we wish to test performance. Note that iterations here refers to Max_Iter_{dic} appearing in Algorithm 6.2. From Figure 6.5, we see that even the testing error is not monotonically decreasing, it is asymptotically decreasing meeting the feature of online algorithms because only part of the training data is sampled to update the dictionary at each iteration. We can also observe that the value of ρ_{psnr} on testing data increases versus the number of iterations. From the sub-figure in Figure 6.5, we see that the recovered PSNR increases dramatically after the 1000th iteration in which we update the sensing matrix again. Moreover, we observe that the PSNR still increases as the iterations continued demonstrating the significance of simultaneously optimizing the SMSD. In Figure 6.6, we display the variation of the dictionary at each iteration and observe that there exist many oscillations which are caused by the fact that we update the dictionary stochastically which only chooses part of training data at each iteration.



(a) $CS_{S-DCS}-small$ (b) CS_{S-DCS} (c) CS_{BL} (d) CS_{Alg3}

Figure 6.3: The recovered test image 'Lena'.



(a) $CS_{S-DCS}-small$ (b) CS_{S-DCS} (c) CS_{BL} (d) CS_{Alg3}

Figure 6.4: The recovered test image 'Mandrill'.

However, the envelope of Figure 6.6(a) (see Figure 6.6(b)) implies the convergence of our algorithm which is consistent with Proposition 6.3.3. Note that all of the observations agree with our statements in Section 6.3 regarding the convergence analysis of Algorithm 6.1. However, the complete investigation of the convergence analysis of Algorithm 6.1 is out of the scope in this chapter and is left to future work.

6.5 Conclusion

In this chapter, an efficient algorithm for jointly learning the SMSD on a large dataset is proposed. The proposed algorithm optimizes the sensing matrix with a closed-form solution and learns a sparsifying dictionary with a stochastic method on a large training dataset. Our experiment results show that training the SMSD on a large dataset yields better performance and the proposed method which considers the efficiency and effectiveness simultaneously is a suitable choice for such a task.

One of the possible directions for future research is to accelerate the proposed method. Combining the Sequential Subspace Optimization (SESOP) with the proposed algorithm may be one of the possible methods to boost efficiency [113].

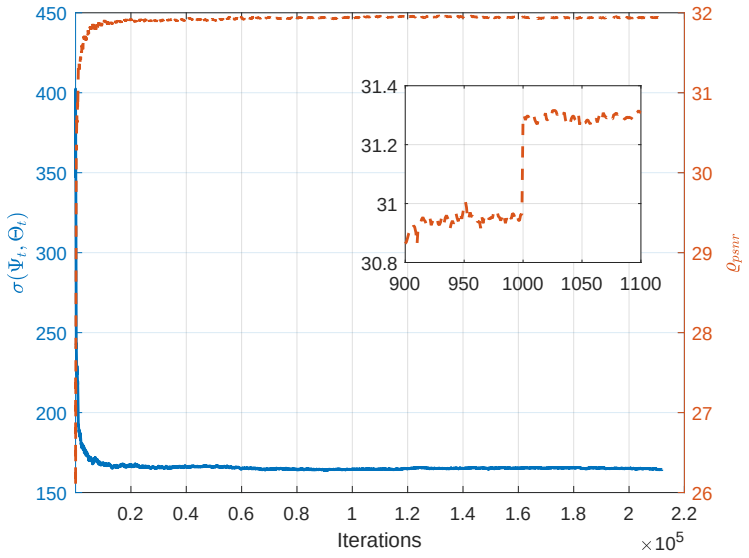
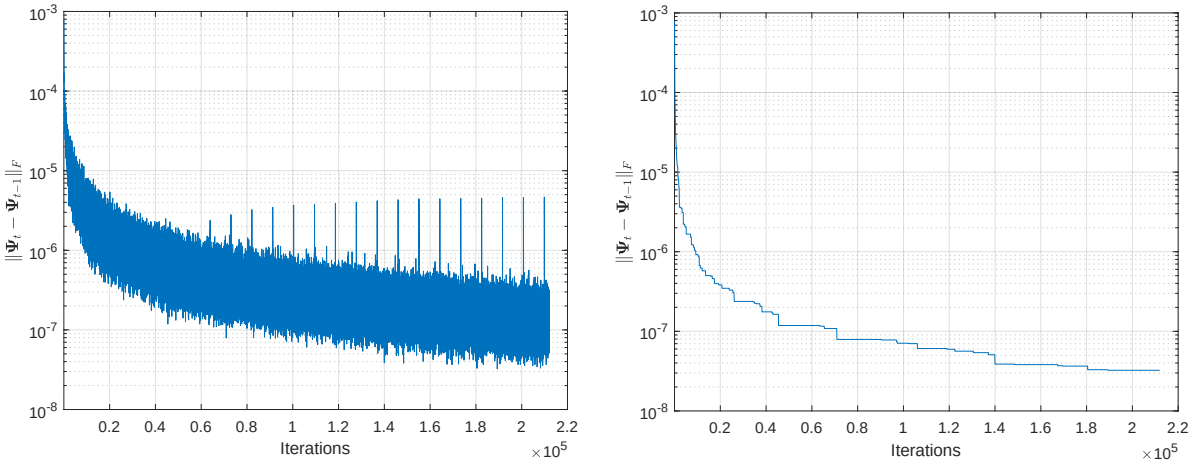


Figure 6.5: Objective value $\sigma(\boldsymbol{\Psi}_t, \boldsymbol{\Theta}_t)$ and σ_{psnr} on testing data using Algorithm 6.3.



(a) (b)

Figure 6.6: (a): The difference between dictionaries of consecutive iterations; (b): The envelope of the differences between consecutive dictionaries.

Chapter 7

Optimized Structured Sparse Sensing Matrices for Compressive Sensing

In this chapter, we describe our work on designing structured sparse sensing matrices for compressive sensing. This chapter is based on the following published paper:

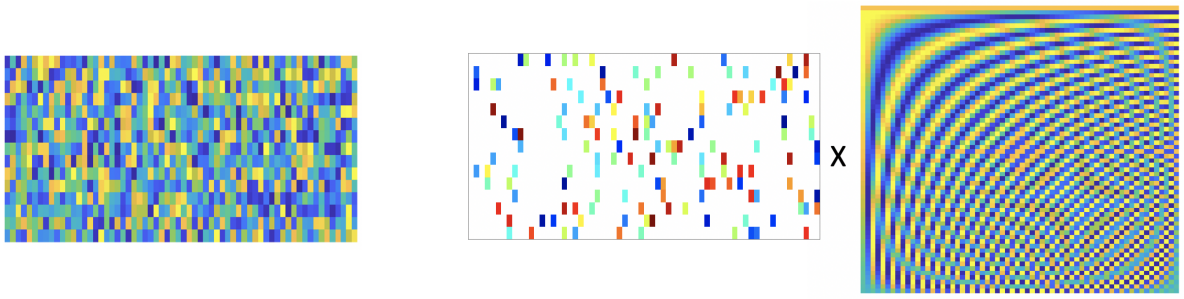
- **Tao Hong**, Xiao Li, Zhihui Zhu, and Qiuwei Li, *Optimized Structured Sparse Sensing Matrices for Compressive Sensing,* Signal Processing, vol. 159, pp. 119-129, Jun. 2019.

The introduction of compressive sensing is omitted in this chapter because it already appears in Chapter 6. Moreover, the notation used in Chapter 6 is also adapted in this chapter, e.g., $\mathbf{\Phi}$–sensing matrix, $\mathbf{\Psi}$–dictionary, $\boldsymbol{\theta}$–sparse coefficients, $\boldsymbol{e}$–sparse representation error, $\|\cdot\|_0$–the number of non-zeros, and MATLAB notation, etc.

7.1 Introduction

In compressive sensing (CS), the sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{M\times N}$ ($M \ll N$) is used for recovering $\boldsymbol{x} \in \mathbb{R}^N$ from its low dimensional measurements $\boldsymbol{y} = \mathbf{\Phi}\boldsymbol{x}$. It has been shown that if the *equivalent dictionary* $\mathbf{\Phi\Psi} \in \mathbb{R}^{M\times L}$ satisfies the restricted isometry property (RIP), the sparse vector $\boldsymbol{\theta} \in \mathbb{R}^L$ as shown in (6.1) can be exactly recovered from $\boldsymbol{y}$ [125, 126]. Although random matrices satisfy the RIP with high probability [125], verifying that a general matrix satisfies the RIP is NP-hard [127]. Alternatively, mutual coherence, another measure of the property of sensing matrices which is much easier to verify, has been introduced in practice to quantify and optimize sensing matrices [102, 107, 128, 103, 105, 104, 108, 114, 106, 129, 120]. Structured sensing matrices (e.g., Toeplitz matrices and sparse matrices) have been proposed [130, 131, 132, 133, 134, 135] to efficiently acquire signals of interest in hardware (such as digital signal processors and FPGA) [136, 137], or applications like electrocardiography (ECG) compression [138] and data streams computing [139]. However, a random sparse sensing matrix is less competitive than an optimized one in terms of the signal recovery accuracy. In this chapter, we consider the design of a structured sparse sensing matrix that can efficiently acquire signals and yields performance that is similar to that of dense matrices. Our main contributions in this chapter are summarized as follows:

(1) We propose a framework to design a *structured sparse sensing matrix* by reducing the mutual coherence (defined in (7.1)) of the equivalent dictionary $\mathbf{\Phi\Psi}$. As shown in Figure 7.1(b), the



(a) (b)

Figure 7.1: (a): A random Gaussian matrix; (b): A structured sparse sensing matrix consisting of a sparse sensing matrix and a base sensing matrix.

structured sparse sensing matrix consists of $\boldsymbol{\Phi} = \bar{\boldsymbol{\Phi}}\boldsymbol{A}$ where $\bar{\boldsymbol{\Phi}} \in \mathbb{R}^{M\times N}$ is a *row-wise sparse matrix* and $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ refers to the *base sensing matrix* such that $\boldsymbol{Ax}$ can be implemented with linear complexity. The choice of $\boldsymbol{A}$ depends on the features of signals of interest, e.g., one can choose a DCT matrix for natural images with a learned $\boldsymbol{\Psi}$ [17]. Moreover, one may simply set $\boldsymbol{A} = \boldsymbol{I}_N$ resulting in a sparse sensing matrix.

(2) We provide an alternating minimization algorithm to address the nonconvex nonsmooth optimization problem (see (7.10)). Despite the nonconvexity and nonsmoothness, we perform a rigorous convergence analysis to show that the sequence of iterates generated by the proposed algorithm with random initialization converges to a critical point.

(3) Experiments on natural images show that the structured sparse sensing matrix obtained—with or without $\boldsymbol{A}$—outperforms a random dense sensing matrix. It is interesting to note that by setting $\boldsymbol{A} = \text{DCT}$, the optimized structured sparse sensing matrix performs, in terms of signal recovery accuracy, almost identically to the optimized dense sensing matrix; see Figure 7.8.

The outline of this chapter is as follows. In Section 7.2, we review the previous approaches for optimizing robust sensing matrices. A new framework for designing the structured sparse sensing matrix is proposed in Section 7.3 with the consideration of minimizing the mutual coherence of the equivalent dictionary and the SREs. Moreover, we provide an alternating minimization algorithm to solve the optimization problem, along with a rigorous convergence analysis, in Section 7.4. We examine the performance of the resulting structured sparse sensing matrix on both synthetic data and real images in Section 7.5. Conclusions are given in Section 7.6.

7.2 Preliminaries

In this section, we briefly review the definition of mutual coherence in CS and introduce the previous approaches for designing robust sensing matrices.

7.2.1 Mutual Coherence

The *mutual coherence* of $\boldsymbol{Q} \in \mathbb{R}^{M\times L}$ is defined as

$$\mu(\boldsymbol{Q}) \triangleq \max_{1\le m\neq n\le L} \frac{|\boldsymbol{q}_m^{\mathcal{T}}\boldsymbol{q}_n|}{\|\boldsymbol{q}_m\|_2\|\boldsymbol{q}_n\|_2} \ge \underline{\mu} \triangleq \sqrt{\frac{L-M}{M(L-1)}}, \tag{7.1}$$

where $\boldsymbol{q}_m$ is the mth column of $\boldsymbol{Q}$, $\underline{\mu}$ is the lower bound of $\mu(\boldsymbol{Q})$ called Welch Bound [140], and $\|\cdot\|_2$ denotes the ℓ_2 vector norm. We refer the reader to [99, Section 5.2.3] for a discussion on the connection between the mutual coherence and the RIP.

Given the measurements $\boldsymbol{y} = \boldsymbol{\Phi x}$, one can recover $\boldsymbol{x}$ by solving

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^L} \|\boldsymbol{y} - \boldsymbol{\Phi\Psi\theta}\|_2^2, \quad \text{s.t.} \quad \|\boldsymbol{\theta}\|_0 \leq K. \tag{7.2}$$

and then $\hat{\boldsymbol{x}} = \boldsymbol{\Psi}\,\hat{\boldsymbol{\theta}}$. Note that (7.2) can be exactly or approximately solved by convex relaxation methods [126, 141, 142] (e.g., basis pursuit) or greedy algorithms [143] (e.g., the orthogonal marching pursuit (OMP)). Moreover, in [143], Tropp showed that OMP can stably find $\boldsymbol{\theta}$ and hence obtain an exact estimation of $\boldsymbol{x}$ if

$$K < \frac{1}{2}\left[1 + \frac{1}{\mu(\boldsymbol{\Phi\Psi})}\right], \tag{7.3}$$

which acts as a crucial criterion for optimizing sensing matrices.

7.2.2 Optimized Robust Sensing Matrix

In light of (7.3), many efforts [115, 102, 128, 103] have been devoted to design the sensing matrix by minimizing the mutual coherence $\mu(\boldsymbol{\Phi\Psi})$ as discussed in chapter 6. We note that most of the previous studies assume $\boldsymbol{e} = \boldsymbol{0}$ (cf. (6.1)), but it was soon realized that such an optimized sensing matrix lacks robustness with respect to $\boldsymbol{e} \neq \boldsymbol{0}$ [104]. For practical signals of interest, $\boldsymbol{e}$ is never nil even for a learned dictionary [17], and therefore we need to design a robust sensing matrix for $\boldsymbol{e} \neq \boldsymbol{0}$.

Denote by $\mathcal{G}_\xi$ the set of relaxed equiangular tight frame (ETF) Gram matrices:

$$\mathcal{G}_\xi = \left\{\boldsymbol{G} \in \mathbb{S}^{L\times L} : \boldsymbol{G}(m,m) = 1, \forall m \max_{m\neq n, \forall m,n} |\boldsymbol{G}(m,n)| \leq \xi\right\}, \tag{7.4}$$

where $\xi \in [0,1)$ is a prescribed parameter which can be set to 0 or $\underline{\mu}$ [128, 103, 104, 114] and $\mathbb{S}^{L\times L}$ is a set of real $L\times L$ symmetric matrices. Denote by $\boldsymbol{X} \in \mathbb{R}^{N\times J}$ a set of training data. We use $\boldsymbol{\Theta}$ to denote the sparse coefficients of $\boldsymbol{X}$ under $\boldsymbol{\Psi}$: $\boldsymbol{X} = \boldsymbol{\Psi\Theta} + \boldsymbol{E}$ where $\|\boldsymbol{\Theta}(:,n)\|_0 \leq K, \forall n$. In [104, 114], the authors used the SRE matrix $\boldsymbol{E} \triangleq \boldsymbol{X} - \boldsymbol{\Psi\Theta}$ as the regularization that the robust sensing matrix is obtained by solving the following problem:

$$\min_{\boldsymbol{\Phi},\boldsymbol{G}\in\mathcal{G}_\xi} \left\|\boldsymbol{G} - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\right\|_F^2 + \lambda\|\boldsymbol{\Phi E}\|_F^2, \tag{7.5}$$

where the first term of (7.5) is used to control the average mutual coherence of the equivalent dictionary and $\lambda \geq 0$ is the trade-off parameter to balance these two terms. As discussed in Chapter 6, one can discard $\boldsymbol{E}$ [106] and solve the following problem instead of (7.5):

$$\min_{\boldsymbol{\Phi},\boldsymbol{G}\in\mathcal{G}_\xi} \left\|\boldsymbol{G} - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{\Phi}^{\mathcal{T}}\boldsymbol{\Phi\Psi}\right\|_F^2 + \lambda\|\boldsymbol{\Phi}\|_F^2, \tag{7.6}$$

as we indeed do in this chapter.

7.3 Optimized Structured Sparse Sensing Matrix

As mentioned above, in applications like ECG compression [138], data streams computing [139], and hardware implementation [136], the classical CS system with a dense sensing matrix $\boldsymbol{\Phi}$ is inefficient. Indeed, applying a dense sensing matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ to acquire $\boldsymbol{x} \in \mathbb{R}^N$ has $O(MN)$ computational complexity, and in practice we will acquire signals many times so we need better efficiency.

In this chapter, we suggest imposing certain structure into the sensing matrix $\boldsymbol{\Phi}$ for improving the efficiency of acquiring signals. To this end, the following structure is adopted in this chapter:

$$\boldsymbol{\Phi} = \bar{\boldsymbol{\Phi}}\boldsymbol{A}, \tag{7.7}$$

where $\bar{\boldsymbol{\Phi}} \in \mathbb{R}^{M\times N}$ is a row-wise sparse matrix and $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ denotes the base matrix. To achieve reduction of acquisition complexity, the base sensing matrix $\boldsymbol{A}$ should be either the *identity matrix* or some other matrix for which $\boldsymbol{Ax}$ *can be computed with low (preferably linear) complexity*. In practice, the choice of $\boldsymbol{A}$ depends on the features of signals of interest, e.g., one can choose a DCT matrix for natural images [144]. Rewrite (7.7) as

$$\boldsymbol{\Phi}^{\mathcal{T}} = \boldsymbol{A}^{\mathcal{T}}\bar{\boldsymbol{\Phi}}^{\mathcal{T}}, \tag{7.8}$$

and consider $\bar{\boldsymbol{\Phi}}^{\mathcal{T}}$ as the sparse representation of $\boldsymbol{\Phi}^{\mathcal{T}}$ in $\boldsymbol{A}^{\mathcal{T}}$. Thus, similarly to (6.1), we also call $\boldsymbol{\Phi}$ a sparse sensing matrix in $\boldsymbol{A}$. Note that the structure of (7.8) also appears in the double-sparsity dictionary learning task [144], the dictionary $\boldsymbol{\Psi} = \boldsymbol{A}\bar{\boldsymbol{\Psi}}$ with $\bar{\boldsymbol{\Psi}}$ column-wise sparse.

Adopting (7.6), we solve the following problem to find $\bar{\boldsymbol{\Phi}}$:

$$\begin{aligned}\{\widetilde{\boldsymbol{\Phi}}, \widetilde{\boldsymbol{G}}\} = \arg\min_{\bar{\boldsymbol{\Phi}}, \boldsymbol{G}\in\mathcal{G}_\xi} & ||\boldsymbol{G} - \boldsymbol{\Psi}^{\mathcal{T}}\boldsymbol{A}^{\mathcal{T}}\bar{\boldsymbol{\Phi}}^{\mathcal{T}}\bar{\boldsymbol{\Phi}}\boldsymbol{A}\boldsymbol{\Psi}||_F^2 + \lambda\|\bar{\boldsymbol{\Phi}}\boldsymbol{A}\|_F^2, \\ \text{s.t.} \quad & \|\bar{\boldsymbol{\Phi}}(m,:)\|_0 \le \kappa, \ \forall\, m,\end{aligned} \tag{7.9}$$

where κ denotes the number of non-zeros in each row of the sensing matrix. For $\boldsymbol{A} = \boldsymbol{I}_N$, the computational complexity of $\bar{\boldsymbol{\Phi}}\boldsymbol{x} = \widetilde{\boldsymbol{\Phi}}\boldsymbol{x}$ is $O(M\kappa)$ which is the same as the one shown in [132]. By choosing a base matrix with fast implementation of $\boldsymbol{Ax}$, we only slightly increase the complexity, e.g.,if $\boldsymbol{A} = \text{DCT}$, $\boldsymbol{Ax}$ only requires $O(N\log N)$. Moreover, in some cases, $\boldsymbol{Ax}$ can be implemented with $O(N)$, e.g., if $\boldsymbol{A}$ is an orthogonal matrix and then $\boldsymbol{A}$ can be decomposed into a series of Givens rotation matrices [145]. Compared with a dense sensing matrix which requires $O(MN)$ operations to acquire signals, our structure $\bar{\boldsymbol{\Phi}}\boldsymbol{A}$ saves significant computation, especially for large N and $N \gg \kappa$. In Figure 7.2, we show the difference between $O(MN)$ and $O(N\log N + M\kappa)$ with varying κ and N.

7.4 Alternating Gradient Projection Algorithm for Designing Structured Sensing Matrix

Besides the fact that $\boldsymbol{\Phi}$ is parameterized by $\bar{\boldsymbol{\Phi}}\boldsymbol{A}$, (7.9) differs from (7.6) in that the former has a sparse constraint on the rows of $\bar{\boldsymbol{\Phi}}$ such that (7.9) becomes highly nonconvex. In this section, we suggest using an alternating projected gradient method to address (7.9) with a rigorous convergence analysis.

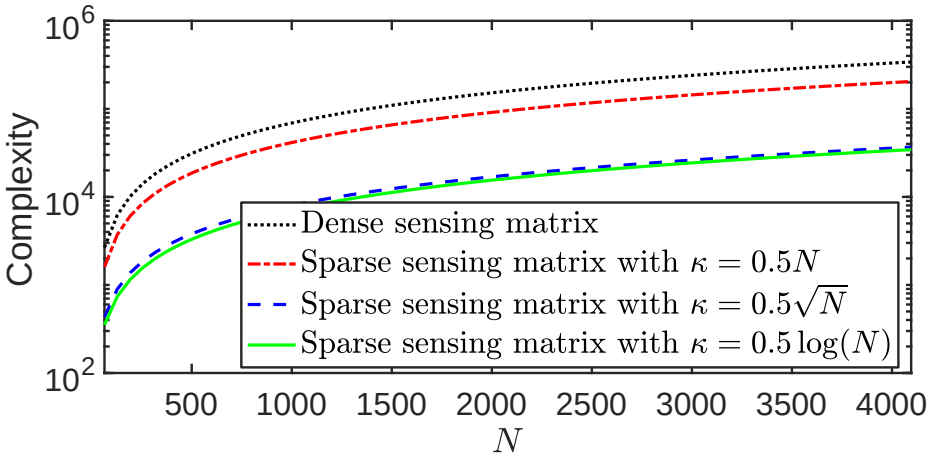


Figure 7.2: Comparison of $O(MN)$ (black line) and $O(N\log N + M\kappa)$ (other three lines, refering to different κ) with $M = 10\log N$.

Assume $\boldsymbol{A}$ is not null and rewrite (7.9) as

$$\begin{aligned} &\min_{\boldsymbol{\Phi},\boldsymbol{G}} f(\boldsymbol{\Phi},\boldsymbol{G}) \triangleq ||\boldsymbol{G} - \bar{\boldsymbol{\Psi}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\bar{\boldsymbol{\Psi}}||_F^2 + \lambda||\boldsymbol{\Phi}||_F^2, \\ &\text{s.t. } \|\boldsymbol{\Phi}(m,:)\|_0 \le \kappa,\ \forall\, m,\ \boldsymbol{G} \in \mathcal{G}_\xi, \end{aligned} \tag{7.10}$$

where $\bar{\boldsymbol{\Psi}} = \boldsymbol{A}\boldsymbol{\Psi}$. Without any confusion, we will use $\boldsymbol{\Phi}$ instead of $\bar{\boldsymbol{\Phi}}$ in the following analysis and distinguish between them only if it is not clear from the context. Denote by $\mathcal{P}_{\mathcal{G}_\xi} : \mathbb{R}^{L\times L} \to \mathbb{R}^{L\times L}$ an orthogonal projection operator on the set $\mathcal{G}_\xi$ such that

$$\left(\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{G})\right)(m,n) = \begin{cases} 1, & m = n, \\ \text{sgn}(\boldsymbol{G}(m,n))\min(|\boldsymbol{G}(m,n)|,\xi), & m \ne n, \end{cases}$$

where $\text{sgn}(\cdot)$ is the sign function. For fixed $\boldsymbol{\Phi}$, the solution of $\min_{\boldsymbol{G}\in\mathcal{G}_\xi} f(\boldsymbol{\Phi},\boldsymbol{G})$ is given by

$$\widehat{\boldsymbol{G}} = \mathcal{P}_{\mathcal{G}_\xi}(\bar{\boldsymbol{\Psi}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\bar{\boldsymbol{\Psi}}). \tag{7.11}$$

Now we consider (7.10) with fixed $\boldsymbol{G}$, i.e.,

$$\min_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi},\boldsymbol{G}), \quad \text{s.t. } \|\boldsymbol{\Phi}(m,:)\|_0 \le \kappa,\ \forall\, m. \tag{7.12}$$

Without the sparsity constraint, recent works [118, 146] showed that any gradient based algorithms can provably solve $\min_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi},\boldsymbol{G})$. To address the constraint, we suggest using projected gradient descent (PGD). Denote by $\mathcal{S}_\kappa$ the set of matrices that has at most κ non-zeros in each row,

$$\mathcal{S}_\kappa \triangleq \left\{\boldsymbol{Z} \in \mathbb{R}^{M\times N} : \|\boldsymbol{Z}(m,:)\|_0 \le \kappa,\ \forall\, m\right\},$$

and $\mathcal{P}_{\mathcal{S}_\kappa} : \mathbb{R}^{M\times N} \to \mathbb{R}^{M\times N}$ an orthogonal project operator on the set $\mathcal{S}_\kappa$ that chooses the largest κ absolute values of each row. Since $\mathcal{S}_\kappa$ is nonconvex, $\mathcal{P}_{\mathcal{S}_\kappa}$ may yield multiple solutions and we randomly choose one

of them in practice. At the kth step, PDG runs

$$\boldsymbol{\Phi}_k = \mathcal{P}_{\mathcal{S}_\kappa}\left(\boldsymbol{\Phi}_{k-1} - \eta \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_{k-1}, \boldsymbol{G}_{k-1})\right), \tag{7.13}$$

where $\nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_{k-1}, \boldsymbol{G}_{k-1})$ is the gradient of $f(\boldsymbol{\Phi}_{k-1}, \boldsymbol{G}_{k-1})$ with respect to $\boldsymbol{\Phi}$ for $\boldsymbol{\Phi}_{k-1}$ and η is the fixed stepsize.

The corresponding alternating gradient projection method for (7.10) is summarized in Algorithm 7.1. We note that the use of alternating minimization algorithms for designing sensing matrix can also be found in [102, 107, 128, 103, 104, 114] but the convergence of these algorithms is usually neither ensured nor seriously considered. Following, we provide a rigorous convergence analysis of Algorithm 7.1.

Algorithm 7.1 Algorithm for Designing Sparse Sensing Matrix

Input: Initial values $\boldsymbol{\Phi}_0$ and $\boldsymbol{G}_0$, maximal iterations Max_Iter, fixed stepsize η, and sparsity level κ.
Output: Sparse sensing matrix $\boldsymbol{\Phi}_{\text{Max\_Iter}}$.

1: $k \leftarrow 1$.
2: **while** $k \leq$ Max_Iter **do**
3: $\quad \boldsymbol{\Phi}_k \leftarrow \mathcal{P}_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k-1} - \eta \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_{k-1}, \boldsymbol{G}_{k-1}))$.
4: $\quad \boldsymbol{G}_k \leftarrow \mathcal{P}_{\mathcal{G}_\xi}(\bar{\boldsymbol{\Psi}}^{\mathcal{T}} \boldsymbol{\Phi}_k^{\mathcal{T}} \boldsymbol{\Phi}_k \bar{\boldsymbol{\Psi}})$.
5: $\quad k \leftarrow k+1$.
6: **end while**

7.4.1 Convergence Analysis

To analyse the convergence of Algorithm 7.1, we rewrite (7.10) in the following equivalent unconstrained form

$$\min_{\boldsymbol{\Phi}, \boldsymbol{G}} \rho(\boldsymbol{\Phi}, \boldsymbol{G}) := f(\boldsymbol{\Phi}, \boldsymbol{G}) + \delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}) + \delta_{\mathcal{G}_\xi}(\boldsymbol{G}), \tag{7.14}$$

where $\delta_{\mathcal{S}}(\boldsymbol{Q}) = \begin{cases} 0, & \boldsymbol{Q} \in \mathcal{S}, \\ \infty, & \boldsymbol{Q} \notin \mathcal{S} \end{cases}$. Denote by L_{ρ_0} a sublevel set of $\rho(\boldsymbol{\Phi}, \boldsymbol{G})$,

$$\mathcal{L}_{\rho_0} = \left\{(\boldsymbol{\Phi}, \boldsymbol{G}) : \rho(\boldsymbol{\Phi}, \boldsymbol{G}) \leq \rho_0, \boldsymbol{G} \in \mathcal{G}_\xi, \boldsymbol{\Phi} \in \mathcal{S}_\kappa\right\},$$

where $\rho_0 = \rho(\boldsymbol{\Phi}_0, \boldsymbol{G}_0)$ with $\boldsymbol{\Phi}_0, \boldsymbol{G}_0$ the initial values of Algorithm 7.1. In the following proof, we will see that

$$\rho(\boldsymbol{\Phi}_k, \boldsymbol{G}_k) \leq \rho(\boldsymbol{\Phi}_k, \boldsymbol{G}_{k-1}) \leq \rho(\boldsymbol{\Phi}_{k-1}, \boldsymbol{G}_{k-1}),$$

after calling Lines 3 and 4 in Algorithm 7.1, that $(\boldsymbol{\Phi}_k, \boldsymbol{G}_k) \in \mathcal{L}_{\rho_0}$, $\forall k$ and $\|\boldsymbol{G}\|_F$, $\|\boldsymbol{\Phi}\|_F$ are finite because $\boldsymbol{G} \in \mathcal{G}_\xi$ and $\rho(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \to \infty$ when $\|\boldsymbol{\Phi}\|_F \to \infty$. With these and the differentiability of $f(\boldsymbol{\Phi}, \boldsymbol{G})$, we have that both $\nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}, \boldsymbol{G})$ and $\nabla_{\boldsymbol{G}} f(\boldsymbol{\Phi}, \boldsymbol{G})$ are Lipshitz continuous:

$$\begin{aligned} \|\nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}, \boldsymbol{G}) - \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}', \boldsymbol{G})\|_F &\leq L_c \|\boldsymbol{\Phi} - \boldsymbol{\Phi}'\|_F, \\ \|\nabla_{\boldsymbol{G}} f(\boldsymbol{\Phi}, \boldsymbol{G}) - \nabla_{\boldsymbol{G}} f(\boldsymbol{\Phi}, \boldsymbol{G}')\|_F &\leq L_c \|\boldsymbol{G} - \boldsymbol{G}'\|_F, \end{aligned} \tag{7.15}$$

for all $(\boldsymbol{\Phi},\boldsymbol{G}),(\boldsymbol{\Phi}',\boldsymbol{G}),(\boldsymbol{\Phi},\boldsymbol{G}')\in\mathcal{L}_{\rho_0}$ where $L_c>0$ is the Lipschitz constant. A direct consequence of the Lipschitz continuity is as follows.

Lemma 7.4.1. *For any $L\geq L_c$, denote by*

$$\begin{aligned}h_L(\boldsymbol{\Phi},\boldsymbol{\Phi}',\boldsymbol{G})\triangleq& f(\boldsymbol{\Phi}',\boldsymbol{G})+\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle\\&+\frac{L}{2}\|\boldsymbol{\Phi}-\boldsymbol{\Phi}'\|_F^2.\end{aligned}$$

Then, $f(\boldsymbol{\Phi},\boldsymbol{G})\leq h_L(\boldsymbol{\Phi},\boldsymbol{\Phi}',\boldsymbol{G})$ for all $(\boldsymbol{\Phi},\boldsymbol{G}),(\boldsymbol{\Phi}',\boldsymbol{G})\in\mathcal{L}_{\rho_0}$.

Proof. Denote by $\upsilon(t)=f(\boldsymbol{\Phi}'+t(\boldsymbol{\Phi}-\boldsymbol{\Phi}'),\boldsymbol{G})$. Evidently, $\upsilon(0)=f(\boldsymbol{\Phi}',\boldsymbol{G}),\ \upsilon(1)=f(\boldsymbol{\Phi},\boldsymbol{G})$ and then we have

$$\begin{aligned}&\upsilon(1)-\upsilon(0)=f(\boldsymbol{\Phi},\boldsymbol{G})-f(\boldsymbol{\Phi}',\boldsymbol{G})=\int_0^1\upsilon'(t)dt\\&=\int_0^1\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}'+t(\boldsymbol{\Phi}-\boldsymbol{\Phi}'),\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle dt\\&=\int_0^1\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}'+t(\boldsymbol{\Phi}-\boldsymbol{\Phi}'),\boldsymbol{G})-\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle dt\\&\quad+\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle\\&\overset{(*)}{\leq}\int_0^1\|\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}'+t(\boldsymbol{\Phi}-\boldsymbol{\Phi}'),\boldsymbol{G})-\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G})\|_F dt\\&\quad\cdot\|\boldsymbol{\Phi}-\boldsymbol{\Phi}'\|_F+\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle\\&\overset{(7.15)}{\leq}L\|\boldsymbol{\Phi}-\boldsymbol{\Phi}'\|_F^2\int_0^1 tdt+\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle\\&=\frac{L}{2}\|\boldsymbol{\Phi}-\boldsymbol{\Phi}'\|_F^2+\langle\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}',\boldsymbol{G}),\boldsymbol{\Phi}-\boldsymbol{\Phi}'\rangle,\end{aligned}$$

where $(*)$ follows from the Cauchy–Schwarz inequality. □

With Lemma 7.4.1, we can establish the following theorem stating that the sequence $(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)$ generated by Algorithm 7.1 is bounded and the limit point of any convergent subsequence is a stationary point of $\rho(\boldsymbol{\Phi},\boldsymbol{G})$.

Theorem 7.1 (Subsequence convergence)**.** *Denote by $\{\boldsymbol{W}_k=(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)\}_{k\geq0}$ the sequence generated by Algorithm 7.1 with fixed stepsize $\eta<\frac{1}{L_c}$. Then the sequence $\{\boldsymbol{W}_k\}$ is bounded and obeys the following properties:*

(P1) sufficient decrease:

$$\begin{aligned}\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)-\rho(\boldsymbol{\Phi}_{k+1},\boldsymbol{G}_k)&\geq\tfrac{\frac{1}{\eta}-L_c}{2}\|\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k+1}\|_F^2,\\\rho(\boldsymbol{\Phi}_{k+1},\boldsymbol{G}_k)-\rho(\boldsymbol{\Phi}_{k+1},\boldsymbol{G}_{k+1})&\geq\|\boldsymbol{G}_k-\boldsymbol{G}_{k+1}\|_F^2.\end{aligned}\tag{7.16}$$

(P2) the sequence $\{\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)\}_{k\geq0}$ is convergent.

(P3) convergence of $\boldsymbol{W}_k$:

$$\lim_{k\to\infty}\|\boldsymbol{W}^{k+1}-\boldsymbol{W}^k\|_F=0.\tag{7.17}$$

(P4) for any convergent subsequence $\{\boldsymbol{W}_{k'}\}$, its limit point $\underline{\boldsymbol{W}}$ is a stationary point of $\rho(\boldsymbol{W})$ and

$$\lim_{k'\to\infty}\rho(\boldsymbol{W}_{k'})=\lim_{k\to\infty}\rho(\boldsymbol{W}_k)=\rho(\underline{\boldsymbol{W}}). \tag{7.18}$$

Proof. We first define the subdifferential for a general lower semicontinuous function which will be used in the following proof.

Definition 7.4.2 (Subdifferentials [147])**.** Let $\sigma:\mathbb{R}^d\to(-\infty,\infty]$ be a proper and lower semicontinuous function, whose domain is defined as

$$\operatorname{dom}\sigma:=\left\{\boldsymbol{u}\in\mathbb{R}^d:\sigma(\boldsymbol{u})<\infty\right\}.$$

The Fréchet subdifferential of σ at $\boldsymbol{u}$ is defined by

$$\partial\sigma_F(\boldsymbol{u})=\left\{\boldsymbol{z}:\liminf_{\boldsymbol{v}\to\boldsymbol{u}}\frac{\sigma(\boldsymbol{v})-\sigma(\boldsymbol{u})-\langle\boldsymbol{z},\boldsymbol{v}-\boldsymbol{u}\rangle}{\|\boldsymbol{u}-\boldsymbol{v}\|}\geq 0\right\},$$

for any $\boldsymbol{u}\in\operatorname{dom}\sigma$ and $\partial_F\sigma(\boldsymbol{u})=\emptyset$ if $\boldsymbol{u}\notin\operatorname{dom}\sigma$. The subdifferential of σ at $\boldsymbol{u}\in\operatorname{dom}\sigma$ is defined by

$$\partial\sigma(\boldsymbol{u})=\{\boldsymbol{z}:\exists\boldsymbol{u}_k\to\boldsymbol{u},\sigma(\boldsymbol{u}_k)\to\sigma(\boldsymbol{u}),\boldsymbol{z}_k\in\partial_F\sigma(\boldsymbol{u}_k)\to\boldsymbol{z}\}.$$

We call $\boldsymbol{u}$ a critical or stationary point if $\partial\sigma(\boldsymbol{u})=\boldsymbol{0}$. Now we prove the four arguments of Theorem 7.1 separately.

Show (P1): Since $\boldsymbol{\Phi}_k\in\mathcal{S}_\kappa$ and $\boldsymbol{G}_k\in\mathcal{G}_\xi$, $\forall k\in\mathbb{N}$, we have $\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}_\ell)=f(\boldsymbol{\Phi}_k,\boldsymbol{G}_\ell)$, $\forall k,\ell\in\mathbb{N}$ and then

$$\begin{aligned}
&\rho(\boldsymbol{\Phi}_{k+1},\boldsymbol{G}_k)-\rho(\boldsymbol{\Phi}_{k+1},\boldsymbol{G}_{k+1})\\
&=\|\boldsymbol{G}_k-\boldsymbol{B}_{k+1}\|_F^2-\|\boldsymbol{G}_{k+1}-\boldsymbol{B}_{k+1}\|_F^2\\
&=\|\boldsymbol{G}_k-\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{B}_{k+1})+\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{B}_{k+1})-\boldsymbol{B}_{k+1}\|_F^2-\|\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{B}_{k+1})-\boldsymbol{B}_{k+1}\|_F^2\\
&=\|\boldsymbol{G}_k-\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{B}_{k+1})\|_F^2+2\underbrace{\left\langle\boldsymbol{G}_k-\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{B}_{k+1}),\mathcal{P}_{\mathcal{G}_\xi}(\boldsymbol{B}_{k+1})-\boldsymbol{B}_{k+1}\right\rangle}_{\geq 0\text{ second projection theorem [54]}}\\
&\geq\|\boldsymbol{G}_k-\boldsymbol{G}_{k+1}\|_F^2.
\end{aligned}$$

where $\boldsymbol{B}_{k+1}=\bar{\boldsymbol{\Psi}}^{\mathcal{T}}\boldsymbol{\Phi}_{k+1}^{\mathcal{T}}\boldsymbol{\Phi}_{k+1}\bar{\boldsymbol{\Psi}}$. From (7.13), we have

$$\begin{aligned}
\boldsymbol{\Phi}_{k+1}&\in\mathcal{P}_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_k-\eta\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_k))\\
&=\operatorname*{argmin}_{\boldsymbol{Z}\in\mathcal{S}_\kappa}\|\boldsymbol{Z}-(\boldsymbol{\Phi}_k-\eta\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_k))\|_F^2\\
&\in\operatorname*{argmin}_{\boldsymbol{Z}\in\mathcal{S}_\kappa}h_{1/\eta}(\boldsymbol{Z},\boldsymbol{\Phi}_k,\boldsymbol{G}_k).
\end{aligned}\tag{7.19}$$

implying

$$h_{1/\eta}(\boldsymbol{\Phi}_{k+1},\boldsymbol{\Phi}_k,\boldsymbol{G}_k)\leq h_{1/\eta}(\boldsymbol{\Phi}_k,\boldsymbol{\Phi}_k,\boldsymbol{G}_k) \tag{7.20}$$

$$=f(\boldsymbol{\Phi}_k,\boldsymbol{G}_k). \tag{7.21}$$

In view of Lemma 7.4.1, we get

$$\begin{aligned}
&f(\boldsymbol{\Phi}_k, \boldsymbol{G}_k) - f(\boldsymbol{\Phi}_{k+1}, \boldsymbol{G}_k) \\
&\overset{\text{Lemma 7.4.1}}{\geq} f(\boldsymbol{\Phi}_k, \boldsymbol{G}_k) - h_{L_c}(\boldsymbol{\Phi}_{k+1}, \boldsymbol{\Phi}_k, \boldsymbol{G}_k) \\
&\overset{(7.21)}{\geq} h_{1/\eta}(\boldsymbol{\Phi}_{k+1}, \boldsymbol{\Phi}_k, \boldsymbol{G}_k) - h_{L_c}(\boldsymbol{\Phi}_{k+1}, \boldsymbol{\Phi}_k, \boldsymbol{G}_k) \\
&= \frac{\frac{1}{\eta} - L_c}{2} \|\boldsymbol{\Phi}_k - \boldsymbol{\Phi}_{k+1}\|_F^2,
\end{aligned}$$

yielding the desired result.

Show (P2): From (P1), we have

$$\begin{aligned}
&\rho_0 \geq \rho(\boldsymbol{\Phi}_1, \boldsymbol{G}_0) \geq \rho(\boldsymbol{\Phi}_1, \boldsymbol{G}_1) \geq \cdots \rho(\boldsymbol{\Phi}_k, \boldsymbol{G}_k) \\
&\geq \rho(\boldsymbol{\Phi}_{k+1}, \boldsymbol{G}_k) \geq \rho(\boldsymbol{\Phi}_{k+1}, \boldsymbol{G}_{k+1}) \geq \cdots \geq 0,
\end{aligned}$$

implying the convergence of sequence $\{\rho(\boldsymbol{\Phi}_k, \boldsymbol{G}_k)\}_{k\geq 0}$.

Show (P3): Summing (7.16) from $k = 1$ to ∞, we get

$$\begin{aligned}
&\sum_{k=0}^{\infty} \frac{\frac{1}{\eta} - L_c}{2} \left(\|\boldsymbol{\Phi}^k - \boldsymbol{\Phi}^{k+1}\|_F^2 \right) + \|\boldsymbol{G}^k - \boldsymbol{G}^{k+1}\|_F^2 \\
&\leq \rho_0 - \lim_{k\to\infty} \rho(\boldsymbol{\Phi}_k, \boldsymbol{G}_k) \leq \rho_0,
\end{aligned}$$

suggesting the convergence of the series $\{\sum_{k=0}^{n} \|\boldsymbol{\Phi}^k - \boldsymbol{\Phi}^{k+1}\|_F^2 + \|\boldsymbol{G}^k - \boldsymbol{G}^{k+1}\|_F^2\}_n$. Since $\|\boldsymbol{\Phi}^k - \boldsymbol{\Phi}^{k+1}\|_F^2 \geq 0$ and $\|\boldsymbol{G}^k - \boldsymbol{G}^{k+1}\|_F^2 \geq 0$, (7.17) is derived.

Show (P4): Rewrite (7.19) as

$$\boldsymbol{\Phi}_{k+1} \in \arg\min_{\boldsymbol{\Phi}\in\mathbb{R}^{M\times N}} h_{\frac{1}{\eta}}(\boldsymbol{\Phi}, \boldsymbol{\Phi}_k, \boldsymbol{G}_k) + \delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}). \tag{7.22}$$

Since $\boldsymbol{\Phi}_{k+1}$ is the optimal solution of (7.22), we get

$$\begin{aligned}
&\langle \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_k, \boldsymbol{G}_k), \boldsymbol{\Phi}_{k+1} - \boldsymbol{\Phi}_k \rangle + \frac{\eta}{2} \|\boldsymbol{\Phi}_{k+1} - \boldsymbol{\Phi}_k\|_F^2 + \delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k+1}) \\
&\leq \langle \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_k, \boldsymbol{G}_k), \underline{\boldsymbol{\Phi}} - \boldsymbol{\Phi}_k \rangle + \frac{\eta}{2} \|\underline{\boldsymbol{\Phi}} - \boldsymbol{\Phi}_k\|_F^2 + \delta_{\mathcal{S}_\kappa}(\underline{\boldsymbol{\Phi}}),
\end{aligned}$$

where $\underline{\boldsymbol{\Phi}} = \boldsymbol{\Phi}$ is the limit of a convergent subsequence $\{\boldsymbol{\Phi}_{k'}\}_{k'}$. By setting $k+1 \to k'$ and $k \to k'-1$ for the above equation and taking the limit of the subsequence $\{\boldsymbol{\Phi}_{k'}\}_{k'}$, we further get

$$\begin{aligned}
&\limsup_{k'\to\infty} \delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'}) - \delta_{\mathcal{S}_\kappa}(\underline{\boldsymbol{\Phi}}) \\
&\leq \limsup_{k'\to\infty} \langle \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_{k'-1}, \boldsymbol{G}_{k'-1}), \underline{\boldsymbol{\Phi}} - \boldsymbol{\Phi}_{k'} \rangle \\
&\quad + \frac{\eta}{2} \|\underline{\boldsymbol{\Phi}} - \boldsymbol{\Phi}_{k'-1}\|_F^2 - \frac{\eta}{2} \|\boldsymbol{\Phi}_{k'} - \boldsymbol{\Phi}_{k'-1}\|_F^2 \\
&= 0,
\end{aligned} \tag{7.23}$$

where the equality follows from the facts that (i) $\limsup_{k'\to\infty} \langle \nabla_{\boldsymbol{\Phi}} f(\boldsymbol{\Phi}_{k'-1}, \boldsymbol{G}_{k'-1}), \underline{\boldsymbol{\Phi}} - \boldsymbol{\Phi}_{k'} \rangle = 0$ (the scalar

product is continuous); (ii) $\|\boldsymbol{\Phi}_{k'}-\boldsymbol{\Phi}_{k'-1}\|_F^2=0$ ((7.17)); (iii) $\|\underline{\boldsymbol{\Phi}}-\boldsymbol{\Phi}_{k'-1}\|_F^2=0$ ($0\leq \lim_{k'\to\infty}\|\underline{\boldsymbol{\Phi}}-\boldsymbol{\Phi}_{k'-1}\|_F=\lim_{k'\to\infty}\|\underline{\boldsymbol{\Phi}}-\boldsymbol{\Phi}_{k'}+\boldsymbol{\Phi}_{k'}-\boldsymbol{\Phi}_{k'-1}\|_F\leq \lim_{k'\to\infty}\|\underline{\boldsymbol{\Phi}}-\boldsymbol{\Phi}_{k'}\|_F+\|\boldsymbol{\Phi}_{k'}-\boldsymbol{\Phi}_{k'-1}\|_F=0$). By using $\delta_{\mathcal{S}_\kappa}(\underline{\boldsymbol{\Phi}})\leq \liminf_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})$ ($\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi})$ is lower semi-continuous) and (7.23), we get

$$\limsup_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})\leq \delta_{\mathcal{S}_\kappa}(\underline{\boldsymbol{\Phi}})\leq \liminf_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'}),$$

which, together with the fact that $\liminf_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})\leq \limsup_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})$, yields $\delta_{\mathcal{S}_\kappa}(\underline{\boldsymbol{\Phi}})=\liminf_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})=\limsup_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})$, and hence $\lim_{k'\to\infty}\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})=\delta_{\mathcal{S}_\kappa}(\underline{\boldsymbol{\Phi}})$.

Since $\mathcal{G}_\xi$ is a compact set and $\boldsymbol{G}_{k'}\in\mathcal{G}_\xi$, $\forall\, k'\in\mathbb{N}$, we know the limit point $\underline{\boldsymbol{G}}\in\mathcal{G}_\xi$. With these, we can get $\lim_{k'\to\infty}\rho(\boldsymbol{\Phi}_{k'},\boldsymbol{G}_{k'})=\lim_{k'\to\infty}f(\boldsymbol{\Phi}_{k'},\boldsymbol{G}_{k'})+\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_{k'})+\delta_{\mathcal{G}_\xi}(\boldsymbol{G}_{k'})=\rho(\underline{\boldsymbol{\Phi}},\underline{\boldsymbol{G}})$. Following, we prove $\underline{\boldsymbol{W}}=(\underline{\boldsymbol{\Phi}},\underline{\boldsymbol{G}})$ is a stationary point of $\rho(\boldsymbol{\Phi},\boldsymbol{G})$ which is equivalent to showing $(\boldsymbol{0},\boldsymbol{0})\in\partial\rho(\underline{\boldsymbol{\Phi}},\underline{\boldsymbol{G}})$. Note that one can obtain a stronger result that $(\boldsymbol{0},\boldsymbol{0})\in\lim_{k\to\infty}\partial\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)$.

Note that the optimality condition of (7.22) reads (by setting $k\leftarrow k-1$) [148]

$$\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_{k-1},\boldsymbol{G}_{k-1})+\frac{1}{\eta}(\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k-1})+\boldsymbol{V}_k=\boldsymbol{0},$$

where $\boldsymbol{V}_k\in\partial\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_k)$. With this equality, we get

$$\underbrace{\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{W}_k)-\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{W}_{k-1})-\frac{1}{\eta}(\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k-1})}_{\boldsymbol{D}_{\boldsymbol{\Phi}_k}}\in\partial_{\boldsymbol{\Phi}}\rho(\boldsymbol{W}_k). \tag{7.24}$$

Note that $\boldsymbol{G}_k$ is the solution of

$$\min_{\boldsymbol{G}\in\mathbb{R}^{L\times L}}\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}), \tag{7.25}$$

and the optimality condition of (7.25) reads

$$\boldsymbol{0}=\underbrace{\nabla_{\boldsymbol{G}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)+\boldsymbol{U}_k}_{\boldsymbol{D}_{\boldsymbol{G}_k}}\in\partial_{\boldsymbol{G}}\rho(\boldsymbol{W}_k), \tag{7.26}$$

where $\boldsymbol{U}_k\in\partial\delta_{\mathcal{G}_\xi}(\boldsymbol{G}_k)$. With (7.24) and (7.26), we have

$$\begin{aligned}
&\|(\boldsymbol{D}_{\boldsymbol{\Phi}_k},\boldsymbol{D}_{\boldsymbol{G}_k})\|_F=\|\boldsymbol{D}_{\boldsymbol{\Phi}_k}\|_F\\
&=\|\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{W}_k)-\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{W}_{k-1})-\frac{1}{\eta}(\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k-1})\|_F\\
&\leq\|\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{W}_k)-\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_{k-1})\|\\
&\quad+\|\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_{k-1})-\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{W}_{k-1})\|\\
&\quad+\frac{1}{\eta}\|\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k-1}\|_F\\
&\overset{(7.15)}{\leq} L_c\|\boldsymbol{G}_k-\boldsymbol{G}_{k-1}\|_F+(L_c+\frac{1}{\eta})\|\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k-1}\|_F\\
&\leq(2L_c+\frac{1}{\eta})\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F.
\end{aligned} \tag{7.27}$$

Combining (7.27) with (7.17), we get

$$\lim_{k\to\infty}(\boldsymbol{D}_{\boldsymbol{\Phi}_k},\boldsymbol{D}_{\boldsymbol{G}_k})=(\mathbf{0},\mathbf{0}),$$

and thus $(\mathbf{0},\mathbf{0})\in\lim_{k\to\infty}\partial\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)$ so that any convergent subsequence of $\{\boldsymbol{W}_k\}_k$ converges to a stationary point of (7.14). The statement $\lim_{k\to\infty}\rho(\boldsymbol{W}_k)=\rho(\underline{\boldsymbol{W}})$ directly follows from (P2), so the sequence $\{\rho(\boldsymbol{W}_k)\}_{k\in\mathbb{N}}$ is convergent. □

From the derivation of Theorem 7.1 (P1), the convergence of Algorithm 7.1 still holds if the chosen stepsize satisfies (7.16) such that one can use a backtracking method (shown in Algorithm 7.2) to choose η in practice. Note that Theorem 7.1 implies that the sequence generated by Algorithm 7.1 has at least one convergent subsequence and the limit point of any convergent subsequence is a stationary point of $\rho(\boldsymbol{\Phi},\boldsymbol{G})$. Moreover, we will prove that the sequence generated by Algorithm 7.1 is a Cauchy sequence and thus the sequence itself is convergent and converges to a stationary point of $\rho(\boldsymbol{\Phi},\boldsymbol{G})$ in Algorithm 7.1.

Algorithm 7.2 Backtracking at kth Iteration

Input: Initial values: (η_0,γ,α) with $\gamma\in(0,1)$, $\alpha\in(0,1)$, and η_0 the initial guess stepsize.
Output: Stepsize η and update $\boldsymbol{\Phi}_{k+1}$.

$\eta\leftarrow\eta_0$.
$\boldsymbol{\Phi}_{k+1}\leftarrow\mathcal{P}_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_k-\eta\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_k))$.
while $\rho(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)-\rho(\boldsymbol{\Phi}_{k+1},\boldsymbol{G}_k)<\frac{\gamma}{2\eta}\|\boldsymbol{\Phi}_{k+1}-\boldsymbol{\Phi}_k\|_F^2$ **do**
  $\eta\leftarrow\alpha\eta$.
  $\boldsymbol{\Phi}_{k+1}\leftarrow\mathcal{P}_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_k-\eta\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{G}_k))$.
end while

Theorem 7.2 (Sequence convergence)**.** *The sequence of iterates $\{(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)\}_{k\geq0}$ generated by Algorithm 7.1 with stepsize $\eta<\frac{1}{L_c}$ converges to a stationary point of ρ.*

Proof. We begin the proof with the definition of the Kurdyka-Lojasiewicz (KL) inequality which will be used in the following proof [147, 149, 148].

Definition 7.4.3 (Kurdyka-Lojasiewicz (KL) inequality)**.** Assume that $\sigma(\boldsymbol{u})$ is a proper semi-continuous function. Then $\exists\ \delta>0,\ \alpha\in[0,1),\ C_1>0$ yields

$$|\sigma(\boldsymbol{u})-\sigma(\underline{\boldsymbol{u}})|^\alpha\leq C_1\|\boldsymbol{v}\|,\ \ \forall\ \boldsymbol{u}\in B(\underline{\boldsymbol{u}},\delta),\ \forall\ \boldsymbol{v}\in\partial\sigma(\boldsymbol{u}),$$

where $\underline{\boldsymbol{u}}$ is a stationary point of $\sigma(\boldsymbol{u})$ and $B(\underline{\boldsymbol{u}},\delta)$ defines to a ball with $\underline{\boldsymbol{u}}$ and δ as the center and radius.

Note that $\rho(\boldsymbol{\Phi},\boldsymbol{G})$ is lower semi-continuous and satisfies the KL inequality. Theorem 7.1 reveals the subsequential convergence of the sequence $\{\boldsymbol{W}_k=(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)\}_k$. Now we show the sequence $\{\boldsymbol{W}_k=(\boldsymbol{\Phi}_k,\boldsymbol{G}_k)\}_k$ itself is convergent and hence it converges to a certain stationary point of $\underline{\boldsymbol{W}}=(\underline{\boldsymbol{\Phi}},\underline{\boldsymbol{G}})$.

From (7.18), we know that there exists an integer n so that $\boldsymbol{W}_k\in B(\underline{\boldsymbol{W}},\delta),\ \forall\ k>n,\ \forall\delta>0$ for some stationary point $\underline{\boldsymbol{W}}\in\mathcal{C}(\rho)$ where $\mathcal{C}(\rho)$ denotes a set of stationary points of ρ. By using the definition of concave function and the concavity of $h(y)=y^{1-\alpha}$ with $y>0$, we have

$$[\rho(\boldsymbol{W}_{k+1})-\rho(\underline{\boldsymbol{W}})]^{1-\alpha}\leq[\rho(\boldsymbol{W}_k)-\rho(\underline{\boldsymbol{W}})]^{1-\alpha}+(1-\alpha)\frac{\rho(\boldsymbol{W}_{k+1})-\rho(\boldsymbol{W}_k)}{[\rho(\boldsymbol{W}_k)-\rho(\underline{\boldsymbol{W}})]^\alpha}.\tag{7.28}$$

Now we show the lower and upper bound for $\rho(\boldsymbol{W}_k)-\rho(\boldsymbol{W}_{k+1})$ and $[\rho(\boldsymbol{W}_k)-\rho(\underline{\boldsymbol{W}})]^\alpha$, respectively. Following (7.16), (7.27), and the KL inequality, we get $\rho(\boldsymbol{W}_k)-\rho(\boldsymbol{W}_{k+1})\geq C_2\|\boldsymbol{W}_{k+1}-\boldsymbol{W}_k\|_F^2$ with $C_2=\min\{\frac{\frac{1}{\eta}-L_c}{2},1\}$ and $[\rho(\boldsymbol{W}_k)-\rho(\underline{\boldsymbol{W}})]^\alpha\leq C_1\|(\boldsymbol{D}_{\boldsymbol{\Phi}_k},\boldsymbol{D}_{\boldsymbol{G}_k})\|_F\leq C_3\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F$ with $C_3=C_1(2L_c+1/\eta)$. Plugging these two inequalities into (7.28), we obtain

$$[\rho(\boldsymbol{W}_k)-\rho(\underline{\boldsymbol{W}})]^{1-\alpha}-[\rho(\boldsymbol{W}_{k+1})-\rho(\underline{\boldsymbol{W}})]^{1-\alpha}\geq(1-\alpha)\frac{C_2\|\boldsymbol{W}_{k+1}-\boldsymbol{W}_k\|_F^2}{C_3\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F}. \tag{7.29}$$

Defining $C_4=(1-\alpha)C_2/C_3$ and summing (7.29) from $k=1$ to ∞, we get

$$\begin{aligned}
&\frac{1}{C_4}[\rho(\boldsymbol{W}_1)-\rho(\underline{\boldsymbol{W}})]^{1-\alpha}-\frac{1}{C_4}[\rho(\boldsymbol{W}_\infty)-\rho(\underline{\boldsymbol{W}})]^{1-\alpha}\\
&\geq\sum_{k=1}^{\infty}\frac{\|\boldsymbol{W}_{k+1}-\boldsymbol{W}_k\|_F^2}{\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F}+\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F\\
&\quad-\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F\\
&\overset{a^2+b^2\geq 2ab}{\geq}2\sum_{k=1}^{\infty}\|\boldsymbol{W}_{k+1}-\boldsymbol{W}_k\|_F-\sum_{k=1}^{\infty}\|\boldsymbol{W}_k-\boldsymbol{W}_{k-1}\|_F\\
&=\sum_{k=1}^{\infty}\|\boldsymbol{W}_{k+1}-\boldsymbol{W}_k\|_F-\|\boldsymbol{W}_1-\boldsymbol{W}_0\|_F.
\end{aligned}$$

With the above result and the boundedness of $\{\boldsymbol{W}_k\}_k$ and (7.18): $\sum_{k=1}^{\infty}\|\boldsymbol{W}_{k+1}-\boldsymbol{W}_k\|_F<\infty$, we know $\{\boldsymbol{W}_k\}_{k\geq 0}$ is Cauchy [148] in a compact set and hence it is convergent. □

Remark 7.1. The KL inequality is introduced to prove Theorem 7.2. We note that the KL inequality is also used to prove the convergence of proximal alternating minimization algorithms [147, 149, 148]. Algorithm 7.1 differs from the proximal alternating minimization algorithms [147, 149, 148] in that we update $\boldsymbol{G}$ (see (7.11)) by exactly minimizing the objective function rather than using proximal operators. We believe that the proof techniques for Theorems 7.1 and 7.2 may also be useful for analyzing the convergence of other algorithms for designing sensing matrices [102, 107, 128, 103, 104, 114].

For $\xi=0$, $\mathcal{G}_\xi=\{\boldsymbol{I}_L\}$, (7.14) is equivalent to

$$\min_{\boldsymbol{\Phi}}\nu(\boldsymbol{\Phi}):=\|\boldsymbol{I}_L-\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Psi}\|_F^2+\lambda\|\boldsymbol{\Phi}\|_F^2+\delta_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}). \tag{7.30}$$

Evidently, Algorithm 7.1 amounts to PGD, known as the iterative hard thresholding (IHT) algorithm for compressive sensing [150]. A direct consequence of Theorems 7.1 and 7.2 is the following corollary that establishes the convergence of PGD for (7.30).

Corollary 7.3. *Let $\{\boldsymbol{\Phi}_k\}_{k\geq 0}$ be the sequence generated by the PGD method with a constant stepsize $\eta<\frac{1}{L_c}$ that $\boldsymbol{\Phi}_{k+1}=\mathcal{P}_{\mathcal{S}_\kappa}(\boldsymbol{\Phi}_k-\eta\nabla_{\boldsymbol{\Phi}}f(\boldsymbol{\Phi}_k,\boldsymbol{I}))$.*

Then

(1) $\nu(\boldsymbol{\Phi}_k)-\nu(\boldsymbol{\Phi}_{k+1})\geq\frac{\frac{1}{\eta}-L_c}{2}\|\boldsymbol{\Phi}_k-\boldsymbol{\Phi}_{k+1}\|_F^2$.

(2) the sequence $\{\nu(\boldsymbol{\Phi}_k)\}_{k\geq 0}$ converges.

(3) the sequence $\{\boldsymbol{\Phi}_k\}_{k\geq 0}$ converges to a stationary point of ν.

Remark 7.2. We note that Corollary 7.3 is valid for applying PGD to a general sparsity-constrained problem if the corresponding objective function is Lipschitz continuous. Compared with [151, Theorem 3.1] which also provides the convergence analysis of PGD for a general sparsity-constrained problem, Corollary 7.3 shows that the sequence generated by PGD is also convergent and converges to a stationary point, while [151, Theorem 3.1] only showed the subsequential convergence property of PGD, i.e., the limit point of any convergent subsequence converges to a stationary point.

We note that one can also add the sparse constraint to the entire sensing matrix rather than each row and Algorithm 7.1 is still valid for such a constraint. However, we empirically observe that such a sensing matrix yields slightly worse performance than the one imposing sparsity in each row. Moreover, we do not choose to impose the sparse constraint in each column because M is much smaller than N and such a constraint reduces the freedom of optimizing $\boldsymbol{\Phi}$. For example, if $M = 10$ and $N = 100$ and the sensing matrix can only have 10% non-zeros, then each column will have one non-zero if we impose the sparsity in each column, whereas each row can have ten non-zeros by imposing the sparsity in each row, which provides more freedom to optimize the sensing matrix.

7.5 Numerical Experiments

A series of experiments on synthetic data and real images is carried out in this section to study the performance of the proposed method for designing structured sparse sensing matrix. Moreover, we also compare our method with previous methods for designing sensing matrices [138, 103, 106]. Following, we list all CS systems that are tested in this chapter.

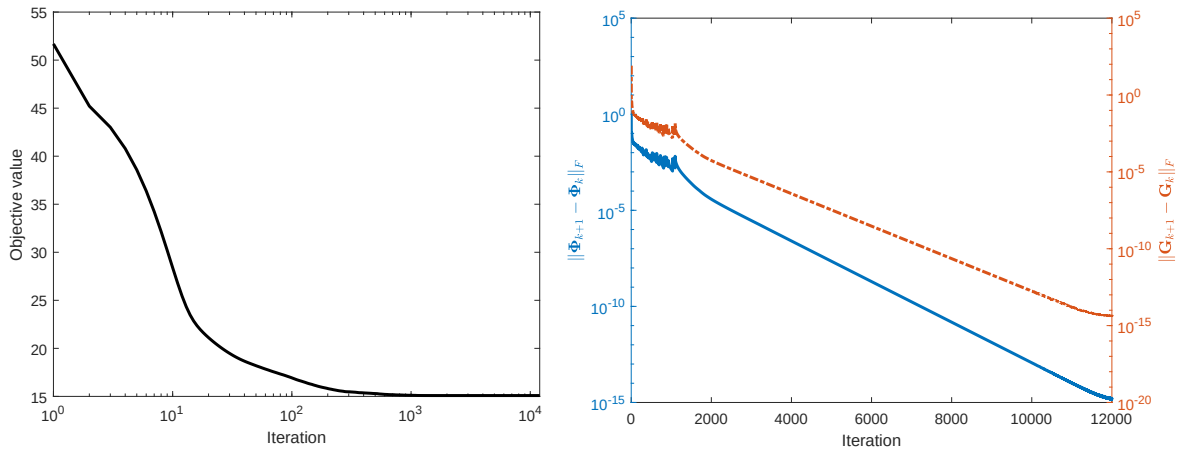
CS_{randn}:	$\boldsymbol{\Psi}$ + A dense random matrix.
CS_{MT}:	$\boldsymbol{\Psi}$ + Sensing matrix [106].
CS_{MT-ETF}:	$\boldsymbol{\Psi}$ + Sensing matrix [106].
CS_{LZYCB}:	$\boldsymbol{\Psi}$ + Sensing matrix [103].
CS_{bispar}:	$\boldsymbol{\Psi}$ + A binary sparse sensing matrix [138].
$\text{CS}_{sparse-A}$:	$\boldsymbol{\Psi}$ + Output of Algorithm 7.1 with $\xi = 0$ (i.e., $\mathcal{G}_\xi = \{\boldsymbol{I}_L\}$) and $\boldsymbol{A} = \text{DCT}$.
CS_{sparse}:	$\boldsymbol{\Psi}$ + Output of Algorithm 7.1 with $\xi = 0$ (i.e., $\mathcal{G}_\xi = \{\boldsymbol{I}_L\}$) and $\boldsymbol{A} = \boldsymbol{I}_N$.
$\text{CS}_{sparse-ETF}$:	$\boldsymbol{\Psi}$ + Output of Algorithm 7.1 with $\xi = \underline{\mu}$ and $\boldsymbol{A} = \boldsymbol{I}_N$.

7.5.1 Synthetic Data

A dictionary $\boldsymbol{\Psi} \in \mathbb{R}^{N\times L}$ with normally distributed entries and a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ are generated for CS_{randn}. The training and testing data are generated as follows: for a given dictionary $\boldsymbol{\Psi}$, we first generate a set of I K-sparse vectors $\{\boldsymbol{\theta}_i \in \mathbb{R}^L\}_{i=1}^I$ where the index of the non-zeros in $\boldsymbol{\theta}_i$ obeys a normal distribution and then obtain the sparse signals $\{\boldsymbol{x}_i\}_{i=1}^I$ through

$$\boldsymbol{x}_i = \boldsymbol{\Psi}\boldsymbol{\theta}_i + \boldsymbol{e}_i, \tag{7.31}$$

where $\boldsymbol{e}_i$ refers to the Gaussian noise with mean zero and variance σ^2. Denote by SNR the signal-to-noise ratio (in dB) of the signals in (7.31). The performance of a CS system is evaluated through the mean



(a) (b)

Figure 7.3: Examination of the convergence of Algorithm 7.1. (a) The change of $f(\boldsymbol{\Phi}_k, \boldsymbol{G}_k)$; (b) The change of $\|\boldsymbol{\Phi}_{k+1} - \boldsymbol{\Phi}_k\|_F$ (blue line) and $\|\boldsymbol{G}_{k+1} - \boldsymbol{G}_k\|_F$ (red line). The parameters are $M = 25, N = 60, L = 80, \lambda = 0.25$, and $\kappa = 20$.

squared error (MSE)

$$\text{MSE} \triangleq \frac{1}{N \times I} \sum_{i=1}^{I} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2, \tag{7.32}$$

where $\hat{\boldsymbol{x}}_i = \boldsymbol{\Psi}\hat{\boldsymbol{\theta}}_i$ is the recovered signal and $\hat{\boldsymbol{\theta}}_i$ is obtained through

$$\hat{\boldsymbol{\theta}}_i = \arg\min_{\boldsymbol{\theta}_i} ||\boldsymbol{\Phi}\boldsymbol{x}_i - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}_i||_2^2 \quad \text{s.t.} \quad ||\boldsymbol{\theta}_i||_0 \leq K.$$

Now we examine the convergence of Algorithm 7.1. The generated dictionary $\boldsymbol{\Psi}$ and $\boldsymbol{A} = \boldsymbol{I}_N$ are chosen for Algorithm 7.1 and also adopted in the following synthetic experiments. From Figure 7.3, we see $f(\boldsymbol{\Phi}_k, \boldsymbol{G}_k)$ decays steadily and $\|\boldsymbol{\Phi}_{k+1} - \boldsymbol{\Phi}_k\|_F$ and $\|\boldsymbol{G}_{k+1} - \boldsymbol{G}_k\|_F$ tend to 0 linearly meeting the theoretical analysis. Next we study the choice of λ which is used to balance the importance of mutual coherence and the robustness of SRE. As seen from Figure 7.4, the optimal λ for which the corresponding sensing matrix yields the highest recovery accuracy becomes large for a lower SNR, showing the importance of emphasizing robustness with respect to noise.

In Figure 7.5, we compare the performance of each CS system with varying SNR. From Figure 7.5, we see that CS_{MT}, CS_{MT-ETF}, CS_{sparse}, and $\text{CS}_{sparse-ETF}$ outperform other CS systems for SNR < 25dB. Comparing CS_{MT-ETF} and $\text{CS}_{sparse-ETF}$ with CS_{MT} and CS_{sparse} for a relatively high SNR, we observe that setting $\xi = \underline{\mu}$ is more attractive than $\xi = 0$. Notice that the performance of CS_{LYZCB} deteriorates fast when we reduce SNR implying that CS_{LYZCB} is not robust with respect to SRE. It is interesting to note that CS_{sparse} and $\text{CS}_{sparse-ETF}$ yield performance that is comparable to that of CS_{MT} and CS_{MT-ETF} and are much better than CS_{randn} and CS_{bispar}, suggesting that using a sparse sensing matrix does not sacrifice the performance. Finally, we study the change in signal recovery accuracy for varying M and K. From Figures 7.6 and 7.7, we observe that the sparse sensing matrix works similarly to the dense one for varying M and K.

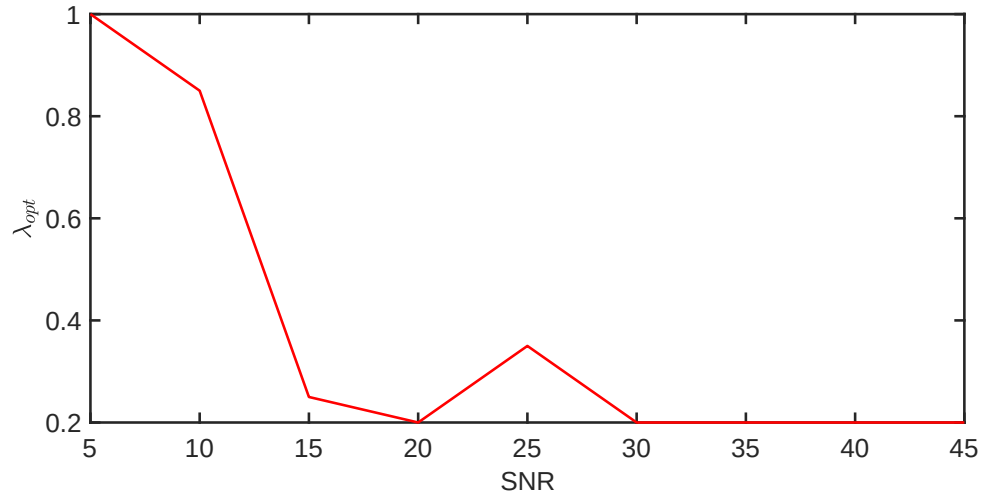


Figure 7.4: Optimal λ with CS_{sparse} for varying SNR. The parameters are $M = 25, N = 60, L = 80, K = 4, I = 2000$, and $\kappa = 20$.

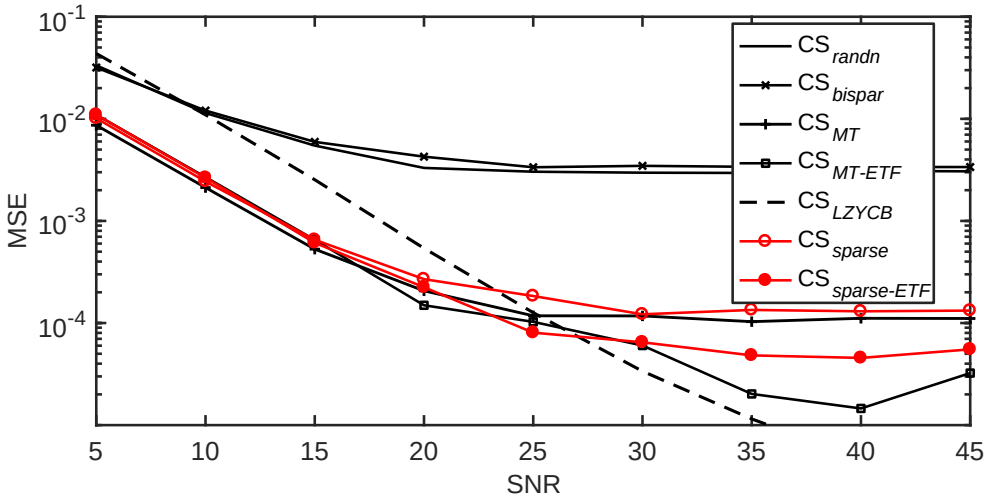


Figure 7.5: Comparison of MSE with varying SNR (dB) for different CS systems. The parameters are $M = 25, N = 60, L = 80, K = 4, I = 2000, \lambda = 0.25$, and $\kappa = 20$. Disappearance from this figure means that MSE is less than 10^{-5}.

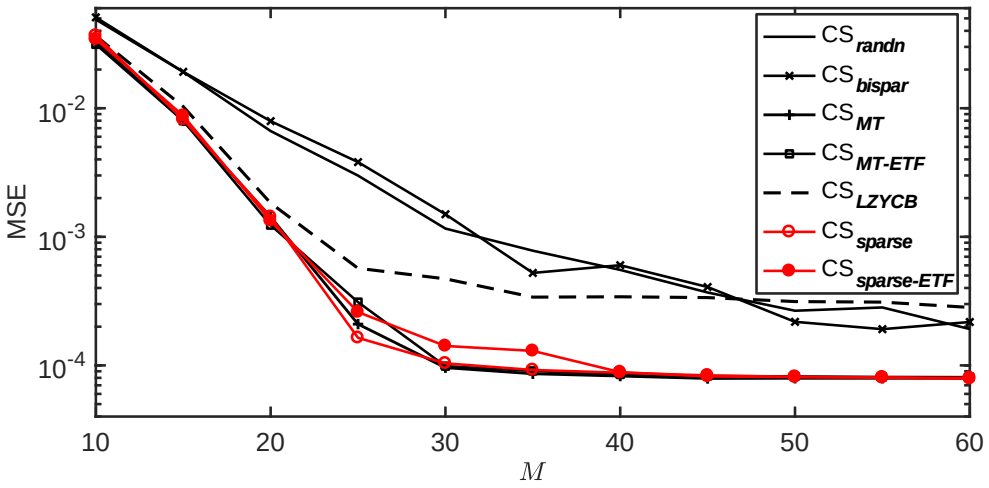


Figure 7.6: Comparison of MSE with varying M for SNR = 20dB. The parameters are $N = 60, L = 80, K = 4, I = 2000, \lambda = 0.25$, and $\kappa = 20$.

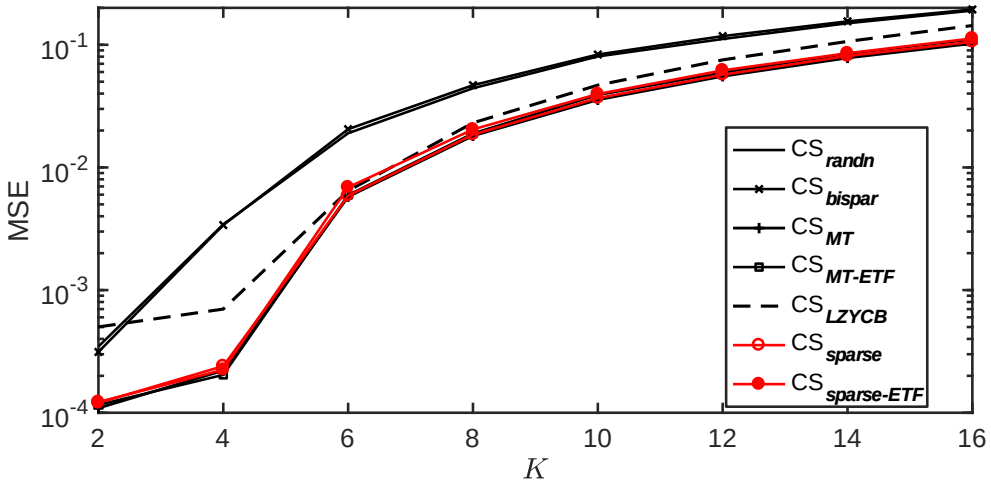


Figure 7.7: Comparison of MSE with different K for SNR $=$ 20dB. The parameters are $M = 25, N = 60, L = 80, I = 2000, \lambda = 0.25$, and $\kappa = 20$.

7.5.2 Real Images

In this subsection, we study the performance of the aforementioned CS systems for the reconstruction of real images with two different sizes of dictionaries, $\boldsymbol{\Psi} \in \mathbb{R}^{64\times 100}$ (low dimensional dictionary) and $\boldsymbol{\Psi} \in \mathbb{R}^{256\times 800}$ (high dimensional dictionary). The low dimensional dictionary $\boldsymbol{\Psi}$ is learnt using the KSVD algorithm [17] with a set of 8×8 non-overlapping patches by extracting randomly 15 patches from each of 400 images in the LabelMe [123] training dataset. The extracted 8×8 patches are re-arranged as a vector, and eventually we obtain a total of 64×6000 signals for training. Compared with the low dimensional dictionary, the high dimensional dictionary gives more freedom to learn, such that we extract more patches from the LabelMe training dataset resulting in 10^6 signals. To address such a large training dataset, we use the online dictionary learning algorithm [109] to learn the high dimensional dictionary. The reconstruction accuracy of each CS system for real images is evaluated by Peak Signal to Noise Ratio (PSNR),

$$\text{PSNR} \triangleq 10\times \log 10\left[\frac{(2^r-1)^2}{\text{MSE}}\right](\text{dB}),$$

where $r = 8$ and MSE is defined in (7.32).

In Figure 7.8, we show the comparison of PSNR for different κ (the number of non-zeros in each row of the sparse sensing matrix) on testing image "Lena", and the values of PSNR for other testing images (i.e., Couple, Barbara, Child, Plane, and Man) in Tables 7.1 and 7.2. The reconstructed testing image "Couple" is displayed in Figure 7.10 for showing the visual effect. From Figure 7.8 and Tables 7.1 and 7.2, we observe that even for a small κ, $\text{CS}_{sparse-A}$ ($\boldsymbol{A} = \text{DCT}$) has better performance than CS_{sparse} ($\boldsymbol{A} = \boldsymbol{I}_N$) implying the effectiveness of using the auxiliary DCT matrix to improve the reconstruction accuracy. It is not surprising to note that CS_{LZYCB} yields very low PSNR for real image experiments, consistent with Figure 7.5, indicating that CS_{LZYCB} is not robust with respect to $\boldsymbol{e} \neq 0$. As expected, $\text{CS}_{sparse-A}$ and CS_{sparse} yield higher PSNR for a larger κ. It is interesting to note that $\text{CS}_{sparse-A}$ is only 0.53dB inferior to CS_{MT} even for a small κ (e.g., $\kappa = 10$) and is still more than 3dB better than CS_{randn} which is dense. We note that the difference between $\text{CS}_{sparse-A}$ and CS_{MT} is almost negligible (0.15 dB) for $\kappa \geq 30$ meeting our expectation that one can design a sparse sensing matrix instead of a dense

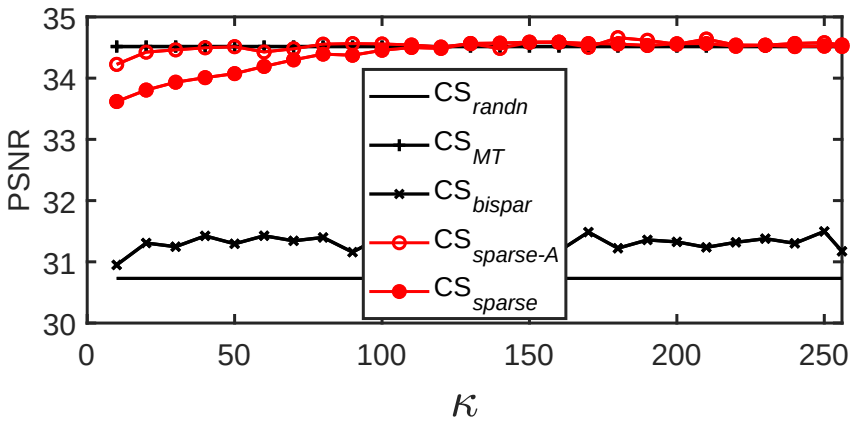


Figure 7.8: Comparison of PSNR(dB) with different κ for each CS system before post-processing on the testing image "Lena". The parameters are $M = 80, N = 256, L = 800, K = 16, \lambda = 0.5$.

Figure 7.9: The original testing image "Couple".

one with comparable performance, so that we can improve the efficiency of acquisition. Comparing the results in Tables 7.1 and 7.2, we observe that one can obtain a higher PSNR by using a high dimensional dictionary. It is interesting to note that the proposed sparse sensing matrix becomes more attractive for a high dimensional dictionary because the reduced acquisition costs become more significant for such high dimensional signals.

Since the patch-based image processing techniques will introduce the blockiness artifacts, we suggest using the deblocking techniques in this part as a post-processing step to reduce such artifacts. To this end, we additionally call the BM3D denoising algorithm for the reconstructed images [20]. From Tables 7.1 and 7.2 and Figure 7.10, we observe that such a post-processing step not only improves the visual effect, but also increases PSNR for each CS system.

7.6 Conclusion

A new framework for designing a structured sparse sensing matrix which is robust with respect to the sparse representation error and can be used to efficiently acquire the signals of interest is introduced in this chapter. An alternating minimization algorithm with analysis of convergence is proposed to solve the optimization problem. Numerical experiments show the promising performance of the proposed

Table 7.1: Comparison of PSNR (dB) for six testing images with $M = 20$, $N = 64$, $L = 100$, $K = 4$, $\lambda = 1.4$ for different κ. First row: PSNR before post-processing; Second row: the increased PSNR after post-processing.

	Lena		Couple		Barbara		Child		Plane		Man	
	$\kappa = 10$	$\kappa = 20$	$\kappa = 10$	$\kappa = 20$	$\kappa = 10$	$\kappa = 20$	$\kappa = 10$	$\kappa = 20$	$\kappa = 10$	$\kappa = 20$	$\kappa = 10$	$\kappa = 20$
CS_{randn}	29.69		27.01		22.44		31.20		28.57		27.41	
	+1.12		+0.95		+0.52		+1.27		+1.08		+1.04	
CS_{MT}	32.75		30.01		25.36		34.22		31.60		30.39	
	+1.31		+1.05		+0.45		+1.72		+0.78		+0.32	
CS_{LZYCB}	12.74		10.38		4.19		15.93		14.51		9.75	
	+1.58		+2.08		+7.32		+1.76		+1.33		+3.20	
CS_{bispar}	29.36	29.27	26.85	26.87	22.42	22.49	30.80	30.87	28.13	28.28	27.19	27.14
	+1.23	+1.24	+0.97	+1.03	+0.65	+0.61	+1.56	+1.46	+0.78	+0.90	+1.07	+1.09
$CS_{sparse-A}$	32.38	32.65	29.63	29.88	24.91	25.19	33.84	34.12	31.28	31.52	30.01	30.27
	+1.86	+1.34	+1.31	+1.07	+1.04	+0.47	+2.03	+1.70	+0.85	+1.64	+0.44	+0.38
CS_{sparse}	32.26	32.56	29.47	29.79	24.76	25.11	33.75	34.07	31.15	31.48	29.83	30.15
	+1.26	+1.33	+1.01	+1.02	+0.54	+0.47	+1.71	+1.68	+0.86	+1.21	+0.61	+0.40

Table 7.2: Similarly to Table 7.1, but with $M = 80$, $N = 256$, $L = 800$, $K = 16$, $\lambda = 0.5$.

	Lena		Couple		Barbara		Child		Plane		Man	
	$\kappa = 10$	$\kappa = 30$	$\kappa = 10$	$\kappa = 30$	$\kappa = 10$	$\kappa = 30$	$\kappa = 10$	$\kappa = 30$	$\kappa = 10$	$\kappa = 30$	$\kappa = 10$	$\kappa = 30$
CS_{randn}	30.73		27.40		22.76		31.99		29.57		27.94	
	+0.69		+0.51		+0.21		+0.59		+0.48		+0.48	
CS_{MT}	34.38		30.90		26.04		35.60		33.34		31.50	
	+0.14		+0.23		+0.13		+0		+0.24		+0.20	
CS_{bispar}	30.23	30.72	27.33	27.36	22.57	22.64	31.70	31.94	29.29	29.49	27.59	27.82
	+0.67	+0.67	+0.50	+0.52	+0.21	+0.20	+0.58	+0.59	+0.48	+0.48	+0.46	+0.47
$\mathrm{CS}_{sparse-A}$	33.89	34.24	30.47	30.75	25.44	25.82	35.09	35.36	32.92	33.15	30.95	31.22
	+0.34	+0.22	+0.37	+0.29	+0.18	+0.16	+0.25	+0.08	+0.38	+0.29	+0.35	+0.27
CS_{sparse}	33.17	33.50	29.69	29.94	24.26	24.60	34.38	34.66	32.41	32.61	30.06	30.39
	+0.45	+0.43	+0.42	+0.40	+0.15	+0.16	+0.24	+0.21	+0.38	+0.38	+0.35	+0.36

(a) CS_{randn}: 27.40 (b) CS_{bispar}: 27.33 (c) CS_{sparse}: 29.69 (d) $\mathrm{CS}_{sparse-A}$: 30.47 (e) CS_{MT}: 30.90

(f) +0.51 (g) +0.50 (h) +0.42 (i) +0.37 (j) +0.23

Figure 7.10: Comparison of PSNR (dB) of reconstructed "Couple" images for each CS system with $M = 80,\ N = 256,\ L = 800,\ K = 16,\ \lambda = 0.5, \kappa = 10$. Upper: PSNR before post-processing; Bottom: Improved PSNR after post-processing.

structured sparse sensing matrix in terms of signal recovery accuracy on synthetic data and real images.

As shown in Section 7.5, the use of the base matrix $\boldsymbol{A}$ improves the performance of the sparse sensing matrix, especially for the case that the number of non-zeros in $\bar{\boldsymbol{\Phi}}$ is very small. Moreover, the use of a base matrix $\boldsymbol{A}$ may become more attractive if it also has some freedom for optimizing so that it is possible to optimize the base matrix $\boldsymbol{A}$ and the sparse matrix $\bar{\boldsymbol{\Phi}}$ simultaneously. One possible direction is to use a series of Givens rotation matrices as $\boldsymbol{A}$ which provides freedom to optimize and can be implemented efficiently. Moreover, adapting the optimized structured sparse sensing matrix into analog-to-digital conversion systems based on compressive sensing [152, 153] would be another interesting direction for the future. To this end, it is important to develop a quantized sparse sensing matrix. We finally note that it remains an open problem to certify certain properties (e.g., RIP) for the optimized sensing matrices [102, 107, 128, 103, 105, 104, 108, 114, 106, 129, 120] which empirically outperforms a random one that satisfies the RIP. Works in these directions are ongoing.

Chapter 8

Conclusion and Future Work

In this dissertation, we have studied numerical optimization and multigrid computational methods, together with applications. This research can be divided into three parts, with contributions summarized as follows:

1. **Efficient Solvers for Regularization by Denoising.** We begin with the application of vector extrapolation (VE) towards accelerating existing solvers for Regularization by Denoising (RED). VE can significantly reduce the number of iterations required for convergence, and the additional computation at each step is small, so we obtain the acceleration almost free. In further work, we introduce a general scheme called weighted proximal methods (WPMs) for RED and show that two previous methods, namely, the fixed point and accelerated proximal method, are two special cases of WPMs. By choosing a slightly more elaborate weighting, the resulting algorithm converges faster than all previous solvers for RED.

2. **Acceleration Schemes.** We adapt Nesterov's scheme to accelerating iterative methods for linear problems. When the iteration matrix of the iterative method has only real eigenvalues, we propose an explicit formula to optimally calculate the parameter of Nesterov's scheme. Moreover, we study the robustness of Nesterov's scheme for the case where the iteration matrix also has complex eigenvalues. In subsequent work, we merge multigrid (MG) methods with sequential subspace optimization (SESOP) yielding a hybrid scheme for acceleration, dubbed SESOP-MG, which inherits the merits of SESOP and multigrid methods. The asymptotic convergence factor of the two-grid version of SESOP-MG is studied for quadratic problems with three search directions, namely, preconditioned gradient, one history (previous iterate), and coarse-grid correction. We propose a fixed-stepsize version for quadratic problems to avoid the calculation of stepsizes at each iteration and study two heuristic ways to estimate the optimal fixed stepsizes cheaply.

3. **The Design of Robust Compressive Sensing Systems.** We introduce a new model to learn the sensing matrix and sparsifying dictionary simultaneously for compressive sensing (CS) systems. The resulting CS system is robust with respect to the case where the signals of interest have a sparse representation error. Moreover, we also propose an efficient online algorithm with convergence analysis to solve the formulated model. Our second work in this part considers the efficiency of acquiring a signal. We propose a structured sensing matrix consisting of a sparse matrix and a prescribed dense matrix which allows fast implementation. Moreover, we introduce a numerical method with the guarantee of global sequence convergence to solve the problem.

8.1 Future Work

Having shown the capabilities of some computational methods for applications in signal processing and computational imaging, we next list possible directions for future research, from both practical and theoretical aspects. Some of these are a direct extension of the work reported in this thesis.

- We have studied here the adaption of Nesterov's scheme to acceleration of iterative methods for linear problems. A natural question is how to adapt these ideas to nonlinear problems. Moreover, even for linear problems, we do not yet know how to accelerate an iterative method whose iteration matrix contains complex eigenvalues beyond the case we discussed in Chapter 4. Another natural research direction is an attempt to extend our results to the case of iteration-dependent coefficients (as in Chebyshev acceleration).

- In nonlinear MG methods, we use the full approximation storage (FAS) scheme, which uses only first-order information. The merits of using second-order information in nonlinear MG were introduced in [154]. However, this approach has not been applied broadly, due to technical challenges in making it robust and efficient. Hence, developing a workable scheme that involves high-order information in nonlinear MG methods is an interesting future direction. One may consider the convex problems as candidate problems first that we may be able to use the well-developed theoretical results of convex optimization to understand the essential difficulties. One interesting direction may be adapting simple approximations to the Hessian matrix, as done in WPM's in the thesis.

- Inverse problems can be modelled as the following composite minimization problems:

$$\min_{\boldsymbol{x}} \quad \underbrace{F(\boldsymbol{x})}_{\text{Data Fidelity}} \quad + \underbrace{R(\boldsymbol{x})}_{\text{Prior}}, \tag{8.1}$$

  where $F(\boldsymbol{x}) = \sum_{\ell=1}^{L} f_\ell(\boldsymbol{x}, \boldsymbol{y}_\ell)$ with $\boldsymbol{y}_\ell$ the ℓth measurement and $L \geq 1$ the number of measurements. For many applications [155, 156], the evaluation of the gradient of $f_\ell(\boldsymbol{x}, \boldsymbol{y}_\ell)$ is expensive, which is similar to the situation in RED when we evaluate the gradient of the prior part. Moreover, L is typically large and $R(\boldsymbol{x})$ is nonsmooth, e.g., total variation regularization. Our plan is to extend the weighted proximal methods for such a class of problems. However, we have to address two main issues: (i) find a way to estimate weighting online because of large L; (ii) address the difficulty of the nonsmooth prior. In this direction, some preliminary results of our ongoing work already show that such an extension is promising.

- For many nonlinear inverse problems [157, 155, 156], the optimization problems formulated in (8.1) are nonconvex and there exist many local minima which pose difficulties for reconstructing the signals of interest. Due to the rapid developments of theoretical research in nonconvex optimization community, it would be interesting to understand the difficulties of the reconstruction in nonlinear inverse problems and study the possibility of developing efficient methods with theoretical guarantee from the optimization viewpoint.

Bibliography

[1] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[2] S. Mallat, *A wavelet tour of signal processing*. Elsevier, 1999.

[3] I. Tošić and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.

[4] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2305–2318, 2018.

[5] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1804–1844, 2017.

[6] T. Hong, Y. Romano, and M. Elad, "Acceleration of RED via vector extrapolation," *Journal of Visual Communication and Image Representation*, vol. 63, p. 102575, 2019.

[7] T. Hong, I. Yavneh, and M. Zibulevsky, "Solving RED with weighted proximal methods," *IEEE Signal Processing Letters*, vol. 27, pp. 501–505, 2020.

[8] T. Hong and I. Yavneh, "On adapting Nesterov's scheme to accelerate iterative methods for linear problems," *arXiv preprint arXiv:2102.09239*, 2021.

[9] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, 1977.

[10] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*. SIAM, second ed., 2000.

[11] K. Stüben, "Algebraic multigrid (AMG): An introduction with applications," in *Multigrid* (U. Trottenberg, C. W. Oosterlee, and A. Schuller, eds.), Academic Press, 2001.

[12] J. Xu and L. Zikatanov, "Algebraic multigrid methods," *Acta Numerica*, vol. 26, pp. 591–721, 2017.

[13] C. W. Oosterlee and T. Washio, "Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows," *SIAM Journal on Scientific Computing*, vol. 21, no. 5, pp. 1670–1690, 2000.

[14] T. Hong, I. Yavneh, and M. Zibulevsky, "Accelerating multigrid optimization via SESOP," *arXiv preprint arXiv:1812.06896*, 2018.

[15] M. Zibulevsky, "Speeding-up convergence via sequential subspace optimization: Current state and future directions," *arXiv preprint arXiv:1401.0159*, 2013.

[16] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[17] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[18] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 60–65, IEEE, 2005.

[19] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[20] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[21] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *International Conference on Computer Vision (ICCV)*, pp. 479–486, IEEE, 2011.

[22] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1620–1630, 2013.

[23] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017.

[24] P. Chatterjee and P. Milanfar, "Is denoising dead?," *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 895–911, 2010.

[25] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 106–128, 2013.

[26] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2833–2840, IEEE, 2011.

[27] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the nonlocal-means to super-resolution reconstruction," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 36–51, 2009.

[28] A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1715–1728, 2012.

[29] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "Optimal recovery from compressive measurements via denoising-based approximate message passing," in *International Conference on Sampling Theory and Applications (SampTA)*, pp. 508–512, IEEE, 2015.

[30] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 945–948, IEEE, 2013.

[31] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman, "Plug-and-play priors for bright field electron tomography and sparse interpolation," *IEEE Transactions on Computational Imaging*, vol. 2, no. 4, pp. 408–423, 2016.

[32] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2017.

[33] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[34] S. Cabay and L. Jackson, "A polynomial extrapolation method for finding limits and antilimits of vector sequences," *SIAM Journal on Numerical Analysis*, vol. 13, no. 5, pp. 734–752, 1976.

[35] N. Rajeevan, K. Rajgopal, and G. Krishna, "Vector-extrapolated fast maximum likelihood estimation algorithms for emission tomography," *IEEE Transactions on Medical Imaging*, vol. 11, no. 1, pp. 9–20, 1992.

[36] A. Sidi, *Vector extrapolation methods with applications*. SIAM, 2017.

[37] A. Kirsch, *An introduction to the mathematical theory of inverse problems*, vol. 120. Springer Nature, 2021.

[38] A. Sidi, "SVD-MPE: An SVD-based vector extrapolation method of polynomial type," *Applied Mathematics*, vol. 7, no. 11, pp. 1260–1278, 2016.

[39] D. A. Smith, W. F. Ford, and A. Sidi, "Extrapolation methods for vector sequences," *SIAM Review*, vol. 29, no. 2, pp. 199–233, 1987.

[40] G. Rosman, L. Dascal, A. Sidi, and R. Kimmel, "Efficient Beltrami image filtering via vector extrapolation methods," *SIAM Journal on Imaging Sciences*, vol. 2, no. 3, pp. 858–878, 2009.

[41] A. Sidi, "Efficient implementation of minimal polynomial and reduced rank extrapolation methods," *Journal of Computational and Applied Mathematics*, vol. 36, no. 3, pp. 305–337, 1991.

[42] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. Johns Hopkins University Press, 2012.

[43] A. Sidi, "Minimal polynomial and reduced rank extrapolation methods are related," *Advances in Computational Mathematics*, vol. 43, no. 1, pp. 151–170, 2017.

[44] A. Sidi and Y. Shapira, "Upper bounds for convergence rates of acceleration methods with initial iterations," *Numerical Algorithms*, vol. 18, no. 2, pp. 113–132, 1998.

[45] A. Sidi, W. F. Ford, and D. A. Smith, "Acceleration of convergence of vector sequences," *SIAM Journal on Numerical Analysis*, vol. 23, no. 1, pp. 178–196, 1986.

[46] A. Sidi and J. Bridger, "Convergence and stability analyses for some vector extrapolation methods in the presence of defective iteration matrices," *Journal of Computational and Applied Mathematics*, vol. 22, no. 1, pp. 35–61, 1988.

[47] A. Sidi, "Convergence and stability properties of minimal polynomial and reduced rank extrapolation algorithms," *SIAM Journal on Numerical Analysis*, vol. 23, no. 1, pp. 197–209, 1986.

[48] S. Skelboe, "Computation of the periodic steady-state response of nonlinear networks by extrapolation methods," *IEEE Transactions on Circuits and Systems*, vol. 27, no. 3, pp. 161–175, 1980.

[49] K. Jbilou and H. Sadok, "Some results about vector extrapolation methods and related fixed-point iterations," *Journal of Computational and Applied Mathematics*, vol. 36, no. 3, pp. 385–398, 1991.

[50] Y. Nesterov, *Lectures on convex optimization*, vol. 137. Springer, 2018.

[51] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[52] K. Gu, G. Zhai, X. Yang, and W. Zhang, "Using free energy principle for blind image quality assessment," *IEEE Transactions on Multimedia*, vol. 17, no. 1, pp. 50–63, 2015.

[53] K. Gu, G. Zhai, W. Lin, X. Yang, and W. Zhang, "No-reference image sharpness assessment in autoregressive parameter space," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3218–3231, 2015.

[54] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[55] S. Roy and A. Borzì, "A new optimization approach to sparse reconstruction of log-conductivity in acousto-electric tomography," *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 1759–1784, 2018.

[56] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Transactions on Computational Imaging*, vol. 5, no. 1, pp. 52–67, 2019.

[57] S. Becker and J. Fadili, "A quasi-Newton proximal splitting method," in *Advances in Neural Information Processing Systems*, pp. 2618–2626, 2012.

[58] S. Becker, J. Fadili, and P. Ochs, "On quasi-Newton forward-backward splitting: Proximal calculus and convergence," *SIAM Journal on Optimization*, vol. 29, no. 4, pp. 2445–2481, 2019.

[59] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[60] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 23, no. 2, pp. 517–541, 2001.

[61] A. V. Knyazev and K. Neymeyr, "Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block conjugate gradient method," *Electronic Transactions on Numerical Analysis*, vol. 15, pp. 38–55, 2003.

[62] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," in *Dokl. Akad. Nauk Sssr*, vol. 269, pp. 543–547, 1983.

[63] W. Su, S. P. Boyd, and E. J. Candès, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," in *Neural Information Processing Systems*, pp. 2510–2518, 2014.

[64] A. Wibisono, A. C. Wilson, and M. I. Jordan, "A variational perspective on accelerated methods in optimization," *Proceedings of the National Academy of Sciences*, vol. 113, no. 47, pp. E7351–E7358, 2016.

[65] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[66] Y. Nesterov and S. U. Stich, "Efficiency of the accelerated coordinate descent method on structured optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 110–123, 2017.

[67] D. Mitchell, N. Ye, and H. De Sterck, "Nesterov acceleration of alternating least squares for canonical tensor decomposition: Momentum step size selection and restart mechanisms," *Numerical Linear Algebra with Applications*, vol. 27, no. 4, p. e2297, 2020.

[68] Y. Nesterov, "Accelerating the cubic regularization of Newton's method on convex problems," *Mathematical Programming*, vol. 112, no. 1, pp. 159–181, 2008.

[69] A. Nitanda, "Stochastic proximal gradient descent with acceleration techniques," in *Advances in Neural Information Processing Systems*, pp. 1574–1582, 2014.

[70] H. De Sterck and Y. He, "On the asymptotic linear convergence speed of anderson acceleration, Nesterov acceleration, and nonlinear GMRES," *SIAM Journal on Scientific Computing*, no. 0, pp. S21–S46, 2021.

[71] L. A. Hageman and D. M. Young, *Applied iterative methods*. Courier Corporation, 2012.

[72] G. H. Golub and H. A. Van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1-2, pp. 35–65, 2000.

[73] I. Yavneh, "Why multigrid methods are so efficient," *Computing in Science & Engineering*, vol. 8, no. 6, pp. 12–22, 2006.

[74] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*. Academic Press, 2000.

[75] R. Wienands and W. Joppich, *Practical Fourier analysis for multigrid methods*. CRC press, 2004.

[76] I. Yavneh, "Multigrid smoothing factors for red-black Gauss–Seidel relaxation applied to a class of elliptic operators," *SIAM Journal on Numerical Analysis*, vol. 32, no. 4, pp. 1126–1138, 1995.

[77] I. Yavneh, "On red-black SOR smoothing in multigrid," *SIAM Journal on Scientific Computing*, vol. 17, no. 1, pp. 180–192, 1996.

[78] I. Yavneh and E. Olvovsky, "Multigrid smoothing for symmetric nine-point stencils," *Applied Mathematics and Computation*, vol. 92, no. 2, pp. 229–246, 1998.

[79] D. Greenfeld, M. Galun, R. Basri, I. Yavneh, and R. Kimmel, "Learning to optimize multigrid PDE solvers," in *International Conference on Machine Learning*, pp. 2415–2423, PMLR, 2019.

[80] J. Dendy, "Black box multigrid," *Journal of Computational Physics*, vol. 48, no. 3, pp. 366–386, 1982.

[81] D. Wang, Y. He, and H. De Sterck, "On the asymptotic linear convergence speed of Anderson acceleration applied to ADMM," *Journal of Scientific Computing*, vol. 88, no. 2, pp. 1–35, 2021.

[82] A. Brandt, "1984 multigrid guide with applications to fluid dynamics." Monograph, GMD-Studie 85, GMD-FIT, Postfach 1240, D-5205, St. Augustin 1, West Germany, 1985.

[83] A. Borzì and V. Schulz, "Multigrid methods for PDE optimization," *SIAM Review*, vol. 51, no. 2, pp. 361–395, 2009.

[84] S. G. Nash, "A multigrid approach to discretized optimization problems," *Optimization Methods and Software*, vol. 14, no. 1-2, pp. 99–116, 2000.

[85] R. M. Lewis and S. G. Nash, "Model problems for the multigrid optimization of systems governed by differential equations," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 1811–1837, 2005.

[86] Z. Wen and D. Goldfarb, "A line search multigrid method for large-scale nonlinear optimization," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1478–1503, 2009.

[87] S. Gratton, M. Mouffe, P. L. Toint, and M. Weber-Mendonça, "A recursive ℓ_∞-trust-region method for bound-constrained nonlinear optimization," *IMA Journal of Numerical Analysis*, vol. 28, no. 4, pp. 827–861, 2008.

[88] S. Gratton, A. Sartenaer, and P. L. Toint, "Recursive trust-region methods for multiscale nonlinear optimization," *SIAM Journal on Optimization*, vol. 19, no. 1, pp. 414–444, 2008.

[89] P. L. Toint, D. Tomanos, and M. Weber-Mendonca, "A multilevel algorithm for solving the trust-region subproblem," *Optimization Methods & Software*, vol. 24, no. 2, pp. 299–311, 2009.

[90] S. Gratton, M. Mouffe, A. Sartenaer, P. L. Toint, and D. Tomanos, "Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization," *Optimization Methods & Software*, vol. 25, no. 3, pp. 359–386, 2010.

[91] H. Calandra, S. Gratton, E. Riccietti, and X. Vasseur, "On high-order multilevel optimization strategies," *SIAM Journal on Optimization*, vol. 31, no. 1, pp. 307–330, 2021.

[92] G. Narkiss and M. Zibulevsky, *Sequential subspace optimization method for large-scale unconstrained problems*. Technion-IIT, Department of Electrical Engineering, 2005.

[93] M. Elad, B. Matalon, and M. Zibulevsky, "Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization," *Applied and Computational Harmonic Analysis*, vol. 23, no. 3, pp. 346–367, 2007.

[94] M. Zibulevsky and M. Elad, "L1-L2 optimization in signal and image processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 76–88, 2010.

[95] M. Schmidt, "minFunc: unconstrained differentiable multivariate optimization in Matlab," *Software available at http://www.cs.ubc.ca/∼schmidtm/Software/minFunc.htm*, 2005.

[96] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov, "On two-grid convergence estimates," *Numerical Linear Algebra with Applications*, vol. 12, no. 5-6, pp. 471–494, 2005.

[97] I. Yavneh, "Coarse-grid correction for nonelliptic and singular perturbation problems," *SIAM Journal on Scientific Computing*, vol. 19, no. 5, pp. 1682–1699, 1998.

[98] P. Hemker, "On the order of prolongations and restrictions in multigrid procedures," *Journal of Computational and Applied Mathematics*, vol. 32, no. 3, pp. 423–429, 1990.

[99] M. Elad, *Sparse and redundant representations: From theory to applications in signal and image processing*. Springer, 2010.

[100] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *International Conference on Acoustics, Speech, and Signal Processing. Proceedings (ICASSP)*, vol. 5, pp. 2443–2446, IEEE, 1999.

[101] Z. Zhu and M. B. Wakin, "Approximating sampled sinusoids and multiband signals using multiband modulated DPSS dictionaries," *Journal of Fourier Analysis and Applications*, vol. 23, no. 6, pp. 1263–1310, 2017.

[102] M. Elad, "Optimized projections for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 55, no. 12, pp. 5695–5702, 2007.

[103] G. Li, Z. Zhu, D. Yang, L. Chang, and H. Bai, "On projection matrix optimization for compressive sensing systems," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2887–2898, 2013.

[104] G. Li, X. Li, S. Li, H. Bai, Q. Jiang, and X. He, "Designing robust sensing matrix for image compression," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5389–5400, 2015.

[105] W. Chen, M. R. Rodrigues, and I. J. Wassell, "Projection design for statistical compressive sensing: A tight frame based approach," *IEEE Transactions on Signal Processing*, vol. 61, no. 8, pp. 2016–2029, 2013.

[106] T. Hong and Z. Zhu, "An efficient method for robust projection matrix design," *Signal Processing*, vol. 143, no. 3, pp. 200–210, 2018.

[107] J. M. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1395–1408, 2009.

[108] H. Bai, G. Li, S. Li, Q. Li, Q. Jiang, and L. Chang, "Alternating optimization of sensing matrix and sparsifying dictionary for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 63, no. 6, pp. 1581–1594, 2015.

[109] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *International Conference on Machine Learning*, pp. 689–696, PMLR, 2009.

[110] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, no. 1, 2010.

[111] J. Sulam, B. Ophir, M. Zibulevsky, and M. Elad, "Trainlets: Dictionary learning in high dimensions," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3180–3193, 2016.

[112] J. Sulam and M. Elad, "Large inpainting of face images with trainlets," *IEEE Signal Processing Letters*, vol. 23, no. 12, pp. 1839–1843, 2016.

[113] E. Richardson, R. Herskovitz, B. Ginsburg, and M. Zibulevsky, "SEBOOST-Boosting stochastic learning using subspace optimization techniques," *Advances in Neural Information Processing Systems*, vol. 29, pp. 1534–1542, 2016.

[114] T. Hong, H. Bai, S. Li, and Z. Zhu, "An efficient algorithm for designing projection matrix in compressive sensing based on alternating optimization," *Signal Processing*, vol. 125, pp. 9–20, 2016.

[115] W.-S. Lu and T. Hinamoto, "Design of projection matrix for compressive sensing by nonsmooth optimization," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 1279–1282, IEEE, 2014.

[116] M. Sadeghi and M. Babaie-Zadeh, "Incoherent unit-norm frame design via an alternating minimization penalty method," *IEEE Signal Processing Letters*, vol. 24, no. 1, pp. 32–36, 2016.

[117] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.

[118] Z. Zhu, Q. Li, G. Tang, and M. B. Wakin, "Global optimality in low-rank matrix optimization," *IEEE Transactions on Signal Processing*, vol. 66, no. 13, pp. 3614–3628, 2018.

[119] G. Li, Z. Zhu, X. Wu, and B. Hou, "On joint optimization of sensing matrix and sparsifying dictionary for robust compressed sensing systems," *Digital Signal Processing*, vol. 73, pp. 62–71, 2018.

[120] Z. Zhu, G. Li, J. Ding, Q. Li, and X. He, "On collaborative compressive sensing systems: The framework, design, and algorithm," *SIAM Journal on imaging sciences*, vol. 11, no. 2, pp. 1717–1758, 2018.

[121] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems*, pp. 801–808, 2007.

[122] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of Computational Statistics*, pp. 177–186, Springer, 2010.

[123] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 157–173, 2008.

[124] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[125] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[126] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[127] A. S. Bandeira, E. Dobriban, D. G. Mixon, and W. F. Sawin, "Certifying the restricted isometry property is hard," *IEEE Transactions on Information Theory*, vol. 59, no. 6, pp. 3448–3450, 2013.

[128] V. Abolghasemi, S. Ferdowsi, and S. Sanei, "A gradient-based alternating minimization approach for optimization of the measurement matrix in compressive sensing," *Signal Processing*, vol. 92, no. 4, pp. 999–1009, 2012.

[129] X. Li, H. Bai, and B. Hou, "A gradient-based approach to optimization of compressed sensing systems," *Signal Processing*, vol. 139, pp. 49–61, 2017.

[130] W. Yin, S. Morgan, J. Yang, and Y. Zhang, "Practical compressive sensing with Toeplitz and circulant matrices," in *Visual Communications and Image Processing*, vol. 7744, p. 77440K, International Society for Optics and Photonics, 2010.

[131] G. Zhang, S. Jiao, X. Xu, and L. Wang, "Compressed sensing and reconstruction with Bernoulli matrices," in *International Conference on Information and Automation*, pp. 455–460, IEEE, 2010.

[132] J. Sun, S. Wang, and Y. Dong, "Sparse block circulant matrices for compressed sensing," *IET Communications*, vol. 7, no. 13, pp. 1412–1418, 2013.

[133] F. Fan, "Toeplitz-structured measurement matrix construction for chaotic compressive sensing," in *International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 19–22, IEEE, 2014.

[134] L. Gan, T. T. Do, and T. D. Tran, "Fast compressive imaging using scrambled block Hadamard ensemble," in *European Signal Processing Conference*, pp. 1–5, IEEE, 2008.

[135] H. Rauhut, "Compressive sensing and structured random matrices," *Theoretical Foundations and Numerical Methods for Sparse Recovery*, vol. 9, pp. 1–92, 2010.

[136] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.

[137] Y. Dou, S. Vassiliadis, G. K. Kuzmanov, and G. N. Gaydadjiev, "64-bit floating-point FPGA matrix multiplication," in *International Symposium on Field-programmable Gate Arrays*, pp. 86–95, ACM, 2005.

[138] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, 2011.

[139] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 937–947, 2010.

[140] T. Strohmer and R. W. Heath, "Grassmannian frames with applications to coding and communication," *Applied and Computational Harmonic Analysis*, vol. 14, no. 3, pp. 257–275, 2003.

[141] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[142] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization," *Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.

[143] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[144] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[145] P. Dita, "Factorization of unitary matrices," *Journal of Physics A: Mathematical and General*, vol. 36, no. 11, p. 2781, 2003.

[146] Q. Li, Z. Zhu, and G. Tang, "The non-convex geometry of low-rank matrix optimization," *Information and Inference: A Journal of the IMA*, vol. 8, no. 1, pp. 51–96, 2019.

[147] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, "Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality," *Mathematics of Operations Research*, vol. 35, no. 2, pp. 438–457, 2010.

[148] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[149] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods," *Mathematical Programming*, vol. 137, no. 1-2, pp. 91–129, 2013.

[150] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[151] A. Beck and Y. C. Eldar, "Sparsity constrained nonlinear optimization: Optimality conditions and algorithms," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1480–1509, 2013.

[152] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk, "Beyond Nyquist: Efficient sampling of sparse bandlimited signals," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 520–544, 2010.

[153] M. A. Davenport and M. B. Wakin, "Compressive sensing of analog signals using discrete prolate spheroidal sequences," *Applied and Computational Harmonic Analysis*, vol. 33, no. 3, pp. 438–472, 2012.

[154] I. Yavneh and G. Dardyk, "A multilevel nonlinear method," *SIAM Journal on Scientific Computing*, vol. 28, no. 1, pp. 24–46, 2006.

[155] T.-a. Pham, E. Soubies, A. Ayoub, J. Lim, D. Psaltis, and M. Unser, "Three-dimensional optical diffraction tomography with Lippmann-Schwinger model," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 727–738, 2020.

[156] T. Hong, T.-a. Pham, E. Treister, and U. Michael, "Diffraction tomography with Helmholtz equation: Efficient and robust multigrid-based solver," *arXiv preprint arXiv:2107.03679*, 2021.

[157] L. Métivier, R. Brossier, J. Virieux, and S. Operto, "Full waveform inversion and the truncated Newton method," *SIAM Journal on Scientific Computing*, vol. 35, no. 2, pp. B401–B437, 2013.

המחקר בוצע בהנחייתו של פרופ. עירד יבנה, בפקולטה למדעי המחשב בטכניון.

אני מודה לטכניון על התמיכה הנדיבה במשך השתלמותי.

אופטימיזציה נומרית ושיטות רב-סריג לחישוב עם שימושים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

דוקטור לפילוסופיה

טאו הונג

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

אלול התשפ"א חיפה אוגוסט 2021

אופטימיזציה נומרית ושיטות רב-סריג
לחישוב עם שימושים

טאו הונג