THE UNIVERSITY
of EDINBURGH

Higher-Order Methods for Large-Scale Optimization

Kimon Fountoulakis

Doctor of Philosophy
University of Edinburgh
March 2015

To everyone who, both directly or indirectly, piqued my interest in research

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Kimon Fountoulakis*)

Abstract

There has been an increased interest in optimization for the *analysis of large-scale data sets* which require gigabytes or terabytes of data to be stored. A variety of applications originate from the fields of signal processing, machine learning and statistics. Seven representative applications are described below.

- Magnetic Resonance Imaging (MRI): A medical imaging tool used to scan the anatomy and the physiology of a body [80].
- Image inpainting: A technique for reconstructing degraded parts of an image [14].
- Image deblurring: Image processing tool for removing the blurriness of a photo caused by natural phenomena, such as motion [64].
- Radar pulse reconstruction [120].
- Genome-Wide Association study (GWA): DNA comparison between two groups of people (with/without a disease) in order to investigate factors that a disease depends on [112].
- Recommendation systems: Classification of data (i.e., music or video) based on user preferences [67].
- Data fitting: Sampled data are used to simulate the behaviour of observed quantities. For example estimation of global temperature based on historic data [65].

Large-scale problems impose restrictions on methods that have been so far employed. The new methods have to be memory efficient and ideally, within seconds they should offer noticeable progress towards a solution.

First-order methods meet some of these requirements. They avoid matrix factorizations, they have low memory requirements, additionally, they sometimes offer fast progress in the initial stages of optimization. *Unfortunately, as demonstrated by numerical experiments in this thesis, first-order methods miss essential information about the conditioning of the problems, which might result in slow practical convergence.* The main advantage of first-order methods which is to rely only on simple gradient or coordinate updates becomes their essential weakness.

We do not think this inherent weakness of first-order methods can be remedied. For this reason, the present thesis aims at the *development and implementation of inexpensive higher-order methods for large-scale problems.*

Acknowledgements

In 2010 I found myself working hard as a PhD student at Imperial College London. Financial difficulties made various aspects of my PhD experience unpleasant. Full of disappointment I took the decision to quit my PhD. Few months later and after a short collaboration with Prof. Jacek Gondzio, I was granted a scholarship for a PhD position at the University of Edinburgh. Jacek believed in me and supported my PhD application to the decision committee. Now I find myself finishing a (hopefully) successful PhD and starting a post-doctorate fellowship position at University of California Berkeley. For this, I am grateful to my supervisor and mentor Jacek. Jacek has taught me the way to survive in the scientific community. His advice on all possible aspects of research was priceless.

I would like to express my gratitude to my examiners, Prof. François Glineur and Dr Julian Hall for their important suggestions which improved my thesis.

I am also grateful to (listed in alphabetical order of last surname) Prof. Coralia Cartis, Prof. Petros Drineas, Dr Andreas Grothey, Prof. Daniel Kuhn, Dr George Tzallas Regas, Dr Peter Richtárik and Prof. Stephen Wright for supporting me during my PhD.

Of course I cannot forget the fruitful discussions with my colleagues: (listed in alphabetical order of last surname) Dr Pablo González Brevis, Dr Ioannis Dassios, Robert Mansel Gower, Tim Schulze, Dr Martin Takáč, Dr Rachael Tappenden and Dr Pavel Zhlobich. I improved myself by learning from all of you, thank you.

I am indebted to Dr Kenton D'Mellow for his technical support in solving a problem of trillion variables using ARCHER, the UK's National Supercomputing Service.

During my undergraduate years in Athens I was very lucky to meet inspiring people such as Prof. Diomidis Spinellis and Prof. Christos Tarantilis. Thank you Diomidi and Christo for introducing me to research.

Most of thanks to people within funding organizations, who took the decision to cover the expenses for conferences that I attended. In particular, I would like to thank SIAM Student Travel Awards, Laura Wisewell Fund, Numerical Algorithms and Intelligent Software (NAIS), Research Development Fund and The General Michael Arnaoutis Foundation.

I would like to thank my parents and brother for occasionally having to financially support me the last four years and tolerate my grumpiness. Finally, I would like to express my gratitude to my beautiful wife Eva, ευχαριστώ Ευάκι!

Thank you all

Contents

Acronyms

ℓ_1-LR ℓ_1-regularized Logistic Regression.

ℓ_1-LS ℓ_1-regularized Least-Squares.

BPDN Basis Pursuit Denoising.

CG Conjugate Gradients.

CS Compressed Sensing.

dB decibel.

DCT Discrete Cosine Transform.

DST Discrete Sine Transform.

ECDF Edinburgh Compute and Data Facility.

FFTW Fastest Fourier Transform in the West.

FISTA Fast Iterative Shrinkage-Thresholding Algorithm.

FPC_AS Fixed Point Continuation Active Set.

IGen Instance Generator.

IPM Interior-Point Method.

KKT Karush-Kuhn-Tucker.

LASSO Least Absolute Shrinkage and Selection Operator.

LR Logistic Regression.

LS Least-Squares.

LSQR Least-Squares QR factorization.

LSVM Linear Support Vector Machine.

mfIPM matrix-free Interior-Point Method.

OsGen Optimal solution Generator.

PCDM Parallel Coordinate Descent Method.

PCG Preconditioned Conjugate Gradients.

PDCO Primal-Dual interior-point method for Convex Objectives.

pdNCG primal-dual Newton Conjugate Gradients.

PSSgb Projected Scaled Sub-gradient, Gafni-Bertsekas.

RIP Restricted Isometry Property.

SNR Signal to Noise Ratio.

SPGL1 Spectral Projected Gradient for ℓ_1-regularized least-squares.

SVM Support Vector Machine.

TFOCS Templates for First-Order Conic Solvers.

TVAL3 Total-Variation minimization by Augmented Lagrangian and ALternating direction ALgorithms.

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Modern unconstrained convex optimization problems require the manipulation of large-scale data sets at every iteration of the methods employed. Unfortunately, traditional optimization methods might be inadequate or they might have poor performance. For instance Newton's method requires at every iteration the solution of a linear system, which in many large-scale applications of our interest the matrix of the linear system might be dense. For example in Compressed Sensing (CS) [41, 54] the matrix of the linear system is of the form $A^\intercal A + \Theta$, where $A \in \mathbb{R}^{m \times n}$ is a partial Discrete Cosine Transform (DCT) with $m < n$, which is fully dense, and Θ is a diagonal matrix. For fully dense matrices, if Gaussian elimination is used to solve systems with matrix $A^\intercal A + \Theta$, this operation can have complexity of order $\mathcal{O}(n^3)$. For more advanced techniques, such as the Strassen's algorithm, the complexity is $\mathcal{O}(n^{\log_2 7})$ [94]. Even in the latter case, the computational cost per iteration is prohibitive for large-scale problems. It is obvious that if n is large, then even a single iteration might require prohibitively high computational time.

Large scale problems are the main reason for the resurgence in methods with computationally inexpensive iterations. For example many first-order methods were recovered and refined, such as coordinate descent [55, 69, 97, 108, 109, 110, 117, 118], alternating direction method of multipliers [17, 48, 57, 66, 113], proximal first-order methods [6, 30, 93] and first-order smoothing methods [10, 11, 89]. These are just a few representative examples, the list is too long for a complete illustration, many other examples can be found in [4, 29]. Often the goal of modern first-order methods is to reduce the computational complexity per iteration, while preserving the theoretical worst-case iteration complexity of classic first-order methods [87]. Many modern first-order methods meet the previous goal. For instance, for least-squares problems coordinate descent methods can have up to n times less computational complexity per iteration than methods which use full gradient steps [96, 97].

First-order methods have been very successful in various scientific fields, such as Support Vector Machine (SVM) [122], Compressed Sensing (CS) [42], image processing [30] and data fitting [65]. However, even for the simple unconstrained problems that arise in these fields there exist more challenging instances. Since first-order methods do not capture sufficient higher-order/curvature information, their performance degrades for ill-conditioned problems. Numerous examples are

number of variables	Processors	Memory (terabytes)	Time (seconds)
2^{30}	64	0.192	1,923
2^{32}	256	0.768	1,968
2^{34}	1024	3.072	1,986
2^{36}	4096	12.288	1,970
2^{38}	16384	49.152	1,990
2^{40}	65536	196.608	2,006

Table 1.1: Performance of pdNCG for synthetic large-scale ℓ_1-regularized Least-Squares (ℓ_1-LS) problems. All problems have been solved to relative error of order 10^{-4}

provided in Chapter 8 to illustrate this issue. The obvious question is: how can we solve efficiently such ill-conditioned problems? Standard higher-order methods such as Interior-Point Methods (IPMs) can be prohibitively expensive, and first-order methods do not capture sufficiently the curvature of the problem.

Our aim is the refinement of robust higher-order methods that sufficiently capture the structure of the underlying problem. The optimization community seems to consider the higher-order methods to be rather expensive. Our goal is to make the proposed methods as inexpensive as possible, while complicated problems can be efficiently solved. To accomplish this, inexact Newton-type methods and IPMs are used in a matrix-free environment. No non-trivial matrix factorization is performed and no excessive memory requirements are needed. Consequently, the main drawbacks of Newton's method are removed, while at the same time similar fast convergence properties are provably retained.

We employ a primal-dual Newton Conjugate Gradients (pdNCG) method [53] for which we present extensive theoretical analysis, i.e., global convergence and a worst-case iteration complexity result, also, we explicitly define the region of fast local convergence of the method. We show empirically in Subsection 8.3.6 that the performance of pdNCG scales linearly with respect to the number of variables of the problem. By consensus higher-order methods were not considered appropriate for large-scale problems. We show that pdNCG can be used to solve problems of a trillion variables by simply exploiting parallel matrix-vector product multiplication. See Table 1.1 as an example of numerical experiments to follow in Chapter 8 using pdNCG.

We develop *provably* efficient and *inexpensive* preconditioners, which capture the second-order information in sparse signal reconstruction problems, i.e., CS [41, 54]. The preconditioners are designed to exploit two important properties of these problems:

- sparsity of solutions and
- near orthogonality of matrices; a property known as Restricted Isometry Property (RIP), which implies triviality of linear algebra.

The preconditioning techniques are implemented within a matrix-free Interior-Point Method (mfIPM) [54] and pdNCG [41]. As a result of the efficiency of the

proposed preconditioning techniques, the computational complexity of the *inexact* solution of linear systems is *substantially* reduced. Overall, the performance of the methods developed is remarkably improved compared to versions of the methods which do not employ preconditioners.

Finally, we develop a mathematically rigorous problem generator [52] for ℓ_1-regularized Least-Squares (ℓ_1-LS) problems. The major advantage of the proposed generator is its flexibility. In particular, it allows the control of the sparsity and the conditioning of the problem, while it is memoryless and is based only on inexpensive operations. The proposed generator allows easy construction of challenging instances. Hence, it might be an important contribution for researchers that are interested in testing the performance and robustness of new methods.

In what follows we describe the problem of interest and main differences between first- and higher-order methods. A similar analysis has appeared in [52].

1.1 Problem formulation

Consider the problem

$$\text{minimize } f_\tau(x) := \tau\psi(x) + \varphi(x), \tag{1.1}$$

where $x \in \mathbb{R}^n$ and τ is a positive parameter. We assume that $\psi : \mathbb{R}^n \to \mathbb{R}$ is a (possibly) nonsmooth convex function and $\varphi : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function.

A variety of problems from many scientific fields such as machine learning [104], regression [107], CS [42, 80] and image processing [119] can be cast in the form of (1.1). Examples from the previous fields which are used in this thesis are discussed in more detail in Chapter 2. Below we give a brief description of various cases of functions ψ and φ.

- ℓ_1-regularized Logistic Regression (ℓ_1-LR): $\psi(x) = \|x\|_1$ and

$$\varphi(x) = \sum_{i=1}^{m} \log(1 + e^{-b_i x^\intercal a_i}),$$

  where $a_i \in \mathbb{R}^n$ $\forall i = 1, 2, \cdots, m$ and $b \in \{-1, 1\}^m$. The vector a_i denotes the i-th sample of n features and b_i denotes the class in which the i-th sample belongs, -1 or 1. An application is binary classification [39], where one is looking for a hyperplane that separates the points a_i $\forall i$ into two categories given their category b_i. By minimizing problem (1.1) we obtain the coefficients of the hyperplane. The purpose of the ℓ_1-norm is to enforce sparsity in the optimal solution, which translates to an automatic selection of features that the classification is based on.

- ℓ_1-regularized Least-Squares (ℓ_1-LS): $\psi(x) = \|x\|_1$ and $\varphi(x) = 1/2\|Ax-b\|_2^2$, where $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and $b \in \mathbb{R}^m$. An application is data fitting, where the aim is to fit a linear function to n-dimensional sampled points (rows of matrix A). This is a standard application of modern large-scale data

analysis, which is frequently found in statistics [107], geological sciences [65] as well as many other areas. Again, the purpose of the ℓ_1-norm is the automatic selection of factors (number of dimensions) that are needed to fit the linear function to the data.

- CS [9]: $\psi(x) = \|W^* x\|_1$ and $\varphi(x) = 1/2\|Ax - b\|_2^2$, where $W \in E^{n \times l}$, $E = \mathbb{C}$ or $\mathbb{R}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and the star superscript denotes the conjugate transpose operator. The aim is to find a representation x of the signal b that is sparse in a space defined by the rows of matrix W. Frequently W is the identity matrix, which implies that we are looking for a sparse representation x. Other instances require W to be a discretization of the nabla operator, which implies that the gradient of x is sparse, i.e., the vector x is piece-wise constant.

1.2 Methods for continuous convex optimization

We are concerned with methods that have the following intuitive setting. At every iteration a convex function $y \to Q(y; x)$ is created that locally approximates f_τ at a given point x. Then, function Q is minimized to obtain the next point. An example that covers the previous setting is the Algorithm: Generic Algorithmic Framework (GFrame), which is given below.

Algorithm 1.1 Generic Framework (GFrame)

1: Initialize $x_0 \in \mathbb{R}^n$ and $y_0 \in \mathbb{R}^n$
For $k = 0, 1, 2, \ldots$ until some termination criteria are satisfied
2: Construct a convex function $Q(y; y^k)$ which approximates f_τ in a neighbourhood of y^k
3: Approximately (or exactly) solve the sub-problem

$$x^{k+1} \approx \arg\min_y \; Q(y; y^k) \tag{1.2}$$

4: Find a step-size $\alpha > 0$ based on some criteria and set

$$y^{k+1} = x^k + \alpha(x^{k+1} - x^k)$$

end-for
5: Return approximate solution x^{k+1} or y^{k+1}

Loosely speaking, close to the optimal solution of problem (1.1), the better the approximation of Q to f_τ at any point x the fewer iterations required to solve (1.1). On the other hand, the practical performance of such methods is a trade-off between careful incorporation of the curvature of f_τ, i.e., second-order derivative information, in Q and the cost of solving sub-problem (1.2) in GFrame.

1.2.1 Curvature preservation or inexpensive iterations?

Let us now discuss two examples of Q that consider a different trade-off. For demonstration purposes we will consider the ℓ_1-LR problem:

$$\text{minimize } \tau \underbrace{\|x\|_1}_{\psi} + \underbrace{\sum_{i=1}^{m} \log(1 + e^{-b_i x^\intercal a_i})}_{\varphi}.$$

We consider the following two cases of Q.

- Separable quadratic:

$$Q(y; y^k) := \tau\|y\|_1 + \varphi(y^k) + (\nabla\varphi(y^k))^\intercal(y - y^k) + \frac{L}{2}(y - y^k)^\intercal(y - y^k),$$

  where L is the Lipschitz constant of the first-order derivative $\nabla\varphi$.

- Non separable quadratic:

$$Q(y; y^k) := f_\tau^\mu(y^k) + (\nabla f_\tau^\mu(y^k))^\intercal(y - y^k) + \frac{1}{2}(y - y^k)^\intercal\nabla^2 f_\tau^\mu(y^k)(y - y^k),$$

  where $f_\tau^\mu = \tau\psi_\mu + \varphi$, and ψ_μ is a smooth approximation of $\|x\|_1$, for example:

$$\psi_\mu(x) = \sum_{i=1}^{n} \Big((\mu^2 + x_i^2)^{\frac{1}{2}} - \mu\Big), \tag{1.3}$$

  where $\mu > 0$ is an approximation parameter. Function (1.3) is called pseudo-Huber, we explain its derivation in Subsection 3.2.2. Moreover, $\nabla^2 f_\tau^\mu(y^k)$ is the Hessian of f_τ^μ at y^k.

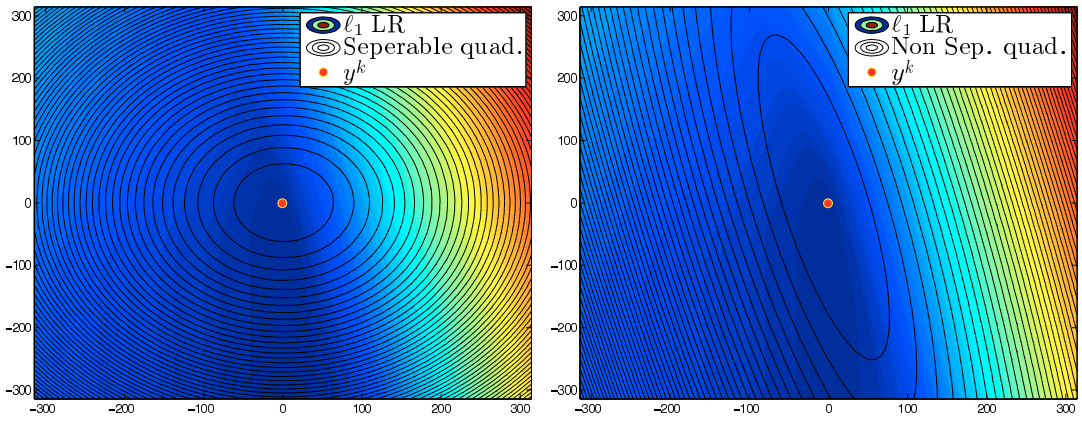
Observe in Figure 1.1a that the non separable quadratic, which is employed by higher-order methods, captures better the structure of f_τ than the separable quadratic in Subfigure 1.1b. However, the minimization of the non separable quadratic involves the solution of a linear system, which can be expensive. This means that an iterative solver will have to be used to minimize approximately or exactly function Q. Exact solution of the sub-problem (1.2) would mean that the sub-problem is as difficult as the initial problem. Fortunately, this cost can be reduced by only solving the sub-problems approximately. We will discuss in Chapters 4, 5 and 6 how much of inexactness in the solution of sub-problems is allowed for the proposed methods.

In what follows we give some more insight into the differences between first- and higher-order methods.

1.2.2 First-order methods

We fix the structure of Q for problem (1.1) to be:

$$Q(y; x) := \tau\psi(y) + \varphi(x) + \nabla\varphi(x)^\intercal(y - x) + \frac{1}{2}(y - x)^\intercal H(x)(y - x), \tag{1.4}$$



(a) Separable quadratic (b) General quadratic

Figure 1.1: Demonstration of two different types of convex function Q which locally approximate the objective function f_τ at point y^k. In the left figure, function Q is a simple separable quadratic that is used by first-order methods. In the right figure, function Q is a non separable quadratic that is used by higher-order methods

where $H(x) \in \mathbb{R}^{n\times n}$ is a positive definite matrix $\forall x$. Notice that the decision of creating Q has been reduced to a decision of selecting H. Ideally, matrix H should be chosen such that it represents curvature information of φ at point x, i.e., matrix H should have similar spectral decomposition to the second-order derivative $\nabla^2\varphi(x)$. Let $B(x) := \{v \in \mathbb{R}^n \mid \|v - x\|_2^2 \leq 1\}$ be a unit ball centered at x. Then, H should be selected in an optimal way:

$$\text{minimize}_{H\succeq 0} \int_{B(x)} |f_\tau(y) - Q(y;x)|dy. \tag{1.5}$$

The previous problem simply states that H should minimize the integral of the absolute values of the residual $f_\tau - Q$ over B. The following problem is similar (but not equivalent) to:

$$\text{minimize}_{H\succeq 0} \int_{B(x)} \Big|(y-x)^\intercal(\nabla^2\varphi(x) - H)(y-x)\Big|dy. \tag{1.6}$$

It is trivial to see that the best possible H is simply $H(x) = \nabla^2\varphi(x)$. However, this makes every sub-problem (1.2) as difficult to be minimized as the original problem (1.1). One has to reevaluate the trade-off between a matrix H which sufficiently well represents curvature information of φ compared to a simple matrix H that is not as good approximation but offers an inexpensive solution of sub-problem (1.2). An example can be obtained by setting H to be a positively scaled identity, which gives a solution to problem (1.6) $H = LI_n$, where L is the largest eigenvalue of $\nabla^2\varphi(x)$. In this case, if ψ is simple, i.e., $\psi(x) = \|x\|_1$, then the sub-problem (1.2) has an inexpensive closed form solution known as proximal operator [93]. Although nearly all spectral properties of $\nabla^2\varphi(x)$ are lost, the computational complexity per iteration is so low that one hopes that this will compensate for the poor curvature information. For easy problems this is indeed the case and

then the first-order methods work sufficiently well. However, for any non-trivial problems such rough approximations of $\nabla^2\varphi$ prove to be insufficient.

The methods which use such simplistic approximation of $\nabla^2\varphi$ are called first-order methods and have been shown to be efficient for well-conditioned large-scale problems of the form of (1.1) [6, 30, 93].

1.2.3 Higher-order methods

The second example involves the approximation of the non-smooth function ψ with a smooth function ψ_μ. For example, if $\psi(x) = \|x\|_1$, then ψ_μ can be the pseudo-Huber function (1.3). Using the smooth function ψ_μ, problem (1.1) is replaced with the following:

$$\text{minimize } f_\tau^\mu(x) := \tau\psi_\mu(x) + \varphi(x). \tag{1.7}$$

The smaller μ is the better the approximation of problem (1.7) to (1.1). The advantage is that f_τ^μ in (1.7) is a smooth function, which has derivatives of all degrees. Hence, smoothing will allow access to second-order information of the function f_τ and essential curvature information will be exploited. For the smooth problem (1.7), the convex approximation Q at x is:

$$Q(y;x) := f_\tau^\mu(x) + \nabla f_\tau^\mu(x)^\intercal(y-x) + \frac{1}{2}(y-x)^\intercal H(x)(y-x), \tag{1.8}$$

where $H(x)$ is a positive definite matrix, which satisfies $H(x) \approx \nabla^2 f_\tau^\mu(x)$. As we have mentioned before, if $H(x) = \nabla^2 f_\tau^\mu(x)$, then Q gives us the best possible (in the sense of minimizing (1.6)) quadratic approximation to f_τ^μ at x. In this case, minimizing the sub-problem (1.2) might be a more expensive operation. Therefore, we rely on an approximate solution of (1.2) using some iterative method which requires only simple matrix-vector product operations with matrix $\nabla^2 f_\tau^\mu$. Such methods are approximately second-order and by consensus they are efficient on medium scale problems or when high precision accuracy is required. It is frequently claimed [6, 10, 11, 63, 103] that second-order methods do not scale favourably with the dimensions of the problem because of the more costly task of solving approximately the sub-problems in (1.2), instead of having an inexpensive closed form solution. Such claims are based on an assumption that *full* second-order information has to be used when solving sub-problem (1.2). Clearly, this is not necessary: ***approximate*** second-order information suffices. In this thesis we provide evidence [53, 60] that for non-trivially ill-conditioned problems, second-order methods can be efficient.

1.3 Outline

In Chapter 2 we describe three applications that we are concerned with in this thesis. The applications are linear data fitting, CS and Linear Support Vector Machine (LSVM). All three applications can be formulated as optimization problems in the form of (1.1).

In Chapter 3 we describe a smoothing technique for function $\psi = \|W^*x\|_1$ in problem (1.1), where $W \in E^{n\times l}$, $E = \mathbb{C}$ or $\mathbb{R}$. The smoothing technique is used in the proposed primal-dual Newton Conjugate Gradients (pdNCG) methods. We discuss also how the smoothing technique is related to a primal-dual reformulation of problem (1.1), which is solved by pdNCG.

In Chapter 4 we present the proposed pdNCG method. Extended theoretical analysis is given, i.e., global convergence, global and local worst-case iteration complexity results. Chapter 4 is based on [53].

In Chapter 5 we present a specialized version of pdNCG for CS problems. A novel preconditioning technique is discussed that speeds up the iterative solution of linear systems using pdNCG. Theoretical and empirical analysis is provided which demonstrates the efficiency of the preconditioning technique. Chapter 5 is based on [41].

In Chapter 6 we present the matrix-free Interior-Point Method (mfIPM) for CS problems. We discuss also an efficient preconditioning technique to speed up the iterative solution of linear systems using mfIPM. Theoretical and empirical analysis is also provided. Chapter 6 is based on [54].

In Chapter 7 we present a flexible and memory-less instance generator for ℓ_1-LS problems. The generator allows control of the sparsity and the conditioning of the problem. Chapter 7 is based on [52].

In Chapter 8 we present numerical experiments which demonstrate the performance of the proposed algorithms, pdNCG and mfIPM, compared to state-of-the-art algorithms. Chapter 7 is based on [41, 52, 53, 54].

In Chapter 9 we make our conclusions regarding the performance of higher-order methods for large-scale problems and we discuss limitations and further developments.

Chapter 2

Applications

In this chapter we discuss applications that are used to test the practical performance of the proposed methods. The first class of applications originates from linear data fitting, which includes CS. The second class of applications originates from LSVM.

Part of the material in this chapter has been presented in [41, 53, 54].

2.1 Linear data fitting

Let us assume that we sample m data points (a_i, b_i), where $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ $\forall i = 1, 2, \cdots, m$. We assume linear dependence of b_i on a_i:

$$b_i = a_i^\intercal x + e_i \quad \forall i = 1, 2, \cdots, m,$$

where e_i is an error term due to the sampling process being inaccurate. Depending on the application some statistical information is assumed about vector e, i.e., each component of e follows a standard normal distribution with some relatively small variance. In matrix form the previous relationship is:

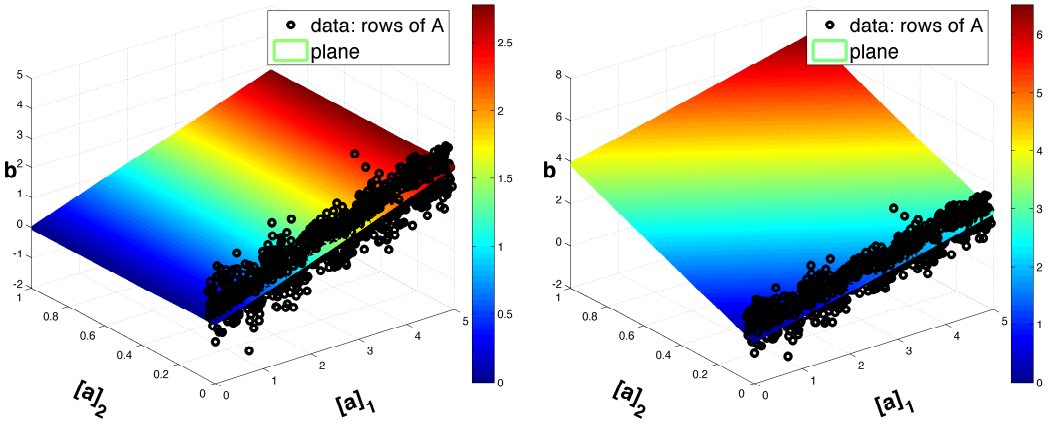
$$b = Ax + e, \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix with a_i's as its rows and $b \in \mathbb{R}^m$ is a vector with b_i's as its components. The goal is to find a sparse vector x (with many zero components) such that the error $\|Ax - b\|_2$ is minimized. To find x one can solve the ℓ_1-LS problem:

$$\text{minimize } \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{2.2}$$

where $\tau > 0$. The ℓ_1-norm is a sparsity inducing norm [4], hence, its role is to promote sparsity in solution $\tilde{x}$ of problem (2.2).

An example for the purpose of the ℓ_1-norm is presented in Figure 2.1. Figure 2.1 shows a two dimensional example where $n = 2$, $m = 1000$ and matrix A is full-rank. Notice that the data points a_i $\forall i = 1, 2, \cdots, m$ have large variations with respect to feature $[a_i]_1$ $\forall i$, where $[\cdot]_j$ is the jth component of the input vector, while there is only a small variation with respect to feature $[a_i]_2$ $\forall i$. This property is captured when problem (2.2) is solved with $\tau = 30$. The fitted plane



(a) ℓ_1-LS (b) LS

Figure 2.1: Demonstration of the purpose of the ℓ_1-norm for data fitting problems. For details see text in Section 2.1

in Figure 2.1a depends only on the first feature $[a]_1$, while the second feature $[a]_2$ is ignored because $[\tilde{x}]_2 = 0$. This can be observed through the level sets of the plane shown with the colored map; for each value of $[a]_1$ the level sets remain constant for all values of $[a]_2$. On the contrary, this is not the case when one solves the Least-Squares (LS) problem ($\tau = 0$ in (2.2)). Observe in Figure 2.1a that the fitted plane depends on both features $[a]_1$ and $[a]_2$.

Before we discuss some particular applications of linear data fitting, let us mention that the dependence of b and A is not necessarily linear. In this case one should fit a nonlinear function instead, see [65] for more details.

2.2 Compressed Sensing (CS)

CS [42] is a special case of data fitting where we are concerned with the solution of the under-determined system, $m < n$, of linear equations:

$$Ax = \hat{b}, \tag{2.3}$$

where $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, $\hat{b} \in \mathbb{R}^m$. In particular, we are interested in the solution x with the smallest possible number of non-zero elements; the sparsest solution $\hat{x}$.

The sparsest solution $\hat{x}$ of system (2.3) can be found by solving the following problem:

$$\begin{array}{ll} \text{minimize} & \|x\|_0 \\ \text{subject to:} & Ax = \hat{b}, \end{array} \tag{2.4}$$

where $\|\cdot\|_0$ is the zero "norm" which returns the number of non-zero components of the input vector. The use of the zero-norm makes the problem combinatorial and intractable in practice. Recent advances in the field of CS show that in certain situations [22] exact recovery of the sparsest solution $\hat{x}$ of (2.3) can be achieved with an overwhelming probability by solving the following basis pursuit

[35] problem:

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to:} & Ax = \hat{b}. \end{array} \tag{2.5}$$

Problem (2.5) has a major advantage over (2.4). Unlike the zero-norm objective in (2.4), the ℓ_1-norm objective in (2.5) can be reformulated as a linear function with non-negativity constraints and therefore the problem (2.5) can be recast as a linear problem and becomes computationally tractable. Having a linear reformulation of (2.5), standard efficient optimization methods can be used to recover the sparsest solution $\hat{x}$.

In real-life applications the right hand side of (2.3) is often corrupted with noise and (2.3) is replaced with:

$$Ax = b = \hat{b} + e, \tag{2.6}$$

where $e \in \mathbb{R}^m$ denotes the error: we assume it has a normal distribution $e_i \sim \mathcal{N}(0, \sigma^2)\ \forall\, i = 1, 2, \ldots, m$. For the noisy case (2.6) the sparsest solution $\hat{x}$ can be found by solving the well-known Basis Pursuit Denoising (BPDN) problem (2.2), which was introduced in [35], or one of the following problems:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2 \\ \text{subject to:} & \|x\|_1 \leq \epsilon_1 \end{array} \tag{2.7a}$$

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to:} & \|Ax - b\|_2 \leq \epsilon_2, \end{array} \tag{2.7b}$$

where ϵ_1 and ϵ_2 are positive scalars that regulate the sparsity and the upper bound on the noise error, respectively. Problem (2.7a) is the Least Absolute Shrinkage and Selection Operator (LASSO) used frequently in the field of computational statistics [83]. It can be shown using Theorem 27.4 from [98] that problem (2.2) and the problems in (2.7) are equivalent for specific values of scalars τ, ϵ_1 and ϵ_2.

2.2.1 Properties of CS matrices

Matrices which appear in sparse reconstruction problems originate from different bases in which signals are represented. What they all have in common are the conditions that guarantee recoverability of the sparsest solution of (2.3) by means of the ℓ_1-norm minimization (2.5). The Restricted Isometry Property (RIP) [27] is one of such conditions which shows how efficiently a measurement matrix captures information about sparse signals. In the following definition a q-sparse vector x is a vector with q non-zero components.

Definition 2.1. *The restricted isometry constant δ_q of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as the smallest δ_q such that*

$$(1 - \delta_q)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_q)\|x\|_2^2, \tag{2.8}$$

for all at most q-sparse $x \in \mathbb{R}^n$.

In words, for small δ_q, statement (2.8) requires that all column sub-matrices

of A with at most q columns are well-conditioned. Informally, A is said to satisfy the RIP if δ_q is small for a reasonably large q. The next theorem due to [51] establishes the relation between the RIP property and the sparse recovery.

Theorem 2.1. *Every q-sparse vector $x \in \mathbb{R}^n$ satisfying $Ax = \hat{b}$ is the unique solution of (2.5) if*

$$\delta_{2q} < \frac{3}{4+\sqrt{6}} \approx 0.4652.$$

The RIP also implies stable recovery by ℓ_1-norm minimization for vectors that can be well approximated by sparse ones, and it further implies robustness under noise on the measurements [27].

RIP is a very restrictive condition that depends on the size of the measurement matrix A. Clearly, the more columns n matrix A has (the larger the size of the vector x to recover) the larger δ_q in (2.8) is (the harder it is to guarantee sparse recovery). On the other hand, number of rows m of A is the number of measurements taken and, hence, the RIP constant δ_q decreases with m.

Quite often in applications a signal is sparse with respect to a basis different from the one in which measurements are made. Then it is said that a measurement/sparsity pair is given [22]. Assume that a vector z is sparse with respect to the basis of columns of a unitary matrix Ψ (*sparsity matrix*), i.e., $z = \Psi x$ for a q-sparse vector x. Further, assume that z is sampled with respect to the basis of columns of a unitary matrix Φ (*measurement matrix*): $y = R_m \Phi^{\mathsf{T}} z$, where R_m is a random sampling operator which satisfies $R_m R_m^{\mathsf{T}} = I_m$, where I_m is the identity matrix of size $m \times m$. Hence, matrix A in (2.3) is equal to $R_m \Phi^{\mathsf{T}} \Psi$ and its rows are orthonormal:

$$AA^{\mathsf{T}} = I_m. \tag{2.9}$$

The recoverability property of matrix A depends on the value of the so-called *mutual coherence* $\mu(\Phi, \Psi)$ of the measurement/sparsity pair (see [43]):

$$\mu(\Phi, \Psi) = \sqrt{n} \max_{i,j} | \langle \phi_i, \psi_j \rangle |, \tag{2.10}$$

where ϕ_i, ψ_i are the i^{th} columns of matrices Φ, Ψ, respectively. Coherence simply measures the largest correlation between any two elements of Φ and Ψ. Next theorem due to [21] shows that the smaller the value of mutual coherence the better the recoverability property of matrix A.

Theorem 2.2. *Fix $z \in \mathbb{R}^n$ and suppose that the coefficient sequence x of z in the unitary $n \times n$ basis Ψ is q-sparse. Select m measurements in the unitary $n \times n$ Φ domain uniformly at random. Then if*

$$m \geq Cq\mu(\Phi, \Psi)^2 \log(n/p) \quad \text{and} \quad m \geq C' \log^2(n/p) \tag{2.11}$$

for some positive constants C, C', then with overwhelming probability exceeding $1 - p$, the vector x is the unique solution to the ℓ_1 minimization problem (2.5) with $A = R_m \Phi^{\mathsf{T}} \Psi$, where $R_m R_m^{\mathsf{T}} = I_m$ and A has orthonormal rows (2.9).

To conclude, CS matrices have many useful properties that must be taken into account in the development of an efficient solver. We make use only of the

most general of them that are satisfied by every CS matrix. First, we weaken a little bit the condition of orthonormality (2.9) to include random matrices such as Gaussian and Bernoulli. In particular, we assume that:

- the rows of matrix A are close to orthonormal, i.e., there exists a small δ such that
$$\|AA^{\mathsf{T}} - I_m\|_2 \leq \delta. \tag{2.12}$$

RIP (2.8) on the contrary assumes that columns of A are normalized. So, our interpretation of the RIP property that will be used throughout this thesis is as follows.

- Every q columns of A with $q \ll m$ are almost orthogonal and have similar norms, i.e., for every matrix B composed of arbitrary q columns of A we have that:
$$\left\| \frac{n}{m} B^{\mathsf{T}} B - I_q \right\|_2 \leq \delta_q, \tag{2.13}$$

where I_q is the identity matrix of size $q \times q$. By treating property (2.13) as the chosen RIP, the bound for the RIP constant in Theorem 2.1, which relies on RIP in (2.8) will change. The following theorem is a modified version of Theorem 2.1 when property (2.13) is used as a RIP.

Theorem 2.3. *Every q-sparse vector $x \in \mathbb{R}^n$ satisfying $Ax = \hat{b}$ is the unique solution of (2.5) if*
$$\delta_{2q} < \frac{3\frac{m}{n}}{1 + 3\frac{m}{n} + \sqrt{6}},$$
where δ_{2q} is the minimum constant such that (2.13) holds for every $2q$ columns of matrix A, denoted by matrix B in (2.13).

Proof. This proof is a trivial modification of the proof of Proposition 2 in [51]. Let $x \in \mathbb{R}^n$ have q non-zero components and B in (2.13) be any q column sub-matrix of A. Then from (2.13) it follows that
$$\frac{m}{n}(1 - \delta_q)\|x\|_2^2 \leq \|Ax\|_2^2 \leq \frac{m}{n}(1 + \delta_q)\|x\|_2^2. \tag{2.14}$$

Proposition 2 in [51] gives bounds for δ_{2q} by using the RIP in (2.8). In our case, we replaced the RIP in (2.8) with (2.14). Therefore, the four modified conditions for δ_{2q} in Proposition 2 in [51], which guarantee that every q-sparse vector $x \in \mathbb{R}^n$ that satisfies $Ax = \hat{b}$ is the unique solution of (2.5), take the following forms:

1) $\delta_{2q} < \frac{1}{2}\frac{m}{n}$ when $q = 1$

2) $\delta_{2q} < \frac{3\frac{m}{n}}{(1 + 3\frac{m}{n} + \sqrt{(6q - 2r)/(q - 1)})}$ when $q = 3\omega + r$ and $1 \leq r \leq 3$

3) $\delta_{2q} < \frac{4\frac{m}{n}}{(1 + 4\frac{m}{n} + \sqrt{(12q - 3r)/(q - 1)})}$ when $q = 4\omega + r$ and $1 \leq r \leq 4$

4) $\delta_{2q} < \frac{2\frac{m}{n}}{(1 + 2\frac{m}{n} + \sqrt{1 + q/(8\omega + \lfloor 8r/5 \rfloor)})}$ when $q = 5\omega + r$ and $1 \leq r \leq 5$,

	Case 2	Case 3	Case 4
$q=2$	$\frac{3\frac{m}{n}}{1+3\frac{m}{n}+2\sqrt{2}}$	$\frac{4\frac{m}{n}}{1+4\frac{m}{n}+3\sqrt{2}}$	$\frac{\mathbf{2\frac{m}{n}}}{\mathbf{1+2\frac{m}{n}+\sqrt{\frac{5}{3}}}}$
$q=3$	$\frac{\mathbf{3\frac{m}{n}}}{\mathbf{1+3\frac{m}{n}+\sqrt{6}}}$	$\frac{4\frac{m}{n}}{1+4\frac{m}{n}+3\sqrt{\frac{3}{2}}}$	$\frac{2\frac{m}{n}}{1+2\frac{m}{n}+\frac{1}{2}\sqrt{7}}$
$q=4$	$\frac{3\frac{m}{n}}{1+3\frac{m}{n}+\sqrt{\frac{22}{3}}}$	$\frac{\mathbf{4\frac{m}{n}}}{\mathbf{1+4\frac{m}{n}+2\sqrt{3}}}$	$\frac{2\frac{m}{n}}{1+2\frac{m}{n}+\sqrt{\frac{5}{3}}}$
$q=5$	$\frac{3\frac{m}{n}}{1+3\frac{m}{n}+\sqrt{\frac{13}{2}}}$	$\frac{4\frac{m}{n}}{1+4\frac{m}{n}+\frac{1}{2}\sqrt{57}}$	$\frac{\mathbf{2\frac{m}{n}}}{\mathbf{1+2\frac{m}{n}+\frac{1}{2}\sqrt{\frac{13}{2}}}}$
$q=6$	$\frac{3\frac{m}{n}}{1+3\frac{m}{n}+\sqrt{6}}$	$\frac{4\frac{m}{n}}{1+4\frac{m}{n}+\sqrt{\frac{66}{5}}}$	$\frac{\mathbf{2\frac{m}{n}}}{\mathbf{1+2\frac{m}{n}+\sqrt{\frac{5}{3}}}}$
$q=7$	$\frac{3\frac{m}{n}}{1+3\frac{m}{n}+2\sqrt{\frac{5}{3}}}$	$\frac{\mathbf{4\frac{m}{n}}}{\mathbf{1+4\frac{m}{n}+\frac{5}{\sqrt{2}}}}$	$\frac{2\frac{m}{n}}{1+2\frac{m}{n}+3\sqrt{\frac{2}{11}}}$
$q=8$	$\frac{3\frac{m}{n}}{1+3\frac{m}{n}+2\sqrt{\frac{11}{7}}}$	$\frac{\mathbf{4\frac{m}{n}}}{\mathbf{1+4\frac{m}{n}+2\sqrt{3}}}$	$\frac{2\frac{m}{n}}{1+2\frac{m}{n}+\sqrt{\frac{5}{3}}}$

Table 2.1: Weakest RIP constant values for sparsity level $2 \leq q \leq 8$

where $\omega = 0, 1, \ldots$ is an integer variable. Table 2.1 shows with bold font which condition of the above four is the weakest for $2 \leq q \leq 8$. This table is equivalent to the table in proof of Theorem 1 in [51]. However, in [51] the table has exact values, while our Table 2.1 has functions depending on the ratio m/n instead. Using the same arguments as in proof of Theorem 1 in [51] and Table 2.1 we conclude that every q-sparse vector $x \in \mathbb{R}^n$ satisfying $Ax = \hat{b}$ is the unique solution of (2.5) if

$$\delta_{2q} < \frac{3\frac{m}{n}}{1 + 3\frac{m}{n} + \sqrt{6}}.$$

□

Comparing the two bounds of the RIP constants in Theorems 2.1 and 2.3 we observe that the former is smaller, see Figure 2.2. For the purpose of the proposed preconditioner, discussed in Section 6.4, the smaller bound on δ_{2q} in Theorem 2.3 results in tighter bounds of the spectral properties of the preconditioned systems. The former is an advantage of property (2.13) against RIP in (2.8), proved in Lemma 6.1. However, property (2.13) and Theorem 2.3 result in a limitation of the maximum sparsity q for which problem (2.5) guarantees an exact recovery of the sparsest solution of $Ax = \hat{b}$. Fortunately, both results in Theorems 2.1 and 2.3 are rather pessimistic. It has been shown in [16] that RIP conditions of the form (2.8) and their scaled versions (2.13) or (2.14) provide worst-case scenarios of δ_{2q} and consequently of the sparsity level q such that problem (2.5) guarantees exact sparse recovery. To support the former argument, we refer the reader to [45], where it is shown that for Gaussian measurement matrices the average max-

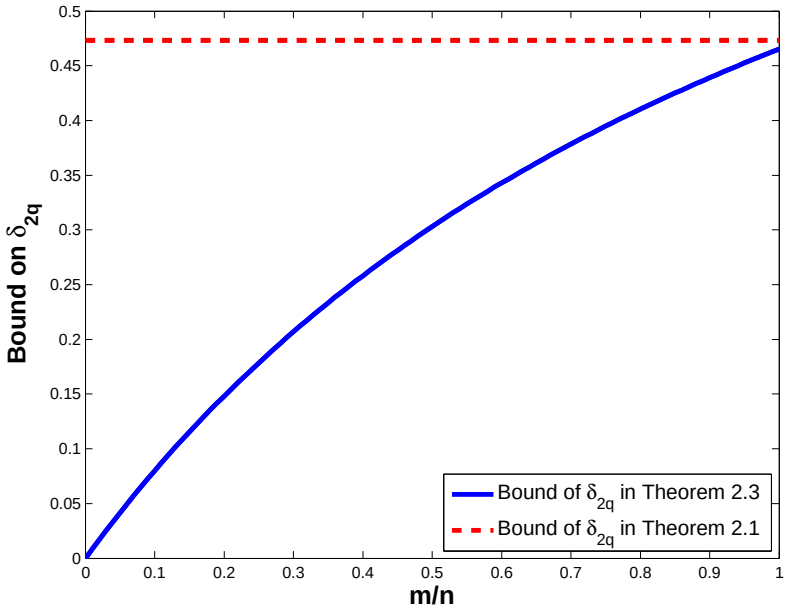


Figure 2.2: Comparison on the bounds of δ_{2q} constants from Theorems 2.1 and 2.3

imum sparsity level q that is guaranteed to be reconstructable by (2.5) is much greater than the one shown in Theorems 2.1 and 2.3. Moreover, it has been shown empirically in [45] that approximately the same result holds for various types of measurement matrices A, i.e., partial Fourier, partial Hadamard, Bernoulli etc. In Subsection 8.2.8 it is shown that the proposed algorithm satisfies approximately the average maximum sparsity q shown in [45]. Therefore, by replacing RIP (2.8) with property (2.13):

- improved bounds on the spectral properties of the preconditioned systems in Section 6.4 are obtained,
- a better approximation of matrix $B^{\mathsf{T}}B$ with a scaled diagonal $(m/n)I_q$ and
- the empirical average reconstruction properties as shown in Subsection 8.2.8 are maintained.

2.2.2 Examples of CS matrices

Currently known measurement matrices satisfying RIP with small number of measurements fall into two categories [99]: (i) random matrices with i.i.d. sub-Gaussian variables, e.g., normalized i.i.d. Gaussian or Bernoulli matrices; (ii) random partial bounded orthogonal matrices obtained by choosing m rows uniformly at random from a normalized $n \times n$ Fourier or Walsh-Hadamard transform matrices. Number of measurements required to satisfy the RIP property for both classes of matrices is given in the table below.

Although it follows from Table 2.2 that Gaussian matrices are optimal for sparse recovery, they have limited use in practice because many applications impose structure on the matrix. Furthermore, recovery algorithms are significantly

$m \times n$ measurement matrix	RIP regime	references
Gaussian	$m \geq Cq \log n$	[5, 99]
partial Fourier	$m \geq Cq \log^4 n$	[99]

Table 2.2: List of measurement matrices that have been proven to satisfy RIP

more efficient when the matrix admits a fast matrix-vector product. Due to the two former practical reasons, and since we are dealing with large-scale CS applications we limit ourselves to applications with measurement matrices A that

- are not stored explicitly,
- admit a low-cost matrix-vector product with A (e.g. $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$).

An important broad class of CS matrices comes from random sampling in bounded orthonormal systems. Partial Fourier matrix mentioned earlier is just one example of this type. Other examples are matrices related to systems of real trigonometric polynomials. For example, the partial Discrete Cosine Transform (DCT) and Discrete Sine Transform (DST) matrices, Haar wavelets and noiselets.

2.3 CS: coherent and redundant dictionaries

An extension of CS which was discussed in the previous section is to assume that $\hat{x}$ (a solution of (2.3)) has a sparse image through a redundant and coherent dictionary $W \in E^{n \times l}$, where $E = \mathbb{R}$ or $\mathbb{C}$ and $n \leq l$. More precisely, $W^* \hat{x}$, is sparse, i.e., it has only few non-zero components, where * denotes the conjugate transpose operator. If $W^* \hat{x}$ is sparse, under certain conditions on matrices A and W (discussed in Subsection 2.3.1) the optimal solution of the linear ℓ_1-analysis problem:

$$\text{minimize } \|W^* x\|_1 \quad \text{subject to:} \quad Ax = \hat{b},$$

is $\hat{x}$.

Frequently, measurements $\hat{b}$ might be contaminated with noise, i.e., one measures $b = \hat{b} + e$ instead, where e is a vector of noise, which is usually modelled as Gaussian with zero-mean and bounded Euclidean norm. In addition, in realistic applications $W^* \hat{x}$ might not be exactly sparse, but its mass might be concentrated only on few of its components, while the rest are rapidly decaying. In this case, (again under certain conditions on matrices A and W) the optimal solution of the following ℓ_1-analysis problem:

$$\text{minimize } \tau \|W^* x\|_1 + \frac{1}{2} \|Ax - b\|_2^2, \tag{2.15}$$

is proved to be a good approximation to $\hat{x}$ for some τ, see Subsection 2.3.1 for details.

2.3.1 Properties of CS matrices

There has been an extensive amount of literature studying conditions and properties of matrices A and W which guarantee recoverability of a good approximation of $\hat{x}$ by solving problem (2.15). For a thorough analysis we refer the reader to [24, 86]. The previously cited papers use a modified version of the well-known RIP [24], which is given below.

Definition 2.2. *The restricted isometry constant of a matrix $A \in \mathbb{R}^{m\times n}$ adapted to $W \in E^{n\times l}$ is defined as the smallest δ_q such that*

$$(1-\delta_q)\|Wz\|_2^2 \leq \|AWz\|_2^2 \leq (1+\delta_q)\|Wz\|_2^2,$$

for all at most q-sparse $z \in E^l$, where $E = \mathbb{R}$ or $\mathbb{C}$.

We will refer to Definition 2.2 as W-RIP. It is proved in Theorem 1.4 in [24] that if $W \in E^{n\times l}$ has orthonormal rows with $n \leq l$ and if A, W satisfy the W-RIP with $\delta_{2q} \leq 8.0e\text{-}2$, then the solution x_τ obtained by solving problem (2.15) satisfies:

$$\|x_\tau - \hat{x}\|_2 \leq C_0\|e\|_2 + C_1\frac{\|W^*x_\tau - (W^*\hat{x})_q\|_1}{\sqrt{q}}, \tag{2.16}$$

where $(W^*\hat{x})_q$ is the best q-sparse approximation of $W^*\hat{x}$, C_0 and C_1 are small constants and only depend on δ_{2q}. It is clear that $W^*\hat{x}$ must have $l-q$ rapidly decaying components, in order for $\|x_\tau - \hat{x}\|_2$ to be small and the reconstruction to be successful. Isotropic Total Variation (iTV) is a special case of ℓ_1-analysis where matrix W does not have orthonormal rows, hence, result (2.16) does not hold. For iTV there are no conditions on δ_{2q} such that a good reconstruction is assured. However, there exist results which directly impose restrictions on the number of measurements m, see Theorems 2, 5 and 6 in [86]. Briefly, in these theorems it is mentioned that if $m \geq q\log(n)$ linear measurements are acquired for which matrices A and W satisfy the W-RIP for some $\delta_q < 1$, then, similar reconstruction guarantees as in (2.16) are obtained for iTV. Based on the previously mentioned results regarding reconstruction guarantees it is natural to assume that for iTV a similar condition applies, i.e., $\delta_{2q} < 1/2$. Hence, we make the following assumption.

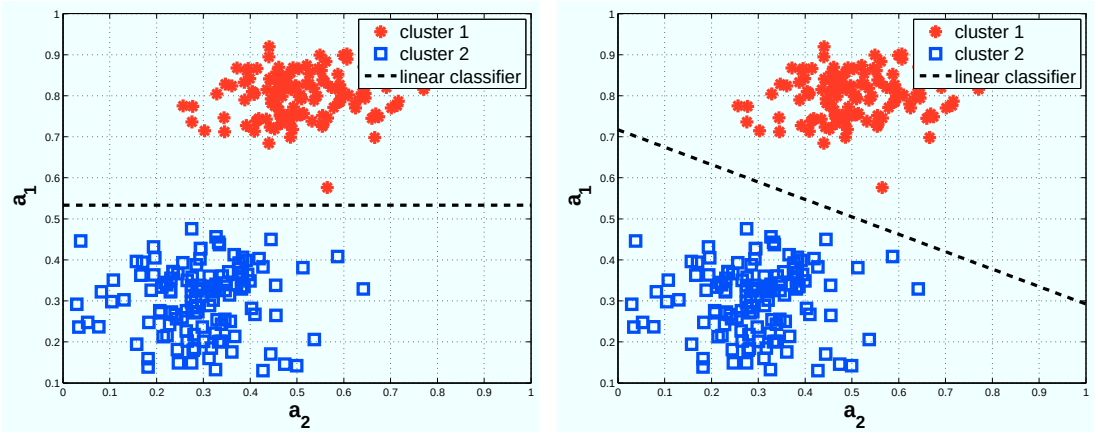
Assumption 2.1. *The number of non-zero components of W^*x_τ, denoted by q, and the dimensions l, m, n are such that matrices A and W satisfy W-RIP for some $\delta_{2q} < 1/2$.*

Another property of matrix A is the near orthogonality of its rows. Indeed many applications in CS use matrices A that satisfy:

$$\|AA^\intercal - I_m\|_2 \leq \delta, \tag{2.17}$$

with a small constant $\delta \geq 0$. Finally, we will make use of the following assumption:

$$\mathrm{Ker}(W^*) \cap \mathrm{Ker}(A) = \{0\}. \tag{2.18}$$



(a) ℓ_1-LR (b) ℓ_2 LR

Figure 2.3: Demonstration of the purpose of the ℓ_1-norm for binary classification problems. For details see text in Section 2.4

This is a commonly used and realistic assumption derived from the optimization literature, see for example [111], which is necessary for problem (2.15) to have a unique solution.

2.3.2 Examples

An example of W being redundant and coherent with orthonormal rows is the curvelet frame where an image is assumed to have an approximately sparse representation [23]. Moreover, for radar and sonar systems it is frequent that Gabor frames are used in order to reconstruct pulse trains from CS measurements [81]. For more applications a small survey is given in [24]. iTV is another application of CS which exploits the fact that digital images frequently have slowly varying pixels, except along edges. This property implies that digital images with respect to the discrete nabla operator, i.e., local differences of pixels, are approximately sparse. For iTV applications, matrix $W \in \mathbb{C}^{n\times n}$ is square, complex and rank-deficient with $rank(W) = n - 1$. An alternative to iTV is ℓ_1-analysis, where matrix W is a Haar wavelet transform. However, a more pleasant to the eye reconstruction is obtained by solving the iTV problem compared to the ℓ_1-analysis problem, see [86].

2.4 Linear support vector machine

In LSVM the goal is to find a linear function that separates the given data into two clusters [38]. Let us assume we are given m data points (a_i, b_i), where $a_i \in \mathbb{R}^n$ and $b_i \in \{-1, 1\}$ $\forall i = 1, 2, \cdots, m$. An example of LSVM is shown in Figure 2.3a. In this figure we plot $m = 250$ points $a_i \in \mathbb{R}^2$ and separate them into two clusters based on their value b_i: -1 (squares), or 1 (stars). The aim is to find the dashed line in Figure 2.3a which separates the points (a_i, b_i) $\forall i = 1, 2, \cdots, m$ into two clusters.

More precisely, in LSVM we are looking for $x \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, the coefficients and the intercept of a linear function $x^\intercal a - \beta$, where a are the independent variables. The linear function $x^\intercal a - \beta$ should satisfy $x^\intercal a - \beta \geq 1$ for all a_i's with $b_i = 1$ and additionally, it should satisfy $x^\intercal a - \beta \leq -1$ for all a_i's with $b_i = -1$. The previous constraints simply state that the function $x^\intercal a - \beta$ guarantees the correct classification of the given data (a_i, b_i) $\forall i = 1, 2, \cdots, m$. There are many x and β that satisfy this, however, in LSVM we choose the x and β that maximize the distance of the two hyperplanes: $x^\intercal a - \beta = 1$ and $x^\intercal a - \beta = -1$. This means that we are looking for x that maximizes $2/\|x\|_2$ (or minimizes $\|x\|_2$) subject to the correct classification of the given data.

Other norms can also be considered. In our experiments we choose the ℓ_1-norm in order to enforce sparsity in x. By using the ℓ_1-norm we are able to capture the separability of the data into clusters based only on few features, instead of n features. For example see Figure 2.3a, where obviously the data can be clustered based only on the value of the first feature $[a_i]_1$ $\forall i$, hence the optimal hyperplane depends only on $[a]_1$. On the other hand when the ℓ_2 norm is used then the optimal hyperplane depends on both features $[a]_1$ and $[a]_2$ and the feature reduction property is lost.

The coefficients x and the intercept β are obtained by solving:

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to:} & b_i(x^\intercal a_i - \beta) \geq 1 \quad \forall i = 1, 2, \cdots, m. \end{array}$$

The objective function represents the distance of the two hyperplanes: $x^\intercal a_i - \beta = 1$ and $x^\intercal a_i - \beta = -1$. The constraints represent the restriction that the obtained function $x^\intercal a - \beta$ should cluster correctly the input data (a_i, b_i) $\forall i = 1, 2, \cdots, m$.

In case that the data are not linearly separable, unlike our example displayed in Figure 2.3, then we can add a penalty term to control the error of the misclassified data. In particular, we introduce some extra variables ξ_i for $i = 1, 2, \cdots, m$ and replace the previous problem with:

$$\begin{array}{ll} \text{minimize} & \tau\|x\|_1 + \sum_{i=1}^m \xi_i \\ \text{subject to:} & b_i(x^\intercal a_i - \beta) \geq 1 - \xi_i \quad \forall i = 1, 2, \cdots, m \\ & \xi_i \geq 0. \end{array}$$

Parameter $\tau > 0$ controls the emphasis on minimizing the misclassification error or the distance between the hyperplanes. This problem can be reformulated as:

$$\text{minimize} \quad \tau\|x\|_1 + \sum_{i=1}^m p(b_i(x^\intercal a_i - \beta)),$$

where $p : \mathbb{R} \to \mathbb{R}$ is a penalty function around one. In this reformulation we replaced the constraints with penalty functions in the objective function. There are a few penalty functions that one can consider. The most important are:

- hinge loss: $p(\omega) = \max(0, 1 - \omega)$ [122], which is not differentiable everywhere,

- ℓ_2 loss: $p(\omega) = (\max(0, 1 - \omega))^2$ [122], which is first-order differentiable and

- LR: $p(\omega) = \log(1 + e^{-\omega})$ [122], which is smooth.

The three penalty functions are demonstrated in Figure 2.4.

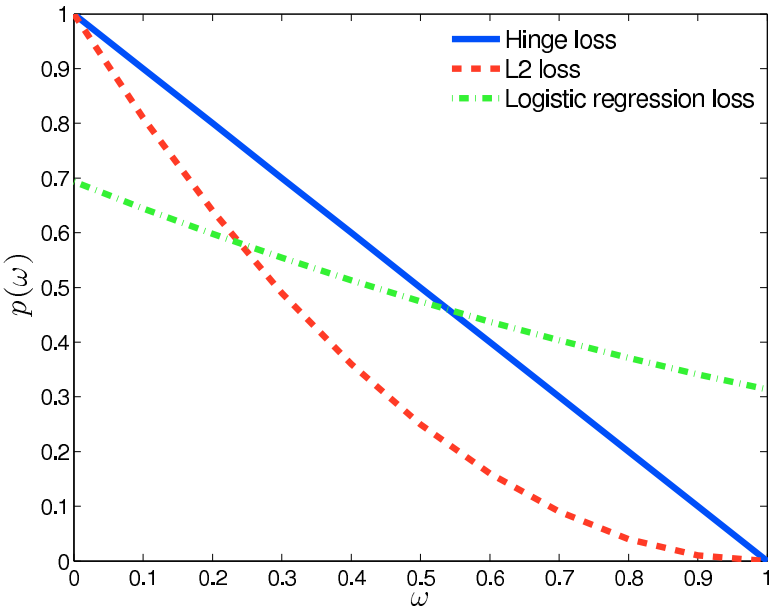


Figure 2.4: Demonstration of penalty functions for linear support vector machine

Chapter 3

Smoothing and Primal-Dual Reformulation

In this chapter we discuss how the Legendre-Fenchel transform can be used to obtain a smooth approximation of a non-smooth function as well as a primal-dual reformulation of the problem of interest. We begin by introducing the Legendre-Fenchel transform. Then, the Moreau proximal smoothing technique [84] is described using the Legendre-Fenchel transform and an example is presented. Generalizations of the Moreau proximal smoothing technique can be found in [88] and [7]. Finally, we discuss the primal-dual reformulation of problems that we are concerned with.

Part of the material in this chapter has been presented in [41] and [53].

3.1 The Legendre-Fenchel transform

The Legendre-Fenchel transform is a way to represent a convex function [123]. The Legendre-Fenchel transform ψ^* of a convex function $\psi : \mathbb{R}^n \to \mathbb{R}$ is:

$$\psi^*(y) = \sup_x y^\intercal x - \psi(x). \tag{3.1}$$

The domain of ψ^* is the set of all slopes of all tangent functions of ψ, i.e. the sub-gradients of ψ. The image of ψ^* is the set of all negative intercepts of tangent functions of ψ. To demonstrate the previous assume for simplicity that function $\psi : \mathbb{R}^n \to \mathbb{R}$ is uni-variate and convex. A tangent function of ψ at a point x' is given by $\psi(x; x') = s(x - x') + \psi(x')$, where $s \in \partial\psi(x')$ is a sub-gradient of ψ at point x'. The intercept of function $\psi(x; x')$ is $\psi(0, x') = -sx' + \psi(x')$. Let y in (3.1) be the sub-gradient s of ψ at x', then the supremum in (3.1) is given for $x = x'$ and has value $-\psi(0, x')$. In Figure 3.1 two examples of tangent functions of a non differentiable function ψ are plotted. Given the slope s_1 or s_2 of one of the two tangent functions the Legendre-Fenchel transform returns the negative of the corresponding intercept, where the intercept is denoted by the red or the cyan point in Figure 3.1, respectively.

The Legendre-Fenchel transform enjoys numerous properties [123] and two of them will help us in the discussion of the smoothing technique in the next section.

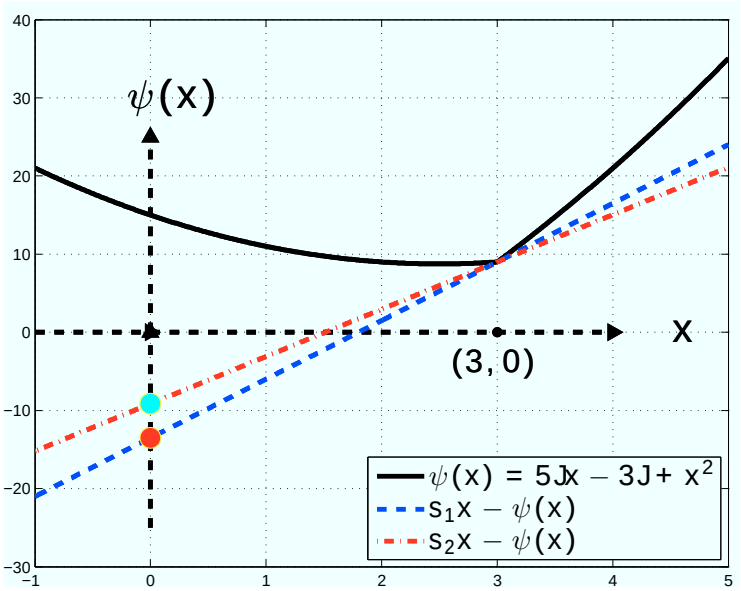


Figure 3.1: Non differentiable convex function ψ and two tangent linear functions at point $(3, 0)$ of non differentiability

The two properties are:

- For a closed convex function ψ, the Legendre-Fenchel transform ψ^{**} of ψ^* is equal to ψ:
$$\psi^{**} = \psi = \sup_y y^\intercal x - \psi^*(y) \tag{3.2}$$
- and
$$\nabla\psi(x) = \arg\max_y y^\intercal x - \psi^*(y). \tag{3.3}$$

3.2 Moreau proximal smoothing

Property (3.3) implies that the convex function ψ is differentiable if and only if the arg max operator returns a unique solution. Therefore function ψ is differentiable if its Legendre-Fenchel transform ψ^* is strongly convex. The Moreau smoothing technique is based on the idea that we can add a strongly convex quadratic function $\mu/2\|y\|_2^2$ ($\mu > 0$) to ψ^* in order to enforce a unique solution in (3.3). Using property (3.2) a smooth approximation ψ_μ of function ψ is obtained:

$$\psi_\mu(x) = \sup_y y^\intercal x - \psi^*(y) - \frac{\mu}{2}\|y\|_2^2. \tag{3.4}$$

The smaller μ is the better the approximation of function ψ_μ to ψ:

$$\psi_\mu(x) \leq \psi(x) \leq \psi_\mu(x) + \mu C \quad \forall x \in \mathbb{R}^n,$$

where C is a positive bounded constant, see page 132 in [88] for details.

Other strongly convex functions can be added which result in different smooth approximations ψ_μ. In the next subsection we present such an example.

3.2.1 A smoothing example

We now present an example of Moreau smoothing technique for function $h(x) = \|\Omega^\intercal x\|_2$, where $\Omega \in \mathbb{R}^{n\times p}$. The Legendre-Fenchel transform of function h is:

$$h^*(y) = \sup_x y^\intercal x - \|\Omega^\intercal x\|_2. \tag{3.5}$$

The optimality conditions of problem (3.5) are:

$$y = \Omega g, \quad g \in \mathbb{R}^p \quad \|g\|_2 \leq 1. \tag{3.6}$$

Substituting the previous in (3.5) we get:

$$h^*(\Omega g) = \begin{cases} \sup_x g^\intercal \Omega^\intercal x - \|\Omega^\intercal x\|_2, & \text{if } \|g\|_2 \leq 1 \\ +\infty, & \text{otherwise,} \end{cases}$$

which is equivalent to:

$$h^*(\Omega g) = \begin{cases} 0, & \text{if } \|g\|_2 \leq 1 \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.7}$$

Using (3.2), (3.6) and (3.7) we have that:

$$h(x) = \sup_y y^\intercal x - h^*(y) = \sup_{\|g\|_2\leq 1} g^\intercal \Omega^\intercal x. \tag{3.8}$$

Clearly the supremum in (3.8) does not correspond to a unique solution g for every x. Hence, based on property (3.3) function h is not differentiable.

To make function h smooth we will regularize h^* with a strongly convex function. We consider two strongly convex functions:

$$\frac{\mu}{2}\|g\|_2^2, \tag{3.9}$$

and

$$\mu - \mu(1 - \|g\|^2)^{\frac{1}{2}}. \tag{3.10}$$

By subtracting (3.9) from (3.8) we obtain:

$$h_\mu(x) = \sup_{\|g\|_2\leq 1} g^\intercal \Omega^\intercal x - \frac{\mu}{2}\|g\|_2^2 = \begin{cases} \frac{\mu}{2}\|\Omega^\intercal x\|_2^2, & \text{if } \|\Omega^\intercal x\|_2 \leq \mu \\ \|\Omega^\intercal x\|_2 - \frac{\mu}{2}, & \text{if } \|\Omega^\intercal x\|_2 \geq \mu. \end{cases} \tag{3.11}$$

By subtracting (3.10) from (3.8) we obtain:

$$\begin{aligned} h_\mu(x) &= \sup_{\|g\|_2\leq 1} g^\intercal \Omega^\intercal x + \mu(1 - \|g\|^2)^{\frac{1}{2}} - \mu \\ &= (\mu^2 + \|\Omega^\intercal x\|_2^2)^{\frac{1}{2}} - \mu. \end{aligned} \tag{3.12}$$

The difference between (3.11) and (3.12) is that the latter has derivatives of all orders, while the former is only first-order differentiable.

3.2.2 Huber and pseudo-Huber functions

We now extend the previous example to function $\psi(x) = \|W^*x\|_1$, where $W \in \mathbb{E}^{n\times l}$ and $E = \mathbb{R}$ or $\mathbb{C}$. Let us define $\Omega_i = [ReW_i, ImW_i] \in \mathbb{R}^{n\times 2}$, where $Re(\cdot)$ takes a complex input and returns its real part, for simplification we use $Re(\cdot)$ without the parenthesis.

First, we rewrite function ψ:

$$\psi(x) = \|W^*x\|_1 = \sum_{i=1}^{l} |W_i^*x| = \sum_{i=1}^{l} \|\Omega_i^\intercal x\|_2. \tag{3.13}$$

Using the smoothing example of the ℓ_2-norm from Subsection 3.2.1 we obtain the following approximation:

$$\psi_\mu(x) = \sum_{i=1}^{l} h_\mu^i(x), \tag{3.14}$$

where $h_\mu^i(x)$ can be either (3.11) or (3.12) using the corresponding Ω_i matrix. In particular, using the approximation (3.11) in (3.14) we obtain:

$$\psi_\mu(x) = \sum_{i=1}^{l} \left\{ \begin{array}{ll} \frac{\mu}{2}\|\Omega_i^\intercal x\|_2^2, & \text{if } \|\Omega_i^\intercal x\|_2 \leq \mu \\ \|\Omega_i^\intercal x\|_2 - \frac{\mu}{2}, & \text{if } \|\Omega_i^\intercal x\|_2 \geq \mu \end{array} \right\}. \tag{3.15}$$

Using the approximation (3.12) in (3.14) we obtain:

$$\psi_\mu(x) = \sum_{i=1}^{l} ((\mu^2 + \|\Omega_i^\intercal x\|_2^2)^{\frac{1}{2}} - \mu). \tag{3.16}$$

Finally, from the definition of Ω_i we have that (3.15) is equivalent to:

$$\psi_\mu(x) = \sum_{i=1}^{l} \left\{ \begin{array}{ll} \frac{\mu}{2}|W_i^*x|^2, & \text{if } |W_i^*x| \leq \mu \\ |W_i^*x| - \frac{\mu}{2}, & \text{if } |W_i^*x| \geq \mu \end{array} \right\}, \tag{3.17}$$

which is known as Huber function. While (3.16) is equivalent to:

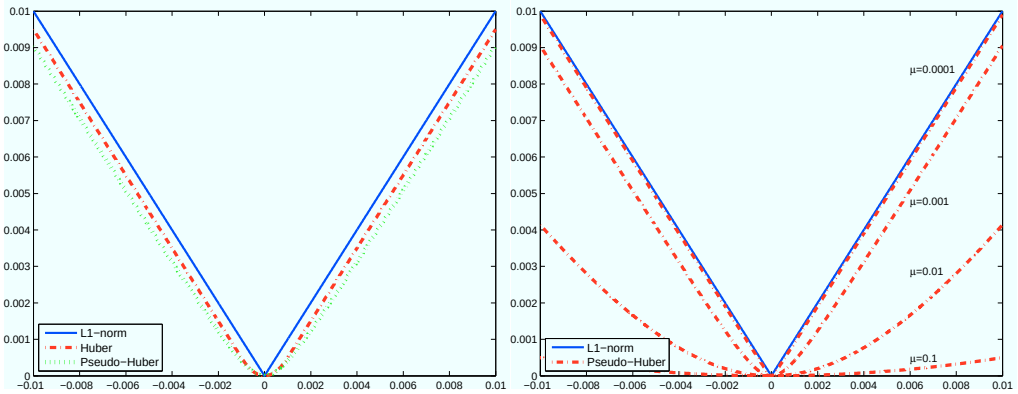
$$\psi_\mu(x) = \sum_{i=1}^{l} ((\mu^2 + |W_i^*x|^2)^{\frac{1}{2}} - \mu), \tag{3.18}$$

which is known as pseudo-Huber function.

Observe that the Huber function is only first-order differentiable, while the pseudo-Huber function is smooth. A comparison of the three functions ℓ_1-norm, Huber and Pseudo-Huber is presented in Figure 3.2.

3.3 Primal-dual reformulation

The Legendre-Fenchel transform can be used to obtain a primal-dual reformulation of problem (1.1). In particular, the non-smooth function ψ in (1.1) can be



(a) ℓ_1-norm, Huber, Pseudo-Huber functions (b) Pseudo-Huber function for $\mu \to 0$

Figure 3.2: Comparison of Huber and pseudo-Huber functions with the ℓ_1-norm. **Fig.3.2a** shows the quality of approximation for Huber and pseudo-Huber functions. **Fig.3.2b** shows how pseudo-Huber function converges to the ℓ_1-norm as $\mu \to 0$

replaced with its equivalent form (3.2). Hence, problem (1.1) is replaced with:

$$\underset{x}{\text{minimize}}[(\sup_{y} \tau y^\intercal x - \tau\psi^*(y)) + \varphi(x)], \tag{3.19}$$

where y are the dual variables. Problem (3.19) is a primal-dual reformulation of (1.1) in the sense that (3.2) is a dual reformulation of ψ. In what follows we discuss an example regarding pseudo-Huber regularized problems.

3.3.1 Primal-dual reformulation of pseudo-Huber regularized problems

We have the following pseudo-Huber regularized problem:

$$\text{minimize } \tau\psi_\mu(x) + \varphi(x), \tag{3.20}$$

where ψ_μ is the pseudo-Huber function defined in (3.16). Using (3.12) we get the following equivalent form of (3.16):

$$\psi_\mu(x) = \sum_{i=1}^{l}(\sup_{g_i \in \mathbb{R}^2, \|g_i\|_2 \leq 1} g_i^\intercal \Omega_i^\intercal x + \mu(1 - \|g_i\|^2)^{\frac{1}{2}} - \mu).$$

Using the fact that every two dimensional vector g_i can be written as a complex number, and using the definition of $\Omega_i = [ReW_i, ImW_i] \in \mathbb{R}^{n\times 2}$ we get:

$$\psi_\mu(x) = \sup_{g \in \mathbb{C}^l, \|g\|_\infty \leq 1} Re(\bar{g}^* W^*)x + \sum_{i=1}^{l} \left(\mu(1 - |g_i|^2)^{1/2} - \mu\right),$$

where g are the dual variables and the bar superscript denotes the complex conjugate. Therefore, the primal-dual reformulation is obtained by replacing ψ_μ with its equivalent form ψ_μ^{**}:

$$\underset{x}{\text{minimize}} \left[\left(\sup_{g \in \mathbb{C}^l, \|g\|_\infty \leq 1} \tau Re(\bar{g}^* W^*) x + \tau \sum_{i=1}^{l} \left(\mu(1 - |g_i|^2)^{1/2} - \mu \right) \right) + \varphi(x) \right]. \tag{3.21}$$

In this thesis we will develop methods that solve such primal-dual reformulation. We will discuss in Subsection 4.4 the reasons why solving the primal-dual problem results in a numerically more stable method.

Chapter 4

Primal-Dual Newton Conjugate Gradients (pdNCG)

In this chapter we discuss the primal-dual Newton Conjugate Gradients (pdNCG) method for ℓ_1-regularized problems:

$$\text{minimize } f_\tau := \tau\|x\|_1 + \varphi(x), \tag{4.1}$$

where $x \in \mathbb{R}^n$, $\tau > 0$. This method has been first presented in [32]. Our contribution concerns the convergence analysis of pdNCG. We show that the analysis of pdNCG can be performed in a variable metric using an important property of CG. The variable metric is the standard Euclidean norm scaled by an approximation of the second-order derivative at every iteration of pdNCG. Based on the variable metric we give a complete analysis of pdNCG, i.e., proof of global convergence, global and local convergence rates, local region of fast convergence rate and worst-case iteration complexity.

4.1 Assumptions

The following three assumptions are made.

- The function φ is twice differentiable, and
- at any x its second derivative $\nabla^2\varphi$ is uniformly bounded:

  $$\lambda_n I \preceq \nabla^2\varphi(x) \preceq \lambda_1 I, \tag{4.2}$$

  with $0 < \lambda_n \leq \lambda_1$. This property implies strong convexity of φ and it is heavily-used in our analysis. However, in practice the method can be trivially modified to solve problems for which φ is convex. For example, see Subsection 8.4.
- The second derivative of φ is Lipschitz continuous:

  $$\|\nabla^2\varphi(y) - \nabla^2\varphi(x)\| \leq L_\varphi\|y - x\|, \tag{4.3}$$

  for any x, y, where $L_\varphi \geq 0$ is the Lipschitz constant.

4.2 Smoothing

The non-smooth ℓ_1-norm is approximated by the pseudo-Huber function (3.18), where W in (3.18) is the identity matrix. Hence, problem (4.1) is replaced by

$$\text{minimize } f_\tau^\mu(x) := \tau\psi_\mu(x) + \varphi(x), \tag{4.4}$$

where

$$\psi_\mu(x) = \sum_{i=1}^{n}((\mu^2 + x_i^2)^{\frac{1}{2}} - \mu).$$

The advantages of such an approach are:

- Availability of second-order information of function φ owing to the differentiability of the pseudo-Huber function.

- Opening the door to using iterative methods to compute descent directions which take into account the curvature of the problem, such as Conjugate Gradients (CG).

There is an obvious cost which comes along with the above benefits, and that is the approximate nature of the pseudo-Huber function. There is a concern that in case that a very accurate solution is required, the pseudo-Huber function may be unable to deliver it. In theory, since the quality of the approximation is controlled by the smoothing parameter μ the pseudo-Huber function can recover any level of accuracy under the condition that sufficiently small μ is chosen. The reader is referred to Section 3.2 for a perturbation analysis when the ℓ_1-norm is replaced by the Pseudo-Huber function. In practice a very small parameter μ might increase the conditioning of the linear algebra of the solver. However, we shall provide numerical evidence that even when μ is set to small values, the proposed method behaves well and remains very efficient.

The material in this chapter has been based on [53], in which we presented the pdNCG method.

4.3 Derivatives

Let $diag(\cdot)$ be the function that takes as input a vector and outputs a diagonal square matrix with the vector in the main diagonal. The gradient of the pseudo-Huber function ψ_μ is:

$$\nabla\psi_\mu(x) = Dx, \tag{4.5}$$

where $D := diag([D_1, D_2, \cdots, D_n])$ with

$$D_i := (\mu^2 + x_i^2)^{-\frac{1}{2}} \quad \forall i = 1, 2, \cdots, n, \tag{4.6}$$

and the Hessian is:

$$\nabla^2\psi_\mu(x) = \mu^2 D^3. \tag{4.7}$$

The gradient of f_τ^μ in (4.4) is given by

$$\nabla f_\tau^\mu(x) = \tau \nabla \psi_\mu(x) + \nabla \varphi(x),$$

and the Hessian matrix of f_τ^μ is:

$$\nabla^2 f_\tau^\mu(x) = \tau \nabla^2 \psi_\mu(x) + \nabla^2 \varphi(x).$$

4.4 Primal-dual formulation and optimality conditions

From Subsection 3.3.1 we have that the primal-dual formulation of problem (4.4) is:

$$\text{minimize} \sup_{g \in \mathbb{R}^n, \|g\|_\infty \leq 1} \tau g^\intercal x + \tau \sum_{i=1}^{n} \left(\mu(1 - g_i^2)^{1/2} - \mu\right) + \varphi(x). \tag{4.8}$$

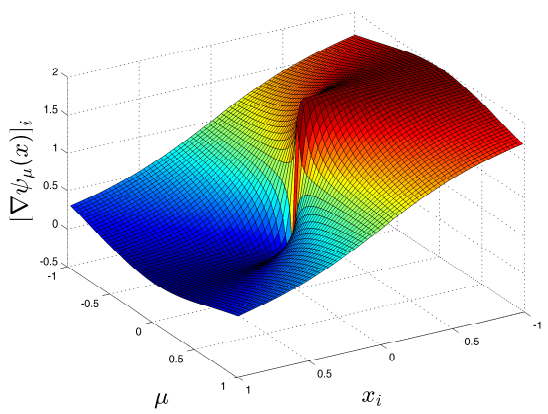
It can be obtained from (3.21) by setting W to be the identity matrix and restricting g to the real variables. The optimality conditions for the primal-dual problem (4.8) are:

$$\begin{aligned} \tau g + \nabla \varphi(x) &= 0, \\ D^{-1} g &= x, \end{aligned} \tag{4.9}$$

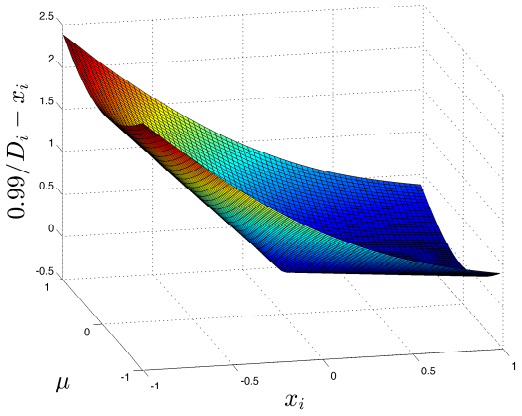
where D is defined in (4.6). Notice for conditions (4.9) that the constraint $\|g\|_\infty \leq 1$ in (4.8) is redundant since any x and g that satisfy (4.9) also satisfy this constraint (see definition of matrix D in (4.6)). Hence, the constraint has been dropped.

The first-order optimality conditions of the primal problem (4.1) are $\nabla f_\tau^\mu(x) = \tau \nabla \psi_\mu(x) + \nabla \varphi(x) = 0$. Therefore, one could simply apply a Newton-CG method in order to find a root of this equation. However, in a series of papers [31, 32] it has been noted that the linearization of $\nabla \psi_\mu$ for Newton-CG method might be a poor approximation of $\nabla \psi_\mu$ close to the optimal solution. To deal with this problem the authors in [32] suggested to solve the optimality conditions (4.9) of the primal-dual problem (4.8).

Let $[\cdot]_{ij}$ be the operator that returns the element at row i and column j of the input matrix, similarly, let $[\cdot]_i$ be the operator that returns the element at position i of the input vector. The idea behind the previous technique is that the linearization of the second equations in (4.9), i.e., $y_i/[D]_{ii} - x_i = 0$ $\forall i = 1, 2, \ldots, m$, is of much better quality than the linearization of $[\nabla \psi_\mu(x)]_i$ for $\mu \approx 0$ and $x_i \approx 0$; a scenario that is unavoidable since for small μ the optimal solution of (4.1) is expected to be approximately sparse. To see why this is true, observe that for small μ and $x_i \approx 0$, the gradient $[\nabla \psi_\mu(x)]_i$ becomes close to singular and its linearization is expected to be inaccurate. On the other hand, $y_i/[D]_{ii} - x_i$ as a function of x_i is not singular for $\mu \approx 0$ and $x_i \approx 0$, hence, its linearization is expected to be more accurate. To visualize such a situation observe in Figure 4.1 the plotted surfaces of $[\nabla \psi_\mu(x)]_i$ and $y_i/[D]_{ii} - x_i$ as a function of x_i and μ when $y_i = 0.99$. Notice, that function $[\nabla \psi_\mu(x)]_i$ becomes locally to $\mu \approx 0$ and $x_i \approx 0$ nearly singular, while $0.99/[D]_{ii} - x_i$ is not. Empirical



(a) surface of partial derivative of pseudo-Huber function as a function of x_i and μ



(b) surface of $0.99(\mu^2 + x_i^2)^{1/2} - x_i$ as a function of x_i and μ

Figure 4.1: Visualization of $[\nabla\psi_\mu(x)]_i$ and $0.99/[D]_{ii} - x_i$ as function of x_i and μ

justification for this is also given in Section 3 in [32], it is also worth mentioning that our empirical experience confirms the results of the previous paper.

4.5 Useful bounds

The next lemma shows that the Hessian of the pseudo-Huber function $\psi_\mu(x)$ is bounded.

Lemma 4.1. *The Hessian matrix $\nabla^2\psi_\mu$ satisfies:*

$$0I_n \prec \nabla^2\psi_\mu(x) \preceq \frac{1}{\mu}I_n \ \forall x,$$

where I_n is the identity matrix of size $n \times n$.

Proof. The result follows easily by observing that $0 < (\mu^2 + x_i^2)^{-\frac{3}{2}} \leq 1/\mu^3$ for any x_i, $i = 1, 2 \ldots, n$. □

The next lemma shows that the Hessian matrix of the pseudo-Huber function is Lipschitz continuous.

Lemma 4.2. *The Hessian matrix $\nabla^2\psi_\mu$ is Lipschitz continuous:*

$$\|\nabla^2\psi_\mu(y) - \nabla^2\psi_\mu(x)\| \leq \frac{1}{\mu^2}\|y - x\| \ \forall x, y.$$

Proof.

$$\begin{aligned}\|\nabla^2\psi_\mu(y) - \nabla^2\psi_\mu(x)\| &= \Big\| \int_0^1 \frac{d\nabla^2\psi_\mu(x + s(y-x))}{ds} ds \Big\| \\ &\leq \int_0^1 \Big\| \frac{d\nabla^2\psi_\mu(x + s(y-x))}{ds} \Big\| ds \end{aligned} \tag{4.10}$$

where $d\nabla^2\psi_\mu(x+s(y-x))/ds$ is a diagonal matrix with each diagonal component, $i = 1, 2, \ldots n$, given by

$$\Big[\frac{d\nabla^2\psi_\mu(x+s(y-x))}{ds}\Big]_{ii} = \frac{-3\mu^2(x_i+s(y_i-x_i))(y_i-x_i)}{(\mu^2+(x_i+s(y_i-x_i))^2)^{\frac{5}{2}}}.$$

Using the previous observation we have that:

$$\Big\|\frac{d\nabla^2\psi_\mu(x+s(y-x))}{ds}\Big\| = \max_{i=1,2,\ldots,n}\Big|\Big[\frac{d\nabla^2\psi_\mu(x+s(y-x))}{ds}\Big]_{ii}\Big| \tag{4.11}$$

Moreover, we have:

$$\begin{aligned}\Big|\Big[\frac{d\nabla^2\psi_\mu(x+s(y-x))}{ds}\Big]_{ii}\Big| &= \Big|\frac{-3\mu^2(x_i+s(y_i-x_i))(y_i-x_i)}{(\mu^2+(x_i+s(y_i-x_i))^2)^{\frac{5}{2}}}\Big| \\ &= \Big|\frac{-3\mu^2(x_i+s(y_i-x_i))}{(\mu^2+(x_i+s(y_i-x_i))^2)^{\frac{5}{2}}}\Big||(y_i-x_i)| \end{aligned}\tag{4.12}$$

where the first absolute value in (4.12) has a maximum at $(\mu-2x_i)/2(y_i-x_i)$, which gives:

$$\Big|\frac{-3\mu^2(x_i+s(y_i-x_i))}{(\mu^2+(x_i+s(y_i-x_i))^2)^{\frac{5}{2}}}\Big| \le \frac{48}{25\sqrt{5}\mu^2} < \frac{1}{\mu^2}. \tag{4.13}$$

Combining (4.12) and (4.13) we get:

$$\Big|\Big[\frac{d\nabla^2\psi_\mu(x+s(y-x))}{ds}\Big]_{ii}\Big| \le \frac{1}{\mu^2}|y_i-x_i|. \tag{4.14}$$

Replacing (4.14) in (4.11) and using the fact that $\|\cdot\|_\infty \le \|\cdot\|$ we get:

$$\Big\|\frac{d\nabla^2\psi_\mu(x+s(y-x))}{ds}\Big\| \le \frac{1}{\mu^2}\|y-x\|.$$

Replacing the above expression in (4.10) and calculating the integral we arrive at the desired result. □

The next lemma shows that the gradient of the pseudo-Huber function is Lipschitz continuous.

Lemma 4.3. *The gradient $\nabla\psi_\mu$ is Lipschitz continuous:*

$$\|\nabla\psi_\mu(y)-\nabla\psi_\mu(x)\| \le \frac{1}{\mu}\|y-x\| \;\forall x, y.$$

Proof. Using the fundamental theorem of calculus, as in proof of Lemma 4.2, and Lemma 4.1 it is easy to show the result. □

Using (9.3) and Lemma 4.1 we get the following bounds on the Hessian matrix

of f_τ^μ:

$$\lambda_n I_n \prec \nabla^2 f_\tau^\mu(x) \preceq \Big(\frac{\tau}{\mu} + \lambda_1\Big) I_n \ \forall x. \tag{4.15}$$

Lemma 4.4. *For any x and $\tilde{x}$ (the minimizer of f_τ^μ), the following hold:*

$$\frac{1}{2\Big(\frac{\tau}{\mu} + \lambda_1\Big)} \|\nabla f_\tau^\mu(x)\|^2 \leq f_\tau^\mu(x) - f_\tau^\mu(\tilde{x}) \leq \frac{1}{2\lambda_n} \|\nabla f_\tau^\mu(x)\|^2$$

and

$$\|x - \tilde{x}\| \leq \frac{2}{\lambda_n} \|\nabla f_\tau^\mu(x)\|.$$

Proof. The right hand side of the first inequality is proved on page 460 of [18]. The left hand side of the first inequality is proved by using smoothness of f_τ^μ:

$$f_\tau^\mu(y) \leq f_\tau^\mu(x) + \nabla f_\tau^\mu(x)^\intercal (y - x) + \frac{\frac{\tau}{\mu} + \lambda_1}{2} \|y - x\|^2$$

and defining $\tilde{y} = x - \frac{1}{\frac{\tau}{\mu} + \lambda_1} \nabla f_\tau^\mu(x)$, we get:

$$f_\tau^\mu(x) - f_\tau^\mu(\tilde{x}) \geq f_\tau^\mu(x) - f_\tau^\mu(\tilde{y}) \geq \frac{1}{2\Big(\frac{\tau}{\mu} + \lambda_1\Big)} \|\nabla f_\tau^\mu(x)\|^2.$$

The last inequality is proved on page 460 of [18]. □

The following lemma shows that the Hessian matrix $\nabla^2 f_\tau^\mu$ is Lipschitz continuous. In this lemma, L_φ is defined in (4.3).

Lemma 4.5. *The function $\nabla^2 f_\tau^\mu$ is Lipschitz continuous:*

$$\|\nabla^2 f_\tau^\mu(y) - \nabla^2 f_\tau^\mu(x)\| \leq L_{f_\tau^\mu} \|y - x\| \ \forall x, y,$$

where $L_{f_\tau^\mu} := \frac{\tau}{\mu^2} + L_\varphi$.

Proof. Using Lemma 4.2 and (4.3) we have:

$$\begin{aligned} \|\nabla^2 f_\tau^\mu(y) - \nabla^2 f_\tau^\mu(x)\| &\leq \tau \|\nabla^2 \psi_\mu(y) - \nabla^2 \psi_\mu(x)\| + \|\nabla^2 \varphi(y) - \nabla^2 \varphi(x)\| \\ &\leq \Big(\frac{\tau}{\mu^2} + L_\varphi\Big) \|y - x\|. \end{aligned}$$

The Lipschitz constant of $\nabla^2 f_\tau^\mu$ is therefore $L_{f_\tau^\mu} := \frac{\tau}{\mu^2} + L_\varphi$. □

The next lemma shows how well the second-order Taylor expansion of f_τ^μ approximates the function f_τ^μ.

Lemma 4.6. *If $q_\tau^\mu(y)$ is a quadratic approximation of the function f_τ^μ at x:*

$$q_\tau^\mu(y) := f_\tau^\mu(x) + \nabla f_\tau^\mu(x)^\intercal (y - x) + \frac{1}{2} (y - x)^\intercal \nabla^2 f_\tau^\mu(x) (y - x),$$

then

$$|f_\tau^\mu(y) - q_\tau^\mu(y)| \leq \frac{1}{6} L_{f_\tau^\mu} \|y - x\|^3.$$

Proof. Using corollary 1.5.3 in [95] and Lemma 4.5 we have:

$$\begin{aligned}
|f_\tau^\mu(y) - q_\tau^\mu(y)| &\leq \|y - x\|^2 \int_0^1 \int_0^t \|\nabla^2 f_\tau^\mu(x + s(y - x)) - \nabla^2 f_\tau^\mu(x)\| ds dt \\
&\leq \|y - x\|^2 \int_0^1 \int_0^t s L_{f_\tau^\mu} \|y - x\| ds dt \\
&= \frac{1}{6} L_{f_\tau^\mu} \|y - x\|^3.
\end{aligned}$$

□

4.6 A property of conjugate gradients

The following property of CG is used in the convergence analysis of pdNCG.

Lemma 4.7. *Let $Ax = b$, where A is a symmetric and positive definite matrix. Furthermore, let us assume that this system is solved using CG approximately; CG is terminated prematurely at the i^{th} iteration. Then if CG is initialized with the zero solution the approximate solution x^i satisfies:*

$$(x^i)^\intercal A x^i = (x^i)^\intercal b.$$

The same result holds when Preconditioned Conjugate Gradients (PCG) is used.

Proof. The following property is shown in proof of Lemma 2.4.1 in [73]. If CG algorithm is initialized with the zero solution $p_0 = 0$, then it returns a solution x^i which satisfies:

$$x^i := \arg\min_p \{\frac{1}{2} p^\intercal A p - p^\intercal b \mid p \in \mathcal{E}_i\},$$

where

$$\mathcal{E}_i := \text{span}(b, Ab, \ldots, A^{i-1}b).$$

Therefore for every $p \in \mathcal{E}_i$, at $t = 0$, we get:

$$\frac{d(\frac{1}{2}(x^i + tp)^\intercal A(x^i + tp) - (x^i + tp)^\intercal b)}{dt} = (Ax^i - b)^\intercal p = 0.$$

Since, $x^i \in \mathcal{E}_i$,

$$(Ax^i - b)^\intercal x^i = 0 \Longleftrightarrow (x^i)^\intercal A x^i = (x^i)^\intercal b.$$

This completes the first part. In the case where PCG is employed with symmetric positive definite preconditioner $P = EE^\intercal$, PCG is equivalent to solving approximately the system $E^{-1}AE^{-\intercal}\xi = E^{-1}b$ using CG and then calculating $x^i = E^{-\intercal}\xi^i$. Therefore, by applying the previous we get that $(\xi^i)^\intercal E^{-1}AE^{-\intercal}\xi^i = (\xi^i)^\intercal E^{-1}b$ and by substituting $\xi^i = E^\intercal x^i$ we prove the second part. □

4.7 Framework of pdNCG

In this section we describe primal-dual Newton Conjugate Gradients (pdNCG) for the solution of the primal-dual optimality conditions (4.9). The method is similar to the one in [32] for signal reconstruction problems, although, the two approaches differ in Step 3 of pdNCG. More precisely, the original version of the method instead of the projection operation in Step 3 it uses a backtracking procedure along the dual direction until the box constraint is satisfied. Additionally, we make a step further and give a complete convergence analysis and a worst-case iteration complexity result in Section 4.8. A detailed pseudo-code of the method is given below.

Algorithm 4.1 primal-dual Newton Conjugate Gradients (pdNCG)

1: **Loop:** For $k = 1, 2, ...,$ until $\|d^k\|_{x^k} \leq \epsilon$, where $\epsilon > 0$.

2: Obtain d^k by solving approximately the system

$$H(x^k, g^k)d = -\nabla f_\tau^\mu(x^k) \tag{4.16}$$

using CG or PCG, where

$$H(x, g) = \tau D(I - Ddiag(x)diag(g)) + \nabla^2\varphi(x) \tag{4.17}$$

and matrix D is defined in (4.6). Obtain Δg^k by calculating

$$\Delta g^k = D(I - Ddiag(x)diag(g))d^k - (g^k - Dx^k). \tag{4.18}$$

3: Set $\hat{g}^{k+1} = g^k + \Delta g^k$ and calculate

$$g^{k+1} := P_{\|\cdot\|_\infty \leq 1}(\hat{g}^{k+1}),$$

where $P_{\|\cdot\|_\infty \leq 1}(\cdot)$ is the orthogonal projection in the ℓ_∞ ball.

4: Find the least integer $j \geq 0$ such that the function $f_\tau^\mu(x)$ is sufficiently decreased along d^k:

$$f_\tau^\mu(x^k + c_3^j d^k) \leq f_\tau^\mu(x^k) - c_2 c_3^j \|d^k\|_{x^k}^2,$$

where $0 < c_2 < 1/2$, $0 < c_3 < 1$, and set $\alpha = c_3^j$.

5: Set $x^{k+1} = x^k + \alpha d^k$.

In Algorithm pdNCG we make use of the local norm:

$$\|\cdot\|_{x^k} := \sqrt{\langle \cdot, H(x^k, g^k)\cdot \rangle}, \tag{4.19}$$

where H is a positive definite matrix. Step 2 of pdNCG is the approximate solution of the linearization of the first two equations in (4.9). The matrix H is obtained by simply eliminating the variables Δg^k in the linearized system. Step 2 is performed by CG or PCG which is always initialized with the zero solution

and it is terminated when

$$\|r_\tau^\mu(x,g)\| \leq \eta \|\nabla f_\tau^\mu(x)\|, \tag{4.20}$$

where $r_\tau^\mu(x,g) = H(x,g)d + \nabla f_\tau^\mu(x)$ is the residual and $0 \leq \eta < 1$ is a user-defined constant. In practice we have observed that setting η^k =1.0e–1 results in very fast convergence for our problems of interest, however, the method will be analyzed for η^k set as in

$$\eta^k = \min\{\frac{1}{2}, \|\nabla f_\tau^\mu(x^k)\|^{c_0}\}, \tag{4.21}$$

with $c_0 = 1$.

Step 3 is a projection of $\hat{g}^{k+1}$ to the set $\|g\|_\infty \leq 1$ such that matrix H is positive definite. The projection operator is:

$$v := P_{\|\cdot\|_\infty \leq 1}(u) = \text{sign}(u)\min(|u|, 1)$$

and it is applied component-wise. Step 4 is a backtracking line-search technique in order to guarantee that the sequence $\{x^k\}$ generated by pdNCG monotonically decreases the objective function $f_\tau^\mu(x)$.

Observe that pdNCG adheres to the structure of Algorithm 1.1 GFrame, which was introduced in Section 1.2 as a general framework of methods for convex optimization. In particular, for pdNCG the following local model is build at every iteration:

$$Q(x; x^k) := f_\tau^\mu(x^k) + (\nabla f_\tau^\mu(x^k))^\intercal (x - x^k) + \frac{1}{2}(x - x^k)^\intercal H(x^k, g^k)(x - x^k),$$

which corresponds to Step 2 of GFrame. Step 2 of pdNCG is an approximate minimization of the local model:

$$x^{k+1} \approx \arg\min_x \; Q(x; x^k),$$

which is followed by setting $d^k := x^{k+1} - x^k$. Hence, Step 2 of pdNCG corresponds to Step 3 of GFrame. In Step 3 of pdNCG the dual variables are determined in order to construct matrix H in the local model. In Step 4 of pdNCG a step-size α is calculated, which corresponds to Step 4 of GFrame. Finally, Step 5 of pdNCG corresponds to Step 5 of GFrame by setting $y^{k+1} := x^k + \alpha d^k$.

4.8 Convergence analysis and worst-case iteration complexity

In this section we present the analysis of the pdNCG method given in [53]. In particular, we prove global convergence, we study the global and local convergence rates and we explicitly define a region in which pdNCG has fast convergence rate. Additionally, a worst-case iteration complexity result of pdNCG is presented. The reader will notice that the results in this section are established when CG is used in step 2 of pdNCG Algorithm 4.1. However, based on Lemma 4.7 it is trivial to

show that the same results hold if PCG is used.

Before we introduce notational conventions for this section, it is necessary to find uniform bounds for matrix $H(x, g)$ in (4.17). This is shown in the following lemma.

Lemma 4.8. *If $\|g\|_\infty \leq 1$, then matrix H is uniformly bounded by*

$$\lambda_n I_n \prec H(x, g) \preceq \Big(\frac{\tau}{\mu} + \lambda_1\Big) I_n \ \forall x.$$

Proof. This result easily follows by using the definition of H in (4.17) and (9.3). A similar argument, but for signal reconstruction problems, is also claimed in [32], page 1970. □

The equivalence of the Euclidean and the local norm (4.19) if $\|g\|_\infty \leq 1$, is given by the following inequality:

$$\lambda_n^{\frac{1}{2}} \|d\| \leq \|d\|_x \leq \Big(\frac{\tau}{\mu} + \lambda_1\Big)^{\frac{1}{2}} \|d\| \ \forall x, d. \tag{4.22}$$

The upper bound of the largest eigenvalue of H if $\|g\|_\infty \leq 1$, will be denoted by $\tilde{\lambda}_1 = (\tau/\mu + \lambda_1)$. An upper bound of the condition number of matrix $H(x, g)$ will be denoted by $\kappa = \tilde{\lambda}_1/\lambda_n$. The Lipschitz constant $L_{f_\tau^\mu}$ defined in Lemma 4.5, will be denoted by L. Finally, the indices τ and μ from function f_τ^μ are dropped.

4.8.1 Global convergence

First, the minimum decrease of the objective function at every iteration of pdNCG is calculated.

Lemma 4.9. *Let $x \in \mathbb{R}^n$ be the current iteration of pdNCG, $d \in \mathbb{R}^n$ be the pdNCG direction for the primal variables, which is calculated using CG. The parameter η of the termination criterion (4.20) of CG is set to $0 \leq \eta < 1$. If x is not the minimizer of problem (4.4), i.e., $\nabla f(x) \neq 0$, then the backtracking line-search algorithm in step 4 of pdNCG will calculate a step-size $\hat{\alpha}$ such that:*

$$\hat{\alpha} \geq c_3 \frac{\lambda_n}{\tilde{\lambda}_1}.$$

For this step-size $\hat{\alpha}$ the following holds:

$$f(x) - f(x(\hat{\alpha})) > c_4 \|d\|_x^2,$$

where $c_4 = c_2 c_3 \frac{1}{\kappa}$ and $x(\hat{\alpha}) = x + \hat{\alpha} d$.

Proof. For $x(\alpha) = x + \alpha d$ and from smoothness of f we have:

$$f(x(\alpha)) \leq f(x) + \alpha \nabla f(x)^\intercal d + \frac{\alpha^2}{2} \tilde{\lambda}_1 \|d\|^2.$$

From Lemma 4.8 we have that $H(x, g)$ is positive definite if $\|g\|_\infty \leq 1$, which is the condition always satisfied by step 3 of pdNCG. Then, if $\nabla f(x) \neq 0$ the

CG algorithm terminated at the i^{th} iteration returns the vector $d^i \neq 0$ which according to Lemma 4.7 satisfies:

$$(d^i)^\intercal H(x,g)d^i = -(d^i)^\intercal \nabla f(x).$$

Therefore, by setting $d := d^i$ we get:

$$f(x(\alpha)) \leq f(x) - \alpha\|d\|_x^2 + \frac{\alpha^2}{2}\tilde{\lambda}_1\|d\|^2.$$

Using (4.22) we get:

$$f(x(\alpha)) \leq f(x) - \alpha\|d\|_x^2 + \frac{\alpha^2}{2}\frac{\tilde{\lambda}_1}{\lambda_n}\|d\|_x^2.$$

The right hand side of the above inequality is minimized for $\alpha^* = \frac{\lambda_n}{\tilde{\lambda}_1}$, which gives:

$$f(x(\alpha^*)) \leq f(x) - \frac{1}{2}\frac{\lambda_n}{\tilde{\lambda}_1}\|d\|_x^2.$$

Observe that for this step-size the exit condition of the backtracking line-search algorithm is satisfied, since

$$f(x(\alpha^*)) \leq f(x) - \frac{1}{2}\frac{\lambda_n}{\tilde{\lambda}_1}\|d\|_x^2 < f(x) - c_2\frac{\lambda_n}{\tilde{\lambda}_1}\|d\|_x^2.$$

Therefore the step-size $\hat{\alpha}$ returned by the backtracking line-search algorithm is in the worst-case bounded by

$$\hat{\alpha} \geq c_3\frac{\lambda_n}{\tilde{\lambda}_1},$$

which also satisfies the exit condition of the backtracking line-search algorithm. Using $c_3\lambda_n/\tilde{\lambda}_1$ as a step-size results in the following decrease of the objective function:

$$f(x) - f(x(\hat{\alpha})) > c_2c_3\frac{\lambda_n}{\tilde{\lambda}_1}\|d\|_x^2 = c_2c_3\frac{1}{\kappa}\|d\|_x^2.$$

□

Global convergence of pdNCG for the primal variables is established in the following theorem.

Theorem 4.1 (Theorem 1 in [53])**.** *Let $\{x^k\}$ be a sequence generated by pdNCG. The parameter η of the termination criterion* (4.20) *of the CG algorithm is set to $0 \leq \eta < 1$. Then the sequence $\{x^k\}$ converges to $\tilde{x}$, which is the minimizer of f in problem (4.4).*

Proof. From Lemma 4.8 and step 3 of pdNCG we have that matrix $H(x,g)$ is symmetric and positive definite at any x^k, g^k. Moreover, if $0 \leq \eta < 1$ in (4.20), then CG returns $d^k = 0$ at a point x^k if and only if $\nabla f(x^k) = 0$. Hence, only at optimality CG will return a zero direction. Moreover, from Lemma 4.9 we get that if $\nabla f(x^k) \neq 0$, then $\hat{\alpha}^k$ is bounded away from zero and the function

$f(x)$ is monotonically decreasing when the step $\hat{\alpha}^k d^k$ is applied. The monotonic decrease of the objective function implies that $\{f(x^k)\}$ converges to a limit, thus, $\{f(x^k) - f(x^{k+1})\} \to 0$. Since $f(x^0) < \infty$ and f is monotonically decreased, where x^0 is a finite first guess given as an input to pdNCG, then the sequence $\{x^k\}$ belongs to a closed, bounded and therefore, compact sublevel set. Hence, the sequence $\{x^k\}$ must have a sub-sequence which converges to a point $\tilde{x}$ and this implies that $\{x^k\}$ also converges to $\tilde{x}$. Using Lemma 4.9 and $\{f(x^k) - f(x^{k+1})\} \to 0$ we get that $\|d^k\|_x \to 0$, hence, due to positive definiteness of H, $\|d^k\| \to 0$, which implies that $\|\nabla f(x^k)\| \to 0$. Therefore, $\tilde{x}$ is a stationary point of function f. Strong convexity of f guarantees that a stationary point must be a minimizer. □

Convergence of the dual variables is shown in the following theorem.

Theorem 4.2 (Theorem 2 in [53])**.** *Let the assumptions of Theorem 4.1 hold. Then we have that the sequences of dual variables produced by pdNCG satisfy $\{g^k\} \to D\tilde{x}$, where $\tilde{x}$ is the optimal solution of problem* (4.4)*. Furthermore, the previous implies that the primal-dual iterates of pdNCG converge to the solution of system* (4.9)*.*

Proof. From Theorem 4.1 we have that $d^k \to 0$ and $x^k \to \tilde{x}$. Hence, from (4.18) we get that $\Delta g^k \to -g^k + D\tilde{x}$. Moreover, we have that the iterates at step 3 of pdNCG satisfy $\tilde{g}^k \to D\tilde{x}$ and consequently:

$$g^k = P_{\|\cdot\|_\infty \leq 1}(\hat{g}) \to P_{\|\cdot\|_\infty \leq 1}(D\tilde{x}) = D\tilde{x}.$$

It is easy to check that these values of $\tilde{g}$ with the optimal variable $\tilde{x}$ satisfy the system (4.9). The proof is complete. □

4.8.2 Region of fast convergence rate

In this subsection we define a region based on $\|d\|_x$, in which by setting parameter η as in (4.21) with $c_0 = 1$, pdNCG converges with fast rate. The lemma below shows the behaviour of the function f when a step along the primal direction in pdNCG is made.

Lemma 4.10. *Let $x \in \mathbb{R}^n$ be the current iteration of pdNCG, $d \in \mathbb{R}^n$ be the pdNCG direction for primal variables calculated by CG, which is terminated according to criterion (4.20) with $0 \leq \eta < 1$. Then*

$$f(x) - f(x(\alpha)) \geq \alpha\|d\|_x^2 - \frac{\alpha^2}{2}\|d\|_x^2 - \frac{\alpha^3}{6}\frac{L}{\lambda_n^{\frac{3}{2}}}\|d\|_x^3,$$

where $x(\alpha) = x + \alpha d$ and $\alpha > 0$.

Proof. Using Lemma 4.6 and setting $y = x(\alpha) = x + \alpha d$ we get:

$$f(x(\alpha)) \leq f(x) + \alpha\nabla f(x)^\intercal d + \frac{\alpha^2}{2} d^\intercal \nabla^2 f(x) d + \frac{\alpha^3}{6} L\|d\|^3.$$

From Lemma 4.8 and step 3 of pdNCG we have that (4.22) holds. Hence, using (4.22) and Lemma 4.7 we get:

$$f(x(\alpha)) \leq f(x) - \alpha\|d\|_x^2 + \frac{\alpha^2}{2}\|d\|_x^2 + \frac{\alpha^3}{6}\frac{L}{\lambda_n^{\frac{3}{2}}}\|d\|_x^3.$$

The result is obtained by the rearrangement of terms. □

The next lemma determines bounds on the norm of the primal direction d^k as a function of $\|\nabla f_\tau^\mu(x^k)\|$.

Lemma 4.11. *Let $d \in \mathbb{R}^n$ be the pdNCG primal direction at a point $x \in \mathbb{R}^n$ that is calculated by CG, which is terminated according to criterion (4.20) with $0 \leq \eta < 1$. Then the following holds:*

$$\frac{1-\eta^2}{2\tilde{\lambda}_1^{\frac{1}{2}}}\|\nabla f(x)\| \leq \|d\|_x \leq \frac{1}{\lambda_n^{\frac{1}{2}}}\|\nabla f(x)\|.$$

Proof. By squaring (4.20) and making simple rearrangements of it we get:

$$d^\intercal H(x,g)^2 d + 2\nabla f(x)^\intercal H(x,g)d + (1-\eta^2)\|\nabla f(x)\|^2 \leq 0. \tag{4.23}$$

From step 3 of pdNCG we have that the condition of Lemma 4.8 is satisfied. Therefore by using Lemma 4.8 and Cauchy-Schwarz inequality in (4.23) we get:

$$\lambda_n^2\|d\|^2 - 2\tilde{\lambda}_1^{\frac{1}{2}}\|\nabla f(x)\|\|d\|_x + (1-\eta^2)\|\nabla f(x)\|^2 \leq 0.$$

By dropping the quadratic term $\lambda_n^2\|d\|^2$ from the previous inequality and dividing by $\|\nabla f(x)\|$, after making appropriate rearrangements we get:

$$\|d\|_x \geq \frac{1-\eta^2}{2\tilde{\lambda}_1^{\frac{1}{2}}}\|\nabla f(x)\|.$$

This proves the left hand side of the result. For the right hand side, we simply use Lemma 4.7 and (4.22):

$$d^\intercal H(x,g)d = \|d\|_x^2 = -d^\intercal\nabla f(x) \leq \|d\|\|\nabla f(x)\| \leq \frac{1}{\lambda_n^{\frac{1}{2}}}\|d\|_x\|\nabla f(x)\|.$$

By dividing with $\|d\|_x$ we obtain the right hand side of our claim. □

The following lemma will be used to prove fast local convergence rate of pdNCG for the primal variables.

Lemma 4.12. *Let the iterates x^k and g^k be produced by pdNCG, then the following holds:*

$$\|\nabla^2 f(x^k) - H(x^k, g^k)\| \leq \gamma\|d^k\|_{x^k},$$

where

$$\gamma = \Big(\frac{8\tilde{\lambda}_1^{\frac{1}{2}}}{\lambda_n}(L + M + \frac{M}{\mu}) + \frac{M}{\lambda_n^{\frac{1}{2}}\mu}\Big)$$

and M is a finite upper bound on the Euclidean norm of the derivative of $H(x, g)$.

Proof. Let $\tilde{x}$ be the optimal solution of problem (4.4). We rewrite

$$\nabla^2 f(x^k) - H(x^k, g^k) = \nabla^2 f(x^k) - \nabla^2 f(\tilde{x}) + \nabla^2 f(\tilde{x}) - H(x^k, g^k).$$

Moreover, let $\tilde{g}$ be the optimal dual variable, which according to Theorem 4.2 satisfies $\tilde{g} = D(\tilde{x})\tilde{x}$. Notice that matrix D in (4.6) is dependent on variable x; for the purposes of this proof we will explicitly denote this dependence. From the definition of H in (4.17) we have that $H(\tilde{x}, \tilde{g}) = \nabla^2 f(\tilde{x})$. The following holds:

$$\|\nabla^2 f(x^k) - H(x^k, g^k)\| \leq \|\nabla^2 f(x^k) - \nabla^2 f(\tilde{x})\| + \|H(\tilde{x}, \tilde{g}) - H(x^k, g^k)\|.$$

By Lipschitz continuity of $\nabla^2 f$ in Lemma 4.5 we get that:

$$\|\nabla^2 f(x^k) - H(x^k, g^k)\| \leq L\|\tilde{x} - x^k\| + \|H(\tilde{x}, \tilde{g}) - H(x^k, g^k)\|. \tag{4.24}$$

We now focus on bounding $\|H(\tilde{x}, \tilde{g}) - H(x^k, g^k)\|$. Using the fundamental theorem of calculus we have:

$$H(\tilde{x}, \tilde{g}) - H(x^k, g^k) = \int_0^1 \frac{dH(\tilde{x}(s), \tilde{g}(s))}{d(\tilde{x}(s), \tilde{g}(s))}[\tilde{x} - x^k; \tilde{g} - g^k]ds,$$

where $\tilde{x}(s) = \tilde{x} + s(\tilde{x} - x^k)$ and $\tilde{g}(s) = \tilde{g} + s(\tilde{g} - g^k)$. Hence,

$$\|H(\tilde{x}, \tilde{g}) - H(x^k, g^k)\| \leq (\|\tilde{x} - x^k\| + \|\tilde{g} - g^k\|) \int_0^1 \Big\|\frac{dH(\tilde{x}(s), \tilde{g}(s))}{d(\tilde{x}(s), \tilde{g}(s))}\Big\| ds.$$

We now prove that $dH(\tilde{x}(s), \tilde{g}(s))/d(\tilde{x}(s), \tilde{g}(s))$ is bounded in the set $\mathbb{R}^n \times \{g \in \mathbb{R}^n \mid \|g\|_\infty \leq 1\} \subset \mathbb{R}^{2m}$. Observe that the partial derivatives of $H(x, g)$ with respect to x or g are continuous. Therefore, $dH(\tilde{x}(s), \tilde{g}(s))/d(\tilde{x}(s), \tilde{g}(s))$ is a continuous tensor. In this case, the only candidates of unboundedness are the limits $x \to \pm\infty$. By calculating the partial derivatives it is easy to show that at the limits all partial derivatives are finite and this implies that every component of the tensor is bounded in $\mathbb{R}^n \times \{g \in \mathbb{R}^n \mid \|g\|_\infty \leq 1\}$. We will denote the bound by a positive constant M, hence,

$$\|H(\tilde{x}, \tilde{g}) - H(x^k, g^k)\| \leq M(\|\tilde{x} - x^k\| + \|\tilde{g} - g^k\|). \tag{4.25}$$

It remains to find a bound for $\|\tilde{g} - g^k\|$. From step 3 of pdNCG we have:

$$\begin{aligned}\|\tilde{g} - g^k\| &\leq \|P_{\|\cdot\|_\infty \leq 1}(\tilde{g}) - P_{\|\cdot\|_\infty \leq 1}(\hat{g}^k)\| \leq \|\tilde{g} - \hat{g}^k\| \\ &\leq \|D(\tilde{x})\tilde{x} - D(x^k)x^k\| \\ &+ \|D(x^k)(I - D(x^k)diag(x^k)diag(g^k))\|\|d^k\| \\ &= \|\nabla\psi_\mu(\tilde{x}) + \nabla\psi_\mu(x^k)\| \\ &+ \|D(x^k)(I - D(x^k)diag(x^k)diag(g^k))\|\|d^k\|.\end{aligned}$$

Using Lemma 4.3 and

$$D(x^k)(I - D(x^k)diag(x^k)diag(g^k)) \preceq D(x^k) \preceq \frac{1}{\mu}I_n,$$

which holds for $\|g\|_\infty \leq 1$, we have that:

$$\|\tilde{g} - g^k\| \leq \frac{1}{\mu}\|\tilde{x} - x^k\| + \frac{1}{\mu}\|d^k\|. \tag{4.26}$$

By combining inequalities (4.25) and (4.26) in (4.24) we get:

$$\|\nabla^2 f(x^k) - H(x^k, g^k)\| \leq (L + M + \frac{M}{\mu})\|\tilde{x} - x^k\| + \frac{M}{\mu}\|d^k\|.$$

Combining Lemmas 4.4, 4.11 and (4.21) for a bound on $\|\tilde{x} - x^k\|$ we get:

$$\|\nabla^2 f(x^k) - H(x^k, g^k)\| \leq \frac{8\tilde{\lambda}_1^{\frac{1}{2}}}{\lambda_n}(L + M + \frac{M}{\mu})\|d^k\|_{x^k} + \frac{M}{\mu}\|d^k\|.$$

Using (4.22) we get the result. □

Based on Lemmas 4.10, 4.11 and 4.12, a region is defined in the following lemma in which unit-step sizes are calculated by the backtracking line-search algorithm. Additionally, for this region, $\|d^{k+1}\|_{x^{k+1}}$ is bounded as a function of $\|d^k\|_{x^k}$. In this lemma the constants c_2 and c_3 have been defined in step 4 of pdNCG, moreover, $x^{k+1} = x^k + d^k$.

Lemma 4.13. *If $\|d^k\|_{x^k} \leq 3(1 - 2c_2)\frac{\lambda_n^{\frac{3}{2}}}{L}$, then the backtracking line-search algorithm in step 4 of pdNCG calculates unit step-sizes. Moreover, if the parameter η^k of the termination criterion (4.20) of CG is set as in (4.21) with $c_0 = 1$, then for two consequent primal directions d^k, d^{k+1} and points x^k, x^{k+1}, the following holds:*

$$\frac{1}{2}\frac{16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L}{\lambda_n^{\frac{3}{2}}}\|d^{k+1}\|_{x^{k+1}} \leq \Big(\frac{1}{2}\frac{16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L}{\lambda_n^{\frac{3}{2}}}\|d^k\|_{x^k}\Big)^2.$$

Proof. By setting $\alpha = 1$ in Lemma 4.10 we get:

$$f(x^k) - f(x^{k+1}) \geq \frac{1}{2}\|d^k\|_{x^k}^2 - \frac{1}{6}\frac{L}{\lambda_n^{\frac{3}{2}}}\|d^k\|_{x^k}^3 = \Big(\frac{1}{2} - \frac{1}{6}\frac{L}{\lambda_n^{\frac{3}{2}}}\|d^k\|_{x^k}\Big)\|d^k\|_{x^k}^2.$$

If $\|d^k\|_{x^k} \leq 3(1-2c_2)\frac{\lambda_n^{\frac{3}{2}}}{L}$ we get:

$$f(x^k) - f(x^{k+1}) \geq c_2 \|d^k\|_{x^k}^2,$$

which implies that $\alpha = 1$ satisfies the exit condition of the backtracking line-search algorithm. Let us define the quantities $\nabla f(x(t))^\intercal h$, where $h \in \mathbb{R}^n$, $x(t) = x^k + td^k$ and $x(\delta) = x^k + \delta d^k$, then we have:

$$\begin{aligned}
\nabla f(x(t))^\intercal h &= \nabla f(x^k)^\intercal h + t(d^k)^\intercal \nabla^2 f(x^k) h \\
&\quad + \int_0^t \int_0^u \nabla^3 f(x(\delta))[d^k, d^k, h] d\delta du \\
&\leq \nabla f(x^k)^\intercal h + t(d^k)^\intercal \nabla^2 f(x^k) h \\
&\quad + \int_0^t \int_0^u \Big| \nabla^3 f(x(\delta))[d^k, d^k, h] \Big| d\delta du \\
&= \nabla f(x^k)^\intercal h + t(d^k)^\intercal \nabla^2 f(x^k) h \\
&\quad + \int_0^t \int_0^u \lim_{\delta \to 0} \Big| \frac{(d^k)^\intercal (\nabla^2 f(x(\delta)) - \nabla^2 f(x^k)) h}{\delta} \Big| d\delta du \\
&\leq \nabla f(x^k)^\intercal h + t(d^k)^\intercal \nabla^2 f(x^k) h \\
&\quad + \|d^k\| \|h\| \int_0^t \int_0^u \lim_{\delta \to 0} \Big\| \frac{1}{\delta} (\nabla^2 f(x(\delta)) - \nabla^2 f(x^k)) \Big\| d\delta du \\
&\leq \nabla f(x^k)^\intercal h + t(d^k)^\intercal \nabla^2 f(x^k) h + \|d^k\| \|h\| \int_0^t \int_0^u L \|d^k\| d\delta du \\
&= \nabla f(x^k)^\intercal h + t(d^k)^\intercal \nabla^2 f(x^k) h + \frac{t^2}{2} L \|d^k\|^2 \|h\|.
\end{aligned}$$

By taking absolute values and setting $t = 1$ we get:

$$\begin{aligned}
|\nabla f(x^{k+1})^\intercal h| &\leq |\nabla f(x^k)^\intercal h + (d^k)^\intercal \nabla^2 f(x^k) h| + \frac{1}{2} L \|d^k\|^2 \|h\| \\
&\leq \|\nabla f(x^k) + \nabla^2 f(x^k) d^k\| \|h\| + \frac{1}{2} L \|d^k\|^2 \|h\| \\
&\leq \|\nabla f(x^k) + H(x^k, g^k) d^k\| \|h\| \\
&\quad + \|\nabla^2 f(x^k) - H(x^k, g^k)\| \|d^k\| \|h\| + \frac{1}{2} L \|d^k\|^2 \|h\|. \qquad (4.27)
\end{aligned}$$

Observe that from (4.21) with $c^0 = 1$ we have that $\eta^k \leq \|\nabla f(x^k)\|$. Hence, combining the previous with Lemma 4.11 and (4.20) in (4.27) we have that:

$$\begin{aligned}
|\nabla f(x^{k+1})^\intercal h| &\leq 8\tilde{\lambda}_1 \|d^k\|_{x^k}^2 \|h\| + \|\nabla^2 f(x^k) - H(x^k, g^k)\| \|d^k\| \|h\| \\
&\quad + \frac{1}{2} L \|d^k\|^2 \|h\|.
\end{aligned}$$

Using Lemma 4.12 we have:

$$|\nabla f(x^{k+1})^\intercal h| \leq 8\tilde{\lambda}_1 \|d^k\|_{x^k}^2 \|h\| + \gamma \|d^k\|_{x^k} \|d^k\| \|h\| + \frac{1}{2} L \|d^k\|^2 \|h\|.$$

From the equivalence of norms (4.22) we get:

$$|\nabla f(x^{k+1})^\intercal h| \leq \frac{1}{2}\frac{16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L}{\lambda_n^{\frac{3}{2}}}\|d^k\|_{x^k}^2\|h\|_{x^{k+1}}.$$

The previous result holds for every $h \in \mathbb{R}^n$, hence, by setting $h = d^{k+1}$ and by using Lemma 4.7 we prove the second part of this lemma. □

The following corollary determines the region of fast convergence rate of Newton-CG. By fast rate it is meant that if pdNCG is initialized in this region, then the worst-case iteration complexity result for convergence to $\tilde{x}$ is of the form $\log_2 \log_2 \frac{\text{constant}}{\text{required accuracy}}$. This statement is proved in Subsection 4.8.3 in Theorem 4.4.

Corollary 4.1. *If the parameter η^k in the termination criterion (4.20) of CG is set as in* (4.21) *with $c_0 = 1$ and $\|d^k\|_{x^k} < \varpi$, $0 < \varpi \leq c_5$, where*

$$c_5 = \min\left\{3(1-2c_2)\frac{\lambda_n^{\frac{3}{2}}}{L}, \frac{\lambda_n^{\frac{3}{2}}}{16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L}\right\},$$

then according to Lemma 4.13 pdNCG convergences with fast rate.

4.8.3 Worst-case iteration complexity

The following theorem shows the worst-case iteration complexity of pdNCG in order to enter the region of fast convergence rate, i.e., $\|d\|_x < \varpi$, where $0 < \varpi \leq c_5$ and c_5 has been defined in Corollary 4.1. In this theorem the constant c_4 has been defined in Lemma 4.9, c_2 and c_3 are constants of the backtracking line-search algorithm in step 4 of pdNCG. Moreover, $\tilde{x}$ denotes the minimizer of problem (4.4).

Theorem 4.3 (Theorem 3 in [53])**.** *Starting from an initial point x^0, such that $\|d^0\|_{x^0} \geq \varpi$ and setting $0 \leq \eta < 1$ in the termination criterion* (4.20) *of CG, then pdNCG requires at most*

$$K_1 = c_6 \log\Big(\frac{f(x^0) - f(\tilde{x})}{c_7\varpi^2}\Big)$$

iterations to obtain a solution x^k, $k > 0$, such that $\|d^k\|_{x^k} < \varpi$, where

$$c_6 = \frac{2\kappa^2}{(1-\eta^2)^2 c_2 c_3} \quad \textit{and} \quad c_7 = \frac{1}{2\kappa}.$$

Proof. Let us assume an iteration index $k > 0$, then from Lemmas 4.4 and 4.11 we get:

$$f(x^k) - f(\tilde{x}) \geq \frac{1}{2\kappa}\|d^k\|_{x^k}^2 \tag{4.28}$$

and

$$f(x^{k-1}) - f(\tilde{x}) \leq \frac{2\kappa}{(1-\eta^2)^2}\|d^{k-1}\|_{x^{k-1}}^2. \tag{4.29}$$

From Lemma 4.9 we have:

$$f(x^k) < f(x^{k-1}) - c_4 \|d^{k-1}\|^2_{x^{k-1}}. \tag{4.30}$$

Combining (4.29), (5.2) and subtracting $f(\tilde{x})$ from both sides we get:

$$\begin{aligned} f(x^k) - f(\tilde{x}) &< \Big(1 - \frac{(1-\eta^2)^2 c_4}{2\kappa}\Big)(f(x^{k-1}) - f(\tilde{x})) \\ &< \Big(1 - \frac{(1-\eta^2)^2 c_4}{2\kappa}\Big)^k (f(x^0) - f(\tilde{x})) \\ &= \Big(1 - \frac{(1-\eta^2)^2 c_2 c_3}{2\kappa^2}\Big)^k (f(x^0) - f(\tilde{x})) \end{aligned}$$

From the last inequality and (4.28) we get:

$$\frac{1}{2\kappa}\|d^k\|^2_{x^k} < \Big(1 - \frac{(1-\eta^2)^2 c_2 c_3}{2\kappa^2}\Big)^k (f(x^0) - f(\tilde{x})).$$

Using the definitions of constants c_6 and c_7 we have:

$$\|d^k\|^2_{x^k} < \Big(1 - \frac{1}{c_6}\Big)^k \frac{1}{c_7}(f(x^0) - f(\tilde{x})).$$

Hence, we conclude that after at most K_1 iterations as defined in the preamble of this theorem, the algorithm produces $\|d^k\|_{x^k} < \varpi$. □

It is worth pointing out that a worst-case iteration complexity result for the global phase (before fast local convergence) of standard Newton method can be obtained by slightly modifying the proof of Theorem 4.3. In particular, in the proof of Theorem 4.3, one simply has to replace matrix H with $\nabla^2 f$ and set $\eta = 0$ (exact Newton directions), the constant κ remains unchanged since the matrices H and $\nabla^2 f$ have the same uniform bounds, see (4.15) and Lemma 4.8. Using the previous adjustments, it is easy to show that the dominant term κ^2 in the result of Theorem 4.3 is preserved for standard Newton method. To the best of our knowledge, the result of $\mathcal{O}(\kappa^2)$ is the tightest that has been obtained for the standard Newton method, see Subsection 9.5 in [18].

The following theorem presents the worst-case iteration complexity result of pdNCG to obtain a solution x^l, of accuracy $f(x^l) - f(\tilde{x}) < \epsilon$, when initialized at a point inside the region of fast convergence.

Theorem 4.4 (Theorem 4 in [53])**.** *Suppose that there is an iteration index k of pdNCG, such that $\|d^k\|_{x^k} < \varpi$. If η in (4.20) is set as in (4.21) with $c_0 = 1$, then pdNCG needs at most*

$$K_2 = \log_2 \log_2 \Big(\frac{c_8}{\epsilon}\Big)$$

additional iterations to obtain a solution x^l, $l > k$, such that $f(x^l) - f(\tilde{x}) < \epsilon$, where

$$c_8 = \frac{16\kappa\lambda_n^3}{(16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L)^2}.$$

Proof. Suppose that there is an iteration index k such that $\|d^k\|_{x^k} < \varpi$, then for an index $l > k$, by applying Lemma 4.13 recursively we get:

$$\frac{1}{2}\frac{16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L}{\lambda_n^{\frac{3}{2}}}\|d^l\|_{x^l} \leq \Big(\frac{1}{2}\frac{16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L}{\lambda_n^{\frac{3}{2}}}\|d^k\|_{x^k}\Big)^{2^{l-k}} < \Big(\frac{1}{2}\Big)^{2^{l-k}}. \tag{4.31}$$

From Lemmas 4.4, 4.11 and η^k in (4.21) we get:

$$f(x^l) - f(\tilde{x}) \leq 4\kappa\|d^l\|_{x^l}^2.$$

By replacing (4.31) in the above inequality we get:

$$f(x^l) - f(\tilde{x}) < \frac{16\kappa\lambda_n^3}{(16\tilde{\lambda}_1\lambda_n + 2\gamma\lambda_n^{\frac{1}{2}} + L)^2}\Big(\frac{1}{2}\Big)^{2^{l-k+1}}.$$

Hence, in order to obtain a solution x^l, such that $f(x^l) - f(\tilde{x}) < \epsilon$, pdNCG requires at most as many iterations as in the preamble of this theorem. □

The following theorem summarizes the complexity result of pdNCG. The constants c_6, c_7 and c_8 in this theorem are defined in Theorems 4.3 and 4.4, respectively.

Theorem 4.5. *Starting from an initial point x^0, such that $\|d^0\|_{x^0} \geq \varpi$, pdNCG requires at most*

$$K_3 = c_6 \log\Big(\frac{f(x^0) - f(\tilde{x})}{c_7\varpi^2}\Big) + \log_2\log_2\Big(\frac{c_8}{\epsilon}\Big)$$

iterations to converge to a solution x^k, $k > 0$, of accuracy

$$f(x^k) - f(\tilde{x}) < \epsilon.$$

Chapter 5

pdNCG for Compressed Sensing

In this chapter we discuss a specialized version of pdNCG for CS problems, which was first presented in [41]. The method is aimed for sparse reconstruction problems using the ℓ_1 analysis formulation:

$$\text{minimize } \tau\|W^*x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{5.1}$$

where $W \in E^{n\times l}$, $E = \mathbb{R}$ or $\mathbb{C}$, $n \leq l$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\tau > 0$ and the star superscript is the conjugate transpose operator. Applications have been discussed in Section 2.3. Assumptions and results that were presented in Section 2.3 will be used in this chapter.

Our contribution is a computationally *inexpensive* and *provably* effective preconditioning technique. At every iteration of pdNCG, the most computationally expensive task is the inexact solution of a linear system with an ill-conditioned matrix. We prove that the preconditioner clusters the eigenvalues of the matrix of the linear system around one. Hence, the linear systems can be solved very fast using an iterative method. The preconditioner exploits the sparsity of the iterations close to optimality and the Restricted Isometry Property (RIP) (Assumption 2.1), i.e., near orthogonality of the columns of matrix A in subspaces defined by subsets of columns of matrix W^*.

This work is also supported by a user friendly implementation of the proposed algorithm, see the hyperlink in [41].

5.1 Smoothing

The non-differentiability of the ℓ_1-norm in (5.1) is treated by applying smoothing. In particular, the ℓ_1-norm is replaced with the pseudo-Huber function (3.18). The original problem in (5.1) is approximated by the following problem:

$$\text{minimize } f_\tau^\mu(x) := \tau\psi_\mu(x) + \frac{1}{2}\|Ax - b\|_2^2, \tag{5.2}$$

where

$$\psi_\mu(x) = \sum_{i=1}^{l}((\mu^2 + |W_i^*x|^2)^{\frac{1}{2}} - \mu).$$

5.2 Derivatives

Let the bar be the complex conjugate operator. Moreover, let us define the function $Re(\cdot)$ which takes a complex input and returns its real part. For simplification of notation, occasionally we will use $Re(\cdot)$ without the parenthesis.

The gradient of pseudo-Huber function in (3.18) is:

$$\nabla\psi_\mu(x) = Re(WDW^*)x, \tag{5.3}$$

where $D := diag(D_1, D_2, \cdots, D_l)$ with

$$D_i := (\mu^2 + |y_i|^2)^{-\frac{1}{2}} \quad \forall i = 1, 2, \cdots, l, \tag{5.4}$$

and $y = [y_1, y_2, \cdots, y_l]^\intercal := W^*x$. The Hessian matrix of ψ_μ is:

$$\nabla^2\psi_\mu(x) := \frac{1}{4}(W\hat{Y}W^* + \bar{W}\hat{Y}\bar{W}^* + W\tilde{Y}\bar{W}^* + \bar{W}\tilde{\hat{Y}}W^*), \tag{5.5}$$

where $\hat{Y} := diag\left[\hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_l\right]$, $\tilde{Y} := diag\left[\tilde{Y}_1, \tilde{Y}_2, ..., \tilde{Y}_l\right]$ and

$$\hat{Y}_i := \mu^2 D_i^3 + D_i, \quad \tilde{Y}_i := -y_i^2 D_i^3, \quad i = 1, 2, ..., l. \tag{5.6}$$

The gradient of function f_τ^μ in (5.2) is:

$$\nabla f_\tau^\mu(x) = \tau\nabla\psi_\mu(x) + A^\intercal(Ax - b).$$

Moreover, the Hessian matrix of f_τ^μ is:

$$\nabla^2 f_\tau^\mu(x) = \tau\nabla^2\psi_\mu(x) + A^\intercal A. \tag{5.7}$$

5.3 Primal-dual formulation and optimality conditions

The primal-dual formulation of problem (5.2) can be obtained from (3.21) by setting $\varphi(x) = 1/2\|Ax - b\|_2^2$:

$$\text{minimize} \sup_{g\in\mathbb{C}^l, \|g\|_\infty\leq 1} \tau Re(\bar{g}^*W^*)x + \tau\sum_{i=1}^{l}\left(\mu(1 - |g_i|^2)^{1/2} - \mu\right) + \frac{1}{2}\|Ax - b\|_2^2.$$

Its optimality conditions are:

$$\begin{aligned} \tau Re(W\bar{g}) + A^\intercal(Ax - b) &= 0, \\ D^{-1}\bar{g} &= W^*x. \end{aligned} \tag{5.8}$$

In (5.8), D is defined in (5.4) and $g \in E^l$. In this chapter we discuss the details of pdNCG, which solves the primal-dual optimality conditions (5.8). The reason that we choose to solve the primal-dual optimality conditions has been discussed

in details in Section 4.4.

5.4 Useful results

In the following lemma we show that $x_{\tau,\mu} := \arg\min f_\tau^\mu(x)$ for τ constant is a continuous and differentiable function of μ.

Lemma 5.1. *Let τ be a constant and consider $x_{\tau,\mu}$ as a functional of μ. If condition $Ker(W^*) \cap Ker(A) = \{0\}$ in (2.18) is satisfied, then $x_{\tau,\mu}$ is continuous and differentiable.*

Proof. The optimality conditions of problem (5.2) are:

$$\tau \nabla \psi_\mu(x) + A^\intercal(Ax - b) = 0.$$

According to definition of $x_{\tau,\mu}$, we have:

$$\begin{aligned}
\tau \nabla \psi_\mu(x_{\tau,\mu}) + A^\intercal(Ax_{\tau,\mu} - b) = 0 \qquad &\Longrightarrow \\
\tau \frac{d\nabla \psi_\mu(x_{\tau,\mu})}{d\mu} + A^\intercal A \frac{dx_{\tau,\mu}}{d\mu} = 0 \qquad &\Longrightarrow \\
\tau\Big(\nabla^2 \psi_\mu(x_{\tau,\mu}) \frac{dx_{\tau,\mu}}{d\mu} + \frac{d\nabla \psi_\mu(x)}{d\mu}\Big|_{x_{\tau,\mu}}\Big) + A^\intercal A \frac{dx_{\tau,\mu}}{d\mu} = 0 \qquad &\Longleftrightarrow \\
\Big(\tau \nabla^2 \psi_\mu(x_{\tau,\mu}) + A^\intercal A\Big) \frac{dx_{\tau,\mu}}{d\mu} + \tau \frac{d\nabla \psi_\mu(x)}{d\mu}\Big|_{x_{\tau,\mu}} = 0 \qquad &\Longleftrightarrow \\
\nabla^2 f_\tau^\mu(x_{\tau,\mu}) \frac{dx_{\tau,\mu}}{d\mu} + \tau \frac{d\nabla \psi_\mu(x)}{d\mu}\Big|_{x_{\tau,\mu}} = 0,
\end{aligned}$$

where $d\nabla\psi_\mu(x)/d\mu|_{x_{\tau,\mu}}$ is the first-order derivative of $\nabla\psi_\mu(x)$ as a functional of μ, measured at $x_{\tau,\mu}$. Notice that due to condition $\mathrm{Ker}(W^*) \cap \mathrm{Ker}(A) = \{0\}$ in (2.18) we have that $\nabla^2 f_\tau^\mu(x)$ is positive definite $\forall x$, hence $x_{\tau,\mu}$ is unique. Therefore, the previous system has a unique solution, which means that $x_{\tau,\mu}$ is uniquely differentiable as a functional of μ with τ being constant. Therefore, $x_{\tau,\mu}$ is continuous as a functional of μ. □

Remark 5.1. *Lemma 5.1 and continuity of $x_{\tau,\mu}$ (as a function of μ) imply that there exists sufficiently small smoothing parameter μ such that $\|x_{\tau,\mu} - x_\tau\|_2 < \omega$ for any arbitrarily small $\omega > 0$.*

5.5 The algorithm

First, we convert the optimality conditions (5.8) to the real case. Let the function $Im(\cdot)$ take a complex input and return its imaginary part. For simplification of notation, occasionally we will use $Im(\cdot)$ without the parenthesis. Then the conversion is done by splitting matrix $W = ReW + \sqrt{-1}ImW$ and the dual variables $g = g_{re} + \sqrt{-1}g_{im}$ into their real and imaginary parts. We do this in order to obtain optimality conditions which are differentiable in the classical sense

of real analysis. This allows a straightforward application of pdNCG method. The optimality conditions with real variables are:

$$\begin{aligned} \tau(ReWg_{re} + ImWg_{im}) + A^\intercal(Ax - b) = 0, \\ D^{-1}g_{re} = ReW^\intercal x, \quad D^{-1}g_{im} = ImW^\intercal x. \end{aligned} \tag{5.9}$$

At every iteration of pdNCG the primal-dual directions are calculated by approximately solving the following linearization of the equality constraints in (5.9):

$$\begin{aligned} B\Delta x &= -\nabla f_\tau^\mu(x) \\ \Delta g_{re} &= D(I - B_1)ReW^\intercal\Delta x + DB_2ImW^\intercal\Delta x - g_{re} + DReW^\intercal x \\ \Delta g_{im} &= D(I - B_4)ImW^\intercal\Delta x + DB_3ReW^\intercal\Delta x - g_{im} + DImW^\intercal x \end{aligned} \tag{5.10}$$

where

$$B := \tau\tilde{B} + A^\intercal A, \tag{5.11}$$

$$\begin{aligned} \tilde{B} := {} & ReWD(I - B_1)ReW^\intercal + ImWD(I - B_4)ImW^\intercal + ReWDB_2ImW^\intercal \\ & + ImWB_3DReW^\intercal, \end{aligned}$$

and $B_i, i = 1, 2, 3, 4$ are diagonal matrices with components:

$$\begin{aligned} [B_1]_{ii} &:= D_i[g_{re}]_i ReW_i^\intercal x, \qquad [B_2]_{ii} := D_i[g_{re}]_i ImW_i^\intercal x, \\ [B_3]_{ii} &:= D_i[g_{im}]_i ReW_i^\intercal x, \qquad [B_4]_{ii} := D_i[g_{im}]_i ImW_i^\intercal x. \end{aligned}$$

Remark 5.2. *Matrix B in* (5.11) *is positive definite (and invertible) if $\|g_{re} + \sqrt{-1}g_{im}\|_\infty \leq 1$ and condition* (2.18) *are satisfied. The former condition will be maintained through all iterations of pdNCG.*

It is straightforward to show the claim in Remark 5.2 for the case of W being a real matrix. For the case of complex W we refer the reader to a similar claim which is made in [32], page 1970. Although matrix B is positive definite under the conditions stated in Remark 5.2, it is not symmetric, except in the case that W is real where all imaginary parts are dropped. Therefore in the case of a complex matrix W, PCG cannot be employed to solve (5.10) approximately. To avoid the problem of non-symmetric matrix B the authors in [32] suggested to ignore the non-symmetric part in matrix B and employ CG to solve (5.10). This idea is based on the following remark.

Remark 5.3. *The symmetric part of B tends to the symmetric second-order derivative of f_τ^μ as pdNCG converges (see Section* 5 *in [32]).*

Hence, system (5.10) is replaced with

$$\begin{aligned} \hat{B}\Delta x &= -\nabla f_\tau^\mu(x) \\ \Delta g_{re} &= D(I - B_1)ReW^\intercal\Delta x + DB_2ImW^\intercal\Delta x - g_{re} + DReW^\intercal x \\ \Delta g_{im} &= D(I - B_4)ImW^\intercal\Delta x + DB_3ReW^\intercal\Delta x - g_{im} + DImW^\intercal x, \end{aligned} \tag{5.12}$$

where

$$\hat{B} := \tau \, \mathrm{sym}(\tilde{B}) + A^\intercal A \tag{5.13}$$

and $\mathrm{sym}(\tilde{B}) := 1/2(\tilde{B} + \tilde{B}^\intercal)$ is the symmetric part of $\tilde{B}$. Moreover, PCG is terminated when

$$\|\hat{B}\Delta x + \nabla f_\tau^\mu(x)\|_2 \leq \eta \|\nabla f_\tau^\mu(x)\|_2, \tag{5.14}$$

is satisfied for $\eta \in [0, 1)$. Then the iterate $g = g_{re} + \Delta g_{re} + \sqrt{-1}(g_{im} + \Delta g_{im})$ is projected orthogonally on the box $\{x : \|x\|_\infty \leq 1\}$. The projection operator for complex arguments is applied component-wise and it is defined as $v := P_{\|\cdot\|_\infty \leq 1}(u) = \min(1/|u|, 1) \odot u$, where $\odot$ denotes the component-wise multiplication. In the last step, line-search is employed for the primal Δx direction in order to guarantee that the objective value f_τ^μ decreases monotonically, see Section 5 of [32]. The pseudo-code of pdNCG is presented in Algorithm 5.1.

Algorithm 5.1 primal-dual Newton Conjugate Gradients (pdNCG) for CS

1: **Input:** $c_1 \in (0, 1)$, $c_2 \in (0, 1/2)$, x^0, g_{re}^0 and g_{im}^0, where $\|g_{re}^0 + \sqrt{-1} g_{im}^0\|_\infty \leq 1$.
2: **Loop:** For $k = 1, 2, ...,$ until termination criteria are met.
3: Calculate Δx^k, Δg_{re}^k and Δg_{im}^k by solving approximately the system (5.12), until (5.14) is satisfied for some $\eta \in [0, 1)$.
4: Set $\hat{g}_{re}^{k+1} := g_{re}^k + \Delta g_{re}^k$, $\hat{g}_{im}^{k+1} := g_{im}^k + \Delta g_{im}^k$ and calculate

$$\hat{g}^{k+1} := P_{\|\cdot\|_\infty \leq 1}(\hat{g}_{re}^{k+1} + \sqrt{-1}\hat{g}_{im}^{k+1}),$$

where $P_{\|\cdot\|_\infty \leq 1}(\cdot)$ is the orthogonal projection on the ℓ_∞ ball.
Then set $g_{re}^{k+1} := Re\hat{g}^{k+1}$ and $g_{im}^{k+1} := Im\hat{g}^{k+1}$.
5: Find the least integer $j \geq 0$ such that:

$$f_\tau^\mu(x^k + c_1^j \Delta x^k) \leq f_\tau^\mu(x^k) + c_2 c_1^j (\nabla f_\tau^\mu(x^k))^\intercal \Delta x^k$$

is satisfied and set $\alpha := c_1^j$.
6: Set $x^{k+1} := x^k + \alpha \Delta x^k$.

5.6 Preconditioning

Practical computational efficiency of pdNCG depends on spectral properties of matrix $\hat{B}$ in (5.13). Those can be improved by a suitable preconditioning. In this section we introduce a new preconditioner for $\hat{B}$ and discuss the limiting behaviour of the spectrum of the preconditioned $\hat{B}$.

First, we give an intuitive analysis on the construction of the proposed preconditioner. In Remark 5.1 it is mentioned that the distance ω of the two solutions $x_\tau := \arg\min f_\tau(x)$ and $x_{\tau,\mu} := \arg\min f_\tau^\mu(x)$ can be arbitrarily small for sufficiently small values of μ. Moreover, according to Assumption 2.1, $W^* x_\tau$ is q sparse. Therefore, Remark 5.1 implies that $W^* x_{\tau,\mu}$ is approximately q sparse with nearly zero components of order $\mathcal{O}(\omega)$. A consequence of the previous statement

is that the components of $W^* x_{\tau,\mu}$ split into the following disjoint sets:

$$\mathcal{B} := \{i \in \{1, 2, \cdots, l\} \mid |W_i^* x_{\tau,\mu}| \gg \mathcal{O}(\omega)\}, \quad |\mathcal{B}| = q = |\text{supp}(W^* x_\tau)|,$$

$$\mathcal{B}^c := \{i \in \{1, 2, \cdots, l\} \mid |W_i^* x_{\tau,\mu}| \approx \mathcal{O}(\omega)\}, \qquad |\mathcal{B}^c| = l - q,$$

where $\text{supp}(u) = \{i \in \{1, 2, \ldots, l\} \mid |u_i| \neq 0\}$ and the superscript c denotes the complement of a set. The behaviour of $W^* x_{\tau,\mu}$ has a crucial influence on the matrix $\nabla^2 \psi_\mu(x_{\tau,\mu})$ in (5.5). Notice that the components of the diagonal matrix D, defined in (5.4) as part of $\nabla^2 \psi_\mu(x_{\tau,\mu})$, split into two disjoint sets. In particular, q components are non-zeros much smaller than $\mathcal{O}(1/\omega)$, while the majority, $l - q$, of its components are of $\mathcal{O}(1/\omega)$,

$$D_i \ll \mathcal{O}(\frac{1}{\omega}) \quad \forall i \in \mathcal{B} \quad \text{and} \quad D_i = \mathcal{O}(\frac{1}{\omega}) \quad \forall i \in \mathcal{B}^c. \tag{5.15}$$

Hence, for points close to $x_{\tau,\mu}$ and small μ, matrix $\nabla^2 f_\tau^\mu$ in (5.7) consists of a dominant matrix $\tau \nabla^2 \psi_\mu$ and of matrix $A^\intercal A$ with moderate largest eigenvalue. The previous argument for $A^\intercal A$ is due to (2.17). Observe that $\lambda_{max}(A^\intercal A) = \lambda_{max}(A A^\intercal)$, hence, if δ in (2.17) is not a very large constant, then $\lambda_{max}(A^\intercal A) \leq 1 + \delta$. According to Remark 5.3, the symmetric matrix $\text{sym}(\tilde{B})$ in (5.7) tends to matrix $\nabla^2 \psi_\mu$ as $x \to x_{\tau,\mu}$. Therefore, matrix $\text{sym}(\tilde{B})$ is the dominant matrix in $\hat{B}$. For this reason, in the proposed preconditioning technique, matrix $A^\intercal A$ in (5.7) is replaced by a scaled identity ρI_n, $\rho > 0$, while the dominant matrix $\text{sym}(\tilde{B})$ is maintained. Based on these observations we propose the following preconditioner:

$$\tilde{N} := c\,\text{sym}(\tilde{B}) + \rho I_n. \tag{5.16}$$

In order to capture the approximate separability of the diagonal components of matrix D for points close to $x_{\tau,\mu}$, when μ is sufficiently small, we will work with approximate guess of $\mathcal{B}$ and $\mathcal{B}^c$. For this reason, we introduce the positive constant ν, such that:

$$\#(D_i < \nu) = \sigma.$$

Here σ might be different from the sparsity of $W^* x_\tau$. Furthermore, according to the above definition we have the sets:

$$\mathcal{B}_\nu := \{i \in \{1, 2, \cdots, l\} \mid D_i < \nu\} \quad \text{and} \quad \mathcal{B}_\nu^c := \{1, 2, \cdots, l\} \backslash \mathcal{B}_\nu, \tag{5.17}$$

with $|\mathcal{B}_\nu| = \sigma$ and $|\mathcal{B}_\nu^c| = l - \sigma$. This notation is being used in the following theorem in which we analyze the behaviour of the spectral properties of preconditioned $\nabla^2 f_\tau^\mu$ with preconditioner $N := \tau \nabla^2 \psi_\mu + \rho I_n$. However, according to Remark 5.3, matrices $\hat{B}$ and $\tilde{N}$ tend to $\nabla^2 f_\tau^\mu$ and N, respectively, as $x \to x_{\tau,\mu}$. Therefore, the following theorem is useful for the analysis of the limiting behaviour of the spectrum of preconditioned $\hat{B}$.

In Theorem 5.1 $\lambda_{min}(\cdot)$ denotes the minimum non-zero eigenvalue of the input matrix and $\text{spec}(\cdot)$ denotes the spectrum of the input matrix.

Theorem 5.1 (Theorem 5.1 in [41]). *Let ν be any positive constant and $\#(D_i <$*

$\nu) = \sigma$ *at a point* x*, where* D *is defined in* (5.4)*. Let*

$$\nabla^2 f_\tau^\mu(x) = \tau\nabla^2\psi_\mu(x) + A^\intercal A \quad \textit{and} \quad N := \tau\nabla^2\psi_\mu(x) + \rho I_n.$$

Additionally, let Assumption 2.1 hold for matrices A *and* W *with* $q = \sigma$*, i.e., they satisfy W-RIP with some constant* $\delta_\sigma < 1/2$ *and let* A *satisfy* (2.17) *for some constant* $\delta \geq 0$*.*
If the eigenvectors of $N^{-\frac{1}{2}}\nabla^2 f_\tau^\mu(x)N^{-\frac{1}{2}}$ *do not belong to* $Ker(W^*_{\mathcal{B}^c_\nu})$ *and* $\rho \in [\delta_\sigma, 1/2]$*, then the eigenvalues of* $N^{-1}\nabla^2 f_\tau^\mu(x)$ *satisfy:*

$$|\lambda - 1| \leq \frac{1}{2}\frac{\chi + 1 + (5\chi^2 - 2\chi + 1)^{\frac{1}{2}}}{\tau\mu^2\nu^3\lambda_{min}(Re(W_{\mathcal{B}^c_\nu}W^*_{\mathcal{B}^c_\nu})) + \rho},$$

where $\lambda \in spec(N^{-1}\nabla^2 f_\tau^\mu(x))$ *and* $\chi := 1 + \delta - \rho$*.*
If the eigenvectors of $N^{-\frac{1}{2}}\nabla^2 f_\tau^\mu(x)N^{-\frac{1}{2}}$ *belong to* $Ker(W^*_{\mathcal{B}^c_\nu})$*, then*

$$|\lambda - 1| \leq \frac{1}{2}\frac{\chi + 1 + (5\chi^2 - 2\chi + 1)^{\frac{1}{2}}}{\rho}.$$

Proof. We analyze the spectrum of matrix $N^{-\frac{1}{2}}\nabla^2 f_\tau^\mu(x)N^{-\frac{1}{2}}$ instead, because it has the same eigenvalues as matrix $N^{-1}\nabla^2 f_\tau^\mu(x)$. We have that:

$$\begin{aligned} N^{-\frac{1}{2}}\nabla^2 f_\tau^\mu(x)N^{-\frac{1}{2}} &= N^{-\frac{1}{2}}(\tau\nabla^2\psi_\mu(x) + A^\intercal A)N^{-\frac{1}{2}} \\ &= N^{-\frac{1}{2}}(\tau\nabla^2\psi_\mu(x) + A^\intercal A + \rho I_n - \rho I_n)N^{-\frac{1}{2}} \\ &= N^{-\frac{1}{2}}(\tau\nabla^2\psi_\mu(x) + \rho I_n)N^{-\frac{1}{2}} + N^{-\frac{1}{2}}A^\intercal A N^{-\frac{1}{2}} - \rho N^{-1} \\ &= I_n + N^{-\frac{1}{2}}A^\intercal A N^{-\frac{1}{2}} - \rho N^{-1}. \end{aligned}$$

Let u with $\|u\|_2 = 1$ be an eigenvector of $N^{-\frac{1}{2}}\nabla^2 f_\tau^\mu(x)N^{-\frac{1}{2}}$ corresponding to eigenvalue λ, then

$$\begin{aligned} (I_n + N^{-\frac{1}{2}}A^\intercal A N^{-\frac{1}{2}} - \rho N^{-1})u = \lambda u \quad &\iff \\ (N + N^{\frac{1}{2}}A^\intercal A N^{-\frac{1}{2}} - \rho I_n)u = \lambda N u \quad &\implies \\ u^\intercal N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}u = (\lambda - 1)u^\intercal N u \quad &\implies \\ |u^\intercal N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}u| = |\lambda - 1|u^\intercal N u. \quad &(5.18) \end{aligned}$$

First, we find an upper bound for $|u^\intercal N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}u|$. Matrices $N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}$ and $A^\intercal A - \rho I_n$ have the same eigenvalues. Therefore,

$$|u^\intercal N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}u| \leq \lambda^+_{max}(A^\intercal A - \rho I_n),$$

where $\lambda^+_{max}(\cdot)$ is the largest eigenvalue of the input matrix in absolute value.

Thus,

$$|u^\intercal N^{\frac{1}{2}}(A^\intercal A-\rho I_n)N^{-\frac{1}{2}}u| \leq \max_{\|v\|_2^2\leq 1} |v^\intercal(A^\intercal A-\rho I_n)v| \\ = \max_{\|Pv\|_2^2+\|Qv\|_2^2\leq 1} |(Pv+Qv)^\intercal(A^\intercal A-\rho I_n)(Pv+Qv)|,$$

where P is the projection matrix to the column space of $W_{\mathcal{B}_\nu}$ and $Q = I_n - P$. Using triangular inequality we get:

$$|u^\intercal N^{\frac{1}{2}}(A^\intercal A-\rho I_n)N^{-\frac{1}{2}}u| \leq \max_{\|Pv\|_2^2+\|Qv\|_2^2\leq 1} \big(|(Pv)^\intercal(A^\intercal A-\rho I_n)Pv| \\ + |(Qv)^\intercal(A^\intercal A-\rho I_n)Qv| + 2|(Pv)^\intercal(A^\intercal A-\rho I_n)Qv|\big).$$

Let us denote by $\hat{v}$ the solution of this maximization problem and set $\|P\hat{v}\|_2^2 = \alpha$ and $\|Q\hat{v}\|_2^2 = 1-\alpha$, where $\alpha \in [0,1]$, then

$$|u^\intercal N^{\frac{1}{2}}(A^\intercal A-\rho I_n)N^{-\frac{1}{2}}u| \leq \big(|(P\hat{v})^\intercal(A^\intercal A-\rho I_n)P\hat{v}| \\ + |(Q\hat{v})^\intercal(A^\intercal A-\rho I_n)Q\hat{v}| + 2|(P\hat{v})^\intercal(A^\intercal A-\rho I_n)Q\hat{v}|\big). \tag{5.19}$$

Since $P\hat{v}$ belongs to the column space of $W_{\mathcal{B}_\nu}$ and $|\mathcal{B}_\nu| = \sigma$, from W-RIP with $\delta_\sigma < 1/2$ we have that

$$\begin{aligned} \|P\hat{v}\|_2^2(1-\delta_\sigma) &\leq \|AP\hat{v}\|_2^2 && \Longrightarrow \\ \|P\hat{v}\|_2^2(1-\rho) &\leq \|AP\hat{v}\|_2^2 && \Longleftrightarrow \\ \|P\hat{v}\|_2^2(1-2\rho) &\leq \|AP\hat{v}\|_2^2 - \rho\|P\hat{v}\|_2^2. \end{aligned}$$

Since $\rho \in [\delta_\sigma, 1/2]$ we have that $\rho\|P\hat{v}\|_2^2 \leq \|AP\hat{v}\|_2^2$, which implies that if the eigenvector corresponding to an eigenvalue of matrix $A^\intercal A$ belongs to the column space of $W_{\mathcal{B}_\nu}$, then the eigenvalue cannot be smaller than ρ. Hence,

$$|(P\hat{v})^\intercal(A^\intercal A-\rho I_n)P\hat{v}| \leq |(P\hat{v})^*(A^\intercal A-\rho I_n)P\hat{v}| = (P\hat{v})^*(A^\intercal A-\rho I_n)P\hat{v}.$$

Moreover, from W-RIP with $\delta_\sigma < 1/2$ and $\rho \in [\delta_\sigma, 1/2]$, we also have that $(P\hat{v})^*(A^\intercal A-\rho I_n)P\hat{v} \leq \|P\hat{v}\|_2^2$. Thus,

$$|(P\hat{v})^\intercal(A^\intercal A-\rho I_n)P\hat{v}| \leq \alpha. \tag{5.20}$$

From property (2.17) and $\lambda_{max}(A^\intercal A) = \lambda_{max}(AA^\intercal)$, we have that $\lambda_{max}(A^\intercal A - \rho I_n) \leq 1+\delta-\rho$. Finally, using the Cauchy-Schwarz inequality, we get that

$$|(Q\hat{v})^\intercal(A^\intercal A-\rho I_n)Q\hat{v}| \leq (1+\delta-\rho)(1-\alpha) \tag{5.21}$$

and

$$|(P\hat{v})^\intercal(A^\intercal A-\rho I_n)Q\hat{v}| \leq (1+\delta-\rho)\sqrt{\alpha(1-\alpha)}. \tag{5.22}$$

Using (5.20), (5.21) and (5.22) in (5.19) we have that:

$$|u^\intercal N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}u| \leq \alpha + (1+\delta-\rho)(1-\alpha) + 2(1+\delta-\rho)\sqrt{\alpha(1-\alpha)}. \tag{5.23}$$

Set $\chi := 1+\delta-\rho$, it is easy to check that in the interval $\alpha \in [0,1]$ the right hand side of (5.23) has a maximum at one of the four candidate points:

$$\alpha_1 = 0, \quad \alpha_2 = 1, \quad \alpha_{3,4} = \frac{1}{2}(1 \pm \left(\frac{(\chi-1)^2}{5\chi^2 - 2\chi + 1}\right)^{1/2}),$$

where α_3 and α_4 correspond to the plus and minus signs, respectively. The corresponding function values are:

$$\chi, \quad 1, \quad \frac{\chi+1}{2} + \frac{1}{2}\frac{3\chi^2+2\chi-1}{(5\chi^2-2\chi+1)^{1/2}}, \quad \frac{\chi+1}{2} + \frac{1}{2}(5\chi^2-2\chi+1)^{1/2},$$

respectively. Hence, the maximum among these four values is given for α_4. Thus, (5.23) is upper bounded by

$$|u^\intercal N^{\frac{1}{2}}(A^\intercal A - \rho I_n)N^{-\frac{1}{2}}u| \leq \frac{\chi+1}{2} + \frac{1}{2}(5\chi^2-2\chi+1)^{\frac{1}{2}}. \tag{5.24}$$

We now find a lower bound for $u^\intercal N u$. Using the definition of D in (5.4), matrix $\hat{Y}$ in (5.6) is rewritten as $\hat{Y}_i = (2\mu^2 + |y_i|^2)D_i^3 \ \forall i = 1, 2, \cdots, l$. Thus $\nabla^2\psi_\mu(x)$ in (5.5) is rewritten as

$$\begin{aligned}\nabla^2\psi_\mu(x) = \frac{1}{4}[&(W\tilde{D}^3W^* + \bar{W}\tilde{D}^3\bar{W}^* + W\tilde{Y}\bar{W}^* + \bar{W}\tilde{\bar{Y}}W^*) \\ &+ 2\mu^2(WD^3W^* + \bar{W}D^3\bar{W}^*)],\end{aligned} \tag{5.25}$$

where $\tilde{D}_i = |y_i|^2 D_i^3 \ \forall i = 1, 2, \cdots, l$. Observe, that matrix $\nabla^2\psi_\mu(x)$ consists of two matrices $W\tilde{D}^3W^* + \bar{W}\tilde{D}^3\bar{W}^* + W\tilde{Y}\bar{W}^* + \bar{W}\tilde{\bar{Y}}W^*$ and $2\mu^2(WD^3W^* + \bar{W}D^3\bar{W}^*)$, which are positive semi-definite. Using (5.25) and the previous statement we get that:

$$\begin{aligned}u^\intercal N u &= u^\intercal(\tau\nabla^2\psi_\mu(x) + \rho I_n)u \\ &= \frac{\tau}{4}u^\intercal(W\hat{Y}W^* + \bar{W}\hat{Y}\bar{W}^* + W\tilde{Y}\bar{W}^* + \bar{W}\tilde{\bar{Y}}W^*)u + \rho \\ &\geq \frac{\tau\mu^2}{2}u^\intercal(WD^3W^* + \bar{W}D^3\bar{W}^*)u + \rho.\end{aligned}$$

Furthermore, using the splitting of matrix D (5.17), the last inequality is equivalent to:

$$\begin{aligned}u^\intercal N u &= \frac{\tau\mu^2}{2}u^\intercal(W_{\mathcal{B}_\nu}D^3_{\mathcal{B}_\nu}W^*_{\mathcal{B}_\nu} + W_{\mathcal{B}^c_\nu}D^3_{\mathcal{B}^c_\nu}W^*_{\mathcal{B}^c_\nu} + \bar{W}_{\mathcal{B}_\nu}D^3_{\mathcal{B}_\nu}\bar{W}^*_{\mathcal{B}_\nu} + \bar{W}_{\mathcal{B}^c_\nu}D^3_{\mathcal{B}^c_\nu}\bar{W}^*_{\mathcal{B}^c_\nu})u + \rho \\ &\geq \frac{\tau\mu^2}{2}u^\intercal(W_{\mathcal{B}^c_\nu}D^3_{\mathcal{B}^c_\nu}W^*_{\mathcal{B}^c_\nu} + \bar{W}_{\mathcal{B}^c_\nu}D^3_{\mathcal{B}^c_\nu}\bar{W}^*_{\mathcal{B}^c_\nu})u + \rho.\end{aligned}$$

Using the defition of $\mathcal{B}_\nu^c$ (5.17) in the last inequality, the quantity $u^\intercal N u$ is further lower bounded by

$$u^\intercal N u \geq \frac{\tau\mu^2\nu^3}{2} u^\intercal (W_{\mathcal{B}_\nu^c} W_{\mathcal{B}_\nu^c}^* + \bar{W}_{\mathcal{B}_\nu^c} \bar{W}_{\mathcal{B}_\nu^c}^*) u + \rho. \tag{5.26}$$

If $u \notin \mathrm{Ker}(W_{\mathcal{B}_\nu^c}^*)$, then from (5.26) we get:

$$u^\intercal N u \geq \tau\mu^2\nu^3 \lambda_{min}(\mathrm{Re}(W_{\mathcal{B}_\nu^c} W_{\mathcal{B}_\nu^c}^*)) + \rho. \tag{5.27}$$

Hence, combining (5.18), (5.24) and (5.27) we conclude that:

$$|\lambda - 1| \leq \frac{1}{2} \frac{\chi + 1 + (5\chi^2 - 2\chi + 1)^{\frac{1}{2}}}{\tau\mu^2\nu^3 \lambda_{min}(\mathrm{Re}(W_{\mathcal{B}_\nu^c} W_{\mathcal{B}_\nu^c}^*)) + \rho}.$$

If $u \in \mathrm{Ker}(W_{\mathcal{B}_\nu^c}^*)$, then from (5.26) we have that $u^\intercal N u \geq \rho$, hence

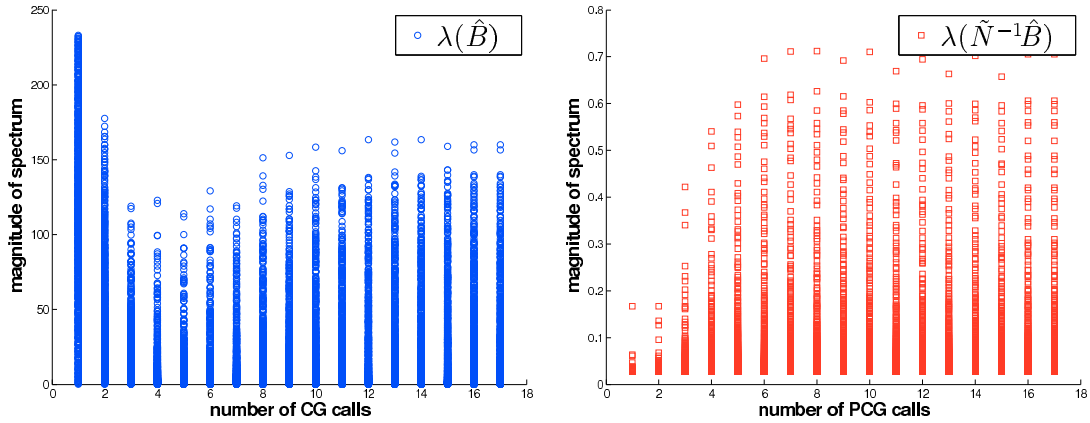
$$|\lambda - 1| \leq \frac{1}{2} \frac{\chi + 1 + (5\chi^2 - 2\chi + 1)^{\frac{1}{2}}}{\rho}.$$

□

Let us now draw some conclusions from Theorem 5.1. In order for the eigenvalues of $N^{-1}\nabla^2 f_\tau^\mu(x)$ to be around one, it is required that the degree of freedom ν is chosen such that $\nu = \mathcal{O}(1/\mu)$ and μ is small. For such ν, the cardinality σ of the set $\mathcal{B}_\nu$ must be small enough that matrices A and W satisfy W-RIP with constant $\delta_\sigma < 1/2$; otherwise the assumptions of Theorem 5.1 will not be satisfied. This is possible if the pdNCG iterates are close to the optimal solution $x_{\tau,\mu}$ and μ is sufficiently small. In particular, for sufficiently small μ, from Remark 5.1 we have that $x_{\tau,\mu} \approx x_\tau$ and $\sigma \approx q$. According to Assumption 2.1 for the q-sparse x_τ, W-RIP is satisfied for $\delta_{2q} < 1/2 \Longrightarrow \delta_q < 1/2$. Hence, for points close to $x_{\tau,\mu}$ and small μ we expect that $\delta_\sigma < 1/2$. Therefore, the result in Theorem 5.1 captures only the limiting behaviour of preconditioned $\nabla^2 f_\tau^\mu(x)$ as $x \to x_{\tau,\mu}$. Moreover, according to Remark 5.3, Theorem 5.1 implies that at the limit the eigenvalues of $\tilde{N}^{-1}\hat{B}$ are also clustered around one.

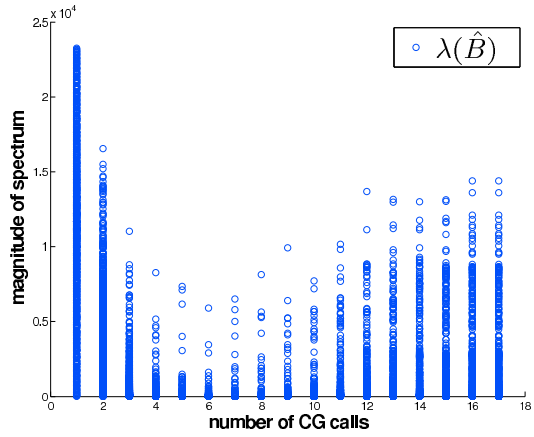
The scenario of limiting behaviour of the preconditioner is pessimistic. Let $\tilde{\sigma}$ be the minimum sparsity level such that matrices A and W are W-RIP with $\delta_{\tilde{\sigma}} < 1/2$. Then, according to the uniform property of W-RIP (i.e., it holds for all at most $\tilde{\sigma}$-sparse vectors), the preconditioner will start to be effective even if the iterates $W^* x^k$ are approximately sparse with $\tilde{\sigma}$ dominant non-zero components. Numerical evidence is provided in Figure 5.1 to confirm this claim.

In Figure 5.1 the spectra $\lambda(\hat{B})$ and $\lambda(\tilde{N}^{-1}\hat{B})$ are displayed for a sequence of systems which arise when an iTV problem is solved. For this iTV problem we set matrix A to be a partial $2D$ DCT, $n = 2^{10}$, $m = n/4$, $\tau = 2.29e\text{-}2$ and $\rho = 5.0e\text{-}1$. For the experiment in Figures 5.1a and 5.1b the smoothing parameter has been set to $\mu = 1.0e\text{-}3$ and in Figures 5.1c and 5.1d $\mu = 1.0e\text{-}5$. Observe that for both cases the eigenvalues of matrix $\tilde{N}^{-1}\hat{B}$ are *nicely clustered around one*. On the other hand the eigenvalues of matrix $\hat{B}$ have large variations and there are

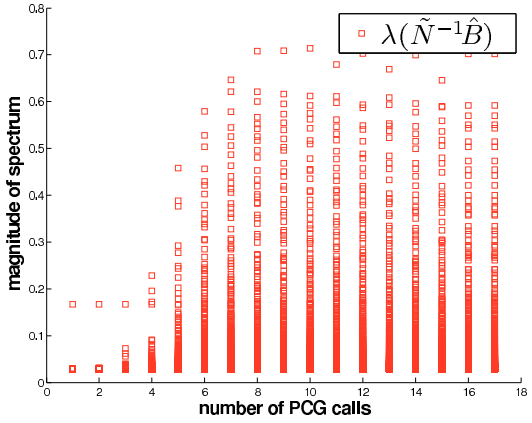


(a) Unpreconditioned

(b) Preconditioned



(c) Unpreconditioned



(d) Preconditioned

Figure 5.1: Spectra of $\lambda(\hat{B})$ and $\lambda(\tilde{N}^{-1}\hat{B})$ when pdNCG is applied with smoothing parameter $\mu = 1.0e\text{-}3$ (top subfigures) and $\mu = 1.0e\text{-}5$ (bottom subfigures). Matrix A in $\hat{B}$ is a $2D$ DCT, $n = 2^{10}$, $m = n/4$ and $\tau = 2.29e\text{-}2$. Seventeen systems are solved in total for each experiment.

many small eigenvalues close to zero. Notice that the preconditioner was effective not only at optimality as was predicted by theory, but through all iterations of pdNCG. This is because starting from the zero solution the iterates W^*x^k were maintained approximately sparse $\forall k$.

We now comment on the second result of Theorem 5.1, when the eigenvectors of $N^{-\frac{1}{2}}\nabla^2 f_\tau^\mu(x)N^{-\frac{1}{2}}$ belong to $\text{Ker}(W^*_{\mathcal{B}_\nu^c})$. In this case, according to Theorem 5.1 the preconditioner removes the disadvantageous dependence of the spectrum of $\nabla^2 f_\tau^\mu(x)$ on the smoothing parameter μ. However, there is no guarantee that the eigenvalues of $N^{-1}\nabla^2 f_\tau^\mu(x)$ are clustered around one, regardless of the distance from the optimal solution $x_{\tau,\mu}$. Again, because of Remark 5.3 we expect that the spectrum of $\tilde{N}^{-1}\hat{B}$ at the limit will have a similar behaviour.

5.7 Continuation

In the previous section we have shown that by using preconditioning, the spectral properties of systems that arise can be improved. However, for initial stages of pdNCG a similar result can be achieved without the cost of having to apply preconditioning. In particular, at initial stages the spectrum of $\hat{B}$ can be controlled to some extent through inexpensive continuation, while preconditioning is enabled only at later stages of the process. Briefly by continuation it is meant that a sequence of "easier" sub-problems is solved, instead of solving problem (5.2) directly. The reader is referred to Chapter 11 in [92] for a survey on continuation methods in optimization.

We use a similar continuation framework to [10, 31, 32, 62, 63]. In particular, a sequence of sub-problems is solved, where each of them is parameterized by τ and μ simultaneously. Let $\tilde{\tau}$ and $\tilde{\mu}$ be the final parameters for which problem (5.2) must be solved. Then the number of continuation iterations ϑ is set to be the maximum order of magnitude between $1/\tilde{\tau}$ and $1/\tilde{\mu}$, i.e., $\lfloor \max(|\log_{10}(1/\tilde{\tau})|, |\log_{10}(1/\tilde{\mu})|) \rfloor$. For instance, if $\tilde{\tau} = 1.0e\text{-}2$ and $\tilde{\mu} = 1.0e\text{-}5$ then $\vartheta := \max(2, 5) = 5$. If $\vartheta \geq 2$, then the initial parameters τ^0 and μ^0 are both always set to $1.0e\text{-}1$ and the intervals $[\tau^0, \tilde{\tau}]$ and $[\mu^0, \tilde{\mu}]$ are divided in ϑ equal subintervals in logarithmic scale. The pseudo-code of the proposed continuation framework is shown in Algorithm 5.2.

Algorithm 5.2 Continuation framework

1: **Outer loop:** For $j = 0, 1, 2, \ldots, \vartheta$, produce $(\tau^j, \mu^j)_{j=0}^{\vartheta}$.
2: **Inner loop:** Solve the sub-problem

$$\text{minimize } f_{\tau^j}^{\mu^j}(x),$$

approximately using pdNCG and by initializing it with the solution of the previous sub-problem.

Figure 5.2 shows the performance of pdNCG for three cases, no continuation with preconditioning, continuation with preconditioning through the whole process and continuation with preconditioning only at later stages. The vertical axis of Figure 5.2 shows the relative error $\|x^k - x_{\tilde{\tau},\tilde{\mu}}\|_2 / \|x_{\tilde{\tau},\tilde{\mu}}\|_2$. The optimal $x_{\tilde{\tau},\tilde{\mu}}$ is obtained by using pdNCG with parameter tuning set to recover a highly accurate solution. The horizontal axis shows the CPU time. The test example is an iTV problem where matrix A is a partial $2D$ DCT, $n = 2^{16}$, $m = n/4$, $\tau = 5.39e\text{-}2$ and $\rho = 5.0e\text{-}1$. The final smoothing parameter $\tilde{\mu}$ is set to $1.0e\text{-}5$. For the experiment preconditioning is used only at later stages of continuation; preconditioning is enabled when $\mu^j \leq 1.0e\text{-}4$, where j is the counter for continuation iterations. All experiments are terminated when the relative error $\|x^k - x_{\tilde{\tau},\tilde{\mu}}\|_2 / \|x_{\tilde{\tau},\tilde{\mu}}\|_2 \leq 1.0e\text{-}1$. Solving the problem approximately is an acceptable practise since the problem is very noisy (SNR is 10 dB) and there is not much improvement of the reconstructed image if more accurate solutions are requested. Finally, all other parameters of pdNCG were set to the same values for all three experiments. Observe in Figure 5.2 that continuation with preconditioning only at late stages was the best

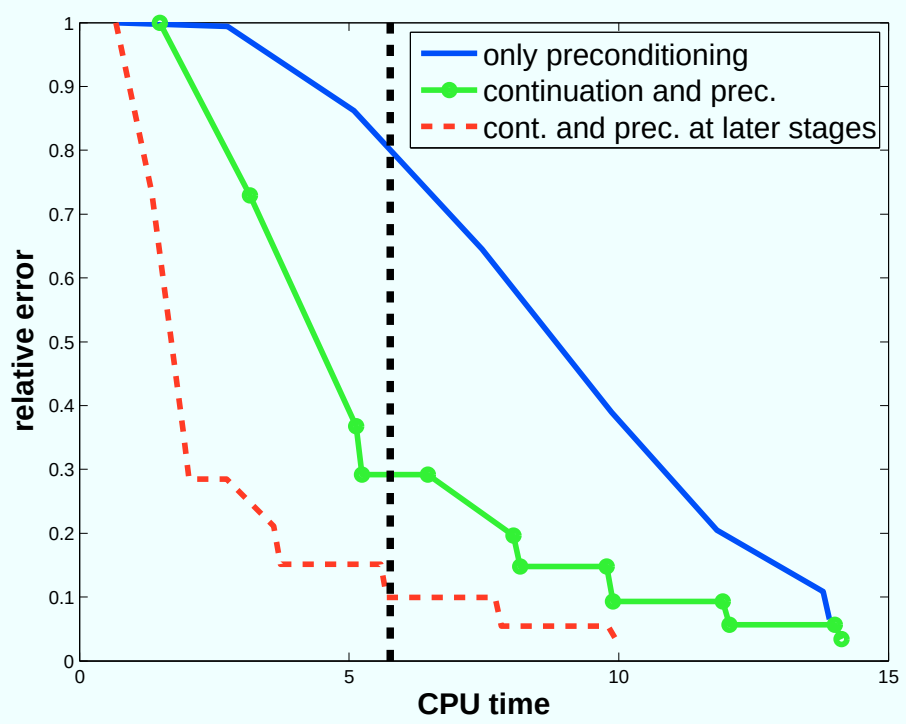


Figure 5.2: Performance of pdNCG for three different settings, i) no continuation with preconditioning, ii) continuation with preconditioning through all iterations and iii) continuation with preconditioning only at later stages. The vertical dashed black line shows when preconditioning is enabled. The vertical axis presents the relative error $\|x^k - x_{\tilde{\tau},\tilde{\mu}}\|_2 / \|x_{\tilde{\tau},\tilde{\mu}}\|_2$, where $x_{\tilde{\tau},\tilde{\mu}}$ is the optimal solution for the parameter setting $\tilde{\tau}$, $\tilde{\mu}$ in problem (5.2).

approach for this problem.

Chapter 6

Matrix-Free Interior-Point Method (mfIPM) for CS

In this chapter we discuss a primal-dual matrix-free Interior-Point Method (mfIPM) specialized for CS problems:

$$\text{minimize } \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{6.1}$$

where $A \in \mathbb{R}^{m\times n}$ is a measurement matrix, $b \in \mathbb{R}^m$ is the sampled signal and $\tau > 0$. See Section 2.2 for details regarding CS applications. The method was first presented in [54].

Primal-dual methods have been shown to have the best worst-case iteration complexity results [75] among various IPMs, but they also enjoy the best practical convergence [59, 116]. Here we give a brief introduction to the structure of primal-dual IPM methods and we discuss important modifications that result in the proposed approach. The actual implementation used in this chapter is given in Subsection 6.5.

Primal-dual interior-point methods rely on the Newton method to calculate primal-dual directions at each iteration. Newton method for primal-dual IPMs finds roots for linearized Karush-Kuhn-Tucker (KKT) systems or their reduced versions known as augmented and normal equations systems. These systems arise as first-order optimality conditions of log-barrier primal-dual pairs. The linearized KKT systems, referred to as Newton linear systems, can be solved in two ways:

- by employing a direct linear solver, or

- by using an iterative solver, such as Krylov subspace methods [72].

The first option delivers a very robust primal-dual IPM, where exact Newton directions are calculated. Despite its robustness this approach has the potential drawback of being computationally expensive, especially in the case when the Newton linear system does not have an exploitable sparsity pattern and the computational effort per iteration reaches $\mathcal{O}(n^3)$.

The second option involves the use of approximate Newton directions. Although this might slightly increase the number of IPM iterations [61, 79], one hopes that the decreased computational effort per iteration should offset such

a disadvantage. The performance of iterative methods depends on the spectral properties of the Newton linear system [72] and benefits from the use of appropriate preconditioning techniques which cluster the eigenvalues of the Newton linear system. If the Newton linear system is ill-conditioned and no low-cost preconditioner is applicable, then a direct approach might be more efficient. To conclude, a criterion to select between the two approaches of solving the Newton linear systems should take into account:

1. the sparsity pattern of the systems,
2. the existence of a computationally inexpensive preconditioner,
3. the memory requirements of storing problem data,
4. the existence of fast matrix-vector product implementations with the matrix of the linear system to be solved.

We focus on the situation where there is no particular sparsity pattern, the memory requirements can be high but conditions 2 and 4 are satisfied. For this reason, a preconditioned conjugate gradient method is more attractive than a direct method. Indeed, in the approach proposed in this thesis, at each step of the primal-dual IPM the preconditioned conjugate gradient method is applied to compute an approximate Newton direction. Since we rely on an iterative method for linear algebra, the proposed primal-dual IPM is matrix-free [60], i.e., the explicit problem formulation is avoided and the measurement matrix A is used only as an operator to produce results of matrix-vector products Ax and $A^{\mathsf{T}}y$. Although matrices A used in CS can be completely dense, i.e., Gaussian, partial Fourier, partial DCT, partial DST, Haar wavelets etc, they do have interesting (exploitable) features. Arguments 3 and 4 are satisfied because for many measurement matrices that appear in sparse signal reconstruction problems there are *super-fast algorithms (e.g. $\mathcal{O}(n)$ or $\mathcal{O}(n\log n)$ complexity) for multiplication by a vector*. For example, for Fourier, DCT and DST matrices there exists the Fastest Fourier Transform in the West (FFTW) implementation [56] with complexity $\mathcal{O}(n\log n)$, for Haar wavelet and Noiselet matrices there exist algorithms of complexity $\mathcal{O}(n)$, see [2] and [36], respectively. Finally, to satisfy argument 2 we propose a preconditioner efficient on certain problems that is based on the fact that *sub-matrices of A with a given number of columns are uniformly well-conditioned*, due to RIP, see the discussion in Section 2.2.1.

The objective of our developments is to design an IPM which preserves the main advantage of IPM, that is, it converges in merely a few iterations, and removes the main drawback of IPM, that is, avoids expensive computations of the Newton direction. Ideally, we would like to solve the CS problems in $\mathcal{O}(\log n)$ IPM iterations and keep the cost of a single IPM iteration as low as possible without exceeding $\mathcal{O}(n\log n)$ running time.

6.1 Problem reformulation

Problem (6.1) can be reformulated into an equivalent convex quadratic problem. This is achieved via linearization of the non-smooth ℓ_1-norm in the objective

function. In particular, new variables $u, v \in \mathbb{R}^n$ are defined such that:

$$|x_i| = u_i + v_i \quad \forall\, i = 1, 2, \ldots, n, \tag{6.2}$$

where $u_i = \max(x_i, 0)$ and $v_i = \max(-x_i, 0)$. Then linearization of the ℓ_1-norm is performed:

$$\|x\|_1 = 1_n^\mathsf{T} u + 1_n^\mathsf{T} v, \tag{6.3}$$

with $u, v \geq 0$ and $1_n \in \mathbb{R}^n$ being a column vector of all ones. Using the above linearization technique, we build the following constrained smooth reformulation of problem (6.1):

$$\begin{array}{ll} \text{minimize} & \tau 1_{2n}^\mathsf{T} z + \frac{1}{2}\|F^\mathsf{T} z - b\|_2^2 \\ \text{subject to:} & z \geq 0, \end{array} \tag{6.4}$$

where $z = [u\ ;\ v] \in \mathbb{R}^{2n}$, $F^\mathsf{T} = [A\ -A] \in \mathbb{R}^{m \times 2n}$. Once optimal values of variables u and v are found the solution x of the initial problem is retrieved by computing:

$$x = u - v.$$

The price for the linearization is that comparing to the initial problem (6.1) the dimension of the problem is doubled and $2n$ new non-negativity constraints are added. At each step of the algorithm a line-search is performed along the negative gradient direction and the new iterate is projected to the feasible set defined by the imposed constraints $z \geq 0$.

6.2 Framework of mfIPM

At every iteration of mfIPM the following problem is formulated:

$$\begin{array}{ll} \text{minimize} & f_\tau^\mu(z) := \tau 1_{2n}^\mathsf{T} z + \frac{1}{2}\|F^\mathsf{T} z - b\|_2^2 - \mu \sum_{i=1}^{2n} \log z_i \\ \text{subject to:} & z > 0, \end{array} \tag{6.5}$$

where $\mu > 0$. Problem (6.5) is obtained from problem (6.4) by introducing a negative logarithmic term for each variable z_i $i = 1, 2, \cdots, 2n$ in the objective function. The purpose of penalization is to maintain the variable z in the interior of the feasible set $\{z \in \mathbb{R}^{2n} \mid z \geq 0\}$, since the negative logarithmic term penalizes variables z_i that are close to zero. Penalization is controlled with the parameter μ, which is also known as the barrier parameter for IPMs. Parameter μ changes at every iteration and it is driven to zero as the algorithm converges.

At every iteration of mfIPM problem (6.5) is approximately minimized through the approximate solution of its KKT system:

$$\begin{aligned} c + FF^\mathsf{T} z - \mu Z^{-1} 1_{2n} &= 0, \\ z &> 0, \end{aligned}$$

where $c = \begin{bmatrix} \tau 1_n - A^\mathsf{T} b \\ \tau 1_n + A^\mathsf{T} b \end{bmatrix} \in \mathbb{R}^{2n}$ and Z is a diagonal matrix with vector z on its diagonal. By introducing extra variables $s = \mu Z^{-1} 1_{2n}$ the KKT system is

reformulated:

$$\begin{aligned} c + FF^{\intercal}z - s &= 0, \\ ZS &= \mu 1_{2n}, \\ z, s &> 0, \end{aligned} \tag{6.6}$$

where S is a diagonal matrix with vector s on its diagonal.

The previous KKT system is linearized and the linearization is solved approximately. The linearization of the KKT system (6.6) is:

$$\begin{bmatrix} FF^{\intercal} & -I_{2n} \\ S & Z \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta s \end{bmatrix} = \begin{bmatrix} r_z \\ r_s \end{bmatrix}, \tag{6.7}$$

where

$$r_z = s - c - FF^{\intercal}z, \quad r_s = \mu 1_{2n} - ZS1_{2n}, \tag{6.8}$$

and $(\Delta z, \Delta s)$ are the directions. Only one linearized system is solved approximately for every problem (6.5). This means that an approximate solution of problem (6.5) is obtained, then the parameter μ is updated and the process is repeated. Short- and long-step interior-point methods which satisfy this framework have the best known worst-case iteration complexity [61].

At this point it is interesting to point out the connection with GFrame (Algorithm 1.1) in Section (1.2). The most important step of Algorithm GFrame is the approximate minimization of a local convex approximation to the objective function at every iteration. Algorithm mfIPM adheres to this structure, since the approximate solution of the linearized KKT system (6.7) corresponds to the approximate minimization of a local convex approximation of problem (6.5). The local convex problem at point z^k is:

$$\begin{array}{ll} \text{minimize} & Q(z; z^k) \\ \text{subject to:} & z > 0, \end{array}$$

where

$$Q(z; z^k) := f_\tau^\mu(z^k) + (\nabla f_\tau^\mu(z^k))^{\intercal}(z - z^k) + \frac{1}{2}(z - z^k)^{\intercal}\nabla^2 f_\tau^\mu(z^k)(z - z^k)$$

and f_τ^μ is defined in (6.5). The previous problem is an approximation of the original problem (6.1), since problem (6.5) approximates problem (6.1) for small μ.

6.3 Primal–dual problems in mfIPM

In this section we explain the connection of mfIPM to a primal-dual IPM. The reader interested in the theory of primal-dual IPMs is referred to the book of Wright [116]. Aspects of practical implementation have been addressed in a recent survey [59].

For the primal problem (6.4) the dual is:

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2}z^{\mathsf{T}}FF^{\mathsf{T}}z \\ \text{subject to:} & c + FF^{\mathsf{T}}z - s = 0 \\ & z, s \geq 0 \end{array} \tag{6.9}$$

where s are the dual slack variables. The penalized version of the dual problem is:

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2}z^{\mathsf{T}}FF^{\mathsf{T}}z + \mu\sum_{i=1}^{2n}\log s_i \\ \text{subject to:} & c + FF^{\mathsf{T}}z - s = 0 \\ & z, s > 0. \end{array} \tag{6.10}$$

At every iteration of mfIPM the penalized primal-dual pair (6.5) and (6.10) is minimized approximately through the approximate solution of their common KKT system (6.6).

6.4 Preconditioned conjugate gradient method

In mfIPM the dual variables Δs in (6.7) are eliminated to get

$$(\Theta^{-1} + FF^{\mathsf{T}})\Delta z = r_z + Z^{-1}r_s, \tag{6.11a}$$

$$\Delta s = Z^{-1}r_s - \Theta^{-1}\Delta z. \tag{6.11b}$$

where $\Theta = S^{-1}Z \in \mathbb{R}^{2n\times 2n}$. The reduced Newton system (6.11a), also known as augmented system, is solved by an appropriate preconditioned iterative method for which *only matrix-vector products with the constraint matrix F are allowed.* Thus, the mfIPM approach has two major components:

- iterative solver for the augmented system,

- special-purpose preconditioner that exploits matrix structure.

The system (6.11a) has a symmetric positive definite matrix and the CG method can be employed to solve it in a matrix-free regime. However, the convergence of the CG method can be too slow when a matrix is ill-conditioned and/or its eigenvalues are not clustered. We discuss an efficient spectrally-equivalent diagonal matrix preconditioner for (6.11a). In particular, we give theoretical and practical justification of our approach to fast iterative solution of the system.

The proposed preconditioner for the system of equations (6.11a) is based on the exploitation of general properties of CS matrices and the behavior of the Θ matrix in (6.11a) close to optimality. Let us recall that in the notation of primal–dual pair (6.4)–(6.9), variable $s \in \mathbb{R}^{2n}$ is a Lagrange multiplier associated with the non-negativity constraint $z \geq 0$. Hence, at optimality $s_j z_j = 0 \;\forall\, j = 1, 2, \ldots, 2n$. IPMs force the convergence to the optimal solution by perturbing this condition $s_j z_j = \mu \;\forall\, j$, where μ is the barrier term of the IPM, and gradually reducing the perturbation μ to zero. At optimality indices $j \in \{1, 2, \ldots, 2n\}$ are split into two

disjoint sets:

$$\begin{gathered}\mathcal{B} = \{j \mid z_j \to z_j^* > 0, s_j \to s_j^* = 0\} \\ \text{and} \\ \mathcal{N} = \{j \mid z_j \to z_j^* = 0, s_j \to s_j^* > 0\}\end{gathered} \tag{6.12}$$

that determine the activity of constraints. This partitioning has highly undesirable consequences for the diagonal scaling matrix $\Theta = S^{-1}Z$. Indeed, when μ approaches zero, for indices $j \in \mathcal{B}$, Θ_j goes to infinity and for indices $j \in \mathcal{N}$, Θ_j goes to zero.

Recall that $z = [u \ ; \ v]$, where u and v are the positive and negative components of vector x (see (6.2)), respectively. For sparse signals there are merely q ($q \ll 2n$) non-zero components in the optimal solution. The positive ones will contribute a non-zero element in u and the negative ones will contribute a non-zero element in v. At optimality the cardinality of set $\mathcal{B}$ is q. Hence, at later iterations of an IPM we have that:

$$\begin{aligned}&\Theta_i \gg 1 \quad \forall\, i \in \mathcal{B}, \quad \operatorname{card}\mathcal{B} = q, \\ &\Theta_i \ll 1 \quad \forall\, i \in \mathcal{N}, \quad \operatorname{card}\mathcal{N} = 2n - q.\end{aligned} \tag{6.13}$$

Let us now return to the question of preconditioning of the system of equations (6.11a). Its matrix is:

$$H = \Theta^{-1} + FF^{\mathsf{T}}. \tag{6.14}$$

The behavior of matrix Θ near optimality is described by (6.13). It is clear that matrix Θ^{-1} has many large entries and only few small entries well before the IPM reaches the optimal solution. Let us introduce a number $C \gg 1$ that separates entries of Θ^{-1} of different magnitudes:

$$\#(\Theta_j^{-1} < C) = l. \tag{6.15}$$

Here l is just the number of small entries in Θ^{-1} and may be different from the sparsity q of the optimal solution. In the regime $l < m$, the second term FF^{T}, whose rank is exactly m, works as a low-rank pertubation for the matrix Θ^{-1} in (6.14). Since in Frobenius norm the first term Θ^{-1} dominates the second term FF^{T}, we propose to replace FF^{T} in the preconditioner by a simple approximant. First, let us write the matrix in system (6.11a) in the block form by using the facts that $\Theta = \operatorname{diag}(\Theta_u, \Theta_v)$ and $F^{\mathsf{T}} = [A \ -A]$,

$$H = \begin{bmatrix}\Theta_u^{-1} & \\ & \Theta_v^{-1}\end{bmatrix} + \begin{bmatrix}A^{\mathsf{T}}A & -A^{\mathsf{T}}A \\ -A^{\mathsf{T}}A & A^{\mathsf{T}}A\end{bmatrix}. \tag{6.16}$$

Our preconditioner is based on the approximation of $A^{\mathsf{T}}A$ by the closest (in Frobenius norm) scaled identity matrix ρI_n, $\rho = m/n$,

$$P = \begin{bmatrix}\Theta_u^{-1} + \rho I_n & -\rho I_n \\ -\rho I_n & \Theta_v^{-1} + \rho I_n\end{bmatrix}. \tag{6.17}$$

To simplify the analysis of the preconditioner, we first consider the case of $n \times n$ matrices H and P rather than block $2n \times 2n$ ones as defined by (6.16) and

(6.17). The following lemma establishes spectral properties of the preconditioned matrix $P^{-1}H$ in the non-block case.

In Lemma 6.1 spec$(\cdot)$ denotes the spectrum of the input matrix.

Lemma 6.1. *Define matrix H as:*

$$H = \Theta^{-1} + A^{\mathsf{T}}A,$$

where $\Theta = \mathrm{diag}(\Theta_1, \Theta_2, \ldots, \Theta_n)$ — diagonal $n \times n$ matrix with $\Theta_j > 0$, and A — $m \times n$ matrix with $m \leq n/2$. Let C be any positive constant and l be defined as in (6.15), $\#(\Theta_j^{-1} < C) = l$. Additionally, let A satisfy property (2.13) *defined for $q = l$ with some constant δ_l. If matrix A has orthonormal rows (2.9), then the eigenvalues of matrix H preconditioned by matrix P:*

$$P = \Theta^{-1} + \rho I_n, \quad \rho = m/n$$

are clustered around 1, i.e.,

$$|\lambda - 1| \leq \delta_l + \frac{1}{4}\frac{(3-\rho)^2}{\rho\delta_l C} \quad \forall\, \lambda \in \mathrm{spec}(P^{-1}H), \tag{6.18}$$

If matrix A has nearly orthonormal rows, i.e., satisfies property (2.12)*, then*

$$|\lambda - 1| \leq \delta_l + \frac{1}{4}\frac{(1+\delta-\rho+2\sqrt{1+\delta})^2}{\rho\delta_l C} \quad \forall\, \lambda \in \mathrm{spec}(P^{-1}H),$$

where δ has been defined in (2.12)*.*

Proof. Let C be any positive constant, then the following two disjoint sets of indices can be defined:

$$\mathcal{B}_C = \{j \in \{1, 2, \ldots, n\} :\ \Theta_j^{-1} < C\}, \quad \mathcal{N}_C = \{1, 2, \ldots, n\} \setminus \mathcal{B}_C.$$

Let B and N be matrices of columns of A with indices from $\mathcal{B}_C$ and $\mathcal{N}_C$, respectively. Without loss of generality we can assume that $\mathcal{B}_C$ are the first l indices, then

$$A = [B\ N], \quad B \in \mathbb{R}^{m\times l}, \quad N \in \mathbb{R}^{m\times(n-l)}.$$

Let λ be an eigenvalue of the preconditioned matrix $P^{-1}H$ corresponding to an eigenvector $v = [v_{\mathcal{B}_C}\ ;\ v_{\mathcal{N}_C}]$ of norm one, then

$$P^{-1}Hv = \lambda v \quad \Longleftrightarrow \quad (H-P)v = \gamma P v, \quad \gamma = \lambda - 1, \tag{6.19}$$

or, in the block form,

$$\left[\begin{array}{c|c} B^{\mathsf{T}}B - \rho I_l & B^{\mathsf{T}}N \\ \hline N^{\mathsf{T}}B & N^{\mathsf{T}}N - \rho I_{n-l} \end{array}\right] \left[\begin{array}{c} v_{\mathcal{B}_C} \\ \hline v_{\mathcal{N}_C} \end{array}\right] = \gamma \left[\begin{array}{c|c} \Theta_{\mathcal{B}_C}^{-1} + \rho I_l & 0 \\ \hline 0 & \Theta_{\mathcal{N}_C}^{-1} + \rho I_{n-l} \end{array}\right] \left[\begin{array}{c} v_{\mathcal{B}_C} \\ \hline v_{\mathcal{N}_C} \end{array}\right] \tag{6.20}$$

Obviously, eigenvalues of $P^{-1}H$ are all real, hence γ is also real. Multiplication

of (6.20) by $[v_{\mathcal{B}_C}\ ;\ v_{\mathcal{N}_C}]^\mathsf{T}$ from the left gives:

$$\gamma\Big[v_{\mathcal{B}_C}^\mathsf{T}\left(\Theta_{\mathcal{B}_C}^{-1}+\rho I_l\right)v_{\mathcal{B}_C}+v_{\mathcal{N}_C}^\mathsf{T}\left(\Theta_{\mathcal{N}_C}^{-1}+\rho I_{n-l}\right)v_{\mathcal{N}_C}\Big]= \\ v_{\mathcal{B}_C}^\mathsf{T}\left(B^\mathsf{T}B-\rho I_l\right)v_{\mathcal{B}_C}+v_{\mathcal{N}_C}^\mathsf{T}\left(N^\mathsf{T}N-\rho I_{n-l}\right)v_{\mathcal{N}_C}+2v_{\mathcal{B}_C}^\mathsf{T}B^\mathsf{T}Nv_{\mathcal{N}_C}. \quad (6.21)$$

Let us denote $\|v_{\mathcal{B}_C}\|_2^2$ by α, then $\|v_{\mathcal{N}_C}\|_2^2=1-\alpha$ since $v=[v_{\mathcal{B}_C}\ ;\ v_{\mathcal{N}_C}]$ has unit norm. Bounding left hand side of (6.21) from below is trivial:

$$\Big|\gamma\Big[v_{\mathcal{B}_C}^\mathsf{T}\left(\Theta_{\mathcal{B}_C}^{-1}+\rho I_l\right)v_{\mathcal{B}_C}+v_{\mathcal{N}_C}^\mathsf{T}\left(\Theta_{\mathcal{N}_C}^{-1}+\rho I_{n-l}\right)v_{\mathcal{N}_C}\Big]\Big|\geq|\gamma|\Big(\rho\alpha+C(1-\alpha)\Big). \quad (6.22)$$

Next, let us bound right hand side of (6.21) from above. We will distinguish two cases, orthonormal and nearly orthonormal rows of matrix A. First, we study the case of nearly orthonormal rows of matrix A. For this purpose we will use the SVD decompositions of matrices B and N:

$$B=U_B\Sigma_BV_B^\mathsf{T},\quad \Sigma_B=\begin{bmatrix}\mathrm{diag}(\sigma_1,\sigma_2,\ldots,\sigma_l)\\ \hline O_{m-l\times l}\end{bmatrix}$$

and

$$N=U_N\Sigma_NV_N^\mathsf{T},\quad \Sigma_N=\left[\ \mathrm{diag}(\varsigma_1,\varsigma_2,\ldots,\varsigma_m)\ \middle|\ O_{m\times(n-m-l)}\ \right].$$

RIP (2.13) implies that:

$$\sigma_1^2\leq\rho(1+\delta_l),\quad \sigma_l^2\geq\rho(1-\delta_l).$$

First, notice that:

$$\left|v_{\mathcal{B}_C}^\mathsf{T}\left(B^\mathsf{T}B-\rho I_l\right)v_{\mathcal{B}_C}\right|\leq\rho\delta_l\alpha. \quad (6.23)$$

Using property (2.12) we have:

$$\begin{aligned}\|AA^\mathsf{T}-I_m\|_2\leq\delta &\iff\\ \|AA^\mathsf{T}\|_2\leq1+\delta &\iff\\ \|BB^\mathsf{T}+NN^\mathsf{T}\|_2\leq1+\delta &\implies\\ \|NN^\mathsf{T}\|_2\leq1+\delta &\iff\\ \varsigma_1^2\leq1+\delta.&\end{aligned} \quad (6.24)$$

Next, using (6.24) we obtain:

$$\|N^\mathsf{T}N-\rho I_{n-l}\|_2\leq\max\{\rho,1+\delta-\rho\}=1+\delta-\rho,\quad(\rho=m/n\leq0.5)$$

and, hence,

$$\left|v_{\mathcal{N}_C}^\mathsf{T}\left(N^\mathsf{T}N-\rho I_{n-l}\right)v_{\mathcal{N}_C}\right|\leq\Big(1+\delta-\rho\Big)\Big(1-\alpha\Big). \quad (6.25)$$

Finally,

$$\|B^\mathsf{T}N\|_2\leq\|B\|_2\|N\|_2\leq\sigma_1\varsigma_1=\sqrt{1+\delta}\sqrt{\rho(1+\delta_l)}<\sqrt{(1+\delta)},$$

because our assumptions $m \leq n/2$ and $\delta_l < 1$ imply $\sigma_i^2 \leq \sigma_1^2 \leq \rho(1+\delta_l) < 1$. We conclude that

$$\left|2v_{\mathcal{B}_C}^{\mathsf{T}} B^{\mathsf{T}} N v_{\mathcal{N}_C}\right| < 2\sqrt{1+\delta}\sqrt{\alpha(1-\alpha)}. \tag{6.26}$$

Bounds (6.25) and (6.26) are sharp and can be used to obtain a very tight estimate of γ but we do not need them so sharp to obtain a sufficiently good estimate. So, we will relax them a little bit to simplify the analysis:

$$\begin{aligned}
&\left|v_{\mathcal{N}_C}^{\mathsf{T}}\left(N^{\mathsf{T}}N - \rho I_{n-l}\right) v_{\mathcal{N}_C}\right| \leq (1+\delta-\rho)(1-\alpha) \leq (1+\delta-\rho)\sqrt{1-\alpha}, \\
&\left|2v_{\mathcal{B}_C}^{\mathsf{T}} B^{\mathsf{T}} N v_{\mathcal{N}_C}\right| < 2\sqrt{1+\delta}\sqrt{\alpha(1-\alpha)} \leq 2\sqrt{1+\delta}\sqrt{1-\alpha}.
\end{aligned} \tag{6.27}$$

Using (6.22) and (6.23) and (6.27) we finally get:

$$|\gamma| \leq \frac{\rho\delta_l\alpha + (1+\delta-\rho+2\sqrt{1+\delta})\sqrt{1-\alpha}}{\rho\alpha + C(1-\alpha)} \leq \delta_l(1+\varepsilon). \tag{6.28}$$

Let us denote $\xi = (1+\delta-\rho+2\sqrt{1+\delta})$ and show that ε is small for large values of C. Indeed (6.28) implies that:

$$\xi\sqrt{1-\alpha} \leq \delta_l\Big(C + C\varepsilon - \rho\varepsilon\Big)(1-\alpha) + \rho\delta_l\varepsilon.$$

It can be checked by simple calculus, that $\sqrt{x} \leq C_1 x + C_2$ on $[0,1]$ whenever $C_1 \geq 1/(4C_2)$. In our case this implies:

$$\frac{\delta_l}{\xi}\Big(C + C\varepsilon - \rho\varepsilon\Big) \geq \frac{\xi}{4\rho\delta_l\varepsilon}.$$

The largest solution of the quadratic equation in ε:

$$\frac{4\rho\delta_l^2}{\xi^2}\varepsilon\Big(C + C\varepsilon - \rho\epsilon\Big) = 1$$

is

$$\varepsilon_+ = \frac{1}{2}\cdot\frac{C}{C-\rho}\left(\sqrt{1 + \frac{\xi^2}{\rho\delta_l^2 C}\cdot\frac{C-\rho}{C}} - 1\right) \leq \frac{\xi^2}{4\rho\delta_l^2 C}.$$

Hence, it is sufficient to take any $\varepsilon \geq \xi^2/(4\rho\delta_l^2 C)$ to satisfy the inequality (6.28):

$$\boxed{|\gamma| \leq \delta_l + \frac{\xi^2}{4\rho\delta_l C} = \delta_l + \frac{1}{4}\frac{(1+\delta-\rho+2\sqrt{1+\delta})^2}{\rho\delta_l C}}. \tag{6.29}$$

This completes the proof for matrix A which satisfies property (2.12). For the case of orthonormal rows of matrix A, i.e., $AA^{\mathsf{T}} = I_m$ simply set $\delta = 0$ in property (2.12) to get:

$$\boxed{|\gamma| \leq \delta_l + \frac{1}{4}\frac{(3-\rho)^2}{\rho\delta_l C}}. \tag{6.30}$$

This completes the proof. □

For the result of Lemma 6.1 to be useful we obviously need the bound in the right-hand side of inequalities in (6.29) and (6.30) to be sufficiently smaller than one. Let us take a closer look at the terms forming this bound. We are free to choose any value for the constant C we want, the larger the better. However, according to (6.15), l increases with the increase in C and, consequently, the restricted isometry constant δ_l also increases. Inequalities (6.29) and (6.30) hold for any value of C, hence we can replace them by:

$$|\gamma| \leq \min_C \left(\delta_l + \frac{1}{4} \frac{(1+\delta-\rho+2\sqrt{1+\delta})^2}{\rho \delta_l C} \right) \tag{6.31}$$

and

$$|\gamma| \leq \min_C \left(\delta_l + \frac{1}{4} \frac{(3-\rho)^2}{\rho \delta_l C} \right), \tag{6.32}$$

respectively, and choose constant C that delivers the minimum.

For number of measurements m just a fraction $\rho = 1/4$ of the length n of the unknown signal, it is natural to assume the restricted isometry constant δ_{2l} to be less than $1/4$ (see Theorem 2.3), hence, according to [15], $\delta_{2l} < 1/4$, implies $\delta_l < 1/4$. Therefore, to have $|\gamma| \leq 17/20$ we need $C = 20(0.75 + \delta + 2\sqrt{1+\delta})^2/3$ in (6.15). For nearly orthonormal rows of matrix A we can assume that $\delta \leq 1$, which gives us $C \approx 139.74$ and certainly holds near optimality in the IPM. For orthonormal rows of matrix A we have $\delta = 0$, hence, $C \approx 50.41$.

The bounds in (6.29) and (6.30) are rather pessimistic. Computational experience suggests that eigenvalues of the preconditioned matrix get well clustered around 1 as long as $l = \#(\Theta_j^{-1} < 1)$ is such that the RIP constant $\delta_l < 1$. For example, for the DCT matrix with $n = 2^{10}$ and $m = 2^8$ the corresponding $l \leq 74$ (this number is obtained in a series of random tests).

Now we are ready to state the spectral properties of the preconditioned matrix $P^{-1}H$ for the system of equations (6.11a).

Theorem 6.1. *Let H and P be block matrices defined in* (6.16) *and* (6.17)*, respectively. Moreover, let the eigenvalues of matrix $A^\intercal A$ be different from ρ. Then the preconditioned matrix $P^{-1}H$ has*

- *the eigenvalue* 1 *of multiplicity n;*

- *remaining n eigenvalues defined in Lemma 6.1 with $\Theta = \Theta_u + \Theta_v$.*

Proof. Initially we prove the first part. Let us define $\xi = [\xi_u, \xi_v] \in \mathbb{R}^{2n}$, where $\xi_u, \xi_v \in \mathbb{R}^n$. We will prove that if matrix $A^\intercal A$ does not have an eigenvalue equal to ρ then for $\xi_u = \xi_v$ the vector ξ is an eigenvector of matrix $P^{-1}H$, which corresponds to the eigenvalue that is equal to one of multiplicity n.

Let us assume that $\xi = [\xi_u, \xi_v]$ is an eigenvector of matrix $P^{-1}H$, which corresponds to the eigenvalue λ:

$$H\xi = \lambda P \xi.$$

Using (6.16) and (6.17) the equation becomes

$$\begin{bmatrix} \Theta_u^{-1} + A^\mathsf{T}A & -A^\mathsf{T}A \\ -A^\mathsf{T}A & \Theta_v^{-1} + A^\mathsf{T}A \end{bmatrix} \begin{bmatrix} \xi_u \\ \xi_v \end{bmatrix} = \lambda \begin{bmatrix} \Theta_u^{-1} + \rho I_n & -\rho I_n \\ -\rho I_n & \Theta_v^{-1} + \rho I_n \end{bmatrix} \begin{bmatrix} \xi_u \\ \xi_v \end{bmatrix}. \quad (6.33)$$

If $\lambda = 1$ the previous system reduces to:

$$(A^\intercal A - \rho I_n)(\xi_u - \xi_v) = 0.$$

If matrix $A^\intercal A$ does not have an eigenvalue equal to ρ, then matrix $A^\intercal A - \rho I_n$ is full rank. Hence, if $\xi_u = \xi_v$ we have that $\xi = [\xi_u, \xi_u]$ is an eigenvector of matrix $P^{-1}H$, which corresponds to the eigenvalue $\lambda = 1$. There exist at most n linearly independent vectors of the form $\xi = [\xi_u, \xi_u]$, hence, the eigenvalue $\lambda = 1$ has multiplicity n.

We now prove the second part. Multiplying the first equation of (6.33) with Θ_u and the second equation with Θ_v we get:

$$\begin{aligned} (I_n + \Theta_u A^\mathsf{T}A)\xi_u - \Theta_u A^\mathsf{T}A\xi_v &= \lambda((I_n + \rho\Theta_u)\xi_u - \rho\Theta_u\xi_v) \\ -\Theta_v A^\mathsf{T}A\xi_u + (I_n + \Theta_v A^\mathsf{T}A)\xi_v &= \lambda(-\rho\Theta_v\xi_u + (I_n + \rho\Theta_v)\xi_v). \end{aligned}$$

By subtracting the second equation from the first we get:

$$(\xi_u - \xi_v) + (\Theta_u + \Theta_v)A^\mathsf{T}A(\xi_u - \xi_v) = \lambda(I_n + \rho(\Theta_u + \Theta_v))(\xi_u - \xi_v).$$

Multiplying the last equation with $(\Theta_u + \Theta_v)^{-1}$ we get:

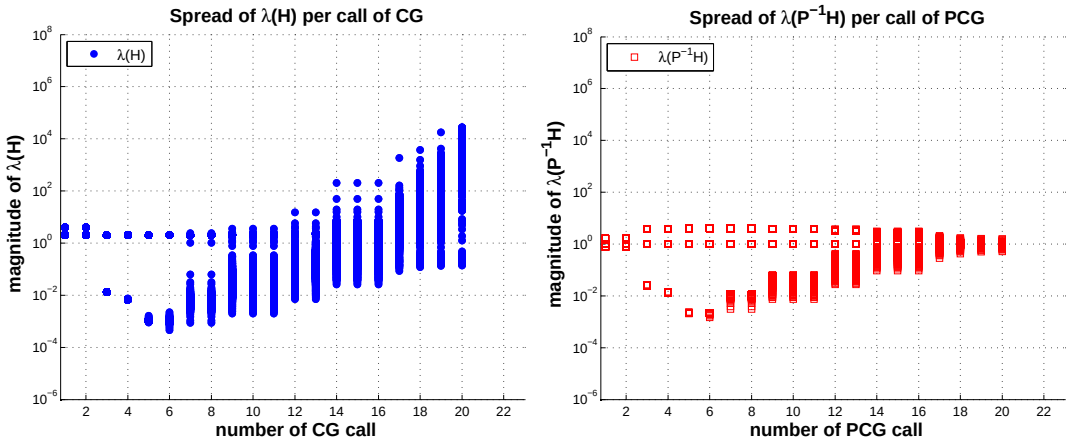
$$((\Theta_u + \Theta_v)^{-1} + A^\mathsf{T}A)(\xi_u - \xi_v) = \lambda((\Theta_u + \Theta_v)^{-1} + \rho I_n)(\xi_u - \xi_v).$$

Let us define $\Theta = (\Theta_u + \Theta_v)$, $\tilde{H} = ((\Theta_u + \Theta_v)^{-1} + A^\mathsf{T}A)$ and $\tilde{P} = ((\Theta_u + \Theta_v)^{-1} + \rho I_n)$. Then we have that:

$$\tilde{P}^{-1}\tilde{H}(\xi_u - \xi_v) = \lambda(\xi_u - \xi_v).$$

Therefore, $\xi_u - \xi_v$ is an eigenvector of matrix $\tilde{P}^{-1}\tilde{H}$, which corresponds to the eigenvalue λ. Hence, the matrices $\tilde{P}^{-1}\tilde{H}$ and $P^{-1}H$ have some of their eigenvalues equal. In particular, matrix $P^{-1}H$ is full rank and has $2n$ eigenvalues where n of them are equal to one. The remaining n eigenvalues are equal to the n eigenvalues of the full rank matrix $\tilde{P}^{-1}\tilde{H}$. Matrices $\tilde{P}$ and $\tilde{H}$ adhere to the structure of matrices that is assumed in Lemma 6.1. Therefore, the n distinct eigenvalues of matrix $\tilde{P}^{-1}\tilde{H}$ satisfy the result in Lemma 6.1. □

Theorem 6.1 establishes the clustering of eigenvalues of $P^{-1}H$ around 1. Hence, iterative method such as conjugate gradient applied to the system of equations (6.11a) is expected to converge in just a few iterations if the preconditioner P in (6.17) is used. The latter theoretical results are also confirmed in practical experiments. Figure 8.6 demonstrates clustering of eigenvalues $\lambda(H)$ and $\lambda(P^{-1}H)$ in the case that the A matrix in H (6.16) is a DCT matrix with normalized rows, $AA^T = I$. The parameters for the size of the problem are set to $m = 2^{10}$, $n = 2^{12}$ and the sparsity level is fixed to $q = 51$. In the left sub-



(a) Unpreconditioned systems, H (b) Preconditioned systems, $P^{-1}H$

Figure 6.1: Clustering of the eigenvalues for the matrices H and $P^{-1}H$ as the mfIPM approaches optimality. The matrix A in H (6.16) is a DCT matrix with normalized rows. The parameters of the problem set to $m = 2^{10}$,$n = 2^{12}$ and $q = 51$. Twenty systems for the matrices H and $P^{-1}H$ are solved in total

Figure 6.1a the clustering of the eigenvalues $\lambda(H)$ is shown. Every vertical line presents the spread of $\lambda(H)$ at a particular CG call as the mfIPM progresses. One can observe that the clustering worsens as the mfIPM approaches optimality. On the contrary, eigenvalues of the preconditioned matrices $P^{-1}H$ show the opposite behavior. In particular, as the mfIPM progresses eigenvalues $\lambda(P^{-1}H)$ start to cluster around one. The latter is depicted with the vertical columns in the right sub-Figure 6.1b.

6.5 Single centrality corrector primal-dual mfIPM

The implementation used in this chapter is a single-corrector primal-dual IPM [58]. The original version proposed in [58] makes use of multiple centrality correctors, however, after computational experimentation it was observed that a single corrector was enough for fast convergence of mfIPM in few iterations. In a standard multiple-corrector variant at every iteration multiple centrality corrector directions are calculated, which are combined with a predictor direction in order to produce the final primal-dual direction [58]. To compute the corrector and predictor directions one needs to solve multiple linear systems (6.11) where only the right hand side varies. In case that a direct solver is used to solve the linear systems, the extra cost of solving several equations instead of one is negligible, because the dominating cost is the decomposition of the matrix $(\Theta^{-1} + FF^{\mathsf{T}})$. However, this is not the case when PCG is used to solve systems (6.11). In particular, the cost of calculating every term in composite direction is approximately the same. In order to avoid the high cost of computing extra corrector directions at every iteration in our single-corrector mfIPM we slightly bias the predictor direction to point to the central path and perform corrector directions only when necessary. Like a long-step variant of primal-dual IPM [116] this guarantees that

at every iteration the objective function is decreased rapidly while the algorithm maintains the small distance to the central path. As proposed in [58], the criterion to decide whether a corrector direction is calculated is the value of the primal and dual step sizes. When many biased predictor directions are performed the primal-dual iterates tend to approach the boundary of the feasible region. This results in small step sizes of the subsequent iterations. When this happens a strong re-centering corrector is employed which pushes the next iteration to the vicinity of central path such that next step sizes are more likely to have large values. Ideally, the values of the step sizes of the primal and dual directions should be bounded away from zero while global convergence of the method is guaranteed. This would allow fast practical convergence of mfIPM, which translates into few iterations. Indeed, one can observe from the computational experience reported in Section 8.2.7 that 10 to 20 iterations of the mfIPM are enough for convergence. This behaviour has been observed also in all computational experiments discussed in Section 8.2.5. The pseudo-code of the implemented single-corrector primal-dual mfIPM follows.

Algorithm 6.1 Single-corrector primal-dual mfIPM

Input Choose $z^0, s^0 > 0$, $0 < \sigma_1 < \sigma_2 < \sigma_3 \leq 1$ and $0 < \tilde{\alpha} < \hat{\alpha} < 1$. For $k = 1, 2, \ldots$ generate z^{k+1} from z^k and s^{k+1} from s^k according to the iteration:

while Duality Gap of (6.4) and (6.9) $\geq \epsilon$ **do**

  if $k \neq 1$ and ($\alpha_P^{k-1} \leq \hat{\alpha}$ or $\alpha_D^{k-1} \leq \hat{\alpha}$) **then**

    $\sigma = \sigma_2$

  else

    $\sigma = \sigma_1$

  end if

  (* predictor step *)

  solve (6.11) using PCG with σ and z^k, s^k in (6.8) to obtain $(\Delta\hat{z}^k, \Delta\hat{s}^k)$

  choose primal and dual step sizes α_P^k, α_D^k in $[0, 1]$ as the largest values of α_P, α_D such that

$$z^k(\alpha_P^k) = z^k + \alpha_P \Delta\hat{z}^k > 0$$
$$s^k(\alpha_D^k) = s^k + \alpha_D \Delta\hat{s}^k > 0$$

  (* corrector step *)

  if $\alpha_P^k \leq \tilde{\alpha}$ or $\alpha_D^k \leq \tilde{\alpha}$ **then**

    solve (6.11) using PCG with $\sigma = \sigma_3$ and $z^k(\alpha_P^k), s^k(\alpha_D^k)$ in (6.8) to obtain $(\Delta\tilde{z}^k, \Delta\tilde{s}^k)$

    set $(\Delta z^k, \Delta s^k)$=$(\Delta\tilde{z}^k, \Delta\tilde{s}^k)$+$(\Delta\hat{z}^k, \Delta\hat{s}^k)$

    choose primal and dual step sizes α_P^k, α_D^k in $[0, 1]$ as the largest values of α_P, α_D such that

$$z^k(\alpha_P^k) = z^k + \alpha_P \Delta z^k > 0$$
$$s^k(\alpha_D^k) = s^k + \alpha_D \Delta s^k > 0$$

  end if

  set (z^{k+1}, s^{k+1})=$(z^k(\alpha_P^k), s^k(\alpha_D^k))$

end while

The input parameters σ_1, σ_2 are user-defined and control the centering bias of the predictor directions, while σ_3 parameter controls the strong centering in the corrector directions. For all experiments they have been set to $\sigma_1 = 0.1$, $\sigma_2 = 0.5$ and $\sigma_3 = 0.8$. The input parameters $\hat{\alpha}$ and $\tilde{\alpha}$ are user-defined, $\hat{\alpha}$ controls whether σ_1 or σ_2 will be used as a centering parameter for the predictor directions and $\tilde{\alpha}$ controls the frequency of the corrector updates. For all experiments they have been set to $\hat{\alpha} = 0.5$ and $\tilde{\alpha} = 0.1$.

Chapter 7

A Problem Generator for ℓ_1 Regularized Least-Squares

We present an instance generator for ℓ_1-LS problems:

$$\text{minimize } f_\tau(x) := \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{7.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $\tau > 0$.

The generator is inspired by the one presented in Section 6 of [91]. The advantage of our modified version is that it allows to control the properties of matrix A and the optimal solution $\tilde{x}$ of (7.1). For example, the sparsity of matrix A, its spectral decomposition, the sparsity and the norm of $\tilde{x}$, since A and $\tilde{x}$ are defined by the user. The conditioning of problem (7.1) is controlled through the singular values of matrix A. Additionally, the generator has very low memory requirements and scales well with the dimensions of the problem. We believe that the flexibility of the proposed generator will cover the need for generation of various good test problems.

First we describe how we measure the conditioning of the problem (7.1). Then the instance generator and examples are presented. We denote with span$(\cdot)$ the span of the columns of the input matrix. For simplicity, we assume that matrix A has more rows than columns, $m \geq n$, and it is full-rank. Extension to the case of matrix A with more columns than rows is easy and we briefly discuss when necessary. Moreover, $0 < \lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$ are the eigenvalues of the matrix $A^\intercal A$. We denote by $\lambda_{min}(\cdot)$ and $\lambda_{max}(\cdot)$ the smallest and the largest non-zero eigenvalues of the input symmetric matrix, respectively. We assume that $\tilde{x}$ has q non-zero elements, where $q \leq \min(m, n)$. Furthermore, let

$$S := \{i \in \{1, 2, \cdots, n\} \mid \tilde{x}_i \neq 0\} \tag{7.2}$$

with $|S| = q$. Additionally, we assume that $A_S \in \mathbb{R}^{m\times q}$, which is the collection of columns from matrix A with indices in S, has rank q.

The material in this chapter has been presented in [52].

7.1 Conditioning of the problem

Two factors are considered that affect the conditioning of the problem. First, the usual condition number of the second-order derivative of $1/2\|Ax-b\|_2^2$ in (7.1), which is simply $\kappa(A^\intercal A) = \lambda_1(A^\intercal A)/\lambda_n(A^\intercal A)$. It is well-known that the larger $\kappa(A^\intercal A)$ is, the more difficult problem (7.1) becomes.

Second, the conditioning of the optimal solution $\tilde{x}$ of problem (7.1). Let us explain what we mean by the conditioning of $\tilde{x}$. We define constant $\rho > 0$ and the index set $\mathcal{I}_\rho := \{i \in \{1, 2, \cdots, n\} \mid \lambda_i(A^\intercal A) \geq \rho\}$. Furthermore, we define the projection $P_\rho = G_\rho G_\rho^\intercal$, where $G_\rho \in \mathbb{R}^{n\times r}$, $r = |I_\rho|$ and matrix G_ρ has as columns the eigenvectors of matrix $A^\intercal A$, which correspond to eigenvalues with indices in $\mathcal{I}_\rho$. Then, the conditioning of $\tilde{x}$ is defined as:

$$\kappa_\rho(\tilde{x}) = \begin{cases} \frac{\|\tilde{x}\|_2}{\|P_\rho \tilde{x}\|_2}, & \text{if } P_\rho \tilde{x} \neq 0 \\ +\infty, & \text{otherwise.} \end{cases} \tag{7.3}$$

For the case $P_\rho \tilde{x} \neq 0$, the denominator of (7.3) is the mass of $\tilde{x}$ which exists in the space spanned by eigenvectors of $A^\intercal A$ which correspond to eigenvalues that are larger than or equal to ρ. Let us assume that there exists some ρ which satisfies $\lambda_n(A^\intercal A) \leq \rho \ll \lambda_1(A^\intercal A)$. If $\kappa_\rho(\tilde{x})$ is large, i.e., $\|P_\rho \tilde{x}\|_2$ is close to zero, then the majority of the mass of $\tilde{x}$ is "hidden" in the space spanned by eigenvectors which correspond to eigenvalues that are smaller than ρ, i.e., the orthogonal space of span(G_ρ). In this case we expect the performance of first-order methods to degrade, since curvature information which corresponds to small eigenvalues is not used by first-order methods. In Subsection 8.1.5 we empirically verify this.

If matrix A has more columns than rows then the previous definitions of conditioning of problem (7.1) are incorrect and need to be adjusted. Indeed, if $m < n$ and $\text{rank}(A) = \min(m, n) = m$ then $A^\intercal A$ is a rank deficient matrix, which has m non-zero eigenvalues and $n - m$ zero eigenvalues. We can restrict the conditioning of the problem to a neighbourhood of the optimal solution of $\tilde{x}$. We define a neighbourhood of $\tilde{x}$ so that all points in this neighbourhood have non-zeros at the same indices as $\tilde{x}$ and zeros elsewhere, i.e., $\mathcal{N} := \{x \in \mathbb{R}^n \mid x_i \neq 0 \ \forall i \in S,\ x_i = 0 \ \forall i \in S^c\}$. In this case, an important feature to determine the conditioning of the problem is the ratio of the largest and the smallest non-zero eigenvalues of $A_S^\intercal A_S$, where A_S is a sub-matrix of A built of columns of matrix A which belong to set S.

7.2 Instance generator for $m \geq n$

Given $\tau > 0$, $A \in \mathbb{R}^{m\times n}$, $\tilde{x} \in \mathbb{R}^n$, the generator returns a vector $b \in \mathbb{R}^m$ such that $\tilde{x} := \arg\min_x f_\tau(x)$. For simplicity we assume that the given matrix A has rank n and $m \geq n$. The generator is described in the Procedure IGen below.

Procedure 7.1 Instance Generator (IGen)

1: Initialize $\tau > 0$, $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and rank n, $\tilde{x} \in \mathbb{R}^n$
2: Construct $g \in \mathbb{R}^n$ such that $g \in \partial\|\tilde{x}\|_1$:

$$g_i \in \begin{cases} \{1\}, & \text{if } \tilde{x}_i > 0 \\ \{-1\}, & \text{if } \tilde{x}_i < 0 \\ [-1,1], & \text{if } \tilde{x}_i = 0 \end{cases} \quad \forall i = 1, 2, \cdots, n \tag{7.4}$$

3: Set $e = \tau A(A^\intercal A)^{-1} g$
4: Return $b = A\tilde{x} + e$

In Procedure IGen, given τ, A and $\tilde{x}$ we are aiming to find a vector b such that $\tilde{x}$ satisfies the optimality conditions of problem (7.1):

$$A^\intercal(A\tilde{x} - b) \in -\tau\partial\|\tilde{x}\|_1,$$

where $\partial\|x\|_1 \in [-1,1]^n$ is the sub-differential of the ℓ_1-norm at point x. By fixing a sub-gradient $g \in \partial\|\tilde{x}\|_1$ as defined in (7.4) and setting $e = b - A\tilde{x}$, the previous optimality conditions can be written as:

$$A^\intercal e = \tau g. \tag{7.5}$$

The solution to the under-determined system (7.5) is set to $e = \tau A(A^\intercal A)^{-1} g$ and then we simply obtain $b = A\tilde{x} + e$; Steps 3 and 4 in IGen, respectively. Notice that for a general matrix A, Step 3 of IGen can be very expensive. Fortunately, by using elementary linear transformations, such as Givens rotations, we can iteratively construct a sparse matrix A with a known singular value decomposition and guarantee that the inversion of matrix $A^\intercal A$ in Step 3 of IGen is trivial. We provide a more detailed argument in Section 7.4.

7.3 Instance generator for $m < n$

In this section we extend the instance generator that was proposed in Section 7.2 to the case of matrix $A \in \mathbb{R}^{m\times n}$ with more columns than rows, i.e., $m < n$. Given $\tau > 0$, $B \in \mathbb{R}^{m\times m}$, $N \in \mathbb{R}^{m\times n-m}$ and $\tilde{x} \in \mathbb{R}^n$, the generator returns a vector $b \in \mathbb{R}^m$ and a matrix $A \in \mathbb{R}^{m\times n}$ such that $\tilde{x} := \arg\min_x f_\tau(x)$. For this generator we need to discuss first some restrictions on matrix A and the optimal solution $\tilde{x}$.

Matrix $A_S \in \mathbb{R}^{m\times q}$ must have rank q otherwise problem (7.1) would not be well-defined. To see this, let $\text{sign}(\tilde{x}_S) \in \mathbb{R}^q$ be the sign function applied component-wise to $\tilde{x}_S$, where $\tilde{x}_S$ is a vector with components of $\tilde{x}$ that correspond to indices in S. Then problem (7.1) reduces to the following:

$$\underset{y}{\text{minimize}} \;\; \tau\text{sign}(\tilde{x}_S)^\intercal y + \frac{1}{2}\|A_S y - b\|_2^2, \tag{7.6}$$

where $y \in \mathbb{R}^q$. The first-order stationary points of problem (7.6) satisfy:

$$A_S^\intercal A_S y = -\tau \text{sign}(\tilde{x}_S) + A_S^\intercal b.$$

If $\text{rank}(A_S) < q$, the previous linear system does not have a unique solution and problem (7.1) does not have a unique minimizer. Having this restriction in mind, let us now present the instance generator for $m < n$ in Procedure IGen2 below.

Procedure 7.2 Instance Generator 2 (IGen2)

1: Initialize $\tau > 0$, $B \in \mathbb{R}^{m\times m}$ with rank m, $N \in \mathbb{R}^{m\times n-m}$, $\tilde{x} \in \mathbb{R}^n$ with $S := \{1, 2, \cdots, q\}$ and $q \leq m$

2: Construct $g \in \mathbb{R}^m$ such that $g \in \partial\|\tilde{x}(1, 2, \cdots, m)\|_1$:

$$g_i \in \begin{cases} \{1\}, & \text{if } \tilde{x}_i > 0 \\ \{-1\}, & \text{if } \tilde{x}_i < 0 \\ [-1, 1], & \text{if } \tilde{x}_i = 0 \end{cases} \quad \forall i = 1, 2, \cdots, m \tag{7.7}$$

3: Set $e = \tau B^{-\intercal} g$

4: Let $\tilde{N}_i$ be the i^{th} column of matrix $\tilde{N}$, then construct matrix $\tilde{N} \in \mathbb{R}^{m\times n-m}$ with the following loop:
For $i = 1, 2, \ldots, n - m$

$$\tilde{N}_i = \frac{\tau}{|N_i^\intercal e|} N_i$$

end-for

5: Return $A = [B, \tilde{N}]$ and $b = A\tilde{x} + e$

In IGen2, given τ, B, N and $\tilde{x}$ we are aiming in finding a vector b and a matrix $\tilde{N}$ such that for $A = [B, \tilde{N}]$, $\tilde{x}$ satisfies the optimality conditions of problem (7.1):

$$A^\intercal(A\tilde{x} - b) \in -\tau\partial\|\tilde{x}\|_1.$$

Without loss of generality it is assumed that all non-zero components of $\tilde{x}$ correspond to indices in $S = \{1, 2, \cdots, q\}$. By fixing a partial sub-gradient $g \in \partial\|\tilde{x}(1, 2, \cdots, m)\|_1$ as in (7.7), where $\tilde{x}(1, 2, \cdots, m) \in \mathbb{R}^m$ is a vector which consists of the first m components of $\tilde{x}$, and defining a vector $e = b - A\tilde{x}$, the previous optimality conditions can be written as:

$$e = \tau B^{-\intercal} g \quad \text{and} \quad \tilde{N}^\intercal e \in \tau[-1, 1]^{n-m}. \tag{7.8}$$

It is easy to check that by defining $\tilde{N}$ as in Step 4 of IGen2 conditions (7.8) are satisfied. Finally, we obtain $b = A\tilde{x} + e$.

Similarly to IGen in Subsection 7.2, for Step 3 in IGen2 we have to perform a matrix inversion, which generally can be an expensive operation. However, in the next section we discuss techniques how this matrix inversion can be executed using a sequence of elementary orthogonal transformations.

7.4 A Paradigm for construction of matrix A

In this subsection we provide a paradigm on how matrix A can be constructed inexpensively such that its singular value decomposition is known and its sparsity is controlled. We examine the case of instance generator IGen where $m \geq n$. The paradigm can be easily extended to the case of IGen2, where $m < n$.

Let $\Sigma \in \mathbb{R}^{m \times n}$ be a rectangular matrix with the singular values $\sigma_1, \sigma_2, \cdots, \sigma_n$ on its diagonal and zeros elsewhere:

$$\Sigma = \begin{bmatrix} \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \\ \hline O_{m-n \times n} \end{bmatrix},$$

where $O_{m-n \times n} \in \mathbb{R}^{m-n \times n}$ is a matrix of zeros, and let $G(i, j, \theta) \in \mathbb{R}^{n \times n}$ be a Givens rotation matrix, which rotates plane i-j by an angle θ:

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix},$$

where $i, j \in \{1, 2, \cdots, n\}$, $c = \cos\theta$ and $s = \sin\theta$. Given a sequence of Givens rotations $\{G(i_k, j_k, \theta_k)\}_{k=1}^{K}$ we define the following composition of them:

$$G = G(i_1, j_1, \theta_1) G(i_2, j_2, \theta_2) \cdots G(i_K, j_K, \theta_K).$$

Similarly, let $\tilde{G}(l, p, \vartheta) \in \mathbb{R}^{m \times m}$ be a Givens rotation matrix, $l, p \in \{1, 2, \cdots, m\}$ and

$$\tilde{G} = \tilde{G}(l_1, p_1, \vartheta_1) \tilde{G}(l_2, p_2, \vartheta_2) \cdots \tilde{G}(l_{\tilde{K}}, p_{\tilde{K}}, \vartheta_{\tilde{K}})$$

be a composition of $\tilde{K}$ Givens rotations. Using G and $\tilde{G}$ we define matrix A as

$$A = (P_1 \tilde{G} P_2) \Sigma G^{\intercal}, \tag{7.9}$$

where $P_1, P_2 \in \mathbb{R}^{m \times m}$ are permutation matrices. Since the matrices $P_1 \tilde{G} P_2$ and G are orthonormal it is clear that the left singular vectors of matrix A are the columns of $P_1 \tilde{G} P_2$, Σ is the matrix of singular values and the right singular vectors are the columns of G. Hence, in Step 3 of IGen we simply set $(A^{\intercal}A)^{-1} = G(\Sigma^{\intercal}\Sigma)^{-1}G^{\intercal}$, which means that Step 3 in IGen costs two matrix-vector products with G and a diagonal scaling with $(\Sigma^{\intercal}\Sigma)^{-1}$. Moreover, the sparsity of matrix A is controlled by the numbers K and $\tilde{K}$ of Givens rotations, the type, i.e. (i, j, θ) and (l, p, ϑ), and the order of Givens rotations. Also, notice that the sparsity of matrix $A^{\intercal}A$ is controlled only by matrix G. Examples are given in Subsection 7.4.1.

It is important to mention that other settings of matrix A in (7.9) could be used, for example different combinations of permutation matrices and Givens

rotations. The setting chosen in (7.9) is flexible, it allows for an inexpensive construction of matrix A and makes the control of the singular value decomposition and the sparsity of matrices A and $A^\intercal A$ easy.

Notice that matrix A does not have to be calculated and stored. In particular, in case that the method which is applied to solve problem (1.1) requires only matrix-vector product operations using matrices A and $A^\intercal$, one can simply consider matrix A as an operator. It is only required to predefine the triplets $(i_k, j_k, \theta_k)\ \forall k = 1, 2, \cdots, K$ for matrix G, the triplets $(l_k, p_k, \theta_k)\ \forall k = 1, 2, \cdots, \tilde{K}$ for matrix $\tilde{G}$ and the permutation matrices P_1 and P_2. This implies that the generator is inexpensive in terms of memory requirements. Examples of matrix-vector product operations with matrices A and $A^\intercal$ in case of (7.9) are given below in Algorithms MvPA and MvPAt, respectively.

Algorithm Matrix-vector product with A (MvPA)

Given a matrix A defined as in (7.9) and an input vector $x \in \mathbb{R}^n$, do
Set $y_0 = x$
For $k = 1, 2, \ldots, K$
    $y_k = G_k^\intercal y_{k-1}$
end-for
Set $\tilde{y}_0 = P_2 \Sigma y_K$
For $k = 1, 2, \ldots, \tilde{K}$
    $\tilde{y}_k = \tilde{G}_{\tilde{K}-k+1} \tilde{y}_{k-1}$
end-for
Return $P_1 \tilde{y}_{\tilde{K}}$

Algorithm Matrix-vector product with $A^\intercal$ (MvPAt)

Given a matrix A defined as in (7.9) and input vector $y \in \mathbb{R}^m$, do
Set $\tilde{x}_0 = P_1^\intercal y$
For $k = 1, 2, \ldots, \tilde{K}$
    $\tilde{x}_k = \tilde{G}_k^\intercal \tilde{x}_{k-1}$
end-for
Set $x_0 = \Sigma^\intercal P_2^\intercal \tilde{x}_{\tilde{K}}$
For $k = 1, 2, \ldots, K$
    $x_k = G_{K-k+1} x_{k-1}$
end-for
Return x_K

7.4.1 An example

Let us assume that m, n are divisible by two and $m \geq n$. Given the singular values matrix Σ and rotation angles θ and ϑ, we construct matrix A as

$$A = (P\tilde{G}P)\Sigma G^\intercal,$$

where P is a random permutation of the identity matrix, G is a composition of $n/2$ Givens rotations:

$$G = G(i_1, j_1, \theta)G(i_2, j_2, \theta)\cdots, G(i_k, j_k, \theta), \cdots, G(i_{n/2}, j_{n/2}, \theta)$$

with

$$i_k = 2k - 1, \quad j_k = 2k \quad \text{for} \quad k = 1, 2, 3, \cdots, \frac{n}{2}$$

and $\tilde{G}$ is a composition of $m/2$ Givens rotations:

$$\tilde{G} = \tilde{G}(l_1, p_1, \vartheta)\tilde{G}(l_2, p_2, \vartheta)\cdots, \tilde{G}(l_k, p_k, \vartheta), \cdots, \tilde{G}(l_{m/2}, p_{m/2}, \vartheta)$$

with

$$l_k = 2k - 1, \quad p_k = 2k \quad \text{for} \quad k = 1, 2, 3, \cdots, \frac{m}{2}.$$

Notice that the angle θ is the same for all Givens rotations G_k, this means that the total memory requirement for matrix G is low. In particular, it consists only of the storage of four values: $\cos\theta$, $\cos\vartheta$, $\sin\vartheta$ and $\sin\theta$. Similarly, the memory requirement for matrix $\tilde{G}$ is also low.

7.4.2 Control of sparsity of matrix A

We now present examples in which we demonstrate how sparsity of matrix A can be controlled through Givens rotations.

In the example of Subsection 7.4.1, two compositions of $n/2$ and $m/2$ Givens rotations, denoted by G and $\tilde{G}$, are applied on an initial diagonal rectangular matrix Σ. If $n = 2^3$ and $m = 2n$ the sparsity pattern of the resulting matrix $A = (P\tilde{G}P)\Sigma G^\intercal$ is given in Subfigure 7.1a and has 28 non-zero elements, while the sparsity pattern of matrix $A^\intercal A$ is given in Subfigure 7.2a and has 16 non-zero elements. Notice in this subfigure that the coordinates can be clustered in pairs of coordinates $(1, 2)$, $(3, 4)$, $(5, 6)$ and $(7, 8)$. One could apply another stage of Givens rotations. For example, one could construct matrix $A = (P\tilde{G}\tilde{G}_2P)\Sigma(G_2G)^\intercal$, where

$$G_2 = G(i_1, j_1, \theta)G(i_2, j_2, \theta)\cdots, G(i_k, j_k, \theta), \cdots, G(i_{n/2-1}, j_{n/2-1}, \theta)$$

with

$$i_k = 2k, \quad j_k = 2k + 1 \quad \text{for} \quad k = 1, 2, 3, \cdots, \frac{n}{2} - 1.$$

and

$$\tilde{G}_2 = \tilde{G}(l_1, p_1, \theta)\tilde{G}(l_2, p_2, \theta)\cdots, \tilde{G}(l_k, p_k, \theta), \cdots, \tilde{G}(l_{m/2-1}, p_{m/2-1}, \theta)$$

with

$$l_k = 2k, \quad p_k = 2k + 1 \quad \text{for} \quad k = 1, 2, 3, \cdots, \frac{m}{2} - 1.$$

Matrix $A = (P\tilde{G}\tilde{G}_2P)\Sigma(G_2G)^\intercal$ has 74 non-zeros and it is shown in Subfigure 7.1b, while matrix $A^\intercal A$ has 38 non-zeros and it is shown in Subfigure 7.2b. By rotating again we obtain the matrix $A = (P\tilde{G}\tilde{G}_2\tilde{G}P)\Sigma(GG_2G)^\intercal$ in Subfig-

ure 7.1c with 104 non-zero elements and matrix $A^\intercal A$ in Subfigure 7.2c with 56 non-zero elements. Finally, the fourth Subfigures 7.1d and 7.2d show matrix $A = (P\tilde{G}_2\tilde{G}\tilde{G}_2\tilde{G}P)\Sigma(G_2GG_2G)^\intercal$ and $A^\intercal A$ with 122 and 62 non-zero elements, respectively.

7.5 Paradigms for construction of an optimal solution

Two different techniques are employed to generate the optimal solution $\tilde{x}$. The first procedure is a simple random generation of $\tilde{x}$, see Procedure OsGen below.

Procedure 7.5 Optimal solution Generator (OsGen)

1: Given the required number $q \leq \min(m, n)$ of non-zeros in $\tilde{x}$ and a positive constant $\gamma > 0$ do:
2: Choose any subset $S \subseteq \{1, 2, \cdots, n\}$ with $|S| = q$.
3: $\forall i \in S$ choose $\tilde{x}_i$ uniformly at random in $[-\gamma, \gamma]$ and $\forall j \notin S$ set $\tilde{x}_j = 0$.

The second more complicated procedure is given below.

Procedure 7.6 Optimal Solution Generator 2 (OsGen2)

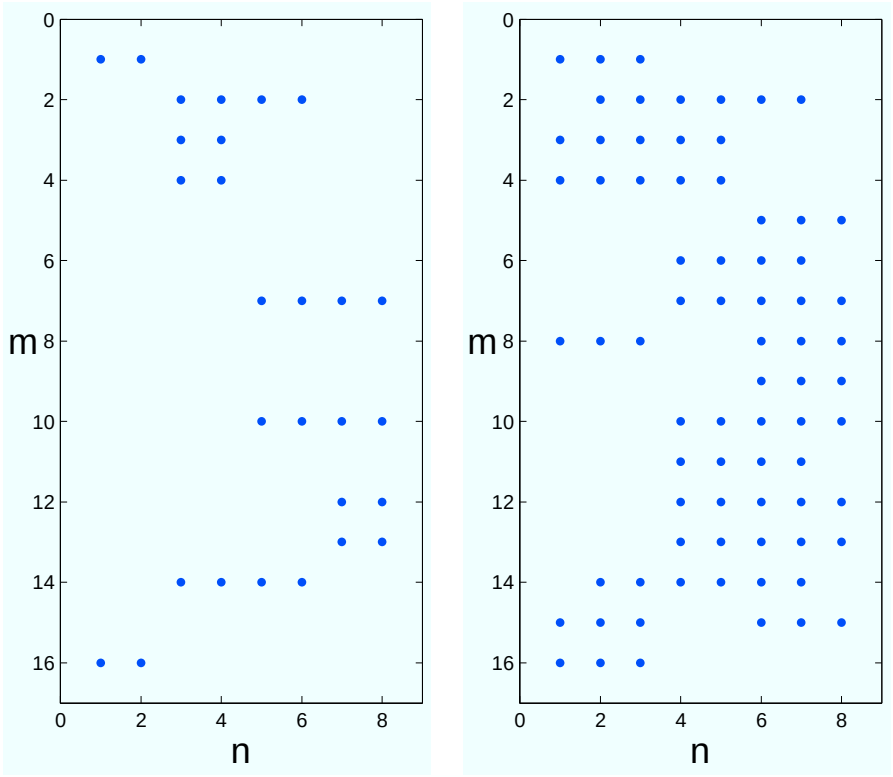
1: Given the required number $q \leq \min(m, n)$ of non-zeros in $\tilde{x}$, a positive constant $\gamma > 0$, the right singular vectors G and singular values Σ of matrix A do:
2: Solve approximately

$$\begin{aligned} \tilde{x} := \quad & \underset{x \in \mathbb{R}^n}{\arg\min} && \|G^\intercal x - \gamma(\Sigma^\intercal \Sigma)^{-1} 1_n\|^2 \\ & \text{subject to:} && \|x\|_0 \leq q, \end{aligned} \tag{7.10}$$

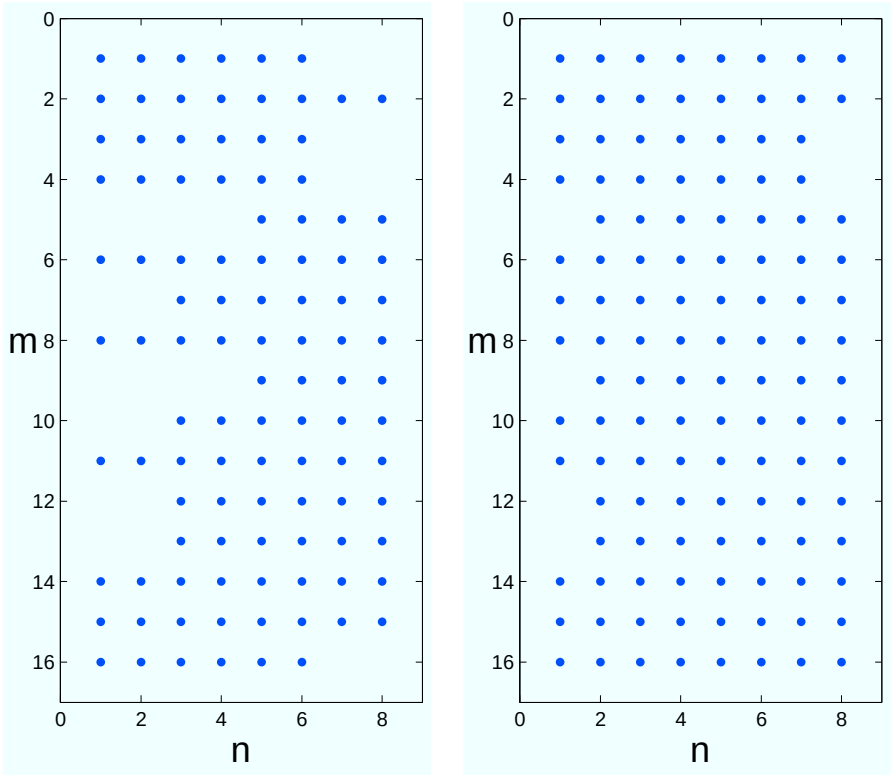
where $1_n \in \mathbb{R}^n$ is a vector of ones and $\|\cdot\|_0$ is the zero norm which returns the number of non-zero components of the input vector. Problem (7.10) can be solved using an Orthogonal Matching Pursuit (OMP) [85] solver implemented in [8].

The aim of Procedure OsGen2 is to find a sparse $\tilde{x}$ with $\kappa_\rho(\tilde{x}) \approx 1$ for some ρ in the interval $\lambda_n(A^\intercal A) \leq \rho \ll \lambda_1(A^\intercal A)$. In particular, Procedure OsGen2 will return a sparse $\tilde{x}$ which can be expressed as $\tilde{x} = Gv$. The coefficients v are close to the inverse of the eigenvalues of matrix $A^\intercal A$. Intuitively, this technique will create an $\tilde{x}$ which has strong dependence on subspaces which correspond to small eigenvalues of $A^\intercal A$. The constant γ is used in order to control the norm of $\tilde{x}$.

The sparsity constraint in problem (7.10), i.e., $\|x\|_0 \leq s$, makes the approximate solution of this problem difficult when we use OMP, especially in the case that q and n are large. To avoid this expensive task we can ignore the sparsity constraint in (7.10). Then we can solve exactly and inexpensively the unconstrained problem and finally we can project the obtained solution in the feasible set defined

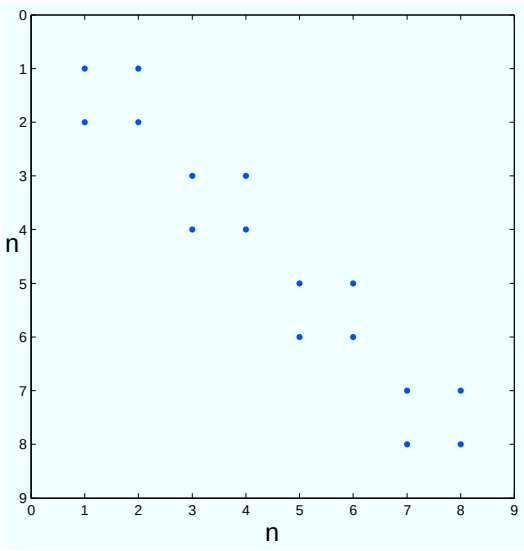


(a) $A = (P\tilde{G}P)\Sigma G^\intercal$ (b) $A = (P\tilde{G}_2\tilde{G}P)\Sigma(G_2G)^\intercal$

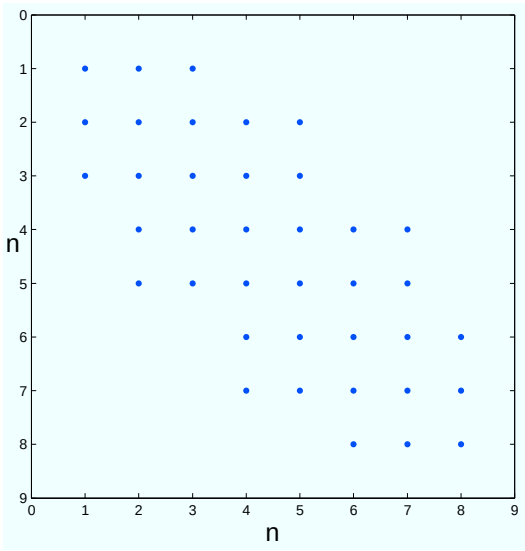


(c) $A = (P\tilde{G}\tilde{G}_2\tilde{G}P)\Sigma(GG_2G)^\intercal$ (d) $A = (P\tilde{G}_2\tilde{G}\tilde{G}_2\tilde{G}P)\Sigma(G_2GG_2G)^\intercal$

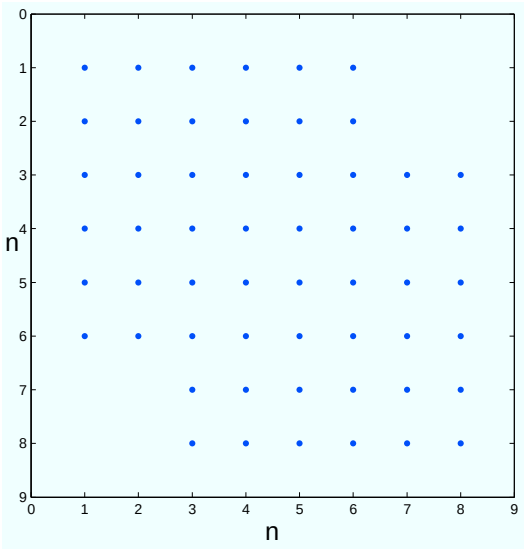
Figure 7.1: Sparsity pattern of four examples of matrix A, the Givens rotations G and G_2 are explained in Subsections 7.4.1 and 7.4.2.



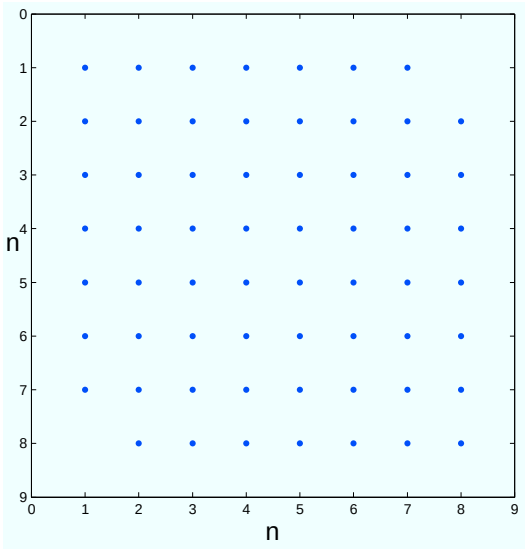
(a) $A^\intercal A$, $A = (P\tilde{G}P)\Sigma G^\intercal$



(b) $A^\intercal A$, $A = (P\tilde{G}_2\tilde{G}P)\Sigma(G_2G)^\intercal$



(c) $A^\intercal A$, $A = (P\tilde{G}\tilde{G}_2\tilde{G}P)\Sigma(GG_2G)^\intercal$



(d) $A^\intercal A$, $A = (P\tilde{G}_2\tilde{G}\tilde{G}_2\tilde{G}P)\Sigma(G_2GG_2G)^\intercal$

Figure 7.2: Sparsity pattern of four examples of matrix $A^\intercal A$, where the Givens rotations G and G_2 are explained in Subsections 7.4.1 and 7.4.2

.

by the sparsity constraint. Obviously, there is no guarantee that the projected solution is a good approximation to the one obtained in Step 2 of Procedure OsGen2. However, for all experiments in Section 8.1 that we applied this modification we obtained large $\kappa_\rho(\tilde{x})$. This means that we maintained our objective to produce ill-conditioned optimal solutions, while we keep the computational costs low. The modified version of Procedure OsGen2 is given in Procedure OsGen3.

Procedure 7.7 Optimal solution Generator 3 (OsGen3)

1: Given the required number $q \leq \min(m, n)$ of non-zeros in $\tilde{x}$, two non negative integers q_1 and q_2 such that $q_1 + q_2 = q$, a positive constant $\gamma > 0$, the right singular vectors G and singular values Σ of matrix A do:

2: Solve exactly

$$\tilde{x} := \underset{x \in \mathbb{R}^n}{\arg\min} \quad \|G^\intercal x - \gamma(\Sigma^\intercal \Sigma)^{-1} 1_n\|^2 \tag{7.11}$$

where $1_n \in \mathbb{R}^n$ is a vector of ones. Problem (7.11) can be solved exactly and inexpensively because $G^\intercal$ is an orthonormal matrix.

3: Maintain the positions and the values of the q_1 smallest and q_2 largest (in absolute values) components of $\tilde{x}$.

4: Set the remaining components of $\tilde{x}$ to zero.

Chapter 8

Numerical experiments

In this chapter we present numerical experiments on large-scale synthetic and real world instances for applications that were discussed in Chapter 2. The performance of mfIPM and pdNCG is compared to those of other state-of-the-art solvers. Every section in this chapter corresponds to an application of our interest which was presented in Chapter 2. All the experiments in this chapter have been presented in [41, 53, 52, 54].

8.1 Data fitting

In this section we present the performance of state-of-the-art methods on synthetic ℓ_1-LS problems. All problems are generated using the instance generator IGen 7.1, which was described in Chapter 7. All numerical experiments can be reproduced by downloading the software from: `http://www.maths.ed.ac.uk/ERGO/trillion/`.

8.1.1 State-of-the-art methods

A number of efficient first- [6, 34, 69, 96, 97, 103, 108, 109, 110, 117, 118] and second-order [19, 53, 60, 101, 102] methods have been developed for the solution of problem (4.1). In this section we examine the performance of the following state-of-the-art methods.

- Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [6] is an optimal first-order method for problem (4.1), which adheres with the structure of GFrame Algorithm 1.1. At a point x, FISTA builds a convex function

$$Q(y;x) := \tau\|y\|_1 + \varphi(x) + (\nabla\varphi(x))^\intercal(y-x) + \frac{L}{2}\|y-x\|_2^2,$$

  where L is the Lipschitz constant of $\nabla\varphi$. Then the sub-problem (1.2) with such objective function Q is solved exactly using shrinkage-thresholding [30, 80]. An efficient implementation of this algorithm can be found as part of the TFOCS (Templates for First-Order Conic Solvers) package [11] under the name N83. In this implementation the parameter L is obtained dynamically.

- Parallel Coordinate Descent Method (PCDM) [96] is a randomized parallel coordinate descent method. The parallel updates are performed asynchronously and the coordinates to be updated are chosen uniformly at random. Let ϖ be the number of processors that are used by PCDM. Then, at a point x PCDM builds ϖ convex approximations:

$$Q_i(y_i; x) := \tau|y_i| + \varphi(x) + [\nabla\varphi(x)]_i(y_i - x_i) + \frac{L_i}{2}(y_i - x_i)^2,$$

$\forall i = 1, 2, \cdots, \varpi$, where $[\nabla\varphi(x)]_i$ is the ith component of $\nabla\varphi(x)$ and $L_i = [A^\intercal A]_{ii}$ is the coordinate-Lipschitz constant for the ith coordinate. The Q_i functions are minimized exactly using shrinkage-thresholding.

- Projected Scaled Sub-gradient, Gafni-Bertsekas (PSSgb) variant [102] is a higher-order method. At each iteration of PSSgb the coordinates are separated into two sets, the working set $\mathcal{W}$ and the active set $\mathcal{A}$. The working set consists of all coordinates for which, the current point x is non-zero. The active set is the complement of the working set $\mathcal{W}$. The following local quadratic model is build at each iteration:

$$Q(y; x) := f_\tau(x) + \vartheta^\intercal(y - x) + \frac{1}{2}(y - x)^\intercal H(y - x),$$

where ϑ is a sub-gradient of f_τ (the objective function of problem (4.1)) at point x with the minimum Euclidean norm. Moreover, matrix H is defined as:

$$H = \begin{bmatrix} H_{\mathcal{W}} & 0 \\ 0 & H_{\mathcal{A}} \end{bmatrix},$$

where $H_{\mathcal{W}}$ is an L-BFGS (Limited memory Broyden-Fletcher-Goldfarb-Shanno) Hessian approximation with respect to the coordinates $\mathcal{W}$ and $H_{\mathcal{A}}$ is a positive diagonal matrix. The local model is minimized exactly since the inverse of matrix H is known due to properties of the L-BFGS Hessian approximation $H_{\mathcal{W}}$.

8.1.2 Implementation details

All solvers are implemented in MATLAB. The experiments in Subsections 8.1.4, 8.1.5, 8.1.6 were performed on a Dell PowerEdge R920 running Redhat Enterprise Linux with four Intel Xeon E7-4830 v2 2.2GHz processors, 20 MB Cache, 7.2 GT/s QPI, Turbo (4x10Cores).

The huge scale experiments in Subsection 8.1.8 were performed on a Cray XC30 MPP supercomputer. This work has made use of the resources provided by ARCHER (`http://www.archer.ac.uk/`), made available through the Edinburgh Compute and Data Facility (ECDF) (`http://www.ecdf.ed.ac.uk/`). According to the most recent list of commercial supercomputers, which is published in TOP500 list (`http://www.top500.org`), ARCHER is currently the $25th$ fastest supercomputer worldwide out of 500 supercomputers. ARCHER has a total of $118,080$ cores with performance $1,642.54$ TFlops/s on LINPACK benchmark and

$2,550.53$ TFlops/s theoretical peak performance. The most computationally demanding experiments which are presented in Subsection 8.1.8 required more than half of the cores of ARCHER, i.e., $65,536$ cores out of $118,080$.

8.1.3 Parameter tuning

For all solvers we use the default parameter setting and they are all initialized to the zero solution. We run pdNCG for sufficient time such that the problems are adequately solved. Then, the rest of the methods are terminated when the objective function f_τ in (4.1) is below the one obtained by pdNCG or when a predefined maximum number of iterations limit is reached. All comparisons are presented in figures which show the progress of the objective function or the relative error against the wall clock time. This way, the reader can compare the performance of the solvers for various levels of accuracy. We use logarithmic scales for the wall clock time and terminate runs which do not converge in about 10^5 seconds, i.e., approximately 27 hours.

8.1.4 Increasing condition number of $A^\intercal A$

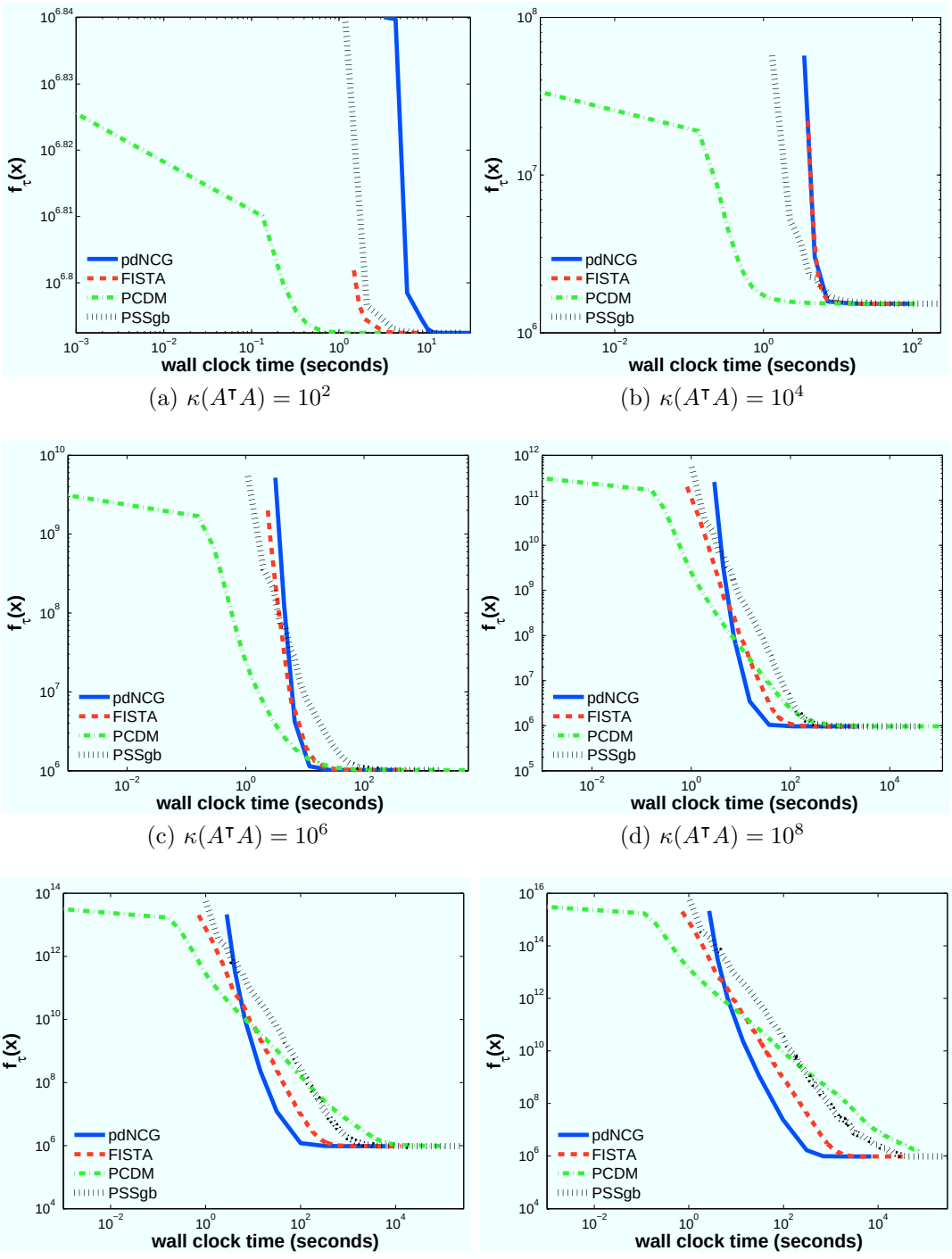
In this experiment we present the performance of FISTA, PCDM, PSSgb and pdNCG for increasing condition number of matrix $A^T A$ when the Procedure 7.5 OsGen is used to construct the optimal solution $\tilde{x}$ of problem (4.1). We generate six matrices A and two instances of $\tilde{x}$ for every matrix A; twelve instances in total.

The singular values of A are chosen uniformly at random in the intervals $[0, 10^\vartheta]$, where $\vartheta = 0, 1, \cdots, 5$, respectively, for each of the six matrices A. Then, all singular values are shifted by 10^{-1}. This resulted in a condition number of matrix $A^\intercal A$ which varies from 10^2 to 10^{12}. Matrix A has $n = 2^{22}$ columns, $m = 2n$ rows and rank n. The rotation angle θ of matrix G in Subsection 7.4.1 is set to $2\pi/3$ radians. The optimal solution $\tilde{x}$ has $q = n/2^7$ non-zero components for all twelve instances.

For the first set of six instances we set $\gamma = 10$ in Procedure 7.5 OsGen, which resulted in $\kappa_{0.1}(\tilde{x}) \approx 1$. The results are presented in Figure 8.1. For these instances PCDM is clearly the fastest for $\kappa(A^\intercal A) \leq 10^4$, while for $\kappa(A^\intercal A) \geq 10^6$ pdNCG is the most efficient.

For the second set of six instances we set $\gamma = 10^3$ in Procedure 7.5 OsGen, which resulted in the same $\kappa_{0.1}(\tilde{x})$ as before for all experiments. The results are presented in Figure 8.2. For these instances PCDM is the most efficient for $\kappa(A^\intercal A) \leq 10^2$, while pdNCG is the fastest for $\kappa(A^\intercal A) \geq 10^4$.

Notice in Figures 8.1 and 8.2 that the performance of pdNCG and FISTA is nearly parallel, although the methods are fundamentally different. In practice, the two methods exploit in a different way the trade-off between number of iterations and computational complexity per iteration. In particular, pdNCG required less than 30 iterations for all experiments, where every iteration costs a few matrix-vector products (the exact number depends on PCG), while FISTA required thousands of iterations, where every iteration costs two matrix-vector products.

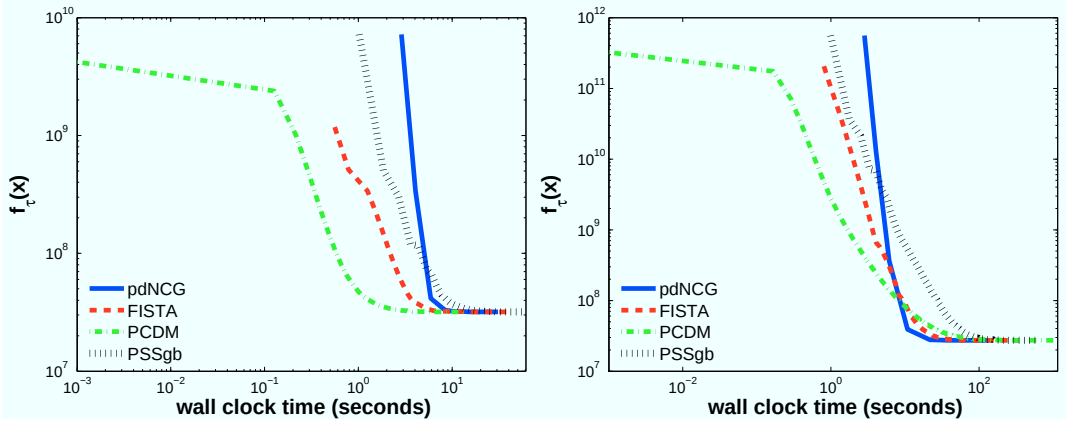


(a) $\kappa(A^\intercal A) = 10^2$ (b) $\kappa(A^\intercal A) = 10^4$

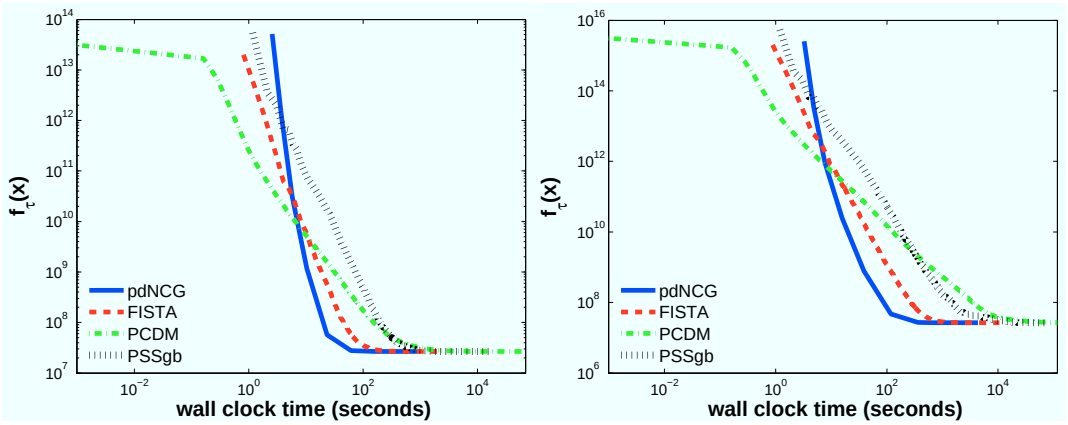
(c) $\kappa(A^\intercal A) = 10^6$ (d) $\kappa(A^\intercal A) = 10^8$

(e) $\kappa(A^\intercal A) = 10^{10}$ (f) $\kappa(A^\intercal A) = 10^{12}$

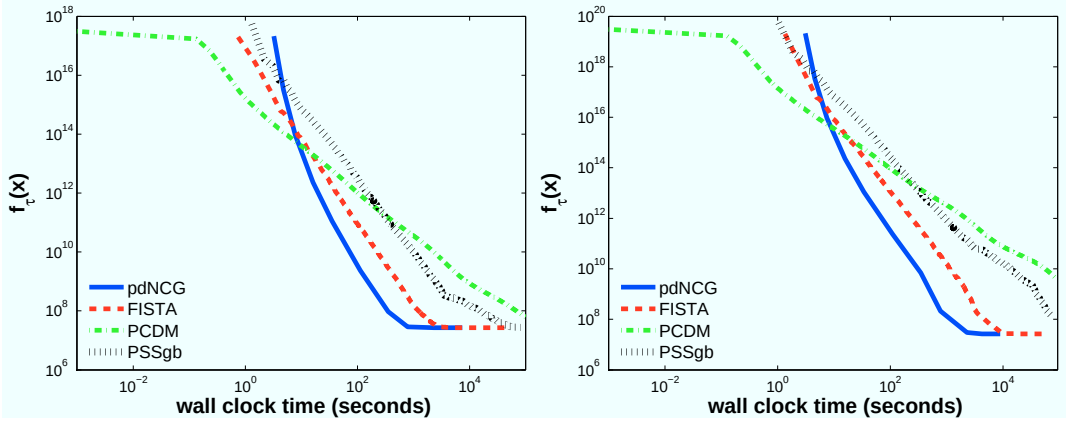
Figure 8.1: Performance of pdNCG, FISTA, PCDM and PSSgb on synthetic ℓ_1-LS problems for increasing condition number of matrix $A^\intercal A$ and $\gamma = 10$ in Procedure 7.5 OsGen. The axis are in log-scale. In this figure f_τ denotes the objective value that was obtained by each solver



(a) $\kappa(A^\intercal A) = 10^2$ (b) $\kappa(A^\intercal A) = 10^4$



(c) $\kappa(A^\intercal A) = 10^6$ (d) $\kappa(A^\intercal A) = 10^8$



(e) $\kappa(A^\intercal A) = 10^{10}$ (f) $\kappa(A^\intercal A) = 10^{12}$

Figure 8.2: Performance of pdNCG, FISTA, PCDM and PSSgb on synthetic ℓ_1-LS problems for increasing condition number of matrix $A^\intercal A$ and $\gamma = 10^3$ in Procedure 7.5 OsGen. The axis are in log-scale

8.1.5 Increasing condition number of $A^\intercal A$: non-trivial construction of the optimal solution

We now perform a similar experiment as in Subsection 8.1.4. In particular, six instances of $(A, \tilde{x})$ are constructed, where m, n and q are maintained the same. The singular values of matrices A are generated the same way as in the experiment in Subsection 8.1.4. However, the rotation angle θ of matrix G is set to $2\pi/10^3$ radians. The optimal solution $\tilde{x}$ is constructed using Procedure 7.7 OsGen3 with $\gamma = 1$ and and $q_1 = q_2 = q/2$. The condition number of the generated optimal solutions was $\kappa_{0.1}(\tilde{x}) \approx 100$ for all six instances.

The results of this experiments are presented in Figure 8.3. For the instances with $\kappa(A^\intercal A) \geq 10^{10}$, FISTA, PCDM and PSSgb were terminated after $300,000$, $1,000,000$ and $80,000$ iterations, respectively, which corresponded to more than 27 hours of wall-clock time.

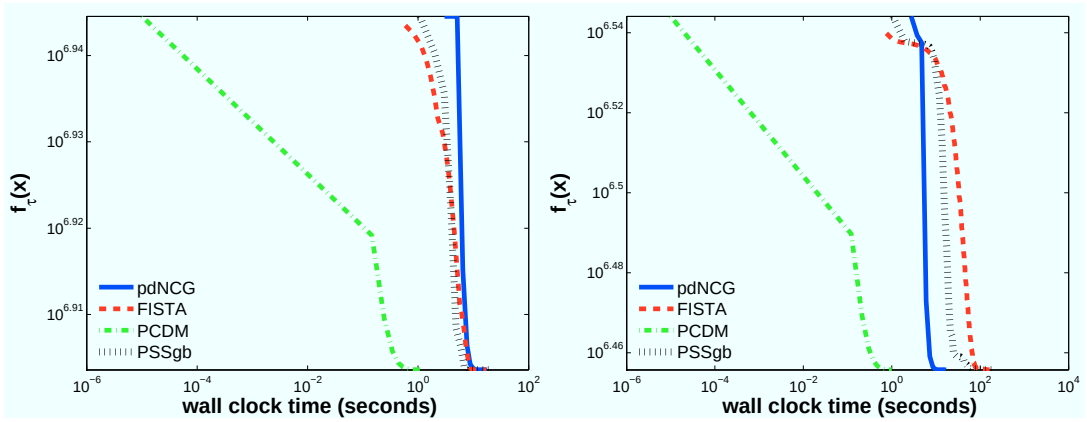
8.1.6 Increasing dimensions

In this experiment we present the performance of pdNCG, FISTA, PCDM and PSSgb as the number of variables n increases. We generate four groups of instances where the number of variables n takes values 2^{20}, 2^{22}, 2^{24} and 2^{26}, respectively. For each group we perform two experiments, one well-conditioned and one ill-conditioned. The structure of the singular value decompositions of matrices A is $A = \Sigma G^\intercal$, see Subsection 7.4 for details. For the well-conditioned problems the singular values in matrix Σ are chosen uniformly at random in the interval $[0, 10]$ and then are shifted by 10^{-1}, which resulted in $\kappa(A^\intercal A) \approx 10^4$. For the ill-conditioned problems the singular values in matrix Σ are chosen uniformly at random in the interval $[0, 10^4]$ and then they are shifted by 10^{-1}, which resulted in $\kappa(A^\intercal A) \approx 10^{10}$. The rotation angle θ of matrix G in Subsection 7.4.1 is set to $2\pi/10^3$ radians. Moreover, matrices A have $m = 2n$ rows and rank n. The optimal solutions $\tilde{x}$ have $q = n/2^7$ non-zero components for each generated instance. For the construction of the optimal solutions $\tilde{x}$ we used Procedure 7.7 OsGen3 with $\gamma = 1$ and $q_1 = q_2 = q/2$, which resulted in $\kappa_{0.1}(\tilde{x}) \approx 100$ for all experiments.

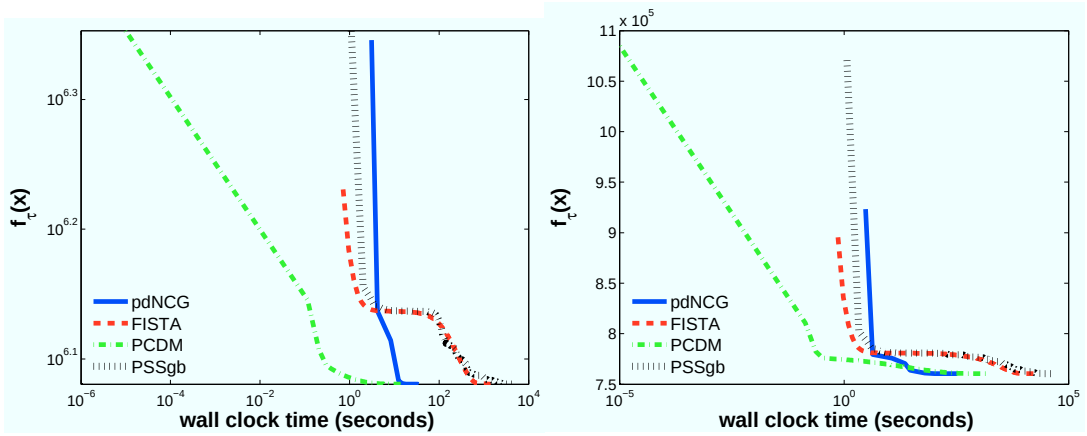
The results of this experiment are presented in Figure 8.4. Notice that all methods have a linear-like scaling with respect to the size of the problem.

8.1.7 Increasing density of matrix A

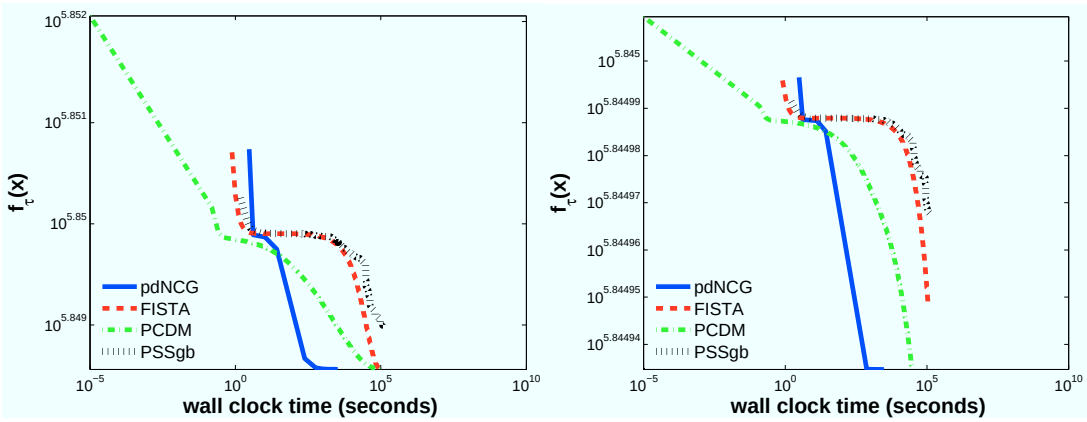
In this experiment we demonstrate the performance of pdNCG, FISTA, PCDM and PSSgb as the density of matrix A increases. We generate four instances $(A, \tilde{x})$, where the four matrices A have the sparsity pattern that is shown in Figure 7.1 and they are generated as described in Subsection 7.4.2. In particular, for the first experiment we generate matrix $A = \Sigma G^\intercal$, where Σ is the matrix of singular values and G is the matrix of right singular vectors, see Subsection 7.4 for details. For the second experiment we generate matrix $A = \Sigma(G_2 G)^\intercal$, where $G_2 G$ is the matrix of right singular vectors and G_2 has been defined in Subsection 7.4.2. Finally, for the third and fourth experiments we have $A = \Sigma(G G_2 G)^\intercal$ and $A = \Sigma(G_2 G G_2 G)^\intercal$, respectively. For each experiment the singular values of matrix



(a) $\kappa(A^\intercal A) = 10^2$ (b) $\kappa(A^\intercal A) = 10^4$

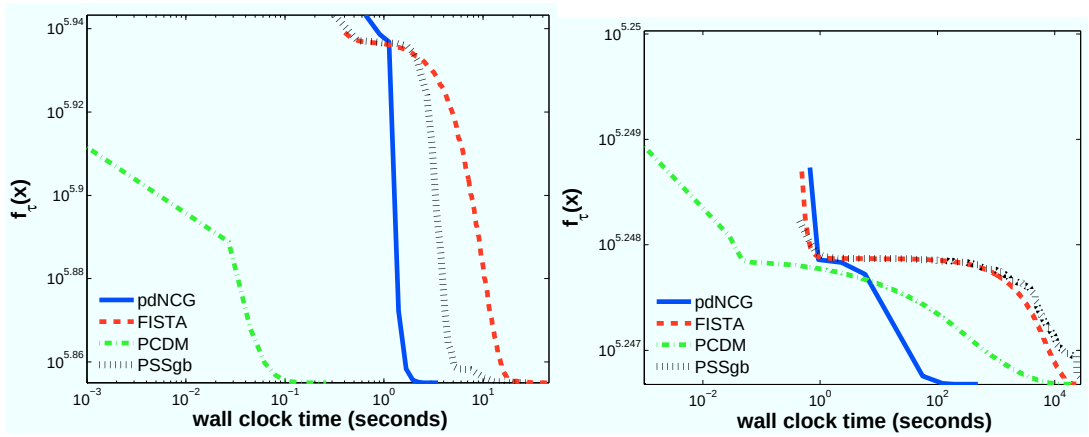


(c) $\kappa(A^\intercal A) = 10^6$ (d) $\kappa(A^\intercal A) = 10^8$

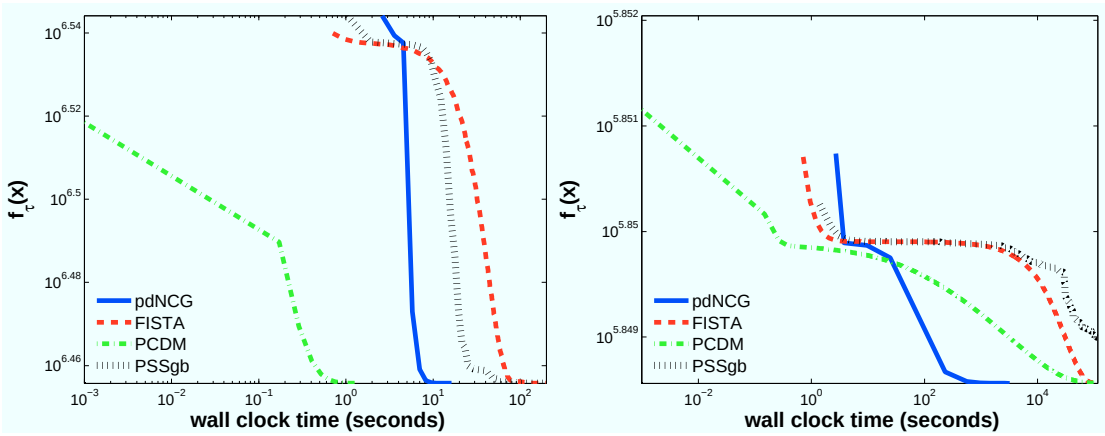


(e) $\kappa(A^\intercal A) = 10^{10}$ (f) $\kappa(A^\intercal A) = 10^{12}$

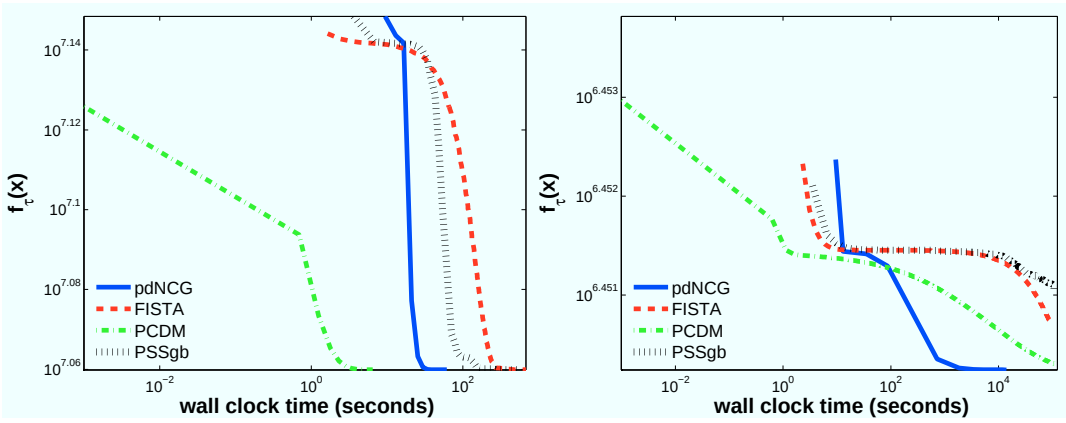
Figure 8.3: Performance of pdNCG, FISTA, PCDM and PSSgb on synthetic ℓ_1-LS problems for increasing condition number of matrix $A^\intercal A$. The optimal solutions have been generated by using Procedure 7.7 OsGen3 with $\gamma = 1$ and $q_1 = q_2 = q/2$. The axis are in log-scale. Notice that for condition number $\kappa(A^\intercal A) \geq 10^{10}$, FISTA, PCDM and PSSgb were terminated after $300,000$, $1,000,000$ and $80,000$ iterations, respectively, which corresponded to more than 27 hours of wall-clock time



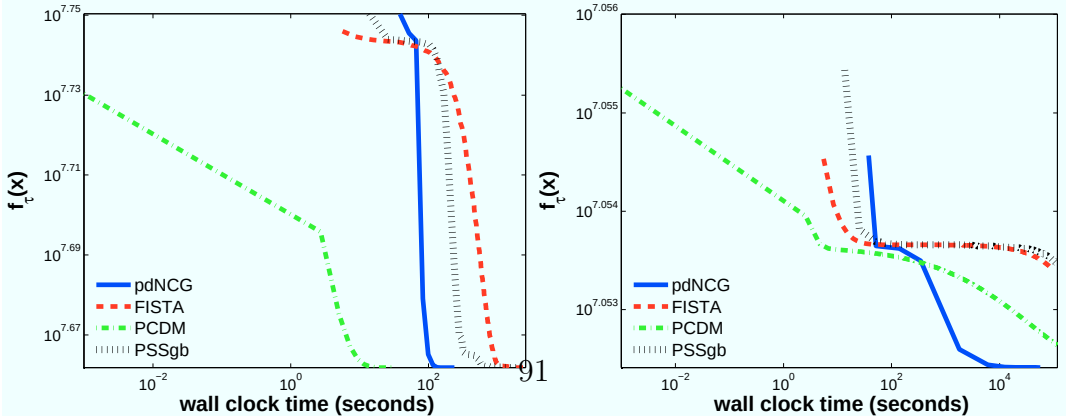
(a) $n = 2^{20}$, $\kappa(A^\intercal A) \approx 10^4$ (b) $n = 2^{20}$, $\kappa(A^\intercal A) \approx 10^{10}$



(c) $n = 2^{22}$, $\kappa(A^\intercal A) \approx 10^4$ (d) $n = 2^{22}$, $\kappa(A^\intercal A) \approx 10^{10}$



(e) $n = 2^{24}$, $\kappa(A^\intercal A) \approx 10^4$ (f) $n = 2^{24}$, $\kappa(A^\intercal A) \approx 10^{10}$

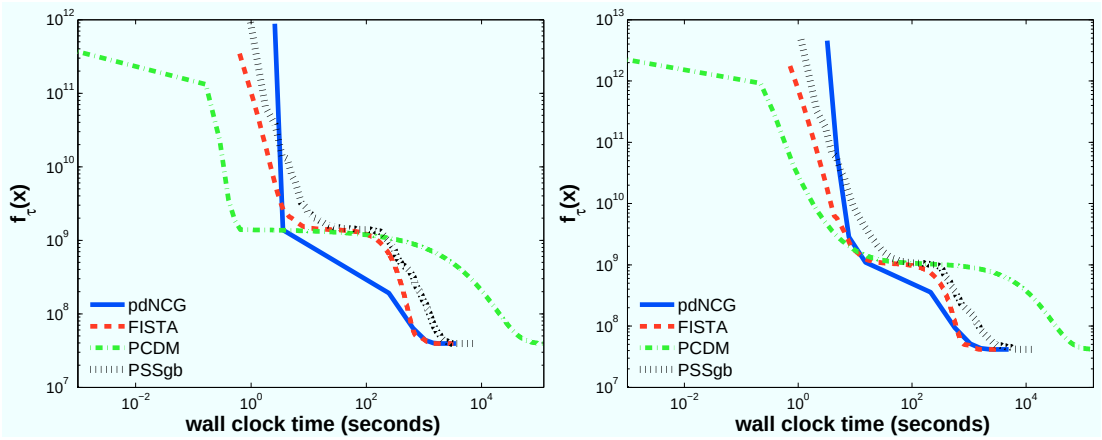


(g) $n = 2^{26}$, $\kappa(A^\intercal A) \approx 10^4$ (h) $n = 2^{26}$, $\kappa(A^\intercal A) \approx 10^{10}$

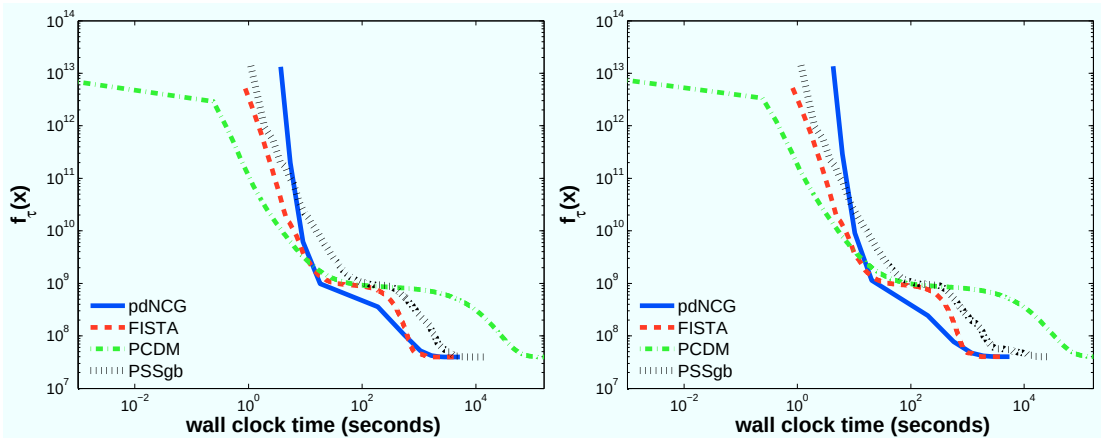
Figure 8.4: Performance of pdNCG, FISTA, PCDM and PSSgb on synthetic ℓ_1-LS problems for increasing number of variables n. The axis are in log-scale

A are chosen uniformly at random in the interval $[0, 10]$ and then are shifted by 10^{-1}, which resulted in $\kappa(A^\intercal A) \approx 10^6$. The rotation angle θ of matrices G and G_2 is set to $2\pi/10$ radians. Matrices A have $m = 2n$ rows, rank n and $n = 2^{22}$. The optimal solutions $\tilde{x}$ have $q = n/2^7$ non-zero components for each experiment. Moreover, Procedure OsGen3 is used with $\gamma = 100$ and $q_1 = q_2 = q/2$ for the construction of $\tilde{x}$ for each experiment, which resulted in $\kappa_{0.1}(\tilde{x}) \approx 10$ for all four instances.

The results of this experiment are presented in Figure 8.5. Observe, that all methods had a robust performance with respect to the density of matrix A.



(a) $nnz(A) = 2^{23}$ (b) $nnz(A) \approx 2^{24}$



(c) $nnz(A) \approx 2^{24} + 2^{23}$ (d) $nnz(A) \approx 2^{25}$

Figure 8.5: Performance of pdNCG, FISTA, PCDM and PSSgb on synthetic ℓ_1-LS problems for increasing number of non-zeros of matrix A. The axis are in log-scale

8.1.8 Performance of pdNCG on huge scale problems

We now present the performance of pdNCG on synthetic huge scale (up to a trillion variables) ℓ_1-LS problems as the number of variables and the number of processors increase. Unfortunately, implementations for FISTA, PCDM and

n	Processors	Memory (terabytes)	Time (seconds)
2^{30}	64	0.192	1,923
2^{32}	256	0.768	1,968
2^{34}	1024	3.072	1,986
2^{36}	4096	12.288	1,970
2^{38}	16384	49.152	1,990
2^{40}	65536	196.608	2,006

Table 8.1: Performance of pdNCG for synthetic huge scale ℓ_1-LS problems. All problems have been solved to relative error of order 10^{-4} of the obtained solution

PSSgb for computer clusters are not available and we were not able to test the performance of these solvers for the experiments of this subsection.

We generate six instances $(A, \tilde{x})$, where the number of variables n takes values 2^{30}, 2^{32}, 2^{34}, 2^{36}, 2^{38} and 2^{40}. Matrix A has $m = 2n$ rows and rank n. The singular values σ_i for $i = 1, 2, \cdots, n$ of A are set to 10^{-1} for odd i's and 10^2 for even i's. The rotation angle θ of matrix G in Subsection 7.4.1 is set to $2\pi/3$ radians. The optimal solution $\tilde{x}$ has approximately $s = n/2^{10}$ non-zero components. In order to simplify the practical generation of this problem the optimal solution $\tilde{x}$ is set to have $s/2$ components equal to -10^4 and the rest of non-zero components are set equal to 10^{-1}.

Details of these experiments are given in Table 8.1. Observe the nearly linear scaling of pdNCG with respect to the number of variables n and the number of processors.

8.2 Compressed sensing

In this section we demonstrate the efficiency of mfIPM, which was presented in Chapter 6, against state-of-the-art methods for CS problems. In what follows we briefly discuss existing methods, we describe the setting of the experiments and finally present the numerical results. All experiments that are demonstrated in this section have appeared in [54]. The experiments can be reproduced by downloading the software from `http://www.maths.ed.ac.uk/ERGO/mfipmcs/`.

All solvers used in this section, including mfIPM are MATLAB implementations. All experiments were performed using MATLAB version R2012b (8.0.0.783) 64-bit on a Dual 8 Core Intel Xeon (Sandybridge) running Redhat Enterprise Linux in 64-bit mode. Finally, the RICE Wavelet toolbox, included in Sparco test suite, was compiled using gcc compiler version 4.4.6 20120305 (Red Hat 4.4.6-4).

Before proceeding to the following subsections it would be convenient for the reader to be familiarized with symbols and abbreviations in Table 8.2, which are used in the subsequent figures and comparison tables.

m, n, q	number of rows and columns of the matrix A and the number of non-zero elements in the optimal sparsest signal representation
$\tilde{x}$	optimal sparse representation
$x_{\mathcal{N}}$	Given a vector $x \in \mathbb{R}^n$ we define $x_{\mathcal{N}_i} = x_i$ if $i \in \mathcal{N}$, otherwise $x_{\mathcal{N}_i} = 0$, where $\mathcal{N} := \{i = 1, 2, \ldots, n \mid \tilde{x}_i \neq 0\}$
$r.e(x_{\mathcal{N}})$	relative error $\|x_{\mathcal{N}} - \tilde{x}\|_2 / \|\tilde{x}\|_2$
$res(x_{\mathcal{N}})$	residual $\|Ax_{\mathcal{N}} - b\|_2$, where b should be replaced with $\hat{b}$ in case of noiseless signals
$n1d(x_{\mathcal{N}})$	distance from the optimal value of ℓ_1-norm, $\|\|x_{\mathcal{N}}\|_1 - \|\tilde{x}\|_1\|$
$obj(x_{\mathcal{N}})$	objective value of problem (6.1), $\tau\|x_{\mathcal{N}}\|_1 + \|Ax_{\mathcal{N}} - b\|_2^2$
$nMat$	total number of matrix vector products Ax and $A^\mathsf{T}y$

Table 8.2: Symbols and abbreviations used in tables and figures in Section 8.2

8.2.1 Existing algorithms

There have been various first-order methods developed for the solution of (6.1) and (2.7). Let us mention the ones known to be the most efficient, based on our experience on how frequently they are being used in the literature.

- Fixed Point Continuation Active Set (FPC_AS) [115] solves problem (6.1). FPC_AS is a two stage algorithm. At the first stage a shrinkage scheme is employed which aims to spot quickly the non-zero components of the sparse representation. Then, the second stage is enabled to solve a smooth version of (6.1) limited to the indexes of non-zero components found by the first stage of the algorithm. We use the FPC_AS CG version of FPC_AS algorithm, where "CG" stands for the conjugate gradient method. The FPC_AS CG has been shown in [115] to be considerably faster than other versions of FPC and FPC_AS software packages. The FPC_AS CG solves problem (6.1). The code of FPC_AS CG package can be found at `http://www.caam.rice.edu/~optimization/L1/FPC_AS/`.

- Spectral Projected Gradient for ℓ_1-regularized least-squares (SPGL1) [12] solves any of the problems (2.5), (2.7a) and (2.7b). The SPGL1 is a spectral projection gradient algorithm which iteratively solves (2.7a) for some values of ϵ_1, each approximate solution of (2.7a) is used to build a root-finding problem, which is equivalent to (2.7b), and is solved by employing a Newton method. We use the SPGL1_bp version of SPGL1 software package for noiseless signals and the SPGL1_bpdn version for noisy signals, where "bp" stands for basis pursuit and "bpdn" for basis pursuit denoising, respectively. The SPGL1_bp solves problem (2.5) and the "bpdn" version solves problem (2.7b). The code of SPGL1 package can be found at `http://www.cs.ubc.ca/labs/scl/spgl1`.

Independently there have been several attempts to design suitable IPM implementations. The most efficient among them which can also handle large-scale CS problems are listed below.

- $\ell_1\_\ell_s$ algorithm [74] solves a constrained smooth reformulation of problem (6.1), which allows a straightforward preconditioning of the Newton equation system that is solved with a conjugate gradient method. The $\ell_1\_\ell_s$ solver can be found at `http://www.stanford.edu/~boyd/l1_ls/`.

- Primal-Dual interior-point method for Convex Objectives (PDCO) algorithm [100] solves regularized constrained smooth reformulations of problems (2.5) and (6.1). The Newton equation system is solved by applying a Least-Squares QR factorization (LSQR) method. The PDCO solver is used through the file SolveFasBP.m of SparseLab software package. The PDCO solver can be found at `http://www.stanford.edu/group/SOL/software/pdco.html` and the SparseLab software package at `http://sparselab.stanford.edu/`.

Both $\ell_{\mathbf{1}}\_\ell_{\mathbf{s}}$ and PDCO have been demonstrated in [50, 74] to be robust in comparison with other IPM implementations.

8.2.2 Benchmarks

In order to have a base of comparison we choose to show the efficiency of mfIPM on already existing benchmarks, which have been used by several researchers including [12, 115]. Experiments are performed on 18 real valued sparse reconstruction problems from the Sparco collection [13], see Table 8.3. In total, the Sparco collection consists of 26 problems, out of which 6 are complex valued and 20 real valued. For the experiments in this section, the complex valued problems with IDs 1, 4, 8, 501 to 503, are ignored since mfIPM handles only real data. Moreover, 2 out of the 20 real valued problems, with IDs 703 and 901, are also ignored because of their difficulty to be generated on any machine in a standalone approach, since they require external packages such as CurveLab [20] and FFTW [56]. For problems in Table 8.3 with IDs 401 to 403, 601 to 603, 701 and 702, the optimal representation $\tilde{x}$ is not given by Sparco toolbox. Therefore, the SPGL1_bp solver is used to obtain $\tilde{x}$ with required high accuracy. In particular to obtain $\tilde{x}$, the parameters of SPGL1_bp are set to:

$$\text{bpTol} = 1.0\text{e-}15, \quad \text{optTol} = 1.0\text{e-}15, \quad \text{decTol} = 20\,000, \tag{8.1}$$

where bpTol controls the tolerance for identifying a basis pursuit solution, optTol controls the optimality tolerance and decTol controls the frequency of Newton updates. Let $\bar{x}$ be the solution obtained from SPGL1_bp. Some of the components of $\bar{x}$ might be nearly but not exactly zero. Hence we consider as non-zero components the ones in the set $nnz(\bar{x}) := \{p \in \{1, 2, \ldots, n\} \mid \sum_{i=1}^{p} |\hat{x}_i| \leq 0.999\|\bar{x}\|_1\}$, where $\hat{x}$ is the vector $\bar{x}$ sorted in decreasing order of absolute values of its components. Then we set $\tilde{x}_j = \bar{x}_j$ if $j \in \{i \in \{1, 2, \ldots, n\} \mid i \in nnz(\bar{x})\}$ otherwise $\tilde{x}_j = 0$.

Problem	ID	m, n	Operator	$\|\tilde{x}\|_1$
[43, 44] blocksig	2	1 024, 1 024	wavelet	4.5e+02
cosspike	3	1 024, 2 048	DCT	2.2e+02
gcosspike	5	300, 2 048	Gausian ens., DCT	1.8e+02
[25] p3poly	6	600, 2 048	Gausian ens., wavelet	1.7e+03
[26] sgnspike	7	600, 2 560	Gaussian ens.	2.0e+01
[43, 44] blkheavi	9	128, 128	heaviside	4.1e+01
[43, 44] blknheavi	10	1 024, 1 024	normal. heaviside	9.8e+02
gausspike	11	256, 1 024	Gausian ens.	2.4e+01
srcsep1	401	29 166, 57 344	windowed DCT	1.0e+03
srcsep2	402	29 166, 86 016	windowed DCT	7.7e+03
srcsep3	403	196 608, 196 608	blurring, wavelet	1.0e+03
[105] soccer1	601	3 200, 4 096	binary, wavelet	4.2e+02
[105] soccer2	602	3 200, 4 096	binary, Haar wavelet	7.4e+02
[50] yinyang	603	1 024, 4 096	wavelet	2.6e+02
[50] blurrycam	701	65 536, 65 536	blurring, wavelet	1.0e+04
[50] blurspike	702	16 384, 16 384	blurring	3.4e+02
[68] jitter	902	200, 1 000	DCT	1.7e+00
[46] spiketrn	903	1 024, 1 024	1D convolution	1.3e+01

Table 8.3: 18 out of 20 real valued problems of Sparco collection

Noise is introduced to the noiseless measurements $\hat{b}$ using the following command in MATLAB:

$$b = \texttt{awgn}(\hat{b}, \text{SNR}, \text{'measured'}). \tag{8.2}$$

The function `awgn` is a MATLAB function from Communications Systems Toolbox, which adds white Gaussian noise to signal $\hat{b}$. The Signal to Noise Ratio (SNR) is measured in decibel (dB). The 'measured' option specifies that the power of the signal is calculated first before the addition of the noise.

8.2.3 Equivalent problems

It has being stated in Section 2.2 that problems (2.7b) and (6.1) are equivalent given particular parameters ϵ_2 and τ. The tested solvers implement problem (2.7b), i.e., SPGL1_bpdn, or problem (6.1), i.e., mfIPM, FPC_AS CG $\ell_1\_\ell_s$ and PDCO. In order to perform a fair comparison among these solvers it has to be made certain that all codes solve equivalent problems. Otherwise, different optimal solutions will be obtained, therefore, a straightforward and clear comparison would be impossible. Unfortunately, exact values of ϵ_2 and τ which make problems (2.7b) and (6.1) equivalent are not known a priori, except for the case of orthogonal matrix A. However, given ϵ_2 an approximate τ can be computed such that an approximate equivalence holds.

According to [12] given ϵ_2 the parameter τ which makes problems (2.7b) and (6.1) equivalent, is the optimal Lagrange multiplier of the dual problem of (2.7b). Since, SPGL1_bpdn outputs both the primal iterates and the optimal Lagrange multiplier of problem (2.7b), it can be used to approximately find τ with high accuracy. Having such a parameter τ the solvers mfIPM, FPC_AS CG $\ell_1\_\ell_s$, PDCO and SPGL1_bpdn can be legitimately compared.

Moreover, in order to be able to compare the quality of the reconstructed representations for each solver when solving equivalent problems, the optimal sparsest representation for a particular level of noise needs to be known in advance. This is definitely the case when noise is added manually by the user to a noiseless signal $\hat{b}$ using (8.2). Due to manual corruption of signal $\hat{b}$, the energy of the added noise $\epsilon_2 = \|e\|_2$ is known in advance. Hence, solving problem (2.7b) will give the optimal sparsest representation for this particular level of noise, ϵ_2. This solution is obtained by first calling SPGL1_bpdn solver to solve problem (2.7b) by setting $\epsilon_2 = \|e\|_2$ with required high accuracy, see (8.1). During this process the approximate τ which makes problems (2.7b) and (6.1) equivalent is obtained from SPGL1_bpdn as has been described before. Hence, it is concluded that approximate τ and optimal sparse representations can be calculated such that a fair comparison can be conducted.

Finally, for noiseless signals $\hat{b}$, the problem is easier. Problems (2.5) and (6.1) are almost equivalent for sufficiently small τ, i.e., 1.0e-10. However, such a small τ can make the ℓ_1-norm in (6.1) numerically negligible and numerical difficulties might occur for the methods, see Figure 6.2 in [12] for numerical examples. For the former reason, if such a case is observed, parameter τ is set to the smallest possible value such that all methods that we use can solve the problem; their values are given in Table 8.5.

8.2.4 Termination criteria and parameter tuning

Termination of the compared solvers is forced when a solution of similar quality to the one of mfIPM is obtained. In order to do so, the termination criteria of the compared solvers are changed. In particular, SPGL1 solver is terminated when the following criteria are satisfied:

$$n1d(x_{\mathcal{N}}^k) \leq n1d(x_{\mathcal{N}}^m), \quad r.e(x_{\mathcal{N}}^k) \leq r.e(x_{\mathcal{N}}^m), \quad res = (x_{\mathcal{N}}^k) \leq res(x_{\mathcal{N}}^m),$$

where $x_{\mathcal{N}}^k$ is the projected representation at the k^{th} iteration of SPGL1 and $x_{\mathcal{N}}^m$ is the projected representation obtained by mfIPM. Solvers FPC_AS, $\ell_1\_\ell_s$ and PDCO are terminated when the following conditions are satisfied:

$$obj(x_{\mathcal{N}}^k) \leq obj(x_{\mathcal{N}}^m), \quad r.e(x_{\mathcal{N}}^k) \leq r.e(x_{\mathcal{N}}^m).$$

Using these criteria for the compared solvers it is made certain that the reconstructed representations have approximately the same ℓ_1-norm, ℓ_2-norm of residual $Ax_{\mathcal{N}} - b$ and number of non zero elements in $x_{\mathcal{N}}$. The differentiation of the termination criteria for solver SPGL1 is done because SPGL1 solves problem (2.7b), unlike all other codes which solve problem (6.1). Hence, it is more natural and fair for SPGL1 to be compared with other solvers using termination criteria in SPGL1 way.

Occasionally, certain solvers required too many matrix-vector products without achieving a solution of similar quality to the one delivered by mfIPM. In this case the solvers were terminated when $nMat > 40\,000$.

Regarding the parameter tuning of the compared solvers, all their parameters are set to their default values. For mfIPM the following parameters need to be set.

- tol: Relative duality gap of primal-dual pair (6.4) and (6.9). For noisy problems, this parameter varies between 1.0e-6 and 1.0e-10. For noiseless problems it varies between 1.0e-7 and 1.0e-14.

- maxiters: Maximum number of iterations. For all problems this parameter is set to 100.

- tolpcg: Tolerance of preconditioned CG method. For noisy problems this parameter varies between 1.0e-1 and 1.0e-2 and for noiseless ones it varies between 1.0e-1 and 1.0e-6.

- mxiterpcg: Maximum number of iterations of PCG method. For all problems this parameter is set to 200.

Since a large number of experiments has been performed, the exact parameter tuning of mfIPM is not given here. The exact tuning can be found in the MATLAB scripts which reproduce the results in this section, see `http://www.maths.ed.ac.uk/ERGO/mfipmcs/`. Based on our experience tuning of mfIPM was important for experiments that the sensing matrix A was deviating from satisfying the RIP with small RIP parameter. This is because such problems might be ill-conditioned

locally to the optimal solution hence tuning of the method is required in order to solve the problems efficiently.

Finally, the parameters ϵ_2 and τ in problems (2.7b) and (6.1), respectively, for noisy problems are set as described in Subsection 8.2.3 for $\epsilon_2 = \|e\|_2$. For noiseless problems τ is set to arbitrarily small values given in Table 8.5.

8.2.5 Comparison

In this section we present computational results obtained for the Sparco collection problems discussed in the Benchmarks Section 8.2.2. Both noisy and noiseless measurements are considered. Noise is added to measurements using (8.2) by fixing the SNR = 60 dB. A comparison among the previously mentioned solvers is made in terms of the quality of reconstruction and computational effort. The results of experiments are shown in Table 8.4. The first column in Table 8.4 shows the IDs of the Sparco problems. For each ID the first and second sub-rows give results for noisy and noiseless measurements, respectively. The second column reports the ℓ_1-norm of the projected reconstructed representation for mfIPM. The third column shows the relative error $r.e$, see Table 8.2, of the projected reconstructed representation that was achieved by mfIPM. The forth column shows the ℓ_2-norm of the residual, denoted by res in Table 8.2, for mfIPM. The rest of the table shows the number of matrix-vector products, *nMat*, that were needed by each solver to reconstruct a solution of similar quality to the one of mfIPM. In cases when the number of matrix-vector products required by a solver exceeded 40 000, the solver was terminated with a failure status. To be precise, it is a failure to converge to a solution similar to the one obtained by mfIPM. Problems for which mfIPM converged with the lowest number of matrix-vector products among all solvers compared are denoted in bold. In Table 8.5 are shown the regularization parameters τ for noiseless signals that were used for solvers mfIPM, FPC_AS, $\ell_1\_\ell_s$ and PDCO. Finally, for noiseless signals the version SPGL1_bp of SPGL1 solver is called.

One can observe in Table 8.4 that mfIPM was the fastest solver in 11 out of 36 noisy and noiseless problems, while it was the second fastest for another 14 problems, denoted by italic font. It is important to mention that the performance of the compared solvers depends crucially on the condition number of matrices built of subsets of columns of matrix A with cardinality q (the number of non-zeros in the optimal solution), i.e., full-rank sub-matrices of A. Unfortunately, it is a computational demanding task to check the condition number of every full-rank sub-matrix for the problems shown in Table 8.3. Nevertheless, by experimenting with a few sub-matrices one can get a picture of how well-conditioned sub-matrices of A might be.

Based on the previous criterion we observed that on problems that mfIPM was first or second, matrix A had relatively ill-conditioned sub-matrices, at least for the ones that we experimented with. The previous implies that the proposed preconditioner was not as efficient as predicted in Section 6.4, since matrix A does not satisfy the RIP with a small RIP parameter. However, the ill-conditioning also adversely affected the performance of SPGL1 and FPC_AS, as shown in Table 8.4. On the contrary, on problems that matrix A seemed to have well-conditioned sub-

				mfIPM	ℓ_1-ℓ_s	PDCO	FPC_AS	SPGL1
ID	$\|x\|_1$	$r.e$	res	$nMat$				
2	4.5e+02	5.3e-04	8.2e-02	*61*	726	6 611	9	40 000
	4.5e+02	1.0e-11	8.4e-10	*65*	644	40 011	40 002	21
3	2.2e+02	9.9e-04	1.3e-01	195	446	5 115	119	70
	2.2e+02	1.8e-08	1.8e-06	387	1 540	40 005	192	146
5	1.8e+02	3.0e-03	2.3e-01	1 367	5 042	28 369	630	510
	1.8e+02	2.4e-05	1.8e-03	*6 239*	20 758	41 479	636	40 000
6	1.7e+03	2.2e-02	1.7e+01	*2 507*	2 838	42 125	720	40 000
	1.7e+03	4.6e-02	3.6e+01	*7 193*	40 011	18 685	573	40 000
7	2.0e+01	5.4e-04	2.3e-03	165	452	955	78	63
	2.0e+01	5.6e-07	1.1e-06	259	952	709	78	87
9	4.1e+01	1.0e-03	1.7e-01	**377**	574	579	446	8 855
	4.1e+01	5.2e-12	1.6e-10	**661**	3 860	7 113	40 002	40 000
10	9.0e+02	9.3e-02	3.3e+00	*2 431*	11 421	1 043	40 001	40 000
	9.8e+02	1.0e-09	8.9e-08	**4 519**	8 192	42 647	40 001	40 000
11	2.4e+01	1.4e-03	1.3e-01	767	2 186	3 291	217	143
	2.4e+01	6.8e-05	5.2e-03	1 241	4 542	4 299	219	189
401	1.0e+03	8.9e-02	1.2e-01	*2 747*	42 622	61 327	40 076	882
	1.0e+03	7.7e-02	9.7e-02	*3 193*	43 512	48 511	40 076	814
402	1.0e+03	1.0e-01	1.9e-01	*4 393*	46 458	44 169	40 078	517
	1.0e+03	8.1e-02	2.0e-01	*4 991*	49 122	43 845	40 078	617
403	7.6e+03	1.2e-02	7.1e-01	2 841	6 136	40 495	2 305	699
	7.7e+03	4.1e-03	9.2e-02	*6 031*	43 278	69 913	40 046	932
601	3.3e+02	6.1e-02	5.7e+01	**1 179**	14 684	40 153	40 080	40 000
	4.0e+02	3.9e-02	4.8e+00	*4 409*	9 664	43 369	40 076	1 116
602	5.9e+02	1.0e-01	4.8e+01	*1 199*	17 097	40 631	40 023	898
	6.4e+02	1.1e-01	3.2e+00	**4 669**	22 392	42 139	40 043	40 000
603	2.6e+02	4.1e-03	4.2e-02	*1 777*	40 693	50 369	40 002	443
	2.5e+02	4.6e-02	5.9e-01	3 545	2 350	40 181	338	95
701	9.1e+03	4.6e-02	1.5e-01	**1 217**	33 160	91 147	40 044	1 658
	1.0e+04	2.4e-07	4.1e-03	**1 907**	4 722	49 093	40 001	40000
702	3.4e+02	4.8e-03	3.4e-03	**711**	1 600	5 525	40 001	40 000
	3.4e+02	6.4e-08	2.4e-03	**1 913**	3 030	49 009	40 037	12 388
902	1.7e+00	5.3e-04	5.2e-04	143	498	237	40	49
	1.7e+00	2.0e-06	9.6e-07	239	675	279	42	59
903	1.3e+01	2.4e-03	1.4e-01	**3 105**	8 466	4 775	8 237	6 735
	1.3e+01	3.5e-06	1.9e-04	**4 163**	25 128	30 979	33 529	40 000

Table 8.4: Results for noisy and noiseless Sparco problems

τ	Problems
1.0e-10	2, 9, 10, 701, 702
1.0e-08	401, 402, 603
1.0e-07	3, 7, 902
1.0e-05	903
1.0e-04	5, 403, 601, 602
1.0e-03	6
1.0e-02	11

Table 8.5: Regularization parameters τ for problem (6.1) and noiseless measurements $\hat{b}$ for the experiments performed in Table 8.4

matrices, the preconditioner was very efficient, which resulted in mfIPM being very fast. However, SPGL1 and FPC_AS were faster. For example, see problems with IDs 2, 3, 7 and 902.

8.2.6 Robustness to Noise

In this subsection we compare mfIPM with SPGL1, FPC_AS CG, $\ell_1\_\ell_s$, in terms of their reconstruction capabilities for different levels of noise. The results collected in Table 8.4 and analysed in Section 8.2.5 reveal that PDCO and $\ell_1\_\ell_s$ demonstrate comparable efficiency but the latter is usually faster. Therefore, solver PDCO will not be used in our further experiment.

For this experiment, the level of noise is varied from SNR = 10 dB to SNR = 120 dB with a step of 10 dB. The quality of reconstruction for all solvers is measured using the amplitude criterion [106]:

$$amp(x_{\mathcal{N}}) = \frac{\sqrt{\frac{1}{n}\|x_{\mathcal{N}} - \tilde{x}\|_2^2}}{\sqrt{\frac{1}{m}\|e\|_2^2}}.$$

The main purpose of using the *amp* criterion, instead of $r.e$, is that the former amplifies the $r.e$, the nominator of *amp*, as $\|e\|_2 \to 0$. Hence, less accurate representations will be emphasized.

As in Section 8.2.5 when the optimal representation $\tilde{x}$ of problem (2.5) is unknown it is calculated approximately using solver SPGL1_bp with required high accuracy (8.1). In order to have a fair comparison it is necessary to know at least approximately the parameter τ which makes problems (2.7b) and (6.1) equivalent and moreover, the optimal sparse representation of problem (2.7b) for $\epsilon_2 = \|e\|_2$. The former issues are solved as described in Subsection 8.2.3.

To compare the solvers the following criterion is defined:

$$rampd(x_{\mathcal{N}}) = \frac{\max(amp(x^*_{\mathcal{N}}) - amp(x^s_{\mathcal{N}}), 0)}{amp(x^s_{\mathcal{N}})}, \tag{8.3}$$

where *rampd* stands for relative amplitude difference, $x^*_{\mathcal{N}}$ is the reconstructed projected representation by solvers mfIPM, FPC_AS CG, $\ell_1\_\ell_s$, and $x^s_{\mathcal{N}}$ is the reconstructed projected representation of solver SPGL1_bpdn. Notice that if *rampd* equals zero, then the representation $x^*_{\mathcal{N}}$ is of better quality than $x^s_{\mathcal{N}}$, otherwise the inverse is true.

In Table 8.6 is shown the average value of *rampd* over all SNR for each solver. The first column of Table 8.6 reports the ID of every tested Sparco problem. From the second to the forth column the average *rampd* over all SNR for each solver is shown. The last three columns report the average *rampd* for SNRs from 10 dB to 60 dB for each solver. Notice in Table 8.6 that mfIPM for problems with IDs 2 to 11 and 701 to 903 was consistently recovering a high quality solution. For problems with IDs 401 to 603 for SNR $>$ 60 dB the solvers mfIPM, FPC_AS CG and $\ell_1\_\ell_s$, were unable to reconstruct an adequate representation and this is in contrast to SPGL1. A similar observation has been reported in [12]. In this work the authors mentioned that this issue of the solvers that failed might be due to very small regularisation parameter τ, obtained from SPGL1 solver as the energy of noise is decreased. In this case, the regularization effect of the ℓ_1-norm starts to be negligible and the solvers face considerable numerical difficulties. However, in our experiments we observed for these problems that not always the τ parameter was small and additionally, there were other problems were τ was even smaller but successful reconstruction was possible. Therefore, we conclude that this failure of the solvers mfIPM, FPC_AS CG and $\ell_1\_\ell_s$ might be problem dependent and not τ-dependent. The relative importance of the term $\tau\|\tilde{x}\|_1$ compared to $1/2\|A\tilde{x} - b\|_2^2$ might be an explanation for this phenomenon.

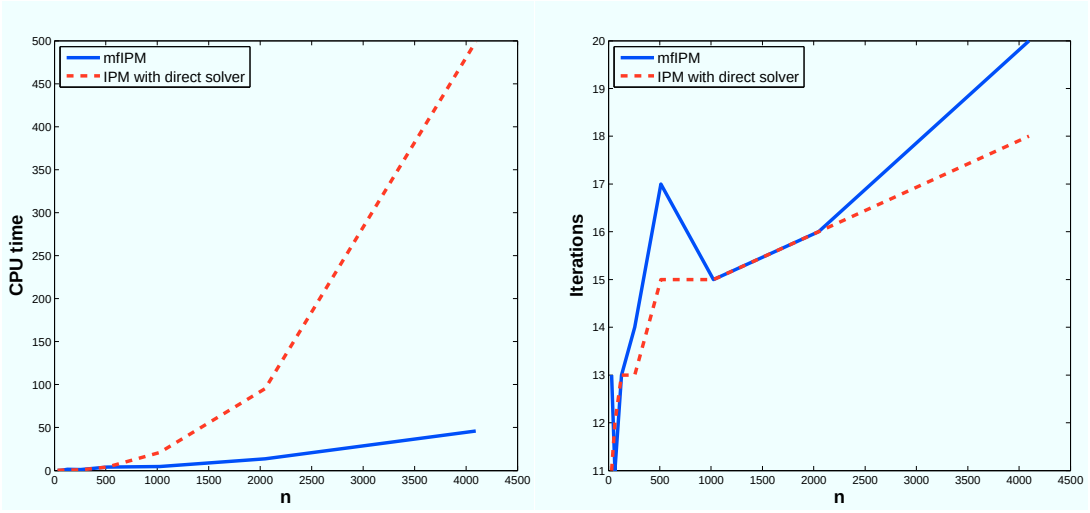
8.2.7 Preconditioned conjugate gradient method against direct linear solver

In this subsection we replace PCG in steps 8 and 10 of mfIPM in Algorithm 6.1 with a direct linear solver. Direct linear solvers tend to be efficient when the system to be solved is sufficiently sparse. However, for CS the systems (6.11) to be solved are completely dense due to density of matrix A. For this reason, large-scale problems cannot be stored in a moderate computer with 8 giga byte of random access memory. Even worse, matrix A might be an algorithmic operator, i.e., DCT, therefore, direct solvers cannot be employed. Hence, direct linear solvers for CS inside an IPM are only applicable when the measurement matrix A is explicitly available, i.e., Gaussian matrix, and only for small-scale problems, i.e., $n = 2^{12}$ or smaller. In addition to the former disadvantages of a direct solver for CS problems, its computational complexity for systems (6.11) will be of order $\mathcal{O}(n^3)$. This is a well known result, for completely dense linear systems. Therefore, it is expected that for very small instances the two approaches might require similar CPU time to converge, while as dimensions grow the CPU time of the IPM version with the direct linear solver will increase rapidly. Indeed, this is confirmed by Figure 8.6a.

Despite the higher computational effort required by direct solvers for CS problems, such an approach will produce exact Newton directions, hence, one would

	Avg. *rampd* for SNR from 10 dB to 120 dB			Avg. *rampd* for SNR from 10 dB to 60 dB		
ID	mfIPM	FPC_AS	ℓ_1_ℓ_s	mfIPM	FPC_AS	ℓ_1_ℓ_s
2	6.1e-09	2.5e-10	0.0e+00	2.8e-13	2.6e-13	0.0e+00
3	1.3e-04	8.7e-05	0.0e+00	4.0e-09	6.4e-14	0.0e+00
5	5.1e-06	5.0e-07	1.7e-01	7.1e-11	9.8e-07	0.0e+00
6	1.2e-07	2.4e-10	1.1e+00	2.5e-07	4.8e-10	0.0e+00
7	1.5e-02	5.7e-08	0.0e+00	4.3e-06	3.5e-15	0.0e+00
9	1.1e-08	1.1e-01	4.9e-06	2.1e-08	2.1e-01	1.0e-08
10	7.3e-04	1.6e-01	0.0e+00	1.5e-03	2.5e-01	0.0e+00
11	4.2e-05	1.8e-05	0.0e+00	1.4e-10	3.7e-12	0.0e+00
401	8.6e+00	1.2e+01	8.5e+00	1.8e-01	1.9e-01	1.7e-01
402	8.0e+00	2.2e+01	8.0e+00	2.0e-01	1.9e+01	2.1e-01
403	1.9e+00	3.8e+00	1.2e+00	8.1e-03	6.4e-12	1.4e-02
601	3.8e+05	4.0e+03	1.5e+01	4.8e-11	8.1e+03	1.9e-01
602	1.6e+00	2.9e+03	7.4e+00	1.1e-10	5.7e+03	1.3e-01
603	8.1e-01	6.7e+00	7.7e-01	2.2e-08	3.7e-01	1.8e-03
701	6.4e-08	1.9e+00	3.2e-03	0.0e+00	3.8e+00	6.4e-03
702	7.9e-02	2.5e+01	6.2e-03	0.0e+00	5.1e+01	1.3e-03
902	9.1e-02	9.7e-09	0.0e+00	2.1e-07	1.3e-08	0.0e+00
903	1.5e-04	3.8e+00	1.5e-04	1.0e-11	7.5e+00	0.0e+00

Table 8.6: Average quality reconstruction results over SNR from 10 dB to 120 dB for solvers mfIPM, FPC_AS and ℓ_1_ℓ_s on Sparco problems in Table 8.3



(a) Scaling of CPU time (b) Scaling of number of iterations

Figure 8.6: Scaling of CPU time and number of iterations as the size of problem n grows for mfIPM and an IPM in which the PCG is replaced with a direct solver

expect that IPM iterations will be the minimum possible. Surprisingly, in Figure 8.6b we show, that mfIPM with PCG requires as few iterations as its IPM version with a direct linear solver. Indeed, recent analysis of [61] indicates that allowing the use of inexact Newton directions in an IPM does not adversely affect the worst-case complexity result of this method.

In the experiments reported in Figures 8.6a and 8.6b matrix A is Gaussian, the sparsity pattern of the optimal representation $\tilde{x}$ is chosen at random, while the non-zero components follow a standard normal distribution. The noiseless measurements are produced by $\hat{b} = A\tilde{x}$. The size of problem n, is varied from 2^5 to 2^{12} with a step of times 2, the measurements m are varied from 2^3 to 2^{10} with a step of times 2 and the number of non-zeros in $\tilde{x}$, which is denoted by q, is set to $\lceil m/20 \rceil$. Finally, the τ parameter in problem (6.1) is set to τ =1.0e-3. To solve the linear systems we use the `mldivide` function of MATLAB, which in case of symmetric real matrices with positive diagonal, i.e., (6.11), performs Cholesky factorization. For details of the `mldivide` function we refer the reader to `http://www.mathworks.co.uk/help/matlab/math/systems-of-linear-equations.html`.

8.2.8 Average phase transition

Recently, it has been shown in [45] that for any problem instance (A, b), where A is Gaussian, there is a maximum ratio $\hat{\nu}_\rho = k/m$ given $\rho = m/n$ that below of it the problems (2.5) or (6.1) guarantee on average reconstruction of the optimal sparse representation. The latter has been introduced as the notion of average phase transition for Gaussian matrices. Moreover, it has been shown empirically that other measurement matrices such as partial Fourier, partial Hadamard, Bernoulli etc, have the same average phase transition properties. Ideally, an efficient ℓ_1-regularization solver should have empirical average phase transition at the same

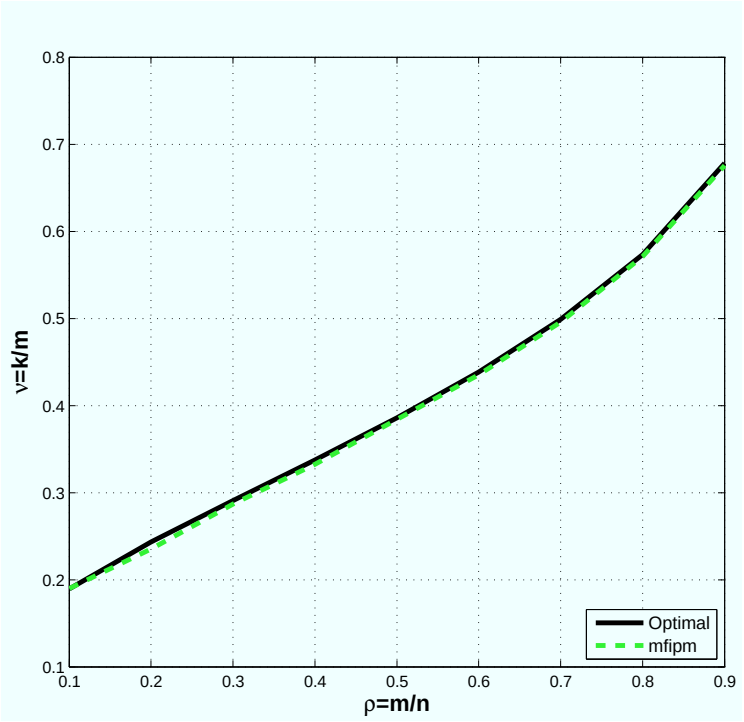


Figure 8.7: Empirical phase transition for mfIPM. The solid curve denotes the theoretically optimal phase transition. The dashed curve denotes the empirical phase transition for 50% success rate of mfIPM

level $\hat{\nu}_\rho$.

In this section we show that the empirical phase transition properties of mfIPM fit the average Gaussian phase transition properties by reproducing a similar experiment as that in Section 2 of [45]. Let us now explain the experiment. The parameter n is fixed to $n = 1000$. The measurements m are varied from $m = 100$ to $m = 900$ with a step of 100. For each of the nine measurements m the sparsity of the optimal representation $\tilde{x}$ is varied from $q = 1$ to $q = m$ with a step of one and for each q, 100 trials are conducted. The sensing matrix A is chosen by taking randomly m rows from an $n \times n$ normalized discrete cosine transform matrix. Each non-zero coefficient of the sparse representation is set to ± 1 with equal probability, while the sparsity pattern is chosen at random. All the generated problems are solved using mfIPM solver, the reconstruction is considered successful when $r.e \leq$ 1.0e-5. For each ratio ν_ρ we compute the success ratio $p(\nu_\rho) = S/100$, where S is the number of trials for which the $r.e \leq$ 1.0e-5. It has been demonstrated empirically in [45] that for any problem instance (A, b), where A is a partial DCT matrix a solver with average phase transition properties has $\max\{\nu_\rho \mid p(\nu_\rho) \geq 0.5\} \approx \hat{\nu}_\rho$. The latter means that the empirical average phase transition for 50% success rate overlaps with the theoretical average phase transition for Gaussian matrices. In Figure 8.7, we plot the empirical phase transition for 50% success rate of mfIPM and the theoretical average phase transition. The two curves overlap.

8.3 CS: coherent and redundant dictionaries

In this section we demonstrate the efficiency of pdNCG (Algorithm 5.1) against state-of-the-art methods for CS problems with coherent and redundant dictionaries. Details of CS problems have been presented in Section 2.3. In what follows we briefly discuss existing methods, we describe the setting of the experiments and finally numerical results are presented. All experiments that are demonstrated in this section have appeared in [41]. The experiments that are demonstrated can be reproduced by downloading the software from `http://www.maths.ed.ac.uk/ERGO/pdNCG/`.

8.3.1 Existing algorithms

We compare pdNCG with two state-of-the-art first-order methods, TFOCS [11] and TVAL3 [77].

- Templates for First-Order Conic Solvers (TFOCS) is MATLAB software for the solution of signal reconstruction problems. TFOCS solves the dual problem of

$$\text{minimize } \tau\|W^*x\|_1 + \frac{\mu_{T_1}}{2}\|x - x^0\|_2^2 + \frac{1}{2}\|Ax - b\|_2^2, \tag{8.4}$$

  where μ_{T_1} is a positive constant. TFOCS also solves the dual problem of

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \|W^*x\|_1 + \frac{\mu_{T_2}}{2}\|x - x^0\|_2^2 \\ \text{subject to:} & \|Ax - b\|_2 \leq \epsilon, \end{array} \tag{8.5}$$

  where $\epsilon > 0$. Although problems (8.4) and (8.5) are non-smooth, the regularization terms $\mu_{T_1}/2\|x - x^0\|_2^2$ and $\mu_{T_2}/2\|x - x^0\|_2^2$ yield smooth convex dual problems, which can be solved by standard first-order methods. In particular, the smooth dual problems are solved using the Auslender and Teboulle's accelerated first-order method [3]. In our experiments we present results for TFOCS for both problems (8.4) and (8.5). We denote by TFOCS_unc the version that solves the unconstrained problem (8.4) and by TFOCS_con the version that solves the constrained problem (8.5). TFOCS can be downloaded from [11].

- Total-Variation minimization by Augmented Lagrangian and ALternating direction ALgorithms (TVAL3) is a MATLAB software for the solution of signal reconstruction problems regularized with the total-variation seminorm. TVAL3 reformulates problem (5.1) to the equivalent problem:

$$\text{minimize } \tau\sum_{i=1}^{l}\|\Omega_i^\intercal x\|_2 + \frac{1}{2}\|Ax - b\|_2^2, \tag{8.6}$$

  where $\Omega_i = [ReW_i, ImW_i] \in \mathbb{R}^{n\times 2}$. Then it solves the augmented La-

grangian reformulation of problem (8.6), which is:

$$\text{minimize } \sum_{i=1}^{l}(\|u_i\|_2 + \frac{\beta}{2}\|\Omega_i^\intercal x - u_i\|_2^2 - v_i^\intercal(\Omega_i^\intercal x - u_i)) + \frac{1}{2}\|Ax - b\|_2^2, \quad (8.7)$$

where $u_i, v_i \in \mathbb{R}^2$ and β is a positive constant. The augmented Lagrangian in (8.7) is minimized for variables $x \in \mathbb{R}^n$ and u_i $i = 1, 2, \cdots, l$. The parameters v_i $i = 1, 2, \cdots, l$ are handled by the method.

Other solvers are NestA [10] and C-SALSA [1], which can also solve (5.1) but they are applicable only in the case that $(AA^\intercal)^{-1}$ is available. Another method is the Primal-Dual Hybrid Gradient (PDHG) in [49]. PDHG has been reported to be very efficient for imaging applications such as denoising and deblurring, for which matrix A is the identity or a square and full-rank matrix which is inexpensively diagonalizable. Unfortunately, this is not the case for the CS problems that we are interested in. However, for all previous methods the matrix inversion can be replaced with a solution of a linear system at every iteration of the methods or a one-time cost of a factorization. To the best of our knowledge, there are no available implementations with such modifications for these methods.

There exists also a generic proximal algorithm for total-variation [37] and the Generalized Iterative Soft Thresholding (GISTA) in [78] for which we do not have generic implementations for CS problems.

8.3.2 Equivalent problems

Solvers pdNCG, TFOCS and TVAL3 solve similar but not equivalent problems. In particular, pdNCG solves problem (5.2), which is parameterized by τ and μ. TFOCS solves problems (8.4) and (8.5), which are parameterized by τ, μ_{T_1} and μ_{T_2}, ϵ, respectively. TVAL3 solves problem (8.7), which is parameterized by τ and β.

In our experiments we put significant effort in calibrating parameters τ, μ, μ_{T_1}, μ_{T_2}, ϵ and β such that all methods solve similar problems. First, we set $\epsilon = \|b - \hat{b}\|_2$ in (8.5), where $\hat{b}$ is the noiseless sampled signal. Hence problem (8.5) is parameterized with the optimal ϵ. Then we find an approximation of the optimal τ. By optimal τ we mean the value of τ for which, problems (5.1) and (8.5) are equivalent if $\epsilon = \|b - \hat{b}\|_2$ and $\mu_{T_2} = 0$. Let ω denote the optimal Lagrange multiplier of (8.5). If $\epsilon = \|b - \hat{b}\|_2$ and $\mu_{T_2} = 0$, then it is easy to show that for $\tau := 2/\omega$ problems (5.1) and (8.5) are equivalent.

The exact optimal Lagrange multiplier ω is not known a-priori. However it can be calculated by solving to high accuracy the dual problem of (8.4) with TFOCS. Unfortunately, the majority of the experiments that we perform are large-scale and TFOCS converges slowly for $\mu_{T_2} \approx 0$. For this reason, we first solve (8.5) using TFOCS with a moderate μ_{T_2}, in order to obtain an approximate optimal Lagrange multiplier ω in reasonable CPU time. Then we set $\tau := 2\gamma/\omega$, where γ is a positive constant, which is calculated empirically such that problems (5.1) and (8.5) have similar solution. If $\hat{b}$ is not available, then ϵ is set such that a visually pleasant solution is obtained.

The smoothing parameters μ_{T_1} and μ_{T_2} of TFOCS in (8.4) and (8.5), respectively, are set such that the corresponding obtained solutions have small relative error. Let x_{T_1} and x_{T_2} denote the obtained solutions from problems (8.4) and (8.5), then $\|x_{T_1} - \tilde{x}\|_2/\|\tilde{x}\|_2$ and $\|x_{T_2} - \tilde{x}\|_2/\|\tilde{x}\|_2$ are of $\mathcal{O}(10^{-1})$ or $\mathcal{O}(10^{-2})$, where $\tilde{x}$ is the known optimal noiseless solution. If $\tilde{x}$ is not available, μ_{T_1} and μ_{T_2} are set such that visually pleasant reconstructions are obtained.

The smoothing parameter μ of pdNCG is set such that $\|x_{pd} - \tilde{x}\|_2/\|\tilde{x}\|_2$ is also $\mathcal{O}(10^{-1})$ or $\mathcal{O}(10^{-2})$, where x_{pd} is the approximate optimal solution obtained by pdNCG. For all experiments that were performed the relative error between the solution of TFOCS and pdNCG is of $\mathcal{O}(10^{-2})$.

For TVAL3 parameter β is set experimentally such that $\|x_{tv} - \tilde{x}\|_2/\|\tilde{x}\|_2$ is $\mathcal{O}(10^{-1})$ or $\mathcal{O}(10^{-2})$, where x_{tv} is the approximate optimal solution obtained by TVAL3. Also, in all experiments TVAL3 obtained similar solution to TFOCS and pdNCG. For TVAL3 we have taken into account the comments of its authors that TVAL3 performs better for $\beta \in [2^4, 2^{13}]$. However, occasionally we had to set $\beta = 2^2$ in order to obtain a solution that was as visually pleasing as the solutions obtained from TFOCS and pdNCG.

8.3.3 Problem sets

We compare the solvers pdNCG, TFOCS and TVAL3 on image reconstruction problems, which are modelled using iTV.

Let p_v be the vertical number of pixels of the image to be reconstructed and p_h be the horizontal number of pixels. For simplicity we will assume that the image is square, hence, $p = p_v = p_h$. Additionally, we assume that the image is handled in a vectorized form, i.e., instead of an image of size $p \times p$ we have a vectorized image of size $p^2 \times 1$ where the columns of the image are stuck one after the other. In this case, for iTV the $W \in \mathbb{C}^{n\times n}$ matrix in problem (5.1) is square with $n = p^2$, complex and rank-deficient with $rank(W) = n - 1$. Matrix W corresponds to a discretization of the nabla operator and it measures local differences of pixels when applied on a vectorized image. In particular,

$$W = W_v + \sqrt{-1}W_h,$$

where $W_v \in \mathbb{R}^{n\times n}$ and $W_h \in \mathbb{R}^{n\times n}$. Matrix W_v measures vertical differences of pixels when applied on a vectorized image and it has the following non-zero components:

$$[W_v]_{p(j-1)+i,p(j-1)+i} = -1 \quad \text{and} \quad [W_v]_{p(j-1)+i,p(j-1)+i+1} = 1$$

$\forall j = 1, 2, \cdots, p$ and $\forall i = 1, 2, \cdots, p-1$. Matrix W_h measures horizontal differences of pixels when applied on a vectorized image and it has the following non-zero pattern:

$$[W_h]_{p(j-1)+i,p(j-1)+i} = -1 \quad \text{and} \quad [W_h]_{p(j-1)+i,p(j-1)+i+p} = 1$$

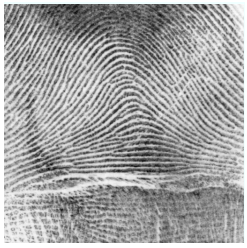
$\forall j = 1, 2, \cdots, p-1$ and $\forall i = 1, 2, \cdots, p$.

We separate the images to be reconstructed into two sets, which are shown

(a) House: 256^2

(b) Peppers: 256^2

(c) Lena: 512^2



(d) Fingerprint: 512^2

(e) Boat: 512^2

(f) Barbara: 512^2

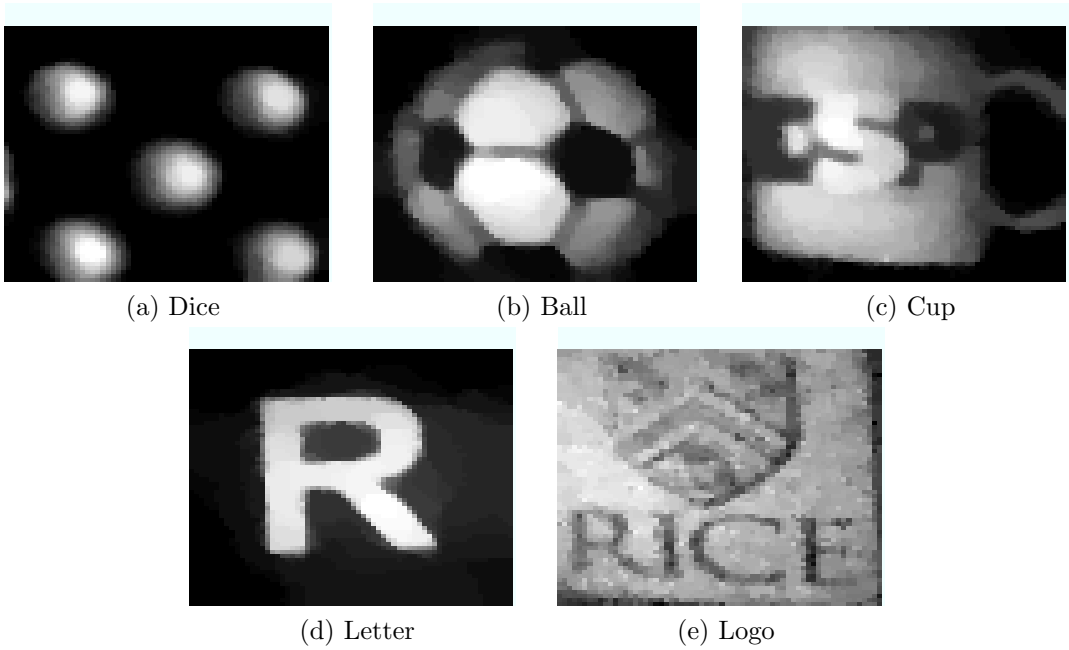
(g) Shepp-Logan: variable size

Figure 8.8: Benchmark images, the number of pixels for each image is given in the sub-captions. For Figure 8.8g the size varies depending on the experiment

in Figures 8.8 and 8.9. Figure 8.8 includes some standard images from the image processing community. There are seven images in total, the house and the peppers, which have 256×256 pixels and Lena, the fingerprint, the boat and Barbara, which have 512×512 pixels. Finally, the image Shepp-Logan has variable size depending on the experiment. Figure 8.9 includes images which have been sampled using a single-pixel camera [47]. Briefly a single-pixel camera samples random linear projections of pixels of an image, instead of directly sampling pixels. The problem set can be downloaded from `http://dsp.rice.edu/cscamera`. In this set there are in total five sampled images, the dice, the ball, the mug the letter R and the logo. Each image has 64×64 pixels.

8.3.4 Termination criteria, parameter tuning and hardware

The version 1.3.1 of TFOCS has been used. The termination criterion of TFOCS is by default the relative step-length. The tolerance for this criterion is set to the default value, except in cases that certain suggestions are made in TFOCS software package or the corresponding paper [11]. The default Auslender and Teboulle's single-projection method is used as a solver for TFOCS. Moreover, as



(a) Dice (b) Ball (c) Cup

(d) Letter (e) Logo

Figure 8.9: Benchmark images which were sampled using the single-pixel camera [47]

suggested by the authors of TFOCS, appropriate scaling is performed on matrices A and W, such that they have approximately the same Euclidean norms. All other parameters are set to their default values, except in cases that specific suggestions are made by the authors. Generally, regarding tuning of TFOCS, substantial effort has been made in guaranteeing that problems are not over-solved.

Regarding pdNCG, parameter η in (5.14) is set to 1.0e-1, which for the problems of our interest was sufficient to solve the problems fast. The maximum number of backtracking line-search iterations is fixed to 10. Moreover, the backtracking line-search parameters c_1 and c_2 in Step 4 of pdNCG (Algorithm 5.1) are set to 9.0e-1 and 1.0e-3, respectively. For iTV the preconditioner is a five-diagonal matrix, hence systems with it are solved exactly. Finally, the constant ρ of the preconditioner in (5.16) is set to 5.0e-1.

Regarding TVAL3, we fix the number of maximum iterations to 1000 and any other parameters that were not discussed previously are set to their default values.

All solvers are MATLAB implementations and all experiments are run on a MacBook Air running OS X 10.10.1 with 2 GHz (3 GHz turbo boost) Intel Core Duo i7 processor using MATLAB R2012a. The cores were working with frequency 2.7 - 3 GHz during the experiments and we did not observe any CPU throttling.

8.3.5 Dependence of pdNCG on smoothing parameter

In this subsection we present the performance of pdNCG for decreasing values of the smoothing parameter μ. For this experiments we use the images from Figures 8.8a to 8.8f. The CS matrix for all experiments is a partial DCT matrix with $m \approx n/4$ and n is equal to the number of pixels of each image in Figure 8.8. For

μ	House	Peppers	Lena	Fingerprint	Boat	Barbara
1.0e-02	4	4	20	20	19	19
1.0e-04	9	9	43	183	43	43
1.0e-07	11	11	49	282	68	102
1.0e-10	12	13	66	282	81	103
1.0e-13	12	13	66	310	71	100

Table 8.7: Performance of pdNCG for decreasing values of the smoothing parameter μ. For this experiment the images from Figures 8.8a to 8.8f have been used. The table shows the CPU time in seconds required for each combination of μ and problem

Solver	64×64	128×128	256×256	512×512	1024×1024
TFOCS_con	17	23	56	260	1018
TFOCS_unc	19	30	77	378	1468
TVAL3	5	8	37	99	365
pdNCG	**2**	**6**	**12**	**62**	**250**

Table 8.8: Performance of pdNCG, TFOCS and TVAL3 for increasing problem size. The image Shepp-Logan from Figure 8.8g has been used for this experiment. The table shows the required CPU time for each solver

all experiments the sampled signals have SNR equal to 15 dB.

The result of the experiments are shown in Table 8.7. In Table 8.7 notice that there is always a large increase in CPU time when μ changes from $\mu = 1.0e$-02 to 1.0e-04. This is because for $\mu = 1.0e$-02 pdNCG relies only on continuation (Algorithm 5.2), while for values of μ equal or smaller than 1.0e-04 preconditioning is necessary and it is automatically activated using the technique described in Section 5.7. Overall pdNCG had a stable performance with respect to the smoothing parameter μ. We believe that the good performance of the proposed preconditioner is responsible for this result. Without the preconditioner the performance of pdNCG for $\mu \leq 1.0e$-04 worsens noticeably.

8.3.6 Dependence on problem size

We now present the performance of methods pdNCG, TFOCS and TVAL3 as the size n of the problem increases. The image from Figure 8.8g has been used for this experiment. Again, the CS matrix for all experiments is a partial DCT matrix with $m \approx n/4$. The sampled signals have SNR equal to 15 dB.

The results are shown in Table 8.8. Observe that all methods exhibit a linear-like increase in CPU time as a function of the size of the problem. We denote with bold the problems for which pdNCG was the fastest method.

Solver	SNR	House	Peppers	Lena	Fingerprint	Boat	Barbara
TFOCS_con	90	57	56	261	271	264	262
	75	56	57	261	267	261	260
	60	56	56	262	274	261	260
	45	56	56	262	268	260	261
	30	56	56	260	267	260	259
	15	56	56	266	280	260	259
TFOCS_unc	90	78	78	383	397	383	378
	75	78	78	381	386	379	376
	60	78	78	381	399	380	380
	45	78	78	381	390	378	379
	30	78	78	380	382	380	379
	15	78	77	384	397	377	378
TVAL3	90	7	4	19	22	20	20
	75	7	4	18	22	19	21
	60	7	4	17	22	17	18
	45	6	4	17	23	17	19
	30	4	9	39	65	41	46
	15	13	16	60	76	61	63
pdNCG	90	18	23	114	119	122	113
	75	18	24	114	117	122	112
	60	18	24	115	114	121	113
	45	18	24	115	113	121	112
	30	19	17	78	113	78	107
	15	**10**	**9**	**41**	229	**44**	111

Table 8.9: Performance of pdNCG, TFOCS and TVAL3 for increasing level of noise (decreasing SNR). SNR is measured in dB. For this experiment the images from Figures 8.8a to 8.8f have been used. The table shows the CPU time in seconds required by each solver

8.3.7 Dependence on the level of noise

In this experiment we compare the solvers pdNCG, TFOCS and TVAL3 as the level of noise increases. For this experiment we use the images from Figures 8.8a to 8.8f. The CS matrix for all experiments is a partial DCT matrix with $m \approx n/4$.

In Table 8.9 we present the results of this experiment. In the second column of Table 8.9 the SNR is shown, which is decreasing from 90 dB to 15 dB in six steps. The rest of the table shows the CPU time, which was required for each solver. Overall pdNCG has good performance for problems with large level of noise, i.e., SNR equal to 15 dB. We denote with bold the problems for which pdNCG was the fastest solver.

Solver	m	House	Peppers	Lena	Fingerprint	Boat	Barbara
TFOCS_con	75%	61	61	255	262	260	273
	50%	60	57	260	256	263	268
	25%	57	60	253	250	273	262
TFOCS_unc	75%	80	78	367	374	375	393
	50%	85	81	374	371	383	384
	25%	77	80	370	364	380	384
TVAL3	75%	13	14	55	56	54	56
	50%	11	14	55	71	57	57
	25%	14	17	57	72	61	64
pdNCG	75%	**9**	**9**	74	196	62	108
	50%	**8**	**9**	57	217	91	106
	25%	**10**	**9**	**40**	222	**42**	113

Table 8.10: Performance of pdNCG, TFOCS and TVAL3 for decreasing number of measurements m. In the second column the percentage of measurements is shown, for example 75% means that $m \approx 3n/4$, where n is the number of pixels in the image to be reconstructed. For this experiment the images from Figures 8.8a to 8.8f have been used. The table shows the CPU time in seconds required by each solver

8.3.8 Dependence on number of measurements

In this experiment we compare the three methods for decreasing number of measurements m. For this experiment we use the images from Figures 8.8a to 8.8f. The CS matrix is a partial DCT matrix. For all experiments the sampled signals have SNR equal to 15 dB.

The results of this experiment are shown in Table 8.10. We denote with bold the problems for which pdNCG was the fastest method.

8.3.9 Single-pixel camera

We now compare TFOCS with pdNCG on realistic image reconstruction problems where the data have been sampled using the single-pixel camera [47]. In this experiment we compare only with TFOCS_con. This is because in all previous experiments TFOCS_con was faster than TFOCS_unc. Additionally, we were not able to make TVAL3 to converge to a solution which was as visually pleasant as the solutions obtained by TFOCS_con and pdNCG. We believe that this due to the different CS matrix A in these experiments. In particular, matrix $A \in \mathbb{R}^{m\times n}$, where $n = 64^2$ and $m \approx 0.4n$, is a partial Walsh basis which takes values $0/1$ instead of ± 1. We noticed that this matrix A does not satisfy the RIP property in Definition 2.2 with small δ_q. Therefore, the least-squares term in problem (5.1) might be ill-conditioned and this causes difficulties for TVAL3.

Moreover the optimal solutions are unknown, additionally the level of noise is unknown. Hence the reconstructed images can only be compared by visual inspection. For all four experiments 40% of measurements are selected uniformly

Problem	$\boldsymbol{m}$	$\boldsymbol{n}$	$\mathbf{nnz(A)/(mn)}$	$\boldsymbol{\tau}$
[82] real-sim	72,309	20,958	2.40e-02	4.00e-02
[76] rcv1	20,242	47,236	1.60e-02	4.00e-02
[71] news20	19,996	1,355,191	3.35e-04	4.00e-02
[121] kdd (algebra)	8,407,752	20,216,830	1.79e-06	2.00e-00
[121] kdd (br. to alg.)	19,264,097	29,890,095	9.83e-07	2.00e-00
[114] webspam	350,000	16,609,143	2.24e-04	4.00e-02

Table 8.11: Properties of six ℓ_1-LR problems, which are used as benchmarks. The second and third columns show the number of training samples and features, respectively. The fourth column shows the sparsity of matrix A. The last column is the τ found using fivefold cross-validation

at random.

The reconstructed images by the solvers TFOCS_con and pdNCG are presented in Figure 8.10. Solver pdNCG was faster on four out of five problems. On problems where pdNCG was faster it required on average 1.5 times less CPU time TFOCS_con. Although it would be possible to tune pdNCG such that it is faster on all problems, we preferred to use its (simple) default tuning in order to avoid a biased comparison.

8.4 ℓ_1-regularized logistic regression

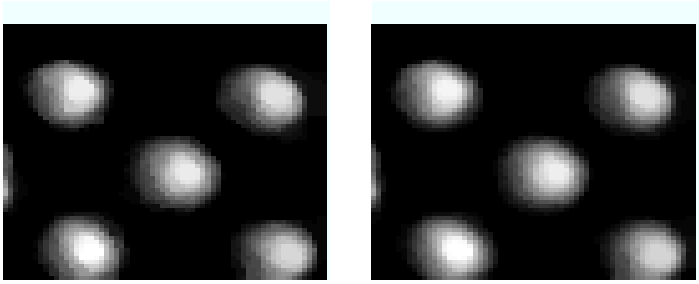
In this section we compare pdNCG, which was presented in Chapter 4, to FISTA and PCDM on six real world ℓ_1-LR problems. For ℓ_1-LR the function $\varphi(x)$ in (4.1) is:

$$\varphi(x) = \sum_{i=1}^{m} \log(1 + e^{-b_i x^\intercal a_i}),$$

where $a_i \in \mathbb{R}^n \; \forall i = 1, 2, \ldots, m$ are the training samples and $b_i \in \{-1, +1\}$ are the corresponding labels. For more details about support vector machine problems we refer the reader to Section 2.4.

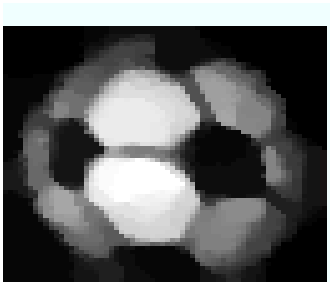
All experiments that are demonstrated in this section have appeared in [53]. The experiments can be reproduced by downloading the software from `http://www.maths.ed.ac.uk/ERGO/pdNCG/`. The description of the compared methods, the implementation details and the parameter tuning is the same as in Subsections 8.1.1, 8.1.2 and 8.1.3, respectively.

We present six ℓ_1-LR problems that are large-scale and sparse. Exact information for these problems is given in Table 8.11. In this table matrix $A \in \mathbb{R}^{m \times n}$ has the training samples in its rows, the fourth column shows the sparsity of matrix A, where $nnz(A)$ is the number of non-zero components in A. The last column shows the τ that gave the classification with the highest accuracy after performing a fivefold cross validation over various τ values as proposed in [70]. The calculated values τ resulted for all problems in more than 90% classification accuracy. All problems in Table 8.11 can be downloaded from the collection of LSVM problems in [33]. Notice that for most of the problems in Table 8.11,

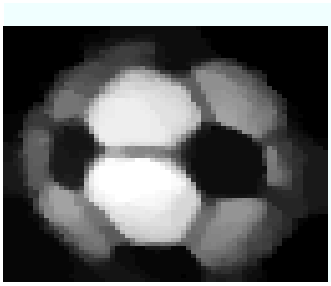


(a) TFOCS_con, 25 sec.

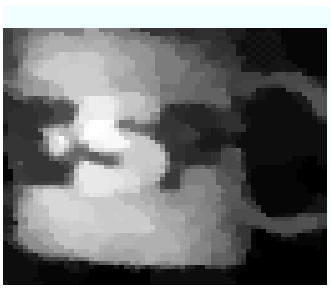
(b) pdNCG, 7 sec.



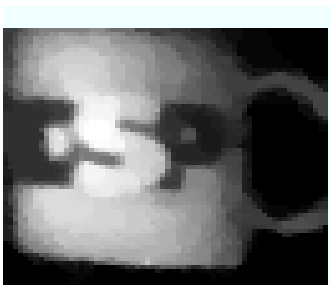
(c) TFOCS_con, 24 sec.



(d) pdNCG, 15 sec.



(e) TFOCS_con, 37 sec.



(f) pdNCG, 15 sec.

(g) TFOCS_con, 26 sec.

(h) pdNCG, 27 sec.

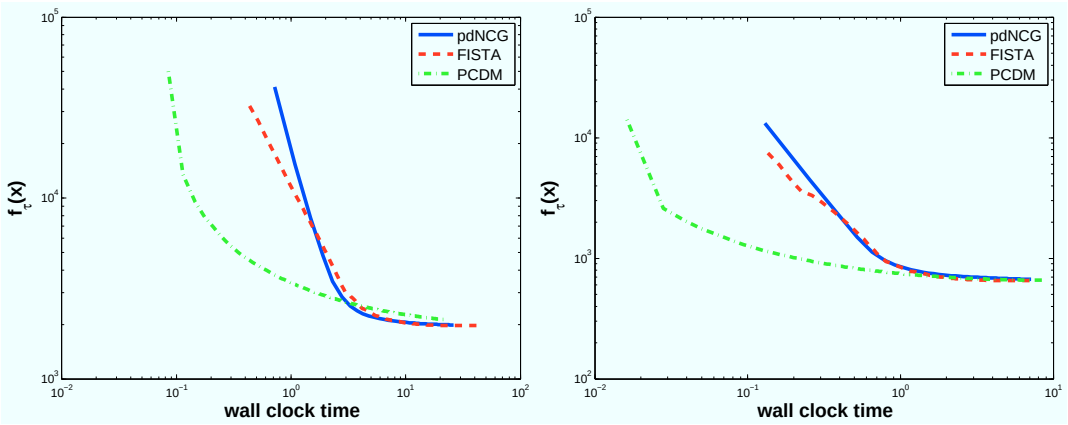
(i) TFOCS, 49 sec.

(j) pdNCG, 33 sec.

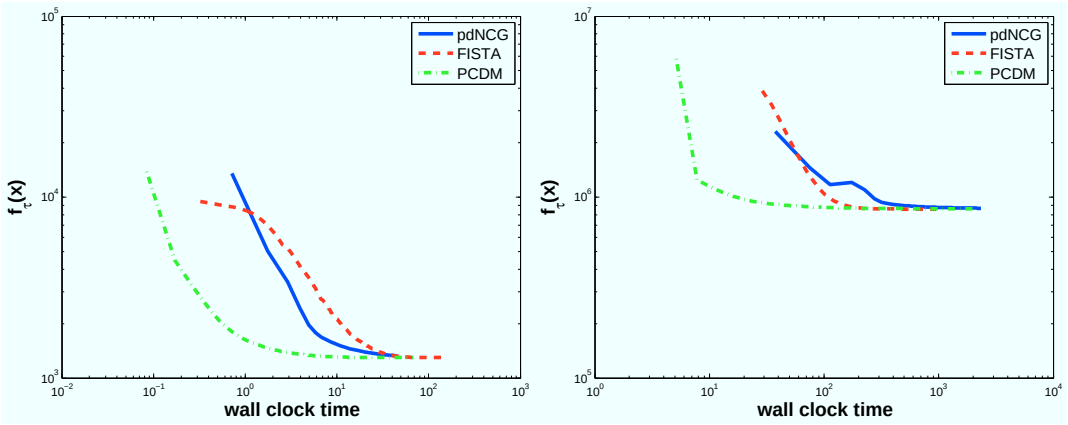
Figure 8.10: Experiment on realistic image reconstruction where the samples are acquired using a single-pixel camera. The subcaptions of the figures show the required seconds of CPU time for the image to be reconstructed for each solver

$m < n$, which means that the problems are not strongly-convex everywhere as assumed in (9.3). However, the problems in Table 8.11 have a unique solution and they are locally strongly-convex to the optimal solution. We choose to solve these instances because the problems for which $m > n$ in collection [33] are small scale, hence, possible numerical experiments might not provide significant insight into the behaviour of the methods. It is important to mention that all implementations of the compared methods can handle such cases without any modification.

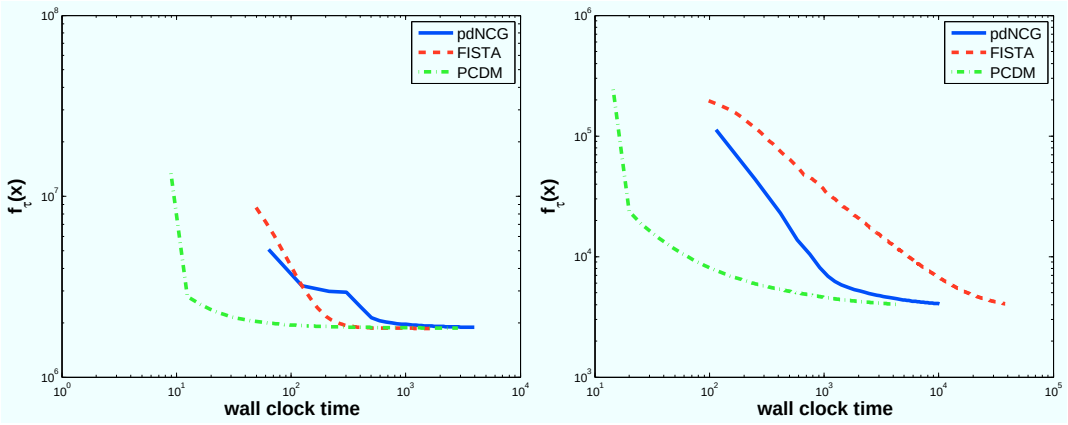
The results of the comparison among the solvers pdNCG, FISTA and PCDM are shown in Figure 8.11. PCDM was the fastest algorithm in the majority of the experiments and pdNCG and FISTA had similar performance in the majority of the experiments. Notice in Subfigure 8.11d that for pdNCG the objective function f_τ seems not to decrease always monotonically. This behaviour might have occurred because backtracking line-search can terminate before the condition in Step 4 of pdNCG (Algorithm 4.1) is satisfied, if the maximum number of backtracking iterations is exceeded.



(a) real-sim (b) rcv1



(c) news20 (d) kdd (algebra)



(e) kdd (bridge to algebra) (f) webspam

Figure 8.11: Comparison of the performance of pdNCG, FISTA and PCDM on ℓ_1-LR problems.

Chapter 9

Conclusions and Further Developments

In this chapter we give our conclusions on the efficiency of techniques and methods that we developed. Additionally, we discuss limitations of the proposed methods and possible further developments and improvements.

9.1 Conclusions

In Figure 9.1 we present a diagram of the efficiency of optimization methods based on the size and the conditioning of problem:

$$\text{minimize } \tau\psi(x) + \varphi(x), \tag{9.1}$$

where $\psi : \mathbb{R}^n \to \mathbb{R}$ is a (possibly) nonsmooth convex function and $\varphi : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. Figure 9.1 is simplistic. However, we believe that it presents the current state of the field in an intuitive way. The diagram is based on our experience on a broad class of numerical experiments performed over the last four years.

Figure 9.1 summarizes the following: if we have a well-conditioned and small-scale problem, then we expect that both first- and higher-order methods will work well, so either method class is appropriate. If we have a well-conditioned and large-scale problem, then a first-order method is possibly a better option. If we have an ill-conditioned and small-scale problem, then a higher-order method is likely to be a better option than a first-order method. The question mark in Figure 9.1 represents the part of the map which, to the best of our knowledge, has not been adequately addressed up to this moment. There have been numerous first- and higher-order methods for the first three combinations of conditioning and size of the problem. However, there are not many developments for the fourth part of the map in Figure 9.1, which corresponds to ill-conditioned and large-scale problems. We believe that the main reason for this is the inherent difficulty in the development of higher-order methods with provable or empirical running time $\mathcal{O}(n^2)$ or even $\mathcal{O}(n)$ (we ignore terms of minor importance). This is because higher-order methods usually involve the solution of a linear system at every iteration and in the worst-case the solution of a linear system can be done in $\mathcal{O}(n^3)$ running time.

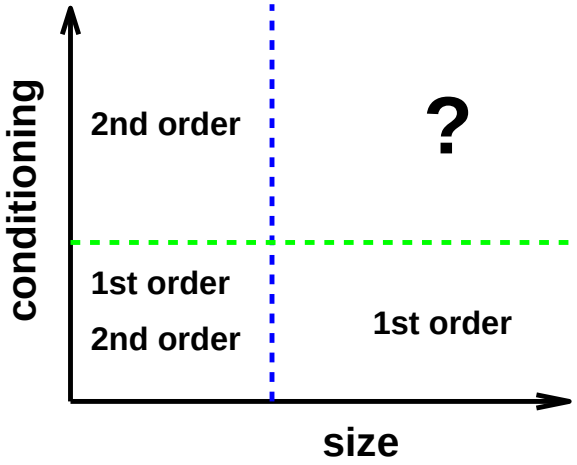


Figure 9.1: Efficient methods for each combination of conditioning and size of problem

It is important to mention that $\mathcal{O}(n^3)$ corresponds to solving a completely dense linear system using a direct method. In many large-scale optimization problems the matrices involved are sparse and the $\mathcal{O}(n^3)$ running time is very conservative. However, in the applications that we studied the linear systems that appear are dense or unstructured, see Subsection 8.2.7. Hence, despite the fact that higher-order methods might require few iterations for convergence, i.e., short-step IPMs can have worst-case iteration complexity $\mathcal{O}(\sqrt{n})$, one single iteration is expensive and this increases the running time of the methods.

This thesis aimed to make higher-order methods inexpensive. In particular, in Chapter 4 we studied a primal-dual Newton Conjugate Gradients (pdNCG) and in Chapter 6 we developed and studied a matrix-free Interior-Point Method (mfIPM). We ensured low computational complexity per iteration for these methods by using inexact Newton directions; linear systems at every iteration are solved inexactly using Conjugate Gradients (CG). We prove that the worst-case iteration complexity of the proposed pdNCG is nearly the same as the standard Newton method in Subsection 9.5 in [18], which is based on the exact solution of linear systems. For both pdNCG and mfIPM we develop provably (asymptotically) efficient and inexpensive preconditioning techniques (systems with the preconditioner are solved in $\mathcal{O}(n)$ running time) for applications such as Compressed Sensing (CS). Using our preconditioning techniques we show empirically that the running time of pdNCG and mfIPM scales linearly with respect to the size of the problem, see Subsections 8.3.6 and 8.2.7.

Finally, in addition to the development of inexpensive higher-order methods in this thesis we aimed to develop a problem generator. The proposed problem generator scales linearly with respect to the problem size in terms of running time and memory requirements, see Subsection 8.1.8. The motivation for studying problem generation is the need to perform controlled testing of newly developed methods as the conditioning of the problem changes and the dimensions of the problem increase. We believe that controlled testing is necessary for understanding weaknesses and strengths of methods. Below we discuss our main contributions in

each chapter.

In Chapter 4 we have developed and studied an inexpensive but still robust pdNCG method. The proposed method is developed for the solution of ℓ_1-regularized problems, that might display some degree of ill-conditioning; that is, display noticeable differences of the magnitude of eigenvalues. For such problems it is crucial that the methods capture information from the second-order derivative of the smooth function. In Subsections 8.1, 8.3 and 8.4 we have given synthetic and real-world machine learning problems which are ill-conditioned, and we provided computational evidence that the proposed method is efficient on these problems. We have shown that by using the properties of CG, the convergence analysis of pdNCG can be performed in a variable metric, which is defined based on approximate second-order derivatives. The variable metric opens the door for a tight convergence analysis of pdNCG, which includes global and local convergence rates, explicit definition of fast local convergence region and worst-case iteration complexity. We prove that pdNCG has similar worst-case iteration complexity with the standard Newton's method which relies on the exact solution of linear systems, although at every iteration linear systems are solved inexactly in pdNCG.

In Chapter 5 we show that for CS problems an inexpensive preconditioner can be designed for pdNCG that speeds up even further the approximate solution of linear systems. Extensive numerical experiments are presented in Subsection 8.3, which verify our arguments. Spectral analysis of the preconditioner is performed and confirms its very good limiting behaviour. Systems with the preconditioner can be solved in $\mathcal{O}(n)$ time, and we also noticed that using our preconditioner pdNCG has $\mathcal{O}(n)$ running time empirically, see Subsection 8.3.6.

In Chapter 6 we present a computationally inexpensive matrix-free Interior-Point Method (mfIPM) for the ℓ_1-regularized problems arising in the field of CS. At every iteration of the proposed mfIPM the direction is obtained by solving a linear system using CG. Unfortunately, the matrices in these systems tend to become ill-conditioned as the algorithm converges, Hence, the conjugate gradient method might suffer from slow convergence. To remedy this ill-conditioning we propose a preconditioner for CG. Systems with the preconditioner can be solved in $\mathcal{O}(n)$ running time. The proposed preconditioning technique exploits features of CS matrices as well as interior point methods. Its efficiency is justified theoretically and confirmed in numerical experiments. Computational experience in Subsection 8.2 shows that although the CS research community seems to favor first-order methods, a specialized (matrix-free) interior point method is very competitive and offers a viable alternative. The numerical experiments seem to confirm that the proposed mfIPM has $\mathcal{O}(n)$ running time, see Subsection 8.2.7.

In Chapter 7 we present an instance generator for ℓ_1-regularized sparse least-squares problems. The purpose of the generator is to construct very large-scale problem instances. Therefore, it scales well as the number of variables increases, both in terms of memory requirements and time. Additionally, the generator allows control of the conditioning and the sparsity of the problem. Examples are provided on how to exploit the previous advantages of the proposed generator. We believe that the optimization community needs such a generator to enable the fair assessment of new algorithms. Using the proposed generator we constructed

very large-scale sparse instances (up to one trillion variables), which vary from very well-conditioned to moderately ill-conditioned.

9.2 Limitations and further developments

9.2.1 Convergence theory

The proposed pdNCG solves the problem:

$$\underset{x}{\text{minimize}} \;\; f_\tau^\mu(x) := \tau\psi_\mu(x) + \varphi(x), \tag{9.2}$$

where

$$\psi_\mu(x) = \sum_{i=1}^{n}((\mu^2 + x_i^2)^{\frac{1}{2}} - \mu),$$

which is a smooth approximation of the ℓ_1-norm, see Chapter 3. Let us assume that for any x the second derivative $\nabla^2 f_\tau^\mu$ is uniformly bounded:

$$\lambda_n I \preceq \nabla^2 f_\tau^\mu(x) \preceq \tilde{\lambda}_1 I, \tag{9.3}$$

with $0 < \lambda_n \leq \tilde{\lambda}_1$. Furthermore, let $\kappa := \tilde{\lambda}_1/\lambda_n$ be a uniform bound on the condition number of the smoothed problem (9.2). For the smoothed problem, optimal first-order methods have convergence rate $\mathcal{O}(1/\kappa)$, see Subsection 9.3 in [18]. On the other hand, Newton-type methods have convergence rate $\mathcal{O}(1/\kappa^2)$, see Subsection 9.5 in [18], or for pdNCG see Subsection 4.8.3, which is worse by an order! To the best of our knowledge, this is an issue for the convergence of Newton-type methods in the global phase. As far as we are concerned, in order to improve the convergence rate of Newton-type methods, techniques like cubic regularization can be employed, see [28, 90] (although cubic regularization settings were not part of the present thesis).

One should not be discouraged in using Newton-type methods. We base this argument on extensive empirical evidence. For all the numerical experiments that we performed in Chapter 8 the proposed pdNCG converged in less than 30 iterations, although κ was large, i.e., $\geq 10^2$. On the other hand the state-of-the-art first-order methods that we used required hundreds or thousands of iterations depending on how large κ was. Therefore, our empirical observations do not follow worst-case theory closely. This implies that the worst-case convergence rates for the global phase of Newton-type methods for unconstrained convex problems can be overwhelmingly pessimistic and one should not entirely rely on them.

9.2.2 Smoothing

In Chapter 3 we discussed the Moreau smoothing technique, which was used to smooth the ℓ_1-norm. However, we did not discuss how the smoothing technique can be applied to other non-smooth functions. In this subsection we present a brief discussion on this issue.

Let $\psi(x)$ be a non-smooth function with domain $x \in \mathbb{R}^n$, its smooth approximation according to Moreau smoothing technique is

$$\psi_\mu(x) := \sup_{y \in \partial\psi(x)} \langle x, y \rangle - \psi^*(y) - \mu d(y) \tag{9.4}$$

and the gradient of ψ_μ is

$$\nabla\psi_\mu(x) := \arg\sup_{y \in \partial\psi(x)} \langle x, y \rangle - \psi^*(y) - \mu d(y), \tag{9.5}$$

where $\partial\psi(x)$ is the sub-differential of ψ at x and the domain of ψ^*, ψ^* is the convex conjugate of ψ, $d(y)$ is a proximity function for $y \in \partial\psi(x)$ and $\mu > 0$ is the smoothing parameter. It is clear that we need to have a closed form expression of (9.4) and (9.5) or at least an inexpensive procedure which can be used to evaluate (9.4) and (9.5). Otherwise, a single evaluation of ψ_μ and $\nabla\psi_\mu$ might be as expensive as the solution of the problem (9.2).

It is easy to check that a closed form expression of (9.4) and (9.5) exists for non-smooth convex vector norms, such as the ℓ_∞ norm, $\psi(x) := \|x\|_\infty$. In this case we have that

$$\psi^*(y) = \begin{cases} 0, & \text{if } \|y\|_1 \leq 1 \\ +\infty, & \text{otherwise.} \end{cases} \tag{9.6}$$

Also, the domain of ψ^* is equivalent to $\{y \in \mathbb{R}^n \mid \|y\|_1 \leq 1\}$. Therefore, problem (9.4) is equivalent to

$$\psi_\mu(x) := \sup_{\|y\|_1 \leq 1} \langle x, y \rangle - \mu d(y), \tag{9.7}$$

similarly for (9.5). We can set $y = u - v$ with $u, v \geq 0$ and rewrite (9.7)

$$\begin{aligned} \psi_\mu(x) := \quad & \sup_{u,v \in \mathbb{R}^n} && \langle x, u - v \rangle - \mu d(u, v) \\ & \text{subject to:} && 1_n^\intercal u + 1_n^\intercal v \leq 1, \\ & && u, v \geq 0, \end{aligned}$$

where $1_n \in \mathbb{R}^n$ is a vector of ones. By defining

$$d(u, v) := \sum_{i=1}^{n} (u_i \log u_i + v_i \log v_i) + \log 2n$$

we obtain

$$\psi_\mu(x) = \mu \sum_{i=1}^{n} \log(\cosh \frac{x_i}{\mu}).$$

However, as we already mentioned such closed form expressions for (9.4) and (9.5) are not always available for any non-smooth function ψ.

A second limitation of smoothing is that a small smoothing parameter $\mu \approx 0$ might result in ill-conditioned smooth functions. In particular, at the point of non-smoothness the kink of the function is replaced with a sharp quadratic. This means that when the current iteration of a method is close to the point of non-smoothness then the method might run into numerical difficulties due to

ill-conditioning. Fortunately, this problem can be avoided by efficient and inexpensive preconditioning. For example, in Chapter 5 we developed a provable (asymptotically optimal) preconditioning technique for CS applications to remedy the ill-conditioning caused due to smoothing. However, it is not always the case that provably efficient preconditioners are easily obtainable. This might discourage many users/researchers from using/studying higher-order methods. We are hopeful though that sophisticated preconditioning techniques can be developed by exploiting properties of matrices in many applications.

9.2.3 Preconditioning

In Chapter 5 we develop a provably (asymptotically) efficient preconditioner for CS. The core of the spectral analysis of the preconditioned matrices is the Restricted Isometry Property (RIP). However, there are many real-world sparse signal reconstruction problems, where the RIP is violated, i.e., see Subsections 8.2.2 and 8.3.9. Although the proposed preconditioner seems to offer significant improvement compared to no preconditioning, a new theory adapted to the properties of such special real-world problems has not been presented. We hope that the exploitation of new properties might result in new and provably efficient preconditioners.

In Subsections 8.3.6 and 8.2.7 we showed empirically that the running time of pdNCG and mfIPM scales linearly as the size of the problem increases for CS applications. We believe that this nice empirical performance of pdNCG and mfIPM is due to the efficient preconditioners that we developed. A possible future direction would be to study applications where if certain properties are exploited then linear or nearly linear running time of higher-order methods can be proved. For example, see [40] where a preconditioner is developed for generalized graph flow problems with multipliers on edges which are at most one.

Finally, it would be interesting to consider different parallel environments when developing new preconditioners. For example, distributed computers or multi-processor shared memory computers.

9.2.4 Problem generator

In Chapter 7 we developed a problem generator with running time and memory requirements that scale linearly if certain settings are considered. We provide MATLAB and C++ (for distributed computers) implementations for certain numerical experiments that were discussed in Section 8.1. The software can be downloaded from: `http://www.maths.ed.ac.uk/ERGO/trillion/`. However, a flexible implementation that allows the users to develop new problems different than the ones proposed in Section 8.1 is needed. Ideally, a state-of-the-art implementation should include distributed computers or other parallel environment settings, which would allow the user to generate huge-scale problems, as the one trillion variable problem (200 terabytes) that we constructed and discussed in Subsection 8.1.8.

Bibliography

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of image inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2011.

[2] N. Ahmed and K. R. Rao. *Orthogonal Transforms for Digital Signal Processing*. Springer-Verlag, Berlin, 1975.

[3] A. Auslender and Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM Journal on Optimization*, 16(3):697–725, 2006.

[4] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Journal Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[5] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[6] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

[7] A. Beck and M. Teboulle. Smoothing and first order methods: A unified framework. *SIAM J. Optim.*, 22(2):557–580, 2012.

[8] S. Becker. CoSaMP and OMP for sparse recovery. `http://www.mathworks.co.uk/matlabcentral/fileexchange/32402-cosamp-and-omp-for-sparse-recovery`, 2012.

[9] S. R. Becker. *Practical Compressed Sensing: modern data acquisition and signal processing*. PhD thesis, California Institute of Technology, April 2011.

[10] S. R. Becker, J. Bobin, and E. J. Candès. Nesta: A fast and accurate first-order method for sparse recovery. *SIAM J. Imaging Sciences*, 4(1):1–39, 2011.

[11] S. R. Becker, E. J. Candés, and M. C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011. Software available at `http://tfocs.stanford.edu`.

[12] E. Van Den Berg and M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal of Scientific Computing*, 31(2):890–912, 2008.

[13] E. van den Berg, M. P. Friedlander, G. Hennenfent, F. J. Herrman, R. Saab, and Ö. Yılmaz. Sparco: A testing framework for sparse reconstruction. *ACM Trans. Math. Software*, 35(4):1–16, 2009.

[14] M. Bertalmio, G. Sapiro, C. Ballester, and V. Caselles. Image inpainting. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 417–424, 2000.

[15] J. D. Blanchard, C. Cartis, and J. Tanner. Decay properties of restricted isometry constants. *Computational Optimization and Applications*, 16(7):572–575, 2009.

[16] J. D. Blanchard, C. Cartis, and J. Tanner. Compressed sensing: How sharp is the restricted isometry property? *SIAM Rev.*, 53(1):105–125, February 2011.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Journal Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[18] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press New York, NY, USA, 2004.

[19] R. H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for convex l-1 regularized optimization. Technical report, Sep. 2013. arXiv:1309.3529 [math.OC].

[20] E. Candés, L. Demanet, D. Donoho, and L. Ying. Curvelab. `http://www.curvelet.org/software.html`.

[21] E. Candés and J. Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969–985, 2007.

[22] E. J. Candés. Compressive sampling. *Proceedings of the International Congress of Mathematicians*, 2006.

[23] E. J. Candés and D. L. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise c^2 singularities. *Comm. Pure Appl. Math.*, 57:219–266, 2004.

[24] E. J. Candés, Y. C. Eldar, and D. Needell. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31(1):59–73, 2011.

[25] E. J. Candés and J. Romberg. Practical signal recovery from random projections. *In Wavelet Applications in Signal and Image Processing XI, Proc. SPIE Conf. 5914.*, 2004.

[26] E. J. Candés and J. Romberg. ℓ_1-magic. *Technical Report, Caltech*, 2007. `http://users.ece.gatech.edu/~justin/l1magic/`.

[27] E. J. Candés, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, 2006.

[28] C. Cartis, N. I. M. Gould, and P. L. Toint. Evaluation complexity of adaptive cubic regularization methods for convex unconstrained optimization. *Optimization Methods and Software*, 27(2):197–219, 2012.

[29] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. *Radon Series Comp. Appl. Math*, 9:263–340, 2010.

[30] A. Chambolle, R. A. DeVore, N. Y. Lee, and B. J. Lucier. Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Trans. Image Process.*, 7(3):319–335, 1998.

[31] R. H. Chan, T. F. Chan, and H. M. Zhou. Advanced signal processing algorithms. *in Proceedings of the International Society of Photo-Optical Instrumentation Engineers, F. T. Luk, ed., SPIE*, pages 314–325, 1995.

[32] T. F. Chan, G. H. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *SIAM J. Sci. Comput.*, 20(6):1964–1977, 1999.

[33] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[34] K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. Coordinate descent method for large-scale ℓ_2-loss linear support vector machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.

[35] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[36] R. Coifman, F. Geshwind, and Y. Meyer. Noiselets. *Appl. Comput. Harmon. Anal.*, 10(1):27–44, 2001.

[37] L. Condat. A generic proximal algorithm for convex optimization - Application to total-variation. *IEEE Signal Processing Letters*, 21(8):985–989, 2014.

[38] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[39] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[40] S. I. Daitch and D. A. Spielman. Faster lossy generalized flow via interior point algorithms. *STOC '08 Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 451–460.

[41] I. Dassios, K. Fountoulakis, and J. Gondzio. A preconditioner for a primal-dual Newton conjugate gradients method for compressed sensing problems. *Technical Report ERGO 14-021*, 2014. Software available at `http://www.maths.ed.ac.uk/ERGO/pdNCG/`.

[42] D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006.

[43] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. Inf. Theory*, 47(7):2845–2862, 2001.

[44] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[45] D. L. Donoho and J. Tanner. Precise undersampling theorems. *Proceedings of the IEEE*, 98(6):913–924, 2010.

[46] C. Dossal and S. Mallat. Sparse spike deconvolution with minimum scale. *In Proceedings of Signal Processing with Adaptive Sparse Structured Representations*, 81(3):123–126, 1994.

[47] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, S. Ting, K. F. Kelly, and R. G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008. `http://dsp.rice.edu/cscamera`.

[48] J. Eckstein. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 2012.

[49] J. E. Esser. *Primal Dual Algorithms for Convex Models and Applications to Image Restoration, Registration and Nonlocal Inpainting*. PhD thesis, University of California, 2010.

[50] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse Problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[51] S. Foucart. A note on guaranteed sparse recovery via $\ell 1$-minimization. *Applied and Computational Harmonic Analysis*, 29(1):97–103, 2010.

[52] K. Fountoulakis and J. Gondzio. Performance of first- and second-order methods for big data optimization. *Technical Report ERGO 15-?*, 2015. Software available at `http://www.maths.ed.ac.uk/ERGO/pdNCG/`.

[53] K. Fountoulakis and J. Gondzio. A second-order method for strongly convex ℓ_1-regularization problems. *Mathematical Programming (accepted)*, 2015. Software available at `http://www.maths.ed.ac.uk/ERGO/pdNCG/`.

[54] K. Fountoulakis, J. Gondzio, and P. Zhlobich. Matrix-free interior point method for compressed sensing problems. *Mathematical Programming Computation*, 6(1):1–31, 2014. Software available at `http://www.maths.ed.ac.uk/ERGO/mfipmcs/`.

[55] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Machine Learning Research*, 9:627–650, 2008.

[56] M. Frigo and S. G. Johnston. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[57] T. Goldstein, B. O'Donoghue, and S. Setzer. Fast alternating direction optimization methods. *Technical report, CAM Report 12-35, UCLA*, 2012.

[58] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.

[59] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[60] J. Gondzio. Matrix-free interior point method. *Computational Optimization and Applications*, 51(2):457–480, 2012.

[61] J. Gondzio. Convergence analysis of an inexact feasible interior point method for convex quadratic programming. *SIAM Journal on Optimization*, 23(3):1510–1527, 2013.

[62] E. T. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM J. Optim.*, 19(3):1107–1130, 2008.

[63] E. T. Hale, W. Yin, and Y. Zhang. Fixed-point continuation method for ℓ_1-minimization: Methodology and convergence. *SIAM J. Optim.*, 19(3):1107–1130, 2008.

[64] Per Christian Hansen, J. G. Nagy, and D. P. O'Leary. Deblurring images: Matrices, spectra and filtering. 17(1), 2008.

[65] Per Christian Hansen, V. Pereyra, and G. Scherer. *Least Squares Data Fitting with Applications*. JHU Press, 2012.

[66] B. He and X. Yuan. On the $\mathcal{O}(1/t)$ convergence rate of alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[67] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[68] G. Hennenfent and F. J. Herrmann. Random sampling: new insights into the reconstruction of coarsely-sampled wavefields. *In SEG International Exposition and 77th Annual Meeting*, 2007.

[69] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. *Proceedings of the 25th international conference on Machine Learning, ICML 2008*, pages 408–415, 2008.

[70] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2010.

[71] S. S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, 2005.

[72] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1995.

[73] C. T. Kelly. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, PA., 1995.

[74] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[75] M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61:263–280, 1993.

[76] D. D. Lewis, Yiming Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[77] C. Li, W. Yin, H. Jiang, and Y. Zhang. An efficient augmented Lagrangian method with applications to total variation minimization. *Computational Optimization and Applications*, 56(3):507–530, 2013.

[78] I. Loris and C. Verhoeven. On a generalization of the iterative soft-thresholding algorithm for the case of non-separable penalty. *Inverse Problems*, 27(12):1–15, 2011.

[79] Z. Lu, R. D. S. Monteiro, and J. W. O'Neal. An iterative solver-based infeasible primal-dual path-following algorithm for convex quadratic programming. *SIAM Journal on optimization*, 17:287–310, 2006.

[80] M. Lustig, D. Donoho, and J. M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[81] S. Mallat. A wavelet tour of signal processing, second ed. *Academic Press, London*, 1999.

[82] A. McCallum. Real-sim: Real vs. Simulated data for binary classification problem. `http://www.cs.umass.edu/~mccallum/code-data.html`.

[83] A. J. Miller. *Subset selection in regression*. Chapmain & Hall/CRC, London, 2002.

[84] J. J. Moreau. Proximité et dualité dans un espace Hilbertien. *Bull. Soc. Math. France*, 93:273–299, 1965.

[85] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[86] D. Needell and R. Ward. Stable image reconstruction using total variation minimization. *SIAM J. Imaging Sciences*, 6(2):1035–1058, 2013.

[87] Y. Nesterov. *Introductory Lecture Notes On Convex Optimization. A Basic Course*. Kluver, Boston, 2004.

[88] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2004.

[89] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[90] Y. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Math. Program. Ser. A*, 108(1):177–205, 2006.

[91] Yu. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[92] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2006.

[93] N. Parikh and S. Boyd. Proximal algorithms. *Journal Foundations and Trends in Optimization*, 1(3):123–231, 2013.

[94] Numerical recipes in C: the art of scientific computing. *M. C. Seiler and F. A. Seiler*. WH Press, 2012.

[95] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. MOS-SIAM Series on Optimization, Cornell University, Ithaca, New York, 2001.

[96] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. Technical report, School of Mathematics, Edinburgh University, 2012. Software available at `https://code.google.com/p/ac-dc/`.

[97] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1–2):1–38, 2014.

[98] R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics, Princeton University Press, 1970.

[99] M. Rudelson and R. Vershynin. On sparse reconstruction from Fourier and Gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

[100] M. Saunders and B. Kim. PDCO: Primal-dual interior method for convex objectives. *Technical Report, Stanford University*, 2002. `http://www.stanford.edu/group/SOL/software/pdco.html`.

[101] K. Scheinberg and X. Tang. Practical inexact proximal quasi-newton method with global complexity analysis. Technical report, March 2014. arXiv:1311.6547 [cs.LG].

[102] M. Schmidt. Graphical model structure learning with l1-regularization. *PhD thesis, University British Columbia*, 2010.

[103] S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12(4):1865–1892, 2011.

[104] S. Sra, S. Nowozin, and S. J. Wright. *Optimization for Machine Learning*. MIT Press, 2011.

[105] D. Takhar, J. N. Laska, M. Wakin, M. Duarte, D. Baron, S. Sarvotham, K. K. Kelly, and R. G. Baraniuk. A new camera architecture based on optical-domain compression. *In Proceedings of the IS&T/SPIE Symposium on Electronic Imaging: Computational Imaging*, 6065, 2006.

[106] A. Thomson. Compressive single-pixel imaging. *Proceedings of the 2nd IMA Conference on Mathematics in Defence, Defence Academy, Shrivenham, UK*, 2011.

[107] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Roy. Statist. Soc.*, 58(1):267–288, 1996.

[108] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[109] P. Tseng. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22:341–362, 2012.

[110] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Math. Program., Ser. B*, 117:387–423, 2009.

[111] S. Vaiter, G. Peyré, C. Dossal, and J. Fadili. Robust sparse analysis regularization. *IEEE Trans. Inf. Theory*, 59(4):2001–2016, 2013.

[112] S. Vattikuti, J. J. Lee, C. C. Chang, S. D. Hsu, and C. C. Chow. Applying compressed sensing to genome-wide association studies. *GigaScience*, 3(10):1–17, 2014.

[113] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[114] S. Webb, J. Caverlee, and C. Pu. Introducing the webb spam corpus: Using email spam to identify web spam automatically. *In Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[115] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation. *SIAM J. Sci. Comput.*, 32(4):1809–1831, 2010.

[116] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, 1997.

[117] S. J. Wright. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization*, 22(1):159–186, 2012.

[118] T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

[119] A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Fast l1-minimization algorithms and an application in robust face recognition: A review. *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1849–1852, 2010.

[120] J. Yoo, C. Turnes, E. B. Nakamura, C. K. Le, S. Becker, E. A. Sovero, M. B. Wakin, M. C. Grant, J. Romberg, A. Emami-Neyestanak, and E. Candes. A compressed sensing parameter extraction platform for radar pulse signal acquisition. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(3):626–638, 2012.

[121] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, J.-Y. Weng, E.-S. Yan, C.-W. Chang, T.-T. Kuo, Y.-C. Lo, P.-T. Chang, C. Po, C.-Y. Wang, Y.-H. Huang, C.-W. Hung, Y.-X. Ruan, Y.-S. Lin, S.-D. Lin, H.-T. Lin, and C.-J. Lin. Feature engineering and classifier ensemble for kdd cup 2010. *In JMLR Workshop and Conference Proceedings*, 2011.

[122] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale ℓ_1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

[123] R. K. P. Zia, E. F. Redish, and S. R. McKay. Making sense of the Legendre transform. *American Journal of Physics*, 77:614–622, 2009.