# Randomized Structure-Adaptive Optimization

*Junqi Tang*

A thesis submitted for the degree of Doctor of Philosophy.
**The University of Edinburgh**.
April 2019

# Abstract

This thesis advances the state-of-the-art of randomized optimization algorithms, to efficiently solve the large-scale composite optimization problems which appear increasingly more frequent in modern statistical machine learning and signal processing applications in this big-data era. It contributes from a special point of view, that the low-dimensional structure of the composite optimization problem's solution (such as sparsity, group-sparsity, piece-wise smoothness, or low-rank structure, etc), can be actively exploited by some purposefully tailored optimization algorithms to achieve even faster convergence rates – namely, the *structure-adaptive* algorithms. Driven by this motivation, several randomized optimization algorithms are designed and analyzed in this thesis. The proposed methods are provably equipped with the desirable structure-adaptive property, including the sketched gradient descent algorithms, the structure-adaptive variants of accelerated stochastic variance-reduced gradient descent and randomized coordinate descent algorithms. The thesis provides successful and inspiring paradigms for the algorithmic design of randomized structure-adaptive methods, confirming that the low-dimensional structure is indeed a promising "hidden treasure" to be exploited for accelerating large-scale optimization.

# Lay Summary

We are now living in the era of big-data. The engineers and computer scientists are often encountered with optimization tasks involving large-scale and high-dimensional datasets, especially in the fields of machine learning, signal processing and computational imaging. The commonly-used computational devices we have are usually limited in speed and storage, while the size of the data in real-world applications are growing explosively. In order to overcome this dilemma, researchers have been endeavoured to develop efficient optimization algorithms using randomization techniques. The randomized optimization algorithms typically either perform random accesses only to a small amount of the data, or compute a small-sized summary which includes enough information about the whole dataset to solve the original optimization task. Due to the efficiency in big-data applications, the randomized optimization algorithms are well-studied and widely-applied in machine learning and signal processing practice.

In this thesis, a new family of randomized optimization algorithms are proposed, which takes into account the fact that many real-world optimization tasks admit solutions which are structured, such as sparsity, piece-wise smoothness or low-rank. These algorithms are specifically tailored to exploit the structure of solutions and achieve further reductions in computational cost. The algorithmic improvements of the proposed algorithms are validated both theoretically and numerically, which shed lights to the research of next-generation optimization algorithms.

# Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Department of Electronics and Electrical Engineering at The University of Edinburgh.

Junqi Tang

# Acknowledgements

Firstly, I would like to thank my beloved supervisor Prof. Mike Davies, who has initialized this project, giving me the opportunity to start my research and walked alongside with me throughout these years of my PhD study and research. He is a great academic whom I admire deeply. I have learned so much from his broad knowledge and sharp insights.

I thank Dr. Mohammad Golbabaee, who has played a key role in my research when he was a research associate in Edinburgh. Since I started my PhD research project on optimization algorithms with very limited background knowledge, his guidance in the early stage as well as his tremendous effort in collaborating with me proved most essential. I would also like to thank Dr. Javier Escudero Rodriguez for his encouragements and helpful advices during these years.

I wish to thank Dr. Alessandro Perelli who has been a great colleague and wonderful friend. He is such a kind, modest, meek and loving person. Thank you Alex for being with me through the peaks and valleys in both work and life! Moreover, I thank my amazing colleagues and friends in my group and IDCOM: Jonathan Mason, Di Wu, Mikey Sheehan, Dongdong Chen, Heyi Li, Yetunde Oluwatayo, Arnold Benjamin, Wajiha Bano, Kimin Kim, Mo Soltani, etc, for the great time together.

I thank Prof. Francis Bach for hosting my scientific secondment in his SIERRA reearch team in INRIA, France. It was a great experience to collaborate with him on the structure-adaptive stochastic gradient project. His sharp insights and guidance have indeed become a great help. I would like to also thank Damien Scieur and Vincent Roulet for very helpful discussions during my visit. I am grateful for Prof. Karen Egiazarian for hosting my industrial secondment in Noiseless Imaging and it has also been a great experience. I wish to thank Alessandro Foi, Vladimir Katkovnik, Enrique Sanchez-Monge, Zhongwei Xu and Cristovao Cruz in Noiseless Imaging for very helpful discussions and companion in my industrial secondment.

I wish to give thanks to the researchers in the field of optimization I have discussed with and learned from, including Peter Richtarik, Mario Figueiredo, Robert Gower, Jakub Konecny, Dominik Csiba, Nicolas Loizou, Olivier Fercoq, Jingwei Liang, Renbo Zhao, Tao Lin, etc.

I wish to give thanks to my dear friends Jacob Crolla, Phil Holden and their families in Life-church Edinburgh for their love, companion, encouragements and prayers throughout these years.

I wish to thank my beloved brothers and flatmates – Chuojie Xian and Dingjian Xie. Their companion and friendship is no-doubt the highlight of my days in Edinburgh. I will never forget the time we make fun together, cook together, eat together, pray together, go to church and fellowship together. I would like to thank my dearest friends in the Chinese Evangelical Church of Edinburgh (Mandarin fellowship) – Zhenyu Wen, Qi Chen, Xuebin Liu, Enlin Chen, Yali Zhou, Jiandan Li, Zihan Chen, Jessica Huang, Zhongchao Ma, Zishan Du, Vicky Lee, Kelvin Yu and Pastor Alvin Kam Tong Lee, etc. This fellowship is my home away from home, with so much love.

I would like to thank my parents for their unconditional love and support as always.

I wish to thank Jesus for all the provisions, breakthroughs and victories.

# Contents

# List of figures

# List of tables

# Acronyms and abbreviations

| | |
|---|---|
| CS | Classical Sketch |
| IHS | Iterative Hessian Sketch |
| ROS | Randomized Orthogonal System |
| FFT | Fast Fourier Transform |
| FJLT | Fast Johnson-Lindenstrauss Transform |
| SJLT | Sparse Johnson-Lindenstrauss Transform |
| PGD | Projected Gradient Descent |
| GPIS | Gradient Projection Iterative Sketch |
| Acc-GPIS | Accelerated Gradient Projection Iterative Sketch |
| LS | Least-Squares |
| TV | Total Variation |
| LASSO | Least Absolute Shrinkage and Selection Operator |
| SGD | Stochastic Gradient Descent |
| SAG | Stochastic Averaged Gradient |
| SVRG | Stochastic Variance-Reduced Gradient Descent |
| SDCA | Stochastic Dual Coordinate Ascent |
| RCD | Randomized Coordinate Descent |
| APPROX | Accelerated Parallel and Proximal Coordinate Descent |
| APCG | Accelerated Proximal Coordinate Gradient |
| RSC | Restricted Strong-Convexity |
| RSM | Restricted Smoothness |
| FISTA | Fast Iterative Shrinkage and Thresholding Algorithm |
| ADMM | Alternating Direction Method of Multipliers |
| PDHG | Primal-Dual Hybrid Gradient |
| SPDHG | Stochastic Primal-Dual Hybrid Gradient |
| Acc-PD-SGD | Accelerated Primal-Dual Stochastic Gradient Descent |

# Nomenclature

| | |
|---|---|
| $\langle \cdot, \cdot \rangle$ | Inner product |
| $(\cdot)^T$ | Transpose |
| $\mathbb{E}(\cdot)$ | Expectation |
| $\mathbb{R}^d$ | $d$-dimensional Euclidean space |
| $\Gamma_0(\mathbb{R}^d)$ | The family of lower semi-continuous convex functions which maps $\mathbb{R}^d \to \mathbb{R}$ |
| $\|\| \cdot \|\|_0$ | $\ell_0$ semi-norm |
| $\|\| \cdot \|\|_1$ | $\ell_1$-norm |
| $\|\| \cdot \|\|_2$ | $\ell_2$-norm |
| $\|\| \cdot \|\|_{1,2}$ | $\ell_{1,2}$-norm |
| $\|\| \cdot \|\|_\star$ | Nuclear norm |
| $\|\| \cdot \|\|_A$ | Semi-norm associated with a linear operator $A$ |
| $\mathcal{B}^d$ | Unit ball in $\mathbb{R}^d$ centered at the origin |
| $\mathbb{S}^{d-1}$ | Unit sphere in $\mathbb{R}^d$ centered at the origin |
| $\mathcal{N}(\cdot, \cdot)$ | Normal distribution |
| $\nabla f(\cdot)$ | Gradient operator of $f$ |
| $\partial f(\cdot)$ | Sub-differential operator of $f$ |
| $x^\star$ | Optimal solution of an (regularized) empirical risk objective |
| $x^\dagger$ | The minimizer of an excess risk / ground truth vector to be estimated |
| $\mathcal{F}$ | The class of convex functions |
| $\lambda$ | Regularization parameter |
| $\mathcal{M}$ | Model subspace |
| $\mathcal{M}^\perp$ | Perturbation subspace |
| $f^*$ | Convex conjugate of the function $f$ |
| $\iota_{\mathcal{K}}(\cdot)$ | Indicator function of a constraint set $\mathcal{K}$ |
| $\mathcal{P}_{\mathcal{K}}(\cdot)$ | Orthogonal projection towards a constraint set $\mathcal{K}$ |
| $\mathrm{nnz}(\cdot)$ | Number of non-zero elements |

# Chapter 1
# Introduction

## 1.1 Optimization with Big-Data

We are now in an era of boosting knowledge and large data with various statistical machine learning and signal/image processing applications which involve the problem of tackling a huge amount of data. These applications occur in all sorts of fields associated with data science, vary from genetic engineering, to medical imaging such as the computed tomography (CT) and magnetic resonance imaging (MRI), marketing, computer vision, natural language processing, robotics, and in machine learning problems where we need to train classifiers or make predictions from a large amount of data samples. Many of these applications involve solving optimization problems with large scales. The term "*optimization*" in this context, essentially implies that it is a computational action of making the best possible utilization of the given information carried by the available data, in order to provide a good solution to real world problems.

Let us start with a simple motivating example in Figure 1.1 of how (supervised) machine learning actually works and understand why optimization plays a key role in machine learning. Suppose we wish to make a software which can enable our computers to classify between two (or multiple) classes of images, for example, images of two men – Billy (class 1, blue dots) and Luke (class 2, red dots). Assume that Billy and Luke look different enough such that the underlying statistics of their photos is distinguishable. The blue dots and red dots in figure 1.1 live in a high-dimensional feature space – if we use the raw image directly, it is in the order of millions of dimensions. Now, suppose we have been given some examples of Billy's and Luke's images as training data, and meanwhile we model our classifier and restrict it to be a hyper-plane (black line) in the same dimension. Then, to capture the statistics behind the training data, we need to optimize the position of the hyper-plane to make it classify reasonably well on the given examples of images[1]. Generally, when the number of training examples are large

[1]together with some additional modelling (regularization) to avoid overfitting the data

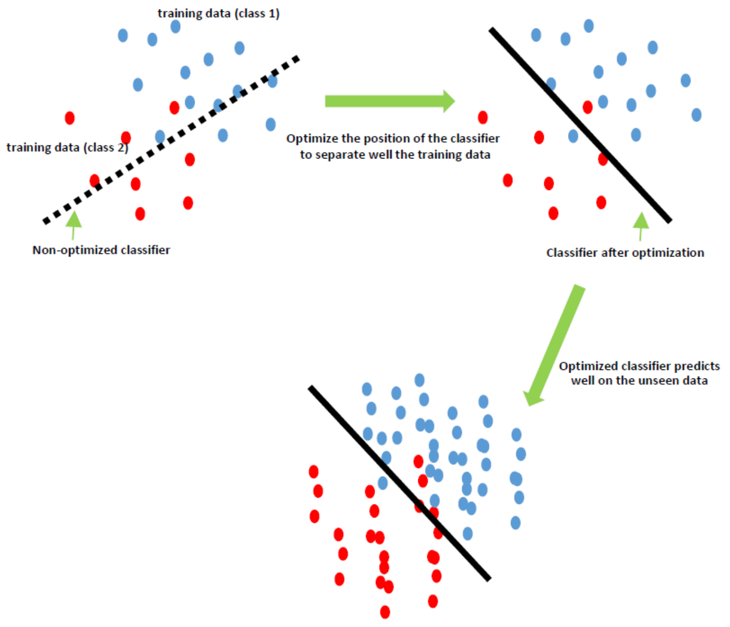
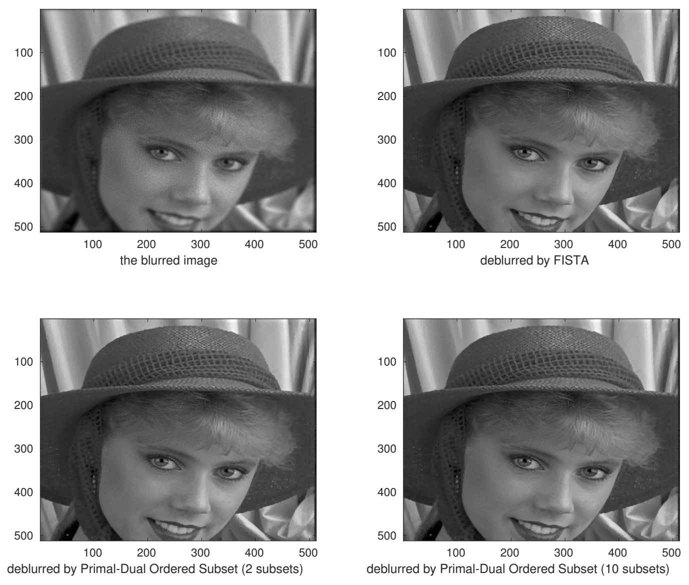

training data (class 1)
training data (class 2)
Non-optimized classifier
Optimize the position of the classifier to separate well the training data
Classifier after optimization
Optimized classifier predicts well on the unseen data

**Figure 1.1:** *Supervised machine learning via optimization.*

enough, the optimized classifier will typically classify with a fairly good accuracy on unseen photos.

From this example we can clearly see that, the core computational step in supervised machine learning is the optimization of the high-dimensional classifier over the training data. In modern machine learning practice, both of the number of training data and feature dimension can be huge. In a large data setting a desirable optimization algorithm should be able to simultaneously address good accuracy of the solutions and small computational costs. Similar to the real-world applications of statistical machine learning, large-scale optimization also plays a central role in signal and image processing, such as the reconstruction of medical images for the inner beings of human bodies via X-ray measurements (Computed Tomography) or magnetic resonance technology (MRI), the denoising and deblurring of images taken by digital cameras, superresolution, etc, where we typically attempt to recover or reconstruct clean signals or images from observations and measurement data. Just like machine learning, the recovering of a signal or image can also be cast as an optimization program where we aim to minimize the error we make to fit the given observation. Meanwhile if the measurement data is not informative enough because of the lack of quantity or quality, to infer a clean signal or image, one needs to encode into the optimization program manually and mathematically the prior information



the blurred image
deblurred by FISTA
deblurred by Primal-Dual Ordered Subset (2 subsets)
deblurred by Primal-Dual Ordered Subset (10 subsets)

**Figure 1.2:** *The blurred image, and the deblurred images by some optimization algorithms*

of the clean signal or image – for example, the image of a real-world object usually has sharp edges and smooth content. In a high level, the optimization in modern machine learning and signal/image processing can generally be summarized as finding a solution vector $x^\star$ which minimizes a composite loss function:

$$(\text{Data fidelity error}) + (\text{Model fitting error})$$

where the first term ensures the outcome of the optimization program is derived based on making the best usage of the information provided by given data, while the second term for "model fitting" ensures the prior information is also exploited for deriving an improved result in practice – in machine learning, it typically leads us to a classifier which works even better on unseen data (formally we say that it has better "*generalization*" ability); while in signal/image processing, it leads us to a better estimate of the unseen ground truth. This model fitting term is formally named as *"regularization"*.

The optimization problems in modern machine learning and signal/image processing practice are very often large-scale and high-dimensional. For instance, the standard image-classification benchmark dataset ImageNet [1] has a size of $1.31$ TB, including 14 millions of real-world images of various classes of objects, while each of image has millions of colored pixels. It is exceedingly challenging for usual computational devices to obtain practically-relevant solutions from training any of state-of-art models from this type of datasets in a reasonable amount of time. In the field of medical imaging, we often need to perform tomographic reconstruction of 3D images or even 4D videos of the inner-being of a patient in a high resolution in order to ensure the clinicians to make good diagnoses. The resulting optimization task can have a very huge dimension by nature if, for example, the 3D image we wish to reconstruct is of a standard size $1024 \times 1024 \times 1024$, while the linear measurement data we obtain is of the same order of size of the image dimension. In clinical practice, due to the forbidding size of the optimization problem and often very limited computational resources, we often need to restrain from solving the optimization problem in full and hence compromise the quality of the reconstructed images, in order to perform real-time diagnoses.

In view of these significant challenges in the real-world big-data optimization, this thesis is pursuing a novel approach for the design of large-scale optimization algorithms.

### 1.1.1 Contributions

In this thesis we present a novel family of randomized first-order algorithms for efficiently solving large-scale composite optimization tasks which occur increasingly at the core of modern machine learning, data science, computer vision and signal processing applications. Our algorithms are particularly designed under the principle of exploiting the solution's low-dimensional structure (such as sparsity, low-rank, and piece-wise smoothness) enforced by the regularization or constraints as the source for computational speed-up – this is exactly the key novelty of this thesis. In the stochastic optimization literature, although numerous stochastic iterative gradient-based algorithms have been proposed in recent years for large-scale convex/non-convex optimization, very little research have been done on understanding and utilizing the effect of the solution's structure towards the convergence speed of an iterative algorithm. This thesis records a detailed research to fill in this gap of understanding formally, and provides both theoretical and algorithmic contributions with several successful structure-adaptive algorithmic-design paradigms which advance the state-of-the-art in various scenarios.

This thesis contains a background chapter (chapter 2), four technical chapters (chapter 3 - 6) and a conclusion chapter (chapter 7). Let's now have a brief overview on the contribution of each technical chapter:

**Chapter 3: Sketched Gradient Algorithms for Large-Scale Constrained Optimization.**

In this chapter we present a class of iterative optimization algorithms, which we call *Sketched Gradient* methods, for efficiently solving large-scale constrained least-squares, based on the combination of projected gradient descent and least-squares sketching (randomized projection) techniques [2, 3]. In scenarios where the constraint set enforces the least-squares solution to have low-dimensional structure, one can apply randomized projection to reduce the dimension of the large-scale problem, and hence reduce the computation cost of (projected) gradient descent by exploiting the solution's intrinsic structure. We show theoretically that the sketched gradient algorithms indeed have convergence rates which scale with the solution's statistical complexity. In numerical experiments, the proposed sketching-based algorithm demonstrates superior performance over the state-of-the-art stochastic variance-reduced gradient methods on constrained lasso regression and multivariate low-rank matrix regression tasks in practical large-minibatch training settings.

**Chapter 4: Structure-Adaptive Accelerated Stochastic Gradient Descent**

In this chapter, we study the potential structure-adaptiveness of stochastic gradient descent algorithms with momentum for acceleration, and propose an algorithm named *Rest-Katyusha* for regularized empirical risk minimization. The proposed method, being an improved variant of a state-of-the-art stochastic variance-reduced gradient method Katyusha by Allen-Zhu [4], is provably able to exploit the low-dimensional structure of the solution for even faster convergence. This algorithmic improvement is achieved by restarting the Katyusha algorithm with a period proportional to the statistical complexity of the solution, which has been formally expressed as the restricted strong-convexity (RSC) by [5]. We also propose an adaptive variant of the Rest-Katyusha which is able to estimate the statistical complexity of the solution on the fly, and demonstrate the effectiveness of this approach in sparse regression tasks on various machine learning datasets.

**Chapter 5: Structure-Adaptive Accelerated Coordinate Descent**

This chapter shares the same spirit of chapter 4, where we extend the restart-based algorithmic structure and the structure-driven theoretical analysis, to advance the state-of-the-art of a different class of randomized optimization algorithms – randomized block coordinate descent methods. We propose adaptive-restart variants of the *accelerated proximal coordinate gradient* (APCG) algorithm proposed by [6] which can exploit the solution's structure and accelerate the convergence speed on separable composite convex optimization problems which have block-coordinate-wise smoothness on the data-fidelity term.

**Chapter 6: the Limitation and Practical Acceleration Strategies of Stochatic Gradient Methods in Inverse Problems**

In this chapter we investigate the practicability of stochastic gradient descent and recently introduced variants with variance-reduction techniques in imaging inverse problems, such as non-uniform image deblurring. Such algorithms have been shown in machine learning literature to have optimal complexities in theory, and provide great improvement empirically over the full gradient methods. Surprisingly, in some tasks such as image deblurring, many of such methods fail to converge faster than the accelerated full gradient method (FISTA), even in terms of epoch counts. We investigate this phenomenon and propose a theory-inspired mechanism to characterize whether a given inverse problem should be preferred to be solved by the stochastic optimization techniques. Furthermore, to overcome another key bottleneck of stochastic optimization which is the potentially heavy computation of proximal operators while maintaining fast convergence, we propose an accelerated primal-dual SGD algorithm and demonstrate the effectiveness of our approach in image deblurring and tomographic image reconstruction experiments. Although unlike the previous technical chapters, the chapter 6 is mainly about empirical study, it lays foundations and provides key insights for the future work – the structure-adaptive algorithmic design tailored specifically for the imaging inverse problems.

### 1.1.2 Publications

This thesis is based on the following peer-reviewed publications [7, 8, 9, 10] and preprints [11, 12] during my PhD study:

**Chapter 3:**

- Junqi Tang, Mohammad Golbabaee, Mike Davies. “Gradient Projection Iterative Sketch for Large-Scale Constrained Least-Squares”, in *Proc. of 34th International Conference on Machine Learning* (ICML), 2017.
- Junqi Tang, Mohammad Golbabaee, Mike Davies. “ Exploiting the Structure via Sketched Gradient Algorithms”, in *Proc. of 5th IEEE Global Conference on Signal and Information Processing* (GlobalSIP), 2017.

**Chapter 4:**

- Junqi Tang, Mohammad Golbabaee, Francis Bach, Mike Davies. “Rest-Katyusha: Exploiting the Solution’s Structure via Scheduled Restart Schemes”, in *Advances in Neural Information Processing Systems* (NeurIPS), 2018.

**Chapter 5:**

- Junqi Tang, Mohammad Golbabaee, Francis Bach, Mike Davies. “Structure-Adaptive Accelerated Coordinate Descent”, *Submitted for publication. hal-01889990v2*, 2018.

**Chapter 6:**

- Junqi Tang, Karen Egiazarian, Mike Davies. “The Limitation and Practical Acceleration of Stochastic Gradient Algorithms in Inverse Problems”, in *Proc. of 44th International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), 2019.

# Chapter 2

# Background

## 2.1 Convex Optimization in Learning and Estimation

Many applications in supervised machine learning and signal processing share the same goal, which is to estimate the minimizer of a population risk via minimizing the empirical risk [13]. To be more specific, for given $n$ samples $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n) \in \mathbb{R}^d \times \mathbb{R}$ from some marginal distribution $\rho$, we define the empirical risk function as:

$$f(x) := \frac{1}{n}\sum_{i=1}^{n} \bar{f}(a_i, b_i, x) \tag{2.1}$$

where $a_i$, $x \in \mathbb{R}^d$, $b_i \in \mathbb{R}$, and $\bar{f} : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a convex function. In supervised machine learning, $a_i$ is often referred to as the training data sample while $b_i$ is the corresponding label. In signal/image processing applications they may be the representation of the observations and measurements. In practice the number of data samples or measurements is limited, and from them we attempt to infer $x^\dagger \in \mathbb{R}^d$ which is the unique minimizer of the population risk:

$$x^\dagger = \arg\min_x \mathbb{E}_{(a,b)}\bar{f}(a, b, x) := \int \bar{f}(a, b, x) d\rho(a, b). \tag{2.2}$$

where the expectation is taken over all possible pairs of $(a, b) \in \mathbb{R}^d \times \mathbb{R}$ drawn from the underlying distribution $\rho$. The ultimate goal is to get a vector $x^\star$ which is a good approximation of $x^\dagger$ from the empirical risk, in order to generalize well on unseen data. Since in many interesting applications, the dimension of parameter space $d$ is of the same order or even larger than the number of data samples $n$, minimizing the empirical risk alone will introduce overfitting and hence leads to poor estimation of the true parameter $x^\dagger$ [14, 15, 16]. In general, avoiding overfitting is a key issue in both machine learning and signal processing, and the most common approach is to add some regularization while minimizing the empirical risk [17, 18] (for the sake of compactness of notations, we denote $f_i(x) := \bar{f}(a_i, b_i, x)$ for the rest of the thesis):

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} \left\{ F(x) := f(x) + \lambda g(x) \right\}, \quad f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{2.3}$$

Each $f_i$ is assumed to be convex and Lipschitz continuous, while the regularization term $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a simple lower semi-continuous convex function and possibly non-smooth. Let $y \in \mathbb{R}^n$ being the target vector such that $y = [b_1, b_2, ..., b_n]^T$ – for classification tasks in supervised machine learning, it is consisted of the labels for each training data point. We illustrate some popular examples of the composite optimization problem (2.3):

**Lasso regression.** Let $f(x) := \frac{1}{n}\sum_{i=1}^{n}(a_i^T x - b_i)^2 = \frac{1}{n}\|Ax - y\|_2^2$ and $g(x) = \|x\|_1$ in (2.3), then we get the *Lasso* [16, 17] objective:

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} \frac{1}{n}\|Ax - y\|_2^2 + \lambda\|x\|_1, \tag{2.4}$$

which is essentially a sparse least-squares regression task. With sufficiently large regularization parameter $\lambda$, the $\ell_1$ norm penalty will enforce sparse solutions of (2.4). In signal/image processing and compressed sensing, the Lasso and its variants are classic tools for recovering sparse or approximately sparse signals from noisy and incomplete/insufficient measurements, with solid recovery guarantees. In machine learning, the Lasso often serves as a tool for feature selection and model compression for better generalization.

**Group-Lasso.** Denote that $x$ consists of $q$ sub-vectors: $x = [x_{(1)}, x_{(2)}, ....x_{(q)}]$. Let $f(x) := \frac{1}{n}\sum_{i=1}^{n}(a_i^T x - b_i)^2 = \frac{1}{n}\|Ax - y\|_2^2$, and $g(x) = \sum_{j=1}^{q}\|x_{(j)}\|_2$ being a $\ell_{2,1}$ regularization in (2.3), then we get the *Group-Lasso* objective:

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} \frac{1}{n}\|Ax - y\|_2^2 + \lambda\sum_{j=1}^{q}\|x_{(j)}\|_2. \tag{2.5}$$

The $\ell_{2,1}$ regularization enforces group sparsity on the solution $x^\star$ [19].

**Logistic regression.** Let $f(x) := \sum_{i=1}^{n}\log(1 + \exp(-b_i a_i^T x))$ in (2.3), we obtain the class of logistic regression which is especially tailored to binary classification tasks in machine learning where each label $b_i$ is set to be either $-1$ or $1$. In practice, the logistic data-fidelity term is often optimized with additional regularization such as $\ell_1$ or $\ell_2$ norms, to avoid overfitting for better generalization and/or model compression [20].

**Low-rank multivariate regression.** The training of multiple classifiers takes a more general matrix form of (2.3). For example, let $X \in \mathbb{R}^{d_1 \times d_2}$ where each column denotes a classifier to a class ($d_2$ classes in total), $Y \in \mathbb{R}^{n \times d_2}$ being the label matrix:

$$X^\star = \arg \min_{X \in \mathbb{R}^{d_1 \times d_2}} |||Y - AX|||_F^2 + \lambda \|X\|_\star, \tag{2.6}$$

where $\|.\|_\star$ denotes the nuclear norm.

The nuclear-norm regularization which enforces low-rank structure, has been shown to be effective in multi-classification tasks in distinguishing objects with high similarities [21, 22].

**Support vector machine.** Let the data-fidelity term be the hinge-loss:

$$f(x) := \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - b_i \langle \phi(a_i), x \rangle), \tag{2.7}$$

with $b_i$ being the labels and a non-linear function $\phi(.)$ which maps the data points to a higher (possibly infinite dimensional) feature space in order to train a non-linear classifier, we recover the objective of the (kernel) support vector machine (SVM) [13] for classification. Most commonly used regularization for SVMs include an $\ell_1$ and $\ell_2$ penalty. In the case of high-dimensions where $d \gg n$, the $\ell_1$-SVM is preferred [23].

It is worth noting that for high-dimensional statistical machine learning, an interesting *"bet on sparsity"* principle has been proposed in [24] which vindicates the superiority of $\ell_1$ regularization over the $\ell_2$ ridge regression penalty. In short, intuitively, in high-dimensional case $d \gg n$, if the true parameter $x^\dagger$ is dense, both of these regularizations are poor in performance since the training data is insufficient for the huge dimension. However, if the true parameter $x^\dagger$ is sparse, then $\ell_1$ will have superior performance.

**Constrained optimization.** In practice, one may have specific prior knowledge about the solution $x^\dagger$ such that the desired output $x^\star$ of the optimization problem should satisfy certain constraints. A constraint can also be regarded as a regularization, with $g(.) := \iota_{\mathcal{K}}(.)$, where

$$\iota_{\mathcal{K}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{K}. \\ +\infty & \text{if } x \notin \mathcal{K}. \end{cases} \tag{2.8}$$

being the indicator function of a constraint set $\mathcal{K}$. For example, let $f(x)$ being the least-squares loss $\|Ax - y\|_2^2$, we obtain the *constrained least-squares* regression objective [17]:

$$x^\star \in \arg\min_{x \in \mathbb{R}^d} \|Ax - y\|_2^2 + \iota_{\mathcal{K}}(x) = \arg\min_{x \in \mathcal{K}} \|Ax - y\|_2^2. \tag{2.9}$$

**Total-variation regularized least-squares.** Let $f(x)$ be a least-squares loss and $g(x) = \|Dx\|_1$ being the Total-Variation (TV) regularization [25, 26], where $D$ is a discrete gradient operator, we obtain the TV regularized Least-squares objective:

$$x^\star \in \arg\min_{x \in \mathbb{R}^d} \frac{1}{n}\|Ax - y\|_2^2 + \lambda\|Dx\|_1 \tag{2.10}$$

For TV regularization in 1-D, the linear operator $D \in \mathbb{R}^{(d-1)\times d}$ has all the diagonal elements $D_{j,j} = 1$ and $D_{j,j+1} = -1$ ($\forall 1 \leq j \leq d-1$), while the rest of the elements are all zeros. The TV regularization enforces piece-wise smoothness structure on the solution vector $x^\star$. It has been widely applied in many fields of statistical machine learning and signal processing [18], and is particularly popular in imaging applications [27], since a real-world image typically has smooth contents and sharp edges.

## 2.2 Convexity, Strong-Convexity and Smoothness

Before the introduction of classic convex optimization algorithms, we first formally describe and recall from the literature the aforementioned key notions of convexity and smoothness (see e.g. [28]) of the objective function which we wish to optimize. For the sake of simplicity and clarity we assume the function $f(.)$ is differentiable in this section.

**Convexity.** A differential function $f$ is convex if:

$$f(x) - f(x') - \langle \nabla f(x'), x - x' \rangle \geq 0, \ \forall x, x' \in \mathbb{R}^d, \tag{2.11}$$

which essentially means $f(x)$ is lower bounded by linear function $f(x') + \langle \nabla f(x'), x - x' \rangle$ with any possible $x' \in \mathbb{R}^d$. It is well-known that if a function is convex, any local minimum of $f$ is global minimum [28, Proposition 1.2].

**Strong-Convexity.** A differential function $f$ is $\mu$-strongly-convex if:

$$f(x) - f(x') - \langle \nabla f(x'), x - x' \rangle \geq \frac{\mu}{2}\|x - x'\|_2^2, \ \forall x, x' \in \mathbb{R}^d, \tag{2.12}$$

which means geometrically the objective function $f(x)$ is lower bounded by quadratic function $f(x')+\langle \nabla f(x'), x-x'\rangle+\frac{\mu}{2}\|x-x'\|_2^2$, for any $x'$. If a function is strongly-convex, its minimizer is not only global but also unique.

**Smoothness.** A differential function $f$ is $L$-smooth if:

$$f(x) - f(x') - \langle \nabla f(x'), x - x' \rangle \leq \frac{L}{2}\|x - x'\|_2^2, \ \forall x, x' \in \mathbb{R}^d. \tag{2.13}$$

In other words, the function $f$ is upper bounded by any quadratic function $f(x')+\langle \nabla f(x'), x-x'\rangle + \frac{L}{2}\|x - x'\|_2^2$ with any $x'$. More specifically, $L$-smoothness implies that the gradient of the function $f$ is $L$-Lipschitz-continuous:

$$\|\nabla f(x) - \nabla f(x')\|_2^2 \leq L\|x - x'\|_2^2, \ \forall x, x' \in \mathbb{R}^d, \tag{2.14}$$

which intuitively means that the gradient changes slowly between nearby points.

Generally speaking, minimizing an arbitrary continuous function is a difficult task, but fortunately a large-variety of machine learning and signal processing tasks can be either directly-formulated, reformulated, or approximated as a well-behaved problem which is convex (or strongly-convex) and smooth.

Moreover, note that in terms of the regularization function $g(.)$, we only consider the choices of lower semi-continuous convex functions [29] in this thesis:

**Lower Semi-Continuity.** A function $g(.)$ is called lower semi-continuous on $\mathcal{X}$ if:

$$g(x) \leq \liminf_{y \to x} g(y), \forall x \in \mathcal{X}, \tag{2.15}$$

where the notation $\lim\inf$ denotes the limit-inferior. The inequality (2.15) essentially means that for any $x \in \mathcal{X}$, the function value $g(x)$ is no greater than the smallest possible $g(y)$ where $\|x - y\|_2 \to 0$. All the regularizers (including the indicator function of a convex set) we have introduced so far in the previous section are lower semi-continuous functions. We denote this class of functions as $\Gamma_0(.)$ throughout this thesis.

In this thesis we will discuss in various places on the convergence rates of algorithms. We now provide the definition of the linear convergence rate.

**Linear Convergence.** Let $x^\star$ be one of the optima of $F(\cdot)$. Suppose an iterative algorithm starts from initial point $x^0$ and generates a sequence $x^1, x^2, ..., x^K$. If for any $k \in [1, 2, ..., K]$,

$$F(x^k) - F(x^\star) \leq Cq^k[F(x^0) - F(x^\star)], \tag{2.16}$$

for $q \in (0, 1)$ and $C$ being a position constant, then we say that the algorithm is guaranteed to have a linear convergence rate. If (2.16) is not satisfied, the algorithm is sublinearly convergent.

## 2.3 First-Order Algorithms for Convex Composite Optimization

In this section we introduce first-order algorithms which are based on only the gradient information for solving the composite optimization task (2.3) in Section 2.1. We denote the strong-convexity parameter and the smoothness parameter of the function $f$ as $\mu_f$ and $L_f$ respectively:

$$\frac{\mu_f}{2}\|x - x'\|_2^2 \leq f(x) - f(x') - \langle \nabla f(x'), x - x' \rangle \leq \frac{L_f}{2}\|x - x'\|_2^2, \ \forall x, x' \in \mathbb{R}^d. \tag{2.17}$$

### 2.3.1 Gradient Descent

The most simple first-order method for (2.3) is the (sub-)gradient descent [30]. We define the subgradient[1] of the function $F$ as:

$$\partial F : x \to \{z \in \mathbb{R}^d \mid (y - x)^T z + F(x) \leq F(y), \forall y \in \mathbb{R}^d\}. \tag{2.18}$$

The (sub-) gradient descent algorithm simply takes the following form:

```
**Gradient Descent** − Initialize $x^0 \in \mathbb{R}^d$
For   $k = 0, 1, 2, ..., K$
    $x^{k+1} = x^k - \eta_k \mathcal{G}_k,\ \mathcal{G}_k \in \partial F(x^k) := \nabla f(x^k) + \lambda \partial g(x^k)$
```

[1]Although $f(x)$ may be smooth the composite cost $F(x)$ may not, hence we need to define the subgradients.

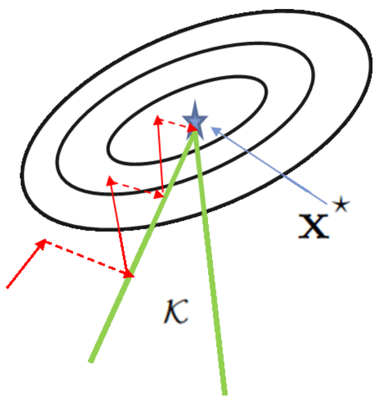


**Figure 2.1:** *Projected gradient descent algorithm for constrained optimization : red solid arrows denote the gradient descent step on the smooth part $f$, while the red dashed arrows denote the projection operation on the convex set $\mathcal{K}$.*

where $\eta_k$ being the sequence of step-size.

With minor assumptions such as Lipschitz continuity of $F$, one can show that with step-size $\eta_k = O(1/k)$, the (sub-)gradient descent has a convergence rate of $F(x^k) - F^\star \leq O(1/\sqrt{k})$, see e.g. [31, Theorem 3.2.2]. If the regularization term $g(.)$ is $L_g$-smooth, then $F$ is $(L_f+\lambda L_g)$-smooth, $\partial F = \nabla F = \nabla f + \lambda \nabla g$, and one can show that with step-size $\eta_k = \frac{1}{L_f+\lambda L_g}$ the gradient descent has a convergence rate of $F(x^k) - F^\star \leq O(1/k)$, see e.g. [28, Theorem 3.3]. We can clearly see that the main drawback of this naive gradient descent in composite optimization is that, as long as one of the function in the composite task (2.3) is non-smooth, it cannot benefit from the smoothness from the smooth part for faster convergence. This key flaw of (sub-)gradient descent motivates the developments of the proximal gradient methods.

### 2.3.2 The Projection and Proximal Operator for Faster Composite Optimization

In composite optimization with a smooth data-fidelity term and a non-smooth regularizer, the (sub-)gradient descent fails to converge fast since it treats the optimization objective as a whole and is blind to the partial smoothness of it. Intuitively, if an iterative algorithm can deal with each part separately, it may have chance to benefit from the smoothness and achieve a faster convergence rate. Let us start with a simple special case of (2.3) with the regularization $g(x)$ being an indicator function of a convex set $\mathcal{K}$:

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} \left\{ F(x) := f(x) + \iota_{\mathcal{K}}(x) \right\}, \tag{2.19}$$

which can be equivalently written simply as minimizing the smooth function $f$ within the convex set $\mathcal{K}$:

$$x^\star \in \arg\min_{x\in\mathcal{K}} f(x). \tag{2.20}$$

The most simple iterative scheme one can think of for solving (2.20) is that, at each iteration, a gradient descent step on $f$ is first performed with step-size $\eta = \frac{1}{L_f}$:

$$x^+ = x^t - \eta\nabla f(x^t), \tag{2.21}$$

and if $x^+$ is outside the feasible regime marked by $\mathcal{K}$, we *project* it back to the set in order to make it remain feasible. A reasonable choice is the orthogonal projection since it is desirable that we pick the closest point of $x^+$ such that while ensuring feasibility, the update does not drift too much from the steepest descent direction given by the gradient of $f$. Let us formally define the projection operator:

**Definition 2.3.1.** *(Projection Operator.) We define $\mathcal{P}_\mathcal{K}(x) : \mathbb{R}^d \to \mathbb{R}^d$ the projection operator of the convex set $\mathcal{K}$ at a point $x \in \mathbb{R}^d$ as:*

$$\mathcal{P}_\mathcal{K}(x) := \arg\min_{y\in\mathbb{R}^d} \iota_\mathcal{K}(y) + \frac{1}{2}\|x-y\|_2^2. \tag{2.22}$$

With the definition of the projection, we can write the *projected gradient descent* (PGD) algorithm (see e.g [32]) for (2.20):

```
**PGD** – Initialize $x_0 \in \mathbb{R}^d$
For   $k = 0, 1, 2, ..., K$
    $x^{k+1} = \mathcal{P}_\mathcal{K}[x^k - \eta\nabla f(x^k)]$
```

Thanks to the splitting scheme, with step size $\eta = \frac{1}{L_f}$, the PGD algorithm enjoys a convergence rate of $F(x^k) - F^\star \leq O(1/k)$ – the same rate of running gradient descent on completely smooth functions. We visually describe the dynamic of running projected gradient descent on a constrained optimization problem in Figure 2.1.

Motivated by the fact that the indicator function $\iota_\mathcal{K}(.)$ is only a special case of the class of lower semi-continuous convex functions $\Gamma_0(\mathbb{R}^d)$, Moreau [33] generalized the projection operator to the case of an arbitrary function in $\Gamma_0(\mathbb{R}^d)$. This extension of the projection operator is named the *proximal operator*:

**Definition 2.3.2.** *(Proximal Operator.) We define* $\text{prox}_g(x) : \mathbb{R}^d \to \mathbb{R}^d$ *the proximal operator of the lower semi-continuous convex function* $g$ *at a point* $x \in \mathbb{R}^d$ *as:*

$$\text{prox}_g(x) := \arg\min_{y \in \mathbb{R}^d} g(x) + \frac{1}{2}\|x - y\|_2^2. \tag{2.23}$$

In the next subsection we shall introduce the proximal gradient descent – a fundamental splitting scheme with the combination of gradient descent and generalized projection (proximal operator) for solving the generic composite optimzation task (2.3) in Section 2.1.

## 2.3.3 Proximal Gradient Descent

The *proximal gradient descent* (Prox-GD) [34] – also known as the forward-backward splitting in the literature, takes the following form :

```
**Prox-GD** – Initialize $x_0 \in \mathbb{R}^d$
For  $k = 0, 1, 2, ..., K$
    $x^{k+1} = \text{prox}_{\lambda g}^{\eta}[x^k - \eta \nabla f(x^k)]$
```

where $\eta$ denotes the step size and a typical choice is $\eta = \frac{1}{L_f}$. We denote the proximal operator as the following form:

$$\text{prox}_{\lambda g}^{\eta}(\cdot) = \arg\min_{x \in \mathbb{R}^d} \frac{1}{2\eta}\|x - \cdot\|_2^2 + \lambda g(x). \tag{2.24}$$

At each iteration of proximal gradient descent, a gradient at point $x^k$ is calculated. Since this algorithm only uses the gradient information – a first-order oracle, to perform the updates, it is considered as a "first-order" optimization algorithm. At iteration $k$, with step size $\eta = \frac{1}{L_f}$, the update $x^k$ satisfies (see e.g. [35, Theorem 3.1]):

$$F(x^k) - F^\star \leq O\left(\frac{L_f\|x^0 - x^\star\|_2^2}{k}\right), \tag{2.25}$$

for convex case $\mu_f \geq 0$. If $f(.)$ is strongly-convex, which means $\mu_f > 0$, a linear convergence guarantee can be obtained [31]:

$$F(x^k) - F^\star \leq \left(1 - \frac{\mu_f}{L_f}\right)^k [F(x^0) - F^\star], \tag{2.26}$$

One important attribute which makes the first-order methods superior to the higher-order methods such as the proximal Newton method [36] is that the computational cost of them is linear with respect to the dimension $d$, while it is usually $d^2$ or $d^3$ for the higher-order methods. The convergence rate of the proximal Newton method is super-linear only locally and requires additional assumptions on higher-order smoothness. Meanwhile, the proximal gradient methods provide us an efficient way to handle the non-smooth regularization, which make the first-order methods even more highly favored in large-scale composite optimization.

### 2.3.4 The Acceleration of Proximal Gradient Descent

In this subsection we introduce the acceleration trick of gradient descent. With this trick, the convergence rate of original gradient descent scheme can be significantly improved. This phenomenon was first reported by Nesterov [37] in 1983, in which he proposed the very first accelerated gradient method for solving smooth convex problems, with a convergence rate $O(1/k^2)$ instead of $O(1/k)$ which is for the original gradient descent scheme. If the objective $F$ is a convex and smooth function, Nesterov's accelerated gradient descent scheme can be written as:

```
**Accelerated GD** – Initialize $x^0 \in \mathbb{R}^d$
For $k = 0, 1, 2, ..., K$
    $y^k = x^k - \eta \nabla F(x^k)$;  $\rightarrow$ gradient descent step
    $x^{k+1} = y^k + \beta(y^k - y^{k-1})$; $\rightarrow$ momentum step
```

The only difference between the acceleration and ordinary scheme of gradient descent is an additional extrapolation step for momentum. Researchers in the field of optimization commonly refer this type of scheme as *Nesterov's acceleration*. Moreover, Nesterov also demonstrated that such an accelerated gradient scheme with a convergence-rate $F(x^k) - F^\star \leq O(1/k^2)$ is worse-case optimal for convex and smooth first-order optimization. We first define formally the function class notation:

**Definition 2.3.3.** *For an Euclidean vector space $\mathcal{X}$, we denote $\mathcal{F}_L^{p,q}(\mathcal{X})$ for the class of convex functions which are p-times differentiable while the q-th derivatives of them are L-Lipschitz continuous on $\mathcal{X}$.*

With this notation, we can present the lower-bound derived by Nesterov [31, Theorem 2.16]:

**Theorem 2.3.4.** *(Lower-bound for convex and smooth optimization [31]) For any* $1 \leq k \leq$

*$\frac{1}{2}(d-1)$ and $x^0 \in \mathbb{R}^d$ there exist a function $F \in \mathcal{F}_L^{\infty,1}(\mathbb{R}^d)$ such that for any iterative algorithm which uses only first-order oracle $\nabla F(.)$, the following inequality holds:*

$$F(x^k) - F^\star \geq \Omega\left(\frac{L_f\|x^0 - x^\star\|_2^2}{(k+1)^2}\right). \tag{2.27}$$

Such a lower-bound suggests that there exists at least one $L$-smooth convex function on which any first-order method cannot converge faster than $O(1/k^2)$ for a limited number of iterations which $1 \leq k \leq \frac{1}{2}(d-1)$.

For the case where $F$ is a non-smooth function, it has been shown by Nesterov [38] in 2005 that one can run the accelerated gradient scheme on an auxiliary objective function which is a smoothed approximation of $F$, and obtain a convergence rate of $F(x^k) - F^\star \leq O(1/k)$, which although being sub-optimal for solving composite problems we are interested in (with the same convergence of proximal gradient descent), is still better than the (sub-) gradient descent's $O(1/\sqrt{k})$ for generic non-smooth optimization. The accelerated gradient scheme has been later extended by [39] and [35] for the composite optimization task (2.3) in which $f(.)$ is smooth but $g(.)$ can be non-smooth. The accelerated gradient method developed by [35] is known as the Beck-Teboulle proximal gradient or the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA):

```
FISTA – Initialize $x^0 \in \mathbb{R}^d,\ a_0 = 0$;
For  $k = 0, 1, 2, ..., K$
    $x^k = \mathrm{prox}_{\lambda g}^{\eta}(y^k - \eta\nabla f(y^k))$; $\rightarrow$ proximal descent step
    $a_{k+1} = (1 + \sqrt{1 + 4a_k^2})/2$;
    $y^{k+1} = x^k + \frac{a_k - 1}{a_{k+1}}(x^k - x^{k-1})$;      $\rightarrow$ momentum step
```

With a computationally-negligible extrapolation step, the FISTA enjoys a significantly improved convergence speed [35]:

$$F(x^k) - F^\star \leq \frac{4L_f\|x^0 - x^\star\|_2^2}{k^2}, \tag{2.28}$$

which means that in order to achieve $F(x^k) - F^\star \leq \epsilon$, only $k \geq \frac{4L}{\sqrt{\epsilon}}$ iterations are needed. Due to the fast convergence, FISTA has become well-known and widely applied by practitioners especially in the fields compressed sensing and signal/image processing applications.

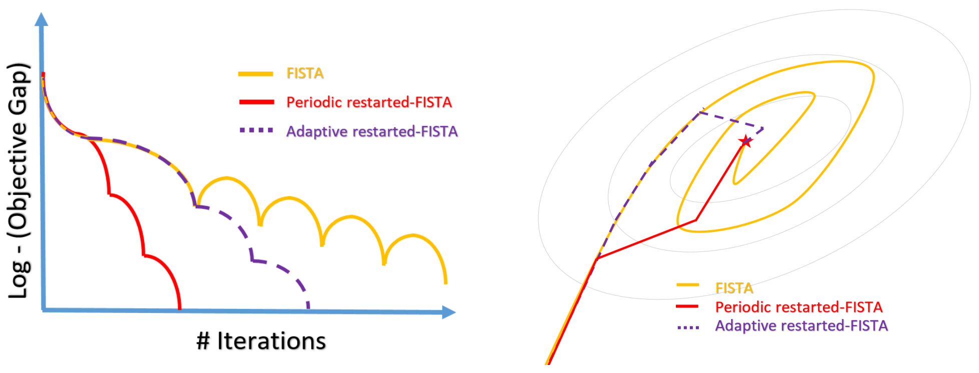


**Figure 2.2:** *Empirical performance illustration of FISTA, periodic restarted FISTA (with exact knowledge of the strong-convexity parameter $\mu$) and adaptive restarted-FISTA (based on enforcing monotonicity) for minimizing strongly-convex functions.*

### 2.3.5 The Restart Schemes for Accelerated Gradient Methods

Although being a fast gradient method which is able to solve generic convex composite optimization task (2.3) with a convergence rate $O(1/k^2)$, the accelerated proximal gradient method FISTA on its own does not have a linear convergence rate when the objective function $F$ is strongly-convex. In contrast, the vanilla proximal gradient descent can automatically converge at a linear rate for minimizing strongly-convex functions. In order to converge at an accelerated linear convergence rate, FISTA need an extra periodic *restart scheme* [39, 40, 41], and such a restart scheme typically requires knowledge of the strong-convexity parameter $\mu$ in order to determine the correct restart period:

Denote $T = \lceil 4\sqrt{L_f/\mu} \rceil$ as the *restart period* for FISTA, we have:

$$F(x^T) - F^\star \leq \frac{4L_f\|x^0 - x^\star\|_2^2}{T^2} \leq \frac{4L_f[F(x^0) - F^\star]}{\mu T^2} \leq \frac{1}{4}[F(x^0) - F^\star]. \tag{2.29}$$

If we reinitialize FISTA with $a_k = 0$ and $y^k = x^k$ every $T = \lceil 4\sqrt{L_f/\mu} \rceil$ iterations, one can easily derive that the restarted-FISTA [42, 40, 43] has an accelerated linear convergence rate:

$$F(x^k) - F^\star \leq \left(\frac{1}{4}\right)^{\lfloor k/T \rfloor} [F(x^0) - F^\star]. \tag{2.30}$$

Now we can see that, in order to achieve an accuracy of $F(x^k) - F^\star \leq \epsilon$ for $\mu$-strongly-convex $F$, the restarted FISTA algorithm needs only $k \geq O(\sqrt{\frac{L_f}{\mu}} \log \frac{1}{\epsilon})$ iterations, while the proximal gradient descent needs $k \geq O(\frac{L_f}{\mu} \log \frac{1}{\epsilon})$ iterations.

The main drawback of the restart scheme is the need for the prior knowledge about the strong-convexity parameter $\mu$, which is difficult in general to be estimated accurately beforehand. To overcome this issue in practice, researchers [42] proposed two heuristic adaptive restart schemes, the *functional restart* and *gradient restart*, which do not need to foreknow the strong-convexity parameter. These adaptive restart schemes are all based on enforcing monotonic descent for each iteration of FISTA, i.e.:

$$F(x^k) \leq F(x^{k-1}), \forall k. \tag{2.31}$$

When we run FISTA with the adaptive restart scheme proposed by [42], the condition (2.31) is examined for each iteration of FISTA. Whenever (2.31) is violated, the adaptive restart scheme re-initializes FISTA with $a_k = 0$ and $y^k = x^k$. Although being the most simple and intuitive heuristic adaptive-restart schemes, they work reasonably well in practice.

In Figure 2.2 we describe visually the practical performance of FISTA (orange line), the FISTA with periodic-restart using the exact knowledge of $\mu$ (red line), and the adaptive-restart heuristic by [42] (purple dashed line) for minimizing a strongly-convex function. The FISTA algorithm typically oscillates around the solution $x^\star$ [42, 40, 44], while the adaptive restarted-FISTA detects the oscillating behavior on the fly and reinitialize the momentum to avoid it. The periodic restart scheme with the exact knowledge of $\mu$ takes the shortest route towards the solution.

## 2.4 Stochastic First-Order Optimization Algorithms

### 2.4.1 Stochastic Gradient Descent

In modern machine learning practice, the optimization problems are often large-scale and high-dimensional, due to the large number of available training data and features. Although the previously introduced deterministic gradient descent methods achieve certain scalablity for large-scale problems, there is still much room for practical improvements. A potentially more powerful approach in practice is the stochastic gradient descent which can be dated back to [45] and has been widely applied in machine learning and data science [46, 47]. If we assume *each* $f_i(.)$ has $L$-Lipschitz continuous gradient:

$$\|\nabla f_i(x) - \nabla f_i(x')\|_2^2 \leq L\|x - x'\|_2^2, \forall x, x' \in \mathbb{R}^d, \tag{2.32}$$

the basic form of stochastic (sub-)gradient descent for (2.3) can be formulated as the following:

```
**SGD** – Initialize $x^0 \in \mathbb{R}^d$
For   $k = 0, 1, 2, ..., K$
    pick $i_k \in [1, 2, ..., n]$ uniformly at random;
    $x^{k+1} = x^k - \eta_k \mathcal{G}_k,\ \mathcal{G}_k \in \nabla f_i(x^k) + \lambda n \partial g(x^k)$
```

In iteration $k$, SGD randomly selects one (or a subset if we use a minibatch scheme) of the functions $f_i(.)$ and compute the gradient $\nabla f_{i_k}(.)$ at $x^k$. It is obvious that the $\nabla f_{i_k}(x^k)$ is an unbiased estimator of $\nabla f(x^k)$ with some variance $\sigma^2$. With necessary shrinking step size sequence $\eta_k$, the SGD and its variants typically achieves $O(1/\sqrt{k})$ convergence in expectation for convex functions [28, Chapter 6]. To be more specific, if $\lambda g(.)$ is $L'$-Lipschitz continuous, this vanilla SGD has an expected convergence rate of:

$$\mathbb{E}F(x^k) - F^\star \leq O\left(\frac{L}{k} + \frac{L' + \sigma}{\sqrt{k}}\right). \tag{2.33}$$

Although being slower than the gradient descent (GD) in terms of the number of iterations, each iteration of SGD is $n$-times cheaper than one iteration of GD, if we ignore the cost of computing the sub-gradient of $g$. Because of this benefit in terms of computation, most of the state-of-the-art and popular optimization algorithms in various applications in machine learning are variants of SGD (even for non-convex problems) – for example the widely applied *Pegasos* algorithm [48] – a stochastic (sub-)gradient descent method for SVM, as well as the *AdaGrad* [49] – a stochastic (sub-) gradient method with adaptive step-sizes for faster online learning, and the *Adam* algorithm which is tailored for training deep neural networks [50, 51, 52], etc.

Inspired by the success of Nesterov's acceleration and proximal gradient descent for deterministic optimization, the SGD can also be extended to have improved convergence rates. For instance, Lan [53] proposes an accelerated (sub-) gradient descent with the Nesterov's acceleration technique, improving the convergence rate to:

$$\mathbb{E}F(x^k) - F^\star \leq O\left(\frac{L}{k^2} + \frac{L' + \sigma}{\sqrt{k}}\right), \tag{2.34}$$

which is still an $O(1/\sqrt{k})$ convergence rate overall. In other words, in order to achieve $\mathbb{E}F(x^k) - F^\star \leq \epsilon$, $k \geq O\left(\frac{n+L'}{\epsilon^2}\right)$ iterations are needed for the accelerated SGD method proposed in [53]. We need to note that such a convergence-rate is worst-case optimal [53] for the online optimization (also known as stochastic-programming) setting where we denote $\xi := (a, b)$ as a random variable:

$$x^\star \in \arg\min_{x \in \mathbb{R}^d} \left\{ F(x) := f(x) + \lambda g(x) \right\}, \quad f(x) := \mathbb{E}_\xi \bar{f}(x, \xi), \tag{2.35}$$

where the data-samples $(a_1, b_1), (a_2, b_2), ..., (a_k, b_k)$ are obtained from a data stream in a sequential manner. In such a setting, at $t$-th iteration, the SGD algorithm computes a stochastic gradient $\triangledown \bar{f}(x^k, a_k, b_k) + \lambda n \partial g(x^k)$ based on the current data-sample $(a_k, b_k)$ and then performs the descent update.

However, although being an optimal algorithm for the generic stochastic optimization objective (2.35), the regularized empirical risk minimization (2.3) is only a subset of this class of objectives[2], where such a rate is far from being optimal. Several researchers [54, 55] have derived important lower-bounds for optimizing the finite-sum objective with stochastic gradient oracle $\triangledown f_i(.)$, and we present here a typical well-known result:

**Theorem 2.4.1.** *(Lower bound for convex and smooth finite-sum optimization [54, Theorem 7].) For any randomized algorithms with access to stochastic gradient oracle $\triangledown f_i(.)$, and any $L$, $R$, $\epsilon \geq 0$, there exist a sufficiently large dimension $d = O(\frac{L^2 R^6}{\epsilon^2} \log \frac{LR^2}{\epsilon} + R^2 n \log n)$, and $n$ functions $f_i \in \mathcal{F}_L^{1,1}(\mathcal{X})$ where $\mathcal{X} \in \left\{x \in \mathbb{R}^d | \|x\|_2 \leq R\right\}$, such that in order to achieve an*

[2]If we set $\xi$ to be a random variable uniformly drawn from the data sample $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)$, we recover the regularized ERM objective (2.3)

*output $\hat{x}$ which satisfies $\mathbb{E}[F(\hat{x}) - F(x^\star)] \leq \epsilon$ for the minimization task:*

$$x^\star \in \arg\min_{x \in \mathcal{X}} \left\{ F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}, \tag{2.36}$$

*a necessary*

$$\Omega \left( n + R\sqrt{\frac{nL}{\epsilon}} \right) \tag{2.37}$$

*number of stochastic gradient evaluation are needed.*

Meanwhile a similar result presented in [54, Theorem 8] demonstrates that if further $F(x)$ is $\mu$-strongly-convex, then such a lower bound can be improved to:

$$\Omega \left[ \left( n + \sqrt{\frac{nL}{\mu}} \right) \log \frac{1}{\epsilon} \right]. \tag{2.38}$$

As we will see in the next subsection, the convergence of SGD on (2.3) can be much improved and can even match these lower-bounds via applying the *variance-reduction* techniques.

### 2.4.2 Stochastic Variance-Reduced Gradient Methods

Although being a huge practical success already, the SGD type methods mentioned in previous subsection are somewhat suboptimal in principle – despite the fact that $\nabla f_{i_k}(x^k)$ is an unbiased estimator for the true gradient $\nabla f(x^k)$, the variance of the gradient estimator is unchanged for all iterations, even when $x^k$ become increasingly nearer to the optimal solution $x^\star$. Hence one needs to either shrink the step size of SGD or increase the minibatch size to reduce this variance – both of these approaches will slow down the convergence of SGD. This is precisely why SGD appears to be much slower than GD in terms of iteration number.

Recently, researchers have come up with novel SGD variants which are able to achieve *progressive variance-reduction* on the stochastic gradient estimator with a small amount of computation or memory overhead, starting from the *stochastic averaged gradient* (SAG) algorithm [56, 57], to *stochastic variance-reduced gradient* (SVRG) algorithm [58, 59, 60], SDCA [61] and SAGA [62], etc. Among these algorithms, researchers commonly regard the SVRG and SAGA being the most representative ones.

We present here the proximal version of SVRG [60] for solving the composite optimization tasks:

```
Prox-SVRG  –  Initialize $x^0 \in \mathbb{R}^d, \hat{x}^0 = x^0,$ return $\hat{x}^{S+1}$
          For  $s = 0, 1, 2, ..., S$
               Compute $\nabla f(\hat{x}^s)$
               For  $j = 0, 1, 2, ..., m$
                   $k = sm + j;$
                   pick $i_k \in [1, 2, ..., n]$ uniformly at random;
                   $\nabla_k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(\hat{x}^s) + \nabla f(\hat{x}^s); \rightarrow$ variance reduction
                   $x^{k+1} = \text{prox}_{\lambda g}^{\eta}[x^k - \eta \nabla_k],$   $\rightarrow$ proximal SGD
               $\hat{x}^{s+1} = \frac{1}{m} \sum_{j=1}^{m} x^{sm+j+1}. \rightarrow$ update the snapshot point
```

The Prox-SVRG algorithm adopts a two-loop structure. It introduces a sequence of "*snapshot*" points $\hat{x}^s$ along the path towards the optima, and obtain the stochastic gradient estimator at point $x^k$ as:

$$\nabla_k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(\hat{x}) + \nabla f(\hat{x}^s). \tag{2.39}$$

Obviously, $\nabla_k$ is still an unbiased estimator of the true gradient at $x^k$:

$$\mathbb{E}_{i_k} \nabla_k = \mathbb{E}_{i_k} \nabla f_{i_k}(x^k) - \mathbb{E}_{i_k} \nabla f_{i_k}(\hat{x}^s) + \nabla f(\hat{x}^s) = \nabla f(x^k), \tag{2.40}$$

however, the variance of this estimation is no longer constant but dependent to the suboptimality of $x^k$ and $\hat{x}$, as shown by [58], if we assume each $f_i(.)$ has $L$-Lipschitz continuous gradient, then we can have:

$$\mathbb{E}_{i_k} \|\nabla_k - \nabla f(x^k)\|_2^2 \leq 4L[f(x^k) - f(x^\star) + f(\hat{x}^s) - f(x^\star)]. \tag{2.41}$$

Intuitively, if we make the $\hat{x}^s$ approaching closer and closer to the optima $x^\star$, then the variance will be reduced as $x^k \rightarrow x^\star$. The outerloop of Prox-SVRG updates the snapshot point $\hat{x}^s$ and the full gradient $\nabla f(\hat{x}^s)$, while the inner-loop performs stochastic gradient descent using the variance-reduced gradient estimator in the form of (2.39).

With the epoch length set to be $m = O(n)$, the Prox-SVRG algorithm combines the merit of both deterministic and stochastic gradient descent algorithm – the fast convergence speed

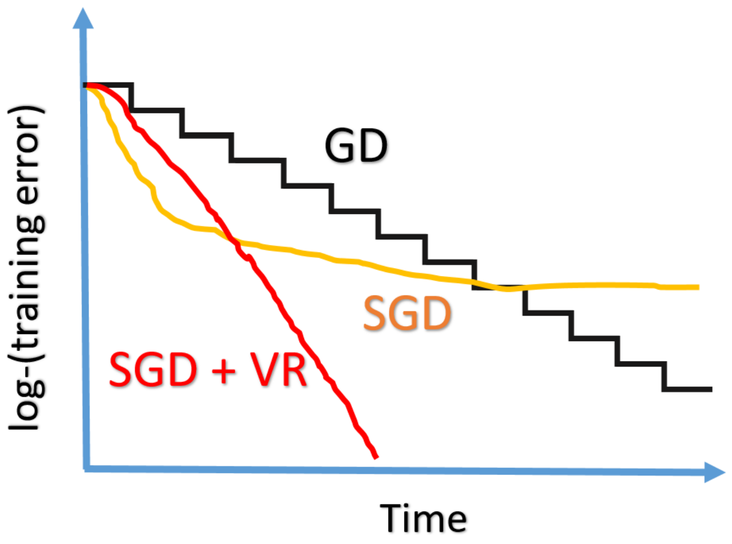


**Figure 2.3:** *Empirical performance illustration of deterministic GD, SGD, and SGD with variance-reduction. The SGD usually has a fast initial convergence, but will slow down gradually due to the non-decreasing variance of stochastic gradient estimator, while the variance-reduction technique overcomes this issue.*

and low computational cost per-iteration. We visually describe the superiority of the variance-reduction technique over the vanilla SGD and deterministic gradient method by Figure 2.3.

Recently, researchers advance the SVRG-type algorithms with extending the Nesterov's acceleration techniques, starting from the work of [63], [6], [64], [65] and [55], with the state-of-the-art algorithms being: the *Katyusha* algorithm [4] which is an accelerated SVRG algorithm with a delicately designed momentum step, the *MiG* algorithm [66] which shares the same spirit of Katyusha, and the accelerated variants of SAGA algorithm [67, 68]. It is worth noting that such accelerated variance-reduction methods matches the worst-case iteration complexity lower bounds we have shown in the previous section proven by [55] and [54] for smooth and convex/strongly-convex finite-sum optimization, hence they are so-called "optimal" algorithms.

We summarize the iteration complexity of both deterministic gradient descent methods and stochastic variance-reduced gradient descent algorithms we have introduced thus-far for solving the composite finite-sum optimization problem (2.3) in table 2.1[3]. Note that the iteration complexity is measured as the number of $\nabla f_i(.)$ evaluations an algorithm needed to achieve an output $\hat{x}$ such that $F(\hat{x}) - F(x^\star) \leq \epsilon$. If we assume $L \approx O(L_f)$, which usually holds

[3]FISTA needs to have a periodic restart to achieve the linear rate described in the table, in the presence of strong-convexity.

| ALGORITHMS | ITERATION COMPLEXITY $\mu = 0$ | ITERATION COMPLEXITY $\mu > 0$ |
|---|---|---|
| PROX-GD | $O\left(\frac{nL_f}{\epsilon}\right)$ | $O\left(\frac{nL_f}{\mu}\log\frac{1}{\epsilon}\right)$ |
| AGD / FISTA | $O\left(\frac{nL_f}{\sqrt{\epsilon}}\right)$ | $O\left(n\sqrt{\frac{L_f}{\mu}}\log\frac{1}{\epsilon}\right)$ |
| PROX-SVRG/SAGA, ETC | $O\left(\frac{n+L}{\epsilon}\right)$ | $O\left((n+\frac{L}{\mu})\log\frac{1}{\epsilon}\right)$ |
| KATYUSHA/MIG, ETC | $O\left(n+\sqrt{\frac{nL}{\epsilon}}\right)$ | $O\left((n+\sqrt{\frac{nL}{\mu}})\log\frac{1}{\epsilon}\right)$ |

**Table 2.1:** *The number of $\triangledown f_i(.)$ evaluations needed for the first-order methods to find a solution $\hat{x}$ which satisfies $F(\hat{x}) - F^\star \leq \epsilon$. We denote by $\mu$ the strong-convexity parameter of $F$, $L_f$ the Lipschitz constant of $\triangledown f(.)$, and $L$ the maximum Lipschitz constant of $\triangledown f_i(.)$.*

true in machine learning practice with modern datasets[4], then the Prox-SVRG type methods have strictly better iteration complexity than Prox-GD, while the accelerated stochastic gradient methods like Katyusha have strictly better iteration complexity than accelerated full gradient method such as FISTA.

### 2.4.3 Randomized Coordinate Descent

Another important type of randomized first-order algorithm for convex composite optimization is the *randomized coordinate descent* methods (RCD) [69, 70] for solving a subset of the convex composite optimization task (2.3), which reads:

$$x^\star = \arg\min_{x\in\mathbb{R}^d} \{F(x) := f(x) + \lambda g(x)\}, \tag{2.42}$$

where $x$ consists of $d$-variables: $[x_{(1)}, ..., x_{(d)}]$ and the regularization term $g(x)$ is potentially non-smooth but separable such that $g(x) = \sum_{i=1}^{d} g_i(x_{(i)})$. If we define the partial gradient $\triangledown_i f(x) = \frac{\partial f}{\partial x_{(i)}}(x)$ and assume a coordinate-wise smoothness of function $f(.)$ which has the following form:

$$|\triangledown_i f(x + he_i) - \triangledown_i f(x)| \leq L_i|h|, \tag{2.43}$$

[4]However, as we will show in chapter 6, there are several important counter examples for signal/image processing applications, such as space-varying deblurring.

for any coordinate index $i$. We denote $e_i$ as a $d$-dimensional vector with $i$-th element being 1 while 0 for any other elements, $x \in \mathbb{R}^d$ and $h \in \mathbb{R}$. The randomized coordinate descent (RCD) algorithm typically takes the following form:

```
RCD – Initialize $x_0 \in \mathbb{R}^d$
For  $k = 0, 1, 2, ..., K-1$
    pick $i_k \in [1, 2, ..., d]$ uniformly at random;
    $x^{k+1} = \text{prox}_{\lambda g_{i_k}}^{L_{i_k}^{-1}}[x^k - L_{i_k}^{-1}\nabla_{i_k} f(x^k)]$,
```

where in each iteration a coordinate index is picked at random and a gradient descent step on the chosen coordinate is performed. To achieve an output of $F(x^K) - F^\star \leq \epsilon$ for a general convex $F(.)$, such an vanilla coordinate descent algorithm needs:

$$O\left(\frac{d \max_i L_i}{\epsilon}\right), \tag{2.44}$$

number of iterations [70, 71]. Note that one may easily replace the $\max_i L_i$ with the average $\frac{1}{d}\sum_{i=1}^d L_i$ via importance sampling according to the Lipschitz constants $L_i$ [69]. Since the complexity of one evaluation of the partial gradient $\nabla_i f(.)$ is usually $O(n)$, to achieve $\epsilon$-accuracy the RCD needs $O\left(\frac{nd \max_i L_i}{\epsilon}\right)$ FLOPs, while the gradient descent needs $O\left(\frac{ndL}{\epsilon}\right)$. Due to the fact that $\max_i L_i \leq L$ (and often $\ll L$ in practice[5]), the RCD algorithm, when applicable, can also achieve improved computational efficiency via randomization on the coordinate to exploit the directional smoothness.

Like the Nesterov's momentum trick for accelerating both the deterministic and stochastic gradient descent, the convergence rate of the RCD can also be much improved by applying the Nesterov's acceleration, as shown by [69], [72], [6] and [73], etc. By further incorporating importance sampling, one may yield a state-of-the-art RCD iteration complexity which is of $O\left(\sqrt{\frac{\sum_{i=1}^d L_i}{\epsilon}}\right)$ for convex functions, and $O\left(\sqrt{(\sum_{i=1}^d L_i)/\mu}\log\frac{1}{\epsilon}\right)$ for $\mu$-strongly-convex functions [74].

[5]For instance, consider the Lasso problem (2.4) in Section 2.1 where $A$ is a random Gaussian matrix. Denote $H = A^TA$ and $\sigma_{\max}(H)$ as the largest singular value of $H$, then we have $\max_i L_i = \max_i H_{ii} \approx \frac{\sigma_{\max}(H)}{d} = \frac{L}{d} \ll L$.

## 2.5 Primal-Dual Gradient Methods

So far we have covered a variety of deterministic and stochastic gradient algorithms which directly solve the original convex composite optimization problem (2.3). We denote the original form of (2.3) as the *primal problem*. However, in many image processing and machine learning applications, we may face scenarios which are more complex and challenging, for example:

**Non-smooth data-fidelity term.**

The data-fidelity term $f(x)$ can be non-smooth, with a typical example in imaging – denoising a (vectorized) image $x^\#$ which is corrupted with salt-and-pepper noise. A successful way of recovering such an image is to solve:

$$x^\star \in \arg\min_x \lambda\|Dx\|_1 + \|x - x^\#\|_1, \tag{2.45}$$

namely the TV-$\ell_1$ model [75], where the first term (TV-regularization) utilizes the prior knowledge that the image to be estimated is piece-wise smooth, while the second-term aims to nullify the salt-and-pepper noise via using the $\ell_1$ norm. It is shown to have superior performance over the traditional TV-denoising model which uses $\|x - x^\#\|_2$ as the second term. However the TV-$\ell_1$ model is the composition of two non-smooth functions, and the first-order splitting algorithms we have introduced so far cannot be directly applied to such a problem.

**Computationally expensive proximal terms.**

The regularization term in (2.3) is with a linear operator: $g(Dx)$, where $D$ can be a pretrained dictionary, or a differential operator (TV regularization as eqn.2.45), or a sparse graph. In such cases, the proximal operator of the regularization term is non-trivial to compute in general and we need to run a separate iterative procedure as a sub-loop. Due to this non-negligible computational cost, it compromises the computational gain of stochastic gradient methods over the full gradient methods in practice, because they need to compute the proximal operator much more often than the full gradient methods.

Moreover, in some applications, one may wish to apply multiple non-smooth regularization terms, in order to obtain a potentially stronger modeling of the prior knowledge and a better estimation of the ground truth, which again limits the direct application of the fast gradient methods such as FISTA which solve directly the primal problem.

In these challenging scenarios, it is often more advantageous to reformulate the original primal problem into a more flexible *primal-dual* saddle-point problem, which is potentially easier to deal with. The corresponding optimization algorithms are called the *primal-dual hybrid gradient* methods, first proposed by Chambolle and Pock [76], and extended by various researchers to stochastic settings [77, 78, 65]. In this section we introduce the basic concepts of the primal-dual framework and its optimization.

### 2.5.1 Fenchel-Rockafellar Duality and Primal-Dual Hybrid Gradient

Following the related literature [76, 27, 79] and the notations therein, let's now focus on the following generic form of convex program:

$$x^\star \in \arg\min_{x\in\mathcal{X}} f(Ax) + \lambda g(x), \tag{2.46}$$

where both $g$ and $f$ are convex and lower semi-continuous functions on finite-dimensional vector spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively, while $A$ is a linear operator which maps $\mathcal{X} \to \mathcal{Y}$. We now introduce the Fenchel conjugate [80] of a function $f$, denoted as $f^*$:

$$f^*(y) = \sup_{x\in\mathcal{X}} \langle x, y\rangle - f(x). \tag{2.47}$$

For a convex function $f$ we can have $f^{**} = f$, and hence (see e.g. [27]):

$$\begin{aligned}
\arg\min_{x\in\mathcal{X}} f(Ax) + \lambda g(x) &= \arg\min_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} [y^T Ax - f^*(y) + \lambda g(x)] &(2.48)\\
&= \arg\max_{y\in\mathcal{Y}} \inf_{x\in\mathcal{X}} [y^T Ax - f^*(y) + \lambda g(x)] &(2.49)\\
&= \arg\max_{y\in\mathcal{Y}} \left\{ -f^*(y) - \sup_{x\in\mathcal{X}} [-y^T Ax - \lambda g(x)] \right\} &(2.50)\\
&= \arg\max_{y\in\mathcal{Y}} -f^*(y) - \lambda g^*(-A^T y). &(2.51)
\end{aligned}$$

The first equality yields the primal-dual saddle-point reformulation and the last equality yields the dual reformulation. Let $x^\star$ denote a solution of the primal problem (2.46), $y^\star$ denote a solution of the dual problem, and denote:

$$\mathcal{Q}(x, y) = y^T Ax - f^*(y) + \lambda g(x), \tag{2.52}$$

then $[x^\star, y^\star]$ is a saddle point of $\mathcal{Q}(x,y)$ such that:

$$\mathcal{Q}(x^\star, y) \leq \mathcal{Q}(x^\star, y^\star) \leq \mathcal{Q}(x, y^\star),\ \forall x \in \mathcal{X},\ y \in \mathcal{Y}. \tag{2.53}$$

Now one may alternatively solve the saddle point problem:

$$[x^\star, y^\star] = \arg\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} [y^T A x - f^*(y) + \lambda g(x)], \tag{2.54}$$

instead of solving the primal problem directly. A classic way of solving this saddle-point problem is the *Primal-Dual Hybrid Gradient* (PDHG) algorithm [76, 79]. In each iteration of PDHG, it first performs proximal gradient descent on the primal variable $x$ and next follows a momentum step to achieve acceleration of convergence, and then the algorithm updates the dual variable again via proximal gradient descent:

```
**PDHG** – Initialize $[x_0, y_0] \in \mathcal{X} \times \mathcal{Y}$
For  $k = 0, 1, 2, ..., K$
    $x_{k+1} = \mathrm{prox}^{\eta}_{f^*}\left(x_k - \eta A^T y_k\right);$ $\rightarrow$ primal descent step
    $\bar{x}_{k+1} = x_{k+1} + \sigma(x_{k+1} - x_k);$   $\rightarrow$ extrapolation step
    $y_{k+1} = \mathrm{prox}^{\alpha}_{\lambda g}(y_k + \alpha A\bar{x}_{k+1});$     $\rightarrow$ dual ascent step
```

We denote $\eta$ as the primal step-size, $\alpha$ as the dual step-size, and $\sigma$ as the momentum step-size. By Moreau's identity [81]:

$$x = \mathrm{prox}_{\alpha f}(x) + \alpha \mathrm{prox}_{\frac{1}{\alpha}f^*}(\frac{x}{\alpha}),\ \forall \alpha > 0, \tag{2.55}$$

the proximal operator of $f^*(x)$ can be computed as the following:

$$\mathrm{prox}^{\eta}_{f^*}(x) = x - \eta \mathrm{prox}^{\eta}_{\frac{1}{\eta}f}(\frac{x}{\eta}). \tag{2.56}$$

With these step-sizes chosen such that $\frac{1}{\eta\alpha} \geq \|A\|^2$, $\sigma = 1$, the ergodic sequence ($\bar{x}^K = \frac{1}{K}\sum_{i=1}^K x_i$, $\bar{y}^K = \frac{1}{K}\sum_{i=1}^K y_i$) obtained by PDHG will satisfy the $O(1/K)$ convergence rate [76, 79]:

$$\mathcal{Q}(\bar{x}^K, y) - \mathcal{Q}(x, \bar{y}^K) \leq \frac{\frac{1}{\eta}\|x - x_0\|_2^2 + \frac{1}{\alpha}\|y - y_0\|}{K}. \tag{2.57}$$

Moreover, if $f^*$ or $g$ is strongly convex[6], the convergence rate can be improved to $O(1/K^2)$ with suitably chosen step-sizes according to the additional strong-convexity parameter.

### 2.5.2 Primal-Dual Gradient Methods for Multiple-Composite Optimization

In some machine learning and signal processing practices, we may wish to apply some sophisticated regularization terms to improve generalization or estimation accuracy. For instance, let's consider the three-composite minimization task which reads:

$$x^\star \in \min_{x\in\mathbb{R}^d} \{f(x) + \lambda g(Dx) + \gamma h(x)\}, \;\; f(x) := \sum_{i=1}^{n} f_i(x), \tag{2.58}$$

where we have two convex and possibly non-smooth regularization terms $\lambda g(Dx) + \gamma h(x)$ with $g : \mathbb{R}^r \to \mathbb{R} \cup \{+\infty\}$ and $h : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, while $D : \mathbb{R}^d \to \mathbb{R}^r$ is a linear operator. One important instance of (2.58) is the *graph-guided fussed lasso*, with $g(Dx)$ being a total-variation (TV) regularization which enforce piece-wise smoothness, and $h(x)$ being an $\ell_1$ norm penalty for sparse solutions. Since by conjugacy (2.47) we have:

$$g^*(Dx) = \sup_{y\in\mathbb{R}^r} y^T Dx - g(y), \tag{2.59}$$

hence,

$$\begin{aligned} f(x) + \lambda g(Dx) + \gamma h(x) &= f(x) + (\lambda g(Dx))^{**} + \gamma h(x) \\ &= f(x) + (\sup_{y\in\mathbb{R}^r} y^T Dx - \lambda g(y))^* + \gamma h(x) \\ &= f(x) + (\sup_{y\in\mathbb{R}^r} y^T Dx - \lambda g^*(y)) + \gamma h(x) \end{aligned}$$

Similar to (2.54), the saddle-point formulation can be written as:

$$[x^\star, y^\star] = \min_{x\in\mathbb{R}^d} \max_{y\in\mathbb{R}^r} f(x) + \gamma h(x) + y^T Dx - \lambda g^*(y). \tag{2.60}$$

With this reformulation the linear operator $D$ and the function $g(.)$ are decoupled and hence one can divide-and-conquer the (potentially) expensive proximal operator on $g(Dx)$, and meanwhile efficiently handle multiple regularization terms, with the primal-dual gradient methods. The PDHG algorithm can be written as the following with $x_0 = z_0 \in \mathbb{R}^d$, $y_0 \in \mathbb{R}^r$ and $(\alpha, \eta, \sigma)$

[6]If $f^*$ is $\mu_f$-strongly-convex, then $f$ is $\frac{1}{\mu_f}$-smooth.

being dual, primal, and extrapolation step sizes respectively [79]:

```
**PDHG** (Three Operator Splitting)
−Initialize $[x_0, y_0] \in \mathcal{X} \times \mathcal{Y}$
For   $k = 0, 1, 2, ..., K$
    $y_{k+1} = \text{prox}_{\lambda g^*}^{\alpha}(y_k + \alpha D z_k);$   $\rightarrow$ dual ascent step
    $x_{k+1} = \text{prox}_{\mu h}^{\eta}\left(x_k - \eta[D^T y_{k+1} + \nabla f(x_k)]\right);$ $\rightarrow$ primal descent step
    $z_{k+1} = x_{k+1} + \sigma(x_{k+1} - x_k);$   $\rightarrow$ extrapolation step
```

Just like the proximal gradient descent in the primal, the Chambolle-Pock algorithm can also be further extended with randomization techniques, such as the recently introduced stochastic primal-dual coordinate descent algorithms [82, 65, 77] and primal-dual stochastic gradient for saddle-point problems [83, 84, 85, 78].

Finally, we briefly mention here that, there is a type of meta-algorithms named alternating direction method of multipliers (ADMM) [86, 87] which share the same spirit[7] of primal-dual gradient methods and achieve a similar advantage on flexibility, as well as comparable theoretical guarantees and practical performance. Since we focus on first-order optimization algorithms in this thesis, we choose to introduce the primal-dual gradient methods which are more relevant for this thesis and do not redundantly present the ADMM here (we refer the interested readers to [86] for details).

[7]In fact, the PDHG has been shown in [76, Section 4.3] to be equivalent to a “preconditioned” version of ADMM.

## 2.6 Sketching for Reduced Computation

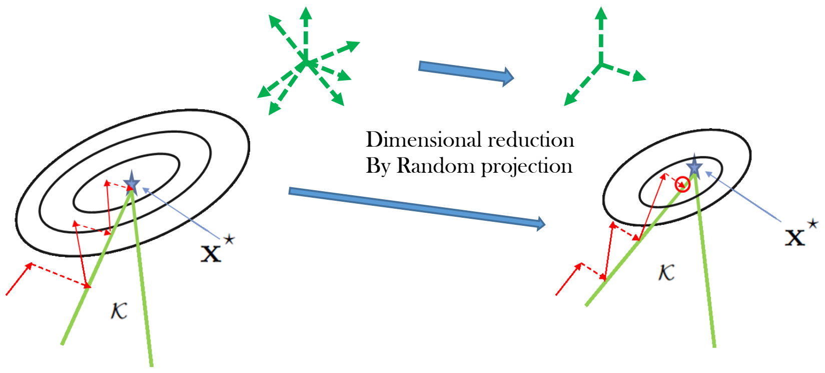


**Figure 2.4:** *Sketching for Faster Computation of Iterative Solvers*

In this section we introduce another successful way of utilizing the power of randomization for efficiency in large-scale optimization – the *randomized projection*, or rather, *sketching*. The sketching techniques have been intensively studied and widely applied in the field of numerical linear algebra in recent years, with most representative applications being the low-rank approximation [88, 89, 90], matrix factorization [91, 92], and least-squares regression [93, 94, 2, 3].

The crux of the sketching idea is that, for a given data matrix $A \in \mathbb{R}^{n\times d}$ with a huge size, one may multiply it with a random matrix $S \in \mathbb{R}^{m\times n}$ where $m \ll n$ and construct a compressed version $SA \in \mathbb{R}^{m\times d}$ which is much smaller than $A$ but preserves most of the information. In practice, the typical computations involved, such as matrix-vector product, matrix-matrix multiplication, matrix inversion, and singular-value decomposition (SVD) can be computationally expensive for the size of original data matrix, but may be significantly cheaper for the sketched data matrix – hence such a scheme may lead us to a fast approximate solution to the original tasks. Meanwhile, in modern computing devices, the memory which allows fast access is usually quite limited in size and may not fit the whole data matrix $A$. In this scenario, $A$ has to be stored in the hard-disks which are slow to be accessed. The sketching technique which directly compresses the size will also provide us a solution for this storage issue by avoiding the frequent access of the full data matrix. Therefore, the benefit provided by sketching is two-fold – both in terms of computation and storage.

### 2.6.1 Randomized Projection for Fast Least-Squares Regression

One interesting line of research focuses on accelerating the constrained least-squares regression task, which is an important instance of the composite finite-sum optimization problem (2.3):

$$\begin{aligned} x^\star &= \arg\min_{x\in\mathbb{R}^d}\left\{F(x) := \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - a_i^T x)^2 + \iota_{\mathcal{K}}(x)\right\} && (2.61)\\ &= \arg\min_{x\in\mathcal{K}}\frac{1}{2}\|Ax-y\|_2^2, && (2.62)\end{aligned}$$

where we assume $\mathcal{K}$ as a convex set in $\mathbb{R}^d$, and the data matrix $A \in \mathbb{R}^{n\times d}$, with an associated semi-norm defined as $\|x\|_A := \|Ax\|_2$. Suppose we use this least-squares estimator to infer a true parameter $x^\dagger \in \mathcal{K}$ from noisy linear measurements:

$$y = Ax^\dagger + w, \tag{2.63}$$

where the noise vector $w$ is drawn from Gaussian distribution $\mathcal{N}(0, \sigma^2 I_d)$. To efficiently solve the linear regression tasks in the large data setting $n \gg d \gg O(1)$, researchers [95, 92, 93, 2] have focused on a classical sketched program:

$$x^{\text{cs}} = \arg\min_{x\in\mathcal{K}}\frac{1}{2}\|SAx - Sy\|_2^2, \tag{2.64}$$

where the "sketching" matrix $S \in \mathbb{R}^{m\times d}$ is a random projection matrix which satisfies:

$$\mathbb{E}\left(\frac{S^T S}{m}\right) = I, \tag{2.65}$$

and is left-multiplied with the data matrix $A$ and observation $y$, and reduces their dimensions to $A_s := SA \in \mathbb{R}^{m\times d}$, $y_s := Sy \in \mathbb{R}^{m\times 1}$. When $n \gg d$, the sketched program (2.64) is potentially much easier to solve than the original least-squares problem, both in terms of computational cost and storage cost. For example, for the unconstrained case $\mathcal{K} = \mathbb{R}^d$ and $A$ has full column rank, the original least-squares problem (2.61) admits a close form solution:

$$x^\star = (A^T A)^{-1} A^T y. \tag{2.66}$$

If we solve the unconstrained LS in this direct way, we need $O(nd^2 + d^3) \approx O(nd^2)$ floating point operations – that is, $O(nd^2)$ for computing $A^T A$ and $O(d^3)$ for inverting $A^T A$. Meanwhile $O(nd)$ memory is require during the computation since the full data matrix needs to be stored. In the same way, the sketched LS can be solve in the close form if the sketched data matrix $A_s$ has full column rank:

$$x^{cs} = (A_s^T A_s)^{-1} A_s^T y_s. \tag{2.67}$$

If the sketch dimension $m = O(d)$, we need $O(md^2 + d^3) \approx O(md^2)$ floating point operation and $O(md)$ storage to solve the sketched program.

The solution of the sketched program $x^{\text{cs}}$ is an approximation of $x^\star$, and there is a clear trade-off between the solution's accuracy and the sketch size. For instance, Pilanci and Wainwright [2] have shown that, for a general convex constraint set $\mathcal{K}$, if the sketching matrix $S$ satisfies some concentration properties, and the sketch size $m \geq O(\frac{d}{\epsilon^2})$, the resulting output $x^{\text{cs}}$ of the sketched program (2.64) satisfies $F(x^{\text{cs}}) \leq (\frac{1+\epsilon}{1-\epsilon})^2 F(x^\star)$ with high probability. The basic (sub-) Gaussian sketch operator can be defined as the following:

**Definition 2.6.1.** *(sub-Gaussian sketch [2, Section 2.B]) We define the sub-Gaussian sketching matrix $S_{\text{Gauss}} \in \mathbb{R}^{m\times n}$ such that each row $s_i$, $i \in [1, 2, ..., m]$ is i.i.d drawn from the $\sigma$-sub-Gaussian distribution.*

A simple special case is the standard Gaussian sketch where $s_i \in \mathcal{N}(0, I_{d\times d})$. Nevertheless, the sub-Gaussian sketch is inefficient, since computing $S_{Gauss}A$ will involve dense matrix-vector multiplication and demand a $O(nmd)$ floating point operations. In practice, the sub-Gaussian sketch is usually replaced by faster randomzied sketching operators – most representative ones are based on efficient orthogonal transforms such as the FFT and the fast Hadarmad Transforms [96, 97]. We will introduce these fast randomized sketching operators in subsection 2.6.2.

Variants of the sketching-based programs have been developed down through the years for various purposes, such as better trade-offs on sketch-size and accuracy via an iterative sketching scheme [3], sketching for fast approximate preconditioning [94, 98, 99], extending beyond the least-squares regression [100], applying the sketching scheme to accelerate the computation in kernel methods for non-parametric regression [101, 102, 103].

## 2.6.2 Fast Sketching Operators

A major computational step in constructing a sketched program is performing the sketching operator $S$ on $A$ to obtain the sketched data matrix $SA$. The (sub-)Gaussian sketches, although having a strong theoretical guarantee which implies stable and reliable dimensional reduction with a relatively small sketch size, require dense matrix-vector products and hence become computationally inefficient – typically $O(mnd)$ floating point operations. Due to this computational drawback, it is often only of theoretical interest. In practice, fast random sketching methods are considered. For instance, one may use the so-called *randomized orthogonal system* (ROS) sketch[8] [96, 97] , which combines the fast orthogonal transforms with random sign-flips and subsampling:

**Definition 2.6.2.** *(Randomized Orthogonal Systems [2, Section 2.C].) Let $F \in \mathbb{R}^{n\times n}$ be a orthogonal matrix with entries in the interval of $[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$, subsampling matrix $E = [e_1, e_2, ...e_m]$ be a random subset (drawn uniformly with replacement) of the rows of identity matrix $I_{n\times n}$, and $D = \mathrm{diag}(v)$ be a diagonal matrix with i.i.d Rademacher variables $v \in \{1, -1\}$, the randomized orthogonal system $S_{\mathrm{ROS}} \in \mathbb{R}^{m\times n}$ can be defined as:*

$$S_{\mathrm{ROS}} = \sqrt{n} \cdot E \cdot F \cdot D \tag{2.68}$$

The orthogonal transform often admits fast operations, with famous instances such as the fast Fourier transform (FFT) and fast Hadamard transform. Since the fast Hadamard transform and the FFT, are fast orthogonal transforms and require only $O(n \log n)$ operations, the ROS's complexity is of $O(nd \log n)$ which is more efficient than the Gaussian sketches for the regime where $m > O(\log n)$.

We also wish to introduce another popular and extremely efficient sketching operator, which is named the *Count Sketch* [88] – a special case of sparse embedding [104, 105, 106, 107]. The Count-Sketch can be defined as the following:

**Definition 2.6.3.** *(Count-Sketch [88, Definition 1.1].) For a sketch size $m$, the sparse embedding sketching operator is defined as*

$$S_c = ED \in \{-1, 0, +1\}^{m\times n}, \tag{2.69}$$

*where:*

- *$h : [n] \to [m]$ is a random map such that for each $i \in [n]$, $h(i) = m'$ for $m' \in [m]$ with probability $\frac{1}{m}$.*

[8]Also known as the Fast Johnson-Lindenstrauss Transform (FJLT).

- $E \in \{0,1\}^{m\times n}$ *is a binary matrix with* $E_{h(i),i} = 1$, *and* $0$ *elsewhere.*
- $D = \mathrm{diag}(v)$ *is a diagonal matrix with i.i.d Rademacher variables* $v \in \{+1, -1\}$.

The count sketch computes a sketched matrix $S_cA$ in only $O(\mathrm{nnz}(A)) \le O(nd)$ time, where we denote the number of non-zeros as "nnz". In each column of $S_c$ there is only 1 non-zero element which is either $+1$ or $-1$. For example, a $4$ by 7 Count-sketch matrix can take the following form:

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.70}$$

The Count-sketch is a very efficient way of computing the sketch of the full data matrix $A$ and has demonstrated excellent practical performance in various applications [108, 109].

### 2.6.3 The Subspace Embedding Accuracy of the Fast Sketching Methods

We have thus far introduced two most popular and practical choices of fast sketching operators compare to the naive (sub-)Gaussian sketch, however, it comes at a cost of sacrificing the *subspace embedding* accuracy, which is directly related to the accuracy of the final output of the sketched program. We introduce the notion of subspace embedding here:

**Definition 2.6.4.** *(Subspace Embedding.) [91]* $S \in \mathbb{R}^{m\times n}$ *is a* $(1\pm\epsilon)$*-subspace embedding of the span of a matrix* $\bar{A} \in \mathbb{R}^{n\times \bar{d}}$, *if* $\forall v \in \mathbb{R}^{\bar{d}}$, *we have:*

$$(1-\epsilon)^2\|\bar{A}v\|_2^2 \le \|S\bar{A}v\|_2^2 \le (1+\epsilon)^2\|\bar{A}v\|_2^2. \tag{2.71}$$

If we apply a sketching operator $S$ to the constrained LS regression objective $F(x)$ defined in (2.61) and obtain the classical sketched program $x^{\mathrm{cs}} \in \arg\min_{x\in\mathcal{K}} \|SAx - Sy\|_2^2$, then the approximation accuracy of $x^{\mathrm{cs}}$ towards the true LS solution $x^\star$ depends on the embedding accuracy of $S$ on the column space of matrix $[A, y]$:

**Proposition 2.6.5.** *[91] If* $S \in \mathbb{R}^{m\times n}$ *is a* $(1\pm\epsilon)$*-subspace embedding of the span of matrix* $[A, y]$ *almost surely, then we have:*

$$F(x^{\mathrm{cs}}) \le \left(\frac{1+\epsilon}{1-\epsilon}\right)^2 F(x^\star), \tag{2.72}$$

*almost surely.*

It is easy to derive this result: let $\bar{A} = [A, y]$, $v^\star = [x^{\star T}, -1]^T$, $v_{\text{cs}} = [x^{\text{cs}T}, -1]^T$ in Definition 2.6.4, then we have:

$$\|S\bar{A}v^\star\|_2^2 = \|S(Ax^\star - y)\|_2^2 \leq (1+\epsilon)^2\|Ax^\star - y\|_2^2. \tag{2.73}$$

Meanwhile since $x^\star \in \mathcal{K}$, $x^{\text{cs}} \in \mathcal{K}$, using the fact that $x^{\text{cs}}$ is the optimal solution of the sketched LS program, we have:

$$\|S(Ax^\star - y)\|_2^2 \geq \|S(Ax^{\text{cs}} - y)\|_2^2 = \|S\bar{A}v_{\text{cs}}\|_2^2 \geq (1-\epsilon)^2\|Ax^{\text{cs}} - y\|_2^2. \tag{2.74}$$

Hence we have $\|Ax^{\text{cs}} - y\|_2^2 \leq \frac{(1+\epsilon)^2}{(1-\epsilon)^2}\|Ax^\star - y\|_2^2$.

In order to achieve a $(1 \pm \epsilon)$ subspace-embedding accuracy of a given linear subspace, the requirement of the sketch size $m$ is different for different sketching matrices. In the following table we summarize from the literature (e.g. [91, 93]) the existing subspace-embedding results for sub-Gaussian sketch, ROS sketch, and the Count-sketch. We also summarize the computational cost measured by number of floating point operation for computing $SA$ for each of these methods.

| Sketching Operator | Sketch size ($m$) | Computational cost of $SA$ |
|---|---|---|
| sub-Gaussian sketch | $O(\frac{d}{\epsilon^2})$ | $O(nmd)$ |
| ROS sketch | $O\left(\frac{(\sqrt{d}+\sqrt{n})^2 \log d}{\epsilon^2}\right)$ | $O(nd\log n)$ |
| Count-sketch | $O(\frac{d^2}{\epsilon^2})$ | $O(nd)$ |

**Table 2.2:** *Necessary sketch size to achieve* $(1\pm\epsilon)$ *subspace embedding for a matrix* $A \in \mathbb{R}^{n\times d}$ *with high probability, and the computational cost of different sketching techniques*

One can clearly observe that there is a trade-off between the computational cost of performing the sketch and the embedding performance. The Count-Sketch method, although it only need $O(nd)$ floating point operations to compute $SA$ due to the fact that it is an extremely sparse sketching matrix, it needs a sketch size $m = O(\frac{d^2}{\epsilon^2})$ to achieve a $(1 \pm \epsilon)$-subspace embedding. Such a sketch size requirement appears to be inferior to the $O(\frac{d}{\epsilon^2})$ result for sub-Gaussian sketch and the $O(\frac{(\sqrt{d}+\sqrt{n})^2 \log d}{\epsilon^2})$ for ROS sketch.

One simple and effective way to construct a sketching operator which interpolates between the accurate sketching method such as $S_{\text{Gauss}}$ and the extremely fast sparse sketching $S_c$ is to make a hybrid sketch based on concatenating them together [110]:

**Definition 2.6.6.** *(Hybrid sketching.) Given the sketch size $m$ and an intermediate sketch size $m_0 \in (m, n]$, the hybrid sketch operator is defined as $S_{\text{hybrid}} = S_2 \cdot S_1 \in \mathbb{R}^{m \times n}$, where $S_1 := S_c \in \{-1, 0, +1\}^{m_0 \times n}$, $S_2 := S_{\text{Gauss}} \in \mathbb{R}^{m \times m_0}$.*

Intuitively, the hybrid sketch first speedily projects the huge dimension $n$ to a moderate dimension $m_0$ via the sparse sketch, and then apply an accurate sketching method such as sub-Gaussian sketch to project from $m_0$ to the targeted final sketch size $m$. With some proper choice of the intermediate sketch size $m_0$ for $S_1$, the authors of [110] demonstrate that with $O(\text{nnz}(A) + m^{1.5} d^3)$ time, the $S_{\text{hybrid}}$ achieves almost the same subspace embedding accuracy as sub-Gaussian sketch. Recall that computing $S_{\text{Gauss}} A$ takes $O(nmd)$ time, the hybrid sketch is provably superior in terms of computation compared to sub-Gaussian sketch in the regime of $n > \sqrt{m} d^2 \gg d$.

### 2.6.4 Statistical Suboptimality of Classical Sketched LS Estimators

Recall that we have introduced at the beginning of this section the classical form of sketched LS estimator for the ground truth vector $x^\dagger$:

$$x^{\text{cs}} = \arg\min_{x \in \mathcal{K}} \frac{1}{2} \|SAx - Sy\|_2^2,$$

as an alternative for using the original constrained least-squares estimator:

$$x^\star = \arg\min_{x \in \mathcal{K}} \frac{1}{2} \|Ax - y\|_2^2.$$

A natural question to ask is that, although the sketched version of least-squares are potentially less demanding on computation and storage, how much do we sacrifice the statistical performance on the estimation accuracy compare to the original LS estimator? Pilanci and Wainwright [3, Theorem 1] reveal that the classical sketching scheme is sub-optimal statistically. We now describe the theorem as the following:

**Theorem 2.6.7.** *(Statistical-Suboptimality of Classical Sketch.) [3, Theorem 1] Define the* *packing number $M(\mathcal{K},\delta)$ being the largest possible number of vectors $\{x_i\}_{i=1}^{M} \in \mathcal{K}$ such that $\|x_i - x_j\|_A := \|A(x_i - x_j)\|_2 > \delta, \forall j \neq i$. For any sketching operator $S \in \mathbb{R}^{m\times n}$ which satisfies:*

$$\|\mathbb{E}[S^T(SS^T)^{-1}S]\| \leq c_0\frac{m}{n}, \tag{2.75}$$

*where $c_0$ is a positive constant, then the estimation error $\|x^{\mathrm{cs}} - x^\dagger\|_A$ of the sketched LS estimator obeys the lower bound:*

$$\sup_{x^\dagger\in\mathcal{K}} \mathbb{E}_{S,w}\|x^{\mathrm{cs}} - x^\dagger\|_A^2 \geq \frac{\sigma^2}{128c_0}\frac{\log(\frac{1}{2}M(\mathcal{K},\frac{1}{2}))}{\min(m,n)} \tag{2.76}$$

Note that, as shown by Pilanci and Wainwright [3], the condition (2.75) is trivially satisfied by sub-Gaussian sketches and ROS sketches which we have introduced in previous subsection[9]. This result suggests that, in order to achieve the same statistical accuracy of the original LS estimator, the sketch size has to be $m = O(n)$. For instance, let $\mathcal{K} = \mathcal{B}$ being a ball in $\mathbb{R}^d$ with a unit radii, we have $M(\mathcal{B},\frac{1}{2}) = 2^d$, hence we have:

$$\sup_{x^\dagger\in\mathcal{B}} \mathbb{E}_{S,w}\|x^{\mathrm{cs}} - x^\dagger\|_A^2 \geq O\left(\frac{\sigma^2 d}{\min(m,n)}\right), \tag{2.77}$$

while it is well-known that the statistical error of the original LS solution can be upper bounded by:

$$\sup_{x^\dagger\in\mathcal{B}} \mathbb{E}_{w}\|x^\star - x^\dagger\|_A^2 \leq O\left(\frac{\sigma^2 d}{n}\right), \tag{2.78}$$

at the worse case. We can conclude that, the classical sketch itself can only serve as a fast way of getting an initial approximation of the least-squares solution $x^\star$, but applying it alone will not provide an accurate estimation of the ground truth $x^\dagger$. This suboptimality motivates the research of the iterative sketching technique which we are going to present in the next subsection.

### 2.6.5 Iterative Hessian Sketch

We wish to particularly highlight an inspiring iterative sketching meta-algorithm named Iterative Hessian Sketch (IHS) proposed by Pilanci and Wainwright [3] for the constrained least-squares. Let's first have a closer look on the constrained least-squares objective (we denote $x^0$

[9]Recent (not yet published) results from some researchers show that the condition (2.75) is also satisfied for Count-Sketch.

as an arbitrary vector in $\mathbb{R}^d$):

$$\begin{aligned} x^\star &= \arg\min_{x\in\mathcal{K}} \frac{1}{2}\|Ax-y\|_2^2 \\ &= \arg\min_{x\in\mathcal{K}} \frac{1}{2}\|A(x-x^0+x^0)-y\|_2^2 \\ &= \arg\min_{x\in\mathcal{K}} \frac{1}{2}\|A(x-x^0)-(y-Ax^0)\|_2^2 \\ &= \arg\min_{x\in\mathcal{K}} \frac{1}{2}(x-x^0)^TA^TA(x-x^0)-(x-x^0)^TA^T(y-Ax^0) \end{aligned}$$

which is consisted of two parts: one linear term associates with $A^T(y-Ax^0)$ – the gradient of the least-squares loss at point $x^0$, the other associates the Hessian matrix $A^TA$ of the least-squares. With some trivial algebra one can show that the classical sketch program (2.64) can be written as:

$$x^{cs} = \arg\min_{x\in\mathcal{K}} \frac{1}{2}(x-x^0)^T(SA)^TSA(x-x^0)-(x-x^0)^T(SA)^T(Sy-SAx^0). \tag{2.79}$$

In this form one can clearly see that the classical sketch performs compression on both the Hessian and gradient information. The classical sketch builds an estimation only based on the compressed information provided by $(SA, Sy)$ and leads to statistical suboptimality. It is intuitive that if a sketching scheme can be developed based on the information of $(SA, y)$ which only performs sketching on the data matrix and does not perform sketching on the observation $y$, then it may have chance to overcome this suboptimality. Consider the following form of sketched LS, which is named the *Hessian Sketch*:

$$x^{HS} = \arg\min_{x\in\mathcal{K}} \frac{1}{2m}\|SA(x-x^0)\|_2^2-(x-x^0)^TA^T(y-Ax^0). \tag{2.80}$$

Although the gradient information is preserved in a single-shot Hessian sketch, Pilanci and Wainwright [3, Proposition 1] show that it suffers from the same inefficiency of classical sketch. However, they propose the iterative hessian sketch (IHS) which constructs a sequence of hessian sketch objective function using the solution of previous sketch programs:

```
**Iterative Hessian Sketch**−Initialize $x^0 \in \mathbb{R}^d$
For   $k = 0, 1, 2, ..., K$
    Compute $\nabla f(x^t) = A^T(y-Ax^t)$ and $A_s^t = S^tA$
    $x^{t+1} = \arg\min_{x\in\mathcal{K}} \frac{1}{2m}\|A_s^t(x-x^t)\|_2^2 - \langle x-x^t, \nabla f(x^t)\rangle$;
```

In each iteration of IHS, a full gradient $\nabla f(x^t) = A^T(y - Ax^t)$ and a sketched matrix $A_s^t = S^t A$ is computed for the construction of the Hessian-sketch objective function. Assume that the sketched programs are solved exactly, the IHS scheme is able to provide a solution approximation of $\|x^t - x^\star\|_A \leq \epsilon$ in $\log \frac{1}{\epsilon}$ iterations – that is, a linear convergence rate to an arbitrary accuracy. Moreover, the convergence rate of IHS scales gracefully with the low-dimensional structure of the constrained least-squares problem. We will demonstrate the structure-adaptive convergence rate of the IHS in the next subsection.

### 2.6.6 The Structure-Adaptive Convergence Rate of Iterative Hessian Sketch

In order to formally describe the intrinsic statistical dimension of the constrained LS problem (2.61) which is enforced by the constraint set $\mathcal{K}$, we first introduce the notion of the tangent cone of the solution $x^\star$ and its associated Gaussian Width.

**Definition 2.6.8.** *(Tangent cone.) [111] The cone $\mathcal{C}_{x^\star}$ is said to be the tangent cone of $x^\star$ w.r.t the convex constraint set $\mathcal{K}$, if:*

$$\mathcal{C}_{x^\star} = \left\{ p \in \mathbb{R}^d |\, p = c(x - x^\star), \forall c \geq 0, x \in \mathcal{K} \right\}. \tag{2.81}$$

From the definition of the tangent cone associated with the solution $x^\star$, we can see that it includes all the descent directions in the constraint set towards the solution $x^\star$. If the constraint set enforces a structured solution, the size of this cone will be relatively small. Next we introduce the definition of Gaussian width measure associated with the cone:

**Definition 2.6.9.** *(Gaussian Width.) [112, 111] Let $\mathbb{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$, then the Gaussian Width of the set $\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}$ is defined as:*

$$\mathcal{W}(\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}) = E_g \left( \sup_{v \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} v^T g \right), \tag{2.82}$$

*where $g \in \mathbb{R}^n$ is draw from i.i.d. normal distribution.*

The Gaussian width effectively measures the range of directions of the cone $\mathcal{C}_{x^\star}$, and the statistical complexity of the solution $x^\star$ enforced by the constraint set $\mathcal{K}$. For example, if the constraint set is a $\ell_1$ ball $\mathcal{K} := \|x\|_1 \leq R$ where the radii $R$ is sufficiently small such that the

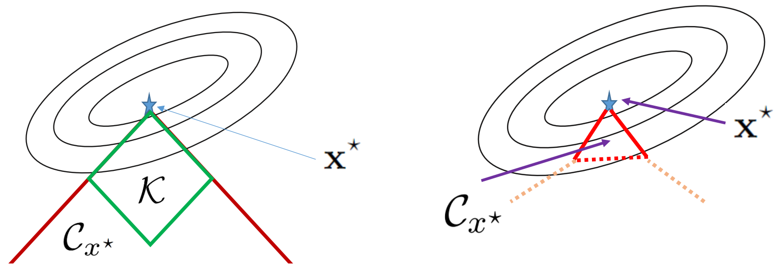


**Figure 2.5:** *Illustration of the geometry of composite optimization. Left: ERM with strict constraint $g(.) := \iota_{\mathcal{K}}(.)$; right: generic regularized ERM*

solution $x^\star$ is made to be an $s$-sparse vector, then the Gaussian width can be upper bounded as:

$$\mathcal{W}(\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}) \leq \sqrt{2s\log(\frac{d}{s}) + \frac{5}{4}s}, \tag{2.83}$$

as shown in [112].

We present here informally the convergence theorem of IHS:

**Theorem 2.6.10.** *(Convergence result for iterative hessian sketch [3]) If the sketching operators $S^t$ are sub-Gaussian sketches or ROS sketches, the solution sequence of IHS sub-programs obeys:*

$$\|x^t - x^\star\|_A \leq \left[O\left(\frac{\mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1})}{\sqrt{m}}\right)\right]^t \|x^0 - x^\star\|_A, \tag{2.84}$$

*almost surely.*

As shown by Pilanci and Wainwright [3], the Gaussian Width $\mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1})$ which associates with the transformed cone $A\mathcal{C}$ can be upper bounded by $\sqrt{d}$ and also scales proportionally to the statistical dimension of the solution. For instance, for $\ell_1$-constrained LS with an $s$-sparse solution we may have $\mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}) = O(\sqrt{s \log d})$, and then the sparser the solution is, the faster the IHS will converge towards the solution $x^\star$ for a fixed suitable sketch size $m \geq O(\mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}))$.

With the convergence result of IHS towards the LS solution $x^\star$, one can easily derive the convergence result on the estimation error regarding $x^\dagger$. Due to the triangle inequality, we have:

$$\|x^t - x^\star\|_A = \|x^t - x^\dagger - (x^\star - x^\dagger)\|_A \geq \|x^t - x^\dagger\|_A - \|x^\star - x^\dagger\|_A. \tag{2.85}$$

Then according to (2.84), we have:

$$\|x^t - x^\dagger\|_A \leq \left[O\left(\frac{\mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1})}{\sqrt{m}}\right)\right]^t \|x^0 - x^\star\|_A + \|x^\star - x^\dagger\|_A, \tag{2.86}$$

which means that the estimation error of the updates $x^t$ obtained by IHS converge exponentially towards the same statistical accuracy $\|x^\star - x^\dagger\|_A$ which is achieved by the constrained LS solution $x^\star$.

As a meta-algorithm, the iterative Hessian sketch has a desirable structure-adaptive convergence rate. However, the convergence guarantee requires we solve each of the sketched programs *exactly*, whereas in practice, it is often more tractable to solve the sketched programs approximately using an iterative algorithm. Hence a further research on combining the IHS meta-algorithm with iterative solvers such as projected gradient descent is essential for this line of research. This motivates the research work we will present in Chapter 3.

## 2.7 Exploiting the Solution's Structure for Faster First-Order Optimization

In many interesting large-scale optimization problems in machine learning and signal/image processing, the solution $x^\star$ in (2.3) has some low-dimensional structure such as sparsity [17], group-sparsity [113], low-rank [114] or piece-wise smoothness [18], enforced by the non-smooth regularization. It is intuitive that an optimal algorithm for this type of problem should take into account and exploit such solution's structure. When being utilized properly, this prior information of the solution will facilitate the convergence of an iterative algorithm.

One important theoretical cornerstone is the *restricted strong convexity* (RSC) framework presented by [5]. In the context of statistical estimation with high-dimensional data where the usual strong-convexity assumption is vacuous[10], these authors have shown that the proximal gradient descent method is able to achieve global linear convergence up to a point $x$ which satisfies $\|x - x^\star\|_2 = o(\|x^\star - x^\dagger\|_2)$, the accuracy level of *statistical precision*. Moreover, the results based on this restricted strong-convexity framework indicate that the convergence

[10]For example, for a lasso problem $x^\star \in \arg\min_{x\in\mathbb{R}^d} F(x) := \|Ax - y\|_2^2 + \lambda\|x\|_1$, if the matrix $A \in \mathbb{R}^{n\times d}$ has more columns then rows (i.e. $n < d$), then $F(x)$ cannot be strongly-convex.

rate of the proximal gradient, introduced in section 2.3.3, becomes faster when the statistical complexity of the solution is lower.

### 2.7.1 Restricted Strong-Convexity and the Solution's Structure

Let's now have a brief overview on the RSC framework and the structure-adaptive convergence result for proximal gradient descent on composite optimization provided by Agarwal et al.[5]. Note that, in order to restrict early iterations of proximal gradient method and yield a global convergence result, an extra side constraint $g(x) \leq R$ is added on (2.3) in the original work:

$$x^\star \in \arg \min_{x \in \Omega | g(x) \leq R} f(x) + \lambda g(x), \tag{2.87}$$

where $R$ is assumed to be large enough such that the solution of (2.3) remains unchanged.

#### 2.7.1.1 From Decomposability to Restricted Strong-Convexity

The work of Agarwal et al [5] focuses on a class of *decomposable* regularizer:

**Definition 2.7.1.** *(Decomposable Regularizer.) Denote $\mathcal{M}$ being a subspace in $\mathbb{R}^d$, a regularizer $g(.)$ is decomposable with respect to a subspace pair $(\mathcal{M}, \mathcal{M}^\perp)$ if it satisfies*

$$g(a + b) = g(a) + g(b), \; \forall a \in \mathcal{M}, \;\; b \in \mathcal{M}^\perp. \tag{2.88}$$

This condition is satisfied for a variety of structure-inducing regularization such as $\ell_1$ norm, $\ell_{2,1}$ norm and nuclear norm penalty[11], while for a more generalized version of it will cover the analysis priors like the total-variation semi-norm [115]. The subspace $\mathcal{M}$ is named the *model subspace* while the orthogonal subspace $\mathcal{M}^\perp$ is called the *pertubation subspace*.

This class of decomposable regularizers have a strong directional restriction ability, and is intuitively good for the structure-adaptive analysis of the proximal splitting schemes. Due to the triangle inequality, let $x^\star \in \mathcal{M}$ and a vector $v \in \mathcal{M}^\perp$, we can always have:

$$g(x^\star + v) \leq g(x^\star) + g(v). \tag{2.89}$$

[11]The nuclear-norm satisfies a slightly more complicated form of (2.88) on $(\mathcal{M}, \bar{\mathcal{M}}^\perp)$ where $\mathcal{M} \in \bar{\mathcal{M}}$.

for any convex $g$. But if further $g$ is decomposable, we have equality $g(x^\star + v) = g(x^\star) + g(v)$, which means that this regularization term enforces the maximal penalization on the perturbation direction of the subspace where the solution $x^\star$ lives in. For a decomposable regularizer $g$ and a specified subspace $\mathcal{M}$, the *subspace compatibility* [5] can be defined as:

**Definition 2.7.2.** *The subspace compatibility of a model subspace $\mathcal{M}$ is defined as:*

$$\Phi(\mathcal{M}) := \sup_{v \in \mathcal{M} \setminus \{0\}} \frac{g(v)}{\|v\|_2}, \tag{2.90}$$

*when $\mathcal{M} \neq \{0\}$.*

The subspace compatibility $\Phi(\mathcal{M})$ provides a direct link to the intrinsic dimension of subspace $\mathcal{M}$. For example if $g(.) = \|.\|_1$ and $\mathcal{M}$ is a subspace which is on a $s$-sparse support in $\mathbb{R}^d$, we will have $\Phi(\mathcal{M}) = \sqrt{s}$.

Now we are ready to present the restricted strong-convexity condition which is at the core of the analysis:

**Definition 2.7.3.** *(Restricted Strong Convexity.) With curvature parameter $\gamma$ and tolerance parameter $\tau$, the restricted strong convexity condition is defined as:*

$$f(x) - f(x') - \langle \triangledown f(x'), x - x' \rangle \geq \frac{\gamma}{2}\|x - x'\|_2^2 - \tau g^2(x - x'), \quad \forall x, x' \in \Omega. \tag{2.91}$$

Different to the usual strong-convexity assumption, the RSC actively encodes the direction restriction effect prompted by the regularization through the term $\tau g^2(x - x')$. Next we will show that, with sufficiently large regularization parameter, the RSC condition will lead to fast linear convergence rate of proximal gradient descent towards an optimal statistical accuracy, and meanwhile this convergence rate scales with the low-dimenisonal structure of the solution.

#### 2.7.1.2 Sufficient Regularization for Statistical Optimality and Effective RSC

Now recall that the ultimate goal of finding a solution $x^\star$ via solving the regularized optimization program is to give a good estimation of the minimizer of the expected risk $x^\dagger$. In order to achieve this, the regularization parameter $\lambda$ needs to be sufficiently large in order to achieve statistical optimality and avoid over-fitting – that is, the statistical error $\|x^\star - x^\dagger\|_2^2$ diminish optimally w.r.t the number of data sample $n$. Meanwhile, purely from the viewpoint of optimization, a structured solution is always the result of sufficient regularization.

Driven by this motivation, the work of Negaban et al [116] and Agarwal et at [5] jointly demonstrate that with $\lambda \geq 2g^\star(\nabla f(x^\dagger))$, one can address both of the statistical optimality and also the optimization efficiency in one go.

**Proposition 2.7.4.** *[116, Theorem 1, informal] Under the assumption of RSC, if furthermore the curvature parameter $\gamma$, tolerance parameter $\tau$ and the subspace compatibility $\Phi(\mathcal{M})$ satisfy $\tau\Phi^2(\mathcal{M}) < \frac{\gamma}{64}$, then for any optima $x^\star$, the following inequality holds:*

$$\|x^\star - x^\dagger\|_2^2 \leq O\bigg(\frac{\lambda^2}{\gamma^2}\Phi^2(\mathcal{M}) + \frac{\lambda}{\gamma}g(x^\dagger_{\mathcal{M}^\perp})\bigg), \tag{2.92}$$

*where $O(.)$ hides deterministic constants for the simplicity of notation.*

Such a bound reveals desirable properties of the regularized ERM when the range of $\lambda$ satisfies $\lambda \geq 2g^\star(\nabla f(x^\dagger))$. For instance, if $x^\dagger$ is the $s$-sparse ground truth vector of a noisy linear measurement system $y = Ax^\dagger + w$, where $w$ denotes the zero-mean sub-Gaussian noise (with variance $\sigma^2$) and the measurement matrix $A$ satisfies a certain restricted eigenvalue condition [116, 117], and we use a Lasso estimator $x^\star \in \arg\min_x \frac{1}{2n}\|Ax - y\|_2^2 + \lambda\|x\|_1$. In such a case, let $\mathcal{M}$ be a subspace in $\mathbb{R}^d$ on $s$-sparse support where $x^\dagger \in \mathcal{M}$ and hence $g(x^\dagger_{\mathcal{M}^\perp}) = 0$, then with the choice of $\lambda = O[g^\star(\nabla f(x^\dagger))]$, from this proposition we can have:

$$\|x^\star - x^\dagger\|_2^2 \leq O\bigg(\frac{\lambda^2 s}{\gamma^2}\bigg) \approx O\bigg(\frac{\sigma^2}{\gamma^2}\frac{s\log d}{n}\bigg), \tag{2.93}$$

which implies the optimal convergence of the statistical error in terms of sample size and dimension. The details of this claim are presented in [116, Corollary 2].

Now due to the sufficient regularization condition, Agarwal et al [5] show that one can derive the desirable fast structure-adaptive convergence rate result for the proximal gradient algorithm via the *Effective RSC* parameter which is defined as the following:

$$\mu_c = \gamma - 64\tau\Phi(\mathcal{M}) > 0, \tag{2.94}$$

such that for a sufficiently large regularization parameter $\lambda \geq 2g^\star(\nabla f(x^\dagger))$ which secures statistical optimality, as shown by [5, Lemma 11], one can have:

$$F(x) - F(x^\star) = f(x) + \lambda g(x) - f(x^\star) - \lambda g(x^\star) \geq \mu_c \|x - x^\star\|_2^2 - \delta^2. \tag{2.95}$$

where the slackness residual term $\delta^2$ is associated with the statistical error $\|x^\star - x^\dagger\|_2^2$. The Effective RSC lemma which we informally presented in (2.95), means that due to the directional restriction by the regularization, the objective $F$ behaves just like a strongly-convex function for the regime where $\|x - x^\star\|_2^2 \geq O(\|x^\star - x^\dagger\|_2^2)$. Since the Effective RSC parameter $\mu_c$ is associated with the model complexity quantified by the subspace-compatibility $\Phi(\mathcal{M})$, it provides a direct link to the convergence result which will depend on the solution's low-dimensional structure.

Moreover, in the same spirit of the RSC, the authors of [5] also consider a restricted smoothness (RSM) assumption, which leads to the convergence result of proximal gradient descent with a greedy step-size which also scales with the solution's intrinsic dimension.

**Definition 2.7.5.** *(Restricted Smoothness.) The restricted smoothness condition is defined with parameter $(L_u, \tau_u)$:*

$$f(x) - f(x') - \langle \nabla f(x'), x - x' \rangle \leq \frac{L_u}{2}\|x - x'\|_2^2 + \tau_u g^2(x - x'), \quad \forall x, x' \in \Omega. \tag{2.96}$$

Then with a contraction factor[12]:

$$\alpha := \left\{1 - \frac{\mu_c}{4L_u} + \frac{64\Phi^2(\mathcal{M})\tau_u}{\mu_c}\right\} / \left\{1 - \frac{64\Phi^2(\mathcal{M})\tau_u}{\mu_c}\right\}, \tag{2.97}$$

[5, Theorem 2] demonstrates that, for the proximal gradient descent algorithm[13], the number of iterations needed to achieve a objective gap error $F(x^K) - F^\star \leq \epsilon$ is[14]:

$$K \geq \frac{1}{\log(\frac{1}{\alpha})} \log \frac{F(x^0) - F^\star}{\epsilon} + O\left(\log\log\frac{1}{\epsilon}\right) \approx \frac{4L_u}{\mu_c} \log \frac{F(x^0) - F^\star}{\epsilon}, \tag{2.98}$$

which is a linear convergence rate scales with the low-dimensionality of the solution quantified in $\mu_c$. To be more specific, we present here a concrete example for sparse regression:

[12]Note that if $\tau_u \to 0$ then the RSM condition reduces to ordinary smoothness assumption, and $\alpha = 1 - \frac{\mu_c}{4L_u} \leq 1$.

[13]In this case the proximal operator is defined as $\text{prox}_g(x) := \arg\min_{y \in \Omega} g(x) + \frac{1}{2}\|x - y\|_2^2$ due to the presence of the additional side-constraint set $\Omega$ to restrict early iterations

[14]Until a statistical accuracy is achieved, see [5] for details.

**Corollary 2.7.6.** *(Quantification of Effective RSC parameter $\mu_c$.) For a lasso problem $x^\star \in \arg\min_{x\in\mathbb{R}^d} F(x) := \|Ax - y\|_2^2 + \lambda\|x\|_1$, if $\lambda \geq 2g^\star(\nabla f(x^\dagger))$ and $\|x^\star\|_0 = s$, then we have*

$$\mu_c = \gamma - 64\tau s \tag{2.99}$$

*for (2.98). Moreover if the rows of $A$ are i.i.d drawn from a distribution $\mathcal{N}(0, \Sigma)$, where the covariance matrix $\Sigma$ has smallest and largest singular value $r_{\min}(\Sigma)$ and $r_{\max}(\Sigma)$, then we can have $\gamma \geq \frac{r_{\min}(\Sigma)}{16}$ and $\tau \leq r_{\max}(\Sigma)\frac{81\log d}{n}$, such that:*

$$\mu_c \geq O\left(r_{\min}(\Sigma) - r_{\max}(\Sigma)\frac{s\log d}{n}\right) \tag{2.100}$$

*almost surely (see e.g. [117]).*

Under the same RSC framework, researchers have recently provided fast structure-inspired convergence analysis for stochastic variance-reduced algorithms such as SVRG [118], SAGA [119], and SDCA [120], revealing that such stochastic gradient methods can also automatically achieve a linear convergence rate which scales with the intrinsic dimension of the solution.

### 2.7.2 Activity Identification and Local Fast Convergence

The RSC framework introduced in the previous subsection is a powerful theoretical framework to be exploited for the structure-adaptive analysis of first-order methods, and has been sucessfully applied for deriving global linear convergence results for both deterministic and stochastic gradient methods. Nevertheless, although it provides great insights to link statistical complexity with convergence and serves as theoretical cornerstone in ours and the other related previous works, it is still imperfect in several ways, such as, the convergence guarantee is up to the statistical accuracy but not an arbitrary accuracy. Although in statistical machine learning and signal processing applications, convergence guarantees towards a statistical accuracy should be the most important, but studying the local convergence towards an arbitrary accuracy is still of great interest.

In this subsection we very briefly introduce at high-level an interesting line of work [121, 44, 122, 123] which is complementary to the RSC framework of Agarwal et al [5]. This line of research focuses on the local linear convergence rates of first-order algorithms on the convex optimization problems with non-smooth regularization terms. They show that, under certain assumptions, the proximal gradient methods, primal-dual gradient methods, and the stochastic variance-reduced gradient methods (SAGA and SVRG), can have local linear convergence after the iterate have arrived on a low-dimensional manifold[15] where the solution lies – such phenomenon is named the *activity identification*.

To be more specific, this line of work focuses on a wider class of *partially smooth* regularizers. Many popular non-smooth regularizers such as $\ell_1$ norm, nuclear norm and TV semi-norm are partially smooth functions w.r.t a Riemannian manifold. For a formal mathematical definition of the partial smoothness, we refer the interested reader to [124] for details. If the regularizer $g$ is partially smooth w.r.t a manifold $\mathcal{M}_{x^\star}$ which contains the solution $x^\star$, then under further technical assumptions[16] on the solution $x^\star$, Liang et al [121] shown that the proximal gradient descent scheme

```
For  $k = 0, 1, 2, ..., K$
    $x^{k+1} = \text{prox}_{\lambda g}^{\eta}[x^k - \eta \nabla f(x^k)]$
```

identifies the manifold $\mathcal{M}_{x^\star}$ after a finite number ($K_0$) of iterations, that is, $x^k \in \mathcal{M}_{x^\star}$ for all $k \geq K_0$. Meanwhile, under the same technical assumptions in [121], the optimization problem is shown to be strongly-convex locally – in the original work of Liang et al, it is named *local quadratic growth*. They show that, there exists a (local) strong-convexity parameter $\alpha > 0$ and a radius $r > 0$, such that:

$$F(x) - F^\star \geq \alpha \|x - x^\star\|_2^2, \quad \forall x : \|x - x^\star\|_2^2 \leq r. \tag{2.101}$$

Then they are able to show a linear local convergence of the proximal gradient method, and similar results also holds for primal-dual gradient methods [122] and some stochastic gradient methods such as proximal SVRG [123]. The local quadratic growth condition is a very nice complement to the Effective RSC condition described previously. The effective RSC (2.95) is active for the regime which is $\|x - x^\star\|_2^2 \geq O(\|x^\star - x^\dagger\|_2^2)$ – "far enough" from the solution, while in contrast the local quaratic growth is active for the regime $\|x - x^\star\|_2^2 \leq r$ which is nearby the solution.

[15]For example, for lasso regression, if the solution is $s$-sparse, then the manifold is referred as the $s$-dimensional subspace on the support set.

[16]The non-degeneracy condition and the restricted injectivity condition, we refer to [123] for details

In short, the RSC framework of Agarwal et al [5] serves as a cornerstone for the structure-adaptive convergence analysis up to the statistical accuracy, while the partial smoothness/local quadratic growth framework of Liang et al [121, 44, 122, 123] provides the mathematical foundation of local structure-adaptive convergence analysis, but such local fast convergence rate can only be established after a finite (but usually unknown) number of iterations for activity identification. This two frameworks are mutually complementary to each other.

## 2.8 Summary

In this chapter we provided the necessary background knowledge for the forthcoming technical chapters of this thesis. We started by introducing the basic and general concepts of convex composite optimization (Section 2.1 and 2.2) for machine learning and statistical inference. Then Section 2.3 introduced the classic deterministic first-order algorithms for solving the composite optimization problems, with a focus on the concepts of projection operator and proximal operator, and the proximal gradient algorithms which are able to efficiently handle the non-smooth regularization in composite optimization. All the algorithms we proposed in the technical chapters are variants of the proximal gradient scheme.

In Section 2.4 we have presented the concepts of randomization techniques in optimization and some paradigms of stochastic gradient methods and randomized coordinate descent methods which are efficient for large-scale problems. This section was aimed at giving the readers the necessary background knowledge for Chapter 4 where we proposed the Rest-Katyusha algorithm – the structure-adaptive variant of a state-of-the-art stochastic gradient algorithm (Katyusha), and Chapter 5 where we advance the accelerated coordinate descent method.

In Section 2.5 we briefly introduced the primal-dual gradient method which is able to provide flexibility and efficiency in solving the optimization problems which have multiple regularization terms and also the regularizers with linear operators (such as TV semi-norm). In Chapter 6, we utilize the flexibility of primal-dual splitting to design fast stochastic gradient methods for imaging inverse problems.

In Section 2.6 we have presented a different type of randomization technique, sketching, from the field of numerical linear algebra. We highlighted the recently introduced iterative Hessian sketch (IHS) for efficiently solving the constrained large-scale least-squares regression which is an important instant of composite optimization. If the sketched sub-problem is solved exactly, the IHS has a linear convergence which scales with the intrinsic dimension of the solution. In Chapter 3, we propose and analyze a family of sketched gradient algorithms which are the combination of the IHS and the projected gradient descent.

Finally, in Section 2.7 we described the restricted strong-convexity framework by Agarwal et al [5] which is the theoretical cornerstone for the design and the structure-adaptive convergence analysis for the randomized algorithms proposed in Chapter 4 and 5. This Section was mainly concerned about providing high-level insights and understandings about this theoretical framework. We will revisit this RSC framework formally and provide mathematical details in Chapter 4. Meanwhile, we also introduced a complementary theoretical framework proposed by Liang et al [121, 44, 123]. Although we do not apply this framework here, it is of great interest for the future work – we believe that our proposed algorithms in Chapter 4 and 5 can be also analyzed in this framework and yield accelerated linear convergence rate locally to an arbitrary accuracy.

# Chapter 3
# Sketched Gradient Algorithms for Constrained Convex Optimization

## 3.1 Sketching for Faster Constrained Optimization

Recent advances in the field of randomized algorithms have provided us with powerful tools for reducing the computation for large scale optimization. From the latest literature we can clearly see two streams of randomized algorithms, the first stream is the stochastic gradient descent (SGD) [45, 46, 47] and its variance-reduced variants [58, 125, 62, 4]. The stochastic variance-reduced gradient techniques are based on the computationally cheap unbiased estimate of the true gradients with progressively reduced estimation variance, and demonstrate state-of-the-art performance on large-scale convex optimization tasks.

This chapter follows a second line of research and uses *sketching* techniques we have introduced in chapter 2, the crux of which is reducing the dimensionality of a large scale problem by random projections (e.g., sub-Gaussian matrices, Randomized orthogonal Systems (ROS) / Fast Johnson-Lindenstrauss Transforms (FJLT) [96, 97], the Count Sketch [88], the Count-Gauss Sketch [110]) so that the resulting sketched problem becomes computationally tractable. The meta-algorithms Classical Sketch (CS)[92, 93, 2] and the Iterative Hessian Sketch (IHS) [3] have been recently introduced for efficiently solving large scale constrained LS problems which utilize the random sketching idea combined with the fact that solutions have low-dimensional structure such as sparsity in a properly-chosen dictionary, low-rank, etc. In this chapter, we propose a class of *sketched gradient algorithms* based on the sketching techniques, for the constrained least-squares regression tasks.

This chapter makes the following contributions:

- **Novel first order solvers based on iterative sketches for constrained Least-squares**

  We propose a basic first order algorithm Gradient Projection Iterative Sketch (GPIS) based on the combination of the projected gradient descent and *Iterative Hessian Sketch*

[3] for efficiently solving the constrained Least-squares, and also an accelerated variant by applying Nesterov's acceleration scheme [39, 126].

- **Theoretical analysis for the sketched gradient algorithms**

  As we have discussed in the previous chapter, although there exists established convergence result of iterative Hessian sketch [3] which describes its linear convergence rate of solution approximation under the assumption that each of the sketched programs are solved exactly, there is no theoretical analysis of the use of first order methods within this framework, where each of the sketched programs are only approximately solved. This type of convergence results are essential for the iterative Hessian sketch scheme, since in practice, the sketched subproblems of IHS can only be computationally tractable in general if we solve them approximately, via some iterative optimization algorithms. The work presented in this chapter is the first one to provide such a convergence analysis.

- **Structure-adaptive first-order algorithms via sketching**

  In related theoretical works in sketching [2, 3], convex relaxation [112], and the convergence analysis of projected gradient descent (PGD) [127] for constrained Least-squares, researchers have discovered that the low-dimensional structure of the solution enforced by the constraint set is able to be exploited to accelerate computation. In this chapter, we will show that, the sketched gradient algorithms we propose, are able to actively exploit the solution's structure of the constrained Least-squares problem for faster convergence rate, due to the dimensional-reduction properties of the randomized sketching techniques.

- **Practical implementations of the sketched gradient algorithms**

  The proposed sketched gradient algorithms draw a different line of research for first order randomized algorithms from the SGD and its recently introduced variance-reduced variants such as SVRG [58] and SAGA [62] by utilizing randomized sketching techniques and deterministic inner-loop iterations instead of the stochastic iterations. This approach leads to convenience in optimally choosing the step size by implementing line search because it follows the classical results and techniques in first order optimization. Numerically, we have designed experiments to show the computational efficiency of the sketched gradients with Count-sketch [88] and an aggressive line-search scheme for near-optimal choice of step size each iteration [39] compared to a mini-batched version of the SAGA algorithm [62] and the accelerated full gradient method [128] in large scale constrained least-square problems.

### 3.1.1 The Classical Sketch and Iterative Hessian Sketch

In this subsection we briefly recall and summarize the concepts of the classical sketch and iterative Hessian sketch we have introduced in chapter 2. Consider a constrained Least-squares regression problem in the large data setting. We have the training data matrix $A \in \mathbb{R}^{n\times d}$ with $n > d$ and observation $y \in \mathbb{R}^n$. Meanwhile we restrict our regression parameter to a convex constrained set $\mathcal{K}$ to enforce some desired structure such as sparsity and low-rank[1]:

$$x^\star = \arg\min_{x\in\mathcal{K}} \left\{ f(x) := \|y - Ax\|_2^2 \right\}. \tag{3.1}$$

Then we define the error vector $e$ as:

$$e = y - Ax^\star. \tag{3.2}$$

As we have introduced in section 2.6.2, throughout the past decade researchers proposed a basic meta-algorithm for approximately solving the Least-squares problem that we call the *Classical Sketch* (CS), see e.g. [129, 93, 2], which compresses the dimension of the Least-squares and makes it cheaper to solve. The Johnson-Lindenstrauss theory [130, 131] and the related topic of Compressed Sensing [132, 133, 134] revealed that random projections can achieve stable embeddings of high dimensional data into lower dimensions and that the number of measurements required is proportional to the intrinsic dimensionality of data (as opposed to the ambient dimension) which is manifested in the set of constraints $\mathcal{K}$. This motivates replacing the original constrained Least-squares problem with a sketched approximation [2]:

$$\hat{x} = \arg\min_{x\in\mathcal{K}} \left\{ f_0(x) := \|Sy - SAx\|_2^2 \right\}, \tag{3.3}$$

(see Section 2.6.1), where the sketching matrix $S \in \mathbb{R}^{m\times n}, m \ll n$ is a random projection operator which satisfies (2.65) and (2.75).

When the embedding dimension $m$ is larger than a certain factor of the true solution's intrinsic dimension measured through the Gaussian Width (see Definition 2.6.9) of the tangent cone $\mathcal{C}_{x^\star}$

[1]In scenarios where we do not know the exact constraint $\mathcal{K}$, we may wish to use *regularized* least-squares instead of strict constraint. This chapter focuses on the constrained case and leaves the extension for the proximal setting as future work.

which is the smallest cone containing the set $\mathcal{K} - x^\star$ (see Definition 2.6.8) – that is, if,

$$m \geq c_0 \mathcal{W}^2(A\mathcal{C}_{x^\star} \cap \mathbb{S}^n) \tag{3.4}$$

where $c_0 \geq O(1)$, then the *Classical Sketch* (3.3) ensures a robust estimation of $x^\star$ with a noise amplification factor compared to the estimator given by solving the original LS problem (3.1), and it has been shown that the smaller the embedding dimension $m$ is, the bigger the noise amplification factor will be. To get a sketching scheme for the scenarios where a high accuracy estimation is demanded, we briefly recall here, a new type of meta-algorithm *Iterative Hessian Sketch* (IHS) proposed by [3]:

$$x^{t+1} = \arg\min_{x \in \mathcal{K}} \{ f_t(x) := \frac{1}{2m} \| S^t A(x - x^t) \|_2^2 - x^T A^T (y - Ax^t) \}. \tag{3.5}$$

(See Section 2.6.5). At the $t$th iteration of IHS a new sketch of the data matrix $S^t A$ and a full gradient $A^T(y - Ax^t)$ at the current estimate $x^t$ is calculated to form a new sketched least-square problem. By repeating this procedure the IHS will converge to the solution of the original problem (3.1) in typically a small number of iterations. However, the fast convergence rate guarantee of IHS which we have presented in Theorem 2.6.10 is based on the assumption that each of the subproblems are solved exactly, while in general, it is often more practical to only solve the sketched programs approximately.

## 3.2 Gradient Projection Iterative Sketch

### 3.2.1 The Proposed Algorithms

Our proposed GPIS algorithm which we describe in Algorithm 1 applies the projected gradient descent (PGD) to solve a sequence of sketched LS, starting with a CS step for a fast initialization, and then is followed by further iterations of IHS. To be more specific, in the initialization stage of GPIS, we consider the combination of classical sketch (CS) with the PGD algorithm:

$$x_{i+1} = \mathcal{P}_{\mathcal{K}}(x_i - \eta (S^0 A)^T (S^0 A x_i - S^0 y)), \tag{3.6}$$

**Algorithm 1** Gradient Projection Iterative Sketch — $\mathcal{G}([\eta],[k])$

```
Initialization: $x_0^0 = 0, m > d$
————————- Run GPCS iterates (Optional) ————————————————-
Generate a random sketching matrix $S^0 \in \mathbb{R}^{m\times n}$ and compute $S^0A$, $S^0y$
**for** $i = 1$ **to** $k_0$ **do**
    $x_{i+1}^0 = \mathcal{P}_\mathcal{K}(x_i^0 - \eta_{0,i}(S^0A)^T(S^0Ax_i^0 - S^0y))$
**end for**
$x_0^1 = x_{k_0}^0$
————————- Run GPIHS iterates ——————————————————
**for** $t = 1$ **to** $N$ **do**
    Calculate $g = A^T(Ax_0^t - y)$
    Generate a random sketching matrix $S^t \in \mathbb{R}^{m\times n}$ and compute $A_s^t = S^tA$
    **for** $i = 1$ **to** $k_t$ **do**
        $x_{i+1}^t = \mathcal{P}_\mathcal{K}(x_i^t - \eta_{t,i}(A_s^{tT}A_s^t(x_i^t - x_0^t) + mg))$
    **end for**
    $x_0^{t+1} = x_{k_t}^t$
**end for**
**Output:** $x_0^{N+1}$
```

we refer this stage as *Gradient Projection Classical Sketch* (GPCS). Then for the main loop of GPIS, we apply PGD to solve the IHS subproblems (3.5):

$$x_{i+1} = \mathcal{P}_\mathcal{K}(x_i - \eta((S^tA)^T(S^tA)(x_i - x^t) + mA^T(Ax^t - y)). \tag{3.7}$$

We name this second stage of GPIS as the *Gradient Projection Iterative Hessian Sketch* (GPIHS).

Note that thanks to the sketching each inner iteration of GPIS is $\frac{n}{m}$ times cheaper than a full PGD iterate in terms of matrix-vector multiplication, so intuitively we can see that there is potential in Algorithm 1 to get computational gain over the standard first order solver PGD.

Since it is well-known that in convex optimization the standard first order method proximal gradient descent can be accelerated by Nesterov's acceleration scheme [39, 126, 128], our Algorithm 1 has potential to be further improved by introducing Nesterov's acceleration. Here we propose Algorithm 2 – *Accelerated Gradient Projection Iterative Sketch* (Acc-GPIS) which is based on the combination of the accelerated PGD and iterative sketching.

One of the benefits that deterministically minimising the sketched cost function can bring is that the implementation of the line-search scheme can be easy and provably reliable since the underlying sketched cost function in each iteration of the outer loop is fixed. For example [39] provides a simple line-search scheme for gradient methods to make the step size of each

**Algorithm 2** Accelerated Gradient Projection Iterative Sketch — $\mathcal{A}([\eta], [k])$

```
Initialization: $x_0^0 = 0, \tau_0 = 1, m > d$
————————- Run GPCS iterates (Optional) ————————————————————
Generate a random sketching matrix $S^0 \in \mathbb{R}^{m \times n}$ and compute $S^0 A$, $S^0 y$
**for** $i = 1$ **to** $k_0$ **do**
    $x_{i+1}^0 = \mathcal{P}_{\mathcal{K}}(z_i^0 - \eta_{0,i}(S^0 A)^T (S^0 A z_i^0 - S^0 y))$
    $\tau_i = (1 + \sqrt{1 + 4\tau_{i-1}^2})/2$
    **Extrapolate** $z_{i+1}^0 = x_{i+1}^0 + \frac{\tau_{i-1} - 1}{\tau_i}(x_{i+1}^0 - x_i^0)$
**end for**
$x_0^1 = z_0^1 = x_{k_0}^0$
————————- Run GPIHS iterates ——————————————————————————-
**for** $t = 1$ **to** $N$ **do**
    Compute $g = A^T(Ax_0^t - y)$
    Generate a random sketching matrix $S^t \in \mathbb{R}^{m \times n}$ and compute $A_s^t = S^t A$
    $\tau_0 = 1$
    **for** $i = 1$ **to** $k_t$ **do**
        $x_{i+1}^t = \mathcal{P}_{\mathcal{K}}(z_i^t - \eta_{t,i}(A_s^{t^T} A_s^t (z_i^t - x_0^t) + mg))$
        $\tau_i = (1 + \sqrt{1 + 4\tau_{i-1}^2})/2$
        **Extrapolate** $z_{i+1}^t = x_{i+1}^t + \frac{\tau_{i-1} - 1}{\tau_i}(x_{i+1}^t - x_i^t)$
    **end for**
    $x_0^{t+1} = z_0^{t+1} = x_{k_t}^t$
**end for**
**Output:** $x_0^{N+1}$
```

iteration to be nearly optimal, with rigorous convergence theory and also an explicit bound for the number of additional gradient oracle calls. The line-search scheme is described by Algorithm 3. On the other hand in the stochastic gradient literature there are no practical strategies for efficient line search in the case of constrained optimization. To the best of our knowledge, only the SAG paper [57] addresses the issue of line-search and their implementation is only for unconstrained optimization.

**Algorithm 3** line-search scheme for GPIS and Acc-GPIS — $\mathcal{L}(x_i, f_t(x), \nabla f_t(x_i), \gamma_u, \gamma_d)$ [39]

```
**Input:** update $x_i$, sketched objective function $f_t(x)$, gradient vector $\nabla f_t(x_i)$, line search parameters $\gamma_u$ and $\gamma_d$, step size of previous iteration $\eta_{i-1}$.
Define composite gradient map $m_L$:
$m_L := f_t(x_i) + (x - x_i)^T \nabla f_t(x_i) + \frac{1}{2\eta}\|x - x_i\|_2^2$
$\eta = \gamma_d \eta_{i-1}$
$x = \mathcal{P}_\mathcal{K}(x_i - \eta \nabla f_t(x_i))$
While $f_t(x) \geq m_L$ do
    $\eta = \eta / \gamma_u$
    $x = \mathcal{P}_\mathcal{K}(x_i - \eta \nabla f_t(x_i))$
End while
Return $x_{i+1} = x$ and $\eta_i = \eta$
```

## 3.3 Convergence Analysis

In this section we provide the convergence analysis of the proposed GPIS algorithm and Acc-GPIS algorithm.

### 3.3.1 General Theory

We start our theoretical analysis by specifying some necessary definitions. While we have introduced generically the concepts of strong-convexity and smoothness in Section 2.2, here we explicitly define the strong-convexity parameter $\mu$ and the smoothness parameter $L$ for the Least-squares objective we consider in this chapter. For least-squares objective, the strong-convexity parameter and smoothness parameter corresponds to the smallest and largest singular value of the Hessian matrix $A^T A$ respectively:

**Definition 3.3.1.** *The Lipschitz constant $L$ and strong convexity $\mu$ for the LS (3.1) are defined as the largest and smallest singular values of the Hessian matrix $A^T A$:*

$$\mu \|z_d\|_2^2 \leq \|A z_d\|_2^2 \leq L \|z_d\|_2^2, \tag{3.8}$$

*for all $z_d \in \mathbb{R}^d$, where $0 \leq \mu < L$ ($\mu = 0$ means the LS (3.1) is non-strongly convex).*

We also define three important quantities which we will need throughout the analysis:

**Definition 3.3.2.** *Let $\mathcal{C}_{x^\star}$ be the smallest closed cone at $x^\star$ containing the set $\mathcal{K} - x^\star$:*

$$\mathcal{C}_{x^\star} = \left\{ p \in \mathbb{R}^d | \; p = c(x - x^\star), \forall c \geq 0, x \in \mathcal{K} \right\}, \tag{3.9}$$

*$\mathbb{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$, $\mathcal{B}^d$ be the unit ball in $\mathbb{R}^d$, $z$ be an arbitrary fixed unit-norm vectors in $\mathbb{R}^n$. The contraction factors $\alpha(\eta, S^tA)$, $\rho(S^t, A)$ and $\sigma(S^t, A)$ are defined as:*

$$\alpha(\eta_t, S^tA) = \sup_{u,v \in \mathcal{B}^d} v^T(I - \eta_t A^T S^{t^T} S^t A)u, \tag{3.10}$$

$$\rho(S^t, A) = \frac{\sup_{v \in A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}} v^T(\frac{1}{m} S^{t^T} S^t - I)z}{\inf_{v \in A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}} \frac{1}{m}\|S^t v\|_2^2}, \tag{3.11}$$

$$\sigma(S^t, A) = \frac{\sup_{v \in \text{range}(A) \cap \mathbb{S}^{n-1}} \|S^t v\|_2^2}{\inf_{v \in \text{range}(A) \cap \mathbb{S}^{n-1}} \|S^t v\|_2^2}, \tag{3.12}$$

For convenience, we denote each of this terms as: $\alpha_t := \alpha(\eta_t, S^tA)$, $\rho_t := \rho(S^t, A)$ and $\sigma_t := \sigma(S^t, A)$. The contraction factor $\alpha_t$ is associated with the convergence rate for the inner loop, while $\rho_t$ is the contraction factor for the outerloop. The third contraction factor $\sigma_t$ is related to the subspace embedding accuracy of the sketching matrix $S^t$. Our theory hangs on these three factors and we will show that they can be bounded with exponentially high probabilities for Gaussian projections.

**Definition 3.3.3.** *The optimal points $x_\star^t$ of the sketch programs $f_t(x)$ are defined as:*

$$x_\star^t = \arg\min_{x \in \mathcal{K}} f_t(x). \tag{3.13}$$

*We also define a constant R that measures the largest distance from the constraint set $\mathcal{K}$ towards the solution $x^\star$, for the simplicity of the theorems:*

$$R = \sup_{x \in \mathcal{K}} \|x - x^\star\|_2^2 \tag{3.14}$$

Following [3], we use the notation $\|v\|_A = \|Av\|_2$ to describe the $A$-norm of a vector $v$ in our theory. After defining these properties we can derive our first theorem for GPIS when $f(x)$ is strongly convex, e.g, $\mu > 0$ :

**Theorem 3.3.4.** *(Linear convergence of GPIS when $\mu > 0$) For fixed step sizes $\eta_t \leq \frac{1}{\|S^tA\|_2^2}$, the following bounds hold: for t $= 0$ (the initialization loop by GPCS),*

$$\|x_0^1 - x^\star\|_A \leq (\alpha_0)^{k_0}\sqrt{\frac{L}{\mu}}\|x_0^0 - x_\star^0\|_A + 2\rho_0\|e\|_2, \tag{3.15}$$

*for* $t \geq 1$ *and* $x_0^t := x_{k_{t-1}}^{t-1}$ *(the consecutive loops by GPIHS),*

$$\|x_0^t - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t^\star\right\} \|x_0^1 - x^\star\|_A; \tag{3.16}$$

*where we denote:*

$$\rho_t^\star = (\alpha_t)^{k_t}\left[(1+\rho_t)\sqrt{\frac{L}{\mu}}\right] + \rho_t \tag{3.17}$$

From Theorem 1 we can see that when we have strong convexity, aka $\mu > 0$, by choosing a appropriate step size the GPCS loop will linearly converge to a sub-optimal solution, the accuracy of which depends on the value of $2\rho_0\|e\|_2$; and the following GPIHS iterations enjoy at a rate associated with linear convergence towards the optimal point.

When the least-squares solution is relatively consistent ($\|e\|_2$ is small), the GPCS loop will provide excellent initial convergence speed, otherwise it is not beneficial – that's why we say that the GPCS loop is optional for our GPIS / Acc-GPIS algorithm.

For the cases where the strong convexity is not guaranteed ($\mu \geq 0$) we show the sub-linear convergence rate for GPIS algorithm:

**Theorem 3.3.5.** *(Convergence guarantee for GPIS when* $\mu \geq 0$*) If we choose a fixed number* $(k)$ *of inner-loops for* $t = 1, ..., N$*, the following bounds hold: for* $t = 0$*,*

$$\|x_0^1 - x^\star\|_A \leq \sqrt{\frac{\beta L \sigma_0 R}{2k_0}} + 2\rho_0\|e\|_2, \tag{3.18}$$

*for* $t \geq 1$ *and* $x_0^t := x_k^{t-1}$

$$\|x_0^t - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t\right\} \|x_0^1 - x^\star\|_A + \frac{\max_t \sqrt{\sigma_t}}{1 - \max_t \rho_t}\sqrt{\frac{\beta L R}{2k}}, \tag{3.19}$$

*where* $\beta = 1$ *for fixed step sizes* $\eta_t = \frac{1}{\|S^t A\|_2^2}$, $\beta = \gamma_u$ *for a line search scheme described by Algorithm 3 with parameter* $\gamma_u > 1$ *and* $\gamma_d = 1$.

For the Accelerated GPIS algorithm we also prove an accelerated convergence rate:

**Theorem 3.3.6.** *(Convergence guarantee for Accelerated GPIS when* $\mu \geq 0$*) If we choose a* *fixed number (k) of inner-loops for* $t = 1, ..., N$*, the following bounds hold: for* $t = 0$ *,*

$$\|x_0^1 - x^\star\|_A \leq \sqrt{\frac{2\beta L \sigma_0 R}{(k_0+1)^2}} + 2\rho_0\|e\|_2, \tag{3.20}$$

*for* $t \geq 1$ *and* $x_0^t := x_k^{t-1}$

$$\|x_0^t - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t\right\} \|x_0^1 - x^\star\|_A + \frac{\max_t \sqrt{\sigma_t}}{1 - \max_t \rho_t}\sqrt{\frac{2\beta L R}{(k+1)^2}}, \tag{3.21}$$

*where* $\beta = 1$ *for fixed step sizes* $\eta_t = \frac{1}{\|S^t A\|_2^2}$, $\beta = \gamma_u$ *for a line search scheme described by Algorithm 3 with parameter* $\gamma_u > 1$ *and* $\gamma_d = 1$.

We include the proofs of these results in this chapter's appendix, where we use standard proof techniques for the projected gradient descent on the sketched objectives. We have discussed in section 2.3.5 that for the case $\mu > 0$, the accelerated gradients can potentially enjoy the improved linear rate $O((1 - \sqrt{\frac{\mu}{L}}))$ via a periodic restart scheme but it demands the exact knowledge of the value $\mu$ (which is often unavailable in practical setups). In our implementation for the Acc-GPIS method in the experiments, we use the adaptive *gradient restart* scheme proposed by [42].

### 3.3.2 Explicit Bounds for Gaussian Sketches

The theorems above provide us with a framework to describe the convergence of GPIS and Acc-GPIS in terms of the constants $\alpha$, $\rho$ and $\sigma$. For Gaussian sketches, these constants find explicit bounding expressions in terms of the sketch size $m$ and the complexity of the constraint cone $\mathcal{C}_{x^\star}$. For this, we use the Gaussian width argument [111]. We recall here the definition of Gaussian width we have introduced previously:

(Definition 2.6.9) The Gaussian width $\mathcal{W}(\Omega)$ is a statistical measure of the size of a set $\Omega$:

$$\mathcal{W}(\Omega) = E_g\left(\sup_{v\in\Omega} v^T g\right), \tag{3.22}$$

where $g \in \mathbb{R}^n$ is draw from i.i.d. normal distribution. The value of $\mathcal{W}(\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1})$ is an useful measure of the tightness of the structure of $x^\star$. For example, if $x^\star$ is $s$-sparse and we model the sparsity constraint using an $\ell_1$ ball, we will have $\mathcal{W}(\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}) \leq \sqrt{2s\log(\frac{d}{s}) + \frac{5}{4}s}$, which means the sparser $x^\star$ is, the smaller the $\mathcal{W}(\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1})$ will be [111]. We now quantify the bounds in our general theorems in terms of the sketch size $m$ and the Gaussian width of the transformed cone $\mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}) \leq \sqrt{d}$, and the ambient dimension of the solution domain $(d)$. Now we are ready to provide the explicit bounds for the factors $\alpha_t$, $\rho_t$ and $\sigma_t$ for the general theorems (we denotes $b_m := \sqrt{2}\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})} \approx \sqrt{m}$ [127] and $\mathcal{W} := \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1})$ for the following lemmas):

**Proposition 3.3.7.** *If the step-size* $\eta_t = \frac{1}{L(b_m+\sqrt{d}+\theta)^2}$, *sketch size* $m$ *satisfies* $b_m > \sqrt{d}$, *and the entries of the sketching matrix* $S^t$ *are i.i.d drawn from a Normal distribution, then:*

$$\alpha_t \leq \left\{ 1 - \frac{\mu}{L}\frac{(b_m - \sqrt{d} - \theta)^2}{(b_m + \sqrt{d} + \theta)^2} \right\}, \tag{3.23}$$

*with probability at least* $(1 - 2e^{-\frac{\theta^2}{2}})$.

**Proposition 3.3.8.** *If the entries of the sketching matrix* $S^t$ *are i.i.d drawn from a Normal distribution, then:*

$$\rho_t \leq \frac{m}{(b_m - \mathcal{W} - \theta)^2}\left( \frac{\sqrt{2}b_m(\mathcal{W} + \theta)}{m} + |\frac{b_m^2}{m} - 1| \right), \tag{3.24}$$

*With probability at least* $(1 - e^{-\frac{\theta^2}{2}})(1 - 8e^{-\frac{\theta^2}{8}})$.

**Proposition 3.3.9.** *If the entries of the sketching matrix* $S^t$ *are i.i.d drawn from a Normal distribution, and the sketch size* $m$ *satisfies* $b_m > \sqrt{d}$, *then:*

$$\sigma_t \leq \frac{(b_m + \sqrt{d} + \theta)^2}{(b_m - \sqrt{d} - \theta)^2} \tag{3.25}$$

*with probability at least* $(1 - 2e^{-\frac{\theta^2}{2}})$.

We include the proofs of these explicit bounds in the appendix of this chapter, where we utilize the concentration properties of the Gaussian projection matrices. We would like to point out that our bound on factor $\rho_t$ in proposition 3.3.8 has revealed that the outer-loop convergence of GPIS and Acc-GPIS relies on the Gaussian Width of the solution $x^\star$ and the choice of the sketch size $m$:

$$\rho_t \lesssim \frac{\sqrt{2}\frac{\mathcal{W}}{\sqrt{m}}}{(1 - \frac{\mathcal{W}}{\sqrt{m}})^2}. \tag{3.26}$$

We can then observe that the larger the sketch size $m$ is with respect to $\mathcal{W}$, the faster the outer loop convergence of GPIS and Acc-GPIS can be, but on the other hand we should not choose $m$ too large otherwise the inner-loop iteration become more costly – this trade-off means that there is always a sweet spot for the choice of $m$ to optimize the computation.

Our theory is conservative in a sense that it does not support a sketch size which is below the ambient dimension $d$ since the factors $\alpha_t$ and $\sigma_t$ which are related to the inner loop prohibit this. Numerically, for large data regressions ($n \gg d$) we are interested in, we observe that a choice of $m \asymp O(d)$ typically provides the best overall performance.

Although the Gaussian sketch provides us strong guarantees, due to the costly computation of dense matrix-multiplication, it is not computationally attractive in practice. In the literature of randomized numerical linear algebra and matrix sketching, people usually use the random projections with fast computational structures such as the Fast Johnson-Lindenstrauss Transform [96, 97], Count sketch [88] and Count-Gauss sketch[110], which cost $O(nd\log(d))$, $O(\mathrm{nnz}(A))$ and $O(\mathrm{nnz}(A) + m^{1.5}d^3)$ respectively. These fast sketching methods provide significant speed up in practice compared to Gaussian sketch when $n \gg d$.

## 3.4 Improved Convergence Analysis of Gradient Projection Classical Sketch

The previous section describes our convergence results for GPIS/Acc-GPIS. However the convergence rate of the inner-loop is conservative due to the technical difficulties with the analysis of iterative sketches. To be more specific, let us first recall the expression of the contraction factor $\alpha_t$ by Definition 3.3.2 which is $\alpha_t = \sup_{u,v\in\mathcal{B}^d} v^T(I - \eta_t A^T S^{t^T} S^t A)u$. In view of the proof of Theorem 3.3.4, the result for the linear convergence rate of GPIS will still hold true if we replace this compromised definition with an improved form:

$$\alpha_t = \sup_{u,v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} v^T(I - \eta_t A^T S^T S A)u \tag{3.27}$$

where the cone $\mathcal{C}_{x_\star^t}$ denotes the smallest cone at the intermediate solutions $x_\star^t$ of IHS objective $f_t(x)$, containing the set $\mathcal{K} - x_\star^t$. However, the widths of these cones associated with the intermediate solutions are usually unknown and difficult to be analyzed generically without strong assumptions. This is precisely the reason that we can only derive a compromised convergence result for the GPIS algorithm so far.

Nevertheless, we are able to provide an improved analysis on the convergence rate of the GPCS step by improving this contraction factor and also a cone-restricted strong convexity condition.

We start by defining some properties of the operator $A$, sketching scheme $S$, and its interaction with the signal model (constraint set), in a similar manner to the analysis in [127] and [2]:

**Definition 3.4.1.** *Let $\mathcal{C}_{x^\star}$ be the smallest closed cone at $x^\star$ containing the set $\mathcal{K} - x^\star$, $\mathbb{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$, $\mathcal{B}^d$ be the unit ball in $\mathbb{R}^d$, $z$ be an arbitrary fixed unit-norm vector in $\mathbb{R}^n$. The contraction factor $\alpha_\star(\eta, SA)$ and the error amplification factor $\beta(S, A)$ are defined as:*

$$\alpha_\star(\eta, SA) = \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T(I - \eta A^T S^T S A)u, \tag{3.28}$$

$$\beta(S, A) = \sup_{v \in A\mathcal{C}_{x^\star} \cap \mathcal{B}^n} v^T \frac{S^T S}{m} z. \tag{3.29}$$

For the convenience of the presentation of the result, we shall denote these two key factors defined in Def.3.4.1 as $\alpha_\star := \alpha_\star(\eta, SA)$ and $\beta := \beta(S, A)$, respectively.

**Definition 3.4.2.** *The cone-restricted strong convexity constant $\mu_c^\star$ is defined as the largest positive constant which satisfies: for all $z_c \in \mathcal{C}_{x^\star}$*

$$\|Az_c\|_2^2 \geq \mu_c^\star \|z_c\|_2^2. \tag{3.30}$$

The cone restricted strong-convexity defined here can be viewed as a recovery/inversion stability measure which ensures that the original Least-squares estimator is reliable and robust to noise: $\|x^\dagger - x^\star\|_2 \leq \frac{2\|w\|_2}{\mu_c^\star}$, see [111, Proposition 2.2]. Note that we always have $\mu_c^\star \geq \mu$ which is the ordinary strong-convexity parameter for least-squares which we have defined in Def. 3.3.1.

Now we are ready to present our main result on linear convergence of the GPCS:

**Theorem 3.4.3.** *Starting from $x_0$, if the step size $\eta$, the sketching operator $S \in \mathbb{R}^{m \times n}$ and sketch size $m$ are properly chosen such that $\alpha_\star(\eta, SA) < 1$, the following error bound holds:*

$$\|x_k - x^\star\|_2 \leq \alpha_\star^k \|x_0 - x^\star\|_2 + \frac{m\eta\beta}{1 - \alpha_\star}\|e\|_2, \tag{3.31}$$

Theorem 3.4.3 reveals that as long as the step size, the sketching matrix and sketch size are chosen properly, the updates sequence $x_1, x_2, ..., x_k$ generated by the GPCS iterations converge linearly towards the Least-squares solution $x^\star$ up to an accuracy scales with $\|e\|_2$. In the forthcoming subsection we explicitly quantify the two factors $\alpha_\star$ and $\beta$ in Theorem 3.4.3 when $S$ is a Gaussian sketch, and hence demonstrates the structure-exploiting property of the GPCS stage.

## 3.4.1 Explicit Analysis Results of GPCS for Gaussian Sketches

Theorem 3.4.3 has provided us a general framework to describe the convergence of GPCS in terms of $\alpha$ and $\beta$. We can derive expressions of these terms in terms of the Gaussian Width of the set $\mathcal{W} := \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1})$ and the sketch dimension $m$ when we choose $S$ to be a Gaussian sketching matrix.

Similar to the step size choices described in [127] (which is about the PGD convergence analysis on solving the original LS problems where $A$ is a Gaussian map), our analysis covers a greedy choice and a conservative choice of the step size $\eta$. For the greedy choice, we can bound $\alpha_\star$ as described in Lemma 3.4.4.

**Lemma 3.4.4.** *(Greedy step size) If the step-size* $\eta = \frac{1}{b_m^2 L}$, *and the entries of the sketching matrix* $S$ *are i.i.d drawn from Normal distribution, then:*

$$\alpha_\star \leq (1 - \frac{\mu_c^\star}{L})(1 + \frac{(\mathcal{W} + \theta)^2}{b_m^2}) + \frac{\sqrt{8}(\mathcal{W} + \theta)}{b_m}, \tag{3.32}$$

*with probability at least* $1 - 8e^{-\frac{\theta^2}{8}}$.

However, Lemma 3.4.4 does not ensure $\alpha < 1$ since it inherently demands a sketch size $m \gtrsim \left(\mathcal{W}\frac{L}{\mu_c^\star}\right)^2$. This sketch size requirement can be moderated, at the cost of a more conservative stepsize:

**Lemma 3.4.5.** *(Conservative step size) If the step-size* $\eta = \frac{1}{L(b_m + \sqrt{d} + \theta)^2}$, *and the entries of the sketching matrix* $S$ *are i.i.d drawn from Normal distribution, then:*

$$\alpha_\star \leq \left\{1 - \frac{\mu_c^\star}{L}\frac{(b_m - \mathcal{W} - \theta)^2}{(b_m + \sqrt{d} + \theta)^2}\right\}. \tag{3.33}$$

*with probability at least* $(1 - 2e^{-\frac{\theta^2}{2}})$.

From Lemma 3.4.5 we can see that when the conservative step size is used, the sketch size we need to ensure $\alpha < 1$ is only $m \gtrsim \mathcal{W}^2$.

**Lemma 3.4.6.** *(Bound on noise amplification factor $\beta$) If the entries of the sketching matrix $S$ are i.i.d drawn from Normal distribution, then:*

$$\beta \leq 1 + \frac{\sqrt{2}b_m(\mathcal{W} + \theta)}{m} + |\frac{b_m^2}{m} - 1|, \tag{3.34}$$

*with probability at least* $(1 - 2e^{-\frac{\theta^2}{2}})$.

Lemma 3.4.5 and 3.4.6 reveals that the sketch size $m$ has an impact on both the convergence speed and the noise amplification. For the convergence speed, the larger the sketch size $m$ w.r.t the Gaussian width $\mathcal{W}$ is, the faster this convergence can be, but on the other hand we should not choose the sketch size to be too large since each iteration will become more expensive to compute hence there exists a trade-off between sketch size and computation. The noise amplification is a decreasing function w.r.t the sketch size $m$, which means the larger sketch size we choose, the more the accuracy of the GPCS output will increase.

## 3.5 Implementation for GPIS and Acc-GPIS in Practice

In this section we describe our implementation of GPIS and Acc-GPIS algorithm in the experiments:

- **Count sketch.**

  In this section we choose the Count Sketch as our sketching method since it can be calculated in a streaming fashion and we observe that this sketching method provides the best computational speed in practice. A MATLAB implementation for efficiently applying the Count Sketch can be found in [135].

- **Line search.**

  We implement the line-search scheme given by [39] and is described by Algorithm 3 for GPIS and Acc-GPIS in our experiments with parameters $\gamma_u = 2$, and $\gamma_d = 2$. Such choice of line-search parameters simply means: whenever we find the condition $f_t(\mathcal{P}_\mathcal{K}(x_i - \eta\nabla f_t(x_i))) \leq m_L$ does not hold, we shrink the step size by a factor of 2; and then at the beginning of each iteration, we increase the step size chosen at previous iteration by a factor of 2, then do backtracking again. Hence our methods are able to ensure we use an aggressive step size safely in each iteration. This is an important advantage of the sketched gradient method since we observe that for stochastic gradient methods such as SAGA a heuristic backtracking method similar to Algorithm 3 may work but it will demand a very small $\gamma_d$ (tends to 1) otherwise SAGA may go unstable, and an aggressive choice like our $\gamma_d = 2$ is unacceptable for SAGA. Hence we see that stochastic variance-reduced gradient methods such as SAGA are unable to benefit from line-search schemes as much as our sketched gradient methods do.

- **Gradient restart for Acc-GPIS.**

  We choose an efficient restarting scheme *gradient restart* proposed by [42] which is used to ensure that the iterates of accelerated gradient methods are monotonically reducing the objective function. The authors of [42] have proposed two heuristic adaptive restart schemes - *gradient restart* and *function restart* for the accelerated gradient methods and have shown significant improvements without the need of the knowledge of the functional parameters $\mu$ and $L$. Such restart methods are directly applicable for the Acc-GPIS due to its sketched deterministic iterations. Here we choose the *gradient restart* since it achieves comparable performance in practice as *function restart* but costs only $O(d)$ operations.

## 3.6 Numerical Experiments

### 3.6.1 Settings for Environments and Baseline Algorithms

We run all the numerical experiments on a DELL laptop with 2.60 GHz Intel Core i7-5600U CPU and 1.6 GB RAM, MATLAB version R2015b.

We choose two recognized algorithms to represent the the full gradient methods and the (incremental) stochastic gradient method. For the full gradient, we choose the Accelerated projected gradient descent [128, 31] with line-search method described in Algorithm 3 and gradient restart to optimize its performance. For the stochastic gradients we choose a mini-batched version of SAGA [62] with various batch sizes ($b = 10$, $b = 50$ and $b = 100$). We use the step size suggested by SAGA's theory which is $\frac{1}{3\hat{L}}$. The code for the minibatch SAGA implementation can be found in (https://github.com/mdeff/saga). We get the estimated value for $\hat{L}$ by averaging the largest singular value of each batch (note that we do not count this in the elapsed time and epoch counts for SAGA). The sketch size of our proposed methods for each experiment are listed in Table 3.1. We use the $\ell_1$ projection operator provided by the SPGL1 toolbox [136] in the experiments.

| SYN1 | SYN2 | SYN3 | MAGIC04 | YEAR |
|---|---|---|---|---|
| 800 | 800 | 400 | 475 | 1000 |

**Table 3.1:** *Sketch sizes ($m$) for GPIS and Acc-GPIS for each experiments*

## 3.6.2 Synthetic Data Sets

We start with some numerical experiments on synthetic problems (Table 3.2) to gain some insights into the algorithms. We begin by focusing on $\ell_1$ norm constrained problems . We generate synthetic constrained least-square problems by first generating a random matrix sized $n$ by $d$, then perform SVD on it and replace the singular values with a logarithmically decaying sequence. (The details of the procedure can be found in the appendix of this chapter.) Similarly we generate a synthetic problem (Syn3) for low-rank recovery using a nuclear-norm constraint. This is also called the *multiple response regression* with a generalized form of the Least-squares:

$$X^\star = \arg \min_{\|X\|_\star \leq r} |||Y - AX|||_F^2. \tag{3.35}$$

## 3.6.3 Real Data Sets

We first run an unconstrained least-squares regression on the Year-prediction (Million-song) data set from UCI Machine Learning Repository [137] after we normalize each column of the data matrix. We use this example to demonstrate our algorithms' performance for unconstrained problems.

Next we choose Magic04 Gamma Telescope data set from [137] to generate a constrained Least-square regression problem. The original number of features for Magic04 are 10 , and we normalize each columns of the original data matrix and additional irrelevant random features in the same way as the experiments in [104, 138] to the data sets so that the regressor $x^\star$ can be chosen to select the sparse set of relevant features by again solving (3.1). For this case we first

| DATA SET | SIZE | (**)$s$ | $\frac{L}{\mu}$ | $\Phi$ |
|---|---|---|---|---|
| SYN1 | (100000, 100) | 10 | $10^7$ | I |
| SYN2 | (100000, 100) | 10 | $10^7$ | (*)U |
| SYN3 (LOW RANK) | (50000, 100) | 5 | $10^4$ | - |

**Table 3.2:** *Synthetic data set settings. (*) U denotes the dense dictionary which is a orthogonal transform. (**) s denotes sparsity or rank of the ground truth* $x_{gt}$

| DATA SET | SIZE | RFS | $\Phi$ |
|---|---|---|---|
| YEAR | (500000, 90) | 90 | - |
| MAGIC04 | (19000, 10 + 40) | 10 | I |

**Table 3.3:** *Chosen data sets for Least-square regression, RFs: number of relevant features*

precalculate the $\ell_1$-norm of the original program's solution and then set it as the radius of our $\ell_1$ constraint. The details of the real data sets can be found in Table 3.3.

### 3.6.4 Discussion

We measure the performance of the algorithms by the wall-clock time (simply using the tic toc function in MATLAB) and the epoch counts. The $y$-axis of each plot is the relative error $\log(\frac{f(x)-f(x^\star)}{f(x^\star)})$. The values below $10^{-10}$ are reported as exact solutions of the least-square problems.

In all the experiments, our methods achieve the best performance in terms of wall-clock time. We show that in many cases the sketched gradient methods can outperform leading stochastic gradient methods. Both sketched gradients and stochastic gradients can achieve reduced complexity compared to the (accelerated) full gradient method, but since the sketched method has inner-loops with deterministic iterations, the line-search scheme of the classic gradient descent method can be directly used to make each iteration's step size be near optimal, and unlike the stochastic gradient, our methods do not need to access new mini-batches from memory each iteration, which can save operational time in practice.

SAGA performs competitively in terms of epoch counts (right hand figures) which is generally achieved using a small batch size of 10. Unfortunately the additional cost of the projection per iteration can severely impact on the wall clock time performance[2]. The experiments on Syn1

[2]For the unconstrained case (Million-song data set, sized $5 \times 10^5$ by 90), we also observe that, SAGA with $b = 10$ is unattractive in wall-clock time since it does not benefit from the vectorized operation of MATLAB as

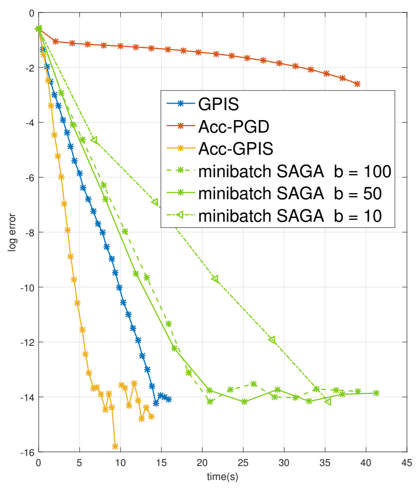


**Figure 3.1:** *Experimental results on Million-song Year prediction data set (unconstrained LS regression experiment)*

and Syn2 are similar but in Syn2 we put the constraint on a dictionary $U$, hence in Syn2 the projection operator has an additional cost of performing this orthogonal transform. In Syn1's wall-clock time plot we can see that SAGA with $b = 10$ has the fastest convergence among all the batch size choices, but in Syn2 it becomes the worst batch size choice for SAGA since it demands more iterations and hence more calls on the projection operator. In Syn3 we have a more expensive projection operator since our constraint is on the nuclear-norm of a matrix $X \in \mathbb{R}^{100\times100}$, and we can observe that the real convergence speed of SAGA with $b = 10$ is much slower than any other methods in terms of wall-clock time. In this scenario the full gradient method is much more competitive. However even here as the error reduces the sketched gradient methods exhibit a computational advantage.

## 3.7 Concluding Remarks

We propose two sketched gradient algorithms GPIS and Acc-GPIS for constrained Least-square regression tasks. We provide theoretical convergence analysis of the proposed algorithms for general sketching methods and high probability concentration bounds for the Gaussian sketches. The numerical experiments demonstrates that for dense large scale overdetermined

larger choices of batch size and takes too many iterations.

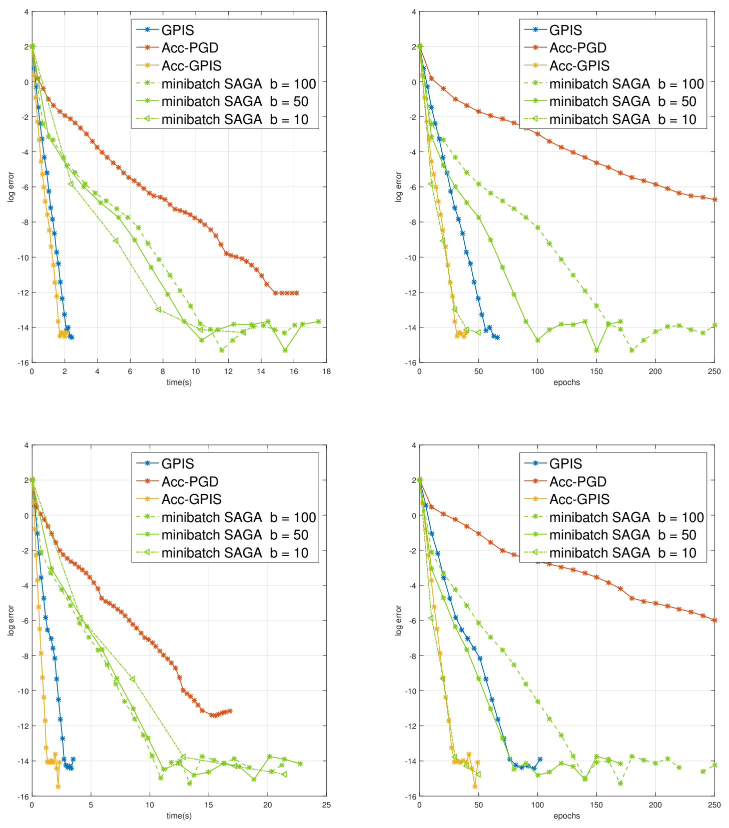


**Figure 3.2:** *Experimental results on (from top to button) Syn1 and Syn2 data sets. The left column is for wall-clock time plots, while the right column is for epoch counts*

data sets our sketched gradient methods perform very well compared to the stochastic gradient method (mini-batch) SAGA and the Accelerated full gradient method in terms of wall-clock time thanks to the benefits of sketched deterministic iterations, the efficient implementation of the Count-sketch and the use of aggressive line-search methods.

## 3.A Appendix

### 3.A.1 The Proof for Theorem 3.3.4

*Proof.* At first we denote the underlying cost function of GPIS as $f_t(x)$:

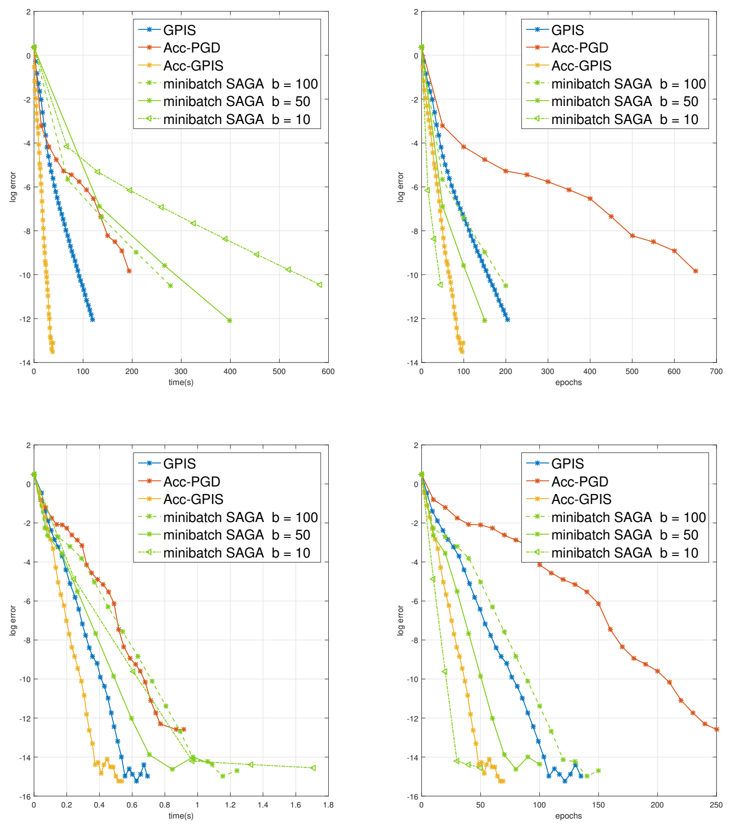


**Figure 3.3:** *Experimental results on (from top to button) Syn3 and Magic04 data sets. The left column is for wall-clock time plots, while the right column is for epoch counts*

For $t = 0$, we have the cost function of the classical sketch (CS):

$$f_0(x) := \frac{1}{2}\|Sy - SAx\|_2^2. \tag{3.36}$$

For $t = 1, 2, ..., N$ we have the the cost function of Iterative Hessian Sketch (IHS):

$$f_t(x) = \frac{1}{2}\|S^{t+1}A(x - x^t)\|_2^2 - mx^T A^T(y - Ax^t). \tag{3.37}$$

We denote the optimal solution of $f_t$ constrained to set $\mathcal{K}$ as $x_\star^t$ and $\|r_{i+1}^t\|_2 = \|x_{i+1}^t - x_\star^t\|_2$ and we have:

$$\|r_{i+1}^t\|_2 = \|x_{i+1}^t - x_\star^t\|_2 = \|\mathcal{P}_\mathcal{K}(x_i^t - \eta\nabla f(x_i)) - x_\star^t\|_2. \tag{3.38}$$

We denote cone $\mathcal{C}_{x_\star^t}$ to be the smallest close cone at $x_\star^t$ containing the set $\mathcal{K} - x_\star^t$, again because of the distance preservation of translation by Lemma 6.3 of [127], we have:

$$\begin{aligned}\|r_{i+1}^t\|_2 &= \|\mathcal{P}_{\mathcal{K}-x_\star^t}(x_i^t - \eta\nabla f(x_i) - x_\star^t)\|_2 \\ &= \sup_{v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} \left\{v^T(x_i - x_\star^t - \mu\nabla f(x_i))\right\}.\end{aligned} \tag{3.39}$$

Then because of the optimality condition on the constrained LS solution $x_\star^t$, we have:

$$\begin{aligned}\|r_{i+1}^t\|_2 &= \sup_{v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} \left\{v^T(x_i - x_\star^t - \eta\nabla f(x_i))\right\} \\ &\le \sup_{v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} \left\{v^T(x_i - x_\star^t - \eta\nabla f(x_i)) + \eta v^T\nabla f(x_\star^t)\right\} \\ &= \sup_{v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} \left\{v^T(x_i - x_\star^t) - \eta v^T(\nabla f(x_i) - \nabla f(x_\star^t))\right\} \\ &= \sup_{v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} \left\{v^T(I - \eta A^TS^TSA)r_i^t\right\} \\ &\le \sup_{u,v\in\mathcal{C}_{x_\star^t}\cap\mathcal{B}^d} \left\{v^T(I - \eta A^TS^TSA)u\right\}\|r_i^t\|_2.\end{aligned} \tag{3.40}$$

Now since $\mathcal{C}_{x_\star^t} \in \mathbb{R}^d$, we can have the following relaxation:

$$\|r_{i+1}^t\|_2 \le \sup_{u,v\in\mathcal{B}^d} \left\{v^T(I - \eta A^TS^TSA)u\right\}\|r_i^t\|_2. \tag{3.41}$$

We denote:

$$\alpha_t = \sup_{u,v\in\mathcal{B}^d} v^T(I - \eta A^TS^TSA)u. \tag{3.42}$$

By recursive subsitution we have:

$$\|r_{i+1}^t\|_2 \le \alpha_t^i\|r_0^t\|_2. \tag{3.43}$$

Suppose we run GPIHS inner loop $k_t$ times, we have:

$$\|r_{k_t+1}^t\|_2 \le \{\alpha_t\}^{k_t}\|r_0^t\|_2, \tag{3.44}$$

and consequently:

$$\|r_{k_t+1}^t\|_A \le \{\alpha_t\}^{k_t}\sqrt{\frac{L}{\mu}}\|r_0^t\|_A. \tag{3.45}$$

From the main theorems of the Classical sketch [2] and Iterative Hessian Sketch [3], we know that the following relationships hold true:

$$\|x_\star^0 - x^\star\|_A \leq 2\rho_0 \|Ax^\star - y\|_2 = 2\rho_0\|e\|_2, \tag{3.46}$$

and,

$$\|x_\star^t - x^\star\|_A \leq \rho_t \|x_0^t - x^\star\|_A. \tag{3.47}$$

Then by applying triangle inequality we can have:

$$\|x_0^1 - x^\star\|_A \leq \|x_0^1 - x_\star^0\|_A + 2\rho_0\|e\|_2, \tag{3.48}$$

and,

$$\|x_0^{t+1} - x^\star\|_A \leq \|x_0^{t+1} - x_\star^t\|_A + \rho_t\|x_0^t - x^\star\|_A. \tag{3.49}$$

Then for $t = 0$ we can have:

$$\begin{aligned}\|x_0^1 - x^\star\|_A &\leq \|x_0^1 - x_\star^0\|_A + 2\rho_0\|e\|_2 \\ &\leq \{\alpha_t\}^{k_t}\sqrt{\frac{L}{\mu}}\|x_0^0 - x_\star^0\|_A + 2\rho_0\|e\|_2,\end{aligned} \tag{3.50}$$

and for $t = 1, 2, ..., N$ we have:

$$\begin{aligned}\|x_0^t - x^\star\|_A &\leq \|x_0^t - x_\star^{t-1}\|_A + \rho_t\|x_0^{t-1} - x^\star\|_A \\ &\leq \{\alpha_t\}^{k_t}\sqrt{\frac{L}{\mu}}\|x_0^{t-1} - x_\star^{t-1}\|_A + \rho_t\|x_0^{t-1} - x^\star\|_A \\ &\leq \left\{\{\alpha_t\}^{k_t}\left((1+\rho_t)\sqrt{\frac{L}{\mu}}\right) + \rho_t\right\}\|x_0^{t-1} - x^\star\|_A.\end{aligned} \tag{3.51}$$

The last inequality holds because:

$$\begin{aligned}\|x_0^{t-1} - x_{f_{N-1}}^\star\|_A &\leq \|x_0^{t-1} - x^\star\|_A + \|x_\star^{t-1} - x^\star\|_A \\ &\leq \{1+\rho_t\}\|x_0^{t-1} - x^\star\|_A.\end{aligned} \tag{3.52}$$

Then we denote:

$$\rho_t^\star = \{\alpha_t\}^{k_t}\left((1+\rho_t)\sqrt{\frac{L}{\mu}}\right) + \rho_t, \tag{3.53}$$

and via recursive substitution we obtain:

$$\|x_0^t - x^\star\|_A \leq \left\{\prod_{t=1}^{N} \rho_t^\star\right\} \|x_0^1 - x^\star\|_A. \tag{3.54}$$

Hence we finish the proof of Theorem 1. □

### 3.A.2 The Proofs for Theorem 3.3.5 and 3.3.6

*Proof.* From the theory of the Classical sketch and Iterative Hessian Sketch we have following relationships:

$$\|x_\star^0 - x^\star\|_A \leq 2\rho_0\|Ax^\star - y\|_2 = 2\rho_0\|e\|_2, \tag{3.55}$$

and,

$$\|x_\star^t - x^\star\|_A \leq \rho_t\|x_0^t - x^\star\|_A. \tag{3.56}$$

Then by triangle inequality we have:

$$\|x_0^1 - x^\star\|_A \leq \|x_0^1 - x_\star^0\|_A + 2\rho_0\|e\|_2, \tag{3.57}$$

and,

$$\|x_0^{t+1} - x^\star\|_A \leq \|x_0^{t+1} - x_\star^t\|_A + \rho_t\|x_0^t - x^\star\|_A. \tag{3.58}$$

The remaining task of this proof is to bound the term $\|x_0^{t+1} - x_\star^t\|_A$ for both GPIS and Acc-GPIS algorithm and then chain it. For all the sketched objective functions $f_t(x)$ , $t = 0, 1, ..., N$, and any pair of vectors $x, x' \in \mathcal{K}$ we have:

$$f_t(x) - f_t(x') - \langle \nabla f_t(x'), x - x' \rangle = \|S^t A(x - x')\|_2^2 \tag{3.59}$$

If we set $x' = x_\star^t$, by using first order optimality condition [139] we immediately have:

$$\begin{aligned} & f_t(x) - f_t(x_\star^t) \geq \|S^t A(x - x_\star^t)\|_2^2 \\ = \quad & \|S^t \frac{A(x - x_\star^t)}{\|A(x - x_\star^t)\|_2} \|A(x - x_\star^t)\|_2\|_2^2 \\ \geq \quad & \left\{\inf_{v \in \text{range}(A) \cap \mathbb{S}^{n-1}} \|S^t v\|_2^2\right\} \|x - x_\star^t\|_A^2. \end{aligned} \tag{3.60}$$

Hence we have:

$$\|x - x_\star^t\|_A \leq \frac{\sqrt{f_t(x) - f_t(x_\star^t)}}{\inf_{v \in \text{range}(A) \cap \mathbb{S}^{n-1}} \|S^t v\|_2}. \tag{3.61}$$

From the established results in [128], for GPIS inner iterates we can have:

$$f_t(x_k) - f_t(x_\star^t) \leq \frac{\beta L R \sup_{v \in \text{range}(A) \cap \mathbb{S}^{n-1}} \|S^t v\|_2^2}{2k}, \tag{3.62}$$

and for Acc-GPIS inner loop we have:

$$f_t(x_k) - f_t(x_\star^t) \leq \frac{2\beta L R \sup_{v \in \text{range}(A) \cap \mathbb{S}^{n-1}} \|S^t v\|_2^2}{(k+1)^2}. \tag{3.63}$$

Hence for GPIS we have:

$$\|x_0^{t+1} - x_\star^t\|_A \leq \sqrt{\frac{\beta L \sigma_t R}{2k}}, \tag{3.64}$$

and also for Acc-GPIS:

$$\|x_0^{t+1} - x_\star^t\|_A \leq \sqrt{\frac{2\beta L \sigma_t R}{(k+1)^2}}. \tag{3.65}$$

Then by recursively substituting the inequalities we shall obtain the desired results. □

### 3.A.3 The Proofs for Quantitative Bounds of $\alpha_t$, $\rho_t$ and $\sigma_t$ for Gaussian Sketches

To prove the results in Proposition 3.3.7, 3.3.8 and 3.3.9 we need the following concentration lemmas as pillars:

**Lemma 1.** *For any $g \in \mathbb{R}^d$, we have:*

$$\sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T g = \max\left\{0, \sup_{u \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} u^T g\right\} \tag{3.66}$$

*Proof.* By the definition for cone projection operator we have:

$$\sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T g = \|\mathcal{P}_\mathcal{C}(g)\|_2 \geq 0. \tag{3.67}$$

If $\sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T g > 0$:

$$\sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T g = \sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} \|v\|_2 \frac{v^T g}{\|v\|_2} \leq \sup_{u \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} u^T g, \tag{3.68}$$

and meanwhile since $\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1} \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d$ we have:

$$\sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T g \geq \sup_{u \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} u^T g. \tag{3.69}$$

Hence we have:

$$\sup_{v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T g = \sup_{u \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} u^T g. \tag{3.70}$$

□

**Lemma 2.** *If* $\sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T M u > 0$, *we have:*

$$\sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T M u = \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} v^T M u \tag{3.71}$$

*Proof.* Since $u, v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d$, $\|u\|_2$ and $\|v\|_2$ are both less than or equal to 1, we can have the following upper bound:

$$\begin{aligned} \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T M u &= \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} \left(\frac{v^T M u}{\|v\|_2 \|u\|_2}\right) \|v\|_2 \|u\|_2 \\ &\leq \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} v^T M u, \end{aligned}$$

and meanwhile since $\mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1} \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d$ we have:

$$\sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T M u \geq \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} v^T M u. \tag{3.72}$$

Hence we have:

$$\sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathcal{B}^d} v^T M u = \sup_{u,v \in \mathcal{C}_{x^\star} \cap \mathbb{S}^{d-1}} v^T M u. \tag{3.73}$$

□

**Lemma 3.** *If the entries of the sketching matrix $S$ are i.i.d drawn from Normal distribution and $v \in \mathcal{C}_{x^\star}$, we have:*

$$\|SAv\|_2 \geq \sqrt{\mu}(b_m - \mathcal{W} - \theta)\|v\|_2, \tag{3.74}$$

$$\|SAv\|_2 \leq \sqrt{L}(b_m + \mathcal{W} + \theta)\|v\|_2, \tag{3.75}$$

*with probability at least* $1 - e^{-\frac{\theta^2}{2}}$. *(*$b_m = \sqrt{2}\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})} \approx \sqrt{m}$, $\mathcal{W} := \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1})$*)*

*Proof.* This Lemma follows the result of the simplified form of the Gordon's Lemma [Lemma

6.7][127]:

$$\begin{aligned}
\|SAv\|_2 &\geq (b_m - \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}) - \theta)\|Av\|_2 \\
&\geq \sqrt{\mu}(b_m - \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}) - \theta)\|v\|_2
\end{aligned}$$

$$\begin{aligned}
\|SAv\|_2 &\leq (b_m + \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}) + \theta)\|Av\|_2 \\
&\leq \sqrt{L}(b_m + \mathcal{W}(A\mathcal{C}_{x^\star} \cap \mathbb{S}^{n-1}) + \theta)\|v\|_2
\end{aligned}$$

□

#### 3.A.3.1 The Proof for Proposition 3.3.7

*Proof.* Let's mark out the feasible region of the step-size $\eta$:

$$\begin{aligned}
\alpha(\eta, S^t A) &= \sup_{u,v \in \mathcal{B}^d} v^T(I - \eta A^T S^T SA)u \\
&\geq \sup_{v \in \mathcal{B}^d} v^T(I - \eta A^T S^T SA)v \\
&= \sup_{v \in \mathcal{B}^d} (\|v\|_2^2 - \eta\|SAv\|_2^2) \\
&\geq \sup_{v \in \mathcal{B}^d} ((1 - \eta L(b_m + \sqrt{d} + \theta - \epsilon)^2)\|v\|_2^2).
\end{aligned}$$

So if we choose a step size $\eta \leq \frac{1}{L(b_m+\sqrt{d}+\theta)^2}$ we can ensure that with probability $1-e^{-\frac{(\theta-\epsilon)^2}{2}}(\epsilon > 0)$ we have $\alpha(\eta, S^t A) > 0$ and the Lemma 2 become applicable:

$$\begin{aligned}
&\alpha(\eta, S^t A) \\
= &\sup_{u,v \in \mathcal{B}^d} v^T(I - \eta A^T S^T SA)u \\
= &\sup_{u,v \in \mathbb{S}^{d-1}} v^T(I - \eta A^T S^T SA)u \\
= &\sup_{u,v \in \mathbb{S}^{d-1}} \frac{1}{4}[(u+v)^T(I - \eta A^T S^T SA)(u+v) - (u-v)^T(I - \eta A^T S^T SA)(u-v)] \\
= &\sup_{u,v \in \mathbb{S}^{d-1}} \frac{1}{4}[\|u+v\|_2^2 - \eta\|SA(u+v)\|_2^2 - \|u-v\|_2^2 + \eta\|SA(u-v)\|_2^2] \\
\leq &\sup_{u,v \in \mathbb{S}^{d-1}} \frac{1}{4}[(1 - \eta\mu(b_m - \sqrt{d} - \theta)^2)\|u+v\|_2^2 + (\eta L(b_m + \sqrt{d} + \theta)^2 - 1)\|u-v\|_2^2].
\end{aligned}$$

The last line of inquality holds with probability at least $1-2e^{-\frac{\theta^2}{2}}$ according to Lemma 3. Then since we have set $\eta \leq \frac{1}{L(b_m+\sqrt{d}+\theta+\epsilon)^2}$, and meanwhile notice the fact that $\|u+v\|_2^2 \leq 4$ we have:

$$\begin{aligned}\alpha(\eta, S^tA) &\leq \sup_{u,v\in\mathbb{S}^{d-1}} \frac{1}{4}(1-\eta\mu(b_m-\sqrt{d}-\theta)^2\|u+v\|_2^2 \\ &\leq (1-\eta\mu(b_m-\sqrt{d}-\theta)^2)\end{aligned}$$

If we chose $\eta = \frac{1}{L(b_m+\sqrt{d}+\theta)^2}$ we have:

$$\alpha(\eta, S^tA) \leq \left(1-\frac{\mu}{L}\frac{(b_m-\sqrt{d}-\theta)^2}{(b_m+\sqrt{d}+\theta)^2}\right). \tag{3.76}$$

Then let $\epsilon \to 0$, we shall get the result shown in Proposition 3.3.7. □

#### 3.A.3.2 The Proof for Proposition 3.3.8

*Proof.* Recall that $\rho_t$ is defined as:

$$\rho(S^t, A) = \frac{\sup_{v\in A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}} v^T(\frac{1}{m}S^{t^T}S^t - I)z}{\inf_{v\in A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}} \frac{1}{m}\|S^tv\|_2^2}. \tag{3.77}$$

We start by lower-bounding the denominator. Using the simplified Gordon's lemma [Lemma 6.7][127] we directly have:

$$\inf_{v\in A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}} \frac{1}{m}\|Sv\|_2^2 \geq \frac{(b_m-\mathcal{W}-\theta)^2}{m}, \tag{3.78}$$

with probability at least $(1-e^{-\frac{\theta^2}{2}})$.Then we move to the upper bound for the numerator:

$$\begin{aligned}v^T\left(\frac{S^{t^T}S^t}{m}-I\right)z &= \frac{1}{4}\{(v+z)^T(\frac{S^{t^T}S^t}{m}-I)(v+z)-(v-z)^T(\frac{S^{t^T}S^t}{m}-I)(v-z)\} \\ &= \frac{1}{4}\{\frac{1}{m}\|S^t(v+z)\|_2 - \|v+z\|_2 + \|v-z\|_2 - \frac{1}{m}\|S^t(v-z)\|_2\},\end{aligned} \tag{3.79}$$

and,

$$\begin{aligned}\mathcal{W}(A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}-z) &= \mathbb{E}_g(\sup_{v\in A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}} g^T(v-z)) \\ &= \mathbb{E}_g(g^Tz+\sup_{v\in A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}} v^Tg) \\ &= \mathcal{W}(A\mathcal{C}_{x^\star}\cap\mathbb{S}^{n-1}).\end{aligned} \tag{3.80}$$

Hence we have the following by [Lemma 6.8][127]:

$$\begin{aligned}v^T\left(\frac{S^{t^T}S^t}{m}-I\right)z &\le \frac{1}{4}\left\{\frac{1}{m}(b_m\|v+z\|_2+\mathcal{W}+\theta)^2-\|v+z\|_2^2\right\} \\ &+\frac{1}{4}\left\{\frac{1}{m}(b_m\|v-z\|_2+\mathcal{W}+\theta)^2-\|v-z\|_2^2\right\} \\ &=\frac{1}{4}\left\{(\frac{b_m^2}{m}-1)\|v+z\|_2^2+\frac{2b_m(\mathcal{W}+\theta)}{m}\|v+z\|_2\right\} \\ &+\frac{1}{4}\left\{(1-\frac{b_m^2}{m})\|v-z\|_2^2+\frac{2b_m(\mathcal{W}+\theta)}{m}\|v-z\|_2\right\},\end{aligned} \tag{3.81}$$

with probability at least $(1-8e^{-\frac{\theta^2}{8}})$. Note that $\|v+z\|_2+\|v-z\|_2\le 2\sqrt{2}$ and $\|v+z\|_2^2+\|v-z\|_2^2\le 4$, we have:

$$\begin{aligned}v^T\left(\frac{S^{t^T}S^t}{m}-I\right)z &\le \frac{2b_m(\mathcal{W}+\theta)}{m}\frac{\|v+z\|_2+\|v-z\|_2}{4}+|\frac{b_m^2}{m}-1| \\ &\le \frac{\sqrt{2}b_m(\mathcal{W}+\theta)}{m}+|\frac{b_m^2}{m}-1|\end{aligned} \tag{3.82}$$

thus finishes the proof. □

### 3.A.3.3 The Proof for Proposition 3.3.9

*Proof.* Recall that $\sigma_t$ is defined as:

$$\sigma(S^t,A)=\frac{\sup_{v\in\text{range}(A)\cap\mathbb{S}^{n-1}}\|S^tv\|_2^2}{\inf_{v\in\text{range}(A)\cap\mathbb{S}^{n-1}}\|S^tv\|_2^2}. \tag{3.83}$$

By simply applying again the Gordon's lemma [Lemma 6.7][127], with $\mathcal{W}(A\mathbb{S}^{d-1})\le\sqrt{d}$, we obtain the upper bound on the numerator:

$$\sup_{v\in\text{range}(A)\cap\mathbb{S}^{n-1}}\|S^tv\|_2^2\le(b_m+\sqrt{d}+\theta)^2, \tag{3.84}$$

and the lower bound:

$$\inf_{v\in \mathrm{range}(A)\cap \mathbb{S}^{n-1}} \|S^t v\|_2^2 \geq (b_m - \sqrt{d} - \theta)^2, \tag{3.85}$$

both with probability at least $1 - e^{-\frac{\theta^2}{2}}$. □

### 3.A.4 The Proof for Theorem 3.4.3

We can start by:

$$\|h_{i+1}\|_2 = \|x_{i+1} - x^\star\|_2 = \|\mathcal{P}_\mathcal{K}(x_i - \eta(A^T S^{t^T} S^t A x_i - A^T S^{t^T} S^t y)) - x^\star\|_2.$$

Because of the distance preservation of translation [127, Lemma 6.3], we have:

$$\overset{(a)}{=} \quad \|\mathcal{P}_{\mathcal{K}-x^\star}(x_i - x^\star - \eta(A^T S^{t^T} S^t A x^i - A^T S^{t^T} S^t y))\|_2.$$

We apply [127, Lemma 6.4], where $\mathcal{C}_{x^\star}$ is the smallest close cone containing the set $\mathcal{K} - x^\star$:

$$\begin{aligned}
&\overset{(b)}{\leq} \quad \|\mathcal{P}_{\mathcal{C}_{x^\star}}(x_i - x^\star - \eta(A^T S^{t^T} S^t A x^i - A^T S^{t^T} S^t y))\|_2 \\
&= \quad \|\mathcal{P}_{\mathcal{C}_{x^\star}}(h_i + \eta A^T S^{t^T} S^t y - \eta(A^T S^{t^T} S^t A(h_i - x^\star))\|_2 \\
&= \quad \|\mathcal{P}_{\mathcal{C}_{x^\star}}((I - \eta A^T S^{t^T} S^t A)h_i + \eta A^T S^{t^T} S^t (y - Ax^\star)\|_2 \\
&= \quad \|\mathcal{P}_{\mathcal{C}_{x^\star}}((I - \eta A^T S^{t^T} S^t A)h_i + \eta A^T S^{t^T} S^t e\|_2.
\end{aligned}$$

Then because of the definition of the cone-projection operator [127, Lemma 6.2] we have:

$$\begin{aligned}
&\overset{(c)}{=} \quad \sup_{v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d} v^T[(I - \eta A^T S^{t^T} S^t A)h_i + \eta A^T S^{t^T} S^t e)] \\
&\leq \quad \{\sup_{v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d} v^T(I - \eta A^T S^{t^T} S^t A)h_i \\
&\qquad +\eta \sup_{v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d} v^T(A^T S^{t^T} S^t e))\}.
\end{aligned}$$

Next we tidy up these terms by the definition of $\alpha(\eta, S^tA)$ and $\beta(S^t, A)$ in Definition 1:

$$\begin{aligned}
&\leq \{\sup_{v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d}(v^T(I-\eta A^TS^{t^T}S^tA)\frac{h_i}{\|h_i\|_2})\|h_i\|_2 + \eta\sup_{v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d}(v^TA^TS^{t^T}S^t\frac{e}{\|e\|_2})\|e\|_2)\} \\
&\leq \{\sup_{u,v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d}(v^T(I-\eta A^TS^{t^T}S^tA)u)\|h_i\|_2 + \eta\sup_{v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d}(v^TA^TS^{t^T}S^t\frac{e}{\|e\|_2})\|e\|_2)\} \\
&\leq \{\sup_{u,v\in\mathcal{C}_{x^\star}\cap\mathcal{B}^d}(v^T(I-\eta A^TS^{t^T}S^tA)u)\|h_i\|_2 + \eta L\sup_{v\in A\mathcal{C}_{x^\star}\cap\mathcal{B}^n}(v^TS^{t^T}S^t\frac{e}{\|e\|_2})\|e\|_2)\} \\
&= \alpha_\star(\eta, S^tA)\|h_i\|_2 + \eta mL\beta(S^t, A)\|e\|_2.
\end{aligned}$$

Then we do recursive substitution:

$$\begin{aligned}
\|h_{i+1}\|_2 &\leq \alpha_\star^i(\eta, S^tA)\|h_0\|_2 + m\eta\beta(S^t, A)\frac{1-\alpha_\star^i(\eta, S^tA)}{1-\alpha_\star(\eta, S^tA)}\|e\|_2 \\
&\leq \alpha_\star^i(\eta, S^tA)\|h_0\|_2 + \frac{m\eta L\beta(S^t, A)}{1-\alpha_\star(\eta, S^tA)}\|e\|_2.
\end{aligned}$$

Thus finishes the proof.

### 3.A.5 The Proofs for the Explicit Bounds of GPCS

#### 3.A.5.1 Proof for Lemma 3.4.4

From the results in [127, Lemma 6.8] we can have the following bounds with probability at least $1 - 4e^{-\frac{\theta^2}{8}}$:

$$\begin{aligned}
\|SA(u+v)\|_2 &\geq b_m\|A(u+v)\|_2 - 2\sqrt{\mu_c^\star}(\mathcal{W}+\theta) \\
&\geq \sqrt{\mu_c^\star}(b_m\|u+v\|_2 - 2(\mathcal{W}+\theta)),
\end{aligned}$$

$$\begin{aligned}
\|SA(u-v)\|_2 &\leq b_m\|A(u-v)\|_2 + 2\sqrt{L}(\mathcal{W}+\theta) \\
&\leq \sqrt{L}(b_m\|u-v\|_2 + 2(\mathcal{W}+\theta)).
\end{aligned}$$

Then we can have:

$$\begin{aligned}
\alpha_\star &\leq \frac{1}{4}\left\{\|u+v\|_2^2 - \eta\|SA(u+v)\|_2^2 - \|u-v\|_2^2 + \eta\|SA(u-v)\|_2^2\right\} \\
&\leq \frac{1}{4}\{\|u+v\|_2^2 - \eta\mu_c^\star(b_m\|u-v\|_2 - 2(\mathcal{W}+\theta))^2 \\
&\quad -\|u-v\|_2^2 + \eta L(b_m\|u-v\|_2 + 2(\mathcal{W}+\theta))^2\} \\
&\leq \frac{1}{4}\{\|u+v\|_2^2 - \eta\mu_c^\star(b_m^2\|u-v\|_2^2 + 4(\mathcal{W}+\theta)^2 - 2b_m(\mathcal{W}+\theta)\|u+v\|_2) - \|u-v\|_2^2 \\
&\quad +\eta L(b_m\|u-v\|_2 + 4(\mathcal{W}+\theta)^2 + 2b_m(\mathcal{W}+\theta)\|u-v\|_2)\} \\
&\leq \frac{1}{4}\{(1-\eta\mu_c^\star b_m^2)\|u+v\|_2^2 + (\eta L b_m^2 - 1)\|u-v\|_2^2 + 4\eta(\mathcal{W}+\theta)^2(L-\mu_c^\star) \\
&\quad +2\eta b_m(\mathcal{W}+\theta)(L\|u-v\|_2 + \mu_c^\star\|u+v\|_2)\}.
\end{aligned}$$

Now we set $\eta = \frac{1}{b_m^2 L}$. Since $L \geq \mu_c^\star$, $\|u+v\|_2^2 \leq 4$, $\|u-v\|_2^2 \leq 4$ and $\|u-v\|_2 + \|u+v\|_2 \leq 2\sqrt{2}$, we have:

$$\begin{aligned}
\alpha_\star &\leq \frac{1}{4}\{4(1-\frac{\mu_c^\star}{L}) + \frac{4(\mathcal{W}+\theta)^2}{b_m^2}(1-\frac{\mu_c^\star}{L}) + \frac{8\sqrt{2}(\mathcal{W}+\theta)}{b_m}\} \\
&= (1-\frac{\mu_c^\star}{L})(1+\frac{(\mathcal{W}+\theta)^2}{b_m^2}) + \frac{2\sqrt{2}(\mathcal{W}+\theta)}{b_m},
\end{aligned}$$

with probability at least $1 - 8e^{-\frac{\theta^2}{8}}$. Thus finishes the proof.

### 3.A.6 The Proof for Lemma 3.4.5

In this proof we first state two inequalities which are extented the from [127, Lemma 6.7]:

$$\|SA(u+v)\|_2 \geq \sqrt{\mu_c^\star}(b_m - \mathcal{W} - \theta)\|u+v\|_2, \tag{3.86}$$

$$\|SA(u-v)\|_2 \leq \sqrt{L}(b_m + \sqrt{d} + \theta)\|u-v\|_2, \tag{3.87}$$

with probability at least $1 - e^{-\frac{\theta^2}{2}}$. Then we can have:

$$\begin{aligned}
\alpha_\star &\leq \frac{1}{4}\left\{\|u+v\|_2^2 - \eta\|SA(u+v)\|_2^2 - \|u-v\|_2^2 + \eta\|SA(u-v)\|_2^2\right\} \\
&\leq \frac{1}{4}\{\|u+v\|_2^2 - \eta\mu_c^\star(b_m - \mathcal{W} - \theta)^2\|u+v\|_2^2 \\
&\quad -\|u-v\|_2^2 + \eta L(b_m + \sqrt{d} + \theta)^2\|u-v\|_2^2\}.
\end{aligned}$$

| DATA SET | SIZE | S | $\Phi$ |
|---|---|---|---|
| SYN4 | (20000, 100) | - | I |

**Table 3.4:** *Synthetic data set for step size experiment*

Let $\eta = \frac{1}{L(b_m+\sqrt{d}+\theta)^2}$ we have:

$$\begin{aligned}\alpha_\star &\leq \frac{1}{4}\{1-\frac{\mu_c^\star(b_m-\mathcal{W}-\theta)^2}{L(b_m+\sqrt{d}+\theta)^2}\}\|u+v\|_2^2 \\ &\leq \{1-\frac{\mu_c^\star(b_m-\mathcal{W}-\theta)^2}{L(b_m+\sqrt{d}+\theta)^2}\}\end{aligned}$$

with probability at least $(1-2e^{-\frac{\theta^2}{2}})$.

The proof for Lemma 3.4.6 is almost the same as the proof of Proposition 3.3.8 hence we have not included it here.

### 3.A.7 Details of the Implementation and Numerical Experiments

#### 3.A.7.1 Procedure to Generate Synthetic Data Sets

The procedure we used to generate a constrained least-square problem sized $n$ by 100 with approximately $s$-sparse solution and a condition number $\kappa$ strictly follows:

1) Generate a random matrix $A$ sized $n$ by 100 with i.i.d entries drawn from $\mathcal{N}(0,1)$.

2) Calculate $A$'s SVD: $A = U\Sigma V^T$ and replace the singular values $diag(\Sigma)_i$ by a sequence:

$$diag(\Sigma)_i = \frac{diag(\Sigma)_{i-1}}{\kappa^{\frac{1}{d}}} \tag{3.88}$$

3) Generate the "ground truth" vector $x^\dagger$ sized 100 by 1 randomly with only $s$ non-zero entries in an orthogonal transformed domain $\Phi$, and calculate the $\ell_1$ norm of it ($r = \|\Phi x^\dagger\|_1$). Hence the constrained set can be described as $\mathcal{K} = \{x : \|\Phi x\|_1 \leq r\}$.

4) Generate a random error vector $w$ with i.i.d entries such that $\frac{\|Ax^\dagger\|_2}{\|w\|_2} = 10$.

5) Set $y = Ax^\dagger + w$

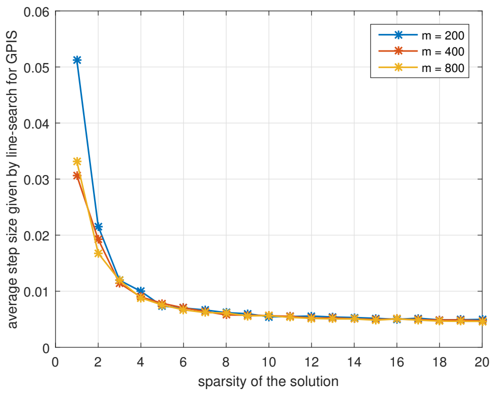


**Figure 3.4:** *Experimental results on the average choices of GPIS's step sizes given by line-search scheme*

| DATA SET | SIZE | (*)$s$ | $\frac{L}{\mu}$ |
|---|---|---|---|
| SYN-SLI | (100000, 1500) | 50 | $10^6$ |
| SYN1 | (10000, 100) | 5 | $10^6$ |
| SYN2 | (100000, 100) | 5 | $10^6$ |
| SYN3 | (200000, 100) | 5 | $10^6$ |

**Table 3.5:** *Experiment settings. (*) s denotes sparsity of the ground truth* $x^\star$

### 3.A.7.2 Extra Experiment for Step Size Choice

We explore the step size choices the GPIS algorithm produced through using the line-search scheme with respect to different sparsity levels of the solution. The results we shown come from the average of 50 random trials.

The result of the step-size simulation demonstrates that the step sizes chosen on average by the line-search scheme for the GPIS algorithm is actually related with the sparsity of the ground truth $x_{gt}$: at a regime when the $x_{gt}$ is sparse enough, the step size one can achieve goes up rapidly w.r.t the sparsity. While in our Proposition 2 we revealed that the outerloop of GPIS/Acc-GPIS can benefit from the constrained set, and here surprisingly we also find out numerically that the inner loops can also benefit from the constrained set by aggressively choosing larger step sizes. Such a result echos the analysis of the PGD algorithm on constrained Least-squares with a Gaussian map $A$ [127]. Further experiments and theoretical analysis of such greedy step sizes for sketched gradients and full gradients on general maps is of great interest.

### 3.A.8 Additional Numerical Experiments on GPCS

We first test the behaviour of GPCS algorithm with various sketch sizes in sparse linear inversion task which recovers a sparse vector $x^{\dagger}$ from noiseless measurements $y = Ax$. The $\ell_1$ norm of $x^{\dagger}$ is assumed known as a prior hence hence we construct the constraint set $\mathcal{K} = \left\{\forall v : \|v\|_1 \leq \|x^{\dagger}\|_1\right\}$.

The details of the experimental setting can be found in Table 3.5, and meanwhile the procedure of generating the synthetic data matrix A with a condition number $\kappa$:

1) Generate a random matrix $A$ sized $n$ by $d$ using MATLAB command Randn.

2) Compute the Singular Value Decomposition of $A$: $A = U\Sigma V^T$ and modify the singular values $e_i = diag(\Sigma)_i$ by:

$$e_i = \frac{e_{i-1}}{\kappa^{\frac{1}{d}}}, \tag{3.89}$$

to achieve the condition number of $\frac{L}{\mu} = 10^6$.

The performance of the Projected Gradient and fast gradient method (FISTA) for SYN-SLI is shown in figure 3.5. The step sizes for all the algorithms are generated by the line-search given by [39]. Although in this work we analyze the explicit convergence speed and noise amplification for the Gaussian Sketch as a motivational theory, this type of sketches are costly to compute. In practice, instead of using directly the Gaussian sketch, people use faster sketches such as the Fast Johnson-Lindenstrauss Transform (FJLT) [96][97], the sparse JLT [105] and the Count Sketch [88]. We choose to use the Count-Sketch [88] for the GPCS to speedily produce the sketched matrix $SA$.

This experimental result confirms that for noiseless inversion $y = Ax^{\dagger}$, the GPCS converges towards the solution with a linear rate as our theory predicts, and also the best convergence is given by appropriate median sketch size choices ($m = 700, 1100$), which are some factors larger than the sparsity $s = 50$, but less than the ambient dimension $d = 1500$. The GPCS provides significant computational benefits over the full gradient methods PGD and FISTA on this large scale inversion task.

We then turn to the noisy set up for SYN-SLI $y = Ax^{\dagger} + w$ (Fig. 3.6). The Gaussian noise vector $w$ satisfies $\frac{\|Ax^{\dagger}\|_2}{\|w\|_2} = 30$. Here we find out that as our theory predicted, the GPCS converges to a vicinity of the solution $x^{\star}$, meanwhile the larger the sketch size is, the more

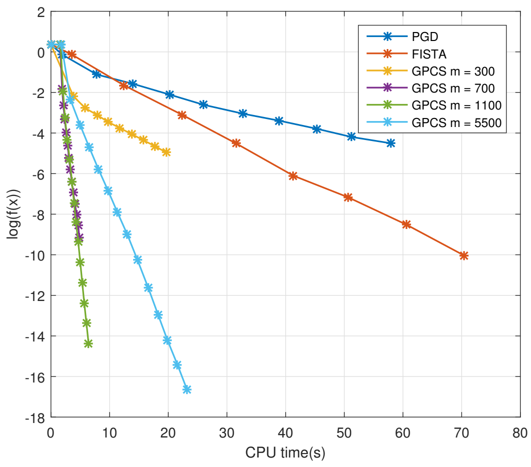


**Figure 3.5:** *Large scale noiseless sparse linear inversion experiment (Syn-SLI),* $y = Ax^{\dagger}$

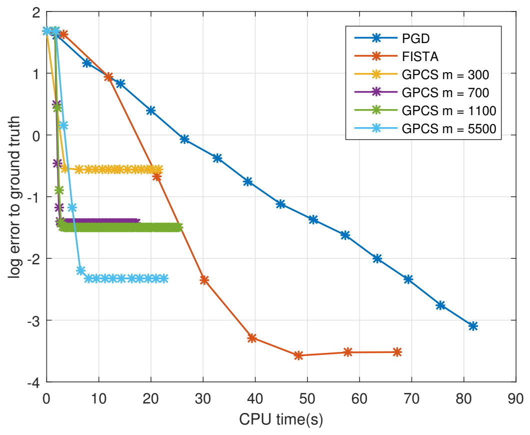


**Figure 3.6:** *Large scale noisy Least-squares regression experiment (Syn-SLI),* $y = Ax^{\dagger} + w$

accurate the solution is. That is, the GPCS converge to an approximated solution of the Least-squares, and the approximation accuracy is determined by the noise level and the sketch size $m$. As discussed, if one needs to solve the LS to machine precision in the presence of noise $\|w\|_2 > 0$, one can use the GPCS as a fast initialization step and then continue by GPIHS which is based on the "iterative" sketches [3].

### 3.A.9 Computational and Sketch-Size Trade-off

We finally examine the computational cost of GPCS on the linear inversion task $y = Ax$ through different choices of sketch size on three synthetic examples (Syn1, Syn2 and Syn3). Here we test GPCS on both the Gaussian Sketch and the Count-Sketch which is more computationally efficient. We run GPCS with sketch size from 10 to 1000 until a fixed accuracy $\|x^t - x^\star\|_2 \leq 10^{-4}$ is achieved, or when the computational budget has run out. We average the results of 20 random trials.

From the experimental results in Figure 3.7 we can observe a sharp trade-off phenomenon as our theory predicted. When the sketch size is too near to the intrinsic dimension, the GPCS takes a huge cost to achieve the targeted accuracy; but if we increase the sketch size by a small amount, the computational cost drops radically to a ”sweet spot”, then if we continue to increase the sketch size, the cost increases again since each iteration’s cost is more expensive for a large sketch size. We can also observe that the optimal choice of sketch size is not a function of the data sample size $n$, as our theory predicted. Surprisingly we see that the larger the $n$ is, the behavior of the Gaussian Sketch and Count Sketch become more similar as shown in the right hand figures in Fig 3.7. It would be an interesting future research direction to investigate in theory the relationship between the performance of the practical sparse embedding schemes such as the Count Sketch, and the properties of $A$, e.g. the sample size $n$, the parameter dimension $d$ and the conditioning $\frac{L}{\mu}$, the distribution of its singular values and singular vectors, etc.

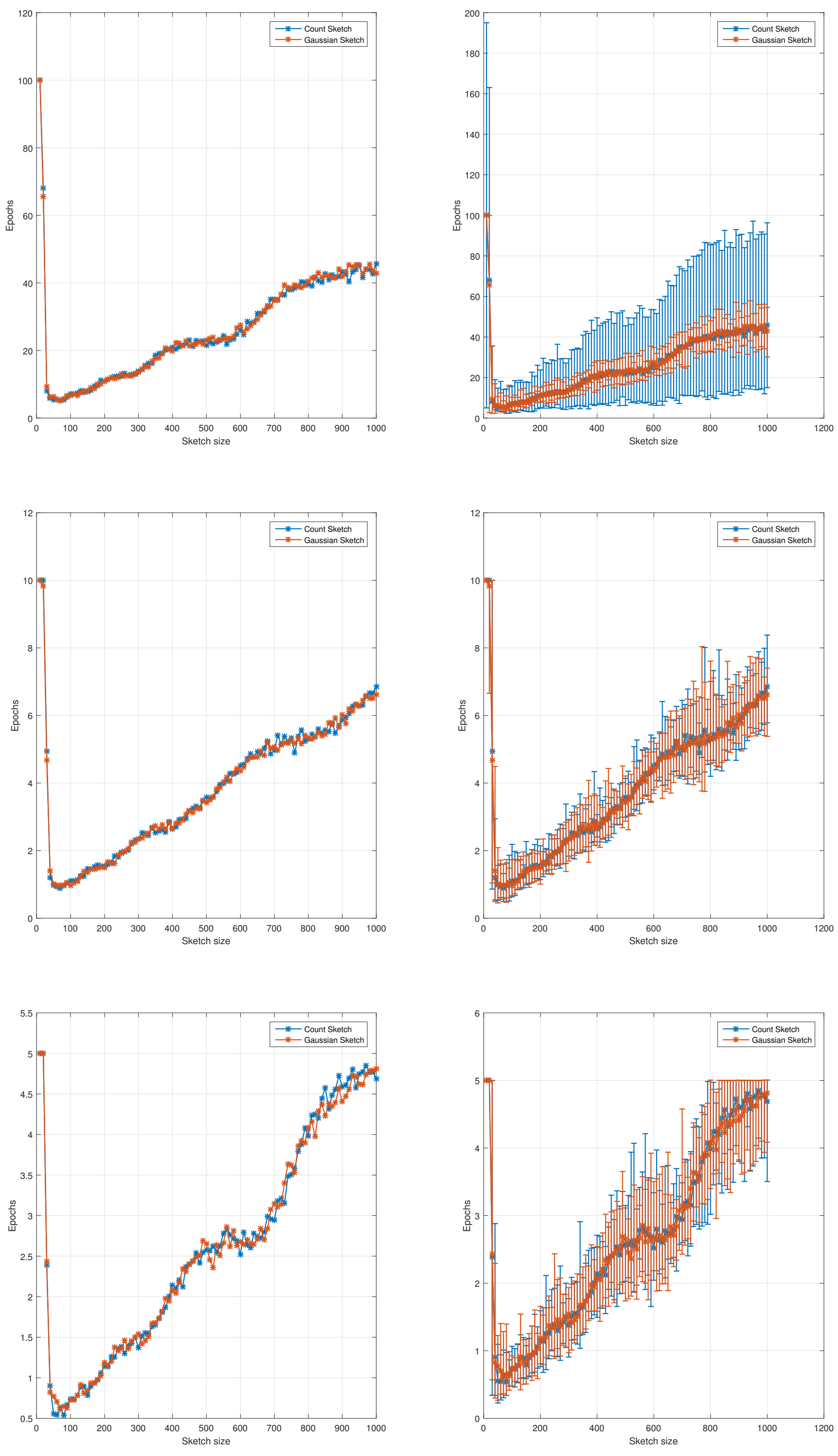


**Figure 3.7:** *Computational and sketch size trade-off experiments for GPCS on $\ell_1$ constrained linear system $y = Ax$. From top to bottom : Syn1, Syn2 and Syn3; from left to right: average curve (on 20 trials) and error bar (the maximum and minimum epoch counts in these 20 trials)*

# Chapter 4
# Structure-Adaptive Accelerated Stochastic Gradient Descent

## 4.1 Towards the Structure-Adaptive, Variance-Reduced and Accelerated Stochastic Optimization

While the previous chapter proposes sketched gradient algorithms for constrained optimization and studies in great detail of the structure-adaptive property of this class of algorithms, there is very little research on the structure-adaptive algorithmic design and analysis of the stochastic gradient methods. We consider the generic empirical risk minimization (2.3) in this chapter:

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} \left\{ F(x) := f(x) + \lambda g(x) \right\}, \quad f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

which we have introduced in Section 2.1. In the "big data" and "big dimension" regimes, stochastic gradient-based iterative algorithms are very often considered. The most basic one is often referred to as *stochastic gradient descent* (SGD) [46, 47], in every iteration of which only one or a few functions $f_i$ are randomly selected, and only their gradients are calculated as an estimation of the full gradient. However, the convergence rate of SGD is sub-linear even when the loss function $F$ is strongly-convex.

To further accelerate the stochastic gradient descent algorithm, researchers have recently developed techniques which progressively reduce the variance of stochastic gradient estimators, starting from SAG [56, 57], SDCA [61], then SVRG [58, 60] and SAGA [62]. Such methods enjoy a linear convergence rate when the cost function $F$ described in (2.3) is $\mu$-strongly-convex and each $f_i$ has $L$-Lipschitz continuous gradients, that is, to achieve an output $\hat{x}$ which satisfies $F(\hat{x}) - F(x^\star) \leq \delta$, the total number of stochastic gradient evaluations needed is $O\big(n + L/\mu\big)\log\frac{1}{\delta}$. *Nesterov's acceleration* [37, 39, 31] has also been successfully applied to construct variance-reduced methods which have an accelerated linear-convergence rate [6, 74, 63, 55, 65, 4, 67, 140]:

$$O\left(n + \sqrt{\frac{nL}{\mu}}\right) \log \frac{1}{\delta}. \tag{4.1}$$

It is worth noting that this convergence rate has been shown to be worst-case optimal [55]. However, all of these algorithms need explicit knowledge of the strong-convexity parameter $\mu$ to achieve the optimal convergence rate [141]. Since in general the strong-convexity is hard to be estimated accurately, researchers proposed adaptive restart schemes [42, 142, 43, 143, 140, 144] for accelerated first-order methods, either by the means of enforcing monotonicity on functional decay, or by estimating the strong-convexity on the fly.

In this chapter we extend the theoretical framework of restricted strong-convexity [5] to design and analyse a structure-adaptive variant of Katyusha [4]. Our proposed method Rest-Katyusha is a restarted version of the original Katyusha method for non-strongly convex functions, where the restart period is determined by the restricted strong-convexity (RSC). The convergence analysis for Rest-Katyusha algorithm is provided, wherein we prove linear convergence up to a statistical accuracy with an accelerated convergence rate characterized by the RSC property.

Like all other accelerated gradient methods which require the explicit knowledge of strong-convexity parameter to achieve accelerated linear convergence, the vanilla Rest-Katyusha algorithm also needs to explicitly know the RSC parameter. We therefore propose a practical heuristic (adaptive Rest-Katyusha) which estimates the RSC parameter on the fly and adaptively tune the restart period, and we show that this adaptive scheme mimics the convergence behavior of the vanilla Rest-Katyusha. Finally we validate the effectiveness of our approach via numerical experiments.

**Algorithm 4** Katyusha $(x^0, m, S, L)$

```
**Initialize:** $y_0 = z_0 = \hat{x}^0$;
**for** $s = 0, \dots, S-1$ **do**
    $\theta \leftarrow \frac{2}{s+4}$, calculate $\nabla f(\hat{x}^s)$, $z_0 \leftarrow z^s$, $y_0 \leftarrow y^s$;
    for $k = 0, 1, 2, ..., m$
        $x_{k+1} = \theta z_k + \frac{1}{2}\hat{x}^s + (\frac{1}{2} - \theta) y_k$;    $\rightarrow$ linear coupling
        $\nabla_{k+1} = \nabla f(\hat{x}^s) + \nabla f_i(x_{k+1}) - \nabla f_i(\hat{x}^s)$;
                                        $\rightarrow$ variance reduced stochastic gradient
        $z_{k+1} = \arg\min_z \frac{3\theta L}{2}\|z - z_k\|_2^2 + \langle \nabla_{k+1}, z\rangle + \lambda g(z)$; $\rightarrow$ proximal mirror descent
        $y_{k+1} = \arg\min_y \frac{3L}{2}\|y - x_{k+1}\|_2^2 + \langle \nabla_{k+1}, y\rangle + \lambda g(y)$;
                                        $\rightarrow$ proximal gradient descent
    $\hat{x}^{s+1} = \frac{1}{m}\sum_{j=1}^{m} y_j, y^{s+1} = y_m, z^{s+1} = z_m$;
**end for**
**Output:** $\hat{x}^S$
```

## 4.2 Restarted Katyusha Algorithm

The Katyusha algorithm [4] listed in Algorithm 4 is an accelerated stochastic variance-reduced gradient method extended from the linear-coupling framework for constructing accelerated methods [145]. This algorithm is an accelerated variant of the proximal SVRG algorithm we introduced in Section 2.4. It is one of the state-of-the-art methods for empirical risk minimization and matches the complexity lower-bound for minimizing smooth-convex finite-sum functions, proven by [55]. Most notably, it is a primal method which directly[1] accelerates stochastic variance-reduction methods.

Recall that in Section 2.3, we have introduced Nesterov's acceleration technique for gradient descent. We need to note that, naively applying Nesterov's acceleration on stochastic gradient cannot yield optimal speed up of convergence, since the momentum step is sensitive to the noise of stochastic gradient estimator [147]. To achieve acceleration in the sense of Nesterov, Katyusha introduces the three-point coupling strategy which includes a combination of Nesterov's momentum and a stabilizing negative momentum which cancels the effect of noisy updates due to stochastic gradients. However, its accelerated linear convergence is only established when the regularization term $g(x)$ is strongly-convex, and fails to benefit from the strong convexity from the data-fidelity term [144], or the intrinsic restricted strong-convexity [12].

[1]On the other hand, one can indirectly accelerate SVRG/SAGA via Catalyst [146].

**Algorithm 5** Rest-Katyusha $(x^0, \mu_c, S_0, \beta, T, L)$

```
Initialize: $m = 2n$, $S = \left\lceil \beta\sqrt{32 + \frac{24L}{m\mu_c}} \right\rceil$;
 First stage —– warm start:
 $x^1$ = Katyusha $(x^0, m, S_0, L)$
 Second stage —- exploit the restricted strong-convexity via periodic restart:
for $t = 1, ..., T$ do
    $x^{t+1}$ = Katyusha $(x^t, m, S, L)$
end for
Output: $x^{T+1}$
```

**Restart to rescue:** as we have mentioned in section 2.3.5, it is well-known that if the cost function $F(x)$ is $\mu$-strongly convex, one can periodically restart the accelerated full gradient method [39], and improve it from a sublinear convergence rate $F(x^k) - F^\star \leq \frac{4L\|x^0 - x^\star\|_2^2}{k^2}$ to a linearly convergent algorithm. For instance if we set $k = \left\lceil 4\sqrt{L/\mu} \right\rceil$, then one can show that the suboptimality can be reduced by $\frac{1}{4}$:

$$F(x^k) - F^\star \leq \frac{4L\|x^0 - x^\star\|_2^2}{k^2} \leq \frac{4L[F(x^0) - F^\star]}{\mu k^2} \leq \frac{1}{4}[F(x^0) - F^\star]. \tag{4.2}$$

Then we can recursively apply this statement (algorithmically speaking, we restart the algorithm every $\left\lceil 4\sqrt{L/\mu} \right\rceil$ iterations), and only $k \geq \left\lceil 4\sqrt{\frac{L}{\mu}} \right\rceil \log_4 \frac{1}{\delta}$ iterations are needed to make $F(x^k) - F^\star \leq \delta$. Hence an accelerated linear rate is achieved. The restart scheme has been recently applied to improve the convergence of the accelerated coordinate descent method [40, 143] and accelerated variance-reduced dual-averaging method [140] for strongly-convex functions.

Inspired by [39] we first propose the Katyusha method with periodic restarts, and meanwhile demonstrate that when the restart period is appropriately chosen, the proposed method is able to exploit the restricted strong-convexity property to achieve an accelerated linear convergence, even when the cost function itself is not strongly-convex. We propose to warm start the algorithm prior to the periodic restart stage, by running the Katyusha algorithm for a number of epochs, which in theory should be proportional to the suboptimality of the starting point $x^0$. We present our Rest-Katyusha method as Algorithm 5.

## 4.3 Convergence Analysis of Rest-Katyusha

### 4.3.1 Generic Assumptions

We start by formally listing out the assumptions we shall engage with in our analysis on Rest-Katyusha's convergence for solving the regularized ERM described in (2.3), Section 2.1. These generic assumptions are necessary for applying the RSC framework [5]. We have briefly introduced these important concepts in Section 2.7.1 of the background chapter, and we recall them here for coherency of this chapter. Meanwhile, we will see some slight changes we make on the assumptions compared to what we have introduced previously. These changes are made due to the fact that we choose not to consider the additional side-constraint on the composite optimization task as the original work of Agarwal et.al. [5] did.

**A. 1.** *(Decomposable regularizer) [5] Given a orthogonal subspace pair $(\mathcal{M}, \mathcal{M}^{\perp})$ in $\mathbb{R}^d$, $g(.)$ is decomposable which means:*

$$g(a + b) = g(a) + g(b), \forall a \in \mathcal{M}, b \in \mathcal{M}^{\perp}. \tag{4.3}$$

In this chapter we focus on cases where the regularizer is decomposable, which includes many popular regularizers which can enforce low-dimensional structure, such as $\ell_1$ norm, $\ell_{2,1}$ norm and nuclear norm penalty[2]. The subspace $\mathcal{M}$ is named the *model subspace*, while its orthogonal complement $\mathcal{M}^{\perp}$ is called the *perturbation subspace*. A similar notion of decomposition would extend the scope of this work to more general gauge functions $g(.)$, such as the so-called *analysis priors*, e.g., total variation regularization (for more details see [115]).

**A. 2.** *(Restricted strong convexity) [5] The function $f(.)$ satisfies restricted strong convexity with respect to $g(.)$ with parameters ($\gamma$, $\tau$) if the following inequality holds true:*

$$f(x) - f(x^{\star}) - \langle \nabla f(x^{\star}), x - x^{\star} \rangle \geq \frac{\gamma}{2}\|x - x^{\star}\|_2^2 - \tau g^2(x - x^{\star}), \quad \forall x \in \mathbb{R}^d. \tag{4.4}$$

Note that compared to Definition 2.7.3, this RSC assumption does not have the additional side-constraint. In [5], $\gamma$ is referred as the *lower curvature parameter*, while $\tau$ is named the *tolerance parameter*. It is clear that if $\tau = 0$, **A.2** reduces to usual strong-convexity assumption. While

[2]The nuclear-norm penalty satisfies a slightly more complicated form of decomposibility on subspace pair $(\mathcal{M}, \bar{\mathcal{M}}^{\perp})$ where $\mathcal{M} \in \bar{\mathcal{M}}$.

in the high-dimensional setting, the strong-convexity often does not hold, but it has been shown in literature that such a milder assumption of RSC does hold in many situations. This notion of RSC is distinctively different from other forms of weak strong-convexity assumption based on the Polyak-Lojasiewicz inequality [148] for the purpose of this work, because it encodes the direction-restricting effect of the regularization, and hence has been shown to have a direct connection with the low-dimensional structure of $x^\star$.

**A. 3.** *Each $f_i(.)$ has L-Lipschitz continuous gradient:*

$$\|\nabla f_i(x) - \nabla f_i(x')\|_2 \leq L\|x - x'\|_2, \forall x, x' \in \mathbb{R}^d. \tag{4.5}$$

As we have previously introduced in section 2.4, this form of smoothness assumption is classic for variance-reduced stochastic gradient methods.

**A. 4.** *Recall the definition of $x^\dagger$ by (2.2) in Section 2.1. The regularization parameter $\lambda$ and $x^\dagger$ satisfies:*

$$\lambda \geq (1 + \frac{1}{c})g^*(\nabla f(x^\dagger)), \tag{4.6}$$

*with constant $c \geq 1$.*

Assumption **A.4** with the choice of $c = 1$ is the fundamental assumption of the analytical framework developed by Agarwal et al. [5] and Negahban et al. [116]. We relax the requirement to $c \geq 1$ for more general results. It is seemly a sophisticated and demanding assumption but indeed is reasonable and suits well the purpose of this work, which is to develop fast algorithms to speedily solve structured problems – which is always the result of sufficient regularization. We can also see more insights of this condition from another perspective: recall that the goal of finding the solution $x^\star$ via optimizing the regularized empirical risk is to get a meaningful approximation of the true parameter $x^\dagger$ which is the unique minimizer of the population risk. Especially in the high dimensional setting where $d > n$, the choice of regularization is rather important since there is no good control over the statistical error $\|x^\dagger - x^\star\|$ for an arbitrarily chosen $\lambda$. Because of this issue, in this work we only focus on the "meaningful" regularized ERM problems which are able to provide trustworthy approximation. Similar to **A.4**, [116] has shown that $\lambda \geq 2g^*(\nabla f(x^\dagger))$ provides a sufficient condition to bound the statistical error $\|x^\star - x^\dagger\|_2^2$, as presented in the background chapter.

### 4.3.2 Main Results

Based on the assumption of the restricted strong convexity on $f(.)$ w.r.t $g(.)$, and also with the definition of subspace compatibility (see Definition 2.7.2):

$$\Phi(\mathcal{M}) := \sup_{v \in \mathcal{M} \backslash \{0\}} \frac{g(v)}{\|v\|_2},$$

one can further derive a more expressive form of RSC, which is named *Effective RSC* [5] which has a direct link to the structure of solution.

**Lemma 4.3.1.** *(Effective RSC)* [3] *Under **A.1, A.2** , **A.4**, while $x$ satisfies $F(x) - F(x^\star) \leq \eta$ for a given value $\eta > 0$ and any minimizer $x^\star$, with $\varepsilon := 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_2 + 4g(x^\dagger_{\mathcal{M}^\perp})$ we have:*

$$F(x) - F^\star \geq \mu_c\|x - x^\star\|_2^2 - 2\tau(1+c)^2 v^2, \tag{4.7}$$

*where $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})$ and $v = \frac{\eta}{\lambda} + \varepsilon$.*

Here we refer $\mu_c$ as the effective restricted strong convexity parameter, which will provide us with a direct link between the convergence speed of an algorithm and the low-dimensional structure of the solution. Note that this lemma relaxes the condition on $\lambda$ in [5, Lemma 11], which is restricted to $c = 1$. Our main theorem is presented as the following:

**Theorem 4.3.2.** *Under **A.1 - 4**, if further **A.2** holds with parameter $(\gamma, \tau)$ such that $\tau\Phi^2(\mathcal{M}) < \frac{\gamma}{16(1+c)^2}$, denote $\varepsilon := 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_2 + 4g(x^\dagger_{\mathcal{M}^\perp})$, $\mathcal{D}(x^0, x^\star) := 16(F(x^0) - F^\star) + \frac{6L}{n}\|x^0 - x^\star\|_2^2$, $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})$, if we run Rest-Katyusha with*

$$S_0 \geq \left\lceil \left(1 + \frac{2}{\rho\lambda}\right)\sqrt{\frac{6L\tau(1+c)^2\mathcal{D}(x^0, x^\star)}{8n\mu_c + 3L}} \right\rceil, \; S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil \text{ with } \beta \geq 2 \tag{4.8}$$

*then the following inequality holds:*

$$\mathbb{E}[F(x^{T+1}) - F^\star] \leq \max\left\{\varepsilon, \left(\frac{1}{\beta^2}\right)^T \frac{\mathcal{D}(x^0, x^\star)}{(S_0+3)^2}\right\} \tag{4.9}$$

*with probability at least $1 - \rho$.*

[3] *This Lemma is first proven in our earlier preprint [12, Lemma 3.3], and is an extension of [5, Lemma 11]. In Appendix 5-A, we provide the proof of a more general version of this Lemma, which includes it as a special case.*

**Corollary 4.3.3.** *Under the same assumptions, parameter choices and notations as Theorem 4.3.2, the total number of stochastic gradient evaluations required by Rest-Katyusha to get an $\delta > \varepsilon$ accuracy is:*

$$O\left(n + \sqrt{\frac{nL}{\mu_c}}\right) \log\frac{1}{\delta} + O(n)S_0. \tag{4.10}$$

**Proof technique.** We extend the proof technique of Agarwal et al. [5] to the proximal gradient descent and also the proof technique of Qu and Xu [118] for SVRG which are both based on applying induction statements to roll up the residual term of (4.7) which is the second term at the RHS. The complete proofs of Theorem 4.3.2 and Corollary 4.3.3 can be found in the appendix of this chapter.

**Accelerated linear convergence.** Under the RSC assumption, Theorem 4.3.2 and Corollary 4.3.3 demonstrate a local accelerated linear convergence rate of Rest-Katyusha up to a statistical accuracy $\delta > \varepsilon$. To the best of our knowledge, this is the first structure-adaptive convergence result for an accelerated incremental gradient algorithm. Note that this result can be trivially extended to a global accelerated linear convergence result (with $S_0 = S$) with the same setting of Agarwal et al. [5] where a side constraint $g(x) \leq R$ for some radii $R$ is added to restrict the early iterations with additional re-projections unto this constraint set[4]. Starting from the objective-gap convergence result (4.9) and the effective RSC (4.7), with some additional algebra one can easily derive the accelerated linear convergence on the optimization variable of the form:

$$\mathbb{E}\|x^{T+1} - x^\star\|_2^2 \leq O\left(\frac{1}{\beta^2}\right)^T \mathcal{D}_1 + o(\varepsilon^2), \tag{4.11}$$

where we denote $\mathcal{D}_1 := \frac{2\mathcal{D}(x^0, x^\star)}{\mu_c(S_0+3)^2}$.

**Structure-adaptive convergence.** The effective RSC $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})$ links the convergence speed of Rest-Katyusha with the intrinsic low-dimensional structure of the solution which is due to the regularization. For instance, if $F(x) := \frac{1}{2n}\|Ax - b\|_2^2 + \lambda\|x\|_1$, $\|x^\star\|_0 = s$ and (**A.4**) holds with $c = 1$, then we have $\mu_c = \frac{\gamma}{2} - 32\tau s$, meanwhile for a wide class of random design matrices we have $\tau = O(\frac{\log d}{n})$ and $\gamma > 0$. More specifically, if the rows of the random design matrix $A$ are drawn i.i.d. from $\mathcal{N}(0, \Sigma)$ with covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ which has largest singular value $r_{\max}(\Sigma)$ and smallest singular value $r_{\min}(\Sigma)$, then

[4]In [5], a side constraint is manually added to the regularized ERM problem, hence in their setting, the effective restricted strong-convexity is valid globally. They provide global linear convergence result of proximal gradient descent (with additional re-projection steps) at a cost of additional side-constraints.

$\gamma \geq \frac{r_{\min}(\Sigma)}{16}$ and $\tau \leq r_{\max}(\Sigma)\frac{81 \log d}{n}$ with high probability as shown by Raskutti et al. [117].

**High probability statement.** Since our proofs utilize the effective RSC which holds in a neighborhood of $x^\star$ as demonstrated in Lemma 4.3.1, we need to bound the functional suboptimality $F(x^t) - F^\star$ in the worst case instead of in expectation. Hence inevitably the Markov inequality has to be applied to provide the convergence statement with high probability (details can be found in the main proof).

**Optimizing the choice of $\beta$.** Theorem 4.3.2 shows that the complexity of the main loop of Rest-Katyusha is $\left\lceil \beta\sqrt{32 + 12L/(n\mu_c)} \right\rceil \log_{\beta^2} \frac{1}{\delta}$, which suggests a trade-off between the choice of $\beta$ and the total computation. With some trival computation one can derive that in theory the best choice of $\beta$ is exactly the Euler's number ($\approx 2.7$). Numerically, we observe that slightly larger choice of $\beta$ often provides better performance in practice (illustrative examples can be found in the appendix of this chapter).

## 4.4 Adaptive Rest-Katyusha

Motivated by the theory above, we further propose our practical adaptive restart heuristic of Rest-Katyusha which is able to estimate the effective RSC on the fly. Based on the convergence theory, we observe that, with the choice of restart period $S = \left\lceil \beta\sqrt{32 + 12L/(n\mu_0)} \right\rceil$ with a conservative estimate $\mu_0 \leq \mu_c$, then we are always guaranteed to have:

$$\mathbb{E}_{\xi_t \backslash \xi_{t-1}}[F(x^{t+1}) - F^\star] \leq \frac{1}{\beta^2}[F(x^t) - F^\star], \tag{4.12}$$

due to the fact that an underestimation of the RSC will lead to a longer restart period than we actually need[5]. The intuition behind our adaptive restart heuristic is: if we overestimate $\mu_c$, the above inequality will be violated. Hence an adaptive estimation of $\mu_c$ can be achieved via a convergence speed check. However the above inequality cannot be evaluated directly in practice since it is in expectation and demands the knowledge of $F^\star$. In [43, Prop. 4], it has been shown that with the composite gradient map:

$$\mathcal{T}(x) = \arg\min_q \frac{L}{2}\|x - q\|_2^2 + \langle \nabla f(x), q - x \rangle + \lambda g(q), \tag{4.13}$$

[5]An inaccurate estimate of the RSC will lead to a compromised convergence rate. Detailed discussion and analysis of Rest-Katyusha with a rough RSC estimate can be found in the Appendix.

the value of $F(x) - F^\star$ can be lower bounded:

$$F(x) - F^\star \geq \frac{1}{2}\|\mathcal{T}(x) - x\|_2^2, \tag{4.14}$$

and also it can be approximately upper bounded by $O(\|\mathcal{T}(x) - x\|_2^2)$ if local quadratic growth is assumed, which reads:

$$\exists \alpha > 0,\ r > 0,\ F(x) - F^\star \geq \alpha\|x - x^\star\|_2^2, \forall x\ s.t.\ \|x - x^\star\|_2^2 < r. \tag{4.15}$$

Hence in our adaptive restart heuristic we check the convergence speed via evaluating the composite gradient map at the snapshot points where full gradients have already been calculated. Because of this, the only main additional computational overhead of this adaptive restart scheme is the proximal operation of $g(.)$ at the restart points.

**Algorithm 6** Adaptive Rest-Katyusha $(x^0, \mu_0, S_0, \beta, T, L)$

```
**Initialize:** Epoch length $m = 2n$; Initial restart period $S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_0}} \right\rceil$;
 $x^1$ = Katyusha $(x^0, m, S_0, L)$
 Calculate the composite gradient map:
 $\mathcal{T}(x^1) = \arg\min_x \frac{L}{2}\|x - x^1\|_2^2 + \langle \nabla f(x^1), x - x^1\rangle + \lambda g(x)$.
**for** $t = 1, \dots, T$ **do**
    $x^{t+1}$ = Katyusha $(x^t, m, S, L)$
    ——*Track the convergence speed via the composite gradient maps:*
        $\mathcal{T}(x^{t+1}) = \arg\min_x \frac{L}{2}\|x - x^{t+1}\|_2^2 + \langle \nabla f(x^{t+1}), x - x^{t+1}\rangle + \lambda g(x)$.
    —— *Update the estimate of RSC and adaptively tune the restart period*
        **if** $\|\mathcal{T}(x^{t+1}) - x^{t+1}\|_2^2 \leq \frac{1}{\beta^2}\|\mathcal{T}(x^t) - x^t\|_2^2$
          **then** $\mu_0 \leftarrow 2\mu_0$, **else** $\mu_0 \leftarrow \mu_0/2$. $S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_0}} \right\rceil$
**end for**
```

The adaptive Rest-Katyusha method is presented in Algorithm 6. We highlight the heuristic estimation procedure of RSC parameter, which is additional to the original Katyusha algorithm. The algorithm starts with an initial guess $\mu_0$ and the restart period $S$, meanwhile we calculate the composite gradient map $\mathcal{T}(x^1)$ at $x^1$ and record the value of $\|\mathcal{T}(x^1) - x^1\|_2^2$ which we use as the estimation of $F(x^1) - F^\star$ (and so on). Then after $S$ outer-loops, we restart the algorithm and then evaluate again the composite gradient map. If $\|\mathcal{T}(x^2) - x^2\|_2^2 \geq \frac{1}{\beta^2}\|\mathcal{T}(x^1) - x^1\|_2^2$, then we suspect that the RSC parameter has been overestimated, and hence we reduce $\mu_0$ by a half, otherwise we double the estimation. We also update the restart period by $S = \left\lceil \beta\sqrt{32 + 12L/(n\mu_0))} \right\rceil$ with the modified $\mu_0$. The forthcoming iterations follow the same updating rule as described.

## 4.5 Numerical Experiments

In this section we describe our numerical experiments on our proposed algorithm Rest-Katyusha (Alg.5) and also the adaptive Rest-Katyusha (Alg.6). We focus on the Lasso regression task:

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} \left\{ F(x) := \frac{1}{2n}\|Ax-b\|_2^2 + \lambda\|x\|_1 \right\}. \tag{4.16}$$

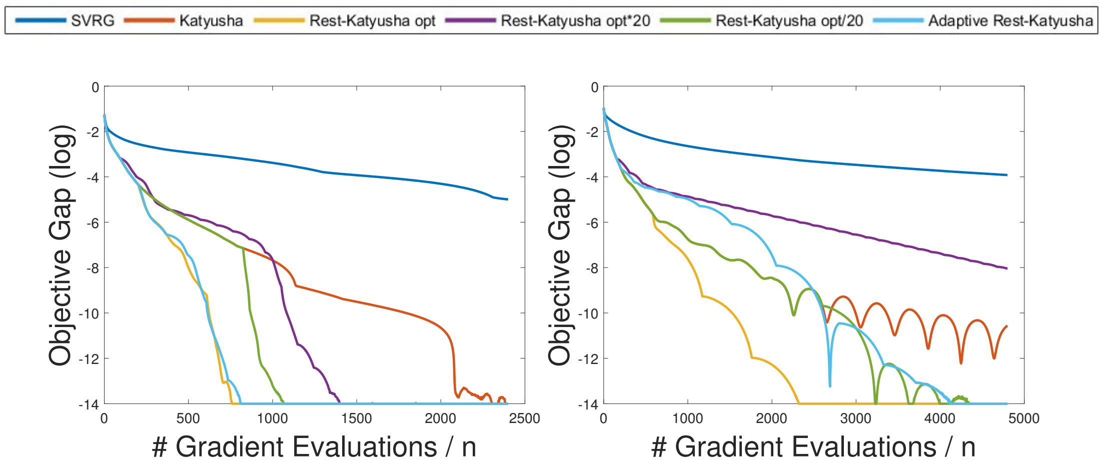
To enforce sparsity on regression parameter we use the $\ell_1$ penalty with various degrees of regularization parameters chosen from the set $\lambda \in \{1\times 10^p, 2\times 10^p, 5\times 10^p, p\in\mathbb{Z}\}$. For comparison, the performance of (proximal) SVRG and the original Katyusha method is also shown in the plots. We run all the algorithms with their theoretical step sizes on Madelon and REGED dataset, while on RCV1 dataset we use minibatch scheme for all the algorithms and grid-search the step sizes which can optimize these algorithms' performance.

| Data set | Size ($n$, $d$) | Minibatch | Ref. |
|---|---|---|---|
| **(A)** Madelon | (2000, 500) | 1 | [137] |
| **(B)** RCV1 | (20242, 47236) | 80 | [137] |
| **(C)** Reged | (500, 999) | 1 | [149] |

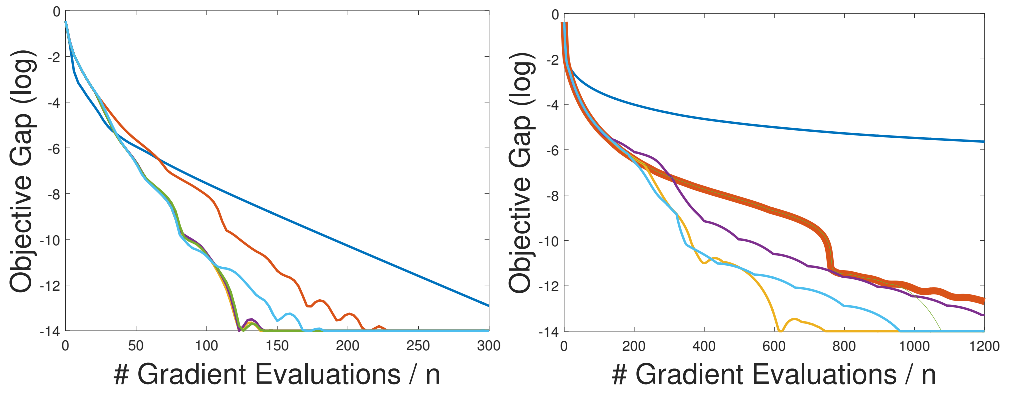
**Table 4.1:** *Datasets for the Experiments and Minibatch Size Choice for the Algorithms*

In all our experiments we set $\beta = 5$ and $S_0 = S$ for convenience. We first do a grid-search on the estimate of $\mu_c$ for Rest-Katyusha which provides the best convergence performance, and denote it as "Rest-Katyusha opt" in the plots. Meanwhile we also run Rest-Katyusha with a RSC estimation which is 20 times larger or smaller than the optimal one, which we denote as "Rest-Katyusha opt*20" and "Rest-Katyusha opt/20" respectively. At the 5th plot in Figure 1 the curves for Rest-Kat opt, opt*20 and opt/20 are indistinguishable which shows that in these particular experiments their performance can be almost identical. For the adaptive Rest-Katyusha we fix our starting estimate of $\mu_c$ as $10^{-5}$ throughout all the experiments.

From these experiments we observe that as our theory has predicted, the Rest-Katyusha achieves accelerated linear convergence even when there is no explicit strong-convexity in the cost function (RCV1 and REGED dataset), and the convergence speed has a direct relationship with the sparsity of solution. For the lasso experiments when the solution is sparser, the linear convergence speed of Rest-Katyusha indeed becomes faster. Meanwhile when we run Rest-Katyusha



**(A)** $\lambda = 5 \times 10^{-5}$, $\|x^{\star}\|_0 = 42$ **(A)** $\lambda = 2 \times 10^{-5}$, $\|x^{\star}\|_0 = 159$



**(B)** $\lambda = 1 \times 10^{-4}$, $\|x^{\star}\|_0 = 902$ **(B)** $\lambda = 1 \times 10^{-5}$, $\|x^{\star}\|_0 = 6315$

**Figure 4.1:** *Lasso experiment results on **(A)** Madelon and **(B)** RCV1, comparing the convergence performance of the proposed algorithms (Rest-Katyusha and Adaptive Rest-Katyusha) with the baseline algorithms SVRG and Katyusha.*

with an inaccurate RSC estimate, we still observe a compromised linear convergence, as predicted by our theory. In all the experiments, we have observed that the adaptive Rest-Katyusha indeed achieves a good estimation of the RSC parameter and properly adapts the choice of restart period automatically on the fly, hence its performance is often comparable with the best tuned Rest-Katyusha. Similar to the experimental results shown in [5, 118, 12], the linear convergence we observe is towards an arbitrary accuracy instead of a threshold nearby the solution. This conservative aspect of the theory is inherently due to the artifact of the RSC framework [5] and we include the extension for arbitrary accuracy regime as our future work. We also include additional experimental results in the appendix of this chapter.

## 4.6 Concluding Remarks

We developed a restart variant of the Katyusha algorithm for regularized empirical risk minimization tasks, which is provably able to actively exploit the intrinsic low-dimensional structure of the solution for the acceleration of convergence. Based on the convergence result we further constructed an adaptive restart heuristic which aimed at estimating the RSC parameter on the fly and adaptively tuning the restart period. The efficiency of this approach is validated through numerical experiments. In future work, we aim to develop more refined and provably-good adaptive restart schemes for Rest-Katyusha algorithm to further exploit the solution's structure for acceleration.

## 4.A Appendix

In this appendix we include the proof of our main result which establishes the structure-adaptive and accelerated linear convergence rate for Rest-Katyusha algorithm. Further, we also extend our analysis to the case where we underestimate the RSC parameter. We also provide an additional numerical result for testing the convergence rate of Rest-Katyusha with different choices of the input parameter $\beta$.

### 4.A.1 The Proof of Theorem 4.3.2 and Corollary 4.3.3

We first state the convergence result of Katyusha algorithm for non-strongly convex functions:

**Lemma 4.A.1.** *[4, Theorem 4.1] Under* **A.3***, starting at $x^0$, with epoch length $m = 2n$, denote $\mathcal{D}(x^0, x^\star) := 16(F(x^0) - F^\star) + \frac{6L}{n}\|x^0 - x^\star\|_2^2$, the $s$-th snapshot point $\hat{x}^s$ of Katyusha algorithm satisfies:*

$$\mathbb{E}[F(\hat{x}^s)] - F^\star \leq \frac{\mathcal{D}(x^0, x^\star)}{(s+3)^2}. \tag{4.17}$$

Now based on the effective RSC by Lemma 4.3.1 in the main text we are able to provide the proof of our main result.

*Proof.* At each iteration, the algorithm chooses an index $i$ uniformly at random to perform the calculation of one stochastic variance-reduced gradient. The update sequences $y_{k+1}$ and $z_{k+1}$ within $t$-th outer-loop of Rest-Katyusha depend on the realization of the following random variable which we denote as $\xi_k^t$:

$$\xi_t = \{i_m^t, i_{m-1}^t, ..., i_1^t, i_0^t, i_m^{t-1}, ..., i_0^{t-1}, ..., i_m^0, ..., i_0^0\}, \tag{4.18}$$

and for the randomness within a single outer-loop of Rest-Katyusha we specifically denote $\xi_t \backslash \xi_{t-1}$ as

$$\xi_t \backslash \xi_{t-1} = \{i_m^t, i_{m-1}^t, ..., i_1^t, i_0^t\}. \tag{4.19}$$

According to Lemma 4.A.1, setting $m = 2n$, for the first stage $t = 0$:

$$\mathbb{E}_{\xi_0}[F(x^1)] - F^\star \leq \epsilon_1 := \frac{4}{n(S_0+3)^2}\left[4n\left(F(x^0) - F^\star\right) + \frac{3L}{2}\|x^0 - x^\star\|_2^2\right].$$

Then, applying Markov's inequality, with probability at least $1 - \frac{\rho}{2}$ we have:

$$F(x^1) - F^\star \leq \frac{2}{\rho}\epsilon_1. \tag{4.20}$$

Now we define three sequences $\epsilon_t$, $\rho_t$ and $v_t$: $\epsilon_{t+1} = \frac{1}{\beta^2}\epsilon_t > \varepsilon$, $\rho_{t+1} = \frac{1}{\beta}\rho_t$ (with $\rho_1 := \rho$), $v_t = \frac{2\epsilon_t}{\lambda\rho_t} + \varepsilon$. Next we use and induction argument to upper bound $\mathbb{E}_{\xi_{t-1}}F(x^t) - F^\star$.

**Induction step 1:** We turn to the first iteration of the second stage, note that due to the effective RSC, we can write:

$$\|x - x^\star\|_2^2 \leq \frac{1}{\mu_c}\left[F(x) - F^\star + 2\tau(1+c)^2 v^2\right]. \tag{4.21}$$

Hence we can have the following:

$$\begin{aligned}
\mathbb{E}_{\xi_1 \backslash \xi_0}[F(x^2) - F^\star] &\leq \frac{16}{(S+3)^2}[F(x^1) - F^\star] + \frac{6L}{n\mu_c(S+3)^2}\left[F(x^1) - F^\star + 2\tau(1+c)^2 v_1^2\right] \\
&\leq \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}[F(x^1) - F^\star] + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}v_1^2.
\end{aligned}$$

Next we take the expectation over $\xi_0$ and we have:

$$\begin{aligned}\mathbb{E}_{\xi_1}[F(x^2) - F^\star] &\leq \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}\mathbb{E}_{\xi_0}[F(x^1) - F^\star] + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}v_1^2 \\ &= \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}\epsilon_1 + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}\left(\frac{2\epsilon_1}{\rho\lambda} + \varepsilon\right)^2 \\ &\leq \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}\epsilon_1 + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}\left(\frac{2\epsilon_1}{\rho\lambda} + \epsilon_1\right)^2.\end{aligned}$$

Now we set:

$$\frac{12L\tau(1+c)^2}{n\mu_c}\left[\left(\frac{2}{\rho\lambda} + 1\right)\epsilon_1\right]^2 \leq \left(16 + \frac{6L}{n\mu_c}\right)\epsilon_1, \tag{4.22}$$

equivalently:

$$\left(\frac{2}{\rho\lambda} + 1\right)^2 \epsilon_1 \leq \frac{8n\mu_c + 3L}{6L\tau(1+c)^2}. \tag{4.23}$$

Denoting $\mathcal{D}(x^0, x^\star) := 16[F(x^0) - F^\star] + \frac{6L}{n}\|x^0 - x^\star\|_2^2$, we have:

$$\epsilon_1 := \frac{\mathcal{D}(x^0, x^\star)}{(S_0+3)^2} \leq \frac{8n\mu_c + 3L}{6L\tau(1+c)^2(\frac{2}{\rho\lambda} + 1)^2}. \tag{4.24}$$

Hence in order to satisfy inequality (4.22), it is enough to set:

$$S_0 \geq \left\lceil \left(1 + \frac{2}{\rho\lambda}\right)\sqrt{\frac{6L\tau(1+c)^2\mathcal{D}(x^0, x^\star)}{8n\mu_c + 3L}} \right\rceil. \tag{4.25}$$

By this choice of $S_0$, according to inequality (4.22) we can write:

$$\mathbb{E}_{\xi_1}[F(x^2) - F^\star] \leq \frac{32 + \frac{12L}{n\mu_c}}{(S+3)^2}\epsilon_1. \tag{4.26}$$

To get $\mathbb{E}_{\xi_1}[F(x^2) - F^\star] \leq \frac{1}{\beta^2}\epsilon_1 = \epsilon_2$, it is enough to set:

$$S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil. \tag{4.27}$$

**Induction step 2:** For the $(t+1)$-th iteration, according to the induction hypothesis, we have

$\mathbb{E}_{\xi_{t-1}}F(x^t) - F^\star \leq \frac{\epsilon_{t-1}}{\beta^2} = \epsilon_t$, and hence with probability $1 - \frac{\rho_t}{2}$ we have:

$$\begin{aligned}
\mathbb{E}_{\xi_t}[F(x^{t+1}) - F^\star] &\leq \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}\mathbb{E}_{\xi_{t-1}}[F(x^t) - F^\star] + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}v_t^2 \\
&= \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}\epsilon_t + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}\left(\frac{2\epsilon_t}{\rho_t\lambda} + \varepsilon\right)^2 \\
&\leq \frac{16 + \frac{6L}{n\mu_c}}{(S+3)^2}\epsilon_t + \frac{12L\tau(1+c)^2}{n\mu_c(S+3)^2}\left(\frac{2\epsilon_t}{\rho_t\lambda} + \epsilon_t\right)^2.
\end{aligned}$$

Then we set:

$$\frac{12L\tau(1+c)^2}{n\mu_c}\left[\left(\frac{2}{\rho_t\lambda} + 1\right)\epsilon_t\right]^2 \leq \left(16 + \frac{6L}{n\mu_c}\right)\epsilon_t, \tag{4.28}$$

equivalently:

$$\left(\frac{2}{\rho_t\lambda} + 1\right)^2 \epsilon_t \leq \frac{8n\mu_c + 3L}{6L\tau(1+c)^2}. \tag{4.29}$$

Now because $\rho_t = \frac{1}{\beta}\rho_{t-1}$, $\epsilon_t = \frac{1}{\beta^2}\epsilon_{t-1}$, we have:

$$\left(\frac{2}{\rho_t\lambda} + 1\right)^2 \epsilon_t = \left(\frac{2}{\rho_{t-1}\lambda} + \frac{1}{\beta}\right)^2 \epsilon_{t-1} \leq \left(\frac{2}{\rho_{t-1}\lambda} + 1\right)^2 \epsilon_{t-1} \leq ... \leq \left(\frac{2}{\rho\lambda} + 1\right)^2 \epsilon_1. \tag{4.30}$$

Hence by the same choice of $S_0$ given by (4.25), inequality (4.28) holds and consequently we can have:

$$\mathbb{E}_{\xi_t}[F(x^{t+1}) - F^\star] \leq \frac{32 + \frac{12L}{n\mu_c}}{(S+3)^2}\epsilon_t. \tag{4.31}$$

To get $\mathbb{E}_{\xi_t}[F(x^{t+1}) - F^\star] \leq \frac{1}{\beta^2}\epsilon_t = \epsilon_{t+1}$, it is enough to set:

$$S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil. \tag{4.32}$$

Hence we finish the induction – by the choice of:

$$S_0 \geq \left\lceil \left(1 + \frac{2}{\rho\lambda}\right)\sqrt{\frac{6L\tau(1+c)^2\mathcal{D}(x^0, x^\star)}{8n\mu_c + 3L}} \right\rceil, \;\; S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil, \tag{4.33}$$

then we will have:

$$\mathbb{E}_{\xi_t}[F(x^{t+1}) - F^\star] \leq \frac{\epsilon_t}{\beta^2} \tag{4.34}$$

where $\epsilon_{t+1} = \frac{1}{\beta^2}\epsilon_t$ and $\epsilon_1 = \frac{\mathcal{D}(x^0,x^\star)}{(S_0+3)^2} = \frac{4}{n(S_0+3)^2}\left[4n\left(F(x^0) - F^\star\right) + \frac{3L}{2}\|x^0 - x^\star\|_2^2\right]$, with probability $1 - \sum_{i=1}^t \frac{\rho_i}{2} \geq 1 - \frac{\rho}{2}\frac{\beta}{\beta-1} \geq 1 - \rho$ (since $\beta \geq 2$). Now we have finished the proof of Theorem 4.3.2.

**Proof of Corollary 4.3.3.** Finally we make a summary of this result for the proof of Corollary 3.5. First we write the number of snapshot point calculation we need to achieve $\mathbb{E}_{\xi_{t-1}} F(x^t) - F^\star \leq \delta$ at the second stage:

$$N_s = \left\lceil \beta \sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil \log_{\beta^2} \frac{F(x^1) - F^\star}{\delta}. \tag{4.35}$$

When $\frac{2n\mu_c}{L} \leq \frac{3}{4}$, $N_s = O\left(\sqrt{\frac{L}{2n\mu_c}} \log \frac{F(x^1)-F^\star}{\delta}\right)$; when $\frac{2n\mu_c}{L} \geq \frac{3}{4}$, $N_s = O\left(\log \frac{F(x^1)-F^\star}{\delta}\right)$. Hence it is enough to run $O\left((1 + \sqrt{\frac{L}{2n\mu_c}}) \log \frac{F(x^1)-F^\star}{\delta}\right) \geq O\left(\max(1, \sqrt{\frac{L}{2n\mu_c}}) \log \frac{F(x^1)-F^\star}{\delta}\right)$ epochs. Since we set the epoch length $m = 2n$ and hence the number of stochastic gradient $\nabla f_i(.)$ calculation is of $O(n)$. Therefore with some more straightforward calculation we conclude that the complexity of the Rest-Katyusha algorithm is:

$$N \geq O\left(n + \sqrt{\frac{nL}{\mu_c}}\right) \log \frac{\frac{1}{\rho(S_0+3)^2}\left[16(F(x^0) - F^\star) + \frac{6L}{n}\|x^0 - x^\star\|_2^2\right]}{\delta} + O(n)S_0. \tag{4.36}$$

□

### 4.A.2 Rest-Katyusha with an Underestimation of $\mu_c$

It is generally not guaranteed that any accelerated stochastic variance-reduced gradient method designed for strongly-convex functions can be directly applied in our modified restricted strong-convexity setting, even when the RSC parameter can be exactly known. It is true that for strongly-convex functions with known strong-convexity parameter, that the convergence rates for a restarted version of non-strongly-convex accelerated gradient descent and the strongly-convex accelerated gradient descent are the same, and we believe that this may be the case for Katyusha as well if the objective is strongly-convex. However, it is still an open question for an objective which only satisfies restricted strong-convexity. One may heuristically replace our algorithm's second stage with the strongly-convex version of Katyusha and this seems to have a comparable result empirically for some datasets if the RSC is accurately given (this is necessary for this method). However, the Rest-Katyusha is superior to this alternative – (1) in terms of theory, as it is a provably convergent algorithm, (2) in terms of practice, Rest-Katyusha appears to be much more robust to the inaccurate estimation of RSC. This section we provide

**Algorithm 7** Rest-Katyusha with a rough RSC estimate $(x^0, \mu_0, \beta, S_0, T, L)$

```
**Initialize:** $m = 2n$, $S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_0}} \right\rceil$;
$x^1$ = Katyusha $(x^0, m, S_0, L)$
**for** $t = 1, ..., T$ **do**
    $x^{t+1}$ = Katyusha $(x^t, m, S, L)$
**end for**
```

an analysis for Rest-Katyusha where we underestimate the RSC parameter.

We have already established the convergence result for Rest-Katyusha algorithm when it is restarted at a frequency $S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil$, but in practice the effective RSC parameter $\mu_c$ is usually unknown and difficult to estimate accurately. We need to find some practical approaches to estimate $\mu_c$ and determine whether to restart or not on the fly. To lay down the basics, we now warm up with the analysis for Rest-Katyusha when only an underestimation of $\mu_c$ is given, to see how the convergence rate of the algorithm will change.

We present the rough RSC estimate version of Rest-Katyusha. The only difference is that the restart period has changed from $\left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_c}} \right\rceil$ to $\left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_0}} \right\rceil$, where $\mu_0$ is a rough (under-)estimate of the effective RSC constant $\mu_c$ and $\beta \geq 2$ is a constant which controls the robustness of possible overestimation. With this restart period, we are able to establish accelerated linear convergence result in the regime where $0 < \mu_0 < \frac{\beta^2}{4}\mu_c$. In other words, with this restart period, as long as $\mu_c$ is no more than $\beta^2/4$ times overestimated by $\mu_0$, the Rest-Katyusha is guaranteed to achieve accelerated linear convergence w.r.t. $\mu_0$.

**Theorem 4.A.2.** *Under **A.1 - 4**, denote* $\varepsilon := 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_2 + 4g(x^\dagger_{\mathcal{M}^\perp})$, $\mathcal{D}(x^0, x^\star) := 16(F(x^0) - F^\star) + \frac{6L}{n}\|x^0 - x^\star\|_2^2$, $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})$, *and* $0 < \mu_0 < \frac{\beta^2}{4}\mu_c$, *with* $\beta \geq 2$, *if we run Rest-Katyusha with* $S_0 \geq \left\lceil \left(1 + \frac{2}{\rho\lambda}\right)\sqrt{2\tau(1+c)^2\mathcal{D}(x^0, x^\star)} \right\rceil$, $S = \left\lceil \beta\sqrt{32 + \frac{12L}{n\mu_0}} \right\rceil$, *then the following inequality holds:*

$$\mathbb{E}[F(x^{T+1}) - F^\star] \leq \max\left\{\varepsilon, \left(\frac{\mu_0}{\mu_c\beta^2}\right)^T \frac{\mathcal{D}(x^0, x^\star)}{(S_0 + 3)^2}\right\}, \tag{4.37}$$

*with probability at least* $1 - \rho$.

**Corollary 4.A.3.** *Under the same assumptions, parameter choices and notations as Theorem 4.A.2, the total number of stochastic gradient evaluation required by Rest-Katyusha to get an*

*$\delta$-accuracy is:*

$$O\left(n+\sqrt{\frac{nL}{\mu_0}}\right)\log_{\frac{\beta^2\mu_c}{\mu_0}}\frac{1}{\delta}+O(n)S_0, \tag{4.38}$$

The proofs of Theorem 4.A.2 and Corollary 4.A.3 follow a very similar procedure as the proofs of Theorem 4.3.2 and Corollary 4.3.3, hence we do not include them here but refer the interested readers to [9, Appendix B] for details.

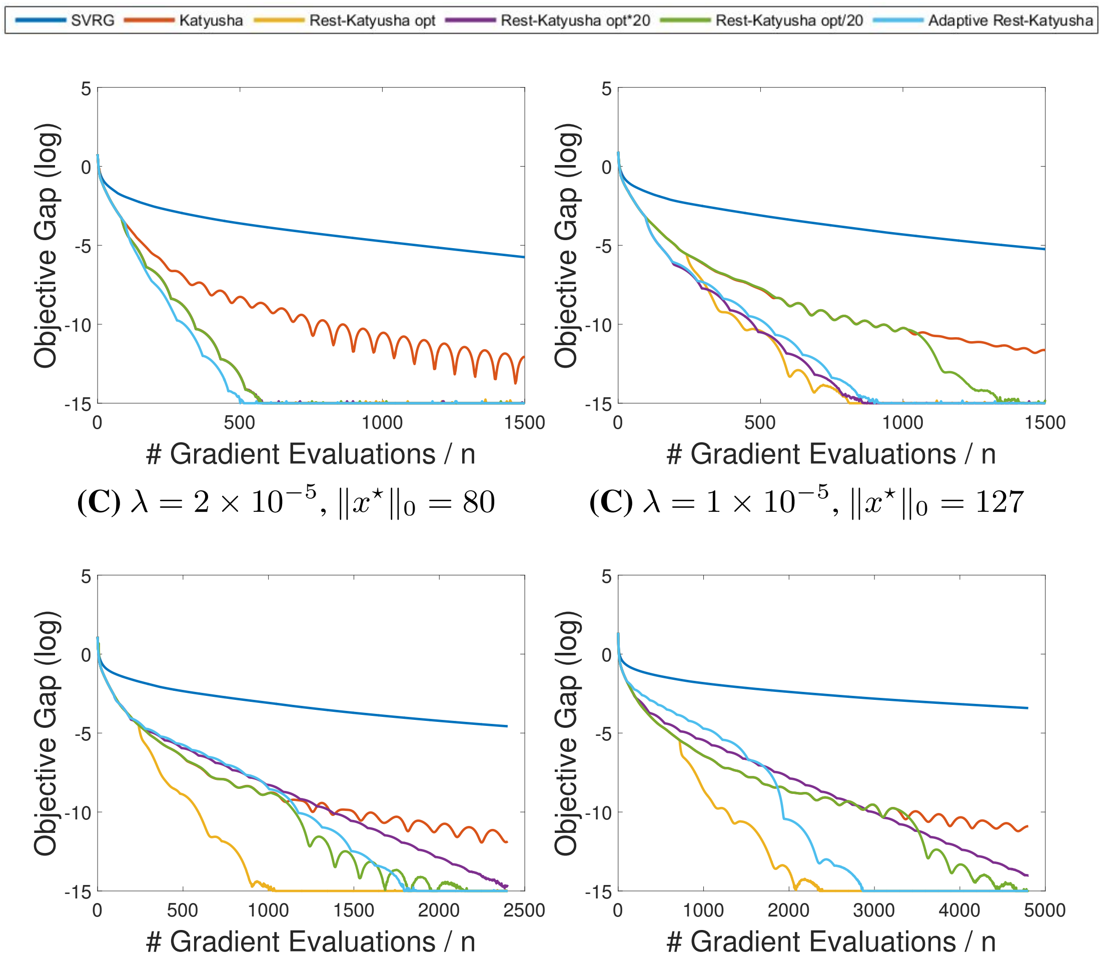
### 4.A.3 Numerical Test for Different Choices of $\beta$

In this section we provide additional experimental results on different choices of $\beta$. We choose to use the REGED dataset in this experiment as an example.

We test the Rest-Katysuha and Adaptive Rest-Katyusha on regularization level $\lambda = 2 \times 10^{-5}$ with 4 different choices of $\beta$ including the theoretically optimal choice which is approximately 2.7. However we found out that the choice of $\beta$ which provides the best practical performance is often slightly larger in experiments for real datasets. For this specific example, we can see that the best choice for $\beta$ is 5 or 10 for both Rest-Katyusha and Adaptive Rest-Katyusha.

### 4.A.4 More Additional Results on High-Dimensional Datasets

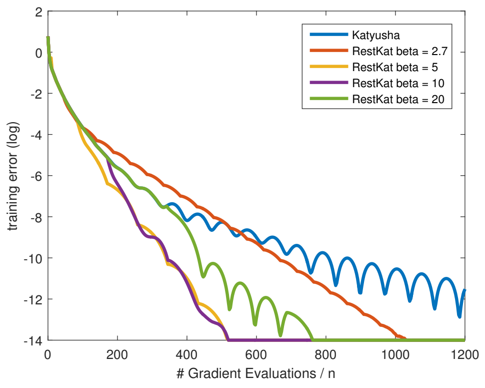
To show the need of high precision optimization for large-scale datasets, we provide here an additional large-scale sparse regression result on the benchmark *News20* dataset (class 1, the version by J. Rennie. [150]) whose size is 15935 by 62061, as well as the *Sector* dataset which is 6412 by 55197, both of these datasets are available online (LIBSVM website). We also plot the $\ell_2$ distance towards a solution $x^\star$ to which after a large number of iterations both of the algorithms will actually converge. We can clearly see that for this specific case, minimizing the objective in a low precision is not enough to ensure that we are close to the solution, i.e. a $10^{-5}$ objective gap accuracy means only $10^{-1}$ accuracy on the optimization variable.



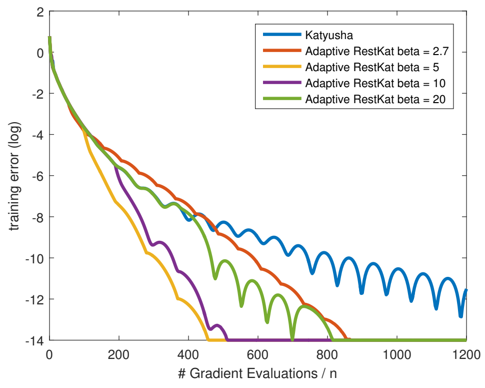
**(C)** $\lambda = 2 \times 10^{-5}$, $\|x^\star\|_0 = 80$ **(C)** $\lambda = 1 \times 10^{-5}$, $\|x^\star\|_0 = 127$

**(C)** $\lambda = 5 \times 10^{-6}$, $\|x^\star\|_0 = 209$ **(C)** $\lambda = 2 \times 10^{-6}$, $\|x^\star\|_0 = 343$

**Figure 4.2:** *Lasso experiment results on* ***(C)*** *REGED dataset, comparing the convergence performance of the proposed algorithms (Rest-Katyusha and Adaptive Rest-Katyusha) with the baseline algorithms SVRG and Katyusha.*



**(C)**$\lambda = 2 \times 10^{-5}$ vanilla Rest-Katyusha



**(C)**$\lambda = 2 \times 10^{-5}$ Adaptive Rest-Katyusha

**Figure 4.3:** *Comparison of different choices of* $\beta$

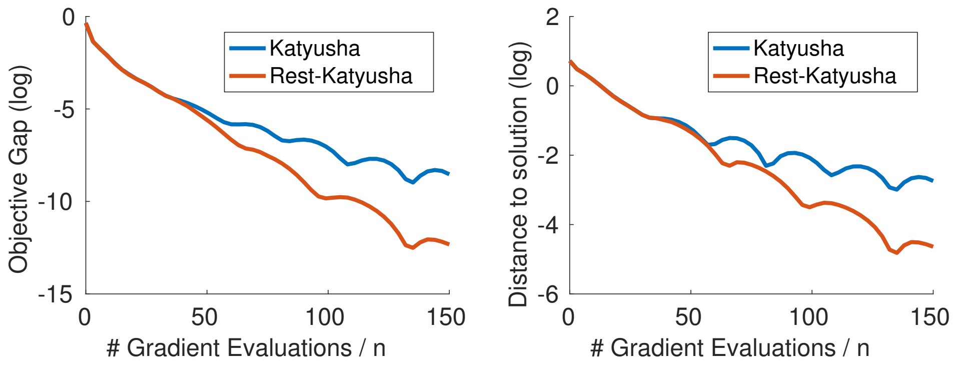


**Figure 4.4:** *Additional Lasso regression experiment results on News20 dataset, with regularization parameter* $\lambda = 1 \times 10^{-4}$

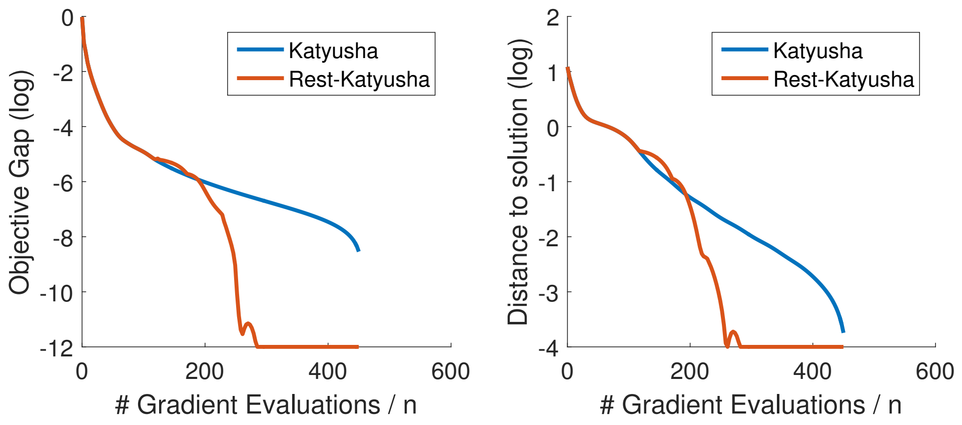


**Figure 4.5:** *Additional Lasso regression experiment results on Sector dataset, with regularization parameter* $\lambda = 5 \times 10^{-4}$

# Chapter 5

# Structure-Adaptive Accelerated Proximal Block Coordinate Descent

## 5.1 Towards the Structure-Adaptive Fast Coordinate Descent

In the previous chapter we have studied the structure-adaptive algorithmic design of the stochastic variance-reduced gradient method for regularized empirical risk minimization in the form of finite-sums. In this chapter we instead consider a different stochastic approach for solving the block-coordinate-wise separable convex composite minimization task which reads:

$$x^\star \in \arg\min_{x \in \mathbb{R}^m} \left\{ F(x) := f(x) + \lambda g(x) \right\}, \quad g(x) = \sum_{i=1}^{d} g_i(x_{(i)}) \tag{5.1}$$

where $x$ consists of $d$-blocks of subvectors: $[x_{(1)}, ..., x_{(d)}]$ and the regularization term $g(x)$ is potentially non-smooth but separable such that $g(x) = \sum_{i=1}^{d} g_i(x_{(i)})$, and $f(x)$ is differentiable with Lipschitz-continuous gradients. When the minimization task is large-scale and high-dimensional, the traditional deterministic gradient methods typically fail to achieve scalablilty. To address this, randomized coordinate descent (RCD) [69, 70, 151] has been intensely studied and widely applied due to its efficiency in solving many types of high-dimensional problems [152, 153, 154, 155]. To further improve the convergence speed of the coordinate descent method, researchers have successfully combined it with Nesterov's acceleration technique [37, 39, 31], and developed *accelerated coordinate descent* algorithms [69, 72, 73, 6, 64] which enjoy optimal worst-case convergence speed in theory, and much improved practical performance over vanilla coordinate descent. Very recently researchers have even proposed several successful variants of accelerated coordinate descent which are based on various schemes such as restart [40, 143], non-uniform sampling [74, 156] and Gauss-Southwell greedy selection rules [157].

### 5.1.1 Faster Coordinate Descent via Exploiting the Structure

While researchers have developed the so-called optimal coordinate descent algorithms for the composite optimization tasks (5.1), these algorithms do not take advantage of the prior information brought forth by the regularization term $g(x)$. Popular non-smooth regularization applied in machine learning and signal processing applications enforce the solution to have low-dimensional structure, for example the sparsity, group-sparsity or low-rank. In this work, by introducing a simple variant of the *accelerated proximal coordinate gradient* (APCG) algorithm of [6], we show that one can significantly improve the convergence speed of these methods if the prior information is properly exploited.

In chapter 4, we proposed and analyzed a structure-adaptive stochastic gradient method Rest-Katyusha [9] for more efficiently solving the large-scale composite optimization problems in the form of (2.3) in Section 2.1, by restarting the original Katyusha algorithm [4] at a period associated with the Restricted Strong-Convexity parameters. In this chapter, we extend the spirit of chapter 4 to design structure-adaptive variants of accelerated coordinate descent methods [6, 73, 72]. We highlight the following contributions of this chapter.

**Theoretical Contributions.**

We analyze the relationship between the solution's low-dimensional structure and the convergence speed of accelerated coordinate descent methods in the primal form. We choose to use the accelerated proximal coordinate descent method APCG [6, 64] as the foundation to build up our novel "Two-Stage APCG" method which is dedicated to actively exploit the intrinsic low-dimension structure of the solution prompted by the (non-smooth) regularization. The convergence analysis shows that our method exhibits global convergence: in the first stage, the method converges sublinearly to the vicinity of the solution, while in the second stage the method converges towards the solution with an accelerated linear rate with respect to the modified restricted strong convexity (RSC) [5] which scales with the solution's intrinsic dimension.

**Algorithmic Contributions.**

We propose an adaptive restart variant of our Two-Stage APCG algorithm which is motivated by our underlying theory and does not need explicit knowledge of the restricted strong convexity (RSC) parameter but still provides excellent practical performance, just like our Adaptive Rest-Katyusha in chapter 4. In practice the strong convexity and also restricted strong convexity parameter cannot be easily obtained beforehand in general practical setups, which is necessary for the accelerated methods to achieve an accelerated linear convergence rate [69, 71, 141]. To overcome this issue we propose an adaptive variant of the two-stage APCG method which is based on a simple heuristic scheme to estimate the RSC on the fly. Tested on a number of high-dimensional datasets, our experiments demonstrate the effectiveness of our algorithm.

## 5.2 Two-Stage APCG

We start by introducing the vanilla accelerated coordinate method APCG developed by [64], and then our new method which has the desirable structure-adaptive property. We first list some standard notations following the accelerated coordinate descent literature [73, 6, 64].

**Definition 5.2.1.** *(Block Coordinate Structure and Partial Gradients.) We split the full space $\mathbb{R}^m$ into d blocks of subspaces, that is, for any vector $x \in \mathbb{R}^m$ with*

$$\{x_{(i)} \in \mathbb{R}^{m_i},\ \ i = 1, ..., d,\ \ \sum_i m_i = m\}, \tag{5.2}$$

*there is a permutation matrix $U \in \mathbb{R}^{m \times m}$ with submatrices,*

$$\{U = [U_1, ..., U_d], U_i \in \mathbb{R}^{m \times m_i}, i = 1, ..., d\} \tag{5.3}$$

*such that $x = \sum_{i=1}^d U_i x_{(i)}$. We also define the partial gradient of the smooth function $f(.)$ w.r.t $x_{(i)}$ as:*

$$\triangledown_i f(x) = U_i^T \triangledown f(x). \tag{5.4}$$

*The regularization term is assumed to have a block-coordinate-wise separable structure:*

$$g(x) = \sum_{i=1}^d g_i(x_{(i)}). \tag{5.5}$$

We assume that $f(.)$ has block-coordinate-wise Lipschitz continuous gradient with parameter $L_i$ for each block of coordinates $i \in [1, d]$, and define a weighted norm

$$\|x\|_L = \big(\sum_{i=1}^d L_i \|x_{(i)}\|_2^2\big)^{1/2}. \tag{5.6}$$

We list the details of the APCG algorithm [64, Alg. 2] for strongly-convex functions:

```
**APCG**$(x_0, K, \alpha)$ – Initialize $z_0 = x_0,\ \alpha = \frac{\sqrt{\mu}}{d}$
For  $k = 0, 1, 2, ..., K$
    $y_k = \frac{x_k + \alpha z_k}{1+\alpha}$;
    pick $i_k \in [1, 2, ..., d]$ uniformly at random;
    $z_{k+1} = \arg\min_{x \in \mathbb{R}^d} \frac{\alpha d}{2}\|x - (1-\alpha)z_k - \alpha y_k\|_L^2 + \langle \nabla_{i_k} f(y_k), x_{(i_k)} \rangle + \lambda g_{i_k}(x_{(i_k)})$
                                                        $\rightarrow$ proximal coordinate descent
    $x_{k+1} = y_k + d\alpha(z_{k+1} - z_k) + d\alpha^2(z_k - y_k)$;           $\rightarrow$ momentum step
Return $x_{K+1}$.
```

with initialization $z_0 = x_0$, $\alpha = \frac{\sqrt{\mu}}{d}$, and we take the result of the last iterate ($x_{K+1}$) as the output. If the objective function $F$ is strongly-convex, then the APCG algorithm enjoys a Nesterov-type accelerated linear convergence rate. Similarly we also provide the details of the APCG algorithm for minimizing non-strongly-convex functions [64, Alg. 3], which we denote as APCG$_0$, with initialization $z_0 = x_0$ and $\alpha_{-1} = \frac{1}{d}$:

```
**APCG$_0$**$(x_0, K)$ – Initialize $z_0 = x_0,\ \alpha_{-1} = \frac{1}{d}$
For  $k = 0, 1, 2, ..., K$
    $\alpha_k = \frac{1}{2}(\sqrt{\alpha_{k-1}^4 + 4\alpha_{k-1}^2} - \alpha_{k-1}^2)$,
    $y_k = (1-\alpha_k)x_k + \alpha_k z_k$.
    pick $i_k \in [1, 2, ..., d]$ uniformly at random;
    $z_{k+1} = \arg\min_{x \in \mathbb{R}^d} \frac{\alpha d}{2}\|x - z_k\|_L^2 + \langle \nabla_{i_k} f(y_k), x_{(i_k)} \rangle + \lambda g_{i_k}(x_{(i_k)})$
                                                        $\rightarrow$ proximal coordinate descent
    $x_{k+1} = y_k + d\alpha_k(z_{k+1} - z_k)$;                          $\rightarrow$ momentum step
Return $x_{K+1}$.
```

If the objective function is convex but non-strongly-convex, the APCG$_0$ has an $O(1/k^2)$ accelerated sublinear convergence rate. These convergence rates match the optimal worse-case rates of Nesterov's accelerated gradient method [39] for $d = 1$ and improve upon the proximal coordinate descent [70] for $d > 1$.

To exploit the structure of the solution for even faster convergence, we propose variants of

**Algorithm 8** Two-Stage APCG

```
Inputs: $x^0$ and restricted strong-convexity parameter $\mu_c$, number of iterations $K_0$ for the first stage; $T \geq 1$; $\beta \geq 2$
1. First stage, start without $\mu_c$:
        $x^1 = \mathrm{APCG}_0(x^0, K_0)$        (5.7)
2. Second stage – exploit local accelerated linear convergence given by $\mu_c$
 Option 1: with $K = \left\lceil 2d\beta\sqrt{2+\frac{1}{\mu_c}} - 2d \right\rceil$
for $t = 1, \ldots, T$ do
        $x^{t+1} = \mathrm{APCG}_0(x^t, K)$        (5.8)
end for
Output: $x^{T+1}$
 Option 2: with $K = \left\lceil \frac{\log 16}{\log \frac{1}{1-\sqrt{\mu_c}/d}} \right\rceil$
for $t = 1, \ldots, T$ do
        $x^{t+1} = \mathrm{APCG}(x^t, K, \mu_c)$        (5.9)
end for
Output: $x^{T+1}$
```

accelerated coordinate descent algorithms base on the APCG in a similar manner to the work in chapter 4, under a two-stage splitting framework inspired by the RSC: at the first stage for warm-starting, we run the non-strongly-convex $\mathrm{APCG}_0$ algorithm to a neighborhood of the solution; at the second stage, since a linear convergence rate is expected due to the RSC, we have two choices: (1) periodically restart the non-strongly-convex $\mathrm{APCG}_0$ at a certain frequency w.r.t the RSC parameter $\mu_c$, which leads to our Option 1, (2) run the APCG algorithm with the momentum parameter $\alpha = \frac{\sqrt{\mu_c}}{d}$ and a restart period also w.r.t $\mu_c$, which leads to Option 2. We describe the two-stage APCG as Algorithm 8, where we use superscript $t$ to index outer-loop and subscript $k$ to index inner-loop of our algorithms.

We need to point out that our algorithm with Option 1 is a two-stage variant of the Restarted-APPROX algorithm of [40] which is also based on restarting the accelerated coordinate descent. This algorithm was originally designed for minimizing functions which satisfy a quadratic error bound condition – a condition which is also weaker than strong-convexity but does not encode the solution's structure enforced by regularization. The Restarted-APPROX algorithm currently on its own does not have theoretical convergence result under the RSC framework of [5] which is relevant to the purpose of this work.

### 5.2.1 Generic Assumptions

In this section we list out the assumptions which we required in our convergence proofs. Similar assumptions have been used in the previous chapter. However, some of these assumptions are tailored for the analysis of coordinate descent and hence appear to be slightly different.

**A. 5.** *(Block-Coordinate Smoothness.) Assume that $f(x)$ has block-coordinate-wise Lipschitz continuous gradient:*

$$\|\nabla_i f(x + U_i h_i) - \nabla_i f(x)\|_2 \leq L_i \|h_i\|_2, \tag{5.10}$$

$\forall h_i \in \mathbb{R}^{m_i}, i = 1, ..., d, x \in \mathbb{R}^m$.

This smoothness assumption is a classic assumption for RCD methods [158].

**A. 6.** *(Restricted Strong-Convexity.) With respect to the weighted norm:*

$$\|x\|_L = \sqrt{\sum_{i=1}^{d} L_i \|x_i\|_2^2}, \tag{5.11}$$

*the function $f(.)$ and $g(.)$ satisfies the following inequality with lower curvature parameter $\gamma$ and tolerance parameter $\tau$:*

$$f(x) - f(x^\star) - \langle \nabla f(x^\star), x - x^\star \rangle \geq \frac{\gamma}{2} \|x - x^\star\|_L^2 - \tau g^2(x - x^\star), \tag{5.12}$$

Note that, assumption **(A.6)** is a more general form of RSC condition[1] presented in **(A.2)** of the previous chapter.

Next, for the sake of self-consistency of this chapter, we recall the basic assumptions of RSC [5] on the regularization term in chapter 4:

**A. 7.** *(Subspace Decomposability.) Given a orthogonal subspace pair $(\mathcal{M}, \mathcal{M}^\perp)$ in $\mathbb{R}^m$, $g(.)$ is decomposable such that:*

$$g(a + b) = g(a) + g(b), \forall a \in \mathcal{M}, b \in \mathcal{M}^\perp. \tag{5.13}$$

**A. 8.** *(Sufficiency of Regularization.) Recall the definition of $x^\dagger$ by (2.2) in Section 2.1. The*

[1]The RSC was originally defined in the $\ell_2$ norm in [5]. We slightly generalize it here by using a weighted norm for the sharper analysis of the coordinate descent algorithms.

*regularization parameter $\lambda$ satisfies the following inequality with some constant $c \geq 1$:*

$$\lambda \geq (1 + \frac{1}{c}) g^\star(\nabla f(x^\dagger)). \tag{5.14}$$

For the analysis of Option 2, we need a further assumption namely the "Non-blowout" property in the literature [159, 43, 160]:

**A. 9.** *(Non-blowout Iterations.) If we start the APCG algorithm at a point $x_0$, and we assume that there exist a positive constant $1 \leq \omega < \infty$, such that the update sequence $\{x_k\}$ generated by the algorithm obeys the following inequality almost surely:*

$$F(x_k) - F^\star \leq \omega \left( F(x_0) - F^\star \right), \quad \forall k \tag{5.15}$$

We assume a relaxed non-blowout property of the APCG iterates, which essentially means that the iterates generated by the algorithm will have optimality gap bounded by that for the first iteration. This assumption hold true for accelerated full gradient and also non-accelerated coordinate descent with $\omega = 1$ which means the iterates are strictly non-blowout. However for accelerated coordinate descent such a result has not been shown and hence we provide it here as a relaxed assumption. Note that the analysis of our Option 1 does not need this assumption.

### 5.2.2 Preliminaries for the Analysis

Since we use a more general form of RSC condition associates with the weighted norm $\|\cdot\|_L$, we also need to slightly generalize the definition of subspace-compatibility for this chapter:

**Definition 5.2.2.** *(Subspace compatibility.) [5] With predefined $g(x)$, we define the subspace compatibility of a model subspace $\mathcal{M}$ as:*

$$\Phi(\mathcal{M}) := \sup_{v \in \mathcal{M} \setminus \{0\}} \frac{g(v)}{\|v\|_L}, \tag{5.16}$$

*when $\mathcal{M} \neq \{0\}$ and $\Phi(\{0\}) := 0$.*

The subspace compatibility leverages the low-dimensional structure of $x^\star$ into our analysis, for example, if $g(.) = \|.\|_1$, $m = d$, $\|x^\star\|_0 = s$, $L_i = \bar{L}\ \forall i$ and $\mathcal{M}$ is an $s$-dimensional subspace in $\mathbb{R}^m$, then we have $\Phi(\mathcal{M}) = \sqrt{s/\bar{L}}$.

With this new notion of subspace compatibility we are able to provide a generalized version of effective RSC (Lemma 4.3.1) tailored specifically for our analysis of coordinate descent algorithms, which enables us to link the solution's structure with the convergence behavior and quantify their dependence (we provide the proof of this lemma in the appendix of this chapter):

**Lemma 5.2.3.** *(Generalized Effective RSC) Under* ***A.5-8****, if further* ***A.6*** *holds with parameters* $(\gamma, \tau)$ *such that* $\tau\Phi^2(\mathcal{M}) < \frac{\gamma}{16(1+c)^2}$*, then with given* $(x^\star, x^\dagger)$ *and a value* $\eta > 0$*, and denote* $\varepsilon := 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_2 + 4g(x^\dagger_{\mathcal{M}^\perp})$ *, for any* $x$ *satisfies* $F(x) - F(x^\star) \leq \eta$ *for any optima* $x^\star$*, we have:*

$$F(x) - F^\star \geq \mu_c\|x - x^\star\|_L^2 - 2\tau(1+c)^2 v^2, \tag{5.17}$$

*where* $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M}) > 0$ *and* $v = \frac{\eta}{\lambda} + \varepsilon$*.*

We also list the convergence result of the APCG$_0$ which has been proven by [64]:

**Lemma 5.2.4.** *[64, Theorem 2.1] Under* ***A.5****, the* $K_0$*-th iteration of APCG*$_0$ *algorithm obeys:*

$$\mathbb{E}F(x_{K_0}) - F^\star \leq \left(\frac{2d}{2d + K_0}\right)^2 \mathcal{D}(x^0, x^\star) := \Omega_{K_0}, \tag{5.18}$$

*where* $\mathcal{D}(x^0, x^\star) := F(x^0) - F^\star + \frac{1}{2}\|x^0 - x^\star\|_L^2$*.*

### 5.2.3 Main Results

Now we are ready to present our main theorems for the two-stage APCG algorithm with Option 1 and Option 2 in this section, based on the RSC framework.

#### 5.2.3.1 Convergence Results of Option 1.

We start by our theorem on the objective gap convergence speed of Option 1 which is based on the periodic restart scheme:

**Theorem 5.2.5.** *Under* ***A.5 – 8****, if further* ***A.6*** *holds with parameters* $(\gamma, \tau)$ *such that* $\tau\Phi^2(\mathcal{M}) < \frac{\gamma}{16(1+c)^2}$*, and we run the two-stage APCG algorithm (Option 1) with* $K = \left\lceil 2d\beta\sqrt{2 + \frac{1}{\mu_c}} - 2d \right\rceil$*,* $K_0 \geq \left\lceil d\left(1 + \frac{2}{\rho\lambda}\right)\sqrt{\frac{8\tau(1+c)^2\mathcal{D}(x^0,x^\star)}{2\mu_c+1}} \right\rceil$ *with* $\beta \geq 2$*, then the following inequality holds:*

$$\mathbb{E}[F(x^{t+1}) - F^\star] \leq \max\left\{\varepsilon, \left(\frac{1}{\beta^2}\right)^t \Omega_{K_0}\right\} \tag{5.19}$$

*with probability at least* $1 - \rho$*.*

We can now summarize the iteration complexity of Option 1 as the following:

**Corollary 5.2.6.** *Under the same assumptions and parameter choices of Theorem 5.2.5, the total number of coordinate gradient evaluation of the Two-Stage APCG (Option 1) algorithm will need in order to achieve a* $\delta > \varepsilon$ *objective gap accuracy is:*

$$O\left(\frac{d}{\sqrt{\mu_c}}\right)\log\frac{1}{\delta} + K_0. \tag{5.20}$$

We can make the following observations.

**(Accelerated Linear Convergence under RSC Framework.)** The technical result presented in Theorem 5.2.5 and Corollary 5.2.6 demonstrates accelerated linear convergence rate for our two-stage APCG algorithm with Option 1 up to a statistical accuracy under the RSC assumption from [5].

**(Structure-Adaptive Convergence.)** The effective RSC $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})$ provides us a way to link the convergence speed of an algorithm with the structure of the solution. For example, if $c = 1$, $m = d$, $L_i = 1$ $\forall i$, $g(x) = \|x\|_1$ and $\|x^\star\|_0 = s$, then $\Phi^2(\mathcal{M}) = s$ and hence $\mu_c = \frac{\gamma}{2} - 32\tau s$. Further if $F(x)$ is a Lasso problem, then for a wide class of random design matrix we have $\tau = O(\frac{\log d}{n})$ and $\gamma > 0$. Moreover, [117] have shown that if the data matrix is a correlated Gaussian design matrix such that each row of it is i.i.d drawn from distribution $\mathcal{N}(0, H)$ where $H$ is the covariance matrix and we denote its largest and smallest singular value as $r_{\max}(H)$ and $r_{\min}(H)$, then it can be shown that $\gamma \geq \frac{r_{\min}(H)}{16}$ and $\tau \leq r_{\max}(H)\frac{81\log d}{n}$ with high probability.

**(The Early Iterations and High Probability Statement.)** From Theorem 5.2.5 we can see that the probability statement of the convergence result hangs on the choice of the number of iterations on the first stage. Such dependence is natural and within our expectation – the Effective RSC condition presented in Lemma 5.2.3 is non-vacuous only at a neighborhood of the solution, where the first-stage of our algorithm is aimed to reach.

**(Convergence on the Optimization Variable.)** Due to the RSC condition we can bound the solution distance to the global optimum by the objective optimality gap (aka, the convergence on the optimization variable). Such results demonstrate that the optimization error on the optimization variable also decays linearly up to a statistical accuracy scaled by a well-behaved constant factor as discussed by [5, 116]:

**Corollary 5.2.7.** *(Convergence of the Iterates) Under the same assumption and parameter choice of Theorem 5.2.5, the iterates generated by Two-Stage APCG (Option 1) obey the following inequality:*

$$\mathbb{E}\|x^{t+1} - x^\star\|_L^2 \leq \left(\frac{1}{\beta^2}\right)^t \frac{\Omega_{K_0}}{\mu_c} + \left(\frac{1}{\beta^4}\right)^t \frac{2\tau(1+c)^2\Omega_{K_0}^2}{\lambda^2\mu_c^2} + \frac{2\tau(1+c)^2}{\mu_c}\varepsilon^2. \tag{5.21}$$

**(Connection with Structure-Adaptive Convergence Result for Finite-Sum Optimization.)** It is worth noting that this extends the spirit of the Rest-Katyusha algorithm we proposed in the previous chapter, which is also inspired by and developed under the same RSC framework. The Rest-Katyusha algorithm is a restarted version of an accelerated variance-reduced SGD method of [4] for efficiently solving regularized empirical risk minimization with a finite-sum structure where $f(x) := \sum_i f_i(x)$ with a smoothness assumption on each $f_i$, while our coordinate descent method two-stage APCG is dedicated to minimizing block-coordinate-wise separable functions with a smoothness assumption on the blocks of coordinates (i.e. **A.5**). Because of this fundamental distinction, we provide here a different complexity result with the RSC framework which complements the contribution provided by the previous chapter.

**(The Optimal Choice of $\beta$.)** For Option 1 of our Two-Stage APCG there is a user defined parameter $\beta$. In theory, any $\beta \geq 2$ will provide us an accelerated linear rate. To be specific, to achieve an $\delta$-accuracy, the second stage algorithm needs to have:

$$\left\lceil 2d\beta\sqrt{2 + \frac{1}{\mu_c}} - 2d \right\rceil \log_{\beta^2}\frac{1}{\delta} \tag{5.22}$$

coordinate gradient oracle calls, and hence there is a clear trade-off on $\beta$. Similar to Rest-Katyusha, with some standard calculation one can conclude that the best choice of $\beta$ to achieve the optimal iteration complexity is roughly the Euler's number ($\approx 2.71$). We use this choice for our algorithm in the numerical experiments.

#### 5.2.3.2 Convergence Results of Option 2.

With the additional non-blowout assumption **A.9**, we are also able to provide a similar result for our second approach (with Option 2, we provide the proof of this theorem in the appendix of this chapter):

**Theorem 5.2.8.** *Under* ***A.5 – 9****, and if further* ***A.6*** *holds with parameters* $(\gamma, \tau)$ *such that* $\tau\Phi^2(\mathcal{M}) < \frac{\gamma}{16(1+c)^2}$ *and we run the Option 2 of the two-stage APCG algorithm with* $K = \left\lceil \frac{\log 16}{\log \frac{1}{1-\sqrt{\mu_c}/d}} \right\rceil$ *and* $K_0 = \left\lceil 8d(1+\frac{\omega}{\lambda\rho})\sqrt{(\sqrt{\frac{1}{\mu_c}}+1)\tau(1+c)^2\mathcal{D}(x^0, x^\star)} \right\rceil$*, then the following inequality holds:*

$$\mathbb{E}[F(x^{t+1}) - F^\star] \leq \max\left\{\varepsilon, \left(\frac{1}{4}\right)^t \Omega_{K_0}\right\} \tag{5.23}$$

*with probability at least* $1-\rho$*.*

Our convergence result for Option 2 provided by Theorem 5.2.8 is slightly weaker than Theorem 5.2.5 which is for Option 1, since Theorem 5.2.8 requires an additional assumption **A.9**. Again, based on the convergence result on the objective we can summarize the iteration complexity of the Two-Stage APCG algorithm with Option 2 as the following corollary:

**Corollary 5.2.9.** *Under the same assumptions and parameter choices of Theorem 5.2.8, the total number of coordinate gradient calculations the Two-Stage APCG (Option 2) algorithm needs in order to achieve a* $\delta > \varepsilon$ *objective gap accuracy is:*

$$O\left(\frac{1}{\log \frac{1}{1-\frac{\sqrt{\mu_c}}{d}}}\right) \log\frac{1}{\delta} + K_0. \tag{5.24}$$

The contraction factor $1 - \frac{\sqrt{\mu_c}}{d}$ occurs in (5.24) in a logarithmic term $\frac{1}{\log \frac{1}{1-\sqrt{\mu_c}/d}}$ which scales nearly as $\frac{1}{1-(1-\sqrt{\mu_c}/d)} = \frac{d}{\sqrt{\mu_c}}$. Hence we conclude that under the assumptions above, the Two-Stage APCG (Option 2) has a local accelerated linear convergence $O(\frac{d}{\sqrt{\mu_c}}\log\frac{1}{\delta})$.

Because of the RSC condition, the convergence of the iterates can be again easily derived for Option 2 similar to Corollary 5.2.7 and we do not illustrate this here.

## 5.3 Adaptive Two-Stage APCG

To the best of our knowledge, all the state-of-the-art accelerated randomized algorithms for solving the composite minimization task (5.1) in Section 5.1 require the explicit knowledge of the strong convexity parameter to run with an Nesterov-type accelerated linear convergence rate exactly. For the case where the data fidelity term $f(.)$ is strongly convex, it is difficult in general to calculate the strong convexity parameter before running the accelerated algorithms, let alone in our case, the restricted strong convexity. Here we propose an adaptive restart scheme for Two-Stage APCG based on a heuristic procedure for estimating $\mu_c$ on the fly with a small fraction of computational overhead. Similar ideas of adaptive restart have been applied in [42, 142, 43] and also our chapter 4 for deterministic and stochastic gradient algorithms with Nesterov's acceleration.

**(Adaptive Variant of Option 1.)**. First we observe that for $K = \left\lceil 2d\beta\sqrt{2+1/\mu_c} - 2d \right\rceil$, the convergence speed of the second stage algorithm reads:

$$\mathbb{E}_{\xi_t \backslash \xi_{t-1}} F(x^{t+1}) - F^\star \leq \frac{1}{\beta^2}[F(x^t) - F^\star]. \tag{5.25}$$

It has been shown by [43, Prop. 4] that $F(x) - F^\star$ can be lower bounded as $O(\|\mathcal{G}(x) - x\|_2^2)$, where $\mathcal{G}(x)$ is the composite gradient map:

$$\mathcal{G}(x) = \arg\min_{u \in \mathbb{R}^d} \frac{d \max_i L_i}{2}\|x - u\|_2^2 + \langle \nabla f(x), u - x\rangle + \lambda g(u). \tag{5.26}$$

Meanwhile we can upper bound this objective gap by $O(\|\mathcal{G}(x) - x\|_2^2)$ under some mild assumptions [43]. Inspired by such a property, we would like to exploit it as a tool to track the convergence speed of the objective gap, in order to evaluate the accuracy of the RSC parameter of the current iteration. If $\|\mathcal{G}(x^{t+1}) - x^{t+1}\|_2^2 \leq \frac{1}{\beta^2}\|\mathcal{G}(x^t) - x^t\|_2^2$ at $t$-th iteration, it is likely that we have underestimated the RSC parameter since if $\mu_0 \leq \mu_c$, (5.25) will always be satisfied. Hence we double the estimate. If otherwise, it is likely that the RSC parameter is overestimated and then we shrink the estimate.

In order to implement the tracking of the objective gap, an extra full gradient is needed to be calculated which will introduce a computational overhead compared to Algorithm 8. However such overhead is durable since the restart period $K$ is lower-bounded[2] by $6d$ , while the cost of a full gradient is at most $d$ times that of one coordinate gradient calculation, hence the overhead amounts $\frac{1}{6}$ of total iteration complexity at worst.

**(Adaptive Variant of Option 2.)** The Option 2 of the Two-Stage APCG algorithm can also be made adaptive with a similar idea of utilizing the composite gradient map to estimate the $\mu_c$

[2]By **A.5** and the definition of $\|.\|_L$, we have $\gamma \leq 1$.

**Algorithm 9** Adaptive Two-Stage APCG

```
Inputs: $(x^0, \mu_0, K_0, \beta, T)$
Initialize: $K = \left\lceil 2d\beta\sqrt{2 + \frac{1}{\mu_0}} - 2d \right\rceil$;
$x^1$ = APCG$_0$ $(x^0, K_0)$
Calculate the composite gradient map $\mathcal{G}(x^1)$ by eq:(5.71).
for $t = 1, \dots, T$ do
    $x^{t+1}$ = APCG$_0$ $(x^t, K)$
    ——Track the convergence speed :
        Calculate $\mathcal{G}(x^{t+1})$ by eq:(5.71)
    —— Update the estimate of RSC
        if $\|\mathcal{G}(x^{t+1}) - x^{t+1}\|_2^2 \leq \frac{1}{\beta^2}\|\mathcal{G}(x^t) - x^t\|_2^2$
         then $\mu_0 \leftarrow 2\mu_0$, else $\mu_0 \leftarrow \mu_0/2$.
    ——Adaptively tune the restart period :
        $K = \left\lceil 2d\beta\sqrt{2 + \frac{1}{\mu_0}} - 2d \right\rceil$
end for
```

on the fly. We include the details of the adaptive variant of Option 2 in the appendix of this chapter.

## 5.4 Numerical Experiments

This section provides the details of numerical results of our proposed algorithms for solving the Lasso regression problem [17, 16]:

$$x^\star \in \arg\min_{x \in \mathbb{R}^m} \left\{ F(x) := \frac{1}{2n}\|Ax - b\|_2^2 + \lambda\|x\|_1 \right\}, \tag{5.27}$$

| DATA SET | SIZE $(n, m)$ | REFERENCE |
|---|---|---|
| MADELON+ | (2000, 4000) | [137] |
| MARTI2 | (500, 1024) | [161] |
| RCV1 | (20242, 47236) | [137] |
| NEWS20 | (15935, 62061) | [150] |

**Table 5.1:** *Chosen Datasets for Lasso Regression*

We set all our examples with $A \in \mathbb{R}^{n \times m}$ where $n < m$, hence there is no explicit strong-convexity. We compare our algorithms with state of the art variance-reduced stochastic gradient algorithm Katyusha [4, Algorithm 2] which has an accelerated sub-linear convergence rate for non-strongly convex functions, and also the vanilla APCG method for non-strongly-convex

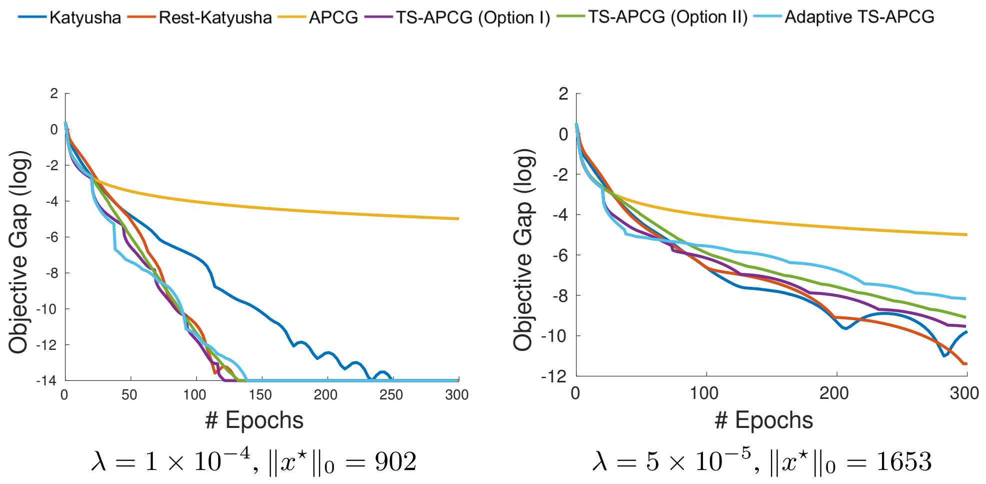


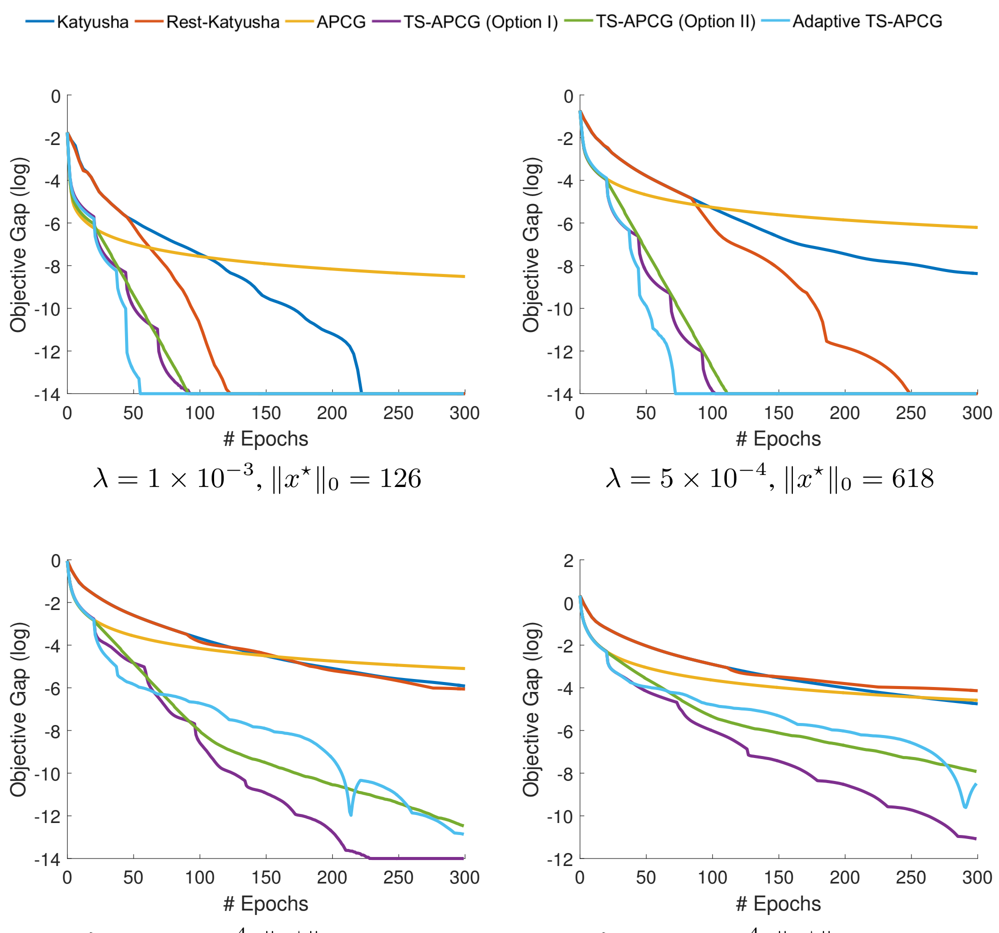
**Figure 5.1:** *Sparse regression experimental results comparing the proposed algorithms (two-stage APCG and its adaptive variant) with the vanilla APCG and Katyusha algorithm on RCV1 Dataset*

functions [64, Algorithm 3] as a comparison. We also include the Rest-Katyusha algorithm which also has provable structure-adaptive convergence. For the Rest-Katyusha algorithm and the two choices of our Algorithm 8 which need the explicit knowledge of the RSC parameter, we grid search to estimate it for the best practical performance. We use the theoretical step sizes for our algorithms as well as the APCG in all experiments. For the large datasets (RCV1 and News20) we use minibatch/block-coordinate versions, which are more relevant in parallel-computing scenarios. For the Katyusha and Rest-Katyusha we use the same minibatch size and grid-search the best possible step-sizes to provide the best performance.

| Experiment | $K_0/d$ | minibatch | $\mu_0$ for Alg.9 |
|---|---|---|---|
| Madelon+ | 20 | 1 | 0.1 |
| Marti2 | 20 | 1 | 0.1 |
| RCV1 | 20 | 80 | 0.1 |
| News20 | 20 | 100 | 0.1 |

**Table 5.2:** *Parameter Setting for Alg. 8 and Alg. 9*

For the Madelon dataset we add 3500 random features to its original 500 features. This represents the scenario where one may wish to use sparse regression via an $\ell_1$ penalty to nullify the effect of irrelevant features [104]. For all the four chosen datasets, the Two-Stage APCG algorithm and the adaptive-restart variant significantly outperform the original non-strongly-convex APCG in Lasso regression tasks, and often have superior performance over the Katyusha al-



$\lambda = 1 \times 10^{-3}$, $\|x^\star\|_0 = 126$

$\lambda = 5 \times 10^{-4}$, $\|x^\star\|_0 = 618$

$\lambda = 2 \times 10^{-4}$, $\|x^\star\|_0 = 1250$

$\lambda = 1 \times 10^{-4}$, $\|x^\star\|_0 = 1594$

**Figure 5.2:** *Sparse regression experimental results comparing the proposed algorithms (two-stage APCG and its adaptive variant) with the vanilla APCG and Katyusha algorithm on Madelon Dataset with 3500 Additional Random Features*

gorithm. From the results we see that while the original APCG method initially exhibits good objective reduction it has very slow final convergence – this demonstrates the necessity of our two-stage algorithmic structure for the accelerated coordinate descent.

Unlike experiments on the other datasets, for RCV1 dataset, the Katyusha and Rest-Katyusha appear competitive with two-stage APCG. This raises a practical question – for a given dataset, how to choose between the families of primal RCD and SGD (e.g. columns vs. rows). [162] provide an analysis comparing the primal RCD and the dual RCD (which also extends to the SGD-type methods in the primal, see [163]). Although restricted to $\ell_2$ regularization, their analysis suggests that the complexity of primal RCD and dual RCD is dependant on the dataset's characteristics such as the density and the distribution of the features. Using their complexity

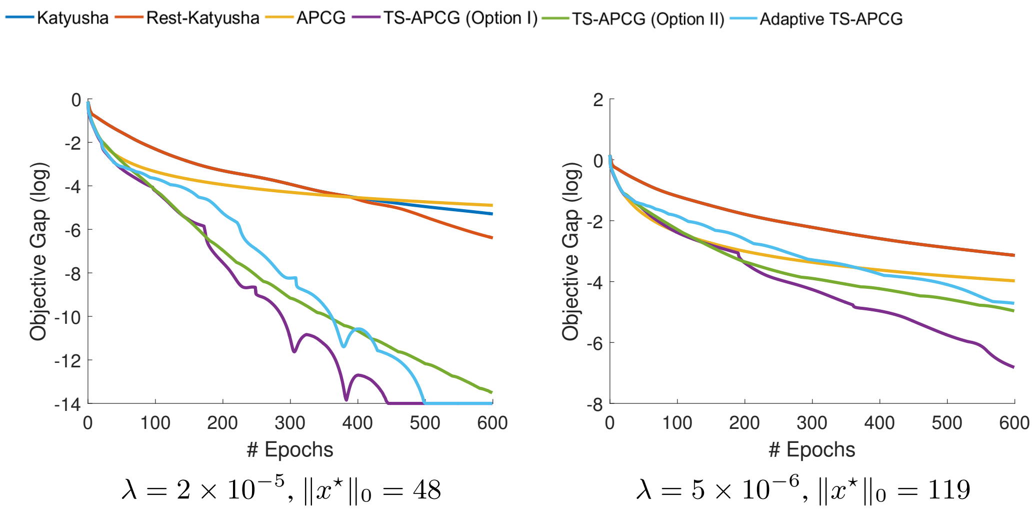


**Figure 5.3:** *Sparse regression experimental results comparing the proposed algorithms (two-stage APCG and its adaptive variant) with the vanilla APCG and Katyusha algorithm on MARTI2 Dataset*

bounds we found that in theory the RCV1 and News20 dataset prefer dual RCD for $\ell_2$ regularized ERM, while the Madelon and Marti2 prefer primal RCD, which is in broad agreement with our Lasso results.

These numerical results on real data sets have demonstrated the effectiveness of our approaches for accelerating the APCG method via actively exploiting the low dimensional structure of the solution. Non-structure-adaptive accelerated methods like Katyusha and APCG are blind to the restricted strong convexity. Hence when the solution is relatively sparse, or rather, the regularization parameter is relatively large for the data set, the two-stage APCG algorithms enjoy local linear convergence and often significantly outperform these baselines. For example, in Figure 5.1 where we present the numerical result for RCV1 dataset, while the solution is sufficiently sparse ($s = 902$), we can observe significant convergence-rate superiority of our two-stage APCG method. However, when the solution is not that sparse ($s = 1653$), we do not observe significant computational benefit of our methods for RCV1 dataset. Moreover our adaptive two-stage APCG algorithm appears to be very successful in estimating the RSC parameter and adaptively tuning the restart period on the fly such that it achieves comparable convergence speed to the two-stage APCG methods which need a reliable RSC estimate beforehand.

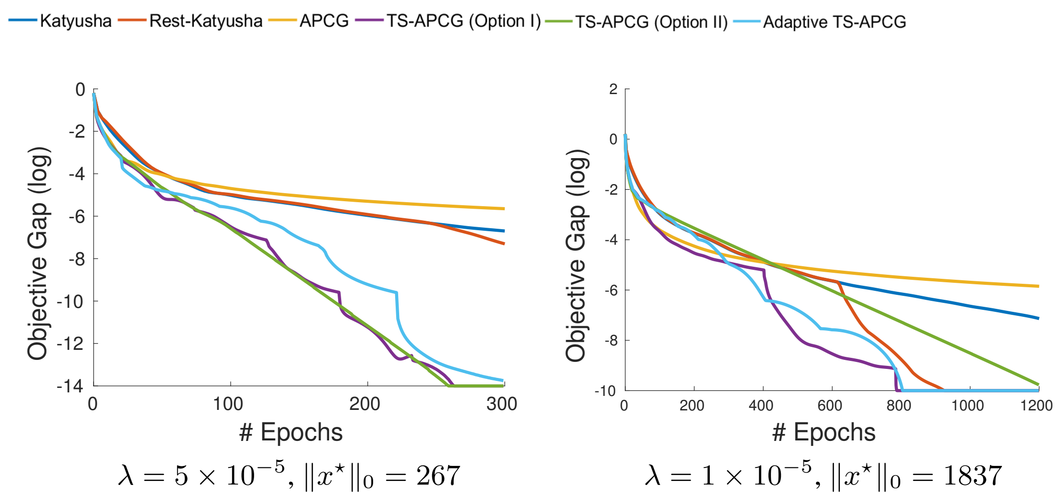


**Figure 5.4:** *Sparse regression experimental results comparing the proposed algorithms (two-stage APCG and its adaptive variant) with the vanilla APCG and Katyusha algorithm on the News20 Dataset (Class 1).*

## 5.5 Concluding Remarks

In this chapter, we provide theoretical and algorithmic contributions to coordinate descent optimization. We analyze the structure-adaptive convergence of a simple variant (namely the Two-stage APCG) of accelerated RCD based on the RSC framework of [5]. Moreover, we propose an adaptive-restart that does not require the explicit knowledge of RSC but estimates it on the fly. We validate the effectiveness of our approach via numerical experiments on sparse regression tasks. This work opens up the potential to develop even faster structure-adaptive accelerated coordinate descent methods incorporating importance sampling [74] for better iteration complexity, screening-rules [164] to predict the zero-elements for sparse regression and skip redundant updates, and continuation methods [159] for even faster initial convergence, etc.

## 5.A Appendix

### 5.A.1 The Proof for Option 1

#### 5.A.1.1 The Proof for Lemma 5.2.3

Let us denote $\Delta = x - x^{\dagger}$. Since we have assumed $F(x) - F(x^{\star}) \leq \eta$, then we also have $F(x) - F(x^{\dagger}) \leq \eta$, hence:

$$f(x^{\dagger} + \Delta) + \lambda g(x^{\dagger} + \Delta) \leq f(x^{\dagger}) + \lambda g(x^{\dagger}) + \eta, \tag{5.28}$$

then substract both side with $\langle \nabla f(x^{\dagger}), \Delta \rangle$ and rearrange:

$$f(x^{\dagger} + \Delta) - f(x^{\dagger}) - \langle \nabla f(x^{\dagger}), \Delta \rangle + \lambda g(x^{\dagger} + \Delta) - \lambda g(x^{\dagger}) \leq -\langle \nabla f(x^{\dagger}), \Delta \rangle + \eta. \tag{5.29}$$

Due to the convexity of $f(.)$ we immediately have:

$$\begin{aligned} \lambda g(x^{\dagger} + \Delta) - \lambda g(x^{\dagger}) &\leq -\langle \nabla f(x^{\dagger}), \Delta \rangle + \eta \\ &\leq g^{*}(\nabla f(x^{\dagger})) g(\Delta) + \eta \\ &\leq \frac{\lambda}{1 + \frac{1}{c}} g(\Delta) + \eta. \end{aligned}$$

Hence by dividing both sides with $\lambda$ and then applying the decomposability of $g$ we have:

$$g(x^{\dagger} + \Delta) - g(x^{\dagger}) \leq \frac{1}{1 + \frac{1}{c}} [g(\Delta_{\mathcal{M}}) + g(\Delta_{\mathcal{M}^{\perp}})] + \frac{\eta}{\lambda}. \tag{5.30}$$

Meanwhile a lower bound on the left-hand-side has been provided in [5], which reads:

$$g(x^{\dagger} + \Delta) - g(x^{\dagger}) \geq g(\Delta_{\mathcal{M}^{\perp}}) - 2g(x^{\dagger}_{\mathcal{M}^{\perp}}) - g(\Delta_{\mathcal{M}}). \tag{5.31}$$

By combining these two bounds we have:

$$\begin{aligned} &g(\Delta_{\mathcal{M}^{\perp}}) + g(\Delta_{\mathcal{M}}) + \frac{(1 + \frac{1}{c})\eta}{\lambda} \\ &\leq (1 + \frac{1}{c}) g(\Delta_{\mathcal{M}^{\perp}}) - 2(1 + \frac{1}{c}) g(x^{\dagger}_{\mathcal{M}^{\perp}}) - (1 + \frac{1}{c}) g(\Delta_{\mathcal{M}}). \end{aligned} \tag{5.32}$$

Then we have:

$$\begin{aligned}\frac{1}{c}g(\Delta_{\mathcal{M}^\perp}) &\leq (2+\frac{1}{c})g(\Delta_{\mathcal{M}}) + 2(1+\frac{1}{c})g(x^\dagger_{\mathcal{M}^\perp}) + \frac{(1+\frac{1}{c})\eta}{\lambda} \\ g(\Delta_{\mathcal{M}^\perp}) &\leq (1+2c)g(\Delta_{\mathcal{M}}) + 2(1+c)g(x^\dagger_{\mathcal{M}^\perp}) + \frac{(1+c)\eta}{\lambda} \\ g(\Delta) &\leq (2+2c)(g(\Delta_{\mathcal{M}}) + g(x^\dagger_{\mathcal{M}^\perp})) + \frac{(1+c)\eta}{\lambda}.\end{aligned}$$

Now let $\Delta_x := x - x^\star$ where $x$ satisfies $F(x) - F(x^\star) \leq \eta$, and $\Delta^\star := x^\star - x^\dagger$. Due to the fact that $x^\star$ is the optimal point, $\eta$ can be set as 0 if $x = x^\star$, then:

$$g(\Delta^\star) \leq (2+2c)(g(\Delta^\star_{\mathcal{M}}) + g(x^\dagger_{\mathcal{M}^\perp})). \tag{5.33}$$

Now we are able to bound $g(\Delta_x)$:

$$\begin{aligned}& g(\Delta_x) \\ \leq\ & g(\Delta) + g(\Delta^\star) \\ \leq\ & (2+2c)g(\Delta_{\mathcal{M}}) + (2+2c)g(\Delta^\star_{\mathcal{M}}) + (4+4c)g(x^\dagger_{\mathcal{M}^\perp}) + \frac{(1+c)\eta}{\lambda} \\ \leq\ & (1+c)\left[2g(\Delta_{\mathcal{M}}) + 2g(\Delta^\star_{\mathcal{M}}) + 4g(x^\dagger_{\mathcal{M}^\perp}) + \frac{\eta}{\lambda}\right].\end{aligned}$$

By the definition of the subspace compatibility $\Phi(\mathcal{M}) := \sup_{v\in\mathcal{M}\backslash\{0\}} \frac{g(v)}{\|v\|_L}$ we can write:

$$\begin{aligned}g(\Delta_x) &= g(x - x^\star) \\ &\leq (1+c)[2\Phi(\mathcal{M})\|x - x^\star\|_L + 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_L + 4g(x^\dagger_{\mathcal{M}^\perp}) + \frac{\eta}{\lambda}] \\ &\leq (1+c)\left[2\Phi(\mathcal{M})\|x - x^\star\|_L + v\right],\end{aligned}$$

where we denote $\varepsilon := 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_L + 4g(x^\dagger_{\mathcal{M}^\perp})$ and $v := \frac{\eta}{\lambda} + \varepsilon$. Then because of the fact that $(a+b)^2 \leq 2a^2 + 2b^2$ we have:

$$g^2(x - x^\star) \leq (1+c)^2\left[8\Phi^2(\mathcal{M})\|x - x^\star\|^2_L + 2v^2\right]. \tag{5.34}$$

Due to **A.6** we can write:

$$\begin{aligned} & f(x) - f(x^\star) - \langle \nabla f(x^\star), x - x^\star \rangle \\ \geq \quad & \frac{\gamma}{2}\|x - x^\star\|_L^2 + \tau(1+c)^2 \left[8\Phi^2(\mathcal{M})\|x - x^\star\|_L^2 + 2v^2\right] \\ \geq \quad & \left[\frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})\right] \|x - x^\star\|_L^2 - 2\tau(1+c)^2 v^2, \end{aligned}$$

Then because $g(.)$ is convex, we can write:

$$g(x) - g(x^\star) - \langle \partial g(x^\star), x - x^\star \rangle \geq 0, \tag{5.35}$$

and also:

$$\begin{aligned} & F(x) - F^\star - \langle \nabla f(x^\star) + \partial g(x^\star), x - x^\star \rangle \\ \geq \quad & \left[\frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})\right] \|x - x^\star\|_L^2 - 2\tau(1+c)^2 v^2. \end{aligned}$$

By first order optimality condition we have $\langle \nabla f(x^\star) + \partial g(x^\star), x - x^\star \rangle \geq 0$, hence we justify the claim.

#### 5.A.1.2 The Proof for Theorem 5.2.5, Corollary 5.2.6 and 5.2.7

We first define a sequence of random variable $\xi_t$ which is the realization of the random choices of coordinates from the 0-th iteration to the end of $t$-th iteration of Two-stage APCG (Option 1). According to the convergence result of APCG, after the first stage we have:

$$\mathbb{E}_{\xi_0} F(x^1) - F^\star \leq \epsilon_1 := \Omega_{K_0}. \tag{5.36}$$

Then with Markov inequality, at a probability at least $1 - \frac{\varrho}{2}$ we have:

$$F(x^1) - F^\star \leq \frac{2}{\rho}\epsilon_1. \tag{5.37}$$

Now we define three sequences through which we will achieve the proof via induction: $\epsilon_{t+1} = \frac{1}{\beta^2}\epsilon_t$, $\rho_{t+1} = \frac{1}{\beta}\rho_t$ with $\rho_1 := \rho$, and $v_t = \frac{2\epsilon_t}{\lambda\rho_t} + \varepsilon$.

**Induction step 1:** we first reformulate the effective RSC presented in Lemma 5.2.3 as the following:

$$\|x-x^\star\|_L^2 \leq \frac{1}{\mu_c}\left[F(x)-F^\star+2\tau(1+c)^2v^2\right], \tag{5.38}$$

and we can have:

$$\begin{aligned}
&\mathbb{E}_{\xi_1\backslash\xi_0}F(x^2)-F^\star \\
\leq\;& \left(\frac{2d}{2d+K}\right)^2[F(x^1)-F^\star]+\left(\frac{2d}{2d+K}\right)^2\frac{1}{2\mu_c}[F(x^1)-F^\star+2\tau(1+c)^2v_1^2] \\
=\;& \frac{4d^2+\frac{2d^2}{\mu_c}}{(k+2d)^2}[F(x^1)-F^\star]+\frac{4d^2\tau(1+c)^2v_1^2}{\mu_c(2d+K)^2}.
\end{aligned}$$

By taking expectation on both sides over $\xi_0$, we have:

$$\begin{aligned}
&\mathbb{E}_{\xi_1}F(x^2)-F^\star \\
\leq\;& \frac{4d^2+\frac{2d^2}{\mu_c}}{(2d+K)^2}\epsilon_1+\frac{4d^2\tau(1+c)^2v_1^2}{\mu_c(2d+K)^2} \\
\leq\;& \frac{4d^2+\frac{2d^2}{\mu_c}}{(2d+K)^2}\epsilon_1+\frac{4d^2\tau(1+c)^2}{\mu_c(2d+K)^2}\left(\frac{2\epsilon_1}{\rho\lambda}+\epsilon_1\right)^2,
\end{aligned}$$

where the second inequality holds due to $\epsilon_t > \varepsilon\ \ \forall t$ . Then we set:

$$\frac{4d^2\tau(1+c)}{\mu_c}\left(\frac{2\epsilon_1}{\rho\lambda}+\epsilon_1\right)^2 \leq (4d^2+\frac{2d^2}{\mu_c})\epsilon_1. \tag{5.39}$$

Hence:

$$\left(\frac{2}{\rho\lambda}+1\right)^2\epsilon_1 \leq \frac{2\mu_c+1}{2\tau(1+c)^2}. \tag{5.40}$$

Since $\epsilon_1 = \frac{4d^2\mathcal{D}(x^0,x^\star)}{(2d+K_0)^2}$, it is enough to set:

$$K_0 = \left\lceil d\left(\frac{2}{\rho\lambda}+1\right)\sqrt{\frac{8\tau(1+c)^2\mathcal{D}(x^0,x^\star)}{2\mu_c+1}}\right\rceil, \tag{5.41}$$

to ensure that:

$$\mathbb{E}_{\xi_1}F(x^2)-F^\star \leq \frac{8d^2+\frac{4d^2}{\mu_c}}{(K+2d)^2}\epsilon_1. \tag{5.42}$$

Then if we choose:

$$K = \left\lceil 2d\beta\sqrt{2+\frac{1}{\mu_c}}-2d\right\rceil, \tag{5.43}$$

we can ensure that:

$$\mathbb{E}_{\xi_1} F(x^2) - F^\star \leq \frac{1}{\beta^2}\epsilon_1. \tag{5.44}$$

**Induction step 2:** At iteration $t+1$, due to the induction hypothesis $\mathbb{E}_{\xi_{t-1}} F(x^t) - F^\star \leq \epsilon_t = \frac{\epsilon_{t-1}}{\beta^2}$ we have:

$$\begin{aligned}
& \mathbb{E}_{\xi_t} F(x^{t+1}) - F^\star \\
\leq\ & \frac{4d^2 + \frac{2d^2}{\mu_c}}{(2d+K)^2}\mathbb{E}_{\xi_{t-1}}[F(x^t) - F^\star] + \frac{4d^2\tau(1+c)^2 v_t^2}{\mu_c(2d+K)^2} \\
=\ & \frac{4d^2 + \frac{2d^2}{\mu_c}}{(2d+K)^2}\epsilon_t + \frac{4d^2\tau(1+c)^2 v_t^2}{\mu_c(2d+K)^2} \\
\leq\ & \frac{4d^2 + \frac{2d^2}{\mu_c}}{(2d+K)^2}\epsilon_t + \frac{4d^2\tau(1+c)^2}{\mu_c(2d+K)^2}\left(\frac{2\epsilon_t}{\rho_t\lambda} + \epsilon_t\right)^2,
\end{aligned}$$

Then we set:

$$\frac{4d^2\tau(1+c)}{\mu_c}\left(\frac{2\epsilon_1}{\rho_t\lambda} + \epsilon_t\right)^2 \leq (4d^2 + \frac{2d^2}{\mu_c})\epsilon_t, \tag{5.45}$$

and reformulate it as:

$$\left(\frac{2}{\rho_t\lambda} + 1\right)^2 \epsilon_t \leq \frac{2\mu_c + 1}{2\tau(1+c)^2}. \tag{5.46}$$

Since we have chosen $\rho_t = \frac{1}{\beta}\rho_{t-1}$, $\epsilon_t = \frac{1}{\beta^2}\epsilon_{t-1}$ with $\beta \geq 2$,

$$\left(\frac{2}{\rho_t\lambda} + 1\right)^2 \epsilon_t \leq \left(\frac{2}{\rho_{t-1}\lambda} + 1\right)^2 \epsilon_{t-1} \leq \left(\frac{2}{\rho\lambda} + 1\right)^2 \epsilon_1. \tag{5.47}$$

Hence with the same choice of $K_0$ and $K$ in induction step 1, with probability at least (due to the choice $\beta \geq 2$):

$$1 - \sum_{i=1}^{t}\frac{\rho_i}{2} \geq 1 - \frac{\rho\beta}{2(\beta-1)} \geq 1 - \rho, \tag{5.48}$$

we can ensure:

$$\mathbb{E}_{\xi_t} F(x^{t+1}) - F^\star \leq \frac{1}{\beta^2}\epsilon_t. \tag{5.49}$$

Thus finishes the proof of Theorem 5.2.5.

In summary, to achieve $\mathbb{E}_{\xi_t} F(x^{t+1}) - F^\star \leq \delta$, the coordinate gradient calculation at the second stage should be:

$$\left\lceil 2d\beta\sqrt{2 + \frac{1}{\mu_c}} - 2d \right\rceil \log_{\beta^2}\frac{1}{\delta}, \tag{5.50}$$

and we justify the claim in Corollary 5.2.6.

We can also provide the convergence result of the optimization variable by the Effective RSC given by Lemma 5.2.3. At point $x^{T+1}$, we set $\eta = F(x^{T+1}) - F^\star$, and we have:

$$\begin{aligned}
& \|x^{T+1} - x^\star\|_L^2 \\
\leq \; & \frac{F(x^{T+1}) - F^\star + 2\tau(1+c)^2 v^2}{\mu_c} \\
\leq \; & \frac{F(x^{T+1}) - F^\star + 2\tau(1+c)^2[(\frac{\eta}{\lambda})^2 + \varepsilon^2]}{\mu_c} \\
\leq \; & \frac{F(x^{T+1}) - F^\star}{\mu_c} + \frac{2\tau(1+c)^2}{\lambda^2\mu_c}\left(F(x^{T+1}) - F^\star\right)^2 + \frac{2\tau(1+c)^2\varepsilon^2}{\mu_c} \\
\leq \; & \left(\frac{1}{\beta^2}\right)^T \frac{\Omega_{K_0}}{\mu_c} + \frac{2\tau(1+c)^2\Omega_{K_0}^2}{\lambda^2\mu_c^2}\left(\frac{1}{\beta^4}\right)^T + \frac{2\tau(1+c)^2}{\mu_c}\varepsilon^2.
\end{aligned}$$

Hence we have finished the proofs for both Theorem 5.2.5, Corollary 5.2.6 and Corollary 5.2.7.

### 5.A.2 Convergence Proof for Option 2

First we present a key lemma for two-stage APCG with Option 2, which is extended from the convergence proof of [6, 64]:

**Lemma 5.A.1.** *Given ($x^\star$, $x^\dagger$), and denote $\varepsilon := 2\Phi(\mathcal{M})\|x^\dagger - x^\star\|_L + 4g(x^\dagger_{\mathcal{M}^\perp})$. Assume* **A.5 - 9**, *the updates of the second stage of the Two-Stage APCG obey:*

$$\mathbb{E}_{\xi_K^t \backslash \xi_K^{t-1}}[F(x_0^{t+1})] - F^\star \leq \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^K \cdot 2\left[F(x_0^t) - F^\star\right] + 2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right)v^2, \tag{5.51}$$

*where $\mu_c = \frac{\gamma}{2} - 8\tau(1+c)^2\Phi^2(\mathcal{M})$, $v = \frac{\eta}{\lambda} + \varepsilon$, $F(x_k^t) - F(x^\star) \leq \eta$ for all $t \geq 1$ and $k$.*

*Proof.* At each iteration, the APCG algorithm chooses a coordinate uniformly at random to perform updates. The update sequences $x_{k+1}^t$ and $z_{k+1}^t$ depend on the realization of the following random variable which we denote as $\xi_k^t$:

$$\xi_k^t = \{i_k^t, i_{k-1}^t, ..., i_1^t, i_0^t, i_k^{t-1}, ..., i_0^{t-1}, ..., i_k^0, ..., i_0^0\}, \tag{5.52}$$

and for the randomness within a single outer-loop of Two-Stage APCG we specifically denote $\xi_k^t \backslash \xi_k^{t-1}$ as

$$\xi_k^t \backslash \xi_k^{t-1} = \{i_k^t, i_{k-1}^t, ..., i_1^t, i_0^t\} \tag{5.53}$$

We achieve the proof of this lemma by extending the original proof for strongly-convex APCG [64, Theorem 2.1]. By replacing the original strong-convexity in [64, equation 3.21] with the effective RSC we immediately have the following:

$$\begin{aligned} &\mathbb{E}_{i_k^t}[f(x_{k+1}^t) + \lambda \hat{g}_{k+1}^t - F^\star + \frac{\mu_c}{2}\|z_{k+1}^t - x^\star\|_L^2] \\ \leq\ &\left(1 - \frac{\sqrt{\mu_c}}{d}\right)\mathbb{E}_{i_{k-1}^t}[f(x_k^t) + \lambda \hat{g}_k^t - F^\star + \frac{\mu_c}{2}\|z_k^t - x^\star\|_L^2] + \frac{2\tau(1+c)^2}{d}v^2, \end{aligned}$$

(Due to the complicated notations in the original work of [64], we refer the readers to [64, Lemma 3.3] for the detailed definition of $\hat{g}_k^t$, which is a convex combination of $g(z_0^t)$, $g(z_1^t)$,

$g(z_2^t)$ ..... $g(z_k^t)$). Then we roll up the bound:

$$\begin{aligned}
&\mathbb{E}_{\xi_k^t \backslash \xi_K^{t-1}}[f(x_{k+1}^t) + \lambda \hat{g}_{k+1}^t - F^\star + \frac{\mu_c}{2}\|z_{k+1}^t - x^\star\|_L^2] \\
\leq\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^k [F(x_0^t) - F^\star + \frac{\mu_c}{2}\|x_0^t - x^\star\|_L^2] + \frac{1-(1-\sqrt{\mu_c}/d)^{k-1}}{1-(1-\sqrt{\mu_c}/d)} \frac{2\tau(1+c)^2}{d} v^2 \\
\leq\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^k [F(x_0^t) - F^\star + \frac{\mu_c}{2}\|x_0^t - x^\star\|_L^2] + \frac{2\tau(1+c)^2}{\sqrt{\mu_c}} v^2 \\
\leq\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^k [2F(x_0^t) - 2F^\star + 2(1+c)^2\tau v^2] + \frac{2\tau(1+c)^2}{\sqrt{\mu_c}} v^2 \\
\leq\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^k \cdot 2\left[F(x_0^t) - F^\star\right] + 2\tau(1+c)^2 \left(\sqrt{\frac{1}{\mu_c}} + 1\right) v^2,
\end{aligned}$$

where we utilize the effective RSC again to bound the term $\frac{\mu_c}{2}\|x_0^t - x^\star\|_L^2$.

Since $\hat{g}_{k+1}^t \geq g(x_{k+1}^t)$ as shown in [64, Lemma 3.3], by simplifying the left hand side we can have:

$$\begin{aligned}
\mathbb{E}_{\xi_k^t \backslash \xi_K^{t-1}}[F(x_{k+1}^t)] - F^\star \leq & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^K \cdot 2\left[F(x_0^t) - F^\star\right] \\
& + 2\tau(1+c)^2 \left(\sqrt{\frac{1}{\mu_c}} + 1\right) v^2.
\end{aligned} \tag{5.54}$$

Thus finishes the proof since $F(x_0^{t+1}) = F(x_{K+1}^t)$. □

### 5.A.2.1 Proof of Theorem 5.2.8 and Corollary 5.2.9

Now we are ready to present the proof of Theorem 5.2.8.

*Proof.* We follow a similar procedure in [5] and [118] to roll up the residual term $v^2$. According to [64] for the first stage of the algorithm we have:

$$\mathbb{E}_{\xi^0}[F(x^1)] - F^\star \leq \epsilon_1 := \left(\frac{2d}{2d + K_0}\right)^2 \mathcal{D}(x^0, x^\star),$$

where $\mathcal{D}(x^0, x^\star) := [F(x^0) - F^\star + \frac{1}{2}\|x^0 - x^\star\|_L^2$. Then with Markov inequality, at a probability at least $1 - \frac{\rho}{2}$ we have:

$$F(x^1) - F^\star \leq \frac{2}{\rho}\epsilon_1. \tag{5.55}$$

Next we derive the complexity of the second stage. We define three sequences through which we will achieve the proof via induction: $\epsilon_{t+1} = \frac{1}{4}\epsilon_t$, $\rho_{t+1} = \frac{1}{2}\rho_t$ with $\rho_1 := \rho$, and $v_t = \frac{\omega\epsilon_t}{\lambda\rho_t} + \varepsilon$.

**Induction part 1:** We turn to our first outer iteration in the second stage of the algorithm. by Lemma 5.A.1 we have:

$$\begin{aligned}\mathbb{E}_{\xi^1\setminus\xi^0}[F(x^2)] - F^\star \leq (1 - \sqrt{\mu_c}/d)^K \cdot 2(F(x^1) - F^\star) \\ + 2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right)v_1^2.\end{aligned} \tag{5.56}$$

Now we take the expectation over $\xi_K^0$:

$$\begin{aligned}\mathbb{E}_{\xi^1}(F(x^2) - F^\star) \leq (1 - \sqrt{\mu_c}/d)^K \cdot 2\mathbb{E}_{\xi_K^0}(F(x^1) - F^\star) \\ + 2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right)v_1^2,\end{aligned} \tag{5.57}$$

where we set:

$$2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right)v_1^2 \leq \frac{\epsilon_1}{8}. \tag{5.58}$$

Note that $v_1 = \frac{\omega\epsilon_1}{\lambda\rho_t} + \varepsilon$ and $\epsilon_1 > \varepsilon$ it is enough if the following inequality is satisfied:

$$2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right)\left(\frac{\omega\epsilon_1}{\lambda\rho_t} + \epsilon_1\right)^2 \leq \frac{\epsilon_1}{8} \tag{5.59}$$

equivalently:

$$2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right)\left(\frac{\omega}{\lambda\rho} + 1\right)^2 \frac{4d^2\mathcal{D}(x^0, x^\star)}{(2d + K_0)^2} \leq \frac{\epsilon_1}{8}. \tag{5.60}$$

Hence it is enough to set:

$$K_0 = \left\lceil 8d(1 + \frac{\omega}{\lambda\rho})\sqrt{(\sqrt{\frac{1}{\mu_c}} + 1)\tau(1+c)^2\mathcal{D}(x^0, x^\star)} \right\rceil \tag{5.61}$$

Then if we choose:

$$K = \left\lceil \frac{\log 16}{\log\frac{1}{(1 - \sqrt{\mu_c}/d)}} \right\rceil, \tag{5.62}$$

we can ensure that:

$$\mathbb{E}_{\xi^1}(F(x^2) - F^\star) \leq \frac{\epsilon_1}{8} + \frac{\epsilon_1}{8} = \frac{\epsilon_1}{4} = \epsilon_2. \tag{5.63}$$

**Induction part 2:** For $(t+1)$-th outer iteration, by induction hypothesis for $t$-th outer iteration which reads: $\mathbb{E}_{\xi^{t-1}}F(x^t) - F^\star \leq \frac{\epsilon_{t-1}}{4} = \epsilon_t$, we can write:

$$\begin{aligned}\mathbb{E}_{\xi^t \backslash \xi^{t-1}}(F(x^{t+1}) - F^\star) \leq (1 - \frac{\sqrt{\mu_c}}{d})^K \cdot 2(F(x^t) - F^\star)\\ + 2\tau(1+c)^2 \left(\sqrt{\frac{1}{\mu_c}} + 1\right) v_t^2,\end{aligned} \tag{5.64}$$

with probability at least $1 - \frac{\rho_t}{2}$. Then we take the expectation over $\xi_K^{t-1}$:

$$\begin{aligned}\mathbb{E}_{\xi^t}(F(x^{t+1}) - F^\star) \leq (1 - \frac{\sqrt{\mu_c}}{d})^K \cdot 2\mathbb{E}_{\xi^{t-1}}(F(x^t) - F^\star)\\ + 2\tau(1+c)^2 \left(\sqrt{\frac{1}{\mu_c}} + 1\right) v_t^2,\end{aligned} \tag{5.65}$$

where we need:

$$2\tau(1+c)^2 \left(\sqrt{\frac{1}{\mu_c}} + 1\right) v_t^2 \leq \frac{\epsilon_t}{8}. \tag{5.66}$$

Since we have chosen that $\rho_t = \frac{1}{2}\rho_{t-1}$ and $\epsilon_t = \frac{1}{4}\epsilon_{t-1}$, then $v_t \leq v_{t-1} \leq .. \leq v_1$, the above inequality is satisfied by our choice of $K_0$.

Again if we choose:

$$K = \left\lceil \frac{\log(16)}{\log \frac{1}{(1 - \sqrt{\mu_c}/d)}} \right\rceil, \tag{5.67}$$

we can ensure that:

$$\mathbb{E}_{\xi^t}(F(x^{t+1}) - F^\star) \leq \frac{\epsilon_t}{8} + \frac{\epsilon_t}{8} = \frac{\epsilon_t}{4} = \epsilon_{t+1}. \tag{5.68}$$

with probability at least $1 - \sum_{i=1}^t \frac{\rho_i}{2} \geq 1 - \rho$, , for $\delta \geq \varepsilon$. Hence we finish the induction and the proof of Theorem 5.2.8.

In summary for Two-Stage APCG if we choose $K := \left\lceil \frac{\log 16}{\log \frac{1}{(1-\sqrt{\mu_c}/d)}} \right\rceil$, if the number of coordinate gradient oracle calls $N$ satisfies:

$$N := tK + K_0 \geq \left\lceil \frac{\log 16}{\log \frac{1}{(1-\sqrt{\mu_c}/d)}} \right\rceil \log_4(\frac{[F(x^1) - F^\star]}{\delta}) + K_0, \tag{5.69}$$

we have $\mathbb{E}_{\xi^{t-1}}F(x^t) - F^\star \leq \delta$ , which is claimed in Corollary 5.2.9. □

### 5.A.3 Adaptive Two-Stage APCG (Option 2)

In this appendix we provide a heuristic approach of estimating $\mu_c$ for the two-stage APCG (Option 2).

We describe the intuition of this procedure. First we observe that for $F(x^t) - F^\star < 1$, the convergence speed of the second stage algorithm reads:

$$\begin{aligned}
&\mathbb{E}_{\xi_K^t \setminus \xi_K^{t-1}}[F(x^{t+1})] - F^\star \\
\leq\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^K 2\left[F(x^t) - F^\star\right] + 2\tau(1+c)^2\left(\sqrt{\frac{1}{\mu_c}} + 1\right) v_t^2 \\
\approx\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^K 2\left[F(x^t) - F^\star\right] + o\left[F(x^t) - F^\star\right] \\
\approx\ & \left(1 - \frac{\sqrt{\mu_c}}{d}\right)^K 2\left[F(x^t) - F^\star\right].
\end{aligned}$$

Directly using this relationship to check the convergence speed is impossible because $F^\star$ is unknown beforehand, but it has been shown in [43, Prop. 4] that $F(x) - F^\star$ can be lower bounded as:

$$F(x) - F^\star \geq O(\|\mathcal{G}(x) - x\|_2^2), \tag{5.70}$$

where $T(x)$ is the composite gradient map:

$$\mathcal{G}(x) = \arg\min_{q \in \mathbb{R}^d} \frac{d \max_i L_i}{2}\|x - q\|_2^2 + \langle \nabla f(x), q - x\rangle + \lambda g(q), \tag{5.71}$$

and meanwhile there is also upper bound : $F(x) - F^\star \leq O(\|\mathcal{G}(x) - x\|_2^2)$.

Hence our heuristic procedure's checking condition is built based on a simplified version of the above relationship:

$$\|\mathcal{G}(x^{t+1}) - x^{t+1}\|_2^2 \lesssim C\left(1 - \frac{\sqrt{\mu_c}}{d}\right)^K \|\mathcal{G}(x^t) - x^t\|_2^2 \tag{5.72}$$

where the variable $C$ represent the strictness of the condition. In the adaptive algorithm we check the condition (5.72) every $K = \left\lceil \frac{\log 16}{\log \frac{1}{(1 - \sqrt{\mu_t}/d)}} \right\rceil$ of iterations where $\mu_t$ is the current estimate of $\mu_c$, if it is violated we suspect that our estimation of $\mu_c$ is too large and hence we shrink it by a factor of 2 and then restart the second stage algorithm, otherwise we double the estimate to ensure that we choose the estimation of $\mu_c$ as aggressively as possible. If we

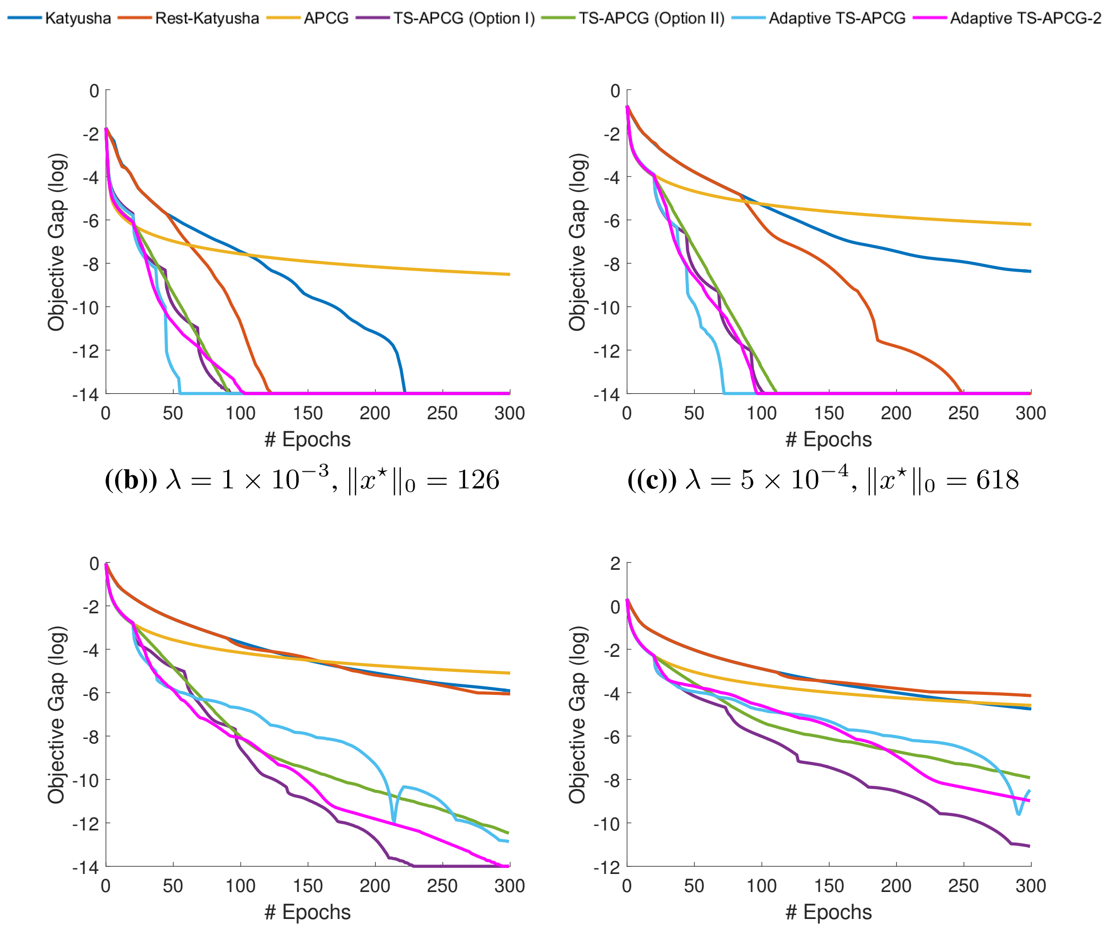
**Algorithm 10** Adaptive Two-Stage APCG - 2 $(x^0, \mu_1, K_0, C, T)$

```
$x^1$ = APCG$_0$ $(x^0, K_0)$
Calculate the composite gradient map $\mathcal{G}(x^1)$ by eq:(5.71).
**for** $t = 1, \ldots, T$ **do**
    $x^{t+1}$ = APCG $(x^t, K, \mu_t)$
    —–Track the convergence speed :
      Calculate $\mathcal{G}(x^{t+1})$ by eq:(5.71)
    —– Update the estimate of RSC
      **if** $\|\mathcal{G}(x^{t+1}) - x^{t+1}\|_2^2 \lesssim C\left(1 - \frac{\sqrt{\mu_t}}{d}\right)^K \|\mathcal{G}(x^t) - x^t\|_2^2$
       **then** $\mu_{t+1} \leftarrow 2\mu_t$, **else** $\mu_{t+1} \leftarrow \mu_t/2$.
    —–Adaptively tune the restart period :
      $K = \left\lceil \frac{\log 16}{\log \frac{1}{(1-\sqrt{\mu_t}/d)}} \right\rceil$
      **if** $\mu_{t+1} \leq 2^{-5}\mu_{t-4}$ **then** $C \leftarrow 2C$
      **if** $\mu_{t+1} \geq 2^5\mu_{t-4}$ **then** $C \leftarrow \max(1, \frac{C}{2})$
    **end if**
**end for**
```

observe that the algorithm is shrinking the $\mu_c$ for a number of times in a row, we suspect that the algorithm's checking condition is too strict and hence we double $C$ to relax the condition.

#### 5.A.3.1 Additional Experimental Results for the Adaptive Variant of Option 2

In this section we present an additional lasso experimental result with the Adaptive Two-Stage APCG-2 algorithm (pink lines) on Madelon dataset with extra 3500 random features. We set the initial guess of the RSC parameter $\mu_1 = 0.1$, the same as the adaptive variant of Option 1 described in the main text. We see that the adaptive variant of the two-stage APCG's option 2 also can achieve comparable results without the explicit knowledge of $\mu_c$ but estimate it on the fly:



**((b))** $\lambda = 1 \times 10^{-3}$, $\|x^\star\|_0 = 126$

**((c))** $\lambda = 5 \times 10^{-4}$, $\|x^\star\|_0 = 618$

**((d))** $\lambda = 2 \times 10^{-4}$, $\|x^\star\|_0 = 1250$

**((e))** $\lambda = 1 \times 10^{-4}$, $\|x^\star\|_0 = 1594$

**Figure 5.5:** *Additional sparse regression results for the Option 2 of adaptive two-stage APCG on Madelon dataset with additional random features ($A \in \mathbb{R}^{2000\times 4000}$)*

# Chapter 6
# Limitation and Practical Acceleration of Stochastic Gradient Algorithms in Inverse Problems

## 6.1 Stochastic Optimization in Imaging Inverse Problems

While having been a proven success both in theory and in machine learning applications, there are very few convincing results so far in the literature which report the performance of the stochastic gradient methods in imaging applications (except for tomography reconstruction [165, 166, 77]), which also involve large-scale optimization tasks in the same form as (2.3) in Section 2.1. In this chapter we investigate the practical performance of such methods, using space-varying deblurring as a running example. Such tasks can be generally formulated as the following:

$$x^\star \in \arg\min_{x \in \mathcal{X}} \left\{ F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \lambda g(x) \right\}, \tag{6.1}$$

where $\mathcal{X} \in \mathbb{R}^d$ is a closed convex set and we denote:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) := \frac{1}{n}\sum_{i=1}^{n} \bar{f}(a_i, b_i, x), \tag{6.2}$$

as the data fidelity term. In imaging tasks, the variable $x$ represents the vectorized image, while $(a_1, a_2, ...a_n)$ represent the forward model/measurements, $(b_1, b_2, ..., b_n)$ denote the observations. One of the most typical examples would be the X-ray CT imaging, where we use vectors $(a_1, a_2, ...a_n)$ to discretely model the X-ray measurements come out from the beam source, and meanwhile $(b_1, b_2, ..., b_n)$ will be the observation data collected at the X-ray sensor array.

In (6.1), each $f_i(x) := \bar{f}(a_i, b_i, x)$ is assumed to be convex and smooth, while the regularization term $g(x)$ is a simple convex function and is possibly non-smooth. With Nesterov's acceleration [37, 39], researchers [55, 4, 140] have developed several "optimal" algorithms which can provably achieve the worse-case optimal convergence rate for (6.1).

We make the following contributions:

**(Evaluating the limitation of stochastic gradient algorithms.)** We investigate the fundamental limit of possible acceleration of a stochastic gradient method over its full gradient counterpart by measuring the *Stochastic Acceleration* (SA) factor which is based on the ratio of the Lipschitz constants of the minibatched stochastic gradient and the full gradient. We discover that the SA factor is indeed able to characterize the potential of an optimization task being speedily solved by applying randomization techniques.

**(Breaking the computational bottleneck of expensive/multiple proximal operators for momentum SGD.)** Another factor in image processing practice which significantly affects the SGD-type methods' actual performance is the frequent calculation of the costly proximal operator for the regularization terms which have a linear operator, such as the TV semi-norm – SGD methods need to calculate it much more frequently than full gradient methods. Moreover most of the fast SGD methods can not cope with more than one non-smooth regularization term. To overcome these we propose an accelerated primal-dual SGD algorithm which can efficiently handle (1) regularization with a linear operator, (2) multiple regularization terms, while (3) maintaining Nesterov-type accelerated convergence speed in practice.

## 6.2 A Deblurring Experiment

Image deblurring is an important type of inverse problem in the field of image processing and have been studied intensely during the recent decades. For uniform deblurring, due to the cyclic structure of the deconvolution, FFT-based ADMM[1] variants have shown to be remarkably efficient [87, 167, 168] when compared to classic gradient-based solvers such as FISTA [35]. Such techniques, although being computationally efficient and exceedingly practical, are specifically tailored to a restricted range of problems where the observation models are diagonalizable by a DFT. For image deblurring, it is often not realistic to assume that the images observed by the imaging devices are uniformly blurred [169]. If the blurring is different across the image, then the efficient implementation of ADMM is not applicable in general, while the standard ADMM and deterministic gradient methods such as FISTA can be computationally burdensome if the

[1]The computation-demanding sub-problems of *alternating direction method of multipliers* (ADMM) at this case can be solved with an efficient matrix inversion by FFT due to the cyclic structure of the uniform deconvolution

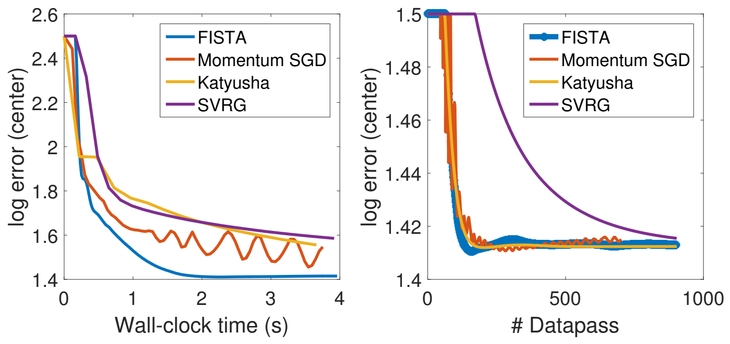


**Figure 6.1:** *The estimation error plot for the deblurring experiment. The plots correspond to the estimation error of the central part (100 by 100) of the image.*

image to be deblurred has a large size. In this work we focus on the space-varying deblurring where the stochastic gradient based solvers can have potential and possibility to provide better convergence over the deterministic algorithms.

We start by a simple space-varying deblurring [169] example where the central part (sized 128 by 128) of the "Kodim05" image from *Kodak Lossless True Color Image Suite* [170] is blurred with a space-varying blur kernel which imposes less blurring at the center but increasingly severe blurring towards the edge. For the shape of the blur kernel, we choose the out-of-focus kernel provided in [167]. We also add a small amount of noise to the blurred image.

We test the effectiveness of several algorithms by solving the same TV-regularized least-squares problem, to get an estimation of the ground truth image. The algorithms we test in the experiments include the accelerated full gradient method FISTA [35], SGD with momentum [53], the proximal SVRG [60] and its accelerated variant, Katyusha [4]. Perhaps surprisingly, on this experiment we report a negative result for the randomized algorithms. The most efficient solver in this task is the full gradient method FISTA both in terms of wall clock time and number of datapasses. The state-of-the-art stochastic gradient method Katyusha even cannot beat FISTA in terms of epoch counts. For all the randomized algorithms we choose a minibatch size which is 10 percent of the total data size. For stochastic gradient methods, a smaller minibatch size in this case did not provide better performance in datapasses and significantly slowed down running time due to the multiple calls on proximal operator.

We are aware that [147] have run a uniform image deblurring example where they compare their stochastic variance-reduced gradient algorithm – minibatch S2GD with the full gradient method FISTA [35]. However their result does not actually indicate superior performance of stochastic gradient methods for the following reasons: (1) Their convergence plot is only in epoch counts and they did not provide the plot in terms of actual run time. The epoch counts does not reflect the computational cost of the proximal operator which usually cannot be neglected in image processing applications, such as the proximal operators for $\ell_1$ penalty on the wavelet domain, total-variation penalty and the nuclear-norm penalty. The real performance of stochastic gradient methods are usually limited at these cases, since they need much more calls on the proximal operator than the full gradient methods. (2) Even in terms of epoch counts, the experiment does not shown significant improvement of mS2GD over the full gradient method FISTA in uniform image deblurring task.

## 6.3 Limitations of Stochastic Optimization

The previous example appears to be contrary to the popular belief among the stochastic optimization community, that stochastic gradient methods are much faster in terms of iteration complexity than deterministic gradient methods in solving large scale problems: to be specific – to achieve an objective gap suboptimality of $F(x) - F(x^\star) \leq \varepsilon$, optimal stochastic gradient methods needs only $\Theta\left(n + \sqrt{nL/\varepsilon}\right)$ evaluations of $\triangledown f_i$, while $\Theta\left(n\sqrt{L/\varepsilon}\right)$ for optimal full gradient methods. Where is the loophole?

It is often easily ignored that the complexity results above are derived under different smoothness assumptions. For the convergence bound for full gradient, the full smooth part of the cost function $f(.)$ is assumed to be $L$-smooth, while for the case of stochastic gradient, every individual function $f_i(.)$ is assumed to be $L$-smooth. Now we can clearly see the subtlety: to compare these complexity results and make meaningful conclusions, one has to assume that these two Lipschitz constants are roughly the same. While this can be true, and is true for many problems, there are exceptions – image deblurring is one of them.

Given a minibatch index $[S_0, S_1, S_2, ..., S_K]$ such that:

$$f(x) = \frac{1}{K}\sum_{k=1}^{K} f_{S_k}(x), \quad f_{S_k}(x) := \frac{K}{n}\sum_{i \in S_k} f_i(x), \tag{6.3}$$

In order to identify the potential of a certain optimization problem to be more efficiently solved using stochastic gradient methods, we start by deriving a motivating theorem comparing generically the convergence of the optimal full gradient methods as well as the optimal stochastic gradient methods.

### 6.3.1 Analysis

We start with the standard smoothness assumption:

**A. 10.** *(Smoothness of the Full-Batch and the Mini-Batches.) $f(.)$ is $L_f$-smooth and each $f_{S_k}$ is $L_b$-smooth, that is:*

$$f(x) - f(y) - \nabla f(y)^T(x-y) \leq \frac{L_f}{2}\|x-y\|_2^2, \quad \forall x, y \in \mathcal{X}, \tag{6.4}$$

*and*

$$f_{S_k}(x) - f_{S_k}(y) - \nabla f_{S_k}(y)^T(x-y) \leq \frac{L_b}{2}\|x-y\|_2^2, \tag{6.5}$$

$\forall x, y \in \mathcal{X}$.

We consider comparing two classes of algorithm: the optimal deterministic gradient methods which meet the lower bound presented in Theorem 2.3.4 and the optimal stochastic gradient methods which are able to match the lower bound presented in Theorem 2.4.1. The FISTA algorithm and the Katyusha algorithm are typical instances from these two classes of algorithms.

**Definition 6.3.1.** *(The class of optimal deterministic gradient algorithms.) A deterministic gradient method $\mathcal{A}_{\text{full}}$ is called optimal if for any $s \geq 1$, the update of $s$-th iteration $x^s_{\mathcal{A}_{\text{full}}}$ satisfies:*

$$F(x^s_{\mathcal{A}_{\text{full}}}) - F^\star \leq \frac{C_1 L_f \|x^0 - x^\star\|_2^2}{s^2}, \tag{6.6}$$

*for some positive constant $C_1$.*

It is know that the FISTA algorithm satisfies this definition with $C_1 = 4$ [35]. We also define the class for optimal stochastic gradient methods:

**Definition 6.3.2.** *(The class of optimal stochastic gradient algorithms.) A stochastic gradient method $\mathcal{A}_{\text{stoc}}$ is called optimal if for any $s \geq 1$ and $m \geq 1$, after a number of $s \cdot m$ stochastic gradient evaluations, the output of the algorithm $x^s_{\mathcal{A}_{\text{stoc}}}$ satisfies:*

$$\mathbb{E}F(x^s_{\mathcal{A}_{\text{stoc}}}) - F^\star \leq \frac{C_2[F(x^0) - F^\star]}{s^2} + \frac{C_3 L_b \|x^0 - x^\star\|_2^2}{ms^2}, \tag{6.7}$$

*for some positive constants $C_2$ and $C_3$.*

Note that the accelerated stochastic variance-reduced gradient methods such as Katyusha [4], MiG[66] and Point-SAGA [67] satisfy this definition with different values of $C_2$ and $C_3$.

Now we are ready to present the main theorem, which follows from simply combining the existing convergence results of the lower bounds for the stochastic and deterministic first-order optimization [31, 54].

**Theorem 6.3.3.** *Under **A.10**, let $g(.) = 0$, $m = K$. Denote an optimal deterministic algorithm $\mathcal{A}_{\text{full}}$ which satisfies Def. 6.3.1, and an optimal stochastic gradient algorithm $\mathcal{A}_{\text{stoc}}$ which satisifes Def. 6.3.2. For a sufficiently large dimension $d$ and $\mathcal{X} = \left\{x \in \mathbb{R}^d : \|x\|_2^2 \leq 1\right\}$, assume that there exists a set of convex and smooth functions $f_i \in \mathcal{F}_{L_b}^{1,1}(\mathcal{X})$, such that $\frac{1}{K}\sum_{i=1}^{K} f_i = f \in \mathcal{F}_{L_f}^{1,1}(\mathcal{X})$ simultaneously satisfies the lower bounds provided by Theorem 2.3.4 and 2.4.1, and all minimizers of it live in the relative interior of $\mathcal{X}$, then for this function $f$ we can have:*

$$c_0 \cdot \frac{L_b}{KL_f} \leq \frac{\mathbb{E}f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star}{f(x^s_{\mathcal{A}_{\text{full}}}) - f^\star} \leq c_1 \cdot \frac{L_b}{KL_f} + c_2 \tag{6.8}$$

*for some positive constants $c_0$, $c_1$, $c_2$ which do not depend on $L_b$, $L_f$ and $K$.*

From this theorem we can see that with the same epoch count, the ratio of the objective-gap sub-optimality achieved by $\mathcal{A}_{\text{full}}$ and $\mathcal{A}_{\text{stoc}}$ can be upper and lower bounded by $\Theta(\frac{L_b}{KL_f})$ at the worst case. Although the constants seem pessimistic, it is within our expectation since the lower bounds on the convergence speed of both algorithms are derived on the worst possible function which satisfies **A.10**. Motivated by the theory, we propose to evaluate the potential of stochastic acceleration simply by the ratio $\frac{L_b}{KL_f}$ which dominates our upper and lower bounds in Theorem 6.3.3.

### 6.3.2 Evaluating the Limitation of SGD-type Algorithms

We introduce a heuristic metric called the *Stochastic Acceleration* (SA) factor. The curve for SA factor as a function of the minibatch number $K$ (for a given minibatch pattern) is able to provide a way of characterizing inherently whether for a given inverse problem and a certain minibatch sampling scheme, randomized gradient methods should be preferred over the deterministic full gradient methods or not.

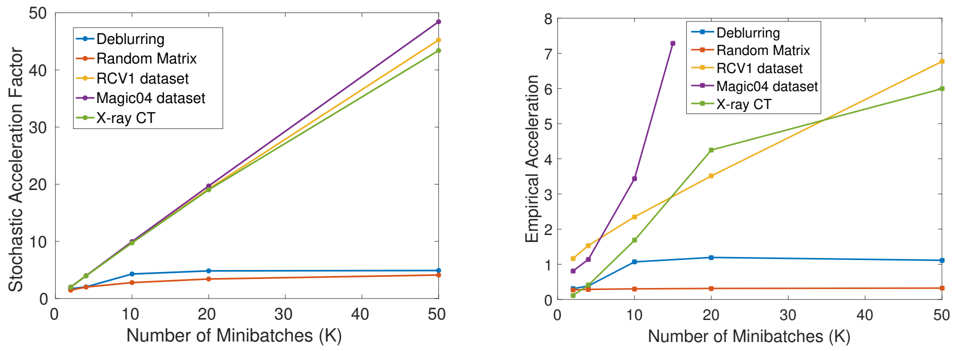


**Figure 6.2:** *Left: Stochastic Acceleration (SA) factor of inverse problems with different forward operators, Right: Empirical observation comparing the objective gap convergence of Katyusha and FISTA algorithm in 15 epochs.*

**Definition 6.3.4.** *For a given data-partitioning index $\bar{S} = [S_1, ...S_K]$, the Stochastic Acceleration (SA) factor is defined as:*

$$\Upsilon(\bar{S}) = \frac{KL_f}{L_b} \tag{6.9}$$

We test the least squares loss function $f(x) = \|Ax - b\|_2^2$ with different types of forward operator. In this case we have

$$f(x) = \|Ax - b\|_2^2 = \frac{1}{K}\sum_{k=1}^{K} f_{S_k}(x), \tag{6.10}$$

$$f_{S_k}(x) := K\|A_{S_k}x - b_{S_k}\|_2^2, \tag{6.11}$$

The examples of forward operator $A$ we consider include the space-varying deblurring ($A_{\text{blur}} \in \mathbb{R}^{262144\times262144}$), a random compressed sensing matrix with i.i.d Guassian random entries (with a size $A_{\text{rand}} \in \mathbb{R}^{500\times2000}$), a fan beam X-ray CT operator ($A_{\text{CT}} \in \mathbb{R}^{91240\times65536}$), and linear regression problem on two machine learning datasets: RCV1 dataset ($A_{\text{rcv1}} \in \mathbb{R}^{20242\times47236}$), and Magic04 ($A_{\text{magic04}} \in \mathbb{R}^{19000\times50}$). For the X-ray CT image reconstruction example and deblurring example we use TV regularization, while for the rest of the examples we use $\ell_1$ regularization. The data-partition we choose is the interleaving sampling, where the $k$-th minibatch is formed as the following:

$$f_{S_k}(x) := \frac{K}{n}\sum_{i=1}^{\lfloor n/K \rfloor} f_{k+iK}(x) = K\sum_{i=1}^{\lfloor n/K \rfloor}(a_{k+iK}^T x - b_{k+iK}) \tag{6.12}$$

From the result demonstrated in the Figure 6.2 we find that indeed the stochastic methods have a limitation on some optimization problems like deblurring and inverse problems with random matrices, where we see that the curve for SA factor of such problems stays low and flat even when we increase the number of minibatches. For the machine learning datasets and X-ray CT imaging, the SA factor increases rapidly and almost linearly as we increase the number of minibatches, which is in line with observations in machine learning on the superiority of SGD and also the observation in CT image reconstruction of the benefits of using the ordered-subset methods [171]. The curves for the SA factor on the left figure qualitatively predict the empirical comparison result of the Katyusha and FISTA algorithms shown on the right, where we observe that Katyusha offers no acceleration over the FISTA on either the deblurring or the Gaussian random inverse problem, but significantly outperforms FISTA on the other cases. Indeed, positive results for applying SGD-type algorithms on these problems are well-known already [47, 60, 171]. Hence we have shown that the SA factor we propose is useful in characterizing whether an inverse problem is inherently a suitable candidate for stochastic gradient methods.

## 6.4 Practical Acceleration for SGD

The previous section suggests that stochastic gradient methods do not always offer an intrinsic advantage for some problems. There are also several other causes for this failure. The most obvious one is that stochastic gradient methods in the primal need to calculate the proximal operator many more times than full gradient methods and hence slow down dramatically the run time. Moreover, in image processing practice often more than one non-smooth regularization term is used, where most of the existing fast stochastic methods such as Katyusha are inapplicable.

To avoid the frequent oracle call on the TV proximal operator, we can first reformulate the original optimization problem as a convex-concave saddle-point form. To be specific, we consider the following optimization problem:

$$x^\star \in \min_{x\in\mathbb{R}^d} \left\{f(x) + \lambda g(Dx) + \gamma h(x)\right\}, \tag{6.13}$$

where $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(a_i, x)$ is the data-fidelity term, $g(Dx)$ is a regularization term with a linear operator – for example the TV regularization ($g(.) = \|.\|_1$, $D \in \mathbb{R}^{r\times d}$ is the differential

**Algorithm 11** Accelerated Primal-Dual SGD (Acc-PD-SGD)

```
Initialization: $x^0 = v^0 = v^{-1} \in \mathrm{dom}(g)$, the step size sequences $[\alpha], [\eta], [\theta], l = 0$.
for $t = 1$ to $N$ do
    $x^t \leftarrow \frac{(3t-2)v^{t-1} + tx^{t-1} - (2t-4)v^{t-2}}{2t+2}$, $x_0 \leftarrow x^t$,
    $z_0 \leftarrow x^t$, $y_0 \leftarrow Dx_0$                                  $\rightarrow$ Katyusha-X Momentum
    for $k = 0$ to $K - 1$ do
        $l \leftarrow l + 1$
        $y_{k+1} = \mathrm{prox}_{\lambda g^*}^{\alpha_l}(y_k + \alpha_l D z_k)$                    $\rightarrow$ Dual Ascent
        Pick $i \in [1, 2, ...K]$ uniformly at random
        $\nabla_k = \nabla f_{S_i}(x_k)$ ;
        $x_{k+1} = \mathrm{prox}_{\gamma h}^{\eta_l}\left(x_k - \eta_l(D^T y_{k+1} + \nabla_k)\right)$      $\rightarrow$ Primal Descent
        $z_{k+1} = x_{k+1} + \theta_l(x_{k+1} - x_k)$                    $\rightarrow$ Innerloop Momentum
    end for
    $v^t \leftarrow x_K$
end for
Output: $x^t$
```

operator), and $h(x)$ is a second convex regularizer. As we have shown in Section 2.5 the saddle-point formulation can be written as:

$$[x^\star, y^\star] = \min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^r} f(x) + h(x) + y^T Dx - \lambda g^*(y) \tag{6.14}$$

The most famous algorithm for solving this saddle-point problem is the primal-dual hybrid gradient (PDHG, also known as the Chambolle-Pock algorithm) [76, 79], which interleaves the update of the primal variable $x$ and the dual variable $y$ throughout the iterates. With this reformulation the linear operator $D$ and the function $g(.)$ are decoupled and hence one can divide-and-conquer the expensive TV-proximal operator with the primal-dual gradient methods. The stochastic variant of the PDHG for the saddle-point problem (6.14) has been very recently proposed by Zhao & Cevher [78, Alg.1, "SPDTCM"] and shown to have state-of-the-art performance when compared to PDHG, stochastic ADMM [172] and stochastic proximal averaging [173].

Additionally, since the effect of acceleration given by Nesterov's momentum appears to be very important [2], we also need to consider a way to ensure that our method is accelerated. Since the SPDTCM method does not have Nesterov-type acceleration, we propose a variant of it which adopts the outerloop acceleration scheme given by the Katyusha-X algorithm [174], which is

[2]For instance, in the experiment from Section 6.2, the non-accelerated methods like SVRG perform badly compared to all the accelerated methods.

a simplified variant of the Katyusha algorithm [4]. We observe that such a momentum step is important for the stochastic primal-dual methods in this application. We present our method as Algorithm 11. One can directly choose the same step-size sequences $[\alpha]$, $[\eta]$, $[\theta]$ as suggested in [78, Section 2.3].

## 6.5 Numerical Experiments

### 6.5.1 Space-Varying Image Deblurring Experiment

We test our algorithm and compare with FISTA [35] and the SPDTCM [78] on a space-varying deblurring task for images sized 512 by 512, with a space-varying out-of-focus blur kernel, and TV-regularization. All algorithms are initialized with a backprojection. We use a machine with 1.6 GB RAM, 2.60 GHz Intel Core i7-5600U CPU and MATLAB R2015b.

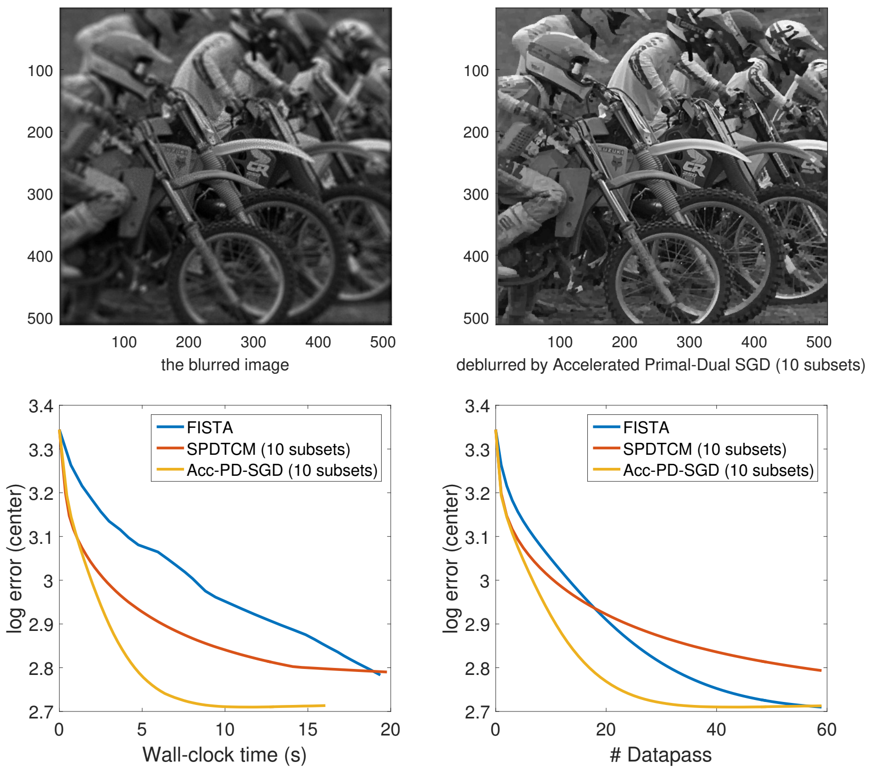


**Figure 6.3:** *The estimation error plot for the deblurring experiment with TV-regularization. Image: Kodim05, with an additive Guassian noise (variance 1).*

We plot the estimation error $\log_{10} \|x - x^{\dagger}\|_2^2$ in Figure 6.3 for each algorithm, where $x^{\dagger}$ denotes the ground truth image. We observe a roughly $4\times$ improvement in run time compared to FISTA since our algorithm can avoid the heavy cost of the TV proximal operator[3] while maintaining the fast convergence provided by Nesterov-type momentum and randomization. We also report a significant improvement over the SPDTCM algorithm both in time and iteration complexity.

In terms of datapasses (number of epochs), the SPDTCM does not show any advantage over the deterministic method FISTA, while our Acc-PD-SGD with 10 minibatches is able to achieve 2 times acceleration over FISTA.

We also report that in this experiment, if we further increase the number of subsets of SPDTCM and Acc-PD-SGD, we do not observe faster convergence for these algorithms. In other words, no matter how we increase the number of subsets, this 2-time acceleration (in terms of number of datapasses) is the limit of our algorithm – such a trend is successfully predicted by the SA factor shown in the Figure 6.2 (left), where we can see that the curve of the SA factor for deblurring task goes flat instead of increasing after the number of minibatches $K > 10$.

### 6.5.2 X-Ray Computed Tomography Image Reconstruction Experiment

In the first experiment, we have demonstrated the superior performance of the proposed Acc-PD-SGD algorithm compared to the deterministic algorithm FISTA, and the state-of-the-art stochastic primal-dual gradient method SPDTCM on a space-varying deblurring problem, although it is not inherently favorable for the application of stochastic gradient methods. In this subsection, we turn to another imaging inverse problem – the computed tomography image reconstruction. As suggested by the curve of the SA factor, the X-ray CT image reconstruction is a nice application for stochastic gradient methods, where we expect them to achieve significant speed-ups over the deterministic methods.

In this experiment we consider a 2D fan-beam CT imaging problem, where we aim to reconstruct a $256 \times 256$ head image from $92532$ noisy X-ray measurements (hence the forward operator $A \in \mathbb{R}^{92532\times 65536}$), using TV-regularization. Denoting $x^\dagger$ to be the (vectorized) ground truth image and $w \in \mathbb{R}^n$ to be an additional random noise vector drawn from an exponential Poisson distribution, we have the observed measurement as $b = Ax^\dagger + w$. The signal-to-noise ratio of the X-ray measurement in this example is set to be: $\log_{10} \frac{\|Ax^\dagger\|_2^2}{\|w\|_2^2} \approx 3.16$. In Figure

[3]For the computation of the TV proximal-operator for FISTA algorithm, we use the popular implementation from the UnLocBox toolbox [175] which is available online (https://epfl-lts2.github.io/unlocbox-html/).

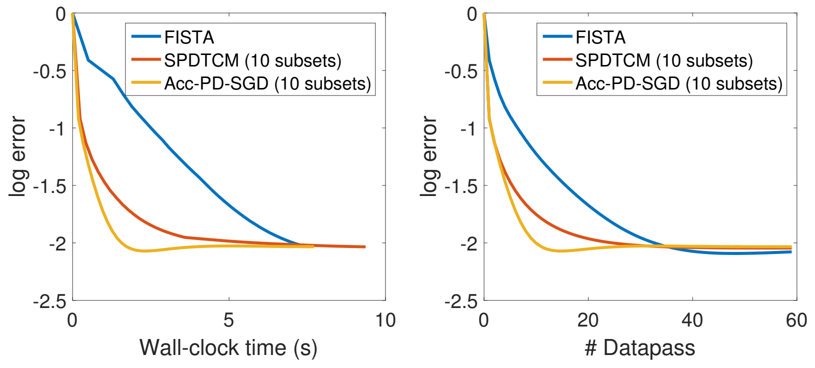


**Figure 6.4:** *The estimation error plot for the X-ray CT image reconstruction experiment with TV-regularization.*

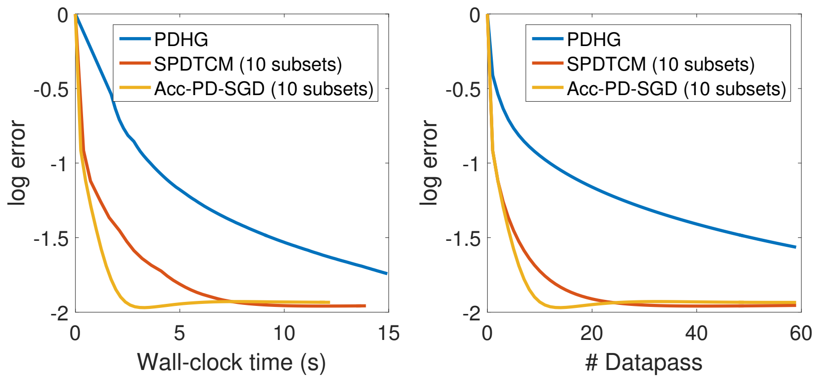


**Figure 6.5:** *The estimation error plot for the X-ray CT image reconstruction experiment with TV-regularization and $\ell_1$ regularization on Haar-wavelet basis.*

6.6, we first demonstrate the reconstructed image by the classic filtered-backprojection (FBP) [176] which is a direct method without considering regularization. From the result of FBP, we can clearly see that the reconstructed image contains a large amount of noise.

From the convergence results of the iterative algorithms in Figure 6.4 we can observe that for this experiment, the stochastic methods SPDTCM and Acc-PD-SGD converges significantly faster than the full gradient method FISTA both in terms of number of datapasses and wall-clock time. Meanwhile, we can also see that, our proposed method Acc-PD-SGD converges faster than the SPDTCM which does not use Katyusha-X momentum for acceleration.

Moreover, in some scenarios (such as the cases where we use low-dose X-ray measurements), we may wish to use more than just one regularizer for a better modelling of the ground truth, in order to ensure an accuracy estimation via additional prior information. In the following experiment, we reduce a half of the dosage of the X-ray measurement, such that $\log_{10} \frac{\|Ax^\dagger\|_2^2}{\|w\|_2^2} \approx 2.86$,

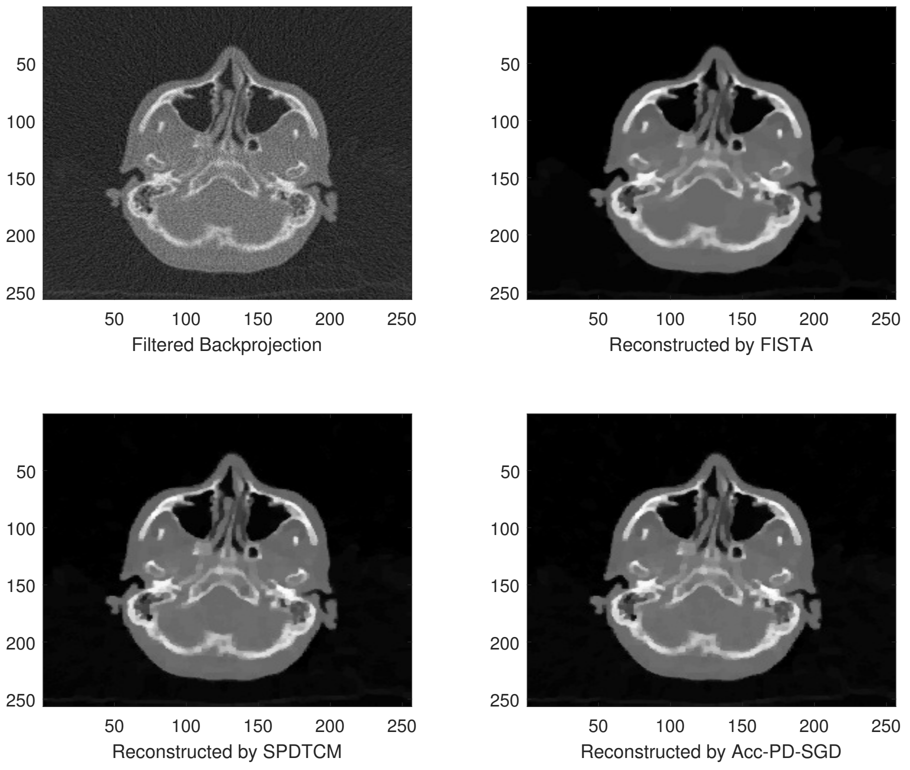


**Figure 6.6:** *The reconstructed images by the compared algorithms with TV-regularization.*

and use two regularization terms jointly for the reconstruction task – the TV regularization and $\ell_1$ regularization on the Haar-wavelet basis. The FISTA algorithm cannot be directly applied for this three-composite optimization task, hence we choose the Chambolle-Pock (PDHG) algorithm as a baseline representing the full gradient methods, and SPDTCM as the representative baseline for the state-of-the-art stochastic gradient methods for the three-composite problems. We present the results of this experiment in Figure 6.5 and 6.7, where we can again clearly observe the superior performance of the proposed method over the baselines on this experiment where we use multiple non-smooth regularization terms.

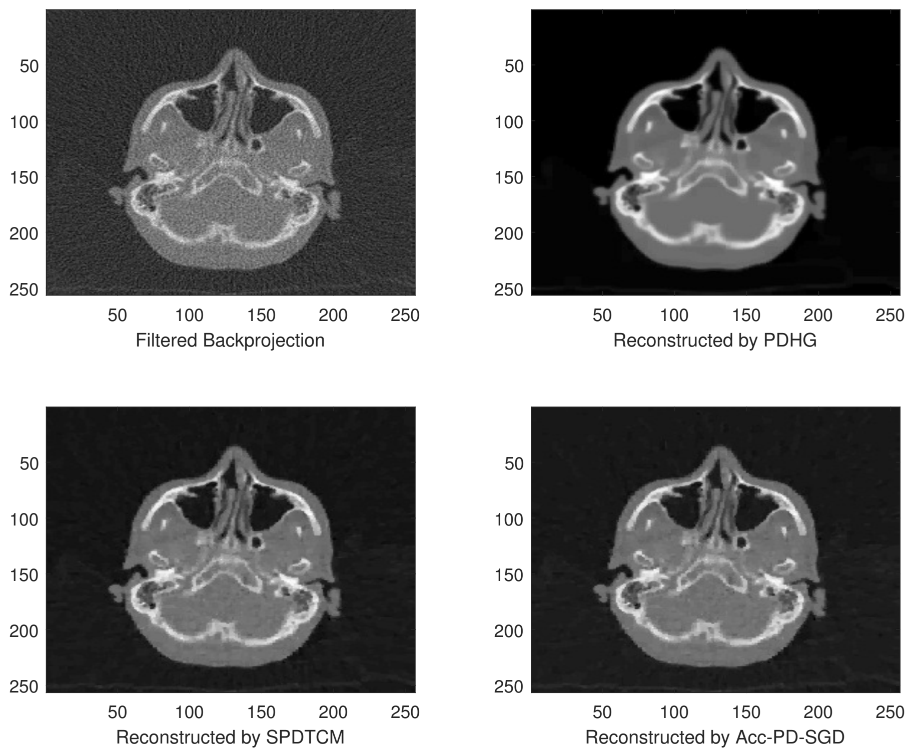


**Figure 6.7:** *The reconstructed images by the compared algorithms at termination using joint TV-$\ell_1$ regularization.*

## 6.6 Concluding Remarks

In this work we investigated the value of the state-of-the-art stochastic gradient methods in imaging inverse problems where we chose image deblurring as a running example. We firstly reveal a surprisingly negative result on existing SGD-type methods, and propose a heuristic metric (SA) to explain such failures and evaluate the possible computational advantage of using stochastic techniques for a given task; finally we combine several practical ideas and propose the Accelerated Primal-Dual SGD to cope with multiple regularizers (potentially) with a linear operator while maintaining the fast convergence, and demonstrate its effectiveness via experiments on space-varying deblurring and X-Ray CT image reconstruction.

Although the work presented in this chapter is mainly empirical and preliminary, and we have not yet done the convergence analysis of the proposed Acc-PD-SGD algorithm, we believe that it provides important insights for the algorithmic design of fast stochastic gradient methods tailored specifically for imaging inverse problems, from understanding the inherent limitation, to the practical algorithmic framework. The future work of this chapter will also be focused on enhancing these two aspects:

Firstly, we will endeavor to better understand the distinguishing features of the inverse problems which have good SA factors, and also the deeper reason for the bad SA factors for some inverse problems. Very recently, we have observed empirically that the inverse problem which has bad SA factors often has a Hessian $A^T A$ which is closer to a diagonal matrix compared to the ones with good SA factors. We have not yet formally analyzed this effect but leave it as a good open direction for further research.

Secondly, in terms of the theoretical analysis and the structure-adaptive algorithmic improvement of Acc-PD-SGD, we wish to apply the RSC framework [5] for analyzing the convergence of this algorithm, and deriving the optimal structure-adaptive variants of it using the restart schemes as we have done in chapters 4 and 5. The analysis of Acc-PD-SGD under the RSC framework will be highly non-trivial and of great interest, since there is no structure-adaptive convergence results using the RSC condition so far for the class of primal-dual algorithms which are often very practical and flexible in imaging inverse problems.

## 6.A Appendix

### 6.A.1 The Proof for Theorem 6.3.3

*Proof.* The proof of this theorem is straight forward and is based on combining the existing results since we have assumed that $f$ satisfies simultaneously the lower bounds in Theorem 2.3.4 and 2.4.1 and the dimension $d$ is large enough for both of the lower-bounds to hold on a domain:

$$\mathcal{X} = \left\{ x \in \mathbb{R}^d : \|x\|_2^2 \leq 1 \right\}. \tag{6.15}$$

By the convergence-rate lower bound for full gradient we have presented in Theorem 2.3.4, we can have:

$$f(x^s_{\mathcal{A}_{\text{full}}}) - f^\star \geq \Omega\left(\frac{L_f\|x^0 - x^\star\|_2^2}{(s+1)^2}\right) := \frac{C_{\text{full}} L_f \|x^0 - x^\star\|_2^2}{s^2}. \tag{6.16}$$

where we denote $C_{\text{full}}$ being a positive constant. Meanwhile, for optimal stochastic algorithm we have the upper bound of convergence by Def. 6.3.2 with setting $m = K$:

$$\mathbb{E} f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star \quad \leq \quad \frac{C_2(f(x^0) - f^\star) + \frac{C_3 L_b}{K}\|x^0 - x^\star\|_2^2}{s^2} \tag{6.17}$$

Combining the two bounds we can have:

$$\frac{\mathbb{E} f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star}{f(x^s_{\mathcal{A}_{\text{full}}}) - f^\star} \leq \frac{C_2}{C_{\text{full}}} \cdot \frac{L_b}{K L_f} + \frac{C_3(f(x^0) - f^\star)}{C_{\text{full}} L_f \|x^0 - x^\star\|_2^2}. \tag{6.18}$$

Recall the definition of smoothness, we can have:

$$f(x^0) - f^\star - \langle \nabla f(x^\star), x^0 - x^\star \rangle \leq \frac{L_f}{2}\|x^0 - x^\star\|_2^2, \tag{6.19}$$

and using the assumption that any solution $x^\star$ lives in the relative interior of $\mathcal{X}$, we can have $\nabla f(x^\star) = 0$, and hence $f(x^0) - f^\star \leq \frac{L_f}{2}\|x^0 - x^\star\|_2^2$. Consequently, we can have:

$$\frac{\mathbb{E} f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star}{f(x^s_{\mathcal{A}_{\text{full}}}) - f^\star} \leq \frac{C_2}{C_{\text{full}}} \cdot \frac{L_b}{K L_f} + \frac{C_3}{2C_{\text{full}}}. \tag{6.20}$$

Similarly, according to the lower bound for the stochastic gradient we have presented in Theorem 2.4.1 [54, Theorem 7], there exist a positive constant $C_{\text{stoc}}$, such that in order to achieve an output $\mathbb{E}f(x^s_{\mathcal{A}}) - f^\star \leq \epsilon$, any stochastic gradient algorithm must take at least:

$$C_{\text{stoc}}\left(K + \sqrt{\frac{KL_b}{\epsilon}}\right) \tag{6.21}$$

calls of the stochastic gradient oracle $\triangledown f_i()$. In other words, for this worst case function, if we run any stochastic gradient method with only $Ks$ calls on the stochastic gradient oracle such that:

$$Ks = C_{\text{stoc}}\sqrt{\frac{KL_b}{\epsilon}}, \tag{6.22}$$

$\mathbb{E}f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star \geq \epsilon$ can be guaranteed. Hence, we have:

$$\mathbb{E}f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star \geq \frac{C^2_{\text{stoc}}L_b}{Ks^2} \tag{6.23}$$

Meanwhile, starting from $x^0 \in \mathcal{X}$, by Def. 6.3.1, for any optimal full gradient method $\mathcal{A}_{\text{full}}$ we can have:

$$f(x^s_{\mathcal{A}_{\text{full}}}) - f^\star \leq \frac{C_1 L_f \|x^0 - x^\star\|_2^2}{s^2} \leq \frac{4C_1L_f}{s^2}. \tag{6.24}$$

Combining these two bounds we can have:

$$\frac{\mathbb{E}f(x^s_{\mathcal{A}_{\text{stoc}}}) - f^\star}{f(x^s_{\mathcal{A}_{\text{full}}}) - f^\star} \geq \frac{C^2_{\text{stoc}}L_b}{4C_1KL_f}. \tag{6.25}$$

Finally, by setting $c_0 = \frac{C^2_{\text{stoc}}}{4C_1}$, $c_1 = \frac{C_2}{C_{\text{full}}}$ and $c_2 = \frac{C_3}{2C_{\text{full}}}$ we yield the claim. $\square$

# Chapter 7
# Conclusion and Future Perspectives

## 7.1 Summary

In this thesis we have advanced the state-of-the-art of large-scale optimization by developing iterative randomized algorithms which are able to actively exploit the solution's low-dimensional structure enforced by regularization within the optimization task.

In chapter 3 we focus on the sketched gradient algorithms, which essentially combine the sketching meta-algorithms and projected gradient descent, for efficiently solving constrained least-squares in the big-data regime $n \gg d \gg O(1)$. Our convergence analysis shows that the sketched gradient algorithms enjoy a fast convergence rate which scales desirably with the intrinsic dimension of the solution. In practice, by implementing the efficient sparse random projection, the tailored line search scheme for aggressive step-sizes, and the adaptive restart scheme, we found that our methods significantly outperforms the state-of-the-art stochastic variance-reduced gradient methods at scenarios where large minibatch-sizes are used (which is very common in modern machine learning practice), in large-scale constrained least-squares regression and low-rank multivariate regression tasks.

The main intuition behind the sketched gradient algorithm is that, the randomized projection preserve the statistical dimension of the optimization problem while massively reduce the computation, and meanwhile the projected gradient descent provides the scalability to the high-dimensions and efficiency in handling the constraints. Moreover, it has been shown that non-accelerated deterministic first-order solvers themselves such as projected/proximal gradient descent achieves automatic adaptiveness towards the intrinsic-dimension of the solution without any further algorithmic enhancement, under a restricted strong-convexity (RSC) framework which links the solution's intrinsic dimension with the convergence speed of a gradient-based algorithm [5, 127]. Under the same RSC framework of [5], researchers [118, 119] derived the structure-adaptive convergence analysis of non-accelerated stochastic gradient methods SVRG and SAGA. However, for first-order methods with Nesterov-type acceleration scheme, which typically enjoy state-of-the-art and worst-case optimal convergence results, such structure-adaptiveness cannot be directly achieved, but requires some more tailored modifications.

In our chapters 4 and 5, we aimed at studying how to maximize the potential structure-exploiting capability of the stochastic first-order methods. We focus on developing structure-adaptive variants of accelerated stochastic first-order algorithms – the accelerated stochastic gradient descent with variance-reduction [4], as well as the accelerated block-coordinate descent [6]. This provable algorithmic improvement is done by adaptively restarting these algorithms according to the RSC parameter. We demonstrate both theoretically and experimentally that, the proposed schemes achieve significant speed up over the vanilla accelerated methods on regularized empirical risk minimization tasks with solutions which have intrinsic low-dimensional structure.

While being the de-facto techniques in machine learning practice, the randomized first-order methods have not been very widely applied in signal and image processing applications (excepting a few applications such as tomographic image reconstruction). Hence in chapter 6, we demonstrate some preliminary but insightful empirical ideas on the practicability of stochastic gradient methods for imaging inverse problems such as image deblurring.

There are several issues hindering the application of stochastic methods in imaging inverse problems: (1) some inverse problems are indeed inherently not suitable for SGD-type methods – for example, the space-varying deblurring task we have discussed in Section 6.2 – in such problems we cannot expect stochastic methods to be significantly better than deterministic gradient methods, such as image deblurring; (2) the regularization term with a linear operator (such as TV-regularization) requires a proximal operator which is non-trivial to compute and the stochastic methods typically need many more frequent calls on the proximal operator than the deterministic methods, (3) multiple terms of regularization are desired for some scenarios. In chapter 6, we provide a numerical scheme to characterize whether an imaging inverse problem is inherently suitable for the application of stochastic gradient methods. For overcoming the issues (2) and (3) of complicated regularization terms, we propose an efficient accelerated primal-dual SGD method and demonstrate its effectiveness in space-varying image deblurring and tomographic imaging. Although seemly being a side-contribution to the main theme of the thesis (structure-adaptive optimization) as compared to chapter 3 - 5, we believe that, the empirical work we presented in chapter 6 provide us important insights and practical algorithmic framework for a crucial line of future research – the design of structure-adaptive algorithms tailored specifically for the imaging applications.

## 7.2 Future Directions and On-Going Research

The technical results of this thesis suggest many interesting and promising future directions. We list a few critical and immediate ones here:

### 7.2.1 Proximal and Primal-Dual Sketched Gradient Methods.

Inspired by the primal-dual saddle-point algorithmic framework and the potential computational benefits which variable metric schemes can bring forth in practice, we wish to extend the algorithms we developed in [7] (which directly solve the primal optimization problem, and restrict to only the constrained least-squares) to the primal-dual and variable metric setting. Since via the primal-dual framework we can presumably perform sketching on both 2 dimensions of the data matrix, we expect to have a sketching based first-order method which is able to : (1) solve the generic regularized empirical risk minimization task (2.3), and being scalable to graph-guided regularization (such as TV) regularization and multiple regularization terms; (2) being efficient when the data matrix is nearly square or even highly under-determined.

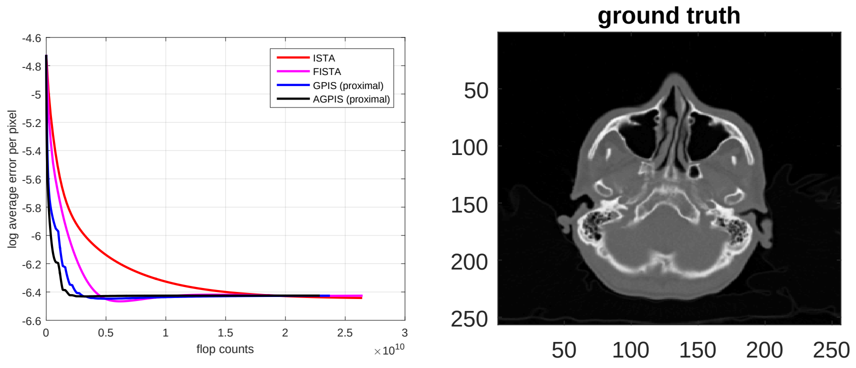


**Figure 7.1:** *Some preliminary results of applying Proximal-GPIS and Proximal-Acc-GPIS on a fan-beam computed tomography (CT) image reconstruction experiments (Total-Variation Regularized least-squares)*

In terms of theory, we could apply the RSC framework [5] to analyze the structure-adaptive convergence of the proximal sketched gradient methods. Moreover, under the same theoretical framework, we will be able to effectively compare the structure-adaptiveness of the two distinct type of randomized algorithms we have considered in this thesis (the sketched gradient and stochastic gradient methods), in a unified and systematic manner. Hence, future research in this direction may also provide us with practical guidance as to how to choose between the sketched gradient algorithms or the stochastic gradient algorithms for different scenarios.

### 7.2.2 Universal Structure-Adaptive Restart Scheme.

The idea of using restart schemes to exploit the RSC which we have successfully applied in chapters 3 and 4 [9, 11] can be extended to any stochastic or deterministic gradient methods with Nesterov-type acceleration. We wish to provide a unified universal structure-adaptive analysis of restarted first-order methods under the RSC framework, viewing the restart scheme as a meta-algorithm. Moreover we wish to develop a practically and provably good adaptive restart scheme which does not need to know (or accurately estimate) the RSC parameter beforehand.

### 7.2.3 The Local Convergence and Acceleration of Rest-Katyusha and Two-Stage APCG

As we have mentioned in the background chapter, there is another framework [44, 121, 122] which is complementary to the RSC, and can also be applied for local convergence analysis of first-order algorithms, using finite activity-identification of gradient-based methods and local partial-smoothness structure of the composite optimization problems. Recent breakthroughs [123] show that the stochastic gradient methods with variance-reduction have the finite activity-identification property and achieve local linear convergence, hence opening the door for the local sharp structure-adaptive convergence analysis of stochastic first-order methods. We believe that, if we further analyze the proposed Rest-Katyusha and two-stage APCG algorithms under this local convergence framework, we will not only be able to enhance these algorithms' convergence guarantees up to an arbitrary accuracy, but also, derive even faster variants, which utilize greedier step-sizes using local smoothness of the composite optimization tasks.

### 7.2.4 Structure-Adaptive Stochastic Approximation for Online Learning

The current work presented in this thesis focuses on the off-line settings where the data samples are obtained beforehand. In many signal processing and machine learning applications, the data samples arrive on the fly, and such online settings can not be described as a finite-sum but as a stochastic programming problem. Extending and developing the randomized structure-adaptive gradient methods in the online setting will also be of great interest. One immediate example of applications would be the sparse adaptive-filtering, where we wish to learn adaptively a tracking filter in order to denoise audio or video sequences in real time, with a random noise having a time-varying statistics [177].

### 7.2.5 Stochastic Structure-Adaptive Methods for Non-Convex Optimization

One important future direction of this thesis could be the extension towards the non-convex optimization problems, such as the training of deep neural networks. In view of the success of structure-exploiting algorithms in the convex optimization tasks, we have enough reason to believe that the low-dimensional structure is a hidden treasure waiting to be discovered and exploited for faster local convergence speed in non-convex optimization.

### 7.2.6 Acc-PD-SGD with Lazy Denoising Schemes for Fast and Accurate Image Processing

In chapter 6 we have presented a preliminary result on applying an accelerated primal-dual SGD method to a space-varying deblurring task, using a TV regularization. In our on-going work we aim at extending Acc-PD-SGD to a more general plug-and-play form, which is not only able to use classic priors efficiently such as the TV regularization, but can also cooperate with the state-of-the-art denoisers such as BM3D [178, 179], TNRD[180], NLM [181] and DnCNN [182] to achieve better estimation accuracy in imaging inverse problems. Directly applying the expensive denoisers with stochastic gradient will lead to slow convergence in time due to the multiple number of calls on the denoiser, and hence we propose a lazy update scheme (inspired by recently introduced gradient-sliding schemes [183, 184, 185, 186]), which only call the denoiser once at the beginning of each epoch and record such a denoising direction. Then in each innerloop iteration, instead of calling the denoiser again, we use this pre-recorded sliding direction as the denoising step.

We present some of our very recent experimental results here, where we consider space-varying deblurring tasks where the blurred image is corrupted by a large amount of Gaussian noise. In the high-noise regime, using TV regularization alone is not enough to provide us a good estimation of the ground truth image, so we apply our algorithm not-only with TV regularization, but also jointly with the BM3D denoiser which enforces an implicit regularization. The experimental setting is the same as the numerical experiment section in chapter 6, but this time we make the noise variance 5 times larger. These preliminary results demonstrate the practicability and efficiency of this generalized plug-and-play version of Acc-PD-SGD. In our on-going work, we also aim to formally analyze the convergence of Acc-PD-SGD as well as this plug-and-play variant. Moreover, we believe that, from a detailed study of the structure-adaptive convergence theory for Acc-PD-SGD, using the RSC framework [5, 116] we have considered throughout this thesis, we can derive the optimal step-size strategy, the tailored adaptive-restart scheme and also the optimal lazy-denoising scheme for the plug-and-play Acc-PD-SGD algorithm.

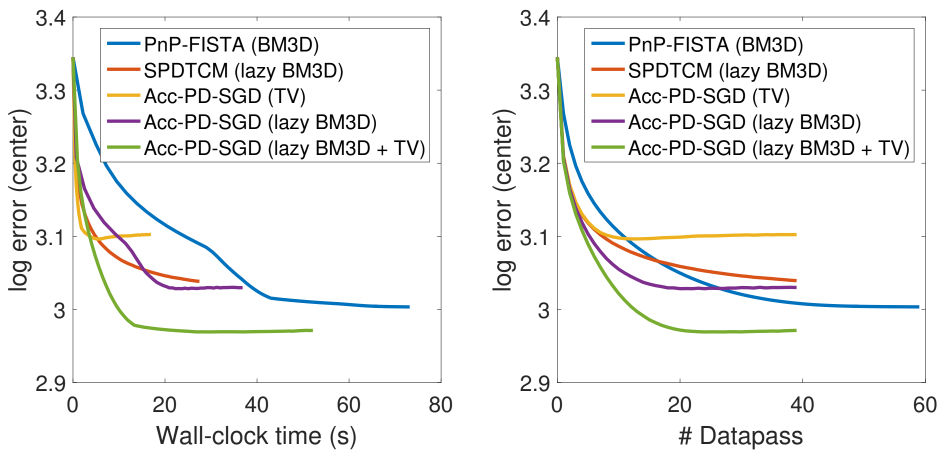


**Figure 7.2:** *Convergence results for the compared algorithms for deblurring Kodim05 image*

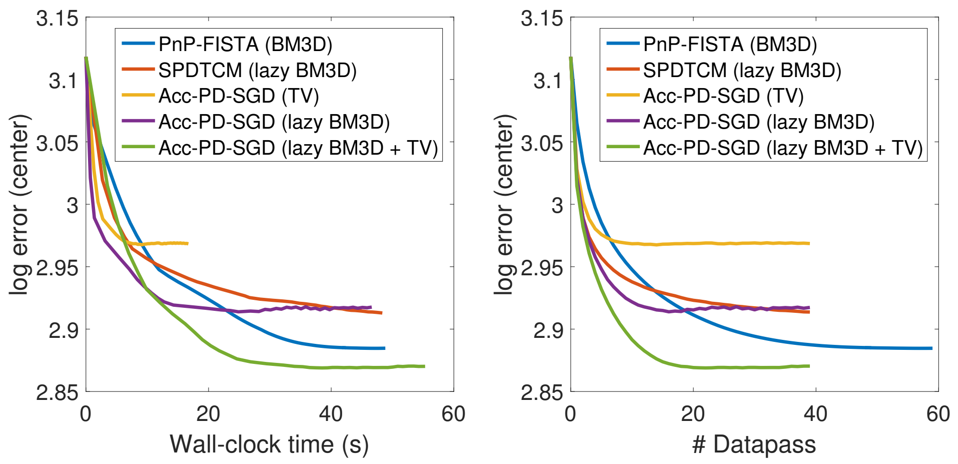


**Figure 7.3:** *Convergence results for the compared algorithms for deblurring Kodim06 image*

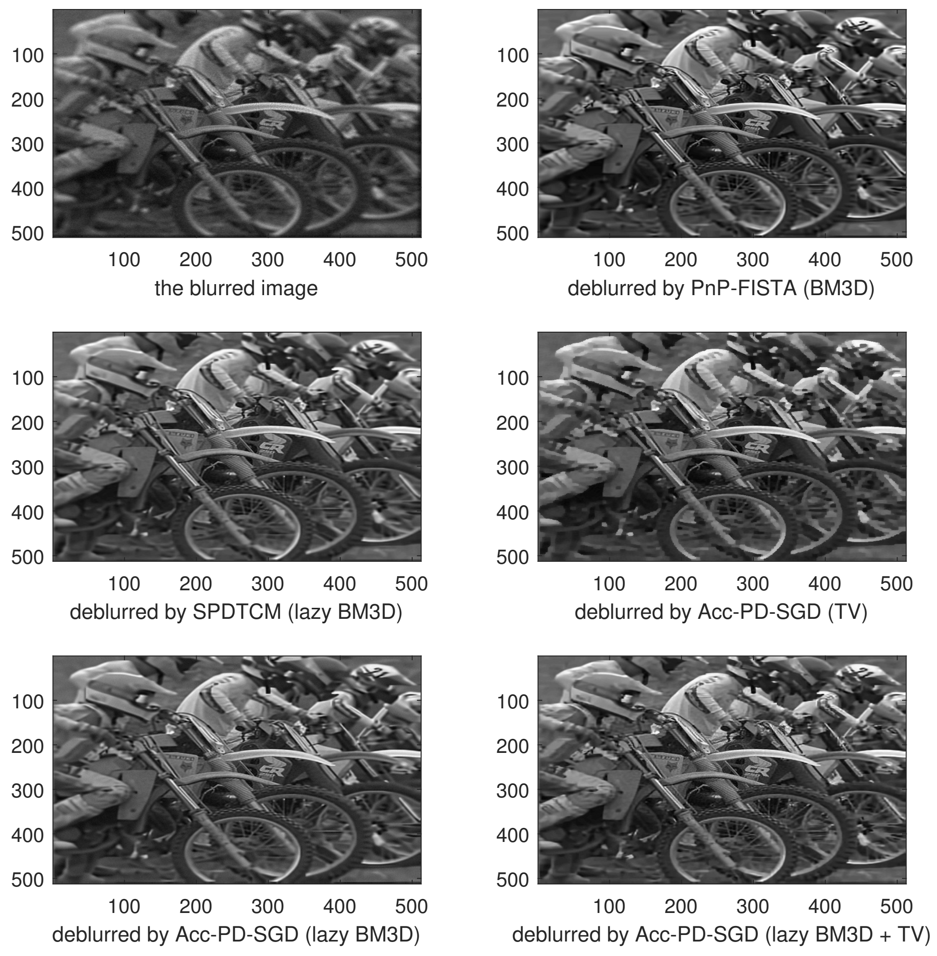

100
200
300
400
500
100 200 300 400 500
the blurred image
deblurred by PnP-FISTA (BM3D)
deblurred by SPDTCM (lazy BM3D)
deblurred by Acc-PD-SGD (TV)
deblurred by Acc-PD-SGD (lazy BM3D)
deblurred by Acc-PD-SGD (lazy BM3D + TV)

**Figure 7.4:** *Deblurred images for Kodim05*

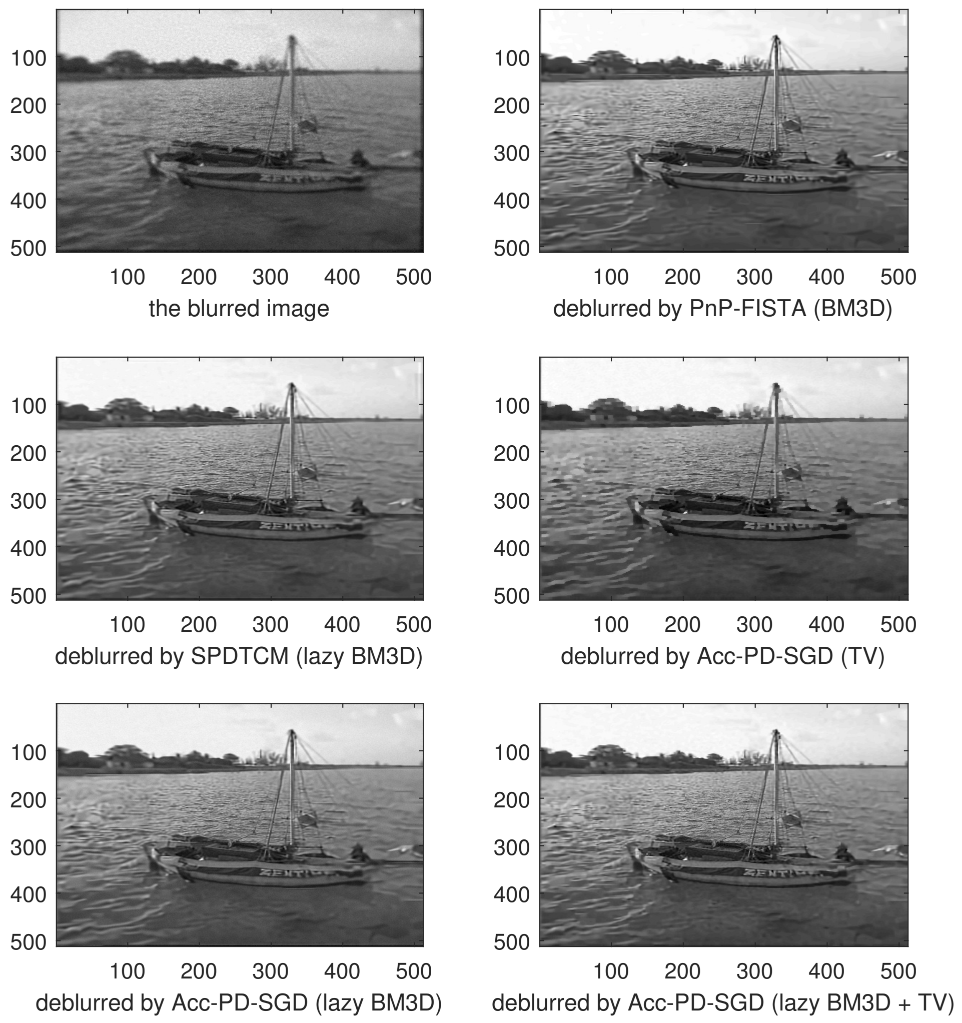

the blurred image
deblurred by PnP-FISTA (BM3D)
deblurred by SPDTCM (lazy BM3D)
deblurred by Acc-PD-SGD (TV)
deblurred by Acc-PD-SGD (lazy BM3D)
deblurred by Acc-PD-SGD (lazy BM3D + TV)

**Figure 7.5:** *Deblurred images for Kodim06*

# References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[2] M. Pilanci and M. J. Wainwright, “Randomized sketches of convex programs with sharp guarantees,” *Information Theory, IEEE Transactions on*, vol. 61, no. 9, pp. 5096–5115, 2015.

[3] M. Pilanci and M. J. Wainwright, “Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares,” *Journal of Machine Learning Research*, vol. 17, no. 53, pp. 1–38, 2016.

[4] Z. Allen-Zhu, “Katyusha: The first direct acceleration of stochastic gradient methods,” in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1200–1205, ACM, 2017.

[5] A. Agarwal, S. Negahban, and M. J. Wainwright, “Fast global convergence rates of gradient methods for high-dimensional statistical recovery,” *The Annals of Statistics*, vol. 40, no. 5, pp. 2452–2482, 2012.

[6] Q. Lin, Z. Lu, and L. Xiao, “An accelerated proximal coordinate gradient method,” in *Advances in Neural Information Processing Systems*, pp. 3059–3067, 2014.

[7] J. Tang, M. Golbabaee, and M. E. Davies, “Gradient projection iterative sketch for large-scale constrained least-squares,” in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 3377–3386, PMLR, 06–11 Aug 2017.

[8] J. Tang, M. Golbabaee, and M. Davies, “Exploiting the structure via sketched gradient algorithms,” in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1305–1309, IEEE, 2017.

[9] J. Tang, M. Golbabaee, F. Bach, and M. E. davies, “Rest-katyusha: Exploiting the solutions structure via scheduled restart schemes,” in *Advances in Neural Information Processing Systems 31*, pp. 427–438, Curran Associates, Inc., 2018.

[10] J. Tang, K. Egiazarian, and M. Davies, “The limitation and practical acceleration of stochastic gradient algorithms in imaging inverse problems,” *to appear in ICASSP*, 2019.

[11] J. Tang, M. Golbabaee, F. Bach, and M. Davies, “Structure-adaptive accelerated coordinate descent,” *Hal-archive hal-01889990v2*, 2018.

[12] J. Tang, F. Bach, M. Golbabaee, and M. Davies, “Structure-adaptive, variance-reduced, and accelerated stochastic optimization,” *arXiv preprint arXiv:1712.03156*, 2017.

[13] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[14] M. J. Wainwright, “Structured regularizers for high-dimensional problems: Statistical and computational issues,” *Annual Review of Statistics and Its Application*, vol. 1, pp. 233–253, 2014.

[15] P. Bühlmann and S. Van De Geer, *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[16] R. Tibshirani, M. Wainwright, and T. Hastie, *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

[17] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[18] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, “Sparsity and smoothness via the fused lasso,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[19] M. Yuan and Y. Lin, “Model selection and estimation in regression with grouped variables,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[20] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, vol. 398. John Wiley & Sons, 2013.

[21] Y. Amit, M. Fink, N. Srebro, and S. Ullman, “Uncovering shared structures in multiclass classification,” in *Proceedings of the 24th international conference on Machine learning*, pp. 17–24, ACM, 2007.

[22] A. Argyriou, T. Evgeniou, and M. Pontil, “Convex multi-task feature learning,” *Machine Learning*, vol. 73, no. 3, pp. 243–272, 2008.

[23] J. Zhu, S. Rosset, R. Tibshirani, and T. J. Hastie, “1-norm support vector machines,” in *Advances in neural information processing systems*, pp. 49–56, 2004.

[24] J. Friedman, T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, “Discussion of consistency in boosting,” *Annals of Statistics*, vol. 32, no. 1, pp. 102–107, 2004.

[25] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[26] K. Bredies, K. Kunisch, and T. Pock, “Total generalized variation,” *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 492–526, 2010.

[27] A. Chambolle and T. Pock, “An introduction to continuous optimization for imaging,” *Acta Numerica*, vol. 25, pp. 161–319, 2016.

[28] S. Bubeck, “Convex optimization: Algorithms and complexity,” *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[29] H. H. Bauschke, P. L. Combettes, *et al.*, *Convex analysis and monotone operator theory in Hilbert spaces*, vol. 408. Springer, 2011.

[30] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, pp. 2004–2005, 2003.

[31] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87. Springer Science & Business Media, 2013.

[32] 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*, vol. 1, pp. 586–597, Dec 2007.

[33] J. J. Moreau, "Fonctions convexes duales et points proximaux dans un espace hilbertien," *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, vol. 255, pp. 2897–2899, 1961.

[34] P.-L. Lions and B. Mercier, "Splitting algorithms for the sum of two nonlinear operators," *SIAM Journal on Numerical Analysis*, vol. 16, no. 6, pp. 964–979, 1979.

[35] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[36] J. D. Lee, Y. Sun, and M. A. Saunders, "Proximal newton-type methods for minimizing composite functions," *SIAM Journal on Optimization*, vol. 24, no. 3, pp. 1420–1443, 2014.

[37] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[38] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical programming*, vol. 103, no. 1, pp. 127–152, 2005.

[39] Y. Nesterov, "Gradient methods for minimizing composite objective function," tech. rep., UCL, 2007.

[40] O. Fercoq and Z. Qu, "Restarting accelerated gradient methods with a rough strong convexity estimate," *arXiv preprint arXiv:1609.07358*, 2016.

[41] J. Liang and C.-B. Schönlieb, "Improving fista: Faster, smarter and greedier," *arXiv preprint arXiv:1811.01430*, 2018.

[42] B. O'Donoghue and E. Candes, "Adaptive restart for accelerated gradient schemes," *Foundations of computational mathematics*, vol. 15, no. 3, pp. 715–732, 2015.

[43] O. Fercoq and Z. Qu, "Adaptive restart of accelerated gradient methods under local quadratic growth condition," *arXiv preprint arXiv:1709.02300*, 2017.

[44] J. Liang, J. Fadili, and G. Peyré, "Local convergence properties of douglas–rachford and alternating direction method of multipliers," *Journal of Optimization Theory and Applications*, vol. 172, no. 3, pp. 874–913, 2017.

[45] H. Robbins and S. Monro, “A stochastic approximation method,” *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[46] T. Zhang, “Solving large scale linear prediction problems using stochastic gradient descent algorithms,” in *Proceedings of the twenty-first international conference on Machine learning*, p. 116, ACM, 2004.

[47] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*, pp. 177–186, Springer, 2010.

[48] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: Primal estimated sub-gradient solver for svm,” *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[49] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[50] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[51] S. J. Reddi, S. Kale, and S. Kumar, “On the convergence of adam and beyond,” 2018.

[52] L. Luo, Y. Xiong, Y. Liu, and X. Sun, “Adaptive gradient methods with dynamic bound of learning rate,” *arXiv preprint arXiv:1902.09843*, 2019.

[53] G. Lan, “An optimal method for stochastic composite optimization,” *Mathematical Programming*, vol. 133, no. 1-2, pp. 365–397, 2012.

[54] B. E. Woodworth and N. Srebro, “Tight complexity bounds for optimizing composite objectives,” in *Advances in neural information processing systems*, pp. 3639–3647, 2016.

[55] G. Lan and Y. Zhou, “An optimal randomized incremental gradient method,” *arXiv preprint arXiv:1507.02000*, 2015.

[56] N. L. Roux, M. Schmidt, and F. R. Bach, “A stochastic gradient method with an exponential convergence _rate for finite training sets,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 2663–2671, Curran Associates, Inc., 2012.

[57] M. Schmidt, N. Le Roux, and F. Bach, “Minimizing finite sums with the stochastic average gradient,” *Mathematical Programming*, pp. 1–30, 2013.

[58] R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in *Advances in neural information processing systems*, pp. 315–323, 2013.

[59] M. Mahdavi, L. Zhang, and R. Jin, “Mixed optimization for smooth functions,” in *Advances in Neural Information Processing Systems*, pp. 674–682, 2013.

[60] L. Xiao and T. Zhang, “A proximal stochastic gradient method with progressive variance reduction,” *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.

[61] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 567–599, 2013.

[62] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

[63] S. Shalev-Shwartz and T. Zhang, "Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization," in *International Conference on Machine Learning*, pp. 64–72, 2014.

[64] Q. Lin, Z. Lu, and L. Xiao, "An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2244–2273, 2015.

[65] Y. Zhang and X. Lin, "Stochastic primal-dual coordinate method for regularized empirical risk minimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 353–361, 2015.

[66] K. Zhou, F. Shang, and J. Cheng, "A simple stochastic variance reduced algorithm with fast convergence rates," in *International Conference on Machine Learning*, pp. 5975–5984, 2018.

[67] A. Defazio, "A simple practical accelerated method for finite sums," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 676–684, Curran Associates, Inc., 2016.

[68] K. Zhou, "Direct acceleration of saga using sampled negative momentum," *arXiv preprint arXiv:1806.11048*, 2018.

[69] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[70] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Mathematical Programming*, vol. 144, no. 1-2, pp. 1–38, 2014.

[71] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[72] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pp. 147–156, IEEE, 2013.

[73] O. Fercoq and P. Richtárik, "Accelerated, parallel, and proximal coordinate descent," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 1997–2023, 2015.

[74] Z. Allen-Zhu, Z. Qu, P. Richtárik, and Y. Yuan, "Even faster accelerated coordinate descent using non-uniform sampling," in *International Conference on Machine Learning*, pp. 1110–1119, 2016.

[75] M. Nikolova, “A variational approach to remove outliers and impulse noise,” *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 99–120, 2004.

[76] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” *Journal of mathematical imaging and vision*, vol. 40, no. 1, pp. 120–145, 2011.

[77] A. Chambolle, M. J. Ehrhardt, P. Richtárik, and C.-B. Schonlieb, “Stochastic primal-dual hybrid gradient algorithm with arbitrary sampling and imaging applications,” *SIAM Journal on Optimization*, vol. 28, no. 4, pp. 2783–2808, 2018.

[78] R. Zhao and V. Cevher, “Stochastic three-composite convex minimization with a linear operator,” in *International Conference on Artificial Intelligence and Statistics*, pp. 765–774, 2018.

[79] A. Chambolle and T. Pock, “On the ergodic convergence rates of a first-order primal–dual algorithm,” *Mathematical Programming*, vol. 159, no. 1-2, pp. 253–287, 2016.

[80] R. T. Rockafellar, *Convex analysis*. Princeton university press, 2015.

[81] J.-J. Moreau, “Proximité et dualité dans un espace hilbertien,” *Bulletin de la Société mathématique de France*, vol. 93, pp. 273–299, 1965.

[82] J.-C. Pesquet and A. Repetti, “A class of randomized primal-dual algorithms for distributed optimization,” *arXiv preprint arXiv:1406.6404*, 2014.

[83] P. L. Combettes and J.-C. Pesquet, “Stochastic forward-backward and primal-dual approximation algorithms with application to online image restoration,” in *Signal Processing Conference (EUSIPCO), 2016 24th European*, pp. 1813–1817, IEEE, 2016.

[84] B. Palaniappan and F. Bach, “Stochastic variance reduction methods for saddle-point problems,” in *Advances in Neural Information Processing Systems*, pp. 1416–1424, 2016.

[85] S. S. Du and W. Hu, “Linear convergence of the primal-dual gradient method for convex-concave saddle point problems without strong convexity,” *arXiv preprint arXiv:1802.01504*, 2018.

[86] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[87] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo, “Fast image recovery using variable splitting and constrained optimization,” *IEEE transactions on image processing*, vol. 19, no. 9, pp. 2345–2356, 2010.

[88] K. L. Clarkson and D. P. Woodruff, “Low rank approximation and regression in input sparsity time,” in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 81–90, ACM, 2013.

[89] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, “Dimensionality reduction for k-means clustering and low rank approximation,” in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 163–172, ACM, 2015.

[90] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, “Practical sketching algorithms for low-rank matrix approximation,” *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1454–1485, 2017.

[91] D. P. Woodruff *et al.*, “Sketching as a tool for numerical linear algebra,” *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[92] M. W. Mahoney, “Randomized algorithms for matrices and data,” *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[93] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, “Faster least squares approximation,” *Numerische Mathematik*, vol. 117, no. 2, pp. 219–249, 2011.

[94] H. Avron, P. Maymounkov, and S. Toledo, “Blendenpik: Supercharging lapack’s least-squares solver,” *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1217–1236, 2010.

[95] S. S. Vempala, *The random projection method*, vol. 65. American Mathematical Soc., 2005.

[96] N. Ailon and E. Liberty, “Fast dimension reduction using rademacher series ondual bch codes,” *Discrete & Computational Geometry*, vol. 42, no. 4, pp. 615–630, 2008.

[97] N. Ailon and B. Chazelle, “The fast johnsonlindenstrauss transform and approximate nearest neighbors,” *SIAM Journal on Computing*, vol. 39, no. 1, pp. 302–322, 2009.

[98] A. Gonen, F. Orabona, and S. Shalev-Shwartz, “Solving ridge regression using sketched preconditioned svrg,” in *International Conference on Machine Learning*, pp. 1397–1405, 2016.

[99] J. Yang, Y.-L. Chow, C. Ré, and M. W. Mahoney, “Weighted sgd for p regression with randomized preconditioning,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7811–7853, 2017.

[100] M. Pilanci and M. J. Wainwright, “Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence,” *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 205–245, 2017.

[101] A. Rudi, R. Camoriano, and L. Rosasco, “Less is more: Nyström computational regularization,” in *Advances in Neural Information Processing Systems*, pp. 1657–1665, 2015.

[102] L. Carratino, A. Rudi, and L. Rosasco, “Learning with sgd and random features,” in *Advances in Neural Information Processing Systems*, pp. 10212–10223, 2018.

[103] Y. Yang, M. Pilanci, M. J. Wainwright, *et al.*, “Randomized sketches for kernels: Fast and optimal nonparametric regression,” *The Annals of Statistics*, vol. 45, no. 3, pp. 991–1023, 2017.

[104] J. Langford, L. Li, and T. Zhang, “Sparse online learning via truncated gradient,” *Journal of Machine Learning Research*, vol. 10, no. Mar, pp. 777–801, 2009.

[105] D. M. Kane and J. Nelson, “Sparser johnson-lindenstrauss transforms,” *J. ACM*, vol. 61, pp. 4:1–4:23, Jan. 2014.

[106] J. Bourgain, S. Dirksen, and J. Nelson, “Toward a unified theory of sparse dimensionality reduction in euclidean space,” *Geometric and Functional Analysis*, vol. 25, no. 4, pp. 1009–1088, 2015.

[107] M. B. Cohen, T. Jayram, and J. Nelson, “Simple analyses of the sparse johnson-lindenstrauss transform,” in *OASIcs-OpenAccess Series in Informatics*, vol. 61, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[108] Y. Dahiya, D. Konomis, and D. P. Woodruff, “An empirical evaluation of sketching for numerical linear algebra,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1292–1300, ACM, 2018.

[109] H. H. Chin and P. P. Liang, “An empirical evaluation of sketched svd and its application to leverage score ordering,” *arXiv preprint arXiv:1812.07903*, 2018.

[110] M. Kapralov, V. K. Potluru, and D. P. Woodruff, “How to fake multiply by a gaussian matrix,” *arXiv preprint arXiv:1606.05732*, 2016.

[111] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, “The convex geometry of linear inverse problems,” *Foundations of Computational mathematics*, vol. 12, no. 6, pp. 805–849, 2012.

[112] V. Chandrasekaran and M. I. Jordan, “Computational and statistical tradeoffs via convex relaxation,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 13, pp. E1181–E1190, 2013.

[113] F. R. Bach, “Consistency of the group lasso and multiple kernel learning,” *Journal of Machine Learning Research*, vol. 9, no. Jun, pp. 1179–1225, 2008.

[114] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[115] S. Vaiter, M. Golbabaee, J. Fadili, and G. Peyré, “Model selection with low complexity priors,” *Information and Inference: A Journal of the IMA*, vol. 4, no. 3, pp. 230–287, 2015.

[116] S. N. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu, “A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers,” *Statistical Science*, pp. 538–557, 2012.

[117] G. Raskutti, M. J. Wainwright, and B. Yu, “Restricted eigenvalue properties for correlated gaussian designs,” *Journal of Machine Learning Research*, vol. 11, no. Aug, pp. 2241–2259, 2010.

[118] C. Qu and H. Xu, “Linear convergence of svrg in statistical estimation,” *arXiv preprint arXiv:1611.01957*, 2016.

[119] C. Qu, Y. Li, and H. Xu, “Saga and restricted strong convexity,” *arXiv preprint arXiv:1702.05683*, 2017.

[120] C. Qu and H. Xu, “Linear convergence of sdca in statistical estimation,” *arXiv preprint arXiv:1701.07808*, 2017.

[121] J. Liang, J. Fadili, and G. Peyré, "Local linear convergence of forward–backward under partial smoothness," in *Advances in Neural Information Processing Systems*, pp. 1970–1978, 2014.

[122] J. Liang, J. Fadili, and G. Peyré, "Local linear convergence analysis of primal–dual splitting methods," *Optimization*, vol. 67, no. 6, pp. 821–853, 2018.

[123] C. Poon, J. Liang, and C.-B. Schönlieb, "Local convergence properties of saga/prox-svrg and acceleration," *arXiv preprint arXiv:1802.02554*, 2018.

[124] A. S. Lewis, "Active sets, nonsmoothness, and sensitivity," *SIAM Journal on Optimization*, vol. 13, no. 3, pp. 702–725, 2002.

[125] J. Konečnỳ and P. Richtárik, "Semi-stochastic gradient descent methods," *arXiv preprint arXiv:1312.1666*, 2013.

[126] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[127] S. Oymak, B. Recht, and M. Soltanolkotabi, "Sharp time–data tradeoffs for linear inverse problems," *arXiv preprint arXiv:1507.04793*, 2015.

[128] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[129] M. W. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[130] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[131] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of johnson and lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[132] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[133] E. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[134] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[135] S. Wang, "A practical guide to randomized matrix computations with matlab implementations," *arXiv preprint arXiv:1505.07570*, 2015.

[136] E. Van Den Berg and M. P. Friedlander, "Spgl1: A solver for large-scale sparse reconstruction," 2007.

[137] M. Lichman, "UCI machine learning repository," 2013.

[138] S. Shalev-Shwartz and A. Tewari, “Stochastic methods for l1-regularized loss minimization,” *Journal of Machine Learning Research*, vol. 12, no. Jun, pp. 1865–1892, 2011.

[139] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[140] T. Murata and T. Suzuki, “Doubly accelerated stochastic variance reduced dual averaging method for regularized empirical risk minimization,” in *Advances in Neural Information Processing Systems*, pp. 608–617, 2017.

[141] Y. Arjevani, “Limitations on variance-reduction and acceleration schemes for finite sums optimization,” in *Advances in Neural Information Processing Systems*, pp. 3543–3552, 2017.

[142] V. Roulet and A. d’Aspremont, “Sharpness, restart and acceleration,” in *Advances in Neural Information Processing Systems*, pp. 1119–1129, 2017.

[143] O. Fercoq and Z. Qu, “Restarting the accelerated coordinate descent method with a rough strong convexity estimate,” *arXiv preprint arXiv:1803.05771*, 2018.

[144] J. Wang and L. Xiao, “Exploiting strong convexity from data with primal-dual first-order algorithms,” in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 3694–3702, PMLR, 06–11 Aug 2017.

[145] Z. Allen-Zhu and L. Orecchia, “Linear coupling: An ultimate unification of gradient and mirror descent,” *arXiv preprint arXiv:1407.1537*, 2014.

[146] H. Lin, J. Mairal, and Z. Harchaoui, “A universal catalyst for first-order optimization,” in *Advances in Neural Information Processing Systems*, pp. 3384–3392, 2015.

[147] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč, “Mini-batch semi-stochastic gradient descent in the proximal setting,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2016.

[148] H. Karimi, J. Nutini, and M. Schmidt, “Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 795–811, Springer, 2016.

[149] C. workbench team, “A genomics dataset,” 09 2008.

[150] J. D. Rennie, “Improving multi-class text classification with naive bayes,” 2001.

[151] Z. Lu and L. Xiao, “On the complexity analysis of randomized block-coordinate descent methods,” *Mathematical Programming*, vol. 152, no. 1-2, pp. 615–642, 2015.

[152] 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,” in *Proceedings of the 25th international conference on Machine learning*, pp. 408–415, ACM, 2008.

[153] T. T. Wu, K. Lange, *et al.*, “Coordinate descent algorithms for lasso penalized regression,” *The Annals of Applied Statistics*, vol. 2, no. 1, pp. 224–244, 2008.

[154] Z. Wen, D. Goldfarb, and K. Scheinberg, “Block coordinate descent methods for semidefinite programming,” in *Handbook on Semidefinite, Conic and Polynomial Optimization*, pp. 533–564, Springer, 2012.

[155] Z. Qin, K. Scheinberg, and D. Goldfarb, “Efficient block-coordinate descent algorithms for the group lasso,” *Mathematical Programming Computation*, vol. 5, no. 2, pp. 143–169, 2013.

[156] Y. Nesterov and S. U. Stich, “Efficiency of the accelerated coordinate descent method on structured optimization problems,” *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 110–123, 2017.

[157] H. Lu, R. Freund, and V. Mirrokni, “Accelerating greedy coordinate descent methods,” in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 3257–3266, PMLR, 10–15 Jul 2018.

[158] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke, “Coordinate descent converges faster with the gauss-southwell rule than random selection,” in *International Conference on Machine Learning*, pp. 1632–1641, 2015.

[159] Q. Lin and L. Xiao, “An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization,” in *International Conference on Machine Learning*, pp. 73–81, 2014.

[160] B. Wen, X. Chen, and T. K. Pong, “Linear convergence of proximal gradient algorithm with extrapolation for a class of nonconvex nonsmooth minimization problems,” *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 124–145, 2017.

[161] C. W-team, “Measurement artifacts,” 09 2008.

[162] D. Csiba and P. Richtárik, “Coordinate descent face-off: Primal or dual?,” *arXiv preprint arXiv:1605.08982*, 2016.

[163] S. Shalev-Shwartz, “Sdca without duality, regularization, and individual convexity,” in *International Conference on Machine Learning*, pp. 747–754, 2016.

[164] E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon, “Gap safe screening rules for sparsity enforcing penalties,” *arXiv preprint arXiv:1611.05780*, 2016.

[165] D. Karimi and R. K. Ward, “A hybrid stochastic-deterministic gradient descent algorithm for image reconstruction in cone-beam computed tomography,” *Biomedical Physics & Engineering Express*, vol. 2, no. 1, p. 015008, 2016.

[166] D. Karimi and R. K. Ward, “Sparse-view image reconstruction in cone-beam computed tomography with variance-reduced stochastic gradient descent and locally-adaptive proximal operation,” *Journal of Medical and Biological Engineering*, vol. 37, no. 3, pp. 420–440, 2017.

[167] M. S. Almeida and M. Figueiredo, “Deconvolving images with unknown boundaries using the alternating direction method of multipliers,” *IEEE Transactions on Image processing*, vol. 22, no. 8, pp. 3074–3086, 2013.

[168] M. S. Almeida and M. A. Figueiredo, "Blind image deblurring with unknown boundaries using the alternating direction method of multipliers," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pp. 586–590, IEEE, 2013.

[169] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform deblurring for shaken images," *International journal of computer vision*, vol. 98, no. 2, pp. 168–186, 2012.

[170] Kodak-Lossless-True-Color-Image-Suite, "http://r0k.us/graphics/kodak/."

[171] H. Erdogan and J. A. Fessler, "Ordered subsets algorithms for transmission tomography," *Physics in Medicine & Biology*, vol. 44, no. 11, p. 2835, 1999.

[172] H. Ouyang, N. He, L. Tran, and A. Gray, "Stochastic alternating direction method of multipliers," in *International Conference on Machine Learning*, pp. 80–88, 2013.

[173] W. Zhong and J. Kwok, "Accelerated stochastic gradient method for composite regularization," in *Artificial Intelligence and Statistics*, pp. 1086–1094, 2014.

[174] Z. Allen-Zhu, "Katyusha x: Practical momentum method for stochastic sum-of-nonconvex optimization," *arXiv preprint arXiv:1802.03866*, 2018.

[175] N. Perraudin, V. Kalofolias, D. Shuman, and P. Vandergheynst, "Unlocbox: A matlab convex optimization toolbox for proximal-splitting methods," *arXiv preprint arXiv:1402.0779*, 2014.

[176] S. Basu and Y. Bresler, "O (n/sup 2/log/sub 2/n) filtered backprojection reconstruction algorithm for tomography," *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1760–1773, 2000.

[177] E. Chouzenoux and J.-C. Pesquet, "A stochastic majorize-minimize subspace algorithm for online penalized least squares estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 18, pp. 4770–4783.

[178] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering.," *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, vol. 16, no. 8, pp. 2080–2095, 2007.

[179] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image restoration by sparse 3d transform-domain collaborative filtering," in *Image Processing: Algorithms and Systems VI*, vol. 6812, p. 681207, International Society for Optics and Photonics, 2008.

[180] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017.

[181] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 60–65, IEEE, 2005.

[182] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[183] G. Lan and Y. Zhou, "Conditional gradient sliding for convex optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1379–1409, 2016.

[184] Z. Allen-Zhu, D. Simchi-Levi, and X. Wang, "The lingering of gradients: How to reuse gradients over time," in *Advances in Neural Information Processing Systems*, pp. 1252–1261, 2018.

[185] G. Lan, "Gradient sliding for composite optimization," *Mathematical Programming*, vol. 159, no. 1-2, pp. 201–235, 2016.

[186] G. Lan and Y. Ouyang, "Accelerated gradient sliding for structured convex optimization," *arXiv preprint arXiv:1609.04905*, 2016.